



UNIVERSITÀ DI PAVIA

Dottorato di Ricerca in Fisica — XXXV ciclo

Ph.D. Thesis

Variational quantum algorithms for machine learning

Theory and Applications

Stefano Mangini

Supervisor: Prof. Chiara Macchiavello

Submitted to the Graduate School of Physics in partial fulfillment of the requirements for the degree of Dottore di Ricerca in Fisica (Doctor of Philosophy in Physics) at the

University of Pavia

May 14, 2023

Copyright © 2023 Stefano Mangini

L^AT_EX Template “[The Legrand Oragne Book](#)” v3.1 by Vel and Mathias Legrand.

Quantum circuits drawn with the [quantikz](#) L^AT_EX package.

Thesis background image by kjpargeter, initial covers by Harryarts, on [Freepik](#).
Chapter cover images generated with [Midjourney](#), a Generative AI for images.

A tutta la mia famiglia.

Acknowledgements

I remember when, on January 2019, already a few months after I moved to Pavia to start my Ph.D., I was still in the guest room with Leonardo, the colleague with whom I would eventually have shared the office, and the highs and lows of this journey, hopelessly waiting for the assignment of a proper office that was still lacking. While trying to figure out where we ended up and what it means to “research”, we also got to know each other cheerfully and innocently read the news about a weird health situation in China, without any concern whatsoever that such a situation would become our situation. Or better, everyone’s situation.

Needless to say, Covid-19 essentially cut down in half the enjoyable part —assuming there is one— of the Ph.D., as the first half of it was essentially a sequence of lockdown, boring remote conferencing, and lone investigations. The original idea of spending the Ph.D. travelling the world for conferences and research visits was brutally set aside. But finally, as the health situation recovered and the possibility to travel was restored, I could participate in an amazing (un)conference in Lapland and a Quantum Hiking in Val d’Aosta (I cannot be grateful enough to Sabrina Maniscalco and Lorenzo Maccone for organising such cool and inspiring events), and then also to London for a research period abroad, and eventually to sunny Malta for a summer school.

At the end of the day, I am more than happy with how things ended up. During these years many things happened to the World (even a bit too much) and to myself personally, and, as I repeatedly experience over and over when something ends, what remains are the people we have met and the experiences we have shared. All the rest is just sand castles.

I cannot but start thanking all the people in the Quantum Information Group (QUIT) at the University of Pavia, my academic family for the last few years, for the coffee breaks, lunch (with or without “schiscetta”), and dinners, and for the teachings and anecdotes of what it means to be a true “quittino”, a title I am very proud to carry. I thank Alessandro Bisio and Alessandro Tosini (aka “gli Alessandri”), Max Sacchi, and Paolo Perinotti. I thank Lorenzo Maccone for the fruitful discussions about science, mountains, and all sorts of things, for always being kind, and for organising the Quantum Hiking Conference, one of the most unique experiences I have had. I deeply thank my supervisor Chiara Macchiavello for giving me space and freedom to follow my interests and experiment with research topics and methods, and for always being kind and open to discussions.

I wish to thank Dario Gerace and Daniele Bajoni for the great collaboration and for being always available for interesting discussions, and to Francesco Scala for being such a cool roommate for the QTML2022 conference in Naples, and for the stimulating chats we had on quantum machine learning. A big thank goes to Francesco Tacchino for helping me out at the start of the Ph.D., for the very fruitful and happy collaboration that succeeded, and for the chats and thoughts we shared



Reviewer: "Could you better explain how your method works?"

Me:



PhD students giving advices to each other

"How's research going?"



Figure 1: A careful selection of high quality memes from “High Impact PhD Memes” that were hung on the walls in the office by Matteo Lugli, our *de facto* meme interior designer.

down the road. Although we overlapped in person for just a few months in Pavia, my Ph.D. may have started in a completely different route if it was not for you, thanks!

I am happy to thank all the people with whom I spent many great days and nights: Giovanni “il Giò” Chesi for teaching all of us how to “calm down now”, to Matteo Lugli for providing our office with top-notch memes on Ph.D. life, Davide Rolino for his unbelievable politeness, Simone Roncallo for the few but inspiring discussions, Simanraj for his very entertaining and weird stories, Lorenzo Trezzini for sharing great memories of Trieste, to Francesca Brero and Margherita Porru for offering what is now known as the best coffee in the Department, and eventually to Marco Erba for introducing me to Pavia’s and QUIT’s world and for letting me experience “atypical adventures”.

Also, I am very happy to thank all the fantastic people I have had the honour of meeting in London, where I spend a wonderful four months in the Spring of 2022 for an internship in Quantinum. I must thank Mattia Fiorentini for giving me the opportunity of joining the team and for being such a friendly person, and Marcello Benedetti for what I consider the most stimulating scientific discussions I have had in these years, and especially for being such a cheerful and humane person. Also, a big thanks to all the Quantum Lads: to Chris, Conor, Mateusz, Enrique, and Miguel for all the amazing chats we have had in front of a pint, at Enrique’s farewell party, or while

eating the street food from near the office. To Sam Duffield for being my Englishman of trust and introducing me to the Brits' lifestyle, to David Amaro not only for being my best vegan buddy, but for the amazing discussions on society, and the wonderful day we spent together in Greenwich. To Kirill, my neighbour near Victoria, for all the days we spent together in the office, for introducing me to "Secret Hitler", and overall for being such a nice friend there in London. And finally to Luuk, for all the stories about drunk people in Dublin, for the exciting discussions we had in the office, but ultimately for being such a wonderful person and a close friend. Although it was only for a relatively brief period, the time spent in London was one of the few highs of the Ph.D., and this is especially because of all of you. Moreover, it only rained once in four months and I even saw The Queen's Platinum Jubilee, probably the last one for a long long time. What else could I have wanted?

Finally, I am obliged to thank my amazing friends Claudio and Leonardo, with whom I had the pleasure of sharing the office these years! It was an honour to have you by my side, in what is easily the best office in the Department, thanks to the people inside ("i Dottorandini"), and of course also the classy balcony. The discussions we had in the office on the most disparate topics (quantum, society, mathematics, teaching, music, ...), the pizzas we ate at "La Botticella", and the coffees we drank at the bar: these memories are the most important outcome of the Ph.D., memories that I am truly grateful to bring with me. A very special thank goes to Leonardo, my most faithful companion of these years, with whom I shared everything (even a sofa), and without whom there would be no thesis or Ph.D. to talk about. Despite Covid, breakups, and stress, we were together fighting against the odds. Thank you for being there and for making these three years worth remembering!

There are many other people that I came across and which I would like to thank, but I will do that when I see you in person! For the moment, take a look at the figure above and enjoy a selected choice of memes about Ph.D. life that accompanied me and my colleagues in these years.

At last, the biggest and deepest thank you goes to my Family. In a world where everything changes, you remain the same, and we remain together.

May 14, 2023

List of Publications

These manuscripts are part of the thesis:

1. **Mangini, S.**, Tacchino, F., Gerace, D., Macchiavello, C., & Bajoni, D. (2020). **Quantum computing model of an artificial neuron with continuously valued input data**. *Machine Learning: Science and Technology*, **1**(4), 045008. [195]
2. Tacchino, F., **Mangini, S.**, Barkoutsos, P. K., Macchiavello, C., Gerace, D., Tavernelli, I., & Bajoni, D. (2021). **Variational Learning for Quantum Artificial Neural Networks**. *IEEE Transactions on Quantum Engineering*, **2**, 1-10. [296]
3. **Mangini, S.**, Tacchino, F., Gerace, D., Bajoni, D., & Macchiavello, C. (2021). **Quantum computing models for artificial neural networks**. *Europhysics Letters*, **134**(1), 10002. [192]
4. **Mangini, S.**, Marruzzo, A., Piantanida, M., Gerace, D., Bajoni, D., & Macchiavello, C. (2022). **Quantum neural network autoencoder and classifier applied to an industrial case study**. *Quantum Machine Intelligence*, **4**(2), 13. [196]
5. **Mangini, S.**, Maccone, L., & Macchiavello, C. (2022). **Qubit noise deconvolution**. *EPJ Quantum Technology*, **9**(1), 1-30. [194]
6. Ballarin, M., **Mangini, S.**, Montangero, S., Macchiavello, C., & Mengoni, R. (2022). **Entanglement entropy production in Quantum Neural Networks**. *arXiv preprint arXiv:2206.02474*. Accepted in *Quantum*. [18]
7. **Mangini, S.**, Benedetti, M. *Manuscript in preparation*. [193]

These manuscripts are not part of the thesis:

8. Skolik, A., **Mangini, S.**, Bäck, T., Macchiavello, C., & Dunjko, V. (2023). **Robustness of quantum reinforcement learning under hardware errors**. *EPJ Quantum Technology*, **10**(1), 1-43. [285]
9. Benatti, F., Mancini, S., & **Mangini, S.** (2019). **Continuous variable quantum perceptron**. *International Journal of Quantum Information*, **17**(08), 1941009. [24]
10. Scala, F., **Mangini, S.**, Macchiavello, C., Bajoni, D., & Gerace, D. (2022). **Quantum variational learning for entanglement witnessing**. In *2022 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8)*. IEEE. [259]
11. Di Sipio, R., Huang, J. H., Chen, S. Y. C., **Mangini, S.**, & Worring, M. (2022). **The dawn of quantum natural language processing**. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 8612-8616)*. IEEE. [84]

Chapter 2 and Chapter 3 are loosely based on manuscripts 3 and 7 with substantial additions; Chapter 4 is based on manuscript 1 [195]; Chapter 5 is based on manuscript 2 [296]; Chapter 6

is based on manuscript 4 [196]; Chapter 7 is based on manuscript 6 [18]; Chapter 8 is based on manuscript 5 [194].

Summary

The topic of the present Ph.D. thesis is Quantum Computation and Information processing, with a specific focus on the relatively new fields of Variational Quantum Algorithms and Quantum Machine Learning. These disciplines not only offer a unique perspective on studying quantum information processing tasks, but also have the potential to provide a useful computational quantum advantage even with the currently available first-generation small and noisy quantum computing devices.

Variational Quantum Algorithms involve a hybrid quantum-classical computational loop, where a quantum computer is used only for some specific ideally quantum-native subroutines, and a classical computer runs an optimisation procedure on variational parameters to minimise a cost function whose minimum corresponds to the solution to the problem to be solved. This framework shares the same foundational idea as state-of-the-art Deep Learning models, where complex parametric models are tuned via optimisation methods to solve various tasks. The intersection of quantum computing and machine learning has led to the development of quantum machine learning, an interdisciplinary area that explores the benefits of combining quantum computation and artificial intelligence.

This thesis provides a comprehensive analysis of the state of the art of the field, including numerous original contributions, from the study of quantum models for artificial neurons to the characterisation of entanglement created in quantum architectures for neural networks, up to discussing the effect of measurement noise on a more quantum information perspective.

The first chapters are devoted to a careful review of the basics of quantum computing and a thorough discussion of variational quantum algorithms. Then the discussion is moved to quantum machine learning, where an introduction to the elements of machine learning and statistical learning theory is followed by a review of the most common quantum counterparts of machine learning models.

Afterward, multiple novel contributions to the field are presented. A newly introduced model for a quantum perceptron is discussed, along with applications to pattern recognition and classification tasks. Such a model is then generalised to include strategies based on variational protocols to reduce the circuit footprint of the proposed architecture, and also analyse its performances where multiple optimisation strategies are considered. Subsequently, a quantum algorithm comprising a quantum autoencoder followed by a quantum classifier is presented to first compress and then label classical data coming from an industrial power plant, thus providing one of the first attempts to integrate quantum computing procedures in a real-case scenario of an industrial pipeline.

The analysis is then broadened to a more quantum information perspective, by first studying the entanglement features of quantum neural networks. Specifically, tensor networks are employed

to study the entanglement entropy in parameterized quantum circuits of up to fifty qubits and show that the entanglement generated in such architectures reaches that of typical random quantum states under various measures. Finally, the focus is shifted from quantum machine learning to that of quantum noise, and a noise deconvolution technique is presented to remove a wide class of noises when performing arbitrary measurements on qubit systems.

The thesis then ends with conclusions where loose ends are discussed, and final remarks are exposed. Overall, the thesis provides a well-balanced investigation of multiple scientific domains, including quantum physics and computer science, through theoretical exploration, computational simulations, and experimental verification on already available quantum computing devices.

Contents

Acknowledgements	3
List of Publications	6
Summary	8
1 Introduction	19
2 Quantum Computing and Variational Quantum Algorithms	22
2.1 Basics of Quantum Computation	23
2.1.1 Single qubit systems and operations	23
2.1.2 Multi qubits systems and two-qubits operations	25
2.1.3 Density matrix formalism	28
2.1.4 Measurements and expectation values	30
2.1.5 The quantum circuit model	33
2.1.6 The NISQ era of quantum computation	34
2.2 Variational Quantum Algorithms	36
2.2.1 The basis of variational quantum algorithms	36
2.2.2 Parameterised quantum ansätze	37
2.2.3 Optimisation of variational quantum algorithms	39
2.2.4 Barren plateaus and unitary designs	43
2.2.5 Expressibility of PQCs	50
2.3 Conclusions	51
3 Quantum Machine Learning	52
3.1 Introduction	53
3.1.1 The four-fold way of Quantum Machine Learning	53
3.2 Classical Machine Learning	55
3.2.1 Basics of (supervised) Machine Learning	57
3.2.2 Machine learning models	62

3.3	Quantum Machine Learning	67
3.3.1	Linear quantum models: quantum classifiers and kernel methods	68
3.3.2	Data reuploading models and Quantum Neural Networks	71
3.3.3	Generalization of QML models	77
3.3.4	The power of quantum machine learning	79
3.4	Conclusions	80
4	Quantum computing model of an artificial continuous neuron ..	81
4.1	Introduction	81
4.2	Continuously valued quantum neuron model	82
4.2.1	Some properties: colour invariance and noise resilience	84
4.2.2	Quantum circuit model of a continuously valued perceptron	85
4.3	Results	87
4.3.1	Testing the quantum neuron for image recognition tasks	88
4.4	Training the quantum neuron	88
4.4.1	Classification tasks	90
4.4.2	MNIST dataset	92
4.5	Conclusions	93
5	Variational learning for quantum neural networks	94
5.1	Introduction	94
5.2	A model of quantum artificial neurons	95
5.2.1	Exact implementation with quantum hypergraph states	97
5.3	Variational realisation of a quantum artificial neuron	97
5.3.1	Global variational training	98
5.3.2	Local variational training	99
5.3.3	Case study: pattern recognition	100
5.3.4	Structure of the ansatz and scaling properties	101
5.4	Conclusions	106
6	Quantum autoencoder and classifier for an industrial use case .	108
6.1	Introduction	108
6.2	Case study	109
6.3	Neural network autoencoder	110
6.3.1	Classical Autoencoders	112
6.4	Quantum Data Compression	113
6.4.1	Quantum Autoencoder	113
6.5	Experiments and Results	115
6.5.1	Data compression	115
6.5.2	Classification	119
6.6	Conclusions	121
7	Entanglement entropy production in quantum neural networks .	123
7.1	Introduction	124
7.2	Methods	125
7.2.1	Tensor Networks and Matrix Product States	125

7.2.2	Entanglement measure in Matrix Product States	126
7.2.3	Entanglement entropy in random quantum states	127
7.2.4	Quantum Neural Networks as Parameterised Quantum Circuits	128
7.2.5	Randomness, Entanglement and Trainability	129
7.3	Results	130
7.3.1	Alternating vs. Sequential data reuploading	130
7.3.2	Entanglement distribution across bonds	131
7.3.3	Entanglement scaling with increasing depth	134
7.3.4	Entanglement Speed	135
7.3.5	Expressibility	138
7.3.6	Distribution of the singular values	138
7.4	Discussion	139
7.5	Conclusion	141
8	Noise deconvolution	142
8.1	Introduction	142
8.2	Methods	144
8.2.1	Quantum channels	145
8.2.2	Qubit systems and Pauli Transfer Matrix formalism	145
8.2.3	Quantum tomographic reconstruction	146
8.3	Noise Deconvolution	147
8.4	Inversion of common noise maps	149
8.5	Experimental deconvolution	154
8.5.1	Decoherence noise model	154
8.5.2	Arbitrary Pauli channel	157
8.6	Conclusions	157
9	Conclusions	159

References

Bibliography	164
---------------------	------------

Appendices

A	Variational Quantum Algorithms	190
A.1	Global and local cost functions	190
A.2	Variance of gradients	191
B	Quantum Machine Learning	194
B.1	Generalisation bound for data-reuploading quantum neural networks	194
B.1.1	Rademacher complexity and generalisation error	194
B.1.2	Rademacher complexity of Linear Classes	195
B.1.3	Generalisation bound of Quantum Neural Networks	196

C	Continuous Quantum Neuron	199
C.1	Proof of the activation function of the quantum neuron	199
C.2	Noise resilience	199
C.3	Alternative schemes for the data encoding operations	200
D	Entanglement of Quantum Neural Networks	202
D.1	Lower bound on entanglement entropy for unitary 2-designs	202
D.2	Details on Haar entanglement	203
D.3	Triviality of the full entangling map	204
D.4	Expressibility of Parameterised Quantum Circuits	205
D.5	Entanglement scaling with increasing depth	205
D.6	Convergence of MPS simulations	206
D.7	Entanglement evolution during training	207
D.7.1	Details on the classification procedure	210
E	Noise Deconvolution	213
E.1	Kraus Decomposition	213
E.2	Tomographic reconstruction formula for qubits	213
E.3	Noise deconvolution for qubits	214
E.4	Inverse maps of Noise channels	214
E.4.1	Bit-flip, phase-flip, and bit-phase-flip channels	214
E.4.2	Depolarizing channel	216
E.4.3	General Pauli channel	217
E.4.4	Amplitude Damping	218
E.4.5	2-Kraus channel	219

List of Figures

1	A selection of very high-impact memes	5
2.1	Bloch sphere representation of a qubit	24
2.2	Examples of currently available quantum computers	34
2.3	Schematic representation of variational quantum algorithms	38
2.4	Sources of Barren Plateaus	44
3.1	Variants of Quantum Machine Learning applications	54
3.2	Generalisation and overfitting in machine learning	61
3.3	A feedforward neural network	66
3.4	Linear quantum models	68
3.5	Parameterised quantum circuit as a Fourier series	73
3.6	Quantum Neural Network	76
4.1	Scheme of a classical perceptron model	83
4.2	Quantum circuit for the quantum perceptron model.	85
4.3	Circuitual implementation of the continuous quantum neuron.	87
4.4	A grey-scale image with pixels intensities.	88
4.5	Image recognition task performed by the artificial quantum neuron	89
4.6	Training the quantum neuron	90
4.7	Classification of two dimensional data	91
4.8	Classification of two dimensional circles	91
4.9	Application of the quantum neuron on the MNIST dataset	92
5.1	Variational learning via unsampling	98
5.2	Comparison of exact to approximate implementations	100
5.3	Optimisation of the global unitary with nearest neighbours entanglement	102
5.4	Final fidelity for different structures and number of qubits	103
5.5	Number of iterations to reach target fidelity	104
5.6	Optimisation in presence of stochastic measurement outcomes	105
5.7	Scaling of circuit depth	106
6.1	Snapshot of a separator	109
6.2	Summary of the approach followed in the work	111
6.3	Details on the procedure	112

6.4	Schematic representation of a quantum autoencoder	114
6.5	Quantum circuit for the quantum autoencoder	116
6.6	Optimisation of the quantum encoder	117
6.7	Performances of the quantum autoencoder	118
6.8	Circuit to evaluate the fidelity	119
6.9	Results of the classification task	120
7.1	Graphical representation of QNN and MPS	124
7.2	Normalised entanglement for various numbers of qubits	132
7.3	Average entanglement entropy across bonds	133
7.4	Number of layers to reach a target Haar-randomness	134
7.5	Equivalence of full and linear entangling topologies	135
7.6	Normalised entanglement versus normalised number of layers	136
7.7	Normalised entanglement for real-world datasets	137
7.8	Expressibility of the quantum neural networks	138
7.9	Convergence to the Marčenko-Pastur (MP) distribution	139
8.1	Summary of the noise deconvolution process	143
8.2	Deconvolution of decoherence noise on real hardware	156
8.3	Deconvolution of Pauli channels	158
C.1	Variational approach for the quantum neuron	201
D.1	Equivalence of full and linear entangling maps	205
D.2	Total entanglement for various quantum neural networks	206
D.3	Approximation errors of MPS	207
D.4	Evolution of entanglement entropy during training	208
D.5	Preprocessing of the IRIS dataset	211
D.6	Mean entanglement entropy per IRIS class	212

List of Tables

2.1	Summary table of two-qubits gates	26
2.2	Summary table of two-qubits gates	28
2.3	Examples of native gates on real quantum hardware	33
6.1	Key figures of compression and classification with a quantum autoencoder	110
7.1	Number of parameters in considered ansätze	131
7.2	Entangling speed	136
8.1	Noise channels and their inverse maps	149

List of Abbreviations and Symbols

This list list summarise the abbreviations and symbols that are used throughout the work.

Abbreviations

BP	Barren Plateau
CPTP	Completely Positive Trace Preserving
ML	Machine Learning
PQC	Parameterised Quantum Circuit
PTM	Pauli Transfer Matrix
QML	Quantum Machine Learning
VQA	Variational Quantum Algorithm

Mathematical symbols

θ	Trainable parameters of a parameterised quantum circuit in a variational quantum algorithm
\mathbf{w}	Trainable parameters of a parametric classical machine learning model
\mathbf{x}	Input sample belonging to input data space, $\mathbf{x} \in \mathcal{X}$
\mathcal{H}	Hilbert space
\mathcal{X}	Input data space, usually $\mathcal{X} \subset \mathbb{R}^d$
\mathcal{Y}	Output data space, usually $\mathcal{Y} \subset \mathbb{R}$
\mathcal{Z}	Data space given by pairs inputs and outputs, $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$
d	Dimension of the input vectors, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$
m	Number of data samples in the training set $\mathcal{S} \subset \mathcal{Z}^m$
p	Dimension of the trainable parameter vectors, $\mathbf{w}, \theta \in \mathbb{R}^d$
\mathcal{S}	Training set consisting of pairs of inputs and outputs
y	Output sample belonging to output data space $y \in \mathcal{Y}$



1. Introduction

Nature isn't classical, dammit, and if you want to make a simulation of Nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem because it doesn't look so easy.

Richard Feynman, 1982 [102]

As far as I can tell from my still brief experience of research into Quantum Computation during these years of Ph.D., it is not possible to start a discussion on this topic without mentioning the (moral) father of such a discipline, the mighty Richard Feynman who, in 1982, roughly at the same time of other scientists of those years [237], firstly proposed the idea of building a quantum-mechanical computing device for simulating Nature. Thus, since I don't feel in the position to interrupt this grand tradition, please take a few moments to enjoy for the umpteenth time Feynman's quote on quantum computing, which you can find at the start of the page.

With a leap forward 40 years into the future, quantum computing is starting to become a solid reality, with the first small-scale prototypes of quantum computers being actively developed and tested, and with a universal fault-tolerant quantum computing device hopefully to appear within a few decades in the future, although the road to this goal is strewn with major technical, experimental, and theoretical challenges. The current generation of quantum computers has been dubbed NISQ, for Noisy Intermediate-Scale Quantum [236], which indicates devices consisting of small quantum processing units consisting of just tens to few hundreds of carriers of quantum information, the *qubits*, and that these qubits are imperfect, subject to noise that diminishes their quantum properties, hence their computational relevance.

With the impossibility of timely orchestrating those elements that make quantum computers unique and powerful objects, namely *superposition*, *entanglement*, and *interference*, many of the celebrated quantum algorithms that were proposed in the previous decades with provable computational speedups, Shor's being the most convincing example [220], currently remain well out of reach. However, the imperfect nature of current quantum devices prompted the creation of seemingly imperfect quantum algorithms that trade off provable guarantees of quantum speedups with reasonable intuitions of classical hardness and the careful human craftsmanship of algorithms with the heavy lifting provided by powerful classical optimisation techniques. However, as we shall see in a few paragraphs, this situation does not only concern quantum computing but rather the field of applied computer science as a whole.

These relatively new types of quantum procedures are called Variational Quantum Algorithms

and are specifically thought to be executable already on available computing devices, and thus take full advantage of the current generation of quantum computing. The variational paradigm prescribes the use of a hybrid quantum-classical computational loop in which a quantum computer is used in tandem with a classical computer, where the former is used only for some specific ideally quantum-native subroutines, and the latter instead to run an optimisation procedure on some variational parameters to minimise a cost function whose minimum (maximum) corresponds to the solution to the problem to be solved. This framework for NISQ-friendly quantum algorithms gained much momentum over the last few years, and it is considered one of the best candidates to achieve some form of useful computational quantum advantage, even though the debate on this expectation is far from being settled.

However, it is very important to recall that quantum computation and quantum information science first remain scientific disciplines that deserve thorough and enthusiastic scientific research on their own, and regardless of possible technological applications, with quantum computers to be considered, at least at this moment, mainly scientific tools. In this respect, borrowing the words from Simone Severini: *“The paradigm with which quantum computers work is totally different from the one with which classical computers work. Therefore, it is too simplistic to make comparisons in terms of speed and efficiency. The most sensible analogy is not with a classical computer, but rather with a telescope: the quantum computer should be regarded as an instrument that allows one to look further”* [147].

Even more than Quantum Computing and Quantum Technologies in general, over the past decade, Artificial Intelligence and Deep Learning have garnered increasing scientific and technological attention. These fields offer a variety of tools that can handle a diverse range of tasks, from achieving superhuman performance on board games [279] to controlling nuclear reactors [83], and even having astonishingly human-like conversational abilities [60]. State-of-the-art Deep Learning models operate by optimising large, complex, and often opaque parametric models to minimise a problem-dependent loss function. Although a complete theoretical understanding of these models’ inner workings is still lacking and the subject of active research, their widespread success is undoubtedly motivated by their empirical success in practical applications.

In recent years, the fields of quantum computing and machine learning joined forces and stimulated the development of Variational Quantum Algorithms, which, as we mentioned above, rely on optimising complex parametric models (parametric quantum circuits instead of artificial neural networks) using optimisation methods like gradient descent to minimise a problem-specific cost function. This union was so fruitful that it was given the name Quantum Machine Learning, an interdisciplinary area that explores the interplay and benefits of combining quantum computation and artificial intelligence [88]. In fact, the overlap between these two fields predates Variational Quantum Algorithms, as linear algebra-based quantum subroutines have been proposed as accelerators in classical machine learning algorithms for some time already [31], and it is possible to find resources about quantum computation and neural networks even from decades ago [120, 182].

The topic of this thesis lies in the most recent incarnation of quantum machine learning, namely the use of variational quantum circuits as machine learning models. It aims to provide a comprehensive analysis of the state of the art of the field, as well as the discussion of numerous original contributions, from the study of quantum models for artificial neurons to the characterisation of entanglement created in common quantum models for neural networks, up to discussing the effect of measurement noise on a more quantum information perspective. As we shall see in the following chapters, this crossover proves very valuable and most importantly interesting both on a theoretical and practical level. Indeed, studies on these topics are stimulating as they require specialised knowledge from multiple disciplines from quantum physics and computer science, and permit a very well-balanced investigation between theoretical exploration, computational simulations, and also experimental verification on already available quantum computing devices.

The rest of the thesis is organised as follows. In Chapter 2 we start reviewing the basics and

introducing the notation of quantum computing, to then move towards a thorough discussion on variational quantum algorithms. Chapter 3 is instead dedicated exclusively to quantum machine learning, where an introduction to the elements of machine learning and statistical learning theory is followed by a review of the most common quantum counterparts of machine learning models.

In Chapter 4 we discuss a newly introduced model for a quantum perceptron, which is a quantum algorithm mimicking the behaviour of a classical artificial neuron, and show how it can be used to implement pattern recognition and classification tasks. This model is then generalised in Chapter 5, where various strategies based on variational protocols are described to reduce the circuit footprint of the quantum neuron model. Afterwards, in Chapter 6 we propose a quantum algorithm comprising a quantum autoencoder followed by a quantum classifier to first compress and then label classical data coming from an industrial power plant, thus providing one of the first attempts to integrate quantum computing procedures in a real-case scenario of an industrial pipeline.

In Chapter 7 we expand the perspective and analyse the entanglement features of quantum neural networks. Specifically, we use tensor network tools to study the entanglement entropy in parameterized quantum circuits of up to fifty qubits and show that the entanglement generated in such architectures reaches that of typical random quantum states under various measures. Finally, in Chapter 8 we step out from quantum computation and machine learning and rather focus on the topic of noise from a quantum information viewpoint and present a noise deconvolution technique to remove a wide class of noises when performing arbitrary measurements on qubit systems.

Finally, in Chapter 9 we try to draw some inspiring conclusions about this journey on variational algorithms, noisy quantum computers, and machine learning. In the appendices, some calculations or additional details regarding the related topics discussed in the main text are reported.

Without further ado, let us start!



2. Quantum Computing and Variational Quantum Algorithms

When life gives you lemons, make lemonade!

Proverbial phrase, and often Scott Aaronson.

2.1	Basics of Quantum Computation	23
2.1.1	Single qubit systems and operations	23
2.1.2	Multi qubits systems and two-qubits operations	25
2.1.3	Density matrix formalism	28
2.1.4	Measurements and expectation values	30
2.1.5	The quantum circuit model	33
2.1.5.1	Executing a quantum circuit on a real quantum device	33
2.1.6	The NISQ era of quantum computation	34
2.2	Variational Quantum Algorithms	36
2.2.1	The basis of variational quantum algorithms	36
2.2.2	Parameterised quantum ansätze	37
2.2.3	Optimisation of variational quantum algorithms	39
2.2.3.1	Parameter shift-rule	40
2.2.3.2	Higher order derivatives	42
2.2.3.3	Discussion	43
2.2.4	Barren plateaus and unitary designs	43
2.2.4.1	Haar measure and random unitary matrices	46
2.2.4.2	Barren plateaus in the optimisation of PQCs	47
2.2.4.3	Discussion and mitigation of barren plateaus	49
2.2.5	Expressibility of PQCs	50
2.3	Conclusions	51

In this chapter, we discuss Quantum Computing and Variational Quantum Algorithms. We start by introducing the necessary tools and definitions of quantum computation and then move on to discuss recent results on variational quantum algorithms, which are a class of algorithms specifically suited for currently available near-term quantum devices. Extended discussions of such topics can be found in [29, 58, 59, 192, 301].

2.1 Basics of Quantum Computation

Quantum Computation and Quantum Information are fields of research that study how physical quantum systems can be used to process information and perform computations. In analogy with the classical setting, the basic element of quantum information is called quantum bit or *qubit*¹, an object describing the behaviour of a two-level quantum system, for example a particle having access to the ground state and first excited state of an energy potential. Independently of its actual physical realisation, the quantum *state* of a qubit is mathematically described as a vector living in a two-dimensional vector space over the complex numbers, namely \mathbb{C}^2 .

2.1.1 Single qubit systems and operations

Let $\mathcal{H} = \mathbb{C}^2$ denote the *Hilbert* space of the qubit, that is the vector space \mathbb{C}^2 equipped with the standard inner product given by the Euclidean dot product for vectors with complex entries. Dirac's bra-ket notation of quantum mechanics prescribes the use of the so-called *kets* $|\cdot\rangle$ to indicate vectors (i.e. quantum states) in this space, namely $|\psi\rangle \in \mathcal{H}$, where ψ indicates a state. Similarly, one uses *bras* $\langle\cdot|$ to indicate the conjugate transpose of a vector

$$\langle\psi| = (|\psi\rangle)^\dagger, \quad |\psi\rangle \in \mathcal{H},$$

where the dagger operation $A^\dagger = (A^*)^T$ is the composition of complex conjugation A^* and transposition A^T . With this notation, the inner product between two states is denoted with the juxtaposition of a bra with a ket, as follows

$$\langle\cdot|\cdot\rangle : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{C}, \quad |\psi\rangle, |\phi\rangle \in \mathcal{H}, \quad \langle\phi|\psi\rangle \in \mathbb{C}.$$

Let $|0\rangle$ and $|1\rangle$ denote an orthonormal basis of \mathcal{H} , and call it the *computational basis* of the space. Then, any (pure) state can be expressed as a linear combination of the computational basis states with complex coefficients

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \alpha, \beta \in \mathbb{C}. \quad (2.1)$$

with a normalisation condition $\langle\psi|\psi\rangle = 1$ which constraints the complex coefficients to satisfy $\alpha\alpha^* + \beta\beta^* = |\alpha|^2 + |\beta|^2 = 1^2$. Also, states that differ only for a global phase factor, like $|\psi\rangle$ and $e^{i\delta}|\psi\rangle$, represent the same *physical* state because overall phases are not observable, in that measurements involving phase-shifted states will yield the same results. Thus, out of the initial four real parameters, a (pure) state of a qubit can be written in terms of just two parameters (θ, φ) as

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle. \quad (2.2)$$

This equation is called *Bloch sphere* representation of the qubit, because it makes it evident that the state of a qubit can be visualised as a point on the unit sphere with coordinates (θ, φ) , where the poles are the orthogonal basis states $|0\rangle$ and $|1\rangle$, see Fig. 2.1.

It is often useful to reason in terms of vector components instead of states. Representing the basis states $|0\rangle$ and $|1\rangle$ as column vectors, Eq. (2.1) can be rewritten as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad \text{with} \quad |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \quad (2.3)$$

¹A term which is surprisingly recent, just as young as me at the time of writing, introduced for the first time in 1995 by Benjamin Schumacher [271].

²Normalisation of states is a useful requirement that makes Born's rule for probabilities of measurement outcomes easier to compute.

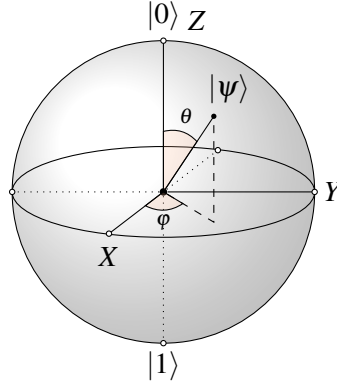


Figure 2.1: Bloch sphere representation of a qubit. A pure state of a qubit can be represented as a point on a unit sphere whose poles are the orthogonal basis states $|0\rangle$ and $|1\rangle$. The angles (θ, φ) are defined in Eq. (2.2). The six points arising from the intersection of the unit sphere with the three orthogonal axes are the eigenstates of the Pauli matrices X , Y , and Z , defined in Eq. (2.4)

Single qubit operations Operations on qubits are linear transformations $U : \mathcal{H} \rightarrow \mathcal{H}$ mapping quantum states to quantum states, and these can be represented by unitary matrices that map unit complex vectors to other unit complex vectors preserving the norm. The unitary evolution of (closed) quantum systems implies that not only the norm of quantum states is preserved, but also the *reversibility* of quantum computation, since for any operation U also the inverse one U^\dagger with $UU^\dagger = \mathbb{I}$ is a valid quantum operation. This is different from classical computation, where operations on bits can be, in general, irreversible.

A set of operators of particular relevance for describing qubit systems are the Pauli matrices $\{X, Y, Z\}$, which in the computational basis $\{|0\rangle, |1\rangle\}$ have matrix representation³

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \quad (2.4)$$

In hindsight, the computational basis is actually *defined* to consist of the eigenstates of the Z operator, which acts on such basis as $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. The X operator is also called the “NOT” operation, again in accordance with the classical computation terminology, as its action is to flip the state of the qubit $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. Together with the identity \mathbb{I} , these matrices form a basis of the space of 2×2 complex matrices. Other useful properties are that the Pauli matrices are unitary ($AA^\dagger = \mathbb{I}$), Hermitian ($A = A^\dagger$), involutory ($A^2 = \mathbb{I}$), traceless ($\text{Tr}[A] = 0$), and the following commutation relations hold $[\sigma_i, \sigma_j] = 2i\varepsilon_{ijk}\sigma_k$, where $\sigma_i, \sigma_j \in \{X, Y, Z\}$, ε_{ijk} is the Levi-Civita tensor, and summation is implied over repeated indices. The Pauli matrices have eigenvalues $\lambda \in \{\pm 1\}$, with corresponding eigenstates

$$\begin{aligned} Z|0\rangle &= +|0\rangle & X|+\rangle &= +|+\rangle & Y|+i\rangle &= +|+i\rangle \\ Z|1\rangle &= -|1\rangle & X|-\rangle &= -|-\rangle & Y|-i\rangle &= -| -i\rangle \\ \text{with } |\pm\rangle &:= \frac{|0\rangle \pm |1\rangle}{\sqrt{2}} & |\pm i\rangle &:= \frac{|0\rangle \pm i|1\rangle}{\sqrt{2}} \end{aligned} \quad (2.5)$$

These states are highlighted in the Bloch sphere of Fig. (2.1), and are located at the intersection of the unit sphere with the three orthogonal axes, which then represent the “directions” of the Pauli matrices.

³Operators and their matrix representation are not the same things, as one can define and use operators without invoking their matrix representation, which is specific to the chosen basis. In quantum computing however, the basis is always considered to be the *computational basis*, and, with a slight abuse of notation, in the following we use the same symbol to denote both the operator and its matrix representation.

In the quantum computing jargon, operations on qubits are called *gates*, in analogy with the terminology used in classical computation to indicate elemental logical operations on bits. A major role in variational quantum algorithms is played by Pauli rotations gates, which are parametrized operations defined via exponentiation of the Pauli as follows

$$R_X(\theta) := e^{-iX\theta/2} \equiv \cos \frac{\theta}{2} \mathbb{I} - i \sin \frac{\theta}{2} X = \begin{bmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix} \quad (2.6)$$

$$R_Y(\theta) := e^{-iY\theta/2} \equiv \cos \frac{\theta}{2} \mathbb{I} - i \sin \frac{\theta}{2} Y = \begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix} \quad (2.7)$$

$$R_Z(\theta) := e^{-iZ\theta/2} \equiv \cos \frac{\theta}{2} \mathbb{I} - i \sin \frac{\theta}{2} Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix} \quad (2.8)$$

where the passage from the exponential to the trigonometric formula is easily obtained via the definition of the exponential function and the involutory property ($A^2 = \mathbb{I}$) of Pauli matrices

$$e^{-i\omega A} := \sum_{k=0}^{\infty} \frac{(-i\omega A)^k}{k!} = \sum_{k \in \text{even}} \frac{\omega^k}{k!} \mathbb{I} - \sum_{k \in \text{odd}} \frac{\omega^k}{k!} A = \cos \omega \mathbb{I} - \sin \omega A. \quad (2.9)$$

These gates are called rotation operations because they act on a qubit rotating it around the respective Pauli axis in the Bloch sphere representation by an amount indicated by the angle parameter. Also, these operations are ubiquitous in Variational Quantum Algorithms because they are the most straightforward way to introduce free parameters in a quantum computation, and are also usually easy to implement on real quantum hardware.

In addition to the aforementioned Pauli and Pauli rotation gates, another fundamental operation is the Hadamard gate, denoted by H , which is used to create *superposition* states as it maps the ground state $|0\rangle$ to the *superposition* state $|+\rangle = H|0\rangle$. In Table 2.1 we report the most common single qubit gates, along with their circuitual representation when depicted as gates in a quantum circuit and their matrix representation in the computational basis. All other single-qubit gates can be expressed in terms of these operations. Indeed, the most general single-qubit operation, that is a general 2×2 unitary matrix, can be written (up to global phase) as

$$U(\theta, \varphi, \lambda) = \begin{bmatrix} \cos \frac{\theta}{2} & -e^{i\lambda} \sin \frac{\theta}{2} \\ e^{i\varphi} \sin \frac{\theta}{2} & e^{i(\varphi+\lambda)} \cos \frac{\theta}{2} \end{bmatrix}, \quad (2.10)$$

and there exists some angles such that it can be decomposed with a sequence of rotations $U(\theta, \varphi, \lambda) = e^{i\alpha} R_Z(\beta) R_Y(\gamma) R_Z(\delta)$ [220]. Note that other equivalent decompositions into elementary rotations are possible.

2.1.2 Multi qubits systems and two-qubits operations

Usually, more qubits together are used to implement a quantum information processing task or a computation. The Hilbert space for a system composed of multiple qubits is built considering the *tensor product* of the single-qubit Hilbert spaces. For example, a two-qubit system lives in the Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 = (\mathbb{C}^2)^{\otimes 2} = \mathbb{C}^4$, and its state can be expressed as a linear combination of the four computational basis states

$$|\psi\rangle = \alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle, \quad \alpha, \beta, \gamma, \delta \in \mathbb{C}, |\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1 \quad (2.11)$$

where the basis states $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ arise from considering tensor products of the single qubits basis states, namely

$$|00\rangle := |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 0 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \quad (2.12)$$

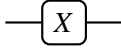
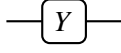
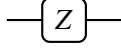
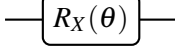
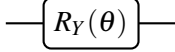
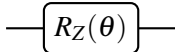
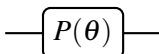
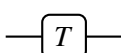
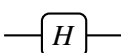
Name	Symbol/Circuitual rep.	Matrix representation
Pauli-X		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Pauli rotation-X		$\begin{bmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix}$
Pauli rotation-Y		$\begin{bmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{bmatrix}$
Pauli rotation-Z		$\begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$
Phase gate		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$
T gate		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Hadamard		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Table 2.1: Summary of the most important single-qubit operations. Here are shown the name of the operations, the abbreviations, the circuitual representations when implemented as a *gates* in a quantum circuit, and the matrix representations in the computational basis. Note that the phase gate P and R_Z are related by just a global phase, but it is useful to keep them separately.

In the same way, an n -qubit (pure) state is a vector in $\mathcal{H} = (\mathbb{C}^2)^{\otimes n} = \mathbb{C}^{2^n}$ and it can be expressed in full generality as a normalized superposition of the 2^n basis states

$$|\psi\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle, \quad c_i \in \mathbb{C}, \quad \sum_{i=0}^{2^n-1} |c_i|^2 = 1 \quad (2.13)$$

where the state $|i\rangle$ is a shorthand to denote multi-qubit computational basis states, where i is the decimal representation of the binary string (or *bit-string*) of zeros and ones composing the basis⁴.

Two-qubit operations As for single qubits, an operation on a n -qubit state can be represented by a unitary matrix in $(\mathbb{C}^2)^n \times (\mathbb{C}^2)^n$, and any such matrix is a valid multi-qubit quantum gate. However, instead of considering general unitaries, one usually constructs multi-qubit gates starting from single- and two-qubit ones, since these form a so-called *universal* set of gates. We elaborate more on this concept at the end of the section and now proceed to describe common two-level gates.

First, two independent single-qubit operations acting on two different qubits are cast in the form of a single two-qubit gate via the tensor product operation. For example, the action of two

⁴For example, the two-qubit state $|11\rangle$ is denoted as $|11\rangle \rightarrow |3\rangle$, as 11_2 in basis 2 is 3_{10} in basis 10 (the subscript denotes the basis).

Pauli-Z gates in parallel on a system of two qubits is described by the operator

$$Z \otimes Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & 0 \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ 0 \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} & -1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.14)$$

At the basis of every quantum computation are the two-qubit controlled operations that are used to create *entangled* states. These operations act on the state of a *target* qubit conditionally on the state of another qubit, called *control*. The prototypical gate in this class is the controlled-NOT operation (CNOT or CX), which does nothing —i.e., acts with the identity— if the control qubit is in the state $|0\rangle$, and acts with the X gate on the target if the control is in the $|1\rangle$ state instead. Its definition and matrix representation is

$$\text{CNOT} := |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes X = \begin{bmatrix} 1 & & & \\ & \overbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}^{\langle 0|} & & \\ & & & \\ & & & \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{\langle 1|} \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & & & \\ & \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{\langle 1|} & & \\ & & & \\ & & & \overbrace{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}}^{\langle 1|} \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (2.15)$$

$$= \begin{bmatrix} 1 & 0 & & \\ & 1 & 0 & \\ & & 0 & 1 \\ & & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & & \\ & 0 & 0 & \\ & & 0 & 1 \\ & & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (2.16)$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (2.17)$$

Similarly, the controlled-Z (CZ) operation does nothing if the control qubit is $|0\rangle$ and applies Z to the target qubit otherwise, it is defined as $\text{CZ} := |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes Z$. It can be checked that the CZ has the same action if the control and target are exchanged, since its overall effect is to add a minus sign to $\text{CZ}|11\rangle = -|11\rangle$, while leaving the remaining three states in the computational basis unchanged. Indeed, its graphical representation, shown in Fig. 2.2, is symmetrical and does not distinguish between a target and a control. In general, one can define a controlled version of any single qubit operation U , via $\text{CU} := |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes U$.

One last very important operation is the SWAP gate, whose action is to exchange the state of two qubits $\text{SWAP}|\psi\rangle \otimes |\phi\rangle = |\phi\rangle \otimes |\psi\rangle, \forall |\psi\rangle, |\phi\rangle \in \mathbb{C}^2$. It can be checked that this operation can be implemented using a sequence of three CNOTs with alternating target and control, as shown in Fig. 2.2. In Fig. 2.2 we summarise some common two qubits gates, showing their circuitual form and matrix representation in the computational basis.

One and two-qubit gates are universal It can be proven that any n -qubit unitary can be written as a product of just two-qubit operations, and further that any such two-level unitary can be approximated *efficiently* with a composition of single-qubit gates and two-qubit controlled operations [20, 220]. This means that a restricted set of operations, namely single qubit gates and a controlled operation like the CNOT, is sufficient to implement an arbitrary n -qubit computation, and for this reason, such a pool of operators is called a *universal* gate set.

Examples of universal gate sets are $\{H, T, \text{CNOT}\}$ or $\{R_x, R_y, R_z, \text{CNOT}\}$, but there exist several different ways of combining single-qubit operations and two-qubit interactions to achieve universality. A closely related concept is that of *native* or *basis* gates, which are the set of physical transformations that can actually be performed on quantum computing hardware, and depends on the specific technology used to build the hardware (superconducting, ion traps, neutral atoms, ...). We briefly touch upon this theme in Sec. 2.1.5 when discussing the quantum circuit model.

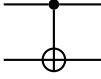

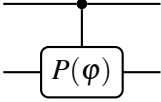
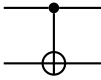
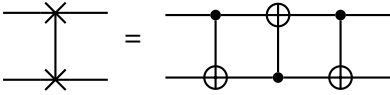
Name (abbr.)	Symbol/Circuitual rep.	Matrix representation
Controlled-NOT (CNOT)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled-Z (CZ)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
Controlled-phase (CP)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\varphi} \end{bmatrix}$
Controlled-U (CU)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_{00} & U_{01} \\ 0 & 0 & U_{10} & U_{11} \end{bmatrix}$
Swap (SWAP)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table 2.2: Summary of the most important two qubits operations. The name of the operation is shown, the symbol used to refer to the operation when implemented as a *gate* in a quantum circuit, and the matrix representation of the gate in the computational basis.

2.1.3 Density matrix formalism

So far we have described quantum states as vectors in a Hilbert space, yet there exists an equivalent, and often more appropriate, description of quantum states via operators, which go by the name of *density matrices*. This alternative formalism arises quite naturally in quantum mechanics when one wants to account for uncertainties in the knowledge of a quantum state.

Consider a set of quantum states $\{|\psi_i\rangle\}$ and a process that selects a state $|\psi_i\rangle$ from such a set with probability $p_i \in [0, 1]$, so that all probabilities sum up to one $\sum_i p_i = 1$. Given that each state is drawn probabilistically from the set, one can describe this statistical uncertainty by considering a weighted *mixture* of the states in the set. Formally, given an ensemble of states with corresponding probabilities $\{p_i, |\psi_i\rangle\}_i$, the *density matrix* which describes the quantum state of the system is given by the convex combination

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \quad \sum_i p_i = 1. \quad (2.18)$$

Such a state is called *mixed* state, and is used to describe the statistical uncertainty one could have about the state of a quantum system. These are opposed to the *pure* states described so far, which are instead used to describe systems whose state is known exactly, with no statistical uncertainty associated with it.

By definition, let \mathcal{H} be the Hilbert space associated with a quantum system, density matrices are linear squared operators on \mathcal{H} that are positive semidefinite $\rho \geq 0$ with unit trace $\text{Tr}[\rho] = 1$, and

are used to represent the quantum state of the system. The density matrix notation can also easily take into account pure states, represented by projectors $\rho = |\psi\rangle\langle\psi|$. Indeed, one defines the *purity* of a quantum state ρ as $\text{Tr}[\rho^2]$, which is highest when the state is pure

$$\text{Tr}[\rho^2] = \text{Tr}\left[|\psi\rangle\langle\psi|^2\right] = \text{Tr}[|\psi\rangle\langle\psi|] = 1,$$

and reaches the minimum value of $1/d$ when the quantum system is in the so-called *completely mixed* state $\rho = \mathbb{I}/d$, where d is the dimension of the Hilbert space of the system⁵ ($d = 2^n$ for a system of n qubits).

As discussed in Chapter 8, density matrices are of fundamental importance when dealing with *quantum channels*, a generalisation of the unitary evolution for open quantum systems, and the backbone of the mathematical description of noise processes acting on quantum systems.

Bloch representation of a single qubit mixed state We argued earlier that the Pauli matrices together with the identity form a basis of the space of 2×2 complex matrices, and in fact the state of a qubit can be expanded in this basis as the linear combination

$$\rho = \frac{\mathbb{I} + r_x X + r_y Y + r_z Z}{2}. \quad (2.19)$$

where the condition $\text{Tr}[\rho] = 1$ along with the fact that the Pauli matrices are traceless imposes that the identity appears with coefficient $1/2$, and $r_x, r_y, r_z \in \mathbb{R}$ because of the positivity of the density matrix and the hermiticity of the Pauli matrices. If the state is pure, then

$$1 = \text{Tr}[\rho^2] = \text{Tr}\left[\left(\frac{\mathbb{I} + r_x X + r_y Y + r_z Z}{2}\right)^2\right] = \frac{1 + r_x^2 + r_y^2 + r_z^2}{2} \implies r_x^2 + r_y^2 + r_z^2 = 1. \quad (2.20)$$

This is the equation of a sphere of radius one, and this is, indeed, another way to derive the Bloch sphere representation of a qubit of Fig. (2.1). On the contrary, if the state is not pure then the purity is less than one, which is equivalent to the condition

$$\text{Tr}[\rho^2] \leq 1 \implies r_x^2 + r_y^2 + r_z^2 = \bar{r}^2 \leq 1, \quad (2.21)$$

where we have introduced the *Bloch vector* $\mathbf{r} := (r_x, r_y, r_z)$, consisting of the coordinates of the quantum state along the Pauli axes. For example, the ground state $\rho = |0\rangle\langle 0|$ has Bloch vector $\mathbf{r} = (0, 0, 1)$, since it can be written in terms of the Pauli matrices as $\rho = (\mathbb{I} + Z)/2 = |0\rangle\langle 0|$. One can check that a general quantum state of the form in Eq. (2.2) has Bloch vector $\mathbf{r} = (\cos \varphi \sin \theta, \sin \varphi \sin \theta, \cos \theta)$.

Equations (2.20) and (2.21) then tell that pure states live on the surface of the Bloch sphere of Fig. 2.1, while mixed states correspond to points inside the sphere. The centre of the sphere is the completely mixed state $\rho = \mathbb{I}/2$, with the corresponding Bloch vector $\mathbf{r} = (0, 0, 0)$.

Evolution While pure states evolve after the application of a unitary operation U as $|\psi\rangle \xrightarrow{U} U|\psi\rangle$, the state obtained by acting with a unitary operation on a quantum system described by density matrix ρ is instead $\rho \xrightarrow{U} U\rho U^\dagger$.

Reduced density matrices One is often interested in studying the state of just a subsystem (for example, a single qubit) of a larger quantum system consisting of multiple parts. Let ρ_{AB} be the density matrix of a bipartite quantum system with Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$, where A and B denote the subsystem we are interested in. One defines the *reduced density matrix* of the system A

⁵This fact can be easily proven by considering an eigendecomposition of the density matrix, and using Lagrange multipliers to minimize the purity $\text{Tr}[\rho^2] = \sum_{i=1}^d p_i^2$ under the constraint $\sum_{i=1}^d p_i = 1$.

the density operator ρ_A given by the partial trace over the uninteresting degrees of freedom of B , namely

$$\rho_A := \text{Tr}_B[\rho_{AB}] = \sum_{i=1}^{d_B} (\mathbb{I}_A \otimes \langle \psi_i |_B) \rho_{AB} (\mathbb{I}_A \otimes |\psi_i \rangle_B) \equiv \sum_{i=1}^{d_B} \langle \psi_i | \rho_{AB} | \psi_i \rangle_B \quad (2.22)$$

where $\text{Tr}_B[\cdot]$ indicates the partial trace operation as defined above, $d_B = |\mathcal{H}_B|$ is the dimension of the Hilbert space of B , $\{|\psi_i \rangle_B\}_i$ is an orthonormal basis in \mathcal{H}_B .

One can check that the reduced density operator defined above is a valid quantum state as it is positive semi-definite $\rho_A \geq 0$ and with trace equal to one $\text{Tr}[\rho_A] = 1$.

2.1.4 Measurements and expectation values

Measurements are processes that extract classical information from quantum states. These usually occur at the end of a quantum computation (but also during it, depending on the task) to read out the final outcome of the implemented data processing task. The measurement process in quantum mechanics is a very delicate topic at a fundamental level since it involves the problem of the quantum to classical transition, but here we take an operational formulation following the standard probabilistic interpretation of quantum mechanics stemming from *Born's rule*, used to determine probabilities of measurement outcomes.

The probability that a quantum system described by quantum state $|\psi\rangle = \sum_i c_i |i\rangle$ when measured in the computational basis reduces to state $|k\rangle$ is given by

$$p_k = |\langle k | \psi \rangle|^2 = |c_k|^2. \quad (2.23)$$

In addition to measurements in the computational basis, a measure can be used to infer some physical properties of the quantum system under investigation. Since these properties can only take real values (as opposed to complex), the set of physical quantities to which an observer has access, which are referred to as *observables*, are represented mathematically by Hermitian operators $O = O^\dagger$, which have real eigenvalues. Examples of common observables are the Pauli matrices X, Y , and Z . The *expectation value* of an observable O on a quantum system described by pure state $|\psi\rangle$ is

$$\langle O \rangle := \langle \psi | O | \psi \rangle. \quad (2.24)$$

More generally, in the density matrix formalism, expectation values instead read

$$\langle O \rangle := \text{Tr}[O\rho], \quad (2.25)$$

where this formula clearly reduces to the first one for pure states $\text{Tr}[O|\psi\rangle\langle\psi|] = \langle \psi | O | \psi \rangle$. Since observables are Hermitian operators, O admits a spectral decomposition $O = \sum_i o_i |o_i\rangle\langle o_i|$, and so Eq. (2.25) can be also expressed explicitly as

$$\text{Tr}[O\rho] = \text{Tr} \left[\sum_i o_i |o_i\rangle\langle o_i| \rho \right] = \sum_i o_i \langle o_i | \rho | o_i \rangle = \sum_i o_i p_i. \quad (2.26)$$

where $p_i = \langle o_i | \rho | o_i \rangle$ is the probability of measuring the state ρ in $|o_i\rangle$.

In real experiments, the estimation of the expectation value of an observable is a resourceful multi-step process that requires the ability to repeatedly prepare and measure the quantum state, and then combine the measurement outcomes via classical post-processing. In the quantum computing jargon, a single act of measure is called *shot*, and the overall number of measurement shots used in an experiment, often indicated with M , affects the statistical accuracy of the estimation, which scales as $\mathcal{O}(1/\sqrt{M})$. In Algorithm 1 we summarise the steps needed to estimate the expectation value of an observable O on a quantum state ρ , and we now proceed to explain them in detail.

Algorithm 1: Estimate expectation value of an observable

Data: Quantum state $\rho \in \mathbb{C}^d \times \mathbb{C}^d$; observable O , number of shots M .
Result: $\bar{O} \approx \langle O \rangle = \text{Tr}[O\rho]$ with associated statistical error $\mathcal{O}(1/\sqrt{M})$.
for $m = 1, \dots, M$ **do**
 Prepare ρ ;
 Apply a change of basis operation U on ρ , such that $U^\dagger O U = \sum_{i=0}^{d-1} o_i |i\rangle\langle i|$ is diagonal
 in the computational basis;
 Do projective measurement on computational basis $\{|i\rangle\}_{i=0}^{d-1}$, find state $|k\rangle$;
 Store result $r_m = o_k$;
end
Classically compute the sample mean $\bar{O} := \frac{1}{M} \sum_{m=1}^M r_m$.

First, it is required that the experimenter has access to many copies of the quantum state ρ , or can prepare it efficiently multiple times. The preparation of the quantum state is then followed by a *change of basis* that makes the observable O diagonal in the computational basis, or equivalently, expresses ρ in the eigenbasis of the observable. This is usually a required step, as common quantum computing hardware can only perform projective measurements on the computational basis, and so one has to reframe every estimation procedure on this basis. Since O is Hermitian, there is a unitary U that diagonalizes the observable as $O \rightarrow O' = U^\dagger O U = \sum_i o_i |i\rangle\langle i|$. Then, by rotating the quantum state $\rho \rightarrow \rho' = U^\dagger \rho U$ and measuring the diagonal—in the computational basis—observable O' , one correctly obtains the desired expectation value

$$\sum_{i=0}^{d-1} o_i \langle i | U^\dagger \rho U | i \rangle = \text{Tr} \left[\sum_{i=0}^{d-1} o_i |i\rangle\langle i| U^\dagger \rho U \right] = \text{Tr} [U^\dagger O U U^\dagger \rho U] = \text{Tr}[O\rho] = \langle O \rangle, \quad (2.27)$$

where the first term on the left is the definition of $\text{Tr}[O'\rho']$ from Eq. (2.26), which, as required, only involves measurement along the computational basis $\{|i\rangle\}$.

As an example, consider the simple case of measuring the expectation value of the Pauli- X operator on a single-qubit quantum state ρ . Using $Z = HXH$, it is easy to check that a rotation of the qubit with a Hadamard gate $\rho \rightarrow \rho' = H\rho H$, followed by a measurement of the experimentally accessible Pauli- Z operator, correctly yields the desired expectation value $\langle X \rangle$, in fact

$$\langle Z \rangle_{\rho'} = \text{Tr}[Z\rho'] = \text{Tr}[ZH\rho H] = \text{Tr}[HZH\rho] = \text{Tr}[X\rho] = \langle X \rangle_\rho.$$

We remark that the change of basis step is only a practical requirement to circumvent the hardware constraints that only allow for measurements on the computational basis, but adds nothing fundamental to the computation. In addition, note that the operator U that diagonalizes the observable O has to be calculated classically, before the measurement procedure begins.

After preparation of the state and subsequent change of basis, the system is finally measured in the computational basis. Suppose that upon measurement the state is found in state $|k\rangle$ corresponding to the eigenvalue o_k : this number constitutes the result of the measurement. By repeating the entire preparation and measurement procedure M times, one gathers a sample of measurement results $\{r_1, \dots, r_M\}$, where each result $r_m \in \{o_0, o_1, \dots, o_{d-1}\}$ is the eigenvalue obtained on the m -th measurement shot. The values r_m are *independent*—because they come from independent measurement events—random variables whose statistical properties are defined by

$$\mathbb{E}[r_m] = \mathbb{E}[O] := \langle O \rangle, \quad \text{Var}[r_m] = \text{Var}[O] := \langle O^2 \rangle - \langle O \rangle^2, \quad \forall m = 1, \dots, M \quad (2.28)$$

where the expectation values and variances of the random variables r_m are evaluated over the probability distribution of measurement outcomes given by Born's rule (2.23). The sample mean

$\bar{O} := \sum_{m=1}^M r_m / M$ is an unbiased estimator of the true expectation value $\langle O \rangle$, with variance

$$\mathbb{E}[\bar{O}] = \frac{1}{M} \sum_{m=1}^M \mathbb{E}[r_m] = \langle O \rangle, \quad (2.29)$$

$$\text{Var}[\bar{O}] = \frac{1}{M^2} \sum_{m=1}^M \text{Var}[r_m] = \frac{1}{M} \text{Var}[O], \quad (2.30)$$

where we have used Eqs. (2.28), and the fact that the r_m are independent random variables to move the variance inside the sum in the second equation.

Assuming that the variance of the observable is bounded and independent of the system size, as it happens for observables made of tensor product of Pauli matrices⁶, then

$$\text{Var}[O] = \langle O^2 \rangle - \langle O \rangle^2 = \text{Tr}[O^2 \rho] - \text{Tr}[O \rho]^2 \in \mathcal{O}(1), \quad (2.32)$$

and so the statistical error associated to the empirical mean \bar{O} scales as $\text{Std}[\bar{O}] \in \mathcal{O}(1/\sqrt{M})$. Alternatively, suppose one wants to calculate the expectation value with precision ε with a failure probability of at most δ , then by Chebyshev's inequality

$$P(|\bar{O} - \langle O \rangle| > \varepsilon) \leq \frac{\text{Var}[\bar{O}]}{\varepsilon^2} = \delta \implies M \in \mathcal{O}\left(\frac{1}{\delta \varepsilon^2}\right), \quad (2.33)$$

which states that the number of measurements M to reach a target accuracy ε scales quadratically with the desired precision. A similar scaling can also be obtained with other statistical inequalities, for example via Hoeffding's inequality [138].

We conclude by noticing that in practical scenarios one is often interested in measuring either simple observables made of tensor products of Pauli matrices, often referred to as *Pauli strings*

$$O = \sigma_1 \otimes \sigma_2 \otimes \cdots \otimes \sigma_n, \quad (2.34)$$

where n is the number of qubits in the system, or in weighted sums of these Pauli strings

$$O = \sum_{k=1}^P \gamma_k O_k = \sum_{k=1}^P \gamma_k \sigma_1^{(k)} \otimes \sigma_2^{(k)} \otimes \cdots \otimes \sigma_n^{(k)}, \quad (2.35)$$

where the coefficients are real to ensure Hermiticity $\gamma_k \in \mathbb{R}$, and $\sigma_i^{(k)} \in \{\mathbb{I}, X, Y, Z\}$ are single-qubit Pauli matrices. In this last case, very prominent in quantum chemistry applications [106], the usual approach is to estimate each Pauli string separately $\langle O_k \rangle$, and then combine the results classically with the coefficients γ_k , via $\langle O \rangle = \sum_k \gamma_k \langle O_k \rangle$.

Clearly, measuring a single Pauli string is much simpler, and the estimation is associated with a lower variance compared with the case of more complex observables when a fixed budget of measurement shots is allowed. Various strategies have been proposed in the literature to optimize the measurement resources needed to estimate expectation values of the form in (2.35), most of them based on a clever grouping of commuting Pauli strings that can be measured at the same time [29], or for example using adaptive methods to reduce the statistical fluctuations associated with the estimation [106].

⁶This can be easily proved as follows. Let $O = \sigma_1 \otimes \sigma_2 \otimes \cdots \otimes \sigma_n$ be an observable made of tensor products of single-qubit Pauli matrices $\sigma_i \in \{\mathbb{I}, X, Y, Z\}$. Since Pauli matrices are involutory, $\sigma_i^2 = \mathbb{I}$, then $O^2 = \mathbb{I}_n$. In addition, since the eigenvalues of $O = \sum_i o_i |o_i\rangle\langle o_i|$ are either $o_i \in \{\pm 1\}$, expectation values $\langle O \rangle$ are bounded

$$|\langle O \rangle| = |\text{Tr}[O \rho]| = \left| \sum_i o_i \langle o_i | \rho | o_i \rangle \right| \leq \sum_i |o_i| \langle o_i | \rho | o_i \rangle = \sum_i \langle o_i | \rho | o_i \rangle = 1, \quad (2.31)$$

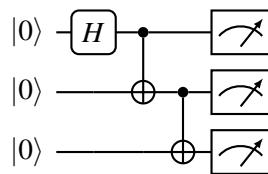
where we have used that $\langle o_i | \rho | o_i \rangle \geq 0$ are positive numbers because $\rho \geq 0$ is a positive operator, the eigenstates $\{|o_i\rangle\}_i$ are a basis of the space, and the state is normalized $\text{Tr}[\rho] = 1$. Thus, finally one has $0 \leq \text{Var}[O] = \text{Tr}[O^2 \rho] - \text{Tr}[O \rho]^2 = 1 - \text{Tr}[O \rho]^2 \leq 1$, where again we have used $\text{Tr}[\rho] = 1$ and $0 \leq \text{Tr}[O \rho]^2 \leq 1$ from (2.31).

2.1.5 The quantum circuit model

As discussed earlier, a generic quantum computation can be expressed in terms of single- and two-qubit operations, which makes it easy to represent it graphically via a sequence of a few circuit symbols, already introduced in Tables 2.1 and 2.2.

In quantum circuit diagrams, time evolves from left to right, single wires are used to indicate separate quantum systems (qubits in our case), and the single- and two-qubit gates are represented with the symbols shown in the corresponding summary tables. Unless otherwise stated, qubits are assumed to start in the ground state, that is all qubits are in the $|0\rangle$ state when the computation is started. At last, at the end of the circuit some or all the qubits may be measured, in which case an appropriate symbol is shown on the corresponding wire. Unless explicitly said otherwise, every measurement is assumed to be in the computational basis.

A simple example of a quantum circuit is the following, which creates and measures the three-qubit *maximally entangled* GHZ state $|\psi_{\text{GHZ}}\rangle = (|000\rangle + |111\rangle)/\sqrt{2}$.



2.1.5.1 Executing a quantum circuit on a real quantum device

The quantum circuit model is a useful tool to describe the logical action of quantum computation through a pictorial representation. Furthermore, although quantum circuits are the standard format for submitting instructions to a real quantum computer for execution, they are by no means a faithful description of what actually happens on the device.

In fact, for a logical quantum circuit to be executed on a real device, some classical preprocessing steps are needed. First, it is necessary to identify a subset of the available physical qubits on the machine that matches the connectivity required by the two-qubits gates in the circuit to be run. Secondly, one has to express every gate in the original circuit in terms of the so-called *native basis gates*, which are the set of physical operations that can actually be implemented on the device. These strongly depend on the actual technology used to build the quantum computer, and in Table 2.3 we report the native gates available on some common quantum computing hardware. Following the discussion in Sec. 2.1.2, these single and two-qubit operations provide different universal sets of gates that can implement any quantum computation.

Manufacturer	Technology	Native gates	Refs.
IBM	Superconducting	$\{\sqrt{X}, X, R_Z, \text{CNOT}\}$	[145]
Google	Superconducting	$\{U(\theta, \phi, \lambda), \text{Sycamore gate}\}$	[16, 114]
IonQ	Trapped-ions	$\{R_X, R_Y, \text{Mølmer-Sørensen}\}$	[217, 328]

Table 2.3: Examples of single- and two-qubits gates available on some current quantum computing hardware. The $U(\theta, \phi, \lambda)$ operation is the general single-qubit rotation of (2.10), and the “Sycamore gate” is a two-qubit gate similar to a combination of a SWAP gate and a controlled phase rotation. The Mølmer-Sørensen gate [212] is a two-qubit gate of the form $XX(\phi) = e^{-i\phi X \otimes X/2}$ common in ion-trap based architectures. The single-qubit rotation gates on available on IONQ devices are variations of R_X and R_Y rotations, for a precise definition see [217].

This process of rewriting the circuit into an appropriate form goes by the name of circuit *compilation*, or *transpilation*, and its result is to output a new quantum circuit which is (approximately) equivalent to the original one, but that can be readily executed on the machine. As discussed in the

next section, current quantum hardware is limited in both size and performance, and a quantum circuit may lend itself to better execution on specific backends, due to a more favourable mapping in terms of required qubit-qubit connection topology, and/or available gates.

In the next section, we provide a brief overview of the state of the art of current quantum computing technology, discussing both the technological aspects as well as their scientific relevance.

2.1.6 The NISQ era of quantum computation

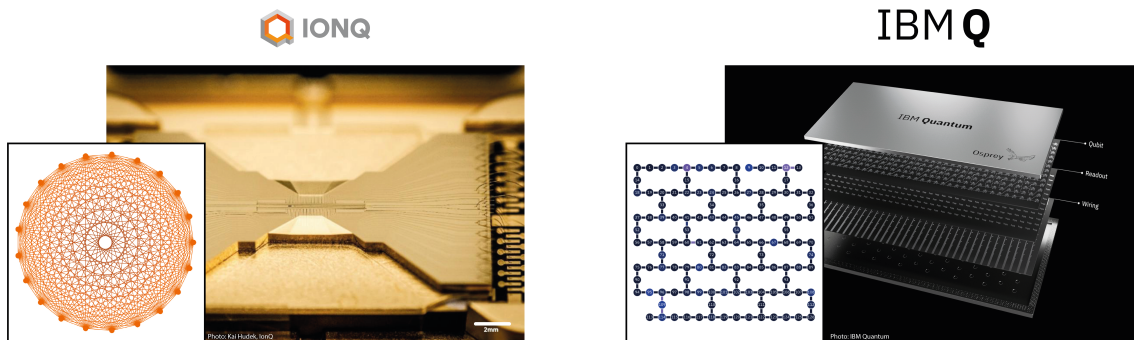


Figure 2.2: Examples of currently available quantum computers based on two different technologies: trapped-ions (IonQ) [148] and superconducting circuits (IBM Quantum) [145]. In the panels next to the devices, examples of the usual connectivity between qubits found in the respective technologies are shown, specifically the Aria device with $n = 21$ qubits with an all-to-all connectivity [10], and IBM’s `ibm_washington` device with $n = 127$ qubits with leveraging a grid-like connectivity [92].

Although enormous technological progress made it possible to build the first generations of quantum computers, these are still far from the realisation of universal fault-tolerant quantum computers, which are ideal quantum computers in which errors in the computation can be detected and corrected, through a procedure called Quantum Error Correction (QEC) [220].

Instead, the current era of quantum computation has been dubbed *Noisy Intermediate-Scale Quantum* [236], *NISQ* for short, which is used to denote near-term quantum devices which are imperfect and not error corrected, hence are strongly affected by noise, and can only leverage a limited number of qubits, and thus are also limited in scale. Several experimental platforms are being used to develop quantum computing solutions [6, 294], based, for example, on superconducting circuits [16, 145], trapped-ions [148, 241], neutral atoms [135, 245], photonic chips [240, 329] and others⁷. In Fig. 2.2 we report two examples of currently available quantum computers based on trapped ions and superconducting circuits, which are arguably the leading technologies for constructing quantum computers in the NISQ era. In the figure, both the physical Quantum Processing Units (QPUs), and a graphical representation of the layout of the qubits on such chips.

Although the pace at which these machines are developed suggests a quantum variant of the classical Moore’s law to hold, NISQ devices still face serious constraints, which we briefly discuss in what follows. First of all, state of the art quantum computers are limited in size, consisting of about dozens to a few hundred qubits depending on the technology, with first generations of QPUs with more than a thousand qubits only planned to appear in the following years [146]. In order to reach a clear sign of quantum advantage and surpass the regime of classical simulability, it is necessary to be able to efficiently scale up these machines. For example, it is estimated that roughly 20 million *physical* qubits —as opposed to *logical*⁸— are required to implement a non-trivial

⁷In this list we omit computing platform based on quantum annealing, like D-Wave’s machines [71], as they cannot implement universal quantum computation.

⁸By *physical* qubit one refers to a single noisy hardware implementation of a two-level quantum system corresponding to a qubit. These are opposed to *logical* qubits, which are instead ideal versions of the qubits that are immune to errors. A single logical qubit can be represented using multiple physical qubits along with quantum error correction codes [220].

application of Shor’s algorithm, arguably the most convincing example of exponential speedup of quantum computation over classical one [108]. Quantum computers of this size and capabilities will probably require new breakthroughs both on the experimental and algorithmic side, and will only appear in the next decades.

In addition, as we already discussed previously in Sec. 2.1.5.1, the types of operation that can be implemented on a device are often quite limited. This is especially true for two-qubit gates, which can only be applied directly to qubits that are physically close on the machine, so that they can be made to interact. The *connectivity* between qubits on a device, as shown in Fig. 2.2, imposes a serious constraint on the class of algorithms that can be run on a specific machine. While the limited connectivity is a clear issue for superconducting-based quantum computers, other platforms, like trapped-ions, implement an all-to-all interaction that makes it possible to operate two-qubit gates on any pair of qubits on the device. However, operations (both single- and two-qubit gates) on these machines can be orders of magnitudes slower than on superconducting chips, with the time to implement a quantum gate ranging in the order of microseconds $t \sim 1 - 100\mu\text{s}$ for the former [43] and nanoseconds $t \sim 1 - 100\text{ns}$ for the latter [145].

The most compelling problem with near-term devices is noise that affects the qubits inside the computer, which strongly limits the computing capabilities of the hardware. Errors in quantum computers occur due to undesired interaction of the quantum systems with the external environment, as well as due to faulty execution of the operations inside the computer. The former type of noise is commonly measured by *coherence times*, which characterise the *quality* of the qubit by assessing the time it takes for it to lose its quantum properties due to unwanted interaction with the external environment. For example, common values of coherence times for current superconducting architectures are of about $T \sim 10 - 100\mu\text{s}$ [145], while trapped-ions usually have longer times of about $T \sim 1 - 100\text{s}$ [148], corresponding to higher quality qubits. We will discuss more in detail how to describe noise in quantum systems in Chapter 8.

As for errors introduced by an imperfect realisation of the quantum gates—especially two-qubit gates—, these are measured in terms of gate error rates via a procedure called Randomised Benchmarking [133, 191]. Current error rates are of the order of 0.01% and 1% for single and two-qubit gates respectively for superconducting circuits, and roughly one order of magnitude less for ion traps.

It is then clear that multiple sources concur to the realisation of an effective device, namely the number of qubits, the connectivity map, and the error rates. For this reason, new figures of merit such as Quantum Volume (QV) have been introduced to provide a unifying measure to assess the capability of near-term quantum computing devices [29, 69]. Roughly speaking, the QV is a single real number that measures the largest square-shaped (i.e. number of qubits equal to the depth of the circuit) random quantum circuit that the quantum computer can execute successfully. It can be understood as the number of “effective” qubits available on the machine, and it is formally defined as [69]

$$\log_2 QV = \arg \max_n \min(n, D(n)) \quad (2.36)$$

where n is the number of qubits and $D(n)$ is the depth of the quantum circuit, that is the minimum number of steps needed for quantum gates to be applied in parallel to implement a quantum circuit. At the time of writing, the highest Quantum Volumes reported for superconducting and trapped-ions architectures are, respectively, $QV = 512$, corresponding to running square circuits of size $\log_2 QV = 9$, for IBM’s superconducting chip Prague with a total of $n = 27$ qubits [243], and $QV = 8192$ ($\log_2 QV = 13$) for Quantinuum’s trapped-ions based System Model H1 [242] which has a total of $n = 20$ qubits.

Every technology has its own advantages and drawbacks, and it is not yet clear which one will provide the most effective solution towards scalable and fault-tolerant quantum computers. In the meantime, near-term devices not only constitute a necessary step towards the implementation of

large-scale universal quantum computers but also provide new tools to investigate the limits of quantum mechanics and provide a new paradigm for performing computation. Indeed, the topic of the next section is to discuss what NISQ devices can be used for, thus introducing *variational quantum algorithms*, which are a class of quantum algorithms specifically tailored for current quantum computing devices.

2.2 Variational Quantum Algorithms

Variational Quantum Algorithms (VQAs) are the leading proposal to exploit current quantum computing platforms, based on the hope that it is possible to achieve a meaningful *quantum advantage* already in the non error-corrected regime, before standard quantum algorithms, like Shor’s factoring, can be realised at scale. Moreover, and regardless of any quantum advantage, the research on variational quantum algorithms is of interest on its own, both on a fundamental level because it conceives a new paradigm of hybrid computation where classical and quantum resources are used in tandem to achieve a task, and on a practical level because it fosters the development of new software and hardware solutions for quantum computing by providing compelling use cases.

NISQ devices are limited in size and inherently noisy, and the idea of variational quantum algorithms is to circumvent the problem simply by using them as little as possible, only for the bare minimum, while outsourcing all remaining tasks to a classical computer. This idea then identifies a class of hybrid quantum-classical algorithms in which quantum and classical computational resources are used in combination to solve a task.

The second ingredient of variational algorithms, to which they owe their name⁹, make them also very similar to the highly successful field of machine learning. In fact, variational quantum algorithms are optimisation-based procedures that tackle a problem by first encoding its solution as the minimum of an appropriately defined *cost function* $C(\boldsymbol{\theta})$ depending on some tunable parameters $\boldsymbol{\theta}$, and then iteratively *vary* these parameters, usually via gradient-based methods, to find the minimum of the function, hence the solution. The incredible results achieved by deep learning in recent years have shown how versatile and powerful learning-based procedures can be [83, 178, 210]. Variational quantum algorithms then introduce free adjustable parameters inside a quantum computation in order to cope with the serious constraints imposed by current NISQ hardware. In this way, however, the success and/or runtime guarantees of standard textbook quantum algorithms like Grover’s and Shor’s are lost, as the optimisation procedures in VQAs are usually highly non-convex, and so theoretical analysis of the performances can only go so far.

In all, VQAs essentially trade off guarantees of success with the feasibility of execution, in the hope of retaining a quantum advantage of some sort.

2.2.1 The basis of variational quantum algorithms

At the basis of any variational algorithm is the definition of a cost, or *loss*, function that measures how well the algorithm is performing, and an *ansatz* circuit, which is a guess quantum circuit with tunable parameters that should be able to represent a good solution to the problem.

Let $U(\boldsymbol{\theta})$ be the unitary of the parameterised quantum circuit (PQC) ansatz, and $\boldsymbol{\theta} \in \mathbb{R}^p$ the vector of the parameters. The goal of VQAs is to find an optimal set of parameters that minimises the cost function $C(\boldsymbol{\theta}) : \mathbb{R}^p \rightarrow \mathbb{R}$, namely

$$\boldsymbol{\theta}_{\text{opt}} = \arg \min_{\boldsymbol{\theta}} C(\boldsymbol{\theta}), \quad (2.37)$$

⁹The term “variational” as originally proposed by Peruzzo et al. [230] comes from the Rayleigh-Ritz *variational* principle to compute lowest energy eigenvalues. The term variational in turn comes from the idea of *varying* the parameters of a trial model in order to solve a task, a concept that is at the core of modern machine learning (ML) as well. The connection between VQAs and ML was developed later as the field gained momentum, and now there is a fruitful exchange of ideas and concepts between these two topics.

where the cost is some function of the expectation values of a set of observables $\{O_k\}$ measured on the parameterized state generated by the variational circuit, that is

$$C(\boldsymbol{\theta}) = \sum_k f_k(\text{Tr}[O_k \rho_{\boldsymbol{\theta}}]) = \sum_k f_k(\text{Tr}[O_k U(\boldsymbol{\theta}) |0\rangle\langle 0| U(\boldsymbol{\theta})^\dagger]). \quad (2.38)$$

This definition highlights that the cost function depends explicitly on the parameters $\boldsymbol{\theta}$, and implicitly actually also on the set of observables $\{O_k\}_k$ and the specific shape of the circuit ansatz $U(\cdot)$, so that $C = C(\boldsymbol{\theta}; \{O_k\}_k; U)$ [58]. While this definition is fully general, in practical scenarios one usually considers simpler costs given by the expectation value of a single observable¹⁰

$$C(\boldsymbol{\theta}) = \langle O \rangle_{\boldsymbol{\theta}} = \text{Tr}[O U(\boldsymbol{\theta}) |0\rangle\langle 0| U(\boldsymbol{\theta})^\dagger]. \quad (2.39)$$

Ideally, the loss function should fulfil certain desirable properties to be considered a good candidate for a cost in a variational quantum algorithm. Firstly, it should be classically hard to compute but *efficiently* calculable with a quantum device, for if it were not, this would negate any hope of quantum advantage. Secondly, the cost function should be *trainable*, that is there exists a procedure capable of finding the minimum of the cost with some guarantees of success. This latter condition is particularly relevant for variational algorithms, as the loss landscape of these procedures is often found to be very flat and thus difficult to navigate via standard optimization methods, a phenomenon which goes by the name of *barren plateaus*, see Sec. 2.2.4.2. Finally, two other useful requirements are that the global minimum of the loss function should only be attained when the true solution to the problem is found, and not otherwise; and that the cost function should always convey a measure of the suitability of the proposed solution, so that smaller values of the cost always correspond to better solutions. When the latter two conditions are met, the cost function is often referred to as *faithful* and *operationally meaningful*, respectively [58].

Hence, given a properly defined cost function, variational algorithms then proceed by combining quantum and classical resources in an iterative loop as follows:

- (i) On the quantum computer: estimate the cost function $C(\boldsymbol{\theta})$ for the current values of the parameters $\boldsymbol{\theta}$ via repeated measurements;
- (ii) On the classical computer: input the outcome in a classical optimisation algorithm that proposes a new value for the parameters $\boldsymbol{\theta}'$, so that the cost is lower $C(\boldsymbol{\theta}') < C(\boldsymbol{\theta})$;
- (iii) repeat steps (i)-(ii) until stop conditions are met (convergence, execution time, ...).

These steps are schematically represented in Fig. 2.3, which shows the usual way of picturing the hybrid quantum-classical loop of variational quantum algorithms.

Two questions remain to be answered. First, what is the explicit form of the parameterised quantum circuit $U(\boldsymbol{\theta})$ and how can one choose it? Secondly, what are the usual strategies for performing the optimisation loop? Let us now proceed by addressing these two topics.

2.2.2 Parameterised quantum ansätze

Just as in standard parametric models, also in variational quantum algorithms the functional form of the model has to be fixed *a priori*, and this is done by considering a specific parameterised quantum circuit $U(\boldsymbol{\theta})$. Such a choice is referred to as an *ansatz* circuit, and while there is no unique prescription for constructing a good one, some guiding principles can be used to identify good candidates.

It is clear that the choice of ansatz plays a key role in determining the effectiveness of the variational algorithm. For example, the model may be too simple to express the target solution, or it may be so complex that it makes optimisation difficult, thus favouring convergence to sub-optimal solutions. Any such ansatz prevents a good result from the variational algorithm. We

¹⁰Note that this simplified loss is obtained whenever the general cost in Eq. (2.38) depends linearly on the expectation values, that is $f_k(\langle O_k \rangle) = c_k \langle O_k \rangle$. Then, by linearity of the trace and considering the overall observable $O = \sum_k c_k O_k$, one obtains Eq. (2.39). This is the case of quantum chemistry applications, as discussed earlier at the end of Sec. 2.1.4.

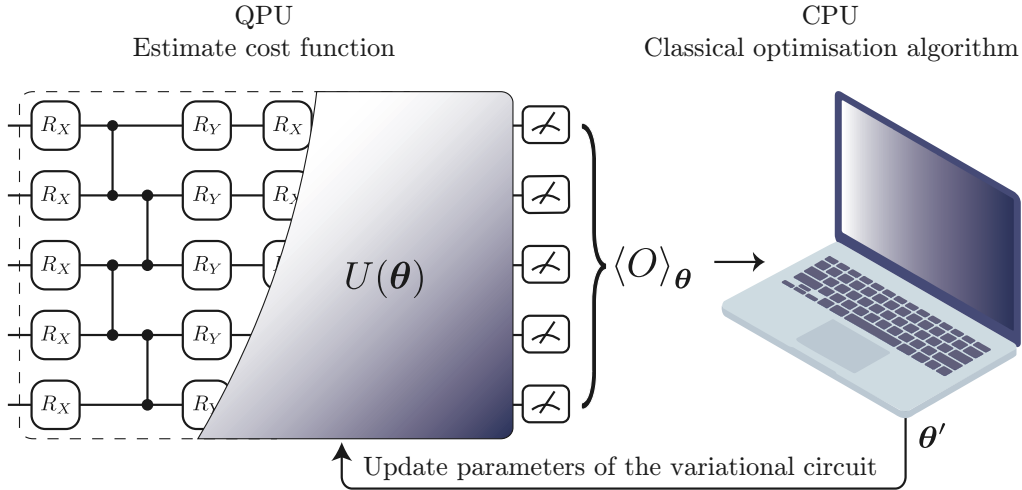


Figure 2.3: Schematic representation of variational quantum algorithms. First, on the left, a parameterised quantum circuit with parameters θ is used to prepare a variational trial state $|\psi\rangle = U(\theta)|0\rangle$ and a cost function $C(\theta) = \langle O \rangle_{\theta} = \langle \psi_{\theta} | O | \psi_{\theta} \rangle$ is measured on a Quantum Processing Unit (QPU). Then, on the right, the result is passed to a classical processing unit (CPU), that runs an optimisation algorithm to find new values for the parameters θ' corresponding to a lower cost. The loop is repeated until convergence to a minimum of the cost is reached or stopping criteria are met.

hereby anticipate that these concepts of expressibility and trainability are indeed related, as will be discussed in detail in Sec. 2.2.4. In addition, the most important—and almost trivial—condition that an ansatz should satisfy is that it should be feasible in practice, in that there is a near-term quantum computing device capable of implementing it.

Generally, all ansätze share the common feature of being defined in terms of a repeated structure of similar blocks arranged in sequential layers. For example, many variational ansätze can be expressed as

$$U(\theta) = \prod_{\ell=1}^L W_{\ell} U_{\ell}(\theta_{\ell}) = W_L U_L(\theta_L) \cdots W_1 U_1(\theta_1), \quad (2.40)$$

where $U_{\ell}(\theta_{\ell})$ are parameterised gates, usually single-qubit Pauli rotations, and W_{ℓ} on the other hand are fixed operations usually consisting of two-qubits entangling gates, such as CNOTs or CZ. The number of layers L in the circuit is an important hyperparameter in the definition of the ansatz and decides the overall depth of the quantum circuit. The circuit shown in Fig. 2.3 is an example of such a layered structure.

In the following, we summarise the most common strategies used in the literature to propose ansätze.

Hardware Efficient Ansätze (HWE) This class of ansätze contains all those parameterised circuits that are specifically thought to be easily run on current quantum hardware [155], and are often used as a starting point for exploring a variational approach to solving a problem, especially when it is not straightforward to incorporate knowledge about the task to be solved in a circuitual form. All the gates used in the ansatz are either taken from the native basis set of the device, or can be implemented with only a few of them. Most importantly, two-qubit operations act only on those pairs of qubits that are physically connected on the device, thus respecting the connectivity map available on the hardware.

Various properties of these circuits have been studied in the literature, regarding their expressibility [281], entanglement [18], performances on benchmark problems [144], as well as trainability

properties [57, 202].

Problem-inspired Ansätze Whenever the ansatz is built by leveraging the knowledge of the underlying physics of the problem to be solved, one says that the ansatz is problem-inspired. This is often the case for quantum chemistry applications, and the most prominent example in this class is the Unitary Coupled Cluster (UCC) ansatz, specifically thought for dealing with electronic structure calculations, also used in the seminal work by Peruzzo et al. [230] on variational quantum algorithms. Various generalisations of the UCC ansatz for several quantum chemistry problems have been introduced, and one can find detailed information in the reviews [8, 29, 58, 301]. Another example of a problem-inspired ansatz for quantum chemistry is the Hamiltonian Variational Ansatz inspired by the adiabatic state preparation theorem [317].

Regarding machine learning applications of variational quantum algorithms, and inspired by the classical machine learning literature on geometric deep learning [40], many recent proposals focus on constructing so-called *equivariant* quantum circuits, that are parameterised quantum circuits that encode the symmetries of the problem to be solved [173, 208, 219, 284].

By focusing specifically on a restricted class of relevant models, problem-inspired ansätze usually perform better and are easier to optimise with respect to problem-agnostic ones, even though this advantage may come at a cost of a more difficult implementation on real hardware.

QAOA-like Ansätze The Quantum Approximate Optimisation Algorithm (QAOA) [100] is a widely studied parameterised ansatz to deal with combinatorial optimisation problems on near-term devices. The variational ansatz is given by a layered alternating structure

$$U(\boldsymbol{\beta}, \boldsymbol{\gamma}) = \prod_{\ell=1}^L e^{-i\beta_{\ell} H_M} e^{-i\gamma_{\ell} H_P}, \quad (2.41)$$

where H_M is called mixer Hamiltonian, H_P is the problem Hamiltonian whose ground state encodes the solution to the combinatorial problem at hand, L is a hyperparameter that defines the accuracy of the procedure, and $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ are two sets of parameters to be optimised.

This ansatz essentially implements a Trotterised adiabatic evolution of degree L , that slowly evolves an easy to prepare ground state of the mixer Hamiltonian H_M , to the target ground state of the problem Hamiltonian H_P . It has been shown that the alternating structure of QAOA implements a universal dynamics for quantum computation [186], and an extension of QAOA to deal with more general classes of Hamiltonians has been proposed with the name of Quantum Alternating Operator Ansatz [122].

Adaptive Ansätze The optimisation of the variational angles alone is often not enough to obtain a well performing algorithm. In this case, one can relax the condition of fixing the circuit ansatz upfront, and instead *adaptively* optimise its structure along with that of the variational parameters. Examples are the RotoSolve algorithm [222] that optimizes both the angle and the axis of single qubit rotation gates, ADAPT-VQE algorithms [118, 298] that procedurally grow an ansatz by adding gates to the circuit from a pool of operators, and other approaches that try to add but also remove gates to keep the circuit shallow [32]. The adaptive generation of superior ansätze comes at the cost of optimisation problems that are hard to solve in general, but that can be addressed using approximate heuristics like Evolutionary Algorithms [248].

2.2.3 Optimisation of variational quantum algorithms

Once a cost function and an ansatz have been defined, one can then proceed with the optimisation loop to find the optimal solution of Eq. (2.37). A simple yet effective strategy to solve optimisation problems of differentiable functions is gradient-descent methods, widely used and studied in the classical machine learning literature. These procedures are first-order methods that need access

to the first derivatives of the function, as opposed to zeroth- and second-order ones, that instead require only function evaluations or access to the Hessian, respectively.

Be $\boldsymbol{\theta}^{(t)}$ the value of the variational parameter at a time step t , and $C(\boldsymbol{\theta}) \in \mathbb{R}$ the scalar cost function to be minimised. Gradient descent methods iteratively change the value of the parameters by moving against the direction of the gradient of the function evaluated at that point, namely

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}^{(t)}}. \quad (2.42)$$

where $0 < \eta \ll 1$ is a hyperparameter called *learning rate* that is used to tune the step size of the algorithm. As long as the learning rate is small enough, this update rule will propose a new value for the parameters corresponding to a lower cost. This can be seen by Taylor expanding the cost at step $t + 1$ around the parameters of the previous step t , as

$$C(\boldsymbol{\theta}^{(t+1)}) = C(\boldsymbol{\theta}^{(t)} - \eta \nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta}^{(t)})) \quad (2.43)$$

$$\approx C(\boldsymbol{\theta}^{(t)}) + \nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta}^{(t)}) \left(-\eta \nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta}^{(t)}) \right) + \mathcal{O}(\eta^2) \quad (2.44)$$

$$= C(\boldsymbol{\theta}^{(t)}) - \eta \left\| \nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta}^{(t)}) \right\|_2^2 \leq C(\boldsymbol{\theta}^{(t)}). \quad (2.45)$$

where $\|\cdot\|_2$ denotes the 2-norm (or Euclidean norm) of a vector, and the last inequality comes from the fact that both the learning rate and the gradient norm are positive quantities.

The basic gradient descent update rule in Eq. (2.42) can be easily improved and generalised, for example taking into account information about the second derivatives of the function (e.g. BFGS [244]), employing a stochastic approximation of the exact gradient (e.g. SPSA [287]), or adding *momentum* terms and adaptive learning rates (e.g. ADAM [165]). The research on numerical optimisation methods is very vast and active, providing useful tools for training variational quantum algorithms [290, 321].

2.2.3.1 Parameter shift-rule

Gradient-based methods require access to the first (partial) derivatives of the target function, which can be approximated numerically with the (central) finite-difference formula

$$\partial_i C(\boldsymbol{\theta}) := \frac{\partial C(\boldsymbol{\theta})}{\partial \theta_i} \approx \frac{C(\boldsymbol{\theta} + \varepsilon \mathbf{e}_i) - C(\boldsymbol{\theta} - \varepsilon \mathbf{e}_i)}{2\varepsilon}, \quad 0 < \varepsilon \ll 1 \quad (2.46)$$

where $\mathbf{e}_i = (0, \dots, 1, \dots, 0)$ is a unit vector with entries zero everywhere except for a one in the i -th position, and the equality sign is recovered in the limit of vanishing displacement $\varepsilon \rightarrow 0$. It turns out that a similar but *exact* formula holds for derivatives of parameterised quantum circuits, a fact which is now commonly referred to as *parameter-shift rule* [57, 198, 209, 270, 322]. This rule is a direct consequence of the rotation-like nature of the parameterised gates used in variational circuits, and we hereby prove this formula following the derivation presented in ref. [198].

Common parameterised operations used in variational quantum circuits are rotation-like gates of the form

$$V_j(\theta_j) = e^{-i\theta_j P_j/2}, \quad (2.47)$$

where θ_j is a variational angle, and P_j is the Hermitian generator of the gate. When P_j is involutory ($P_j^2 = \mathbb{I}$), then the exponential can be recast in trigonometric form as (see Eq. (2.9))

$$V_j(\theta_j) = \cos \frac{\theta_j}{2} \mathbb{I} - i \sin \frac{\theta_j}{2} P_j. \quad (2.48)$$

Note that many operations can be expressed in this way, including single-qubit rotations and more generally any rotation generated by tensor products of Pauli matrices.

Let $C(\boldsymbol{\theta}) = \text{Tr}[OU(\boldsymbol{\theta})\rho U(\boldsymbol{\theta})^\dagger]$ be the cost function to be differentiated, and $V_j(\theta_j)$ be the parameterised gate depending on the variable θ_j with respect to which we want to calculate the derivative. Consider a bipartition of the circuit $U(\boldsymbol{\theta})$ containing all the gates that act before (U_B) and after (U_A) the operation of interest $V_j(\theta_j)$ takes place¹¹

$$U(\boldsymbol{\theta}) = U_A V_j(\theta_j) U_B, \quad (2.49)$$

where the dependence on the remaining variational parameters in U_B and U_A is suppressed for ease of notation. The cost function can then be written as

$$C(\boldsymbol{\theta}) = \text{Tr}[OU_A V_j(\theta_j) U_B \rho U_B^\dagger V_j^\dagger(\theta_j) U_A^\dagger] = \text{Tr}[O_A V_j(\theta_j) \rho_B V_j^\dagger(\theta_j)] \quad (2.50)$$

where $O_A = U_A^\dagger O U_A$ and $\rho_B = U_B \rho U_B^\dagger$, and we isolated the dependence of the cost on the variable of interest θ_j . Substituting the trigonometric formula (2.48) in the expression above one obtains

$$\begin{aligned} C(\boldsymbol{\theta}) &= \text{Tr} \left[O_A \left(\cos \frac{\theta_j}{2} \mathbb{I} - i \sin \frac{\theta_j}{2} P_j \right) \rho_B \left(\cos \frac{\theta_j}{2} \mathbb{I} + i \sin \frac{\theta_j}{2} P_j^\dagger \right) \right] \\ &= \text{Tr} \left[\cos^2 \frac{\theta_j}{2} O_A \rho_B + i \sin \frac{\theta_j}{2} \cos \frac{\theta_j}{2} \left(O_A \rho_B P_j^\dagger - O_A P_j \rho_B \right) + \sin^2 \frac{\theta_j}{2} O_A P_j \rho_B P_j^\dagger \right] \\ &= \frac{1 + \cos \theta_j}{2} \text{Tr}[O_A \rho_B] + \frac{1 - \cos \theta_j}{2} \text{Tr}[O_A P_j \rho_B P_j^\dagger] + \frac{i}{2} \sin \theta_j \text{Tr} \left[O_A \left(\rho_B P_j^\dagger - P_j \rho_B \right) \right] \\ &= C_0 + C_1 \cos(\theta_j) + C_2 \sin(\theta_j) \end{aligned} \quad (2.51)$$

where C_0 , C_1 , and C_2 are real numbers that do not depend on the parameter θ_j , defined as

$$\begin{aligned} C_0 &= \frac{\text{Tr} \left[O_A \left(\rho_B + P_j \rho_B P_j^\dagger \right) \right]}{2}, & C_1 &= \frac{\text{Tr} \left[O_A \left(\rho_B - P_j \rho_B P_j^\dagger \right) \right]}{2}, \\ C_2 &= \frac{\text{Tr} \left[O_A \left(\rho_B - P_j \rho_B P_j^\dagger \right) \right]}{2}, \end{aligned} \quad (2.52)$$

Equation (2.51) is particularly interesting because it shows that the cost function is essentially a trigonometric polynomial with respect to each individual variational parameter. Then, using the following identities for the derivatives of sine and cosine functions

$$\begin{aligned} \frac{d \cos x}{dx} &= \frac{\cos(x+s) - \cos(x-s)}{2 \sin s} \\ \frac{d \sin x}{dx} &= \frac{\sin(x+s) - \sin(x-s)}{2 \sin s} \end{aligned} \quad \forall s \neq m\pi, m \in \mathbb{Z}, \quad (2.53)$$

the derivative of the cost in Eq. (2.51) with respect to θ_j can be written as

$$\begin{aligned} \frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} &= C_1 \frac{d \cos \theta_j}{d \theta_j} + C_2 \frac{d \sin \theta_j}{d \theta_j} \\ &= \frac{C_0 + C_1 \cos(\theta_j + s) + C_2 \sin(\theta_j + s)}{2 \sin s} - \frac{C_0 + C_1 \cos(\theta_j - s) + C_2 \sin(\theta_j - s)}{2 \sin s} \\ &= \frac{1}{2 \sin s} [C(\boldsymbol{\theta} + s \mathbf{e}_j) - C(\boldsymbol{\theta} - s \mathbf{e}_j)], \end{aligned} \quad (2.54)$$

where in the last line we recognised that both numerators are instances of the cost (2.51) with an appropriate redefinition of the variational parameter. By setting $s = \pi/2$ in the expression above, one eventually arrives at the usual formulation of the parameter-shift rule [56, 209, 270]

$$\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{2} \left[C\left(\boldsymbol{\theta} + \frac{\pi}{2} \mathbf{e}_j\right) - C\left(\boldsymbol{\theta} - \frac{\pi}{2} \mathbf{e}_j\right) \right], \quad (2.55)$$

¹¹If the same parameter appears more than once in the circuit, one can check that the full derivative is given by shifting independently each parameterised gate, and then summing all contributions [270].

As argued earlier, the parameter-shift rule (2.55) is indeed very similar to the finite-difference formula (2.46), with the big difference that while the former is an *exact* relation between the derivative of a function and its value at specific points, the latter is only an *approximation*.

In general, a parameter-shift rule (2.55) can be derived when the parameterised operation is of the form (2.47) and the generator of the rotation has only two unique eigenvalues, which is related to the requirement that the generator is involutory [68, 270]. Whenever these conditions are not met, one can use generalisations of the parameter-shift rules that overcome the issue for example by decomposing the parameterised gate as a product of rotation-like operations [68], or expanding the generator of the unitary evolution in the Pauli basis and evaluating the gradients of each component with a stochastic approach [19].

The parameter-shift rule is a useful tool because it gives a precise recipe on how to calculate gradients of variational circuits directly on real quantum hardware, simply by composing the result of two appropriately defined quantum circuits. For example, a single-step of the gradient descent algorithms in Eq. (2.42) requires measuring the outcome of $2p$ different circuits, where p is the number of parameters $\boldsymbol{\theta} \in \mathbb{R}^p$. While such linear scaling of resources is not bad per se, this procedure can still be daunting on near-term quantum devices with limited access, especially when the number of parameters is large and many iterations are required to reach a good solution. For example, suppose that the optimisation of a variational circuit with p parameters requires T applications of the gradient-descent update rule, and that each expectation value is estimated on quantum hardware with M measurements shots, then the total number of circuit executions on the quantum hardware scales like $\mathcal{O}(pTM)$, which can be quite big already for modest instances ($p \sim T \sim 10^2$, $M \sim 10^3$). As a comparison, gradients in classical machine learning algorithms can be calculated much more efficiently, within a single computational step, via automatic differentiation techniques and backpropagation. This is done by storing intermediate values of the computation while it takes place, and then combining these values at the end via the chain rule to calculate derivatives of composed functions. Unfortunately, since it is not possible to measure intermediate states in a quantum computation without disturbing the system, there is no straightforward way of applying backpropagation to variational quantum algorithms.

Finally, note that gradients of parameterised quantum circuits can also be calculated with an ancilla-based measurement scheme similar to a Hadamard test [270, 301].

2.2.3.2 Higher order derivatives

The parameter shift rule can be applied iteratively to calculate also higher-order derivatives of the cost function [56, 198]. For example, second mixed derivatives read

$$\begin{aligned} \frac{\partial^2 C(\boldsymbol{\theta})}{\partial \theta_j \partial \theta_i} &= \frac{1}{2} \left[\frac{\partial}{\partial \theta_j} C\left(\boldsymbol{\theta} + \frac{\pi}{2} \mathbf{e}_i\right) - \frac{\partial}{\partial \theta_j} C\left(\boldsymbol{\theta} - \frac{\pi}{2} \mathbf{e}_i\right) \right] \\ &= \frac{1}{4} \left[C\left(\boldsymbol{\theta} + \frac{\pi}{2} \mathbf{e}_i + \frac{\pi}{2} \mathbf{e}_j\right) - C\left(\boldsymbol{\theta} + \frac{\pi}{2} \mathbf{e}_i - \frac{\pi}{2} \mathbf{e}_j\right) \right. \\ &\quad \left. - C\left(\boldsymbol{\theta} - \frac{\pi}{2} \mathbf{e}_i + \frac{\pi}{2} \mathbf{e}_j\right) + C\left(\boldsymbol{\theta} - \frac{\pi}{2} \mathbf{e}_i - \frac{\pi}{2} \mathbf{e}_j\right) \right], \end{aligned} \quad (2.56)$$

which assume the rather simple form for diagonal elements $i = j$

$$\frac{\partial^2 C(\boldsymbol{\theta})}{\partial \theta_i^2} = \frac{1}{2} [C(\boldsymbol{\theta} + \pi \mathbf{e}_i) - C(\boldsymbol{\theta})], \quad (2.57)$$

since one can easily check that $C(\boldsymbol{\theta} + \pi \mathbf{e}_i) = C(\boldsymbol{\theta} - \pi \mathbf{e}_i)$, which is due to the 2π -periodicity of parameterised gates (2.47).

In general, any derivative of a parameterised quantum circuit for which the parameter shift

holds, can be expressed as a linear combination of circuit executions

$$\frac{\partial^{\alpha_1+\dots+\alpha_M} C(\boldsymbol{\theta})}{\partial \theta_1^{\alpha_1} \dots \partial \theta_M^{\alpha_M}} = \frac{1}{2^{\alpha_1+\dots+\alpha_M}} \sum_{m=1}^{2^{\alpha_1+\dots+\alpha_M}} s_m C(\tilde{\boldsymbol{\theta}}_m), \quad (2.58)$$

where $s_m \in \{\pm 1\}$ are signs, and $\tilde{\boldsymbol{\theta}}_m$ are parameters obtained by accumulating shifts along different directions.

2.2.3.3 Discussion

The easy access to the derivatives of the cost function provided by the parameter-shift rule opens up the possibility of using a plethora of gradient-based optimisation methods, like vanilla gradient descent, BFGS or ADAM, but these are not the only viable route. In fact, also zeroth-order methods like Nelder-Mead [218] and COBYLA [235] have been widely used in the literature to optimise parameterised quantum circuits [29, 301], as well as more quantum-native approaches developed specifically for variational algorithms. Examples are the Rotosolve algorithm [222] that offers an analytic solution to the optimisation problem, or the Quantum Natural Gradient (QNG) method, that, inspired by its classical counterpart, proposes an update of the parameters that takes into account the metric of the space of the quantum states created by the parameterised quantum model, rather than the euclidean metric of parameter space [290].

With every approach having its own perks and disadvantages, no consensus has been reached at the moment regarding the best practices for optimising variational quantum algorithms, and the performances are usually studied on a case-by-case basis.

2.2.4 Barren plateaus and unitary designs

In the previous section, we discussed how to optimise variational circuits, but now we discuss why this procedure may turn out to be a particularly arduous task in practice. Indeed, various theoretical challenges hinder the optimisation of variational algorithms, a phenomenon dubbed in the literature as the *barren plateau* (BP) problem. At the current state of the art, by barren plateaus one usually refers to a number of different scenarios where it can be shown that variational quantum algorithms suffer from trainability issues related to vanishing gradients and very flat optimisation landscapes, hence their name.

The emergence of barren plateaus was first observed in [202], where the occurrence of vanishing gradients was related to the way parameterised quantum ansätze behave in the large qubit regime upon assignment of random parameters, as they resemble high-dimensional random unitary matrices. A large set of results in measure theory in high-dimensional spaces underline the phenomenon of concentration of measure, an example being Levy's lemma, according to which smooth functions on these spaces strongly concentrate around their mean values, and are essentially constant on the space [180]. Since the average value of gradients of parameterised circuits is often found to be zero, then by concentration of measure they are zero almost always, for most random choices of the parameters, hence any optimisation method will likely fail to find any interesting minimising direction. Formally, given a PQC acting on a system of n qubits, it can be theoretically argued and numerically confirmed that [57, 137, 201]

$$\mathbb{E}_{\boldsymbol{\theta}} \left[\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_k} \right] = 0, \quad (2.59)$$

$$\text{Var}_{\boldsymbol{\theta}} \left[\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_k} \right] \in \mathcal{O}(b^{-n}), \quad b > 0. \quad (2.60)$$

That is, upon random assignment of the variational parameters, the expectation value of the gradients of the cost function is zero, and their variance is exponentially vanishing with the number of qubits. Then, as the number of qubits grows large, the gradients become exponentially small

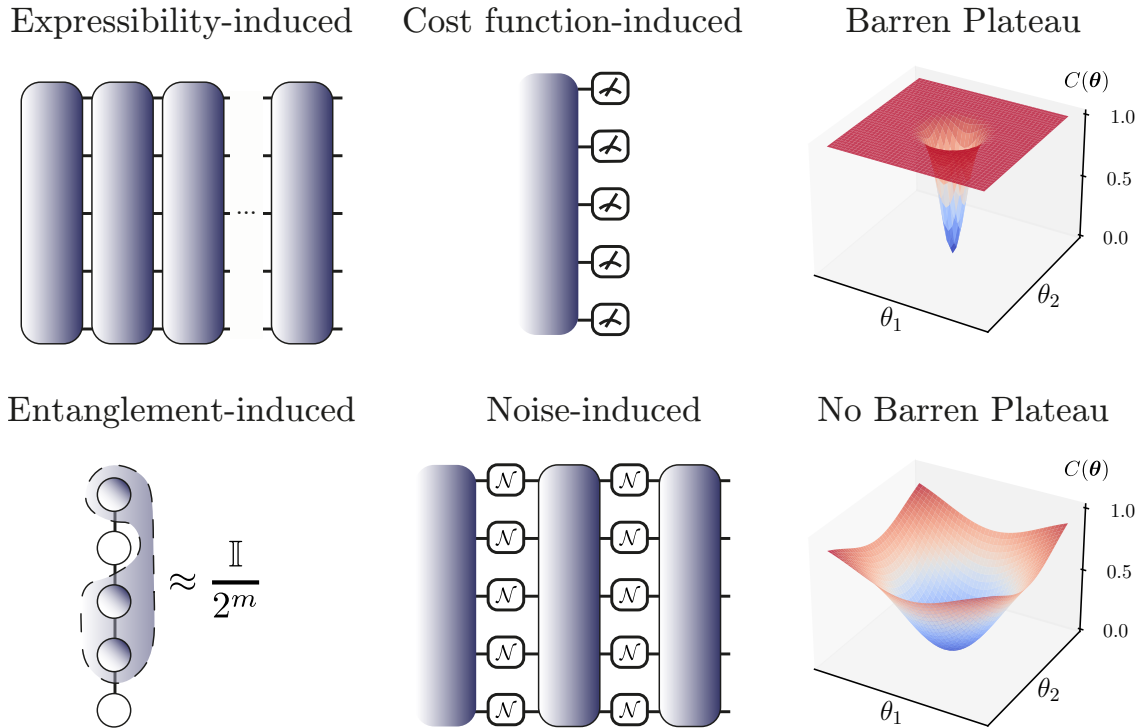


Figure 2.4: Summary of the sources of Barren Plateaus (BP) in the optimisation landscape of variational quantum algorithms. Trainability issues arise in variational algorithms whenever the circuit is too deep and expressive, global observables are used in the cost function, too much entanglement is created inside the circuit, or too much noise is present in the circuit. In these cases, as the number of qubits grows large, the optimisation landscape becomes exponentially flat almost everywhere, hence the name barren plateau.

on average, and thus one would need an exponential number of measurement shots to reliably estimate their value on real hardware, for example via parameter-shift rules. Such impossibility of properly training parameterised circuits clearly hampers the applicability of any quantum variational procedure suffering from a barren plateau (concentration of measure) phenomenon.

Trainability issues in variational quantum algorithms have been studied extensively in the literature, and the emergence of barren plateaus in the optimisation landscape parameterised circuits has been found in various settings, that can be grouped into four major classes [300]¹²:

1. **Expressibility-induced BP** [13, 202]: this is the case explained above, where the untrainability stems from the randomness induced by deep random ansätze initialised with random parameters. This causes an exponential flattening of the loss landscape and its gradients due to concentration of measure.
2. **Cost function-induced BP** [57]: the use of a global observable, that is, an observable acting nontrivially on all qubits in the circuit, in the cost function (2.39), is also linked to vanishing gradients. Intuitively, this can be understood as a consequence of measuring quantities in an exponentially big Hilbert space, which effectively results in a wash-out of information. Most importantly, while expressibility-induced barren plateaus are obtained only for deep-enough (or better, random-enough) ansätze, the use of a global observable will likely result in exponentially vanishing gradients irrespectively of the depth of the circuit, and a barren plateau will occur even at shallow depths $L \in \mathcal{O}(1)$, where L is the number of layers in the

¹²As briefly mentioned here when discussing Entanglement-induced BP and then carefully explained in Ch. 7, the sources of barren plateaus can be reduced to three, not four, since Expressibility-induced BP and Entanglement-induced BP are intimately connected to each other through randomness.

ansatz (2.40).

On the contrary, it was shown that a layered ansatz leveraging local operations on neighbouring qubits, when used with a local cost function, that is a loss defined in terms of expectation values of observables acting non-trivially only on a small subset –one or two– of qubits, can be trainable at least for shallow depths $L \in \mathcal{O}(\log n)$, where n is the number of qubits.

For completeness, in Appendix A.1 we report the toy model proposed in ref. [57] to show the occurrence of barren plateaus even for depth-one circuits when a global cost is used, and how the issue can be mitigated by leveraging an equivalent local version of the cost.

3. **Entanglement-induced BP** [221]: it was also shown the creation of large entanglement inside a circuit can be detrimental to variational algorithms. As for the expressibility-induced case, also entanglement-induced BP arise as a consequence of the behaviour of random quantum states and matrices. Essentially, whenever the ansatz is expressible enough to create highly-entangled states, then any reduced density operator on m qubits will be very close to the maximally mixed state

$$\rho = \text{Tr}_{n-m}[|\psi_{\theta}\rangle\langle\psi_{\theta}|] \approx \frac{\mathbb{I}}{2^m}, \quad |\psi_{\theta}\rangle = U(\theta)|\mathbf{0}\rangle\langle\mathbf{0}|U(\theta)^\dagger \text{ highly entangled}, \quad (2.61)$$

and thus little information can be retrieved from measurements of observables on reduced systems. This source of barren plateaus motivates the study performed in Chapter 7, where the production of entanglement in common variational quantum circuits is studied. In particular, we anticipate that entanglement-induced BP and expressibility-induced BP both stem from the resemblance of the parameterised quantum circuit to random unitary matrices, and these two sources are indeed one and the same [258], as extensively discussed in Chapter 7.

4. **Noise-induced BP** [312]: noise that occurs in the quantum circuit can be on its own the cause of the emergence of flat loss landscapes and barren plateaus. Indeed, it was shown that local Pauli errors happening throughout the circuit perturb the quantum state by moving it close to the fixed point of the noise map¹³, that is the maximally mixed state. By this mechanism, parameterised circuits whose depth scales linearly with the number of qubits $L \in \mathcal{O}(n)$ will again suffer from barren plateaus.

It is interesting to notice that this class of barren plateaus is conceptually different from all the cases treated previously, since the former are generally probabilistic statements regarding the expectation value of the gradients and their variance, which is exponentially suppressed, when random initialisation of the parameters are considered. The latter instead is not a probabilistic statement on the variance of the gradient, but rather is the gradient itself that is vanishing, due to the loss landscape becoming essentially flat *everywhere* in parameter space. On the contrary, in the previous cases the loss landscape is flat *almost* everywhere, in fact there are exceptional optimal points (called in the literature *narrow gorges*) where the loss is very steep and optimisation is possible [11, 57].

In the following, we show how to derive Equations (2.60) for the case of expressibility-induced barren plateaus, as they are the most studied and common source of trainability issues in variational algorithms, and are also the most amenable to a concise exposition. The analysis of the problem requires knowledge of random unitary matrices, which we briefly introduce below. Then, after showing how to use these results to prove the emergence of barren plateaus, we discuss how this issue is related to the concept of *expressibility* of quantum circuits, which is a measure of how well a parameterised ansatz addresses the full unitary space.

¹³The fixed point of a noise channel $\mathcal{E}[\rho]$ is the state that is unchanged after application of the map, namely $\mathcal{E}[\rho] = \rho$. It is interesting to notice that every quantum noise channel has a fixed point [220]. The definition of noise channels and their properties can be found in Chapter 8.

2.2.4.1 Haar measure and random unitary matrices

Expressibility-induced barren plateaus arise whenever a parameterised quantum circuit resembles a random unitary matrix when random parameters are assigned to the circuit. Equations (2.60) are then derived by characterising the statistics of random unitaries, and studying what happens when they are used inside a variational optimisation routine. In order to evaluate average values, namely integrals, of functions of random unitary matrices, one requires a probability measure that weights the elements of the unitary group and that is used inside the integrals. The most natural distribution one can think of is the *uniform* distribution, and the *Haar measure* is the mathematical tool that formally defines a uniform probability measure on compact groups [204].

Let $\mathcal{U}(d)$ be the group of $d \times d$ unitary matrices, the Haar measure $d\mu(U)$ on the group $\mathcal{U}(d)$ is the *unique* translationally invariant measure, i.e. for any integrable function $f(\cdot)$ and unitary matrix $V \in \mathcal{U}(d)$ it holds

$$\int d\mu(U)f(VU) = \int d\mu(U)f(UV) = \int d\mu(U)f(U), \quad \forall V \in \mathcal{U}(d). \quad (2.62)$$

The property of translational invariance, which is also referred to as right- and left-invariance, encodes the fact that the Haar measure represents the uniform measure of the group. Thus, we say that a unitary matrix is Haar distributed to indicate a unitary matrix that is sampled uniformly from the space of unitary matrices. In addition, the volume element is normalised, namely

$$\int d\mu(U) = 1. \quad (2.63)$$

We will use the following simplified notation to indicate expectation values and integration over Haar-distributed random unitary matrices

$$\mathbb{E}_U[f(U)] := \int d\mu(U)f(U). \quad (2.64)$$

Explicit formulas exist for evaluating integrals over random unitary matrices [104, 238], and here we recall two useful identities regarding the expectation values for low degree polynomials of random unitary matrices [57, 137, 141, 202]

$$\mathbb{E}_U[UAU^\dagger] = \int d\mu(U)UAU^\dagger = \frac{\text{Tr}[A]\mathbb{I}}{d} \quad (2.65)$$

$$\begin{aligned} \mathbb{E}_U[AUBU^\dagger CUDU^\dagger] &= \int d\mu(U)AUBU^\dagger CUDU^\dagger \\ &= \frac{\text{Tr}[BD]\text{Tr}[C]A + \text{Tr}[B]\text{Tr}[D]AC}{d^2 - 1} \\ &\quad - \frac{\text{Tr}[BD]AC + \text{Tr}[B]\text{Tr}[C]\text{Tr}[D]A}{d(d^2 - 1)} \end{aligned} \quad (2.66)$$

$$\begin{aligned} \mathbb{E}_U[\text{Tr}[UAU^\dagger B]\text{Tr}[UCU^\dagger D]] &= \int d\mu(U)\text{Tr}[UAU^\dagger B]\text{Tr}[UCU^\dagger D] \\ &= \frac{\text{Tr}[A]\text{Tr}[B]\text{Tr}[C]\text{Tr}[D] + \text{Tr}[AC]\text{Tr}[BD]}{d^2 - 1} \\ &\quad - \frac{\text{Tr}[AC]\text{Tr}[B]\text{Tr}[D] + \text{Tr}[A]\text{Tr}[C]\text{Tr}[BD]}{d(d^2 - 1)} \end{aligned} \quad (2.67)$$

Here A, B, C and D are unitary matrices in $\mathcal{U}(d)$, where $d = 2^n$ for matrices acting on a system of n qubits.

Strictly related to uniform random matrices, is the concept of unitary designs, which are ensembles of unitaries that replicate properties of the Haar distribution up to a given moment.

Formally, let $\mathbb{U} = \{U_1, \dots, U_K\}$ be an ensemble of unitary matrices $U_i \in \mathcal{U}(d)$, and $P_{t,t}(U_i)$ a polynomial of degree at most t in the entries of U_i , and degree at most t in the entries of U_i^\dagger . Then, the ensemble \mathbb{U} is said to be a unitary t -design if [80]

$$\mathbb{E}_{\mathbb{U}}[P_{t,t}(U)] := \frac{1}{K} \sum_{i=1}^K P_{t,t}(U_i) = \int d\mu(U) P_{t,t}(U), \quad (2.68)$$

that is, averaging over the discrete set $\mathbb{U} = \{U_1, \dots, U_K\}$ is *equivalent* to averaging over the full unitary group with respect to the Haar measure. Intuitively, unitary designs are sets of operators that are random enough to match moments of the Haar distribution up to a given degree.

In particular, looking back at Eqs. (2.65) and (2.66), one can obtain the same expectation values by averaging over a unitary 2-design instead of considering the full unitary group. This is by definition of unitary 2-design, and noticing that both UAU and $AUBU^\dagger CUDU^\dagger$ are polynomials $P_{2,2}(U)$ of degree of most $t = 2$ in the entries of U and U^\dagger .

Examples of unitary designs are the Pauli group, defined as the set of all possible tensor products of Pauli matrices, which is a unitary 1-design, and the Clifford group, defined as the set of operators that normalises the Pauli group, that can be shown to be a unitary 3-design (hence also a 1- and 2-design) [316]. The Pauli group P_n on n qubits is defined as all possible combinations of Pauli matrices with coefficients ± 1 and $\pm i$

$$P_n := \{\pm 1, \pm i\} \times \{\sigma_1 \otimes \sigma_2 \otimes \dots \otimes \sigma_n \mid \sigma_i \in \{\mathbb{I}, X, Y, Z\}\}, \quad (2.69)$$

while the Clifford group \mathcal{C}_n consists of set of unitary operators that maps the Pauli group to the Pauli group under conjugation (and up to a phase), namely

$$\mathcal{C}_n := \{U \in \mathcal{U}(2^n) \mid \forall \sigma \in P_n \implies U\sigma U^\dagger \in P_n\} / \mathcal{U}(1). \quad (2.70)$$

2.2.4.2 Barren plateaus in the optimisation of PQCs

The theory of random unitary matrices, and specifically unitary 2-designs, is at the core of the barren plateau phenomenon, as it will be now clear.

Let $U(\boldsymbol{\theta}) \in \mathcal{U}(2^n)$ a parameterised quantum circuit with variational parameters $\boldsymbol{\theta} \in \mathbb{R}^P$ acting on a system of n qubits, and $C(\boldsymbol{\theta}) = \text{Tr}[OU(\boldsymbol{\theta})\rho U(\boldsymbol{\theta})^\dagger]$ a loss function to be minimised. As done previously for deriving the parameter-shift rule (2.49), given a parameter $\theta_k \in \boldsymbol{\theta} = (\theta_1, \dots, \theta_P)$, consider a bipartition of the circuit happening at the position of the parameterised gate depending on the parameter θ_i

$$U(\boldsymbol{\theta}) = U_A V_k(\theta_k) U_B, \quad (2.71)$$

where $V_k(\theta_k) = e^{-i\theta_k P_k/2}$ is again a parameterised Pauli rotation, and for simplicity of notation we suppressed the dependence of U_R and U_L , on the remaining variational parameters $U_{R,L} = U_{R,L}(\theta_1, \dots, \theta_{k-1}, \theta_{k+1}, \dots, \theta_P)$. The derivative of the unitary $U(\boldsymbol{\theta})$ with respect to θ_k then amounts to

$$\partial_k U(\boldsymbol{\theta}) = U_A \partial_k \left(e^{-i\theta_k P_k/2} \right) U_B = -\frac{i}{2} U_A P_k U_B, \quad (2.72)$$

and similarly for $U(\boldsymbol{\theta})^\dagger$, $\partial_k U(\boldsymbol{\theta})^\dagger = iU_L^\dagger P_k U_R^\dagger/2$. Then, the derivative of the cost with respect to parameter θ_k can be written as

$$\partial_k C(\boldsymbol{\theta}) = \text{Tr}[O(\partial_k U(\boldsymbol{\theta}))\rho U(\boldsymbol{\theta})^\dagger] + \text{Tr}[OU(\boldsymbol{\theta})\rho(\partial_k U(\boldsymbol{\theta})^\dagger)] \quad (2.73)$$

$$= -\frac{i}{2} \text{Tr}\left[OU_A P_k U_B \rho U_B^\dagger U_A^\dagger\right] + \frac{i}{2} \text{Tr}\left[OU_A U_B \rho U_B^\dagger P_k U_A^\dagger\right] \quad (2.74)$$

$$= -\frac{i}{2} \text{Tr}\left[U_A^\dagger O U_A \left[P_k, U_B \rho U_B^\dagger\right]\right] = -\frac{i}{2} \text{Tr}[O_A [P_k, \rho_B]], \quad (2.75)$$

where again we have defined the evolved observable and state $O_A = U_A^\dagger O U_A$ and $\rho_B = U_B \rho U_B^\dagger$, respectively. Note that this is an alternative expression for the derivative of the cost function, as opposed to the one derived above when discussing the parameterised-shift role. One can check that all the calculations that follow can be applied identically if using the expression (2.55).

Suppose now that one of the parameterised circuits, either U_A or U_B , is complex enough so that the unitaries generated by assigning random parameters to the circuit closely resemble random unitary matrices. More formally, given a set of parameter vectors $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \dots, \boldsymbol{\theta}_K\}$, suppose that corresponding set of unitaries $\mathbb{U}_{A,B} = \{U_{A,B}(\boldsymbol{\theta}_1), \dots, U_{A,B}(\boldsymbol{\theta}_K)\} = \{U_{A,B}^{(1)}, \dots, U_{A,B}^{(K)}\}$ form a 1-design, as defined in (2.68). Then, the expected value of the derivative $\partial_k C(\boldsymbol{\theta})$ over a random assignment of the parameter vector can be evaluated as follows

$$\mathbb{E}_{\mathbb{U}_A}[\partial_k C(\boldsymbol{\theta})] = -\frac{i}{2} \mathbb{E}_{\mathbb{U}_A}[\text{Tr}[O_A [P_k, \rho_B]]] = -\frac{i}{2} \text{Tr}[\mathbb{E}_{\mathbb{U}_A}[U_A^\dagger O U_A] [P_k, \rho_B]] \quad (2.76)$$

$$= -\frac{i}{2} \text{Tr}\left[\frac{\mathbb{I} \text{Tr}[O]}{2^n} [P_k, \rho_B]\right] = -\frac{i \text{Tr}[O]}{2^{n+1}} \text{Tr}[P_k \rho_B - \rho_B P_k] = 0, \quad (2.77)$$

where in the first line we exchanged the trace and the expectation value since they are both linear operations, and in the second line we first made use of the formula for first moments randomly distributed unitary matrices (2.65), and then used the fact that the trace of a commutator is zero, due to cyclicity of the trace. The same result is obtained if U_B is a 1-design instead, in fact

$$\mathbb{E}_{\mathbb{U}_B}[\partial_k C(\boldsymbol{\theta})] = -\frac{i}{2} \text{Tr}\left[O_A [P_k, \mathbb{E}_{\mathbb{U}_B}[U_B \rho U_B^\dagger]]\right] = -\frac{i}{2} \text{Tr}\left[O_A \left[P_k, \frac{\mathbb{I} \text{Tr}[\rho]}{2^n}\right]\right] = 0 \quad (2.78)$$

where the last equality follows from the vanishing commutator $[P_k, \mathbb{I}] = 0$. Thus, one can summarise Eqs. (2.76) and (2.78) as follows.

Theorem 2.1 — Zero average gradient for random PQCs. Let $U(\boldsymbol{\theta})$ represent a parameterised quantum circuit with variational parameters $\boldsymbol{\theta} \in \mathbb{R}^M$ acting on a system of n qubits, and let $C(\boldsymbol{\theta}) = \text{Tr}[O U(\boldsymbol{\theta}) \rho U(\boldsymbol{\theta})^\dagger]$ be a cost function depending on the variational parameters via gates of the form $V_k(\theta_k) = e^{-i\theta_k P_k/2}$. For any parameter θ_k , $k = 1, \dots, M$, consider the bipartition of the circuit $U(\boldsymbol{\theta}) = U_A(\boldsymbol{\theta}) V_k(\theta_k) U_B(\boldsymbol{\theta})$, and let $\mathbb{U}_{A,B} = \{U_{A,B}(\boldsymbol{\theta}_1), \dots, U_{A,B}(\boldsymbol{\theta}_K)\}$ denote the set of unitaries generated when some randomly generated parameters $\{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K\}$ are assigned to the circuit sections $U_{A,B}$. Then, if at least one of \mathbb{U}_A or \mathbb{U}_B forms a 1-design, the expected value of any partial derivative of the cost function when random parameters are assigned to the quantum circuit is zero

$$\mathbb{E}_{\mathbb{U}_{A,B}} \left[\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_k} \right] = 0 \quad \forall k, \quad \text{if either } \mathbb{U}_A, \text{ or } \mathbb{U}_B, \text{ or both, are at least a 1-design.} \quad (2.79)$$

Note that no particular assumptions are made on the actual probability distribution used to sample the parameters θ_k , the only requirement is that the corresponding set of unitaries $\mathbb{U}_{A,B}$ form 1-designs. In practical scenarios, it is common practice to initialise parameterised quantum circuits with random parameters uniformly distributed in $\theta_k \sim \text{Unif}[0, 2\pi]$. In this case, one can also use the trigonometric nature of the cost function shown before (2.55) to show that derivatives are indeed biased towards zero

$$\mathbb{E}_{\theta_k \sim \text{Unif}[0, 2\pi]}[\partial_k C(\boldsymbol{\theta})] = \int_0^{2\pi} d\theta_k \partial_k (C_0 + C_1 \cos \theta_k + C_2 \sin \theta_k) \quad (2.80)$$

$$= \int_0^{2\pi} (-C_1 \sin \theta_k + C_2 \cos \theta_k) d\theta_k = 0. \quad (2.81)$$

So far, we have reproduced the first result of the barren plateau phenomenon described in (2.60), and now we move on to estimating the variance of the cost function derivatives when random

parameters are assigned to the circuit. Since the variance is a function of second degree with respect to the circuit, the calculations are more involved, as they entail computing expectation values over 2-designs. We are now interested in studying the following quantity

$$\text{Var}[\partial_k C(\boldsymbol{\theta})] := \mathbb{E}[(\partial_k C(\boldsymbol{\theta}))^2] - \mathbb{E}[\partial_k C(\boldsymbol{\theta})]^2 = \mathbb{E}[(\partial_k C(\boldsymbol{\theta}))^2] \quad (2.82)$$

$$= -\frac{1}{4} \mathbb{E} \left[\text{Tr} \left[U_A^\dagger O U_A \left[P_k, U_B \rho U_B^\dagger \right] \right]^2 \right], \quad (2.83)$$

which can be computed using Eq. (2.66) and (2.67) for second degree polynomials of random unitary matrices, assuming that the ensembles $\mathbb{U}_{A,B}$ are now random enough to be also 2-designs. We report the full calculation in Appendix A.2, and hereby state only the final result.

Theorem 2.2 — Vanishing gradients for random PQCs. In the same conditions of Th. 2.1, but assuming that the ensembles of unitaries $\mathbb{U}_{A,B}$ are now random enough to be also 2-designs, then the following holds:

If \mathbb{U}_A is at least a 2-design, then $\forall k$:

$$\text{Var}_{\mathbb{U}_A}[\partial_k C(\boldsymbol{\theta})] = -\frac{1}{4} \frac{1}{2^{2n}-1} \left(\text{Tr}[O^2] - \frac{\text{Tr}[O]^2}{2^n} \right) \text{Tr} \left[\left[P_k, U_B \rho U_B^\dagger \right]^2 \right] \quad (2.84)$$

If \mathbb{U}_B is at least a 2-design, then $\forall k$:

$$\text{Var}_{\mathbb{U}_B}[\partial_k C(\boldsymbol{\theta})] = -\frac{1}{4} \frac{1}{2^{2n}-1} \left(\text{Tr}[\rho^2] - \frac{1}{2^n} \right) \text{Tr} \left[\left[U_A^\dagger O U_A, P_k \right]^2 \right] \quad (2.85)$$

If $\mathbb{U}_{A,B}$ are both at least 2-designs, then $\forall k$:

$$\begin{aligned} \text{Var}_{\mathbb{U}_{A,B}}[\partial_k C(\boldsymbol{\theta})] = & -\frac{1}{4} \frac{2}{2^{2n}-1} \left(\text{Tr}[\rho^2] - \frac{1}{2^n} \right) \left(\frac{\text{Tr}[O^2] \text{Tr}[P_k]^2 + \text{Tr}[O]^2 \text{Tr}[P_k^2]}{2^{2n}-1} \right. \\ & \left. - \frac{\text{Tr}[O^2] \text{Tr}[P_k^2] + \text{Tr}[O]^2 \text{Tr}[P_k]^2}{2^n(2^{2n}-1)} - \frac{\text{Tr}[O^2] \text{Tr}[P_k^2]}{2^n} \right) \end{aligned} \quad (2.86)$$

In practical instances when for example the parameterised gates are generated by Pauli rotations P_k , the measured observable O is a Pauli string, and the input state is a pure state $\rho = |0\rangle\langle 0|$, then the variance vanish exponentially with the number of qubits

$$\text{Var}[\partial_k C(\boldsymbol{\theta})] \in \mathcal{O}(2^{-n}). \quad (2.87)$$

As shown in the Appendix, despite the minus sign in the expressions above, one can check that the variances are correctly positive values.

By combining the results of Theorem 2.1 and Theorem 2.2 together with Chebyshev's inequality, one eventually arrives at

$$\Pr(|\partial_k C(\boldsymbol{\theta})| > \delta) \leq \frac{\text{Var}[\partial_k C(\boldsymbol{\theta})]}{\delta^2} \in \mathcal{O}(2^{-n}) \quad (2.88)$$

which states that the probability of having a non-negligible gradient in a random parameterised quantum circuit vanishes exponentially with the number of qubits. Thus, as it is not possible to determine any interesting minimising direction, it is not possible to train variational circuits efficiently in the large qubit regime $n \gg 1$, where one expects to find a quantum advantage of sort.

2.2.4.3 Discussion and mitigation of barren plateaus

The emergence of randomness-induced barren plateaus is tightly connected to the concentration on measure phenomenon in high-dimensional spaces, and indeed it was clearly shown that vanishing

gradients (i.e. barren plateaus) are the flip side of the exponential concentration of the cost function around the mean [13], which makes the loss landscape flat almost everywhere, except for so-called *narrow gorges* corresponding to the minimum of the function. If, by chance, one initialises the parameters of the quantum circuit inside or close to a narrow gorge, then optimisation is possible. Moreover, note that while barren plateaus are usually defined in terms of vanishing derivatives, gradient-free optimisation methods are not a solution to the problem [12].

Several mitigation strategies have been proposed to alleviate trainability issues related to barren plateaus, for example initialising the parameters in the quantum circuit such that the circuits initially acts as an identity [115], correlating parameters inside the quantum circuit [309], using heuristics to initialise parameters already close to an optimal solution [331], leveraging classical algorithms based on tensor-network pre-training [257] or recurrent neural networks [305] to propose good values for the parameters, optimise the parameters procedurally in a layer-wise fashion [283], opting for local—instead of global—cost functions [57], and eventually restricting the expressibility of the circuit ansatz by carefully choosing problem-inspired ansätze [173, 208, 284].

While all these strategies may seem unrelated to each other, most of them work by imposing some type of constraint on the parameterised quantum circuit, so that it is far from being a general random circuit and resembling a unitary design. Indeed, while trainability issues may arise as a consequence of many different factors (see Fig. 2.4), all of these sources can be tamed by appropriately controlling the expressibility of the circuit, which comes at the cost of either using very specific problem-inspired ansätze, or strongly limiting the depth of the quantum circuit to include only logarithmically many operations $L \in \mathcal{O}(\log n)$ [57, 66, 231]. The use of shallow circuits does not only avoid randomness-induced barren plateaus, but also noise-induced ones, as errors cannot accumulate and grow to the extent of causing a flattening of the loss landscape.

2.2.5 Expressibility of PQCs

Up until now we frequently used the term *expressibility* of parameterised quantum circuits to intuitively convey a measure of their ability to represent a general (random) unitary matrix. This statement can be made formal by defining the expressibility with the superoperator [137, 281]

$$A_{\mathbb{U}}^{(t)}(\cdot) := \int_{\text{Haar}} d\mu(V) V^{\otimes t}(\cdot)(V^\dagger)^{\otimes t} - \int_{\mathbb{U}} dU U^{\otimes t}(\cdot)(U^\dagger)^{\otimes t} \quad (2.89)$$

where the first integral is evaluated over the Haar distribution on the unitary group $\mathcal{U}(d)$, and the second one is over the uniform distribution on the ensemble of unitaries \mathbb{U} to be characterised. Small values of $A_{\mathbb{U}}^{(t)}(\cdot)$ means high expressivity. Indeed, if the ensemble \mathbb{U} is a t -design, then $A_{\mathbb{U}}^{(t)}(X) = 0$ will be zero for all operators X . The expressibility then measures the “power” of an ensemble of unitaries \mathbb{U} in terms of its generality, that is measuring how faithful it is at reproducing the same statistical moments of the Haar distribution.

With this definition, it is possible to derive a formal connection between the expressibility and the barren plateau phenomenon described previously. In fact, authors in [137] provided a generalisation of the randomness-induced barren plateau phenomenon, by extending its validity to circuits forming *approximate* rather than *exact* 2-designs. In particular, it was shown that the variance of gradients of an arbitrary ansatz can be upper bounded by

$$\begin{aligned} \text{Var}[\partial_k C(\boldsymbol{\theta})] &\leq \text{Var}_{2\text{-des}}[\partial_k C(\boldsymbol{\theta})] + f(\boldsymbol{\varepsilon}_{\mathbb{U}}^O, \boldsymbol{\varepsilon}_{\mathbb{U}}^\rho) \\ \boldsymbol{\varepsilon}_{\mathbb{U}}^O &:= \left\| A_{\mathbb{U}}^{(2)}(O^{\otimes 2}) \right\|_2, \quad \boldsymbol{\varepsilon}_{\mathbb{U}}^\rho := \left\| A_{\mathbb{U}}^{(2)}(\rho^{\otimes 2}) \right\|_2 \end{aligned} \quad (2.90)$$

where the first term on the right is the variance obtained if the circuit — or better, parts of it as shown in Th. 2.2 — was an exact 2-design, and the second term is a function that depends on the expressibility of the circuit, specifically through some observable (O) and state (ρ) dependent quantities. The notation $\|\cdot\|_2$ denotes the *Frobenius norm* for operators, defined as $\|A\|_2 :=$

$\sqrt{\text{Tr}[A^\dagger A]}$. We refer to [137] for the complete statement of the result, including the explicit form of the function $f(\cdot, \cdot)$.

For our discussion, it is sufficient to note that if the circuit is an exact 2-design, then the expressibility term $f(\boldsymbol{\varepsilon}_U^O, \boldsymbol{\varepsilon}_U^P)$ vanishes and the inequality becomes an equality, thus obtaining again the exponentially vanishing gradients of Theorem 2.2. On the contrary, if the circuit is not very expressible and $f(\boldsymbol{\varepsilon}_U^O, \boldsymbol{\varepsilon}_U^P) \notin \mathcal{O}(2^{-n})$, then the upper bound allows for non-vanishing gradients. Thus, in conclusion, this result implies that while highly expressive ansätze have vanishing gradients and hence are harder to train, imperfectly expressive guarantee a viable solution to restore trainability. This is in line with recent proposals in the literature, already briefly mentioned in Sec. 2.2.2, that advocate for the use of problem-inspired constrained ansätze for variational quantum algorithms.

Finally, we note that while the definitions in Eq. 2.90 turn out useful for theoretical analysis, an alternative formulation in terms of so-called *frame potentials* can be used for numerical evaluations of the expressibility, as discussed in Chapter 7.

2.3 Conclusions

In this chapter, we gave an overview of the state of the art of quantum computing, specifically focused on the class of quantum algorithms that can be run on near-term devices.

We started by recalling the basic building blocks of quantum computation — qubits, gates, and measurement — and then proceeded by describing the current situation on the practical realisation of quantum computing machines. As opposed to ideal universal fault-tolerant quantum computers, current quantum devices are referred to as *Noisy Intermediate-Scale Quantum* (NISQ) computers, to remark that they are imperfect devices which are not error-corrected and are of limited size.

Meanwhile future experimental and theoretical advancements pave the way toward the construction of large-scale noise-resilient quantum computers, near-term devices give the opportunity to experiment with quantum information processing, and also invite researchers to explore a new paradigm for computation based on hybrid quantum-classical algorithms, namely Variational Quantum Algorithms (VQAs).

In the next chapter, we will introduce a sub-field of variational quantum algorithms called Quantum Machine Learning (QML), which has gained significant attention over the past years, and presents itself as one of the most interesting application for near-term devices.



3. Quantum Machine Learning

'Isn't it a shame that with the tremendous amount of work you have done you haven't been able to get any results?' Edison turned on me like a flash, and with a smile replied: 'Results! Why, man, I have gotten a lot of results! I know several thousand things that won't work.'

Thomas Edison, to one of its associate [91]

3.1	Introduction	53
3.1.1	The four-fold way of Quantum Machine Learning	53
3.2	Classical Machine Learning	55
3.2.1	Basics of (supervised) Machine Learning	57
3.2.1.1	Training dataset	57
3.2.1.2	Hypothesis class	57
3.2.1.3	Empirical Risk Minimisation	57
3.2.1.4	Loss functions and Learning	58
3.2.1.5	Generalisation	59
3.2.2	Machine learning models	62
3.2.2.1	Linear models and kernel methods	62
3.2.2.2	Neural Networks	65
3.3	Quantum Machine Learning	67
3.3.1	Linear quantum models: quantum classifiers and kernel methods	68
3.3.1.1	Explicit models	69
3.3.1.2	Quantum kernel (or implicit) models	70
3.3.1.3	Explicit or Implicit?	71
3.3.2	Data reuploading models and Quantum Neural Networks	71
3.3.2.1	Deriving the Fourier expansion	73
3.3.2.2	A single-qubit data reuploading circuit	75
3.3.2.3	Quantum Neural Networks	76
3.3.3	Generalization of QML models	77
3.3.4	The power of quantum machine learning	79

In this chapter, we explore the field of Quantum Machine Learning (QML), one of the leading proposals to achieve a meaningful quantum advantage already with current near-term devices based on variational quantum algorithms. We start by giving a broad overview of the field in the Introduction 3.1, and then proceed by introducing the main elements of classical machine learning in Sec. 3.2. These concepts will be used to present quantum versions of learning models in Sec 3.3, which is dedicated to variational Quantum Machine Learning. In-depth analyses and reviews on various aspects of Quantum Computing and Machine Learning can be found in the following references [15, 25, 31, 52, 59, 63, 82, 87, 88, 192, 266].

3.1 Introduction

So far, we have discussed how the current generation of noisy quantum computers impelled the use of a new paradigm of computation which uses classical and quantum resources in tandem to perform a computation. At the core of variational quantum algorithms is the optimisation process, which tunes the trainable parameters of a parameterised quantum circuit in order to minimise an appropriately chosen cost function.

Needless to say, this approach of optimising — or *training* — a parametric model to solve a complicated problem also accurately describes the field of Machine Learning (ML), especially its latest version called Deep Learning (DL) [112, 178]. Indeed, state of the art Deep Learning prescribes the use of massive parametric models with billions of tunable parameters, a recipe which has shown incredible success over the past decade in a variety of tasks, ranging from controlling nuclear fusion reactors [83] to generating human-level original digital art [247].

The strong connection between Variational Quantum Algorithms and Deep Learning made it possible to borrow many of the concepts from the well-established field of classical machine learning and apply them to variational quantum algorithms, so much so that this field is also often referred to with the more captivating name of “Quantum Machine Learning” (QML) [4, 25, 59], even though a border between the two can be drawn, as discussed later. However, it is important to stress that the field of Quantum Machine Learning has been around since earlier than Variational Quantum Algorithms gained momentum over the last few years [88, 120, 182, 267, 268, 325].

Earlier works at the boundaries of machine learning and quantum physics include quantum algorithms with provable speedups devoted to solving linear algebra tasks relevant for some machine learning models [31, 188, 250]. The interest in these approaches however progressively declined over recent years, giving way to the rise of variational quantum algorithms as best representatives of quantum machine learning. This happened for a variety of reasons, including the lack of good-enough experimental hardware to run these linear algebra-based quantum subroutines — for example, based on the HHL procedure for matrix inversion [125]—, subtleties in the assumptions that hinders the applicability of these algorithms in real cases [1], and, arguably the most important, the discovery that many of these algorithms can be *dequantised*, in that there exists a class of quantum-inspired classical algorithms that has the same runtime complexity [297]. Nonetheless, while dequantisation proved the absence of *exponential speedups* of some quantum algorithms over their classical counterparts, important *polynomial* improvements may still be attained in practical scenarios [14, 67, 88].

3.1.1 The four-fold way of Quantum Machine Learning

The most general definition of Quantum Machine Learning is that a research field whose aim is to investigate the interplay between (classical) Machine Learning (ML) — and more generally, Artificial Intelligence (AI) — and Quantum Computing, with the hope that a fruitful exchange

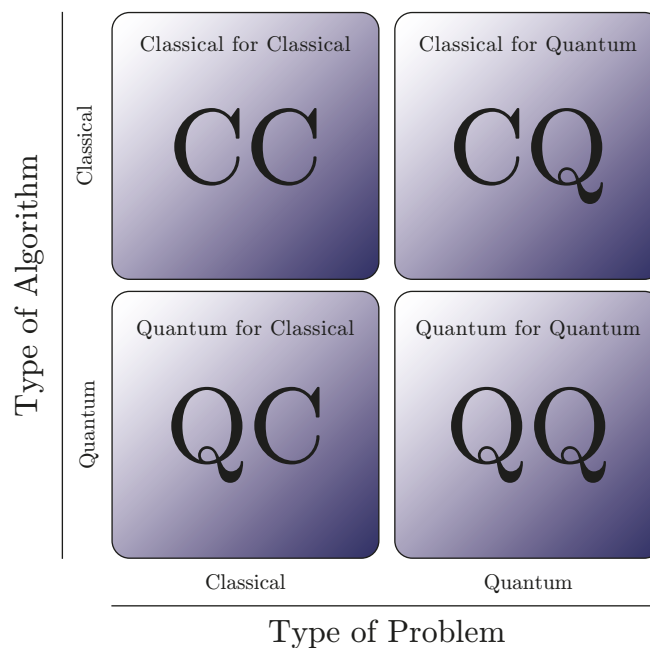


Figure 3.1: Quantum Machine Learning (QML) aims at studying the interplay between (classical) Machine Learning and Quantum Computation. Broadly speaking, the field can be schematically divided into four main areas, “Classical for Classical” (CC), “Classical for Quantum” (CQ), “Quantum for Classical” (QC) and “Quantum for Quantum” (QQ), depending on the type of algorithm used (hence the computing device, being classical or quantum), and the type of problem to be solved. See the main text for detailed explanation on such division. Such schematic representation of QML applications is customary in the literature, and can be found, for instance, in [206, 266].

between these two subjects can lead to mutual computational benefits and improved theoretical understandings.

The field is quite broad and it requires specialised knowledge from various domains of science, including mathematics, computer science, statistics, and of course quantum physics. However, usual investigations regarding Machine Learning and Quantum Computation can be factorised in four main areas, almost unrelated to each other, which are usually represented as shown in Fig. 3.1.

Indeed, depending on the type of problem to be solved, and the type of algorithm, hence the computing device, to solve it, the field of Quantum Machine Learning can be divided in:

1. **Classical for Classical (CC)** This area actually has actually no quantum part to it, and indicates those purely classical cases when a classical machine learning algorithm is used to solve a classical problem, that is task defined on objects and/or datasets which does not come from a quantum process. Examples are using Deep Neural Networks to classify images [113] or mastering board games [279];
2. **Classical for Quantum (CQ)** This area indicates those studies which aim to use classical machine learning procedures to deal with problems in the quantum physics domain [52, 55, 82]. Examples are using neural networks to represent quantum states [51], or using reinforcement learning techniques to compile quantum circuits for a quantum hardware [215];
3. **Quantum for Classical (QC)** This area instead refers to using quantum resources or algorithms, for example variational quantum algorithms, to analyse or process classical information, that is data that come from a classical source [25, 293]. Examples are the use of quantum subroutines to speed up linear algebra tasks in classical machine learning algorithms [160, 188], the use of parameterised quantum circuits to implement reinforcement learning agents for classical problems [151, 282, 285], or lastly the use of quantum computers to solve

optimisation problems in finance [94, 136];

4. **Quantum for Quantum (QQ)** This last area is arguably the most compelling yet unexplored application of Quantum Machine Learning, regarding the use of quantum processors to learn or study properties of quantum systems. Indeed, while it is reasonable to assume that quantum computing devices should be particularly suited to learning properties of quantum systems, at the moment this area is the most challenging to investigate, not only for the unavailability of reliable quantum computers to run the algorithms, but especially for the lack of so-called *quantum data*, on which the algorithms should be applied [59]. Whilst a clear definition of quantum data is yet to be found, one generally refers to either: in a weaker sense, to classical information extracted from a quantum system, for example via a measurement process; or, in a stronger sense, to sets of quantum states or objects stored on a quantum memory or device, and that can be accessed or generated on demand. Examples of this area are the use of convolutional quantum neural networks to identify phase transitions or devise error correction schemes [66], and the use of variational algorithms to learn quantum circuits to prepare mixed states [99].

Thus, as clear from the — somewhat arbitrary — division above, the subdomains of Quantum Machine Learning can be very broad and diverse, both in terms of topics and goals. In fact, as described in the review [325], while some investigations are more applied and specifically devoted to *quantum-enhanced* version of machine learning, others aim at a more theoretical exchange of ideas between machine learning and quantum physics, for example via *quantum-inspired* algorithms for classical machine learning [291, 297], or devising *quantum-generalised* versions of machine learning models and tasks. Examples of this last case are generalisations of neural networks to the quantum domain [21, 163, 195, 267], or the use of formalism of quantum mechanics to describe natural language processing (NLP) tasks [65, 320]. Needless to say, the diversity of QML is also a consequence of the great diversity and very rapid development of classical Machine Learning and Artificial Intelligence.

In this work, we are mainly interested in the use of quantum resources, specifically variational quantum algorithms, to analyse classical or quantum data, thus corresponding to the “QC” and “QQ” cases described above and in Fig. 3.1. Despite the variety of approaches, the current leading proposal for quantum machine learning is based on variational quantum algorithms, which, thanks to the use of limited quantum circuits in tandem with classical computers, can be run on near-term quantum devices.

In the following section, we introduce some basic concepts and definitions of (classical) machine learning, which will then be used later to introduce their quantum variants or generalisations.

3.2 Classical Machine Learning

Machine Learning (ML) is a subdomain of the broad field of Artificial Intelligence (AI), and it is a research field which investigates how to devise algorithms capable of discovering hidden patterns in data automatically, providing no —or very little— expert knowledge about the data to be analysed. The “discovery” process is called *learning* or *training*, and usually consists of some form of optimisation procedure where a penalty (reward) function, called *loss* function, depending on the data and on the task to be solved, is minimised (maximised).

Machine learning is usually divided into three main paradigms, *supervised*, *unsupervised* and *reinforcement* learning, depending on how the algorithm interacts with the data to be analysed, and the corresponding task to be solved. These can be summarised as follows:

1. **Supervised learning:** the algorithm is asked to reproduce the mapping between a set of inputs and desired outputs, and then extrapolate the acquired knowledge also on other data that was not used during training. In this case, the dataset provided to the algorithms consists of a pair of numbers, the input to be processed, and the correct result that the machine should output. The name *supervised* conveys the fact that, while training, the algorithm has

a target output, chosen by a human, that it has to reproduce. Regression over a series of two-dimensional points is an example of this type of learning.

This is arguably the most common and pedagogical use case of machine learning, even though in recent years it has been overshadowed by other approaches, due to the difficulty of meeting the criteria of having very large sets of *annotated* datasets, that are data accompanied with the corresponding correct label. Indeed, with the advent of the big data era, the acquisition of input data is often automated and efficient, but the annotation of the dataset usually requires human intervention, which may cause a bottleneck.

2. **Unsupervised learning:** in this case the algorithm has only access to a set of inputs, with no additional information about desired outputs. The goal, in this case, is to find patterns in the inputs, and extract relevant information and understanding about the whole dataset fed to the machine. The term unsupervised refers to the fact that the algorithm requires no human supervision to function.

Traditional examples of unsupervised learning are clustering algorithms, which divide input data into groups sharing similar features, or Principal Component Analysis (PCA), used to compress a large-dimensional dataset into a lower-dimensional representation, depending on the direction of the highest variance of the inputs. These procedures are often used as a preprocessing step to simplify the dataset before this is used as input to other (machine learning) algorithms.

State of the art research however points out how this paradigm of learning, when accompanied by very large parametric models and computational power, can reach amazing performances. An obvious example is Large Language Models (LLMs), which are advanced deep learning algorithms that are capable of understanding written language [42, 60].

3. **Reinforcement learning:** in this case the machine has no dataset at all to work with. Rather, this is *generated* by the algorithm itself while running, specifically through so-called interaction with the environment. Indeed, in the reinforcement learning framework, the algorithm, also referred to as an *agent*, interacts with an *environment* by performing *actions*. The environment then responds to the agent by giving it a *reward* if the performed action was good, where “good” is appropriately defined by the task to be solved.

Prototypical examples of reinforcement learning are algorithms capable of playing games, such as Chess or Go. In cases like these, there is no single obvious solution, and the algorithm is trained by trial and error, repeatedly playing the game, and rewarding it when it wins until it has learned a good-enough strategy. Again, state of the art research showed that reinforcement learning powered by large deep learning models can achieve super-human performances [279].

While these three paradigms summarise most of the machine learning methods, state of the art research on the field also comprises new approaches, like semi-supervised learning [98], self-supervised learning [85], continual learning [226] and transfer learning [318].

Before moving on, it is very important to stress that the most important feature of learning models is the so-called *generalisation*. As the name suggests, generalisation indicates the ability of a model (or a human, for that matter) to use the knowledge acquired over a restricted set of observations on a larger set of data, that was never seen earlier and without losing performances. This is indeed the most striking feature of machine learning models, which are empirically seen to generalise very well, and justify their recent enormous success. Generalisation performances are what ultimately distinguish machine learning (especially supervised learning) from standard fitting techniques.

In the following, we examine the basic definitions and tools of machine learning, particularly in the context of supervised learning, which is not only the most common and easiest way to introduce machine learning concepts, but is also the learning scenario used for the quantum machine learning models discussed in this work.

3.2.1 Basics of (supervised) Machine Learning

The goal of machine learning is to discover patterns from a set of limited observations of a given problem, and extrapolate such knowledge to other previously unseen instances of the problem. We now introduce the main components of machine learning borrowing the terminology from statistical learning theory, which is the branch of machine learning devoted to its theoretical understanding and mathematical formulation. A thorough discussion on statistical learning theory is beyond the scope of the present work, and we refer to [211, 274] for in-depth treatment of these topics.

3.2.1.1 Training dataset

The set of observations the learning algorithm has access to is called *training set*, and in the supervised learning scenario it consists of a series of pairs of input data, accompanied by a corresponding desired output. In full generality, let \mathcal{X} denote the input space from which inputs are drawn, and \mathcal{Y} the space of the outputs. Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ be the data space given by pairs of inputs and outputs, then the training set is usually defined as a set of *identically independently distributed iid* random variables

$$S := \{z_i = (\mathbf{x}_i, y_i) \mid z_i \sim \mathcal{D}\}_{i=1}^m, \quad S \in \mathcal{Z}^m, \quad (3.1)$$

where $\mathbf{x}_i \in \mathcal{X}$ are inputs, $y_i \in \mathcal{Y}$ are outputs, \mathcal{D} is a probability distribution over the data domain \mathcal{Z} from which samples $z_i \in \mathcal{Z}$ are sampled, and m is the number of samples in the training dataset.

In the vast majority of cases, inputs are vectors with real entries, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$, and outputs are either real numbers $y \in \mathcal{Y} \subset \mathbb{R}$, or integers $y \in \mathcal{Y} \subset \mathbb{N}$. In the former case, the problem is referred to as *regression* problem, while the latter is an example of a *classification* problem, since the desired outputs are discrete, are called *labels* in this context. However, there is no restriction on the nature of the inputs and the outputs, and the data space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ varies depending on the problem to be solved and the learning model used. For the sake of generality, in the following we keep using arbitrary domains for the data.

3.2.1.2 Hypothesis class

A learning model \mathcal{M} is defined as a family of functions which maps the input data space \mathcal{X} to the output data space \mathcal{Y} , namely

$$\mathcal{M} \subset \{h : \mathcal{X} \mapsto \mathcal{Y}\}. \quad (3.2)$$

This set of functions is called *hypothesis class*, and represents the set of functions that the chosen model can implement. For example, this represents the set of all possible mappings that a given neural network (see Sec. 3.2.2.2) with a fixed architecture could implement.

More often than not, the hypothesis class consists of a parametric model whose tunable parameters can be tuned to change the type of function implemented by the model. For simplicity, suppose the tunable parameters are real numbers $\mathbf{w} \in \mathbb{R}^p$, then the hypothesis class can be rewritten also as

$$\mathcal{M} := \{h_{\mathbf{w}} : \mathcal{X} \mapsto \mathcal{Y} \mid h_{\mathbf{w}} = h(\cdot ; \mathbf{w}), \mathbf{w} \in \mathbb{R}^p\}, \quad (3.3)$$

where $h_{\mathbf{w}} = h(\cdot ; \mathbf{w})$ is the specific parameterised function depending on trainable parameters \mathbf{w} .

3.2.1.3 Empirical Risk Minimisation

Given the training set and a hypothesis class, we now need to introduce a measure of the performance of the model. As with variational quantum algorithms, such measure is defined in terms of an

objective function (or *loss function*, or *cost function*) which measures how well a model $h \in \mathcal{M}$ is performing over the dataset S , and depends heavily on the task to be solved. The introduction of the loss function as a way to measure the fitness of the model effectively renders the training process an optimisation procedure.

Let $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ be a loss function, the performances of a model h in an hypothesis class \mathcal{M} are measured by the *empirical risk* over the training dataset S , defined as

$$L_S(h) := \frac{1}{m} \sum_{i=1}^m \ell(\hat{y}_i, y_i), \quad \text{with } \hat{y}_i = h(\mathbf{x}_i), \quad (3.4)$$

where $\hat{y}_i(\mathbf{w})$ is the prediction of the model when evaluated on input \mathbf{x}_i , and y_i is the desired output corresponding to such input, as prescribed by the training set S (3.1). Alternatively, if the hypothesis class is determined by a parametric model as in Eq. (3.3), the equivalent definitions hold

$$L_S(\mathbf{w}) := \frac{1}{m} \sum_{i=1}^m \ell(\hat{y}_i, y_i), \quad \text{with } \hat{y}_i(\mathbf{w}) = h_{\mathbf{w}}(\mathbf{x}_i) = h(\mathbf{x}_i; \mathbf{w}). \quad (3.5)$$

The optimal model is then defined as the one that minimises (or maximises) the empirical risk

$$h_{\text{opt}} = \arg \min_{h \in \mathcal{M}} L_S(h) \quad \text{or equivalently} \quad \mathbf{w}_{\text{opt}} = \arg \min_{\mathbf{w}} L_S(\mathbf{w}), \quad (3.6)$$

where the second equation above is equivalent to the definition of the optimal solution for variational quantum algorithms, as discussed in Eq (2.37).

The empirical risk (3.5) is also called *training error*, as it measures the fitness of the model on the available data contained in the training set. Such learning paradigm of defining the optimal model as the one having the lowest error on the training set is called *Empirical Risk Minimisation* (ERM) [274]. Empirical risk is opposed to the *true risk*, which is defined as the average value of the loss evaluated over the actual probability distribution \mathcal{D} from which observations $z_i = (\mathbf{x}_i, y_i) \sim \mathcal{D}$ are sampled, that is

$$L_{\mathcal{D}}(h) := \mathbb{E}_{z \sim \mathcal{D}}[\ell(\hat{y}; y)], \quad \text{with } \hat{y} = h(\mathbf{x}), z = (\mathbf{x}, y). \quad (3.7)$$

It is important to note that true risk (3.7) represents the actual quantity of interest that characterises the real performance of a learning model when dealing with the task. Indeed, by definition, the true risk (or *expected risk*) measures the performances of the learner on the whole probability distribution defining the task, not just on a restricted set of samples. However, this distribution is not known to the learner and the expected loss cannot be calculated directly, even though bounds on it can be derived, as discussed later in Sec. 3.2.1.5.

Thus, it is reasonable to use Empirical Risk Minimisation to select the optimal model, because it is the one that minimises the error over the available information of the problem to be solved, namely the one contained in the training dataset.

3.2.1.4 Loss functions and Learning

Supervised machine learning tasks can be grouped into two main classes, namely *Regression* and *Classification* problems, which differ essentially for the type of output that the learners have to reproduce.

Regression problems are those problems where the output data space is the real line, $\mathcal{Y} \subset \mathbb{R}$, and the goal of the learner is thus to learn a function mapping the inputs $\mathbf{x}_i \in \mathcal{X}$ to their corresponding outputs in the training set $y_i \in \mathcal{Y}$. The go-to loss function used in this scenario is the Mean Squared Error (MSE), defined as

$$L_S(h) = \frac{1}{m} \sum_{i=1}^m (\hat{y}_i - y_i)^2, \quad (3.8)$$

where again $\hat{y}_i = h(\mathbf{x}_i)$ is the prediction of the model h on input \mathbf{x}_i . A straightforward generalisation to the case of multidimensional outputs $\mathcal{Y} \subset \mathbb{R}^d$ is obtained by substituting the scalar squared difference with the Euclidean norm of the difference, namely $\ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) := \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2$.

Classification tasks instead are those for which the outputs, called *labels* in this context, are integers values $\mathcal{Y} \subset \mathbb{N}$. Thus, in this case, the goal of the learner is to split the inputs into separate classes, by assigning to each input in the dataset the correct label. For simplicity, let's consider the case of a binary classification task where the labels can only take two distinct values $\mathcal{Y} = \{0, 1\}$. The prediction of the learning model is then a real number $\hat{y}_i \in [0, 1]$ that encodes the probability that the input \mathbf{x}_i belongs to the class 0, or 1. The standard loss function used in this case is the so-called Crossentropy, defined as

$$L_S(h) = \frac{1}{m} \sum_{i=1}^m -(\hat{y}_i \log y_i + (1 - \hat{y}_i) \log(1 - y_i)). \quad (3.9)$$

which measures the difference between the predicted probability and the ground truth. Generalisations to *multiclass* classification problems (that is those where the number of classes is greater than two) are straightforward.

The introduction of a loss function makes the training, that is the procedure by which an optimal model is selected out of the hypothesis class, an optimisation procedure, as clear from the ERM approach of Eq. (3.5). Specifically, whenever the hypothesis class is a parametric model, training consists in adjusting the trainable parameters to minimise the empirical loss $L_S(\mathbf{w})$. As with variational quantum algorithms, such minimisation is implemented via variants of gradient descent (2.42), which we report also here for simplicity

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \nabla_{\mathbf{w}} L_S(\mathbf{w}) \Big|_{\mathbf{w}^{(t)}}. \quad (3.10)$$

Gradient descent-like update rules are particularly suited to neural network architectures, since there is an efficient strategy, called *backpropagation*, to compute the partial derivatives of the empirical loss with respect to each parameter in the model. However, as shown later in the section regarding linear models 3.2.2.1, there are cases where performing gradient-descent in the parameters space is not needed, as the optimal parameters can be found using an analytical closed-form solution.

3.2.1.5 Generalisation

The success of machine learning models, especially state of the art Deep Learning ones, is rooted in their *generalisation* performances, that is their ability to effectively use the knowledge extracted from the limited set of observations in the training dataset, also to new observations, which were not used during the training procedure. Generalisation conveys the desirable requirement that the learner has truly *understood* something of the problem to be solved, whereas it didn't just learn by heart the patterns in the training set.

Test set As we argued earlier, the true measure of performance (and generalisation) of a learner is given by the expected loss $L_{\mathcal{D}}(h)$, which is however impossible to access because the probability distribution of the samples \mathcal{D} is not known.

A practical solution to the estimation of generalisation performances of a model is to take the available corpus of observations $\{z_i\}_i \in \mathcal{Z}^{m+m'}$, and split it into two separate datasets: the training set $S \in \mathcal{Z}^m$, and a *test* set $T \in \mathcal{Z}^{m'}$. The former, defined before, is used during the training procedure to select the optimal model h_{opt} via minimisation of the empirical risk $L_S(h_{\text{opt}})$; while the latter is used only at the end of training, to measure the performance of the model on previously unseen observations, called test error $L_T(h_{\text{opt}})$, which is exactly a measure of the generalisation capabilities of the model. Generally, it is common practice to use about 80% of the available data to build the

training set, and the remaining 20% for the test set¹.

Generalisation bounds It is desirable for a learner h that its training error $L_T(h)$ and generalisation error $L_D(h)$ are close, so that the performance shown on the training dataset are representative of those obtained also on new data drawn from the same distribution.

This is true when the number of samples m in the training dataset is large, equivalent to saying that the model has access to a large amount of information about the problem to be solved. Given an hypothesis h , one can show that the probability that the true risk $L(h)$ (3.7) and the empirical risk $L_S(h)$ (3.5) are different goes to zero as the size m of the training set $S \sim \mathcal{D}^m$ goes to infinity, namely

$$\lim_{m \rightarrow \infty} P(|L_D(h) - L_S(h)| \geq \varepsilon) = \lim_{m \rightarrow \infty} P\left(\left|\mathbb{E}_{z \sim \mathcal{D}}[\ell(z)] - \frac{1}{m} \sum_{i=1}^m \ell(z_i)\right| \geq \varepsilon\right) \rightarrow 0. \quad (3.11)$$

This result is an application of the law of large numbers, which states that the average loss $L_S(h)$ tends to its expected value $L_D(h)$ (3.7) as the sample size goes to infinity. Note that, for ease of notation, we used the simplified expressions for the risk $\ell(z) = \ell(h(x), y)$ and $\ell(z_i) = \ell(h(x_i), y)$.

While interesting, the result in Equation (3.11) holds only asymptotically and gives no information about real case scenarios when the training set has a finite size. Luckily, one of the greatest achievements of statistical learning theory was to show that guarantees on the generalisation performances of a learning model can be obtained also in the finite size case [34, 274, 303, 304]. Indeed, making use of concentration inequalities [35], one can derive probabilistic statements about the generalisation performances of a hypothesis class \mathcal{M} , given a training dataset $S \in \mathcal{Z}^m$ consisting of *iid* samples z_i drawn from a probability distribution \mathcal{D} . These statements are referred to as *generalisation bounds*, and roughly take the following form [53, 251, 274].

Definition 3.1 — A general Generalisation Bound. Let \mathcal{M} be an hypothesis class and \mathcal{D} a probability distribution, for all $\delta \in (0, 1)$ with probability $1 - \delta$ over randomly drawn samples $S \sim \mathcal{D}^m$, and for all $h \in \mathcal{M}$ it holds that

$$L_D(h) \leq L_S(h) + f(\mathcal{M}, m, \delta), \quad (3.12)$$

where $f(\mathcal{M}, m, \delta)$ is a function that depends on the sample size m , the probability of error δ , and the hypothesis class \mathcal{M} , specifically through measures of its *complexity* (or *capacity*), which is a measure of the expressible power or richness of the hypothesis class. Examples of such complexity measures are the VC-dimension [304] or the Rademacher Complexity [274].

Generalisation bounds are statements about the predictive power of a learning model, expressing its generalisation performances in terms of two contributions: the empirical error obtained on the available data, and the flexibility of the hypothesis class. Roughly, if the complexity of the model can be controlled, and the empirical error is low, then one can be confident that the true error will be also small, and so some form of generalisation will take place. These bounds are interesting because, based on the available information on the performance of the learner, $L_S(h)$, it is possible to bound the true quantity of interest, namely the generalisation error $L_D(h)$, even though this is not directly accessible. In a certain sense, generalisation bounds can be seen as a mathematical formulation of Occam's razors: out of many possible explanations (the models), the simplest one (low complexity of the model) should be preferred.

There exists a plethora of generalisation bounds in the statistical learning literature, each one stemming from different assumptions and complexity measures. The interested reader can find

¹Actually, the best practice for implementing a safe training procedure involves the use of three distinct datasets: training, test and *validation* set, where the latter is used to monitor the generalisation error of the model *while* the model is training.

detailed information in [211, 274], as well as in Appendix B, where we show a concrete example of a generalisation bound for linear models based on Rademacher Complexity. For our discussion, it is sufficient to know that such bounds exist, since, as shown in Sec 3.3.3, we will discuss a specific scenario in which it is possible to derive a generalisation bound also for some type of quantum machine learning models, specifically data-reuploading quantum neural networks.

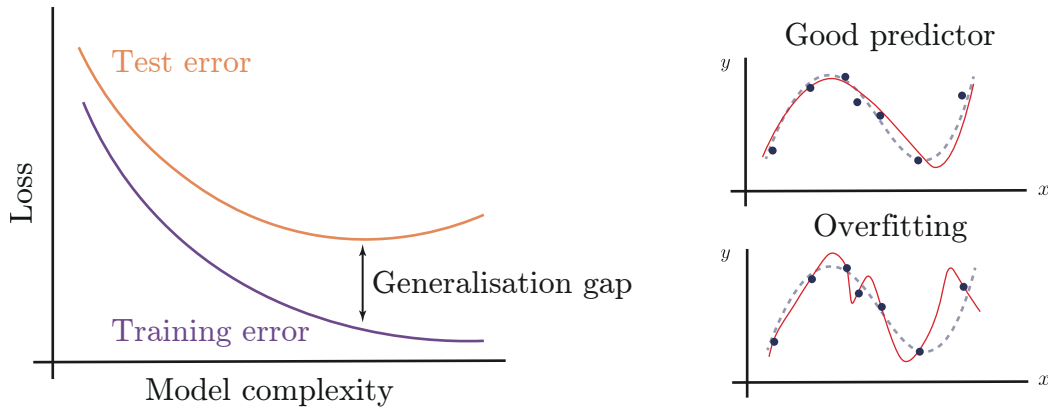


Figure 3.2: Generalisation and overfitting in supervised machine learning. There is a trade-off between the minimisation of the training error and the complexity of the learning model: if the model is not complex enough, it won't be able to solve the problem (large training and test error), however, if the model is too expressible (complex) it will fit precisely the training data (low training error), at the cost of compromising its generalisation (high test error). The difference between the loss attained on the training and test set is called *generalisation gap*. On the right, is an explicit example of overfitting in fitting a sinusoidal function (dashed line) given only a few data samples (the training dataset, blue dots): while a good model is capable of reproducing the desired sinusoidal behaviour of the data, and an overfitting model has zero error on the training data but miss the sinusoidal behaviour. Such a representation of the generalisation properties of a model is common in the machine learning literature, and can be found for example in [113, 128].

Overfitting Practically, generalisation is not guaranteed to happen in machine learning models unless specific actions, like regularisation techniques, are used to enforce it. Whenever the learner shows remarkable performances on the training set but fails to do the same on the test set, the model is said to be *overfitting* the training data. In this scenario, the learning model has specialised to reproduce exactly the mapping in the training dataset but not on other samples, thus showing poor generalisation.

Such a scenario is graphically depicted in the left panel of Fig. 3.2. As the complexity of the hypothesis class increased the test error of the model usually follows an “U”-shaped curve, which is a clear sign that the model has started overfitting the training data. Indeed, a first increase in complexity is needed to give the model enough flexibility to deal with the problem, but there is a moment after which the model is so expressible —i.e. complex— that it can explain every feature in the training data (even noisy data) so well that its knowledge cannot be generalised any more to other samples, and the error on the test set thus start increasing. An example of a good and bad learner is shown in the right panels of Fig. 3.2, for the task of reproducing a sinusoidal signal: whereas a good model (red line) is able to reproduce with good accuracy the required input-output mapping of the training data (blue dots) without compromising generalisation, an overfitting model reproduce exactly the training data at the cost of introducing data-dependent artefacts that miss the salient features of the underlying true problem. The trade-off between being to accurately model the pattern in the training data without compromising the generalisation performances is known as

the *bias-variance trade-off*.

In order to combat overfitting one resorts to regularisation techniques, which are set of procedures that impose constraints on the model to limit its complexity, with the hope that these help preserve generalisation. Examples are early-stopping, which stops the gradient-descent minimisation of the training loss before this reaches zero, at which point the model would have presumably overfitted the data, or the use of penalty terms in the loss function to favour simpler models over more complex ones [128].

3.2.2 Machine learning models

We now give two examples of machine learning models, specifically linear models (and kernel methods) and neural networks, of which generalised quantum versions have been proposed in the literature.

3.2.2.1 Linear models and kernel methods

Linear models correspond to the parametric hypothesis class (3.3)

$$\mathcal{M}_{\mathbf{w}} = \left\{ h_{\mathbf{w}} : \mathcal{X} \subset \mathbb{R}^d \mapsto \mathcal{Y} \subset \mathbb{R} \mid h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}, \mathbf{w} \in \mathbb{R}^d \right\}, \quad (3.13)$$

where we added the subscript $\mathcal{M}_{\mathbf{w}}$ to make it explicit that the hypothesis class is implemented by a parametric model depending *linearly* on parameters $\mathbf{w} \in \mathbb{R}^d$. Linear models like the one in (3.13) can be used for regression tasks with the empirical risk given by the mean squared error

$$L_S(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m (y_i - h_{\mathbf{w}}(\mathbf{x}_i))^2 = \frac{1}{m} \sum_{i=1}^m (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2. \quad (3.14)$$

By grouping inputs and outputs in the following matrix from

$$\mathbf{X} := \begin{bmatrix} -\mathbf{x}_1- \\ -\mathbf{x}_2- \\ \vdots \\ -\mathbf{x}_m- \end{bmatrix} \in \mathbb{R}^{m \times d}, \quad \mathbf{y} := \begin{bmatrix} -y_1- \\ -y_2- \\ \vdots \\ -y_m- \end{bmatrix} \in \mathbb{R}^m, \quad (3.15)$$

the empirical mean squared error can be rewritten more compactly as

$$L_S(\mathbf{w}) = \frac{1}{m} (\mathbf{y} - \mathbf{X}\mathbf{w})^2 = \frac{1}{m} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2, \quad (3.16)$$

The optimal model is the one minimising the empirical loss

$$\mathbf{w}_{\text{opt}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 \quad (3.17)$$

and has a closed form expression, known as *least square estimator*, which amounts to [82]

$$\mathbf{w}_{\text{opt}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}. \quad (3.18)$$

In this case, we assumed that the matrix $\mathbf{X}^T \mathbf{X}$ is non-singular, hence admits an inverse. In general, the solution to the least square minimisation problem (3.16) is given by the Moore-Penrose pseudoinverse, a generalisation of the inverse of a matrix, denoted as $\mathbf{w}_{\text{opt}} = \mathbf{X}^+ \mathbf{y}$ [22, 23, 112, 129]².

²When there are more samples than parameters $n > p$, the learning model is said to be *underparameterised*, and the related system of equations $\mathbf{X}\mathbf{w} = \mathbf{y}$ is *underdetermined*, which means that a solution may not exist. In this case, if \mathbf{X} has full column rank, then $(\mathbf{X}^T \mathbf{X})^{-1}$ exists and the Moore-Penrose pseudoinverse reduces to $\mathbf{X}^+ = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$, as in Eq. (3.18) for ordinary least squares. On the contrary, if there are more parameters than training data $n < p$, the model is said to be *overparameterised* and the system $\mathbf{X}\mathbf{w} = \mathbf{y}$ *overdetermined*, that is multiple solutions exist. In this case, if \mathbf{X} has full row rank, then $(\mathbf{X}\mathbf{X}^T)^{-1}$ exists and the Moore-Penrose pseudoinverse reduces to $\mathbf{X}^+ = \mathbf{X}^T (\mathbf{X}\mathbf{X}^T)^{-1}$. At last, if $n = p$ and \mathbf{X}^{-1} exists, then the solution is simply $\mathbf{w}_{\text{opt}} = \mathbf{X}^{-1} \mathbf{y}$. The introduction of a regularisation term as in Eq. (3.19) is not only useful to enforce generalisation, but also to ameliorate issues related to the singularity of the data matrices.

A more comprehensive treatment of linear regression is given by introducing a regularisation term in the loss (3.16) which penalises solutions with large norm, and also avoids subtleties related to the invertibility of the data matrices. This approach is known as *ridge regression*, and the optimal solution is defined as

$$\mathbf{w}_{\text{opt}} = \arg \min_{\mathbf{w}} \left\{ \frac{1}{m} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{m} \|\mathbf{w}\|_2^2 \right\}. \quad (3.19)$$

where $\lambda > 0$ is a parameter that controls the trade-off between minimising the empirical risk and preferring lower-norm solutions. Similarly to the previous case, the optimal parameters can be written explicitly as

$$\mathbf{w}_{\text{opt}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbb{I})^{-1} \mathbf{X}^\top \mathbf{y}, \quad (3.20)$$

where the matrix $(\mathbf{X}^\top \mathbf{X} + \lambda \mathbb{I})$ is non singular. Moreover, by making use of the matrix equality $(\mathbf{X}^\top \mathbf{X} + \lambda \mathbb{I}) \mathbf{X}^\top = \mathbf{X}^\top (\mathbf{X} \mathbf{X}^\top + \lambda \mathbb{I})$ [23, 82, 233], the optimal model can be eventually written in a more convenient form

$$\begin{aligned} h_{\text{opt}}(\mathbf{x}) &= \mathbf{w}_{\text{opt}} \cdot \mathbf{x} = \mathbf{x}^\top \mathbf{w}_{\text{opt}} \\ &= \mathbf{x}^\top \mathbf{X}^\top (\mathbf{X} \mathbf{X}^\top + \lambda \mathbb{I})^{-1} \mathbf{y}. \end{aligned} \quad (3.21)$$

which is often referred to as the solution of the *dual* problem of linear ridge regression. Let's now analyse the terms in this expression. First, one can easily check that the elements of the matrix $\mathbf{X} \mathbf{X}^\top \in \mathbb{R}^{m \times m}$ are the inner products of the training input vectors, namely

$$K_{ij} := [\mathbf{X} \mathbf{X}^\top]_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j \quad (3.22)$$

where $K = K_{ij}$ is usually called *kernel*, or *Gram*, matrix. Then, noticing that

$$\mathbf{x}^\top \mathbf{X}^\top = [\mathbf{x} \cdot \mathbf{x}_1, \mathbf{x} \cdot \mathbf{x}_2, \dots, \mathbf{x} \cdot \mathbf{x}_m] \in \mathbb{R}^m, \quad (3.23)$$

$$\hat{\boldsymbol{\alpha}} := (K + \lambda \mathbb{I})^{-1} \mathbf{y} \in \mathbb{R}^m. \quad (3.24)$$

where $\hat{\boldsymbol{\alpha}}$ are often called *dual* variables of the optimal weights \mathbf{w}_{opt} (3.20), one can finally rewrite the optimal predictor (3.21) as

$$h_{\text{opt}}(\mathbf{x}) = (\mathbf{x}^\top \mathbf{X}^\top) \cdot \hat{\boldsymbol{\alpha}} = \sum_{i=1}^m \hat{\alpha}_i (\mathbf{x} \cdot \mathbf{x}_i) = \mathbf{x} \cdot \left(\sum_{i=1}^m \hat{\alpha}_i \mathbf{x}_i \right). \quad (3.25)$$

This expression makes it evident that the optimal model depends on the data only through the inner products among data samples, and that it consists of a linear combination of the inner product between the new sample \mathbf{x} , with all the input data in the training set $S \in \mathcal{Z}^m$. In addition, as clear from the last equality in (3.25), the optimal weights can be expressed as a linear combination of such training samples.

Feature maps The derivation above follows identically even when we allow the input data $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ to go through an arbitrary function $\boldsymbol{\phi} : \mathcal{X} \mapsto \mathcal{F} \subset \mathbb{R}^s$. Indeed, consider the arbitrary mapping

$$\mathbf{x} \mapsto \boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_s(\mathbf{x})), \quad (3.26)$$

this is called *feature map*, and the inputs now belong to a new space called *feature space*, in this case $\mathcal{F} \subset \mathbb{R}^s$. The linear model now acts on such feature vectors as $h(\mathbf{x}) = \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x})$. The construction

of the optimal predictor derived before can be applied identically also in this case, simply by substituting the data matrix \mathbf{X} with the feature matrix

$$\mathbf{F} := \begin{bmatrix} -\boldsymbol{\phi}(\mathbf{x}_1) - \\ -\boldsymbol{\phi}(\mathbf{x}_2) - \\ \vdots \\ -\boldsymbol{\phi}(\mathbf{x}_m) - \end{bmatrix} \in \mathbb{R}^{m \times s}, \quad (3.27)$$

with the optimal predictor now being

$$h_{\text{opt}}(\mathbf{x}) = \sum_{i=1}^m \hat{\alpha}_i (\boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}_i)), \quad \hat{\boldsymbol{\alpha}} = (\mathbf{F}\mathbf{F}^\top + \lambda\mathbb{I})^{-1}\mathbf{y} \quad (3.28)$$

$$= \sum_{i=1}^m \hat{\alpha}_i \kappa(\mathbf{x}, \mathbf{x}_i), \quad \hat{\boldsymbol{\alpha}} = ([\kappa(\mathbf{x}_i, \mathbf{x}_j)]_{ij} + \lambda\mathbb{I})^{-1}\mathbf{y}. \quad (3.29)$$

In the last line we defined the *kernel function* $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, which takes two inputs and outputs the scalar value $\kappa(\mathbf{x}, \mathbf{x}_i) := \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{x}_i)$, from which one can define again the associated kernel matrix $\mathbf{K} \in \mathbb{R}^{m \times m}$, $K_{ij} := \kappa(\mathbf{x}_i, \mathbf{x}_j)$, evaluated on the training points.

The idea of using a feature map is to enrich the expressibility of the parameterised model by introducing a nonlinear dependence on the input data. Specifically, while the regression or classification task may not be solvable (or have high error) in the original data space \mathcal{X} , it may be much easier to solve in an appropriately chosen feature space [82]. Additionally, as clear from the expression (3.29), it is important to remark that the optimal model *only* depends on the data samples directly but *only* on inner products. This turns out useful because there are cases where a closed form expression for $\kappa(\cdot, \cdot)$ exists, and it is thus not necessary to transform the data with feature map $\boldsymbol{\phi}(\cdot)$ and then computing the inner product in the feature space \mathcal{F} . This simplification is known as *kernel trick*.

Kernel machines Equation (3.29) may suggest that instead of considering the hypothesis class of linear models (B.8), one could start directly from considering the parameterised class of predictors of the form

$$h(\mathbf{x}) = \sum_{i=1}^m \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i), \quad (3.30)$$

where $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a general but fixed *kernel function*, the parameters $\boldsymbol{\alpha} \in \mathbb{R}^m$ are to be optimised, and the model thus consists of a linear combination of kernel evaluations of the new data \mathbf{x} with those in the training set $\{\mathbf{x}_i\}_{i=1}^m$. Models like the one in Eq. (3.30) are called *kernel methods*, as they are based on the choice of an appropriate kernel function, which is a measure of similarity between data samples. Moreover, if the kernel function is symmetric and positive definite (known as Mercer's conditions), then there exist a *feature map* $\boldsymbol{\phi} : \mathcal{X} \mapsto \mathcal{F}$ and a *feature Hilbert space* \mathcal{F} such that the kernel function is equivalent to the inner product of vectors in such feature space [211, 274]

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}(\mathbf{x}) | \boldsymbol{\phi}(\mathbf{x}') \rangle_{\mathcal{F}} \quad (3.31)$$

where $\langle \cdot | \cdot \rangle_{\mathcal{F}}$ is the inner product on the Hilbert space \mathcal{F} , which is called the Reproducing Kernel Hilbert Space (RKHS) of the kernel function $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. Interestingly, it turns out that kernel predictors of the form (3.30) are very general and powerful, since they arise as a result of common machine learning optimisation problems. This is a consequence of the renowned *Representer theorem* of statistical learning, which shows that for supervised learning tasks that minimise an empirical risk in an RKHS, the optimal solutions can be expressed simply as a linear combination of kernel evaluations of with the training data [211, 260, 264].

While this seems obscure, this result is important because it guarantees that one can formulate the search of an optimal model in a possibly infinite-dimensional RKHS, simply as a search of kernel expansion coefficients $\{\alpha_i\}_{i=1}^m$ as in (3.30). For example, suppose one wants to minimise a linear model with a feature map

$$h(\mathbf{x}) = \langle \mathbf{w} | \phi(\mathbf{x}) \rangle_{\mathcal{F}} \quad (3.32)$$

where $\phi : \mathcal{X} \rightarrow \mathcal{F}$, and the dimension of the feature space \mathcal{F} is large, possibly even infinite. By virtue of the Representer theorem, one knows that the optimal predictor can be expressed as $h_{\text{opt}}(\mathbf{x}) = \sum_{i=1}^m \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i)$ where $\kappa(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}) | \phi(\mathbf{x}') \rangle_{\mathcal{F}}$ is the kernel induced by the feature map. Thus, one only has to search for the m real parameters α_i instead of looking directly for \mathbf{w}_{opt} in the large dimensional space \mathcal{F} . An example of infinite dimensional feature space is that corresponding to the so-called Radial Basis Function (RBF) kernel $\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2)$, which is induced by an infinite feature map $\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots]$ [82]. Instead, an example of finite but large feature space is that of quantum states, which will be the topic of the next sections.

A common application of a kernel method is *Kernel Ridge Regression* (KRR), where the kernel model (3.30) is trained with the regularised squared loss

$$L_S(h) = \sum_{i=1}^m (y_i - h(\mathbf{x}_i))^2 + \lambda \|h\|_{\mathcal{F}}^2 = (\mathbf{y} - K\boldsymbol{\alpha})^2 + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha}, \quad (3.33)$$

where $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel, or Gram, matrix evaluated on the training samples. Note that the first term is the usual squared loss (3.8), and the second term is a regularisation term similar to the one used in standard ridge regression (3.19), and depends on the norm of the kernel model in the corresponding RKHS³. The empirical loss is convex with respect to the parameters, and the minimum can be found analytically, achieved with parameters $\hat{\boldsymbol{\alpha}} = (K + \lambda \mathbb{I})^{-1} \mathbf{y}$.

Another ubiquitous example of kernel methods for classification —rather than regression— task is Support Vector Machines (SVM), which minimize the so-called *hinge loss* instead of the mean squared error [82, 211, 274]. To summarise, the general idea of kernel methods, is to define an appropriately chosen measure of similarity between samples (i.e. the kernel function $\kappa(\mathbf{x}, \mathbf{x}')$), and then search just for the expansion coefficients. The supervised learning task of finding the optimal parameters essentially translates to that of finding a proper kernel function for the task to be solved.

3.2.2.2 Neural Networks

A second very important example of a machine learning model are Neural Networks (NN). While they inherit their name from early mathematical models for biological neurons in the brain [255], modern neural networks have very little in common with how information is processed in the brain, but the name eventually stuck in the machine learning literature. Broadly speaking, by neural networks one refers to a broad class of parametric models where information is processed by single units, often referred to as neurons, and then transferred to other units in the network, which is composed of several of these neurons connected to each other according to a specific connectivity pattern. Depending on the specifics of the network, several different versions of neural networks

³If the kernel κ is a legitimate symmetric positive definite kernel, then it induces an inner product in a Reproducing Kernel Hilbert Space (RKHS) \mathcal{F} , so that $\kappa(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}) | \phi(\mathbf{x}') \rangle_{\mathcal{F}}$. Thus the kernel model can be written as

$$h(\mathbf{x}) = \sum_{i=1}^m \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i) = \left\langle \phi(\mathbf{x}) \left| \sum_{i=1}^m \alpha_i \phi(\mathbf{x}_i) \right. \right\rangle_{\mathcal{F}} = \langle \phi(\mathbf{x}) | \Phi \rangle_{\mathcal{F}} \quad (3.34)$$

which is a “linear model” in the RKHS of the kernel function, and the model “parameters” are given by the state $|\Phi\rangle = \sum_{i=1}^m \alpha_i |\phi(\mathbf{x}_i)\rangle$. Then, the norm of the model in the RKHS is defined as $\|h\|_{\mathcal{F}} = \sqrt{\langle h | h \rangle_{\mathcal{F}}}$, and so $\|h\|_{\mathcal{F}}^2 = \langle \Phi | \Phi \rangle = \sum_{i,j=1}^m \alpha_i \alpha_j \langle \phi(\mathbf{x}_i) | \phi(\mathbf{x}_j) \rangle = \sum_{i,j=1}^m \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\alpha}^T K \boldsymbol{\alpha}$, where $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel matrix.

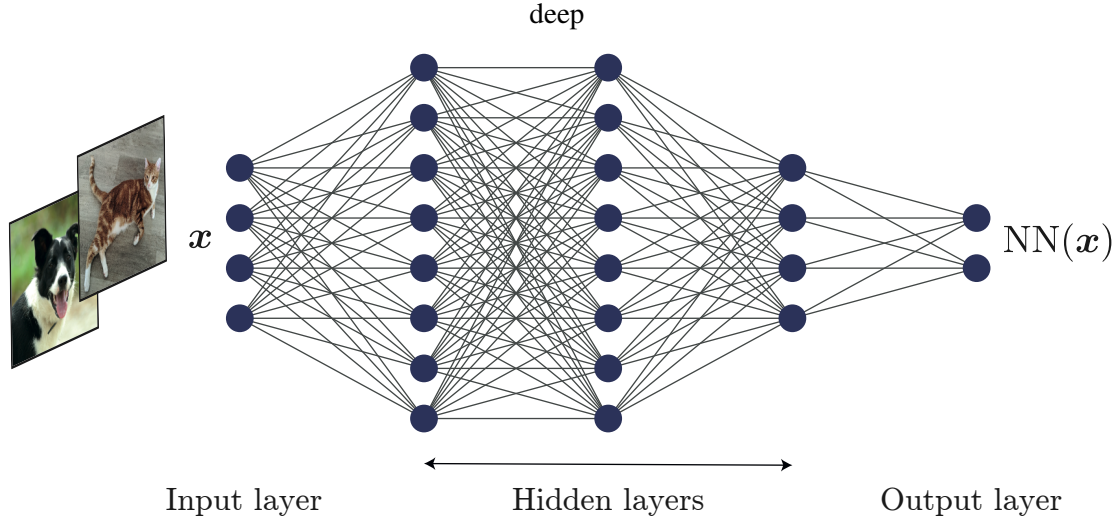


Figure 3.3: Graphical representation of a feedforward Neural Network. Inputs are fed to the network in the Input layer and then processed by a series of *hidden* layers, up until the end of the network is reached, corresponding to the output layer. Each node in the network is called *neuron* and implements the transformation (3.35), while neurons aligned vertically constitute a *layer*, whose action is reported in Eq. (3.36). The overall action of the network is given by concatenating each layer, as reported in Eq. (3.38). In this specific case, the neural network accepts inputs $\mathbf{x} \in \mathbb{R}^4$, has 3 hidden layers of variable sizes (8, 8 and 6), and outputs a 2-dimensional vector $NN(\mathbf{x}) \in \mathbb{R}^2$. The graph for the neural network was generated using [181], and similar representations are customary in standard literature on the subject, e.g. [112, 113, 128].

have been proposed, and in the following we introduce the most common example, feed-forward Neural Networks (ffNN), graphically represented in Fig. 3.3.

Feed-forward Neural Networks are called this way because information flows only in one direction in the network, where inputs are progressively processed by a series of *layers* of the network, until a final output is reached. A single unit in the neural network (drawn as a coloured node in Fig. 3.3) implements the mapping

$$\mathbf{x} \xrightarrow{\text{neuron}} \sigma(\mathbf{w} \cdot \mathbf{x} + b) \quad (3.35)$$

where $\mathbf{w} \in \mathbb{R}^p$ and $b \in \mathbb{R}$ are parameters to be optimised, which in the neural network jargon are called *weights* and *biases* respectively, and $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is a generic *nonlinear* function which is called *activation function*.

A set of neurons acting on the same input data form a *layer* of the neural network, and are represented by the vertically aligned nodes of Fig. 3.3. Let $\mathbf{x} \in \mathbb{R}^d$ and let the first layer be composed of h neurons, then the overall action of the layer on the input can be written as

$$\mathbf{x} \xrightarrow{\text{layer}} \sigma(W\mathbf{x} + \mathbf{b}), \quad W \in \mathbb{R}^{h \times d}, \mathbf{b} \in \mathbb{R}^b \quad (3.36)$$

where W is a matrix whose rows are the weights of the neurons in the layer, and similarly for the bias vector \mathbf{b} , and with a slight abuse of notation we impose that the activation function σ acts *elementwise* on the entries of a vector, namely $\sigma(\mathbf{v}) = [\sigma(v_1), \sigma(v_2), \dots]$.

In general, let $W^{(l)} \in \mathbb{R}^{h_l \times h_{l-1}}$ and $\mathbf{b}^{(l)}$ denote the weights and biases of the l -th layer of the neural network, an L -layers (excluding the input layer) feedforward neural network consists of the

following parameterised hypothesis class

$$\mathcal{M}_{\text{NN}} = \left\{ \text{NN}(\mathbf{x}) = \sigma \left(W^L \sigma \left(W^{(L-1)} \sigma \left(\dots \sigma \left(W^{(1)} \mathbf{x} + \mathbf{b}^{(1)} \right) \dots \right) + \mathbf{b}^{(L-1)} \right) + \mathbf{b}^{(L)} \right) \right. \\ \left. \mid W^{(l)} \in \mathbb{R}^{h_l \times h_{l-1}}, \mathbf{b}^{(l)} \in \mathbb{R}^{h_l}, l = 1, \dots, L \right\}, \quad (3.37)$$

which consists of a nested application of the transformation (3.36) using different parameters, but using the same activation function σ , even though this last constraint is not necessary. By defining the activation vector at layer l as $a^{(l)}(\mathbf{x}) = \sigma(W^{(l)}a^{(l-1)}(\mathbf{x}) + \mathbf{b}^{(l)})$, one can write the action of the neural network more compactly as

$$\text{NN}(\mathbf{x}) = a^{(L)}(\mathbf{x}) = \sigma \left(W^{(L)} a^{(L-1)}(\mathbf{x}) + \mathbf{b}^{(L)} \right), \quad (3.38)$$

with $a^{(0)}(\mathbf{x}) = \mathbf{x}$ being the trivial input layer containing simply the input vector.

Noteworthy, in addition to the trainable parameters given by the weights and biases, a neural network is specified by other additional parameters specifying its architecture, like the number of layers L (or depth of the network) and the activation function, that have a direct impact on the type of functions that the network can implement. As for the activation function, the key requirement is that it has to be nonlinear, as if this is not the case one can easily prove that the whole neural network collapse to a single-layer architecture implementing a simple affine transformation of the input $\mathbf{x} \mapsto W\mathbf{x} + \mathbf{b}$. Standard choices for the activation function are the sigmoid or the Rectified Linear Unit (ReLU)

$$\sigma(x) = \frac{e^x}{e^x + 1}, \quad \sigma(x) = \text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}. \quad (3.39)$$

A neural network can be trained in a supervised fashion by minimising the empirical mean squared loss over the training set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ (3.1). Let $\mathbf{W} = \{W^{(l)}, \mathbf{b}^{(l)} \mid l = 1, \dots, L\}$ denote the set of all trainable parameters in the neural network, the optimal model is implicitly defined by

$$L_S(\text{NN}) = \frac{1}{m} \sum_{i=1}^m (y_i - \text{NN}(\mathbf{x}_i))^2, \quad \mathbf{W}_{\text{opt}} = \arg \min_{\mathbf{W}} L_S(\text{NN}). \quad (3.40)$$

In this case however, there is no way analytical solution to this optimisation problem because the loss function is highly non-convex with respect to the parameters, as opposed to the previous case in Sec. 3.2.2.1 where there was a linear dependence on the trainable parameters.

Instead, neural network models are trained with gradient descent (3.10) via *backpropagation*, which is a very efficient method for calculating gradients of composed functions as neural networks. Indeed, one can compute the derivative of any parameter inside the network by a repeated application of the chain rule: starting from the output of the network, one proceeds backwards by “peeling-off” layers and accumulating gradients, up until the desired weight has been reached [82, 112]. The backpropagation algorithm permits a very efficient computations of gradients in neural networks and paved the way towards the adoption of large-scale Deep Learning models, with state of the art ones now leveraging up to hundreds of billions of parameters [60]. At last, it is worth noticing that the backpropagation algorithm is a specific example of automatic differentiation (AD), which is a set of techniques aiming at calculating the gradients of a computation algorithmically.

3.3 Quantum Machine Learning

In the previous section, we introduced the main concepts and tools of machine learning, and also discussed two prototypical examples of learning models: kernel methods and neural networks. In this section, we first introduce the idea of Quantum Machine Learning models distinguishing them

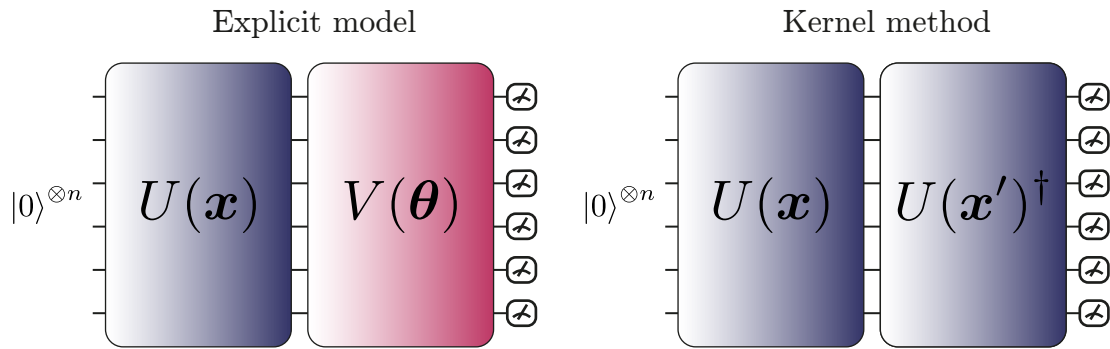


Figure 3.4: Circuit representation of linear quantum models: an *explicit model* on the left, and a kernel method (also called *implicit model*) on the right. In the former, a data encoding unitary operation is followed by a trainable variational block before the measurement of an observable takes place. In quantum kernel methods, the quantum computer is used to calculate the kernel function (i.e. inner products), while the tunable parameters are instead purely classical.

from Variational Quantum Algorithms, and then provide examples of quantum versions of the two classical models explained previously.

At the core of variational quantum algorithms and machine learning is the optimisation procedure, by which the algorithms are progressively adjusted to reach good performances. On the other hand, a striking difference is that machine learning models are algorithms that learn from observations (the training set), and in fact the presence of data is the hallmark of any machine learning model. Thus, a reasonable definition of a quantum machine learning model could be

Definition 3.2 — (Informal) Quantum Machine Learning model. A Quantum Machine Learning model is an algorithm that uses also, but not exclusively, quantum computational resources to solve a problem defined in terms of data.

Far from being a rigorous statement, such definition is admittedly general and omits a proper definition of “data”, still it suffices to describe many of the approaches described in Sec. 3.1.

Specifically, in this work we are focused on near-term quantum algorithms, namely variational algorithms, and thus, by the definition above, a near-term quantum machine learning model is a variational quantum algorithm where the cost function to be optimised is defined in terms of a set of observations or data. For example, a variational quantum algorithm for estimating the ground state energy of a molecule is not a quantum machine learning model, but a parameterised quantum circuit to implement a classification task of a set of observations is. In this sense, quantum machine learning models are a subset of variational quantum algorithms. However, while such characterisation can be useful to draw a boundary between QML and VQAs, these are often used interchangeably to indicate a quantum computation with tunable parameters that needs optimisation.

In the following, we introduce two quantum machine learning models that have been proposed in the literature: quantum classifiers and quantum kernel machines, which can be seen as quantum counterparts of classical linear models, and quantum neural networks which, as the name suggests, are inspired by classical neural networks.

3.3.1 Linear quantum models: quantum classifiers and kernel methods

The counterpart of linear models in the quantum domain are often called *linear quantum models*, because they effectively implement a linear separation in the Hilbert space of the quantum computer. Also in this case, such models are also strongly connected to quantum kernel methods, a connection which we explore in the following sections.

3.3.1.1 Explicit models

A quantum machine learning model deals with data, and thus it is necessary to find a way to load such information onto the quantum computer, so that it can be used as inputs for a quantum algorithm. The data to be analysed could be either *classical*, for example a set of input vectors $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$, or *quantum*, such as a set of quantum states $|\psi_i\rangle \in \mathcal{H} = \mathbb{C}^{2^n}$, where \mathcal{H} is the Hilbert space of a quantum system made of n qubits. In the following, we restrict our attention to the case of classical data, even though the discussion could be adapted to quantum data as well with minor modifications⁴.

Suppose we are given a set of classical data $\{\mathbf{x}_i\}_{i=1}^m$, the idea is to map these data to the quantum Hilbert space of the quantum computer, by a procedure which is often called *feature embedding*. Specifically, this is done by using a parameterised unitary operation $U(\mathbf{x})$ depending on the classical data to be loaded, so that the feature embedding consists of the map

$$\begin{aligned} \phi : \mathcal{X} &\rightarrow \mathcal{F}, \quad \mathcal{X} \subset \mathbb{R}^d, \quad \mathcal{F} = \mathcal{H} = \mathbb{C}^{2^n}, \\ \mathbf{x} &\mapsto |\phi(\mathbf{x})\rangle = U(\mathbf{x})|0\rangle^{\otimes n} \quad \text{or} \quad \mathbf{x} \mapsto \rho(\mathbf{x}) = |\phi(\mathbf{x})\rangle\langle\phi(\mathbf{x})|, \end{aligned} \quad (3.41)$$

where \mathcal{F} is the feature space, as used in Sec. 3.2.2.1 when discussing linear models in feature spaces, in this case given by the qubits Hilbert space $\mathcal{F} = \mathcal{H}$, and $|\phi(\mathbf{x})\rangle$ and $\rho(\mathbf{x})$ are the input quantum state and density matrix, respectively.

The parameterised block $U(\cdot)$ that actually maps the data to a quantum state is general, and various ansätze can be used to accomplish this task. A natural example, which has the benefit of being easily implementable on actual quantum computers, are parameterised Pauli rotations that use one qubit per dimension of the input data $d = n$, defined as

$$U(\mathbf{x}) = \bigotimes_{i=1}^n R_{P_i}(x_i), \quad \mathbf{x} = [x_1, x_2, \dots, x_n], \quad P_i \in \{X, Y, Z\}, \quad (3.42)$$

where $R_P(x)$ are Pauli rotations (2.6) around one of the Pauli axis $P \in \{X, Y, Z\}$. This approach of encoding the data as a rotational angle in a parameterised Pauli gate goes by the name of *angle embedding*. In addition, more complicated ansatz can be used involving multiple parameterised operations as well as two-qubits entangling gates, as those discussed in Chapter 7. It is fundamental to remark that, whenever inputs are encoded in a circuit as angles of parameterised rotations, these have to be rescaled in an appropriate angular range, for example $\mathbf{x} \in \mathcal{X} \subset [0, 2\pi]^d$, before feeding them to the circuit. If this is not the case then inputs differing by even multiples of 2π would be mapped on the same quantum state, since $U(\mathbf{x} + 2\pi) = U(\mathbf{x})$.

After the encoding phase, a variational unitary $V(\boldsymbol{\theta})$ acts on the system, thus yielding the parameterised state $|\phi_{\boldsymbol{\theta}}(\mathbf{x})\rangle = V(\boldsymbol{\theta})|\phi(\mathbf{x})\rangle = V(\boldsymbol{\theta})U(\mathbf{x})|0\rangle^{\otimes n}$. An observable O is measured at the end of the computation, and the final result is then

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \langle O \rangle_{\boldsymbol{\theta}; \mathbf{x}} = \langle \phi_{\boldsymbol{\theta}}(\mathbf{x}) | O | \phi_{\boldsymbol{\theta}}(\mathbf{x}) \rangle = \langle \phi(\mathbf{x}) | \underbrace{V(\boldsymbol{\theta})^\dagger O V(\boldsymbol{\theta})}_{O_{\boldsymbol{\theta}}} | \phi(\mathbf{x}) \rangle \quad (3.43)$$

$$= \text{Tr}[O_{\boldsymbol{\theta}} \rho(\mathbf{x})] = \langle O_{\boldsymbol{\theta}}, \rho(\mathbf{x}) \rangle_{HS}, \quad (3.44)$$

where in the last line we used the density matrix $\rho(\mathbf{x}) = |\phi(\mathbf{x})\rangle\langle\phi(\mathbf{x})|$, and then expressed the expectation value in terms of the Hilbert-Schmidt inner product of operators $\langle A, B \rangle_{HS} := \text{Tr}[A^\dagger B]$. A graphical representation of the quantum circuit implementing this evolution is shown in Fig. 3.4.

⁴Specifically, instead of considering the parameterised encoding unitary $|\psi_{\mathbf{x}}\rangle = U(\mathbf{x})|0\rangle^{\otimes n}$ depending on the input data \mathbf{x} , one can consider the unitary U_i that prepares the desired input state when acting on the ground state $|\psi_i\rangle = U_i|0\rangle^{\otimes n}$. In this last case however, it is important to note the circuit implementation U_i to implement a general (possibly Haar random) quantum state ψ_i can be exponentially —with respect to the number of qubits— deep. Thus, ideally, either the input quantum states are efficiently preparable on the quantum computer, or there is a controllable physical evolution that outputs the desired quantum state to be then processed by an appropriate quantum computing device

The last expression in Eq. (3.44) clearly exhibits the *linear* nature of this model in the Hilbert space \mathcal{H} of the quantum system⁵, in that the outcome of the circuit is a simple inner product between two Hermitian operators: the data-dependent state $\rho(\mathbf{x})$ and the trainable observable $O_{\boldsymbol{\theta}}$, which is the quantum analogue of the trainable weights \mathbf{w} of a classical linear model (3.13). However, note that while the results of the circuit depend *linearly* on such parameterised observable (and also on the quantum state), the dependence on the parameters and the input data is clearly nonlinear, usually trigonometric, since Pauli rotations are used to parameterise the encoding block $U(\mathbf{x})$ and the variational ansatz $V(\boldsymbol{\theta})$.

As with a regular machine learning model, the output of the quantum circuit $\hat{y}_i = f_{\boldsymbol{\theta}}(\mathbf{x}_i) = \langle O_{\boldsymbol{\theta}}, \rho(\mathbf{x}_i) \rangle_{HS}$ can then be used in a loss function to drive the training procedure, as discussed previously in Sec. 3.2.1.4. Moreover, whenever the required conditions hold one can use the parameter shift rule (2.55) to calculate the gradients of the circuit and use gradient descent to find the optimal value of the parameters.

The quantum machine learning model we have just described often goes by the name of *explicit* model [131, 152, 206, 264], because the optimal observable $O_{\boldsymbol{\theta}_{\text{opt}}}$ — actually, the optimal parameters $\boldsymbol{\theta}_{\text{opt}}$ — is searched directly in the Hilbert space via optimisation of the variational parameters, and the classification or regression problem is implemented by measuring it on a quantum computer. Explicit models are opposed to *implicit* models (or quantum kernel methods), which is the topic of the next section.

3.3.1.2 Quantum kernel (or implicit) models

We have discussed in Section 3.2.2.1 that linear models and kernel methods are strictly related, since kernel models can be seen as linear models in the so-called Reproducing Kernel Hilbert Space (RKHS) of the kernel function, and vice versa. These relations also hold for quantum kernel methods (often called *implicit models* [264]), which are parameterised predictors of the form [131, 206, 263]

$$f(\mathbf{x}) = \sum_{i=1}^m \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i) = \sum_{i=1}^m \alpha_i \text{Tr}[\rho(\mathbf{x})\rho(\mathbf{x}_i)]. \quad (3.45)$$

As with their classical counterparts (3.30), quantum kernel models make predictions using a linear combination of kernel evaluations between the new sample \mathbf{x} and those in the training set $\{\mathbf{x}_i\}_{i=1}^m$, where now the kernel function $\kappa(\mathbf{x}, \mathbf{x}') = \text{Tr}[\rho(\mathbf{x})\rho(\mathbf{x}')] = |\langle \phi(\mathbf{x}) | \phi(\mathbf{x}') \rangle|^2$ is given by the inner product in the feature quantum space. Models like these use the quantum computer only to compute the kernel, while the parameters $\{\alpha_i\}_{i=1}^m$ remain purely classical. A quantum circuit for evaluating the kernel function (i.e. inner product) for pure states is shown in the right panel of Fig. 3.4, but other strategies exist, for example leveraging the so-called SWAP test.

As we briefly mentioned when discussing classical kernel models, models of the form (3.45) are proven to be optimal by the Representer theorem, in the sense that any model in the RKHS that minimises an empirical risk can be expressed as a simple linear combination of kernel evaluations with the training set. Moreover, the complexity of the optimisation problem is sensibly reduced, since rather than optimising a parameterised observable $O_{\boldsymbol{\theta}} \in \mathbb{C}^{2n \times 2n}$ which can be a quite daunting task for $n \gg 1$, one only has to find the m real parameters $\{\alpha_i\}_{i=1}^m$.

Finally, note that by linearity of the trace, the model (3.45) implicitly defines a linear model of the form (3.44) where the observable is given by a linear combination of the feature quantum states from the training set, namely

$$f(\mathbf{x}) = \text{Tr} \left[\rho(\mathbf{x}) \left(\sum_{i=1}^m \alpha_i \rho(\mathbf{x}_i) \right) \right] = \text{Tr}[\rho(\mathbf{x}) O_S]. \quad (3.46)$$

⁵Note that for ease of exposition we are ignoring some subtleties related to the definition of the feature quantum space, as this can be the space of states $|\phi(\mathbf{x})\rangle$ or that of Hermitian operators (density matrices) $\rho(\mathbf{x})$. For a detailed discussion on this topic, we refer to [263].

Using the inner products between quantum states as kernel, these quantum kernel machines can then be used within regular ridge regression (3.19) or for classification tasks in the form of quantum support vector machines [131, 264].

3.3.1.3 Explicit or Implicit?

We have seen two examples of quantum machine learning models, explicit and implicit models, both belonging to the class of linear quantum models since they can be expressed as the inner product of the input quantum states with an observable $f(\mathbf{x}) = \text{Tr}[O\rho(\mathbf{x})] = \langle O, \rho(\mathbf{x}) \rangle_{HS}$. But what is the difference between the two, and are there any reasons to prefer one over the other? Two main distinctions can be made, regarding the optimisation and the classification performances of these models.

Let's first discuss the optimisation properties of these two models. Implicit models like (3.30) require $\mathcal{O}(m^2)$ queries to the quantum computer to estimate the kernel matrix of inner products between the m samples in the training dataset⁶, and additional $\mathcal{O}(m^3)$ classical post-processing steps to compute the optimal weights via inversion of the Gram matrix (3.23) for ridge regression [152, 211, 260]. On the other hand, variational training of explicit models is cheaper, because it is expected to terminate after a number of iterations proportional to the number of training samples, and thus it requires $\mathcal{O}(pm)$ calls to the quantum computer, where p is the number of parameters in the trainable observable O_{θ} , $\theta \in \mathbb{R}^p$. Thus, the most efficient strategy depends on the scenario: if a moderate amount of parameters are sufficient to fit a large amount of training data, then variational training may be the solution. On the contrary, if the required number of parameters scales with the number of data, then the two methods are equivalent in terms of computational resources. Clearly, the number of parameters in the variational block $V(\theta)$ to parameterise a general observable O_{θ} , which is needed to explore the whole space of observables to look for the optimal one, scales exponentially with the number of qubits. In practical scenarios, however, one restricts the expressibility of the model by selecting a specific parameterised ansatz with a limited amount of parameters.

Regarding the classification performances, implicit models are guaranteed by the Representer theorem to achieve the lowest possible empirical risk on the training set, when compared to explicit models trained with the same feature encoding. On the other hand, as argued in ref. [152], the latter may be desirable in terms of generalisation error, because their limited expressivity could help avoid overfitting. Hence, the use of constrained observables in explicit models —instead of the optimal observable of implicit models, see eq. (3.46)— may be advantageous to learning performances.

At last, we remark that if one is concerned with quantum advantages or speedups in these types of machine learning models, these can only be achieved if the kernel function $\kappa(\mathbf{x}, \mathbf{x}') = \text{Tr}[\rho(\mathbf{x})\rho(\mathbf{x}')] is hard to compute classically. Indeed, if this is not the case, then the quantum kernel model (3.45) can be simulated classically without the need for a quantum computer, and thus no advantages can be attained.$

3.3.2 Data reuploading models and Quantum Neural Networks

A second class of quantum machine learning models are data reuploading quantum circuits, often referred to as Quantum Neural Networks (QNNs). Differently from linear quantum models, the output of a quantum neural network cannot be written as an inner product between a data-dependent quantum state and a trainable observable, because these circuits use a repeated structure where data encoding blocks are interleaved trainable unitaries, so that the action of these two elements cannot be separated anymore.

Notably, it was recently shown that models in this class can be mapped to linear quantum models using approximate strategies or leveraging post-selection and gate teleportation, even though

⁶We hereby only count the number of values to be estimated on the quantum computer, ignoring the actual number of measurements needed to estimate each expectation value.

these constructions require additional conditions that cannot be easily met in practice [152]. Thus, while such a result suggests a unifying theoretical framework for describing quantum learning models, in practical instances (that are those that one would execute on an actual near-term quantum computer), linear models and data reuploading ones behave rather differently, as we shall see.

Data reuploading quantum circuits have appeared independently multiple times in the literature [109, 229, 269], with all of them underlining the tight connection between input redundancy inside a quantum circuit, and the capability of the latter of expressing more complicated — specifically, higher-frequency trigonometric — functions on the input data. In a sense, this need of loading the inputs multiple times in the circuits, hence the name data reuploading, can be seen as an analogue of feeding the same input to multiple neurons in the first layer of a classical neural network, as depicted in Fig. 3.3.

Indeed, provided that input data enters the circuit via rotation-like gates (see below for a more rigorous statement), it is possible to show that the outcome of any parameterised quantum circuit depending on data can be written as a truncated Fourier series, or as a Generalised Trigonometric Polynomial (GTP) [53, 109, 269, 322].

Theorem 3.1 — Data-dependent parameterised quantum circuits are truncated Fourier series (53, 109, 269). Let $U_{\boldsymbol{\theta}}(\mathbf{x})$ be the unitary matrix of a parameterised quantum circuit depending on some input data $\mathbf{x} \in \mathcal{X} \subset [0, 2\pi]^d$ and trainable parameters $\boldsymbol{\theta}$. If data coordinates $x_i \in \mathbf{x}$ enter the quantum circuit via data encoding operations of the form $U(x_i) = \exp(-ix_i H)$ where H is an Hermitian operator, then the following holds

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \langle O \rangle_{\mathbf{x}; \boldsymbol{\theta}} = \langle \mathbf{0} | U_{\boldsymbol{\theta}}(\mathbf{x})^\dagger O U_{\boldsymbol{\theta}}(\mathbf{x}) | \mathbf{0} \rangle = \sum_{\boldsymbol{\omega} \in \Omega} c_{\boldsymbol{\omega}} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}}, \quad (3.47)$$

where $\Omega = \{\boldsymbol{\omega} \in \mathbb{R}^d\}$ is the *frequency spectrum* associated with the quantum circuit and depends solely on the number of data encoding operations present in the circuit, as well as their generators, a piece of information denoted as *data encoding strategy*. The expansion coefficients $\{c_{\boldsymbol{\omega}} \in \mathbb{C}\}$ instead depend on the structure of the circuit (including the data encoding strategy), the trainable parameters $\boldsymbol{\theta}$, and finally also the observable O . For any frequency $\boldsymbol{\omega} \in \Omega$, also $-\boldsymbol{\omega} \in \Omega$. Also, the coefficients satisfy $c_{\boldsymbol{\omega}} = c_{-\boldsymbol{\omega}}^*$, which ensures that the Fourier expansion correctly evaluates to a real number.

As an example, parameterised quantum circuits that encode the input data via Pauli rotations of the form (3.42) belong to this class of model, hence admit a Fourier expansion.

The Fourier representation in Eq. (3.47) beautifully summarises the class of functions that parameterised quantum models depending on data via parameterised rotations can implement, namely trigonometric functions^{7,8}. A clear example of such circuits are those that encode data via Pauli rotations of the form (3.42).

Interestingly, a Fourier expansion of the form (3.47) holds for *any* quantum circuit provided that inputs are loaded via an angle-embedding scheme, irrespectively of the choice of the gates in the circuit, their position, or their number, as shown in Fig. 3.5. Such formulation clearly demonstrates that the Fourier-like nature of the circuit is a consequence of the data encoding strategy, and also this implicitly determines the set of frequencies $\boldsymbol{\omega} \in \Omega$ that the quantum model has access to. Instead, the trainable parameters, along with the observable and the structure of the circuit, control

⁷It is important to note that we are neglecting classical preprocessing step of the input data. That is, the data $\mathbf{x} \in \mathcal{X}$ are fed directly as parameters to the data encoding operations $U(\mathbf{x})$ without any classical preprocessing. If classical preprocessing is used, $\mathbf{y} = g(\mathbf{x})$, then the PQC is clearly a Fourier series of \mathbf{y} , not of the original data \mathbf{x} . Classical preprocessing can be used to change the functional dependence of the expectation value on the original input [209, 276].

⁸We already had an hint of this fact in Eq. (2.51), when we discussed how to derive the parameter-shift rule for parameterised quantum circuits. Also in that case, when fixing all the parameters but one, we saw that the circuit effectively implements a trigonometric function of that remaining parameter with frequency spectrum $\Omega = \{0, 1\}$.

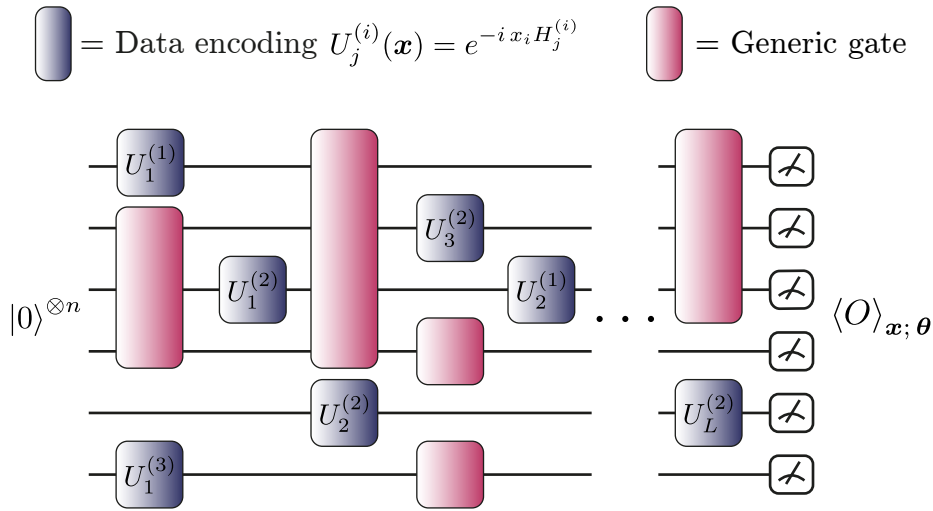


Figure 3.5: A generic quantum circuit can be expressed as a truncated Fourier series (3.47) over the input data \mathbf{x} , provided that the coordinates of the input are loaded in the quantum circuit via gates of the form $U(x_i) = \exp(-ix_i H)$. This figure is a custom reproduction of Fig. 3 in [53].

which frequencies can actually be expressed by the circuit, by tuning the expansion coefficients $\{c_{\omega}\}$.

3.3.2.1 Deriving the Fourier expansion

In the following, we derive the Fourier expansion (3.47) along the lines of the elegant treatment in ref. [53]. Different proofs and further discussions can be found also in refs. [109, 229, 269].

As for the previous quantum machine learning models, also in this case we distinguish between encoding gates $U(\mathbf{x})$, depending on the input data $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$, and trainable operations $\{V_l(\boldsymbol{\theta}_l)\}$, depending on trainable parameters $\{\boldsymbol{\theta}_l\}_l$. However, as we shall see, the explicit form of such trainable gates is not necessary to derive the desired Fourier-like expansion of the quantum circuit. In fact, any gate non depending on data, be it fixed or parameterised, is effectively absorbed into the definitions of the expansion coefficients, possibly in a very intricate way, without playing any major role in the mathematical derivation.

Let $\mathbf{x} = [x_1, x_2, \dots, x_d] \in \mathbb{R}^d$ be the usual input vector, we consider encoding unitary operations that encode the coordinates of the input via evolutions of the form

$$U_j^{(i)}(\mathbf{x}) = \exp(-i x_i H_j^{(i)}) = \exp(-i \mathbf{e}_i \cdot \mathbf{x} H_j^{(i)}), \quad (3.48)$$

where $H_j^{(i)}$ is the j -th Hermitian operator encoding the i -th coordinate x_i , and \mathbf{e}_i is a unit vector having zeros everywhere except on the i -th component, so that $\mathbf{e}_i \cdot \mathbf{x} = x_i$. As the notation suggests, we allow for each data coordinate to be uploaded in the circuit multiple times throughout the circuit, even with different generators.

Let's consider the action of the data encoding block (3.48) on a generic quantum state with density matrix ρ . Let $U(\mathbf{x}) = \exp(-i \mathbf{e} \cdot \mathbf{x} H)$ with $H = \sum_k \lambda_k |\lambda_k\rangle\langle\lambda_k|$ the spectral decomposition of the generator, by expanding state ρ on the eigenbasis of H , one obtains

$$\begin{aligned} \rho \rightarrow \rho(\mathbf{x}) &= U(\mathbf{x}) \rho U(\mathbf{x})^\dagger = U(\mathbf{x}) \left(\sum_{kl} \rho_{kl} |\lambda_k\rangle\langle\lambda_l| \right) U(\mathbf{x})^\dagger \\ &= \sum_{kl} e^{-i(\lambda_k - \lambda_l) \mathbf{e} \cdot \mathbf{x}} \rho_{kl} |\lambda_k\rangle\langle\lambda_l|. \end{aligned} \quad (3.49)$$

This expression can be simplified by grouping together those indexes (k, l) that give rise to the same frequency difference $\lambda_k - \lambda_l$. In order to do so, it is convenient to introduce the frequency spectrum associated with the generator, that is the set of all possible differences of its eigenvalues multiplied by the unit vector

$$\Omega(H) = \{(\lambda_k - \lambda_l)\mathbf{e} \mid \forall \lambda_k, \lambda_l \in \text{eigvals}(H)\}. \quad (3.50)$$

Note that, by definition of the frequency spectrum, for any frequency $\boldsymbol{\omega} \in \Omega(H)$, also the negative frequency belongs to the spectrum $-\boldsymbol{\omega} \in \Omega(H)$. It is possible to group together those terms that correspond to the same frequency difference, that is defining

$$\rho_{\boldsymbol{\omega}} = \sum_{(k,l) \in I(\boldsymbol{\omega})} \rho_{kl} |\lambda_k\rangle\langle\lambda_l| \quad \text{with} \quad I(\boldsymbol{\omega}) := \{(i, j) \mid (\lambda_i - \lambda_j)\mathbf{e} = \boldsymbol{\omega}\}. \quad (3.51)$$

With these definitions, the evolved quantum state can be finally rewritten as

$$\rho(\mathbf{x}) = \sum_{\boldsymbol{\omega} \in \Omega(H)} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \rho_{\boldsymbol{\omega}}, \quad (3.52)$$

with the expectation value of an observable O on this state thus being

$$\langle O \rangle = \text{Tr}[O\rho(\mathbf{x})] = \sum_{\boldsymbol{\omega} \in \Omega(H)} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \text{Tr}[O\rho_{\boldsymbol{\omega}}] = \sum_{\boldsymbol{\omega} \in \Omega(H)} c_{\boldsymbol{\omega}} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}}, \quad (3.53)$$

where we defined the coefficients $c_{\boldsymbol{\omega}} = \text{Tr}[O\rho_{\boldsymbol{\omega}}]$. Since the original state ρ is Hermitian, using the definitions (3.51) one can check that $\rho_{\boldsymbol{\omega}} = \rho_{-\boldsymbol{\omega}}^\dagger$, hence the coefficients satisfy $c_{\boldsymbol{\omega}} = c_{-\boldsymbol{\omega}}^*$, as needed to ensure that $\langle O \rangle$ is real-valued as expected. We have thus recovered the Fourier representation of the circuit (3.47) for the case of a single encoding step on a generic quantum state. The same derivation can be applied straightforwardly when more encoding gates are applied, and also when other operations, like the trainable unitaries, act on the system.

Indeed, starting from the former case, one can see that the action of any *non-data-encoding* unitary only amounts to a change of basis which can be absorbed inside the operators $\rho_{\boldsymbol{\omega}}$. In fact, suppose a unitary V acts on the quantum state $\rho(\mathbf{x})$ (3.52), this is consequently changed to

$$\rho(\mathbf{x}) \rightarrow V\rho(\mathbf{x})V^\dagger = V \left(\sum_{\boldsymbol{\omega} \in \Omega(H)} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \rho_{\boldsymbol{\omega}} \right) V^\dagger = \sum_{\boldsymbol{\omega} \in \Omega(H)} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} V\rho_{\boldsymbol{\omega}}V^\dagger \quad (3.54)$$

$$= \sum_{\boldsymbol{\omega} \in \Omega(H)} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \tilde{\rho}_{\boldsymbol{\omega}}, \quad (3.55)$$

where the action of the unitary can be just absorbed in the operators $\{\rho_{\boldsymbol{\omega}}\} \rightarrow \{V\rho_{\boldsymbol{\omega}}V^\dagger\}$, hence the in coefficients $\{c_{\boldsymbol{\omega}}\}$, without changing the frequency spectrum of the circuit Ω or its Fourier representation. Thus, we can concentrate on the action of the data encoding gates only.

Indeed, let $U_2(\mathbf{x}) = \exp(-i\mathbf{e}_2 \cdot \mathbf{x} H_2)$ be another encoding operation (for clarity, in what follows we add the subscript “1” to indicate the previous encoding operation), the state $\rho(\mathbf{x})$ is evolved according to

$$\rho(\mathbf{x}) \rightarrow \rho'(\mathbf{x}) = U_2(\mathbf{x})\rho(\mathbf{x})U_2(\mathbf{x})^\dagger = \sum_{\boldsymbol{\omega}_1 \in \Omega(H_1)} e^{-i\boldsymbol{\omega}_1 \cdot \mathbf{x}} U_2(\mathbf{x})\rho_{\boldsymbol{\omega}_1}U_2(\mathbf{x})^\dagger \quad (3.56)$$

$$= \sum_{\boldsymbol{\omega}_1 \in \Omega(H_1)} e^{-i\boldsymbol{\omega}_1 \cdot \mathbf{x}} \sum_{\boldsymbol{\omega}_2 \in \Omega(H_2)} e^{-i\boldsymbol{\omega}_2 \cdot \mathbf{x}} \rho_{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2} \quad (3.57)$$

where in the second line we expressed the operators $\{\rho_{\boldsymbol{\omega}_1}\}$ in the eigenbasis of the generator H_2 , and then re-indexed the sum in terms of the corresponding frequency spectrum $\Omega(H_2)$, namely

$$U_2(\mathbf{x})\rho_{\boldsymbol{\omega}_1}U_2(\mathbf{x})^\dagger = U_2(\mathbf{x}) \left(\sum_{kl} [\rho_{\boldsymbol{\omega}_1}]_{kl} |\lambda_k^{(2)}\rangle\langle\lambda_l^{(2)}| \right) U_2(\mathbf{x})^\dagger \quad (3.58)$$

$$= \sum_{kl} e^{-i(\lambda_k^{(2)} - \lambda_l^{(2)})\mathbf{e}_2 \cdot \mathbf{x}} [\rho_{\boldsymbol{\omega}_1}]_{kl} = \sum_{\boldsymbol{\omega}_2 \in \Omega(H_2)} e^{-i\boldsymbol{\omega}_2 \cdot \mathbf{x}} \rho_{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2} \quad (3.59)$$

where the frequency spectrum $\Omega(H_2)$ is defined as before (3.50), and the sum was re-indexed accordingly.

One more ingredient is needed to simplify the expression (3.57), namely how to compose the frequency spectrums $\Omega(H_1)$ and $\Omega(H_2)$ arising from the two different encoding operations. This can be done by the so-called Minkowski sum between the two sets of frequencies, defined as

$$\Omega = \Omega(H_1) + \Omega(H_2) := \{\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2 \mid \forall \boldsymbol{\omega}_1 \in \Omega(H_1), \forall \boldsymbol{\omega}_2 \in \Omega(H_2)\}. \quad (3.60)$$

With this, the quantum state can thus be written as

$$\rho'(\mathbf{x}) = \sum_{\boldsymbol{\omega}_1 \in \Omega(H_1)} \sum_{\boldsymbol{\omega}_2 \in \Omega(H_2)} e^{-i(\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2) \cdot \mathbf{x}} \rho_{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2} = \sum_{\boldsymbol{\omega} \in \Omega} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \rho_{\boldsymbol{\omega}}, \quad (3.61)$$

where, as before, the operators $\{\rho_{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2}\}$ corresponding to the same frequency $\boldsymbol{\omega} = \boldsymbol{\omega}_1 + \boldsymbol{\omega}_2$ were summed together, and the sum was then re-indexed according to the frequencies in the joint spectrum Ω . From this, one can calculate the expectation value of the observable $\langle O \rangle = \text{Tr}[O\rho'(\mathbf{x})]$, thus obtaining again Eq. (3.53).

The same derivation can be applied multiple times for all data encoding gates in the circuit, which have the net effect of adding more frequencies in the accessible spectrum Ω . All other operations instead just impact the coefficients in the series. Thus, we showed that *any* quantum circuit that encodes input data via evolutions of the form (3.48) can be expressed as a truncated Fourier series (3.47) of the inputs, as desired. ■

3.3.2.2 A single-qubit data reuploading circuit

Although the Fourier expansion (3.61) is a powerful and concise statement about the output of a quantum circuit, an inexperienced user might find it difficult to use it on real instances, mainly because of the rather opaque composition rule (3.60) by which the total frequency spectrum Ω is defined. To make things clearer, let's then consider a simple example of a single-qubit data-reuploading circuit for a univariate input data $x \in \mathbb{R}$, namely

$$f_{\boldsymbol{\theta}}(x) = \langle 0 | U_{\boldsymbol{\theta}}(x)^\dagger O U_{\boldsymbol{\theta}}(x) | 0 \rangle, \quad U_{\boldsymbol{\theta}}(x) = V_L(\boldsymbol{\theta}_L) U_L(x) \cdots V_1(\boldsymbol{\theta}_1) U_1(x) V_0(\boldsymbol{\theta}_0), \quad (3.62)$$

where the encoding gates are $U_l(x) = \exp(-ixP_l)$, $P_l \in \{X/2, Y/2, Z/2\}$ are Pauli rotations. The eigenvalues of every Pauli matrix are $\text{eigvals}(P_l) = \{\pm 1/2\}$, so by the definition (3.50), the frequency spectrum associated to any such matrix is

$$\Omega(P_l) = \{\lambda_k - \lambda_l \mid \forall \lambda_k, \lambda_l \in \{\pm 1/2\}\} = \{-1, 0, 1\}, \quad P_l \in \{X, Y, Z\} \times \frac{1}{2}. \quad (3.63)$$

When L encoding gates $U_l(x)$ are used load the data, the total spectrum (3.60) will be

$$\begin{aligned} \Omega &= \Omega(P_1) + \dots + \Omega(P_L) = \{\boldsymbol{\omega}_1 + \dots + \boldsymbol{\omega}_L \mid \boldsymbol{\omega}_1 \in \Omega(P_1), \dots, \boldsymbol{\omega}_L \in \Omega(P_L)\} \\ &= \{\boldsymbol{\omega}_1 + \dots + \boldsymbol{\omega}_L \mid \boldsymbol{\omega}_i \in \{-1, 0, 1\}\} \\ &= \{-L, -(L-1), \dots, -1, 0, 1, \dots, L-1, L\}, \end{aligned} \quad (3.64)$$

The size of the spectrum $|\Omega| = 2L + 1$ directly depends on the number of times the input x appears in the circuit, as each Pauli encoding gate effectively increases the accessible frequency spectrum by adding higher order *integer* frequencies.

Noteworthy, an integer-valued spectrum is a general feature of Pauli encodings, which is due to their generators having eigenvalues $\pm 1/2$. Indeed, this also holds for the more complicated case of multivariate inputs and multiple qubits: whenever Pauli rotations are used to encode the data $\mathbf{x} \in \mathbb{R}^d$ in the quantum circuit, the generated spectrum is integer-valued $\boldsymbol{\omega} \in \Omega \subset \mathbb{Z}^d$.

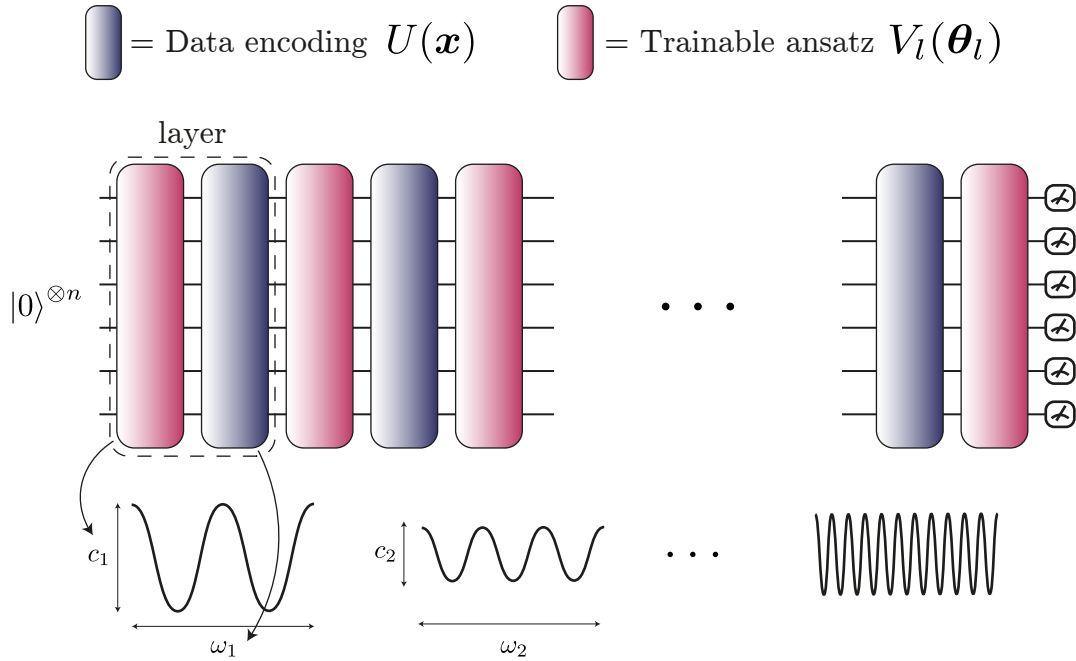


Figure 3.6: Data reuploading quantum circuit in the form of Quantum Neural Network (3.66), consisting of multiple layers of encoding gates $U(\mathbf{x})$ and trainable unitaries $V(\boldsymbol{\theta})$. Leveraging on the Fourier analysis developed so far (3.47), we know that the data encoding blocks define which frequencies $\{\omega_i\}$ are accessible by the quantum neural network, while the variational unitaries control the coefficients $\{c_i\}$ of the Fourier expansion. This figure is a custom reproduction of Fig. 1 in [269].

Finally, given the spectrum (3.64), the action of a data reuploading single-qubit quantum circuit using Pauli encodings can then be expressed as

$$f_{\boldsymbol{\theta}}(x) = \sum_{n=-L}^L c_n(\boldsymbol{\theta}) e^{-inx}, \quad (3.65)$$

which is a truncated Fourier series of degree L of the input data x .

With a little imagination, one can convince himself that an equivalent result can be obtained also for higher-dimensional inputs on bigger circuits with multiple qubits. Essentially, whenever a data coordinate x_i is encoded in the circuit, more frequencies of that coordinate appear in the Fourier expansion of the circuit (3.47).

In conclusion, one can *design* the frequency spectrum accessible by the circuit by changing the data encoding strategy, namely the number of encoding gates per coordinate and their generators. The size of the accessible spectrum, hence the complexity of the Fourier expansion, depends uniquely on the data encoding strategy, and by considering different eigenvalues of the encoding generators one can create rather different frequency spectrum [232, 276].

3.3.2.3 Quantum Neural Networks

We have just seen that uploading the data multiple times inside a circuit is of paramount importance to enrich the class of functions that a quantum machine learning model can express.

Building on such intuition, a class of quantum models that appeared prominently in the literature are so-called *Quantum Neural Networks* (QNNs). Although this term is also often used to indicate

any variational quantum circuit using a machine learning jargon, it seems reasonable to indicate with quantum neural networks specifically those parameterised circuits which depend on data (see definition 3.2) and that use a repeated structure of encoding and variational layers to add input redundancy, thus increasing the expressivity of the model. More formally, QNNs usually take the following form

$$U_{\boldsymbol{\theta}}(\mathbf{x}) = \prod_{l=1}^L U(\mathbf{x}) V_l(\boldsymbol{\theta}_l), \quad (3.66)$$

which is a repeated structure of data encoding blocks $U(\mathbf{x})$ and trainable operations $V_l(\boldsymbol{\theta})$, and L is the number of *layers* in the quantum neural networks. A graphical representation of this circuit is shown in Fig. 3.6.

Such circuitual architectures made of repeated layers of similar operations make them similar to how feed-forward neural networks 3.3 are built, and thus “justifies” the name *quantum neural networks*. Needless to say, classical and quantum neural networks are completely different objects, with different properties and corresponding hypothesis classes, and such juxtaposition only holds on at a conceptual level.

3.3.3 Generalization of QML models

We have just shown that the action of a parameterised quantum model can be written concisely in closed form as a truncated Fourier series. Such a formulation is very helpful because it exposes the class of functions that the quantum model can implement, and we can thus use the formalism of statistical learning not only to define the hypothesis class implemented by data-reuploading circuits, but also derive statements about their generalisation performances. Indeed, in this section we give an example of how concepts from classical statistical learning can be applied to characterise quantum learning models, thus hinting at the fruitful exchange that there can be between these two fields and consequently highlighting the multidisciplinary nature of subjects like quantum machine learning.

Let $\rho_{\boldsymbol{\theta}}(\mathbf{x}) = U_{\boldsymbol{\theta}}(\mathbf{x})|\mathbf{0}\rangle\langle\mathbf{0}|U_{\boldsymbol{\theta}}(\mathbf{x})^\dagger$ be the quantum state generated by a data-reuploading quantum circuit satisfying the assumptions of Th. 3.1, and O the observable estimated on such state. The hypothesis class implemented by such quantum neural network is

$$\mathcal{M}_{\text{QNN}} = \left\{ \mathbf{x} \mapsto f(\mathbf{x}) = \text{Tr}[O\rho_{\boldsymbol{\theta}}(\mathbf{x})] = \sum_{\boldsymbol{\omega} \in \Omega} c_{\boldsymbol{\omega}} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \mid \{c_{\boldsymbol{\omega}}\}_{\boldsymbol{\omega}} \text{ such that } |f(\mathbf{x})| \leq \|O\|_{\infty} \right\}, \quad (3.67)$$

where the constraint on the coefficients is a consequence of the output of the circuit being the expectation value of an observable, which is upper bounded by its largest eigenvalue

$$|\langle O \rangle| = |\text{Tr}[O\rho]| \leq \|\rho\|_1 \|O\|_{\infty} = \|O\|_{\infty} \quad (3.68)$$

where we used Hödler’s inequality to upper bound the expectation value in term of the *Shatten p -norms* $\|\cdot\|_p$ of the operators⁹ [30, 315], and $\|\rho\|_1 = 1$ because it is a density matrix, and $\|O\|_{\infty} := \max\{|o_i| \mid O = \sum_i o_i |o_i\rangle\langle o_i|\}$ is the maximum absolute eigenvalue of the observable.

⁹Given an operator A , its Shatten p -norm is equal to the vector p -norm of its singular values $\|A\|_p = \|\mathbf{s}(A)\|_p$ where $\mathbf{s}(A)$ is the vector of the singular values of A . A direct proof of the bound (3.68) can also be obtained without resorting to operator norms, but by direct computation as in eq. (2.31), namely

$$|\langle O \rangle| = |\text{Tr}[O\rho]| = \left| \sum_i o_i \underbrace{\langle o_i | \rho | o_i \rangle}_{\geq 0, \rho \geq 0} \right| \leq \sum_i |o_i| \langle o_i | \rho | o_i \rangle \leq \max_i |o_i| \underbrace{\sum_i \langle o_i | \rho | o_i \rangle}_{=\text{Tr}[\rho]=1} = \max_i |o_i|. \quad (3.69)$$

As shown and discussed in Appendix B.1, the hypothesis class \mathcal{M}_{QNN} can be further simplified, so much so that it can be interpreted as a classical linear model (3.26) using a specific Fourier-like feature map. Indeed, a data-reuploading QNN can be equivalently rewritten as

$$\mathcal{M}_{\text{QNN}} = \left\{ \mathbf{x} \mapsto f(\mathbf{x}) = \mathbf{w}_\theta \cdot \boldsymbol{\phi}(\mathbf{x}) \mid \mathbf{w}_\theta, \boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^{|\Omega|}, \|\mathbf{w}_\theta\|_2 \leq 2\|O\|_\infty \right\}, \quad (3.70)$$

where $\mathbf{w}_\theta \in \mathbb{R}^{|\Omega|}$ are parameters that depend in a highly non-trivial way on the trainable parameters \mathbf{x} , and $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathbb{R}^{|\Omega|}$ is a feature map that takes an input \mathbf{x} and maps it to a larger vector whose entries are Fourier features of the input, see Eq. (B.32). Essentially, this formulation tells us that data-reuploading quantum circuits can be seen as “linear” models in the Fourier features.

Interestingly, now that the hypothesis class is in the form of a classical “linear” model, one can use the results from classical statistical learning to characterise the generalisation performances of a data-reuploading quantum neural network. Indeed, as proved in Appendix B.1 by adapting results for classical linear models, one can derive generalisation bounds of the form (3.12) also for data-reuploading circuits. In fact, using the concepts and notation introduced earlier in Sec. 3.2.1.5, one can prove the following theorem.

Theorem 3.2 — Generalisation Bound for Quantum Neural Networks (see also Theorem 6 in Ref. (53)). Be $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} \subset [0, 2\pi]^d \times \mathbb{R}$ a data space of inputs and outputs. Consider a data reuploading quantum circuit whose encoding scheme generates an integer-valued spectrum Ω , whose model class is $\mathcal{M}_{\text{QNN}} := \{ \mathbf{x} \mapsto \text{Tr}[\rho(\mathbf{x}; \theta) O] = \sum_{\boldsymbol{\omega} \in \Omega} c_{\boldsymbol{\omega}} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \}$. Be $\ell : \mathbb{R} \times \mathbb{R} \rightarrow [0, c]$ an L -Lipschitz loss function and define $\mathcal{G}_{\text{QNN}} := \{ z = (x, y) \mapsto \ell(hf(x), y) \mid f \in \mathcal{M}_{\text{QNN}} \}$. For any $\delta > 0$ and probability measure \mathcal{D} over \mathcal{Z} , with probability at least $1 - \delta$ over the drawn of a training set $S \in \mathcal{Z}^m$ of size m , for all $g \in \mathcal{G}_{\text{QNN}}$:

$$L_{\mathcal{D}}(g) - L_S(g) < 4\|O\|_\infty L \sqrt{\frac{|\Omega|}{m}} + 3c \sqrt{\frac{\log 2/\delta}{2m}} \quad (3.71)$$

A complete derivation of Theorem 3.2 can be found in Appendix B.1, where we first show how to use Rademacher complexity to prove a generalisation bound for classical linear models, and then show how this result directly translates to data-reuploading quantum neural networks [193]. Various encoding-dependent generalisation bounds, like the one reported in Theorem 3.2, were first proved in ref. [53], where the authors use different measures of complexity (Rademacher Complexity and Covering Numbers) to discuss generalisation performances of data-reuploading circuits. We remark that, even though our derivation is based on the Rademacher complexity to measure the complexity of the model, the derivation reported in Appendix B.1 follows an arguably simpler and more straightforward strategy, based on the realisation that quantum neural networks can be seen as linear models in a Fourier space (3.67).

The gist of the generalisation bound (3.71) is to show that the generalisation error, namely the difference between the expected (test) error and the empirical (training) error scales as

$$\text{Generalisation Error of QNN} \leq \mathcal{O} \left(\sqrt{\frac{|\Omega|}{m}} \right), \quad (3.72)$$

that is, if the Fourier spectrum Ω of the circuit is too large (which corresponds to a high complexity of the model) then there are poor guarantees that the model will generalise. However, as expected, an increased training set size m helps in overcoming the issue.

The generalisation bound (3.71) is specific to data-reuploading quantum circuits that admit a Fourier representation, where a strong accent is posed on the accessible spectrum defined by the data encoding strategy. However, many other results have appeared in the quantum machine literature aiming at characterising the complexity and generalisation performances of quantum models. Examples are works dedicated to the study of the complexity and performances of linear (implicit

or explicit) variational models [86, 121, 141, 170], based on a quantum information-theoretic approach linking the generalisation performances of a quantum model to the mutual information between the data-embedded quantum states and the classical data space [19], or generalisation bounds based on the Fisher information [4]. An extended discussion of such results is far beyond the scope of the present work, but we refer to [53] for a concise summary of recent results on the generalisation of quantum machine learning models.

3.3.4 The power of quantum machine learning

Part, if not all, of the reasons for the great success of classical machine learning, is that these methods perform incredibly well in practice and across a wide variety of domains, even reaching super-human performances in some cases. Although the theory behind learning models is rich, interesting and far from being completely understood, it is the practical benefits that drive the adoption of these learning models for data-intensive tasks.

One would then expect (near-term) quantum machine learning to hold these promises and even surpass its “ordinary” classical counterpart in terms of performance. However, despite much recent effort in exploring this topic, the question regarding the power of quantum machine learning model is still far from being settled, with no clear signs of quantum advantage. However, some investigations point out directions where a quantum advantage of some sort could be attained.

In terms of model complexity, quantum neural networks were shown to be richer than classical neural networks of comparable size [4], thus hinting at a possible way to ensure an advantage. However, not only theoretical analysis on the capacity of learning models does not directly translate to practical benefits, but generalisation bounds of the form (3.12) suggest care when using complex models, as these can incur in generalisation performances. Moreover, we have seen that data reuploading circuits essentially are to truncated Fourier series in disguise, and thus admit a simple and fully classical interpretation. Consequently, authors in ref. [261] underline how quantum advantages with these models can only be achieved at training rather than prediction time, since a *surrogate* classical model mimicking the prediction of the quantum circuit can be constructed efficiently via a discrete Fourier transform¹⁰.

Regarding quantum kernel methods, authors in ref. [141] show that there may be cases where these methods have lower prediction error than their classical counterparts, at least on some engineered datasets, as discussed also in [185]. Specifically, based on the available training data, the authors introduce a geometric test between the kernel functions implemented by the quantum and classical models to check whether there is room for a quantum advantage. If such geometric difference is small, then classical methods will perform similarly or even better. If, on the other hand, the geometric difference is large, then one can construct classification tasks (training data and labels) on which quantum models have better prediction errors.

As for quantum data, that is allowing for the possibility of storing and processing quantum information coming for example from experiments, quantum-native learning models can be shown to perform better than classical ones [59, 140, 141, 143], even though classical algorithms learning on data can still perform well, better than classical non-learning procedures, when given access to quantum datasets [141, 142]. At last, regarding quantum machine learning applications for classical data on future fault-tolerant quantum computers, there are hopes that QML could provide polynomial speedups for algebra-based subroutines [59].

Researchers should not be discouraged by the lack of clear advantages, both because the study

¹⁰This result can be seen as an application of *Nyquist–Shannon sampling theorem* to reconstruct a continuous periodical signal given a discrete set observations. The output of a data-reuploading circuit is a truncated Fourier series (3.47), which is a periodic signal of the inputs composed of multiple waves having different frequencies. By sampling the signal—that is, measuring the output of the circuit—at inputs which are distant less than $1/2\omega_{\max}$, where ω_{\max} is the largest frequency in the signal, then one can reconstruct classically the complete action of the circuit. The procedure is efficient as long as the maximum frequency in the spectrum of the circuit scales polynomially with the number of qubits, even though this is not always the case [232, 276].

of quantum machine learning is inherently interesting from a theoretical point of view [265], and because the field is still in its early stages and there is much work to be done to fully understand and harness the power of quantum algorithms for machine learning. As was the case for classical machine learning, it took many decades and several “winter periods” to transform the first proposals of learning algorithms in the 1950 [255] into today’s surprisingly powerful artificial intelligence models. With the strong hope for a shorter incubation period, in the coming years it will be interesting to see how the field develops and if quantum machine learning can live up to its potential.

3.4 Conclusions

In this chapter, we have gone through a long journey about classical and quantum learning models. We have started with a general definition of quantum machine learning, the topic of the chapter and this entire thesis work, and we argued that different declinations of this field exist, thus explaining the four-fold way of QML 3.1. Moreover, we underlined our focus here is on near-term applications of quantum machine learning based on variational quantum algorithms, especially for analysing classical data.

We then moved to laying the basics of classical machine learning, introducing the main concepts and tools and discussing two common models, namely linear models and kernel methods, and neural networks. With these tools, we proceeded to the main part of this chapter where we discussed examples of quantum machine learning models, showing how they relate to their classical counterparts. Moreover, we showed how results from classical statistical learning theory can be applied also to quantum models, which is a very helpful tool to characterise their performances. At last, we concluded with a birds-eye view of state-of-the-art QML, talking over how and where quantum computers could bring advantages for learning tasks.

In the following chapters, we will present explicit examples of variational algorithms implementing machine learning models, starting from models of quantum neurons and also showing how these can be applied to analyse data from real use cases. Then, we will discuss the entanglement properties of common parameterized quantum neural networks clinging on the relation between randomness and simulability of the circuit, and finally mention a technique that is not directly related to quantum machine learning, but can be used to mitigate the noise happening in quantum measurement procedures.



4. Quantum computing model of an artificial continuous neuron

4.1	Introduction	81
4.2	Continuously valued quantum neuron model	82
4.2.1	Some properties: colour invariance and noise resilience	84
4.2.2	Quantum circuit model of a continuously valued perceptron	85
4.3	Results	87
4.3.1	Testing the quantum neuron for image recognition tasks	88
4.4	Training the quantum neuron	88
4.4.1	Classification tasks	90
4.4.2	MNIST dataset	92
4.5	Conclusions	93

In this chapter¹ we discuss a newly introduced model for a quantum perceptron, that is a quantum algorithm mimicking the behaviour of a classical neuron, the building block of artificial neural networks, see Sec. 3.2.2.2. Specifically, we show how the design for the implementation of a previously introduced quantum artificial neuron [293], which fully exploits the use of superposition states to encode binary valued input data, can be further generalised to accept continuous- instead of discrete-valued input vectors, without increasing the number of qubits. This further step is crucial to allow for a direct application of gradient descent-based learning procedures, which would not be compatible with binary-valued data encoding.

4.1 Introduction

Quantum computers hold the promise to greatly enhance the computational power of not-so-distant in future computing machines [16, 236]. In particular, improving machine learning techniques by means of quantum computers is the essence of the thriving field of Quantum Machine Learning, as discussed in Sec 3.1. Several models for the quantum computing version of artificial neurons have been proposed [50, 163, 168, 268, 293, 302, 319], together with novel quantum machine learning techniques implementing classification tasks [131, 262, 264], quantum autoencoders [171, 253], quantum convolutional networks [66, 134] and quantum Boltzmann machines [7, 330] to

¹The content of this chapter is based on the author's work [195], and all the figures in this chapter are taken from, or are adaptations of, the figures present in such work.

give a non-exhaustive list. In this context, quantum signal processing leverages the capabilities of quantum computers to represent and elaborate exponentially large arrays of numbers, and it could be used for enhanced pattern recognition tasks, going beyond the capabilities of classical computing machines [159]. In these regards, the development of artificial Neural Networks dedicated for quantum computers [267] is of fundamental importance, due to the preponderance of this type of classical algorithms in image processing [252].

In the commonly accepted terminology of graph theory, (feed-forward) neural networks are directed acyclic graphs (DAG), that is a collection of nodes where information flows only in one direction, without any loop, as shown previously in Fig. 3.3. Each node is called artificial neuron, since it represent a very simplistic mathematical model for a natural neuron, and consists of an object that takes some input data, processes them using some internal parameters (*weights*), and eventually gives an output value. In their simplest form, these are called McCulloch-Pitts neurons [203] and only deal with binary values, while in the most common and most useful form, named Perceptrons [255], they accept real, continuously-valued inputs and weights.

Continuous inputs are not possible in conventional digital computers, and these are usually represented using bitstrings: for instance, a grey scale image pixel is rendered with integer numbers ranging from 0 to 255 using 8-bit binary strings. Some approaches propose to use a similar representation in quantum computers by assigning several qubits per value [175, 177, 183]. However, these approaches are particularly wasteful, especially in light of the fact that quantum mechanical wavefunctions can be inherently represented as continuously valued vectors.

Recent work has introduced a model for a quantum circuit that mimics a McCulloch-Pitts neuron [293], and in this chapter we generalise this model to the case of a quantum circuit that also accepts continuous-value input vectors. We thus present a model for a continuous quantum neuron which, as we shall see, can be used for pattern recognition in grey-scale images without the need to increase the number of qubits to be employed. This represents a further memory advantage with respect to classical computation, where an increase in the number of encoding bits is required to deal with continuous numbers. We employ a phase-based encoding, and show that it is particularly resilient to noise.

Differently from classical perceptron models, artificial quantum neurons as described, e.g., in Ref. [293] can be used to classify linearly non separable sets. In the continuously valued case, we thus harness the behaviour of our quantum perceptron model to show its ability to correctly classify several notable cases of linearly non separable sets. Furthermore, we test this quantum artificial neuron for digit recognition on the MNIST dataset [101], with remarkably good results. We further stress that the present generalisation of the binary-valued artificial neuron model is a crucial step towards the use of gradient descent-based optimisation techniques (see Eq. (3.10)) that cannot be applied to the oversimplified integer-valued McCulloch-Pitts neuron model.

4.2 Continuously valued quantum neuron model

Let us consider a perceptron model with real-valued input and weight vectors, respectively indicated with $\mathbf{i} = (i_0, \dots, i_{d-1})$ and $\mathbf{w} = (w_0, \dots, w_{d-1})$, with $i_k, w_k \in \mathbb{R}$. A schematic representation of a classical perceptron model is depicted in Fig. 4.1, whereas its mathematical formulation was already introduced previously in Eq. (3.35), and also discussed in detail for the case of binary McCulloch-Pitts neurons in Sec. 5.2 in the next chapter.

Similarly, we define a model of a quantum neuron capable of accepting continuously valued input and weight vectors, by extending a previous proposal for the quantum computing model of an artificial neuron only accepting binary valued input data [293], of which we give an extended review in the next Chapter 5. In order to encode data on a quantum state, we make use of a phase encoding. Given an input $\boldsymbol{\theta} = (\theta_0, \dots, \theta_{d-1})$ with $\theta_i \in [0, \pi]$, consisting of the classical data to be

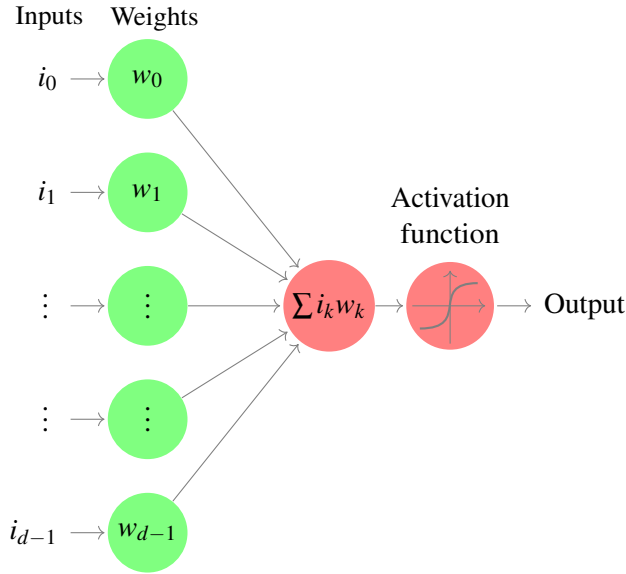


Figure 4.1: Scheme of a classical perceptron model. The artificial neuron evaluates a weighted sum between the input vector, \mathbf{i} , and the weight vector, \mathbf{w} , followed by an activation function which determines the actual output of the neuron, see Eq. (3.35).

analyzed, we consider the vector:

$$\mathbf{i} = \left(e^{i\theta_0}, e^{i\theta_1}, \dots, e^{i\theta_{d-1}} \right), \quad (4.1)$$

which we will be referring to as the input vector in the following. For data not lying in the interval $[0, \pi]$ but more generally in $[a, b]$, a normalisation scheme can be used to transform the data in the appropriate range, for example $\theta_i \rightarrow \pi(\theta_i - a)/(b - a)$. Explicit examples will be given later. With the input vector in Eq. (4.1), we define the corresponding input quantum state of $n = \log_2 d$ qubits

$$|\psi_i\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} i_k |k\rangle, \quad (4.2)$$

where the states $|k\rangle$ denote the computational basis states of n qubits ordered by increasing binary representation, namely $|00\dots 0\rangle, |00\dots 1\rangle, \dots, |11\dots 1\rangle$. Since we are dealing with an artificial neuron, we have to encode another vector, namely that of the weights $\boldsymbol{\varphi} = (\varphi_0, \dots, \varphi_{d-1})$ with $\varphi_i \in [0, \pi]$, and the corresponding vector

$$\mathbf{w} = \left(e^{i\varphi_0}, e^{i\varphi_1}, \dots, e^{i\varphi_{d-1}} \right) \quad (4.3)$$

which in turn defines the weight quantum state

$$|\psi_w\rangle = \frac{1}{2^{n/2}} \sum_{k=0}^{2^n-1} w_k |k\rangle. \quad (4.4)$$

Note that (4.2) and (4.4) have the same structure, in that they consist of an equally weighted superposition of all the computational basis states, although with varying phases. By means of such encoding scheme, we can fully exploit the exponentially large dimension of the n qubits Hilbert space, i.e., by only using n qubits it is possible to encode and analyse data of dimension $d = 2^n$. Due to global phase invariance, the number of actual independent phases is $2^n - 1$, but this does not spoil the overall efficiency of the algorithm, as we will see. We also note that states of the form

$\frac{1}{2^{n/2}} \sum_i e^{i\alpha_i} |i\rangle$, as those in (4.2) and (4.4), are known as locally maximally entanglable (LME) states, as introduced in [169].

Having defined the input and weight quantum states, their similarity is estimated by considering the inner product

$$\langle \psi_w | \psi_i \rangle = \frac{1}{2^n} \sum_{k,j=0}^{2^n-1} i_k w_j^* \langle j|k \rangle = \frac{1}{2^n} \mathbf{i} \cdot \mathbf{w}^* = \frac{1}{2^n} \left(e^{i(\theta_0 - \varphi_0)} + \dots + e^{i(\theta_{2^n-1} - \varphi_{2^n-1})} \right), \quad (4.5)$$

which corresponds to evaluating the scalar product between the input vector in Eq. (4.1) and the complex conjugate of the weight vector \mathbf{w}^* in Eq. (4.3), thus implementing a similar processing of the classical perceptron algorithm. Since probabilities in quantum mechanics are represented by the squared modulus of wavefunction amplitudes, we consider $|\langle \psi_w | \psi_i \rangle|^2$, which can be calculated explicitly as (see App. C.1):

$$|\langle \psi_w | \psi_i \rangle|^2 = \frac{1}{2^n} + \frac{1}{2^{2n-1}} \sum_{i < j}^{2^n-1} \cos((\theta_j - \varphi_j) - (\theta_i - \varphi_i)). \quad (4.6)$$

One can easily check that $|\langle \psi_w | \psi_i \rangle|^2 = 1$ for $\theta_i = \varphi_i \forall i$, since the two states would coincide in such case. The trigonometric formula in Equation (4.6) represents the *activation function* implemented by the proposed quantum neuron. Even if it does not remind any of the activation functions conventionally used in classical machine learning techniques, such as the Sigmoid or ReLu functions shown in (3.39), its nonlinearity suffices to accomplish classification tasks, as we will discuss in the following sections.

4.2.1 Some properties: colour invariance and noise resilience

From Eq. (4.6), we define the activation function of the quantum artificial neuron as

$$f(\boldsymbol{\theta}, \boldsymbol{\varphi}) = |\langle \psi_w | \psi_i \rangle|^2. \quad (4.7)$$

Keeping the weight vector $\boldsymbol{\varphi}$ fixed, suppose two different input vectors are passed to the quantum neuron, namely $\boldsymbol{\theta}$ and $\boldsymbol{\theta}' = \boldsymbol{\theta} + \boldsymbol{\Delta}$, with $\boldsymbol{\Delta} = (\Delta_1, \dots, \Delta_n)$. One can easily check that whatever the value of $\boldsymbol{\Delta}$, both input vectors will give rise to the same activation function, that is $f(\boldsymbol{\theta}, \boldsymbol{\varphi}) = f(\boldsymbol{\theta}', \boldsymbol{\varphi})$. Hence, two input vectors only differing by a constant, albeit real valued, quantity will be equally classified by such model of quantum perceptron. Hence, in the context of image classification, we can state that the present algorithm has a built-in colour translational invariance. This should not come as a surprise, since the activation function actually depends of the *differences* between phases. In fact, the artificial neuron tends to recognise as similar any dataset that displays the same overall differences, instead of perfectly coincident states.

Next, we assume that the input and weight vectors do coincide, but only up to some noise corrupting the input vector, such that $\boldsymbol{\theta} = \boldsymbol{\varphi} + \boldsymbol{\Delta}$, where $\boldsymbol{\Delta} = (\Delta_0, \Delta_1, \dots, \Delta_{2^n-1})$ represents the small variations, now assumed to be different on each coordinate. Substituting the above values in Eq. (4.7), we obtain

$$f(\boldsymbol{\theta}, \boldsymbol{\varphi}) = f(\boldsymbol{\Delta}) = \frac{1}{2^n} + \frac{1}{2^{2n-1}} \sum_{i < j}^{2^n-1} \cos(\Delta_j - \Delta_i). \quad (4.8)$$

For simplicity of calculation, assume the noise factors are distributed according to a uniform distribution in the interval $\Delta_i \sim \text{Unif}[-a/2, a/2]$, $a \in \mathbb{R}$. Then, the activation function averaged over the probability distribution of Δ_i can be calculated as (see App. C.2)

$$\mathbb{E}_{\boldsymbol{\Delta}}[f(\boldsymbol{\Delta})] = \frac{1}{2^n} + \frac{2^n - 1}{2^{2n-1}} \left(\frac{1 - \cos(a)}{a^2} \right). \quad (4.9)$$

Input data lie in the interval $[0, \pi]$, thus a reasonable noise is of the order of a some small fraction of π , which implies $a < 1$. Specifically, in the case of small noise, Eq. (4.9) reduces to

$$\mathbb{E}_\Delta[f(\Delta)] = 1 - \frac{2^n - 1}{2^n} \frac{a^2}{12} + \mathcal{O}(a^4) \quad \text{for } a \ll 1. \quad (4.10)$$

Thus, the output of the quantum neuron is only slightly perturbed by the presence of noise corrupting an input vector which would otherwise have a perfect activation. As shown in Appendix C.2, a similar result can be derived for any input vectors, not only those having perfect activation, and also for Gaussian —instead of uniform— noise. In this respect, one can find a more recent and comprehensive analysis on the effect of Gaussian noise on the parameters of variational quantum algorithms in Ref. [285].

Having outlined the main steps defining the quantum perceptron model for continuously valued input vectors, we now proceed to build a quantum circuit that allows implementing it on a qubit-based quantum computing hardware.

4.2.2 Quantum circuit model of a continuously valued perceptron

A quantum circuit implementing the quantum neuron model described above is schematically represented in Fig. 4.2. It consists of three main parts: the first section of the circuit, denoted as U_i , transforms the initial quantum state $|\mathbf{0}\rangle = |0\rangle^{\otimes n}$ to the input quantum state $|\psi_i\rangle$ defined in Eq. (4.2); then the operation U_w performs the inner product of Eq. (4.5) between the input and weight quantum state; and finally a multi-controlled CNOT gate targeting an ancillary qubit is used to extract the final result of the computation, namely Eq. (4.6). We now proceed by explaining in detail how each of these transformations can be implemented in practice.

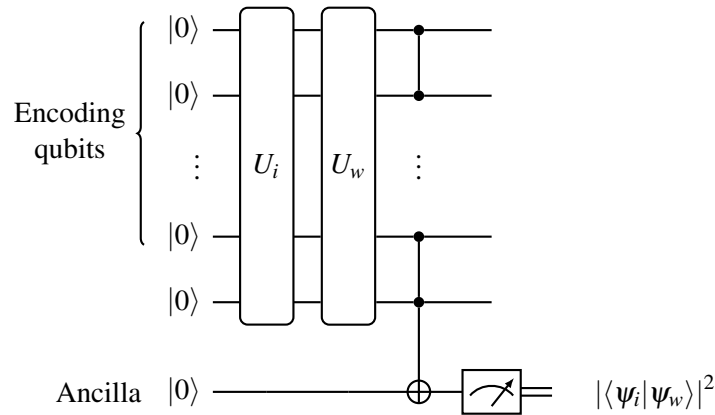


Figure 4.2: Quantum circuit model of a perceptron with continuously valued input and weight vectors.

Data encoding The U_i operation creates the quantum input state $U_i |\mathbf{0}\rangle = |\psi_i\rangle$ (4.2), and it can be implemented by means of a brute-force approach. First of all, we apply a layer of Hadamard gates, $H^{\otimes n}$, which creates the balanced superposition state $H^{\otimes n} |\mathbf{0}\rangle = |+\rangle^{\otimes n}$, with $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$. The quantum state $|+\rangle^{\otimes n}$ consists of the equally weighted superposition of all the states in the n -qubits computational basis, hence we can target each of them and add the appropriate phase to it in order to obtain the desired result (4.2). Such transformation is implemented by the diagonal (in the computational basis) unitary matrix

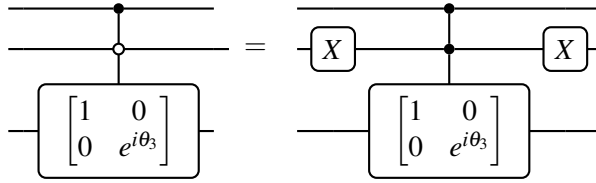
$$U(\boldsymbol{\theta}) := \begin{bmatrix} e^{i\theta_0} & 0 & \dots & 0 \\ 0 & e^{i\theta_1} & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & e^{i\theta_{2^n-1}} \end{bmatrix}, \quad (4.11)$$

whose action is to phase shift each state of the computational basis as $|i\rangle \rightarrow e^{i\theta_i} |i\rangle$, with phases $\theta_i \in \mathbb{R}$, that are (in general) independent from each other. One can decompose such overall unitary in smaller pieces

$$U(\boldsymbol{\theta}) = \prod_{i=0}^{2^n-1} \tilde{U}_i(\theta_i), \quad (4.12)$$

where each $U_i(\theta_i)$ acts as $\tilde{U}_i(\theta_i) |i\rangle = e^{i\theta_i} |i\rangle$, while leaving all the other states in the computational basis unchanged. These unitaries can be constructed with an appropriate combination of Pauli- X gates and a multi-controlled phase shift gates, $C^{n-1}P(\theta)$, where the phase shift $P(\theta) = \text{diag}(1, e^{i\theta})$, was already introduced earlier in Tab. 2.1 while discussing single-qubit gates.

For example, suppose having $n = 3$ qubits, and consider the state $|101\rangle$ to be phase shifted to $e^{i\theta_3} |101\rangle$. Such transformation can be achieved with the following quantum circuit



whose action is indeed $\tilde{U}_3(\theta_3) |101\rangle = e^{i\theta_3} |101\rangle$, while leaving all other states of the computational basis untouched. Iterating a similar gate sequence for each state of the computational basis $\{|i\rangle\}$, one eventually obtains the desired overall unitary operation $U(\boldsymbol{\theta})$ (4.11).

Summarising, we have shown how to build the data encoding quantum circuit U_i that creates the quantum state $|\psi_i\rangle$ (4.2) by means of the operation $U_i|\mathbf{0}\rangle := U(\boldsymbol{\theta})H^{\otimes n}|\mathbf{0}\rangle = |\psi_i\rangle$, where the parameters $\boldsymbol{\theta}$ are the input classical data to be analysed (4.2).

Inner product The unitary U_w can then be constructed in a similar fashion. First, one has to notice that the inner product $\langle \psi_w | \psi_i \rangle$ resides in the overlap between the quantum state $|\varphi_{i,w}\rangle := (U(\boldsymbol{\varphi})H^{\otimes n})^\dagger |\psi_i\rangle$ and the ground state $|\mathbf{0}\rangle$. In fact, by definition of $U(\boldsymbol{\varphi})$ in Eq. (4.11), it holds that $U(\boldsymbol{\varphi})H^{\otimes n}|\mathbf{0}\rangle = |\psi_w\rangle$ and thus the scalar product is given as

$$\langle \mathbf{0} | \overbrace{(U(\boldsymbol{\varphi})H^{\otimes n})^\dagger |\psi_i\rangle}^{|\varphi_{i,w}\rangle} = \underbrace{\langle \mathbf{0} | H^{\otimes n} U(\boldsymbol{\varphi})^\dagger |\psi_i\rangle}_{\langle \psi_w |} = \langle \psi_w | \psi_i \rangle. \quad (4.13)$$

Then, in order to extract the result, a final layer of Pauli- X gates are applied to all encoding qubits, such that the desired coefficient now multiplies the state $|1\rangle$ instead of $|\mathbf{0}\rangle$, namely

$$X^{\otimes n} |\varphi_{i,w}\rangle = |\tilde{\varphi}_{i,w}\rangle = \sum_{k=0}^{2^n-2} c_k |k\rangle + c_{2^n-1} |11\dots 1\rangle \quad \text{with} \quad c_{2^n-1} = \langle \psi_w | \psi_i \rangle. \quad (4.14)$$

Thus, by combining (4.13) and (4.14), one finds that the transformation U_w of Fig. 4.2 actually consists in the quantum operations $U_w := X^{\otimes n} H^{\otimes n} U(\boldsymbol{\varphi})^\dagger$.

Measurement-induced activation function By means of a multi-controlled C^n NOT, one can load the coefficient of interest c_{2^n-1} on an ancillary qubit as follows

$$C^n \text{NOT} |\tilde{\varphi}_{i,w}\rangle \otimes |0\rangle_{\text{ancilla}} = \sum_{k=0}^{2^n-1} c_k |k\rangle \otimes |0\rangle_{\text{ancilla}} + c_{2^n-1} |11\dots 1\rangle \otimes |1\rangle_{\text{ancilla}}. \quad (4.15)$$

In fact, a final measurement of the ancilla qubit will yield result $|1\rangle$, interpreted as a *firing* neuron, with probability $p_{\text{out}} = |c_{2^n-1}|^2 = |\langle \Psi_w | \Psi_i \rangle|^2 = |\mathbf{i} \cdot \mathbf{w}^*|^2 / (2^{2n})$, which is the quantum neuron activation function we introduced in Eq. (4.6).

We now make a few remarks before proceeding with the application of the proposed neuron model to practical learning tasks. First of all, we note that the input vector $\mathbf{i} \in \mathbb{C}^d$ (4.2) contains $d = 2^n$ elements, while only $n + 1$ qubits are required to implement the quantum neuron circuit in Fig. 4.3. Moreover, the additional ancillary qubit can be easily removed by performing a joint measurement on all n qubits in the state $|\varphi_{i,w}\rangle$ of Eq. (4.13), and now considering the probability of measuring $|0\rangle$ instead. However, since machine learning techniques yield their full potential when used with connected structures of multiple single neurons, and having in mind the idea of implementing a quantum version of a feedforward neural network, it is essential to have a model for which information can be easily transferred from a neuron to another. This can be accomplished by using an ancilla qubit per artificial neuron, as discussed in [295].

Regarding the time complexity, the number of operations required by this quantum circuit scales linearly with the dimension of the input vectors, but exponentially with the number of qubits. Indeed, the quantum circuit introduced above requires $\mathcal{O}(d) \equiv \mathcal{O}(2^n)$ operations to implement all the phase shifts necessary to build the LME states of Eq. (4.2). Depending of the relation between the input data, $\theta_i \in \boldsymbol{\theta} \in \mathbb{R}^d$, other preparation schemes involving less operations could be devised [169], and we refer to Appendix C.3 for a more in-depth discussion of this topic.

Finally, it is worth noticing that due to the global phase invariance of quantum states and probabilities, the activation function in Eq. (4.6) can be recast as

$$|\langle \Psi_w | \Psi_i \rangle|^2 = \frac{1}{2^{2n}} \left| \sum_{i=0}^{2^n-1} e^{i(\theta_i - \varphi_i)} \right|^2 = \frac{1}{2^{2n}} \left| 1 + \sum_{i=1}^{2^n-1} e^{i(\tilde{\theta}_i - \tilde{\varphi}_i)} \right|^2, \quad (4.16)$$

with $\tilde{\theta}_i = \theta_i - \theta_0$, $\tilde{\varphi}_i = \varphi_i - \varphi_0$, $\forall i \geq 1$. By exploiting this redefinition of the parameters, it is possible to implement the same activation function but employing fewer gates, and it is equivalent to leaving the state $|0\rangle$ unchanged without associating any phase to it. Depending on the actual quantum hardware and data, further simplifications to the circuit could be obtained at compiling time. In Fig. 4.3, the scheme of a quantum circuit implementing the artificial neuron model is shown for the specific case involving $n = 2$ qubits.

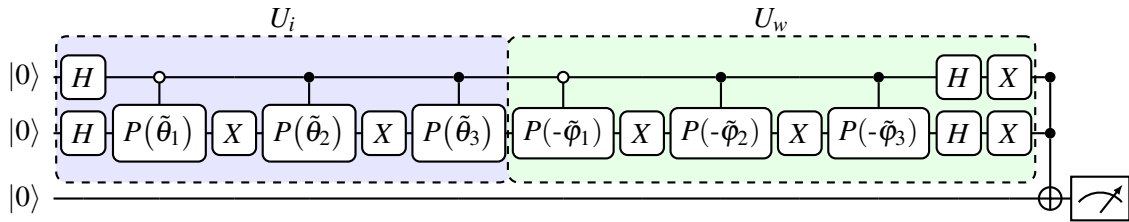


Figure 4.3: Circuitual implementation of the continuous quantum neuron with $n = 2$ qubits. The parameters are redefined as $\tilde{\theta}_i = \theta_i - \theta_0$, $\tilde{\varphi}_i = \varphi_i - \varphi_0$, as detailed in Eq. (4.16).

4.3 Results

The quantum neuron model introduced above is particularly suited to perform classification tasks involving gray-scale images. A gray-scale image consists of a grid of pixels whose intensities are usually represented by integer numbers in the range $[0, 255]^2$, as shown below in Fig. 4.4.

²This encoding of gray-scale images employs a single byte (i.e., 8 bits) per pixel on a classical computing register.

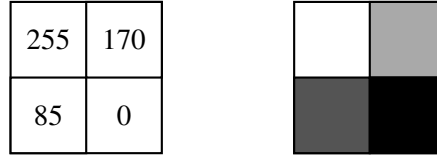


Figure 4.4: A grey-scale image with corresponding pixel intensities. This image can be encoded in the array $(255, 170, 85, 0)$, ordered from top-left to bottom-right. Highest value of intensity (255) corresponds to a white pixel, while black pixels are associated to minimum (0) intensity.

Since we make use of a phase encoding, all inputs (and weights) of the artificial neuron should be normalised in an appropriate angular range, such as $[0, 2\pi]$. However, in this work we further restrict this domain for two important reasons. First, the values in $[0, \pi]$ and $[\pi, 2\pi]$ are equivalent due to the periodicity in phase and the squared modulus in Eq. (4.6). Secondly, for the same reason, states with phase $\varphi = 0$ or $\varphi = \pi$ yield the same activation function, which in turn means that images with inverted colours (obtained by exchanging white with black) would be recognised as equivalent by this perceptron model. Hence, to distinguish a given image from its negative, we further restrict the input and weight elements to lie in the range $[0, \pi/2]$. Thus, an image such as the one reported in Fig. 4.4 is subject to the normalisation $(255, 170, 85, 0) \rightarrow \boldsymbol{\theta} = \frac{\pi/2}{255}(255, 170, 85, 0)$, before using it as an input vector to be encoded into the quantum neuron model.

We implemented and tested the quantum perceptron model both on simulators using Qiskit [5] and also on superconducting real quantum hardware provided by IBM [145], and we now presents the results in what follows.

4.3.1 Testing the quantum neuron for image recognition tasks

To better appreciate the potentialities of the continuously valued quantum neuron, we analyse its performance in recognising similar images. Specifically, we fix the weight vector to $\boldsymbol{\varphi}_{\text{target}} = (\pi/2, 0, 0, \pi/2)$ corresponding to the checkerboard pattern represented in the left panel of Fig. 4.5, and then generate a few random images to be used as inputs to the quantum neuron.

For each input, the circuit is executed with $n_{\text{shots}} = 8192$ measurement shots to gather enough statistics to recover an accurate estimation of the activation function (4.15). With $m = 30$ random generated images, the results of the classification are depicted in Fig. 4.5, which includes the analytic results, the results of numerical simulations obtained with the Qiskit QASM Simulator that simulates stochastic measurement outcomes, and finally the results obtained by executing the quantum circuits on the `ibmqx2-yorktown` real device (accessed in March 2020).

The images producing the largest activation are the ones corresponding to input vectors similar to the checkerboard-like weight vector, thus confirms the desired behaviour of the quantum neuron in recognising similar images. On the contrary, the images with lowest activation are similar to the negative of the target weight vector, as desired. Due to noise in the actual quantum processing device (see Sec. 2.1.6), the statistics of the outcomes from real experiments differ from either the simulated one and the analytic result. Nevertheless, the same overall behaviour can be easily recognised, thus showing that the quantum neuron model can be successfully implemented also on current quantum processors with reliable results, at least for such image recognition task.

4.4 Training the quantum neuron

As we extensively discussed in the previous chapters, in the machine learning jargon the process of finding the appropriate value for the weights to implement a given task is called *training* (or *learning*), and it is generally based upon an optimization procedure in which a cost function is minimised by some gradient descent technique (3.10). Ideally, the minimum of the cost function corresponds to the targeted solution.

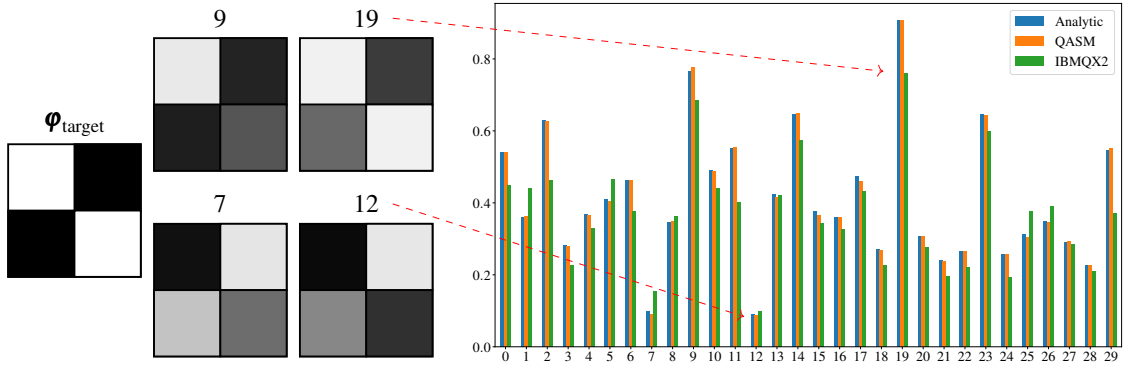


Figure 4.5: Results of the image recognition task performed by a quantum artificial neuron, obtained by running the corresponding quantum circuit with either the Qiskit QASM Simulator backend, and with the `ibmqx2-yorktown` real quantum processor, and we also report the corresponding analytical values. The target weight vector $\boldsymbol{\varphi}_{\text{target}}$ is fixed, and $m = 30$ random images are given as input vectors to the quantum artificial neuron. For each input, the corresponding quantum circuit is executed $n_{\text{shots}} = 8192$ times. The checkerboard-like image corresponds to the target weight vector $\boldsymbol{\varphi}_{\text{target}} = (\pi/2, 0, 0, \pi, 2)$, while the images displaying respectively largest and lowest activation are the ones labelled as ‘19’ and ‘12’. Input vectors labelled as ‘9’ and ‘7’ are examples of images with high and low activation, respectively.

A simple learning task for the proposed quantum neuron is to recognise a single given input. That is, starting from an input vector $\boldsymbol{\theta}_{\text{target}}$, we aim at finding a weight vector $\boldsymbol{\varphi}$ yielding an high activation. A naive yet efficient choice for the loss function driving the learning process is $L(\boldsymbol{\varphi}) = (1 - f(\boldsymbol{\theta}_{\text{target}}, \boldsymbol{\varphi}))^2$, where $f(\boldsymbol{\theta}_{\text{target}}, \boldsymbol{\varphi})$ is the activation function of the artificial neuron as in Eq. (4.7), $\boldsymbol{\theta}_0$ is the input vector, and $\boldsymbol{\varphi}$ is the trainable vector of the weights. Clearly, the minimum of the loss $L(\boldsymbol{\varphi}_{\text{opt}}) = 0$ is achieved when the quantum neuron has full activation, that is $f(\boldsymbol{\theta}_{\text{target}}, \boldsymbol{\varphi}_{\text{opt}}) = 1$. Since the activation function implemented by the quantum perceptron is given in Eq. (4.6), we know that a perfect activation can be obtained when the input and weight vectors are exactly coincident, $\boldsymbol{\varphi}_{\text{opt}} = \boldsymbol{\theta}_{\text{target}}$, although other solutions may do equivalently well.

In our experiments, the minimization process is driven by the Simultaneous Perturbation Stochastic Approximation (SPSA) [287] algorithm, which is built for optimization processes characterized by the presence of noise and is thus particularly effective in the presence of probabilistic measurement outputs. Indeed, we implement all training procedures in the presence of shot noise by simulating the quantum neuron circuits with Qiskit `qasm_simulator`.

In Fig. 4.6 we report the training results for the the task of recognising the input vector $\boldsymbol{\theta}_{\text{target}} = (\pi/5, 0, \pi/3, 0.1)$, whose equivalent grey-scale representation is shown in panel 4.6b. As expected, the cost function is readily minimised by varying the weight vector, and it rapidly converges to values close to zero after a few iteration steps. In this case, the minimum of the loss is achieved at $\boldsymbol{\varphi}_{\text{final}} = (1.03, 0.19, 1.47, 0.61)$, whose grey-scale representation is plotted in Figure 4.6b. Even though the input and weight vector are not numerically equivalent, we see that the final weight image actually looks very much like the target one. In fact, the two images retain almost the same shades of grey, with the optimised one being a bit shifted towards the brightest end of the spectrum. This is in agreement with the discussion in Sec. 4.2.1 regarding colour invariance, where we noticed that the neuron is blind to overall colour shifts.

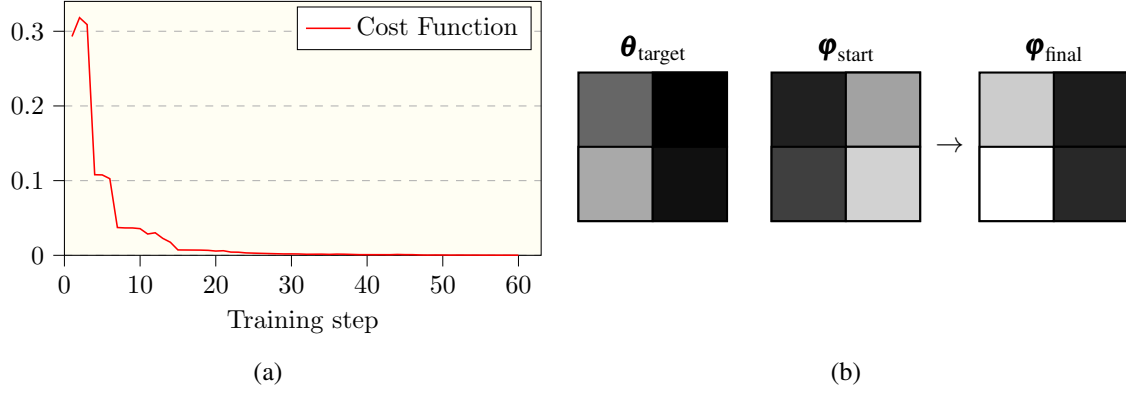


Figure 4.6: Training the quantum neuron to recognise an input pattern. **(a)** Minimisation of the cost function $\mathcal{L}(\boldsymbol{\varphi}) = (1 - f(\boldsymbol{\theta}_{\text{target}}, \boldsymbol{\varphi}))^2$, as a function of the training steps using the SPSA optimiser and simulating the quantum circuits with Qiskit qasm_simulator. **(b)** Images corresponding to the target input vector $\boldsymbol{\theta}_{\text{target}}$, the randomly initialised starting weight vector $\boldsymbol{\varphi}_{\text{start}}$, and the one obtained at the end of the optimisation procedure $\boldsymbol{\varphi}_{\text{final}}$.

4.4.1 Classification tasks

We now move our attention to classification tasks, where we want the neuron to assign a given desired label to an input. Specifically, we consider supervised binary classification problems³, where training inputs $\{\boldsymbol{\theta}_i\}_{i=1}^m$ are associated to binary labels, $\{y_i\}_{i=1}^m$, $y_i \in \{0, 1\}$, and the artificial neuron is asked to reproduce such mapping between inputs and labels.

In order to implement a binary dichotomy with a perceptron model, it is common practice to introduce a *threshold* value t , such that if the activation of the neuron is higher than t , then the output of the neuron is 1, and it is 0 otherwise. Formally, given an input $\boldsymbol{\theta}_i$ and weight vector $\boldsymbol{\varphi}$, the label \hat{y}_i predicted by the quantum neuron is

$$\tilde{y}_i(\boldsymbol{\varphi}) = \begin{cases} 1 & \text{if } f(\boldsymbol{\theta}_i, \boldsymbol{\varphi}) > t \\ 0 & \text{otherwise} \end{cases}. \quad (4.17)$$

where $f(\boldsymbol{\theta}_i, \boldsymbol{\varphi})$ is the activation of the neuron (4.6). Note that the threshold t is actually a *hyperparameter* for our model, and in the following simulations it was heuristically optimized in order to achieve the best classification accuracy.

As with any supervised learning task, the training process is driven by empirical risk minimisation (3.4), where in this case we picked as a loss function an Mean Squared Error like distance between the correct label from the one predicted by the artificial neuron, namely

$$L(\boldsymbol{\varphi}) = \frac{1}{m} \sum_{i=1}^m (y_i - \tilde{y}_i(\boldsymbol{\varphi}))^2, \quad (4.18)$$

where m is the number of samples in the training set, y_i is the correct label associated to input data $\boldsymbol{\theta}_i$, and $\hat{y}_i(\boldsymbol{\varphi})$ is the label predicted by the neuron according to the decision rule (4.17). We now proceed discussing two specific classification tasks on which the quantum neuron was tested.

4.4.1.1 Classification of two-dimensional data

As a first example, we considered a classification problem of the form $\{(\boldsymbol{\theta}_i, y_i)\}_{i=1}^m$, in which inputs $\boldsymbol{\theta}_i = (\theta_1^{(i)}, \theta_2^{(i)}) \in [0, \pi/2]^2$ are two dimensional input data, and $y_i \in \{0, 1\}$ their labels, indicated by red ($y = 0$) and blue ($y = 1$) dots in Fig. 4.7. Since the data are two dimensional we only need a

³This can be generalised in the case of a multi-class classification by adopting a *one versus all* approach.

single qubit to encode the information in the quantum state, and also in this case all simulations were performed using Qiskit `qasm_simulator`, thus taking into account the statistical noise due to finite number of shots.

The panel 4.7a shows the data samples in the training set, while those shown in panel 4.7b are those in the test set, along with the decision boundary that the quantum neuron learnt at the end of the optimisation procedure. The cost function in Eq. (4.18) is minimised using the SPSA optimiser and its behaviour is reported in Fig. 4.7c. Training proceeds towards a minimum of the empirical loss, and, as shown in Fig. 4.7b, at the end of training the optimised neuron implements a perfect classification also on the test set.

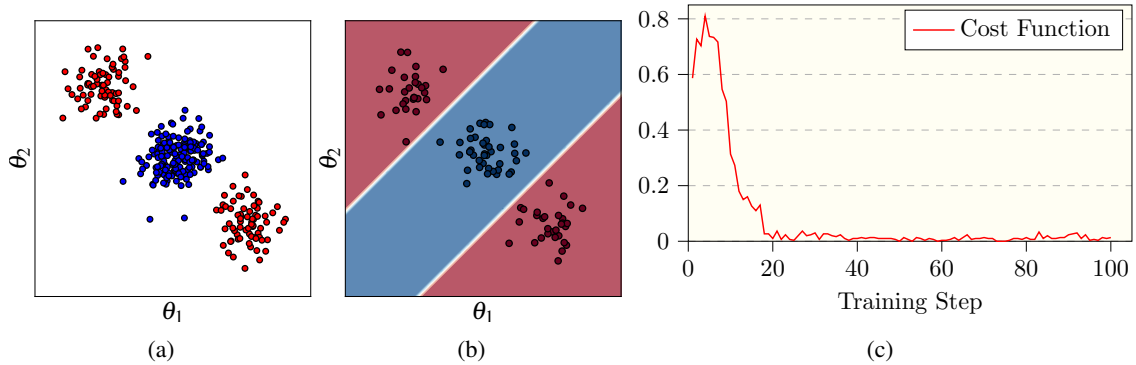


Figure 4.7: Classification of two dimensional data. **(a)** Input data used as training set. **(b)** Test set and decision boundary implemented by the optimised quantum neuron at the end of the learning procedure. The threshold used in the decision rule (4.17) is set to $t = 0.95$. **(c)** Optimisation with the SPSA optimiser run on Qiskit `qasm_simulator`.

4.4.1.2 Classification of non separable data using a bias

We have just seen that a single neuron is capable of classifying some type of two dimensional data, but this procedure is not guaranteed to succeed on more structured dataset. Indeed, for a classification task with the data shown Fig. 4.8a, the single-qubit encoding of the previous model is not enough.

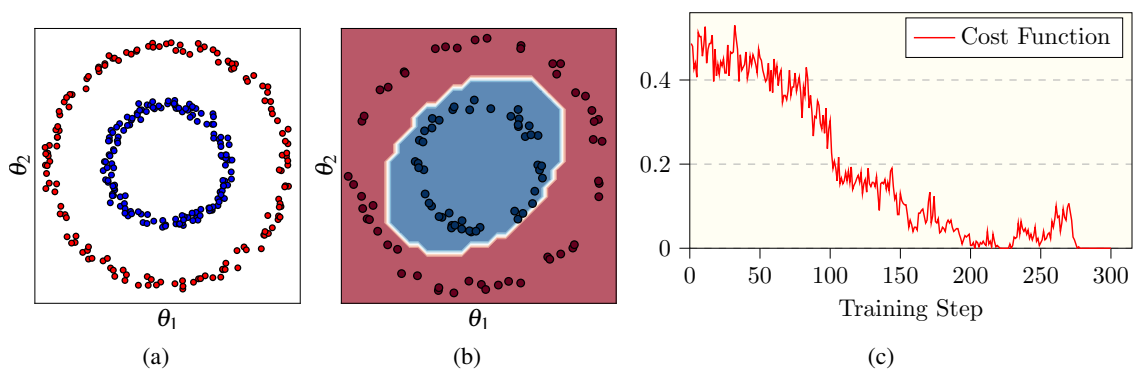


Figure 4.8: Classification of two dimensional circles. **(a)** Input data used as the training set. **(b)** Test set and decision boundary implemented by the optimised neuron at the end of the learning procedure. The threshold used in this example is $t = 0.95$, and the bias $b = 0.25$. **(c)** Optimisation by the SPSA optimiser run on the QASM Simulator. Each iteration in the optimisation procedure was performed calculating the loss over a batch of just 20 samples instead of the full training dataset, which explains why the error is not smooth but presents several spikes.

However, such a problem can be tackled successfully by a quantum neuron using two qubits,

which allows for additional degrees of freedom and thus increased expressivity. In fact, with $n = 2$ qubits it is possible to encode $2^2 = 4$ parameters on the quantum states of interest: one can be kept fixed to zero⁴, two of these are used to encode the actual input data to be analysed, and the last free parameter can be interpreted as a *bias* term, to be tuned accordingly to ensure good performances.

Thus, a convenient encoding scheme is to consider a two-qubit quantum neuron whose input vectors are four-dimensional vectors of the form $\boldsymbol{\theta} = (0, \theta_1, \theta_2, 0)$, and the weight vector is given by $\boldsymbol{\varphi} = (0, \varphi_1, \varphi_2, b)$, where b denotes the bias term, which in our simulations is an hyperparameter that we choose heuristically. In Fig. 4.8 we show the results of training such model by minimising the empirical risk (4.18) on the training data shown in panel (a). After the learning procedure, reported in Fig. 4.8c, the neuron learns a decision boundary that achieves perfect score also on the test set, as shown in Fig. 4.8b.

4.4.2 MNIST dataset

As a concluding example, we briefly discuss how the proposed quantum neuron model could be used to analyse the widely known MNIST dataset [179], which is a set containing 70.000 gray-scale images of digits, from zero to nine. For simplicity, we limit our analysis to only images of zeros and ones, of which we show two examples in the left panel of Fig. 4.9. Note that each image in the MNIST dataset is composed of 28×28 pixels, which is not of the form $2^{n/2} \times 2^{n/2}$ required to be encoded on the quantum state of an artificial neuron of n qubits, admitting $d = 2^n$ input data. Thus, we preprocess the images by adding a number of white pixels so that modified images have dimension 32×32 pixels, which can be given as inputs to a quantum neuron of $n = 10$ qubits.

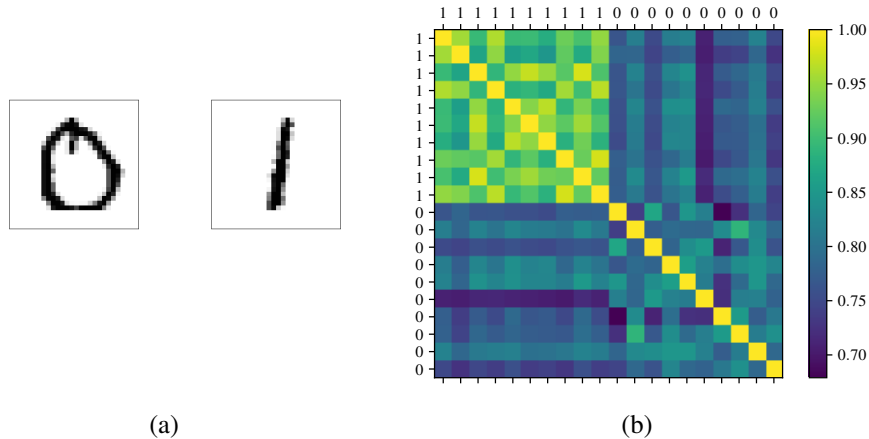


Figure 4.9: Application of the quantum neuron on the MNIST dataset. (a) Examples of a ‘0’ and a ‘1’ drawn from the MNIST dataset. (b) Matrix containing the fidelities of some samples of “one” and “zero” images taken from the MNIST dataset, evaluated with the activation function in Eq. (4.6) and implemented by our quantum neuron model.

The focus of the analysis is to check whether the activation function implemented by the proposed model (4.6) is effective at discriminating between the encoded images of zeros and ones. With this goal in mind, one way to proceed is to fix the weight vector of the artificial neuron to a sample of a ‘1’ selected from the MNIST dataset, and then calculate the activation function of the neuron when other images of zeros and ones are fed as inputs. By using such a quantum neuron with the decision rule in Equation (4.17) with threshold set to $t = 0.85$, the cost function (4.18) evaluated on a set of $m = 2060$ images amounts to $L \sim 0.02$, implying a remarkably good classification

⁴The activation function in Eq. (4.6) only depends on the *differences* between the parameters, thus fixing one of the parameters to a constant value can be thought just as choosing a reference point. In other words, the additional parameter is a global phase that can be ignored.

accuracy of 98%⁵. Moreover, in the right panel of Fig. 4.9, we also report the activation function obtained for several pairs on zeros and ones. According to the artificial neuron, ones are more similar to each other with respect to the zeros, as the fidelities are higher generally much higher (top left corner of the matrix). Even if classical machine learning techniques can easily yield a classification accuracy above 99%, the present results show a remarkable degree of precision, also considering that in this particular example just a single quantum neuron has been used for the classification.

In addition, we also tested a training-based procedure in which each image in the MNIST dataset is first compressed to a 4×4 image by means of a *mean pooling* filter that aggregates values of neighbouring pixels to a single value, and then passed to the neuron. After training, the neuron reaches a best accuracy of about 80%, which, although far from perfect, shows the potential of the activation function implemented by the quantum neuron to be used also for recognition of complex patterns, such as numerical digits.

Our quantum neuron model performs well when compared with other proposed quantum algorithms for the classification of the MNIST dataset. In fact, alternative algorithms have been proposed for this task, some of them using a hybrid classical-quantum approach, such as leveraging well established classical pre/post processing of data through classical machine learning techniques [5, 160]. These hybrid approaches may yield higher (although comparable) classification accuracy when compared to our quantum neuron model. However, we emphasise that in our case the artificial neuron model is fully quantum in nature. When compared to other works [41, 101] using only quantum resources and reporting accuracies of the order of 85% to 98%, our model seems to yield comparable or better results.

4.5 Conclusions

We have reported on a novel quantum algorithm allowing to implement a generalised perceptron model on a qubit-based quantum device that accepts and analyses continuously valued input data. The proposed algorithm is translated into a quantum circuit model to be readily run on existing quantum hardware. It takes full advantage of the exponentially large Hilbert space available to encode input data on the phases of large superposition states, known as locally maximally entangleable (LME) states. These LME states can be constructed with a bottom-up approach, by imprinting each single phase separately. However, it should be stressed that alternative and possibly more efficient strategies could directly yield such states as ground states of suitable Hamiltonians, or as stationary states from dissipative processes [169].

The proposed continuously valued quantum neuron proves to be a good candidate for classification tasks and pattern recognition involving grey-scale images. In particular, the activation function implemented by the quantum neuron yields very high accuracy in the order of 98% when used to discriminate between images of zeros and ones from the MNIST dataset, thus indicating the ability to distinguish also complex patterns. Moreover, thanks to the phase encoding, the neuron can leverage a built-in “colour translational” invariance, as well as significant noise resilience.

A further step would be to consider multiple layers of connected quantum neurons to build a continuous quantum feed-forward neural network, as proposed in [295] for binary-valued quantum neurons. Another important investigation is the design of approximate methods to perform the weight unitary transformation in a way which scales more favourably with the number of encoding qubits, a topic which is at the core of the following Chapter 5. Moreover, in Chapter 6 we will discuss how the application of phase encoding to other quantum machine learning techniques, namely quantum autoencoders.

⁵True and predicted labels are $y_i, \hat{y}_i \in \{0, 1\}$, and so the loss function (4.18) effectively measures the number of misclassified data, that is the number samples for which the true and predicted labels are different.



5. Variational learning for quantum neural networks

5.1	Introduction	94
5.2	A model of quantum artificial neurons	95
5.2.1	Exact implementation with quantum hypergraph states	97
5.3	Variational realisation of a quantum artificial neuron	97
5.3.1	Global variational training	98
5.3.2	Local variational training	99
5.3.3	Case study: pattern recognition	100
5.3.4	Structure of the ansatz and scaling properties	101
5.4	Conclusions	106

In this chapter¹, we first review a series of recent works describing the implementation of artificial neurons and feed-forward neural networks on quantum processors, and subsequently present an original realisation of efficient individual quantum nodes based on variational unsampling protocols. While keeping full compatibility with the overall memory-efficient feed-forward architecture, our constructions effectively reduce the quantum circuit depth required to determine the activation probability of single neurons upon input of the relevant data-encoding quantum states. This suggests a viable approach towards the use of quantum neural networks for pattern classification on near-term quantum hardware.

5.1 Introduction

This chapter is dedicated to the study and improvement of a recently proposed quantum algorithm implementing the activity of binary-valued artificial neurons for classification purposes, whose generalisation to continuous variables was the topic of the previous chapter 4.

Although formally exact, this algorithm for a quantum neuron in general requires quite large circuit depth for the analysis of the input classical data. To mitigate for this effect we introduce a variational learning procedure, based on quantum unsampling techniques, aimed at critically reducing the quantum resources required for its realisation. Indeed, we investigate different learning strategies involving global and local layer-wise cost functions, and we assess their performances also in the presence of statistical measurement noise. By combining memory-efficient encoding

¹The content of this chapter is based on the author's work [296], and all the figures in this chapter are taken from, or are adaptations of, the figures present in such work.

schemes and low-depth quantum circuits for the manipulation and analysis of quantum states, the proposed methods suggest a practical route towards problem-specific instances of quantum computational advantage in machine learning applications.

5.2 A model of quantum artificial neurons

The simplest formalisation of an artificial neuron can be given following the classical model proposed by McCulloch and Pitts [203]. In this scheme, a single node receives a set of binary inputs $\{i_0, \dots, i_{d-1}\} \in \{-1, 1\}^d$, which can either be signals from other neurons in the network or external data. The computational operation carried out by the artificial neuron consists in first weighting each input by a synapse coefficient $w_j \in \{-1, 1\}$ and then providing a binary output $y \in \{-1, 1\}$ denoting either an active or rest state of the node determined by an integrate-and-fire response

$$y = \begin{cases} 1 & \text{if } \sum_j w_j x_j \geq t \\ -1 & \text{otherwise} \end{cases} \quad (5.1)$$

where t represents some predefined threshold.

A quantum procedure closely mimicking the functionality of a binary valued McCulloch-Pitts artificial neuron can be designed by exploiting, on one hand, the superposition of computational basis states in quantum registers, and on the other hand the natural non-linear activation behaviour provided by quantum measurements. In this section, we will briefly outline a device-independent algorithmic procedure [293] designed to implement such a computational model on a gate-based quantum processor. More explicitly, we show how classical input and weight vectors of size m can be encoded on a quantum hardware by using only $n = \log_2 d$ qubits [249, 262, 295]. For loading and manipulation of data, we describe a protocol based on the generation of quantum hypergraph states [256]. This exact approach to artificial neuron operations will be used in the main body as a benchmark to assess the performances of approximate variational techniques designed to achieve more favourable scaling properties in the number of logical operations with respect to classical counterparts.

Let \mathbf{i} and \mathbf{w} be binary input and weight vectors of the form

$$\mathbf{i} = \begin{pmatrix} i_0 \\ i_1 \\ \vdots \\ i_{d-1} \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{d-1} \end{pmatrix} \quad (5.2)$$

with $i_j, w_j \in \{-1, 1\}$ and $d = 2^n$. A simple and qubit-effective way of encoding such collections of classical data can be given by making use of the relative quantum phases (i.e. factors ± 1 in our binary case) in equally weighted superpositions of computational basis states. We then define the states

$$|\psi_i\rangle = \frac{1}{\sqrt{d}} \sum_{j=0}^{d-1} i_j |j\rangle, \quad |\psi_w\rangle = \frac{1}{\sqrt{d}} \sum_{j=0}^{d-1} w_j |j\rangle \quad (5.3)$$

where, as usual, we label computational basis states with integers $j \in \{0, \dots, d-1\}$ corresponding to the decimal representation of the respective binary string. The set of all possible states which can be expressed in the form above is known as the class of hypergraph states [256] and are a special case of the more general states treated in the previous Chapter 4, where we allowed the coefficients of the computational basis to be arbitrary phases $e^{i\theta}$ rather than just signs ± 1 .

According to Eq. (5.1), the quantum algorithm must first perform the inner product $\mathbf{i} \cdot \mathbf{w}$. As shown earlier for the case of the continuous neuron, the inner product between inputs and weights

is contained in the overlap $\langle \psi_w | \psi_i \rangle = \mathbf{w} \cdot \mathbf{i} / d$ [293], and one can explicitly compute such overlap on a quantum register through a sequence of \mathbf{i} - and \mathbf{w} -controlled unitary operations. For clarity of exposition, we hereby summarise again the steps necessary to calculate the overlap of interest, and we refer to the previous chapter for a more detailed discussion.

First, assuming that we operate on a n -qubit quantum register starting in the blank state $|0\rangle^{\otimes n}$, we can load the input-encoding quantum state $|\psi_i\rangle$ by performing a unitary transformation U_i such that $U_i|0\rangle^{\otimes n} = |\psi_i\rangle$. It is important to mention that this preparation step would most effectively be replaced by, e.g., a direct call to a quantum memory [111], or with the supply of data encoding states readily generated in quantum form by quantum sensing devices to be analyzed or classified. It is indeed well known that the interface between classical data and their representation on quantum registers currently constitutes one of the major bottlenecks for Quantum Machine Learning applications [31].

Let now U_w be a unitary operator such that

$$U_w|\psi_w\rangle = |1\rangle^{\otimes n} = |d-1\rangle \quad (5.4)$$

In principle, any $d \times d$ unitary matrix having the elements of \mathbf{w} appearing in the last row satisfies this condition. If we apply U_w after U_i , the overall n -qubits quantum state becomes

$$U_w|\psi_i\rangle = \sum_{j=0}^{d-1} c_j |j\rangle =: |\phi_{i,w}\rangle \quad (5.5)$$

Using Eq. (5.4), we then have

$$\langle \psi_w | \psi_i \rangle = \langle \psi_w | U_w^\dagger U_w | \psi_i \rangle = \langle m-1 | \phi_{i,w} \rangle = c_{m-1} \quad (5.6)$$

We thus see that, as a consequence of the constraints imposed to U_i and U_w , the desired result $\mathbf{i} \cdot \mathbf{w} \propto \langle \psi_w | \psi_i \rangle$ is contained up to a normalisation factor in the coefficient c_{d-1} of the final state $|\phi_{i,w}\rangle$.

The final step of the algorithm must access the computed input-weight scalar product and determine the activation state of the artificial neuron. In view of constructing a general architecture for feed-forward neural networks [295], it is useful to introduce an ancilla qubit a , initially set in the state $|0\rangle$, on which the $c_{d-1} \propto \langle \psi_w | \psi_i \rangle$ coefficient can be written through a multi-controlled CNOT gate, where the role of controls is assigned to the n encoding qubits [293]:

$$|\phi_{i,w}\rangle |0\rangle_{\text{ancilla}} \rightarrow \sum_{j=0}^{d-2} c_j |j\rangle |0\rangle_{\text{ancilla}} + c_{d-1} |d-1\rangle |1\rangle_{\text{ancilla}} \quad (5.7)$$

At this stage, a measurement of ancillary qubit a in the computational basis provides a probabilistic non-linear threshold activation behaviour, producing the output $|1\rangle$ state, interpreted as an active state of the neuron, with probability $|c_{d-1}|^2$. Although this form of the activation function is already sufficient to carry out elementary classification tasks and to realise a logical XOR operation [293], more complex threshold behaviours can in principle be engineered once the information about the inner product is stored on the ancilla [50, 302]. Equivalently, the ancilla can be used, via quantum controlled operations, to pass on the information to other quantum registers encoding successive layers in a feed-forward network architecture [295]. It is worth noticing that directing all the relevant information into the state of a single qubit, besides enabling effective quantum synapses, can be advantageous when implementing the procedure on real hardware on which readout errors constitute a major source of inaccuracy. Nevertheless, multi-controlled NOT operations, which are inherently non-local, can lead to complex decompositions into hardware-native gates especially in the presence of constraints in qubit-qubit connectivity. When operating a single node to carry out simple classification tasks or, as we will do in the following sections, to assess the performances of individual portions of the proposed algorithm, the activation probability of the artificial neuron can then equivalently be extracted directly from the register of N encoding qubits by performing a direct measurement of $|\phi_{i,w}\rangle$ targeting the $|d-1\rangle \equiv |1\rangle^{\otimes n}$ computational basis state.

5.2.1 Exact implementation with quantum hypergraph states

A general and exact realisation of the unitary transformations U_i and U_w can be designed by using the generation algorithm for quantum hypergraph states [293]. The latter have been extensively studied as useful quantum resources [107, 256], and are formally defined as follows. Given a collection of n vertices V , we call a k -hyper-edge any subset of exactly k vertices. A hypergraph $g_{\leq n} = \{V, E\}$ is then composed of a set V of vertices together with a set E of hyper-edges of any order k , not necessarily uniform. Notice that this definition includes the usual notion of a mathematical graph if $k = 2$ for all (hyper)-edges. To any hypergraph $g_{\leq n}$ we associate a n -qubit quantum hypergraph state via the definition

$$|g_{\leq n}\rangle = \prod_{k=1}^n \prod_{\{q_{v_1}, \dots, q_{v_k}\} \in E} C^k Z_{q_{v_1}, \dots, q_{v_k}} |+\rangle^{\otimes n} \quad (5.8)$$

where q_{v_1}, \dots, q_{v_k} are the qubits connected by a k -hyper-edge in E and, with a little abuse of notation, we assume $C^2 Z \equiv CZ$ and $C^1 Z \equiv Z = R_Z(\pi)$. For n qubits there are exactly $2^{2^n - 1}$ different hypergraph states. We can make use of well known preparation strategies for hypergraph states to realise the unitaries U_i and U_w with at most a single n -controlled $C^n Z$ and a collection of p -controlled $C^p Z$ gates with $p < n$. It is worth pointing out already here that such an approach, while optimising the number of multi-qubit logic gates to be employed, implies a circuit depth which scales linearly in the size of the classical input, i.e. $\mathcal{O}(d) \equiv \mathcal{O}(2^n)$, in the worst case corresponding to a fully connected hypergraph [293].

To describe a possible implementation of U_i , assume once again that the quantum register of n encoding qubits is initially in the blank state $|0\rangle^{\otimes n}$. By applying parallel Hadamard gates ($H^{\otimes n}$) we obtain the state $|+\rangle^{\otimes n}$, corresponding to a hypergraph with no edges. We can then use the target collection of classical inputs \mathbf{i} as a control for the following iterative procedure:

Algorithm 2: Quantum hypergraph states generation routine [293]

```

for  $P = 1, \dots, n$  do
  for  $j = 0, \dots, d - 1$  do
    if  $|j\rangle$  has exactly  $P$  qubits in  $|1\rangle$  and  $i_j = -1$  then
      Apply  $C^P Z$  to those qubits;
      Flip the sign of  $i_k$  in  $\mathbf{i} \ \forall k$  such that  $|k\rangle$  has the same  $P$  qubits in  $|1\rangle$ ;
    end
  end
end

```

Similarly, U_w can be obtained by first performing the routine outlined above (without the initial parallel Hadamard gates) tailored according to the classical control \mathbf{w} : since all the gates involved in the construction are the inverse of themselves and commute with each other, this step produces a unitary transformation bringing $|\psi_w\rangle$ back to $|+\rangle^{\otimes n}$. The desired transformation U_w is completed by adding parallel $H^{\otimes n}$ and $\text{NOT}^{\otimes n}$ gates [293].

5.3 Variational realisation of a quantum artificial neuron

Although the implementation of the unitary transformations U_i and U_w outlined above is formally exact and optimises the number of multi-qubit operations to be performed by leveraging on the correlations between the ± 1 phase factors, the overall requirements in terms of circuit depth pose in general severe limitations to their applicability in non error-corrected quantum devices. Moreover, although with such an approach the encoding and manipulation of classical data is performed in an efficient way with respect to memory resources, the computational cost needed to control the

execution of the unitary transformations and to actually perform the sequences of quantum logic gates remains bounded by the corresponding classical limits. Therefore, the aim of this section is to explore conditions under which some of the operations introduced in our quantum model of artificial neurons can be obtained in more efficient ways by exploiting the natural capabilities of quantum processors.

In the following, we will mostly concentrate on the task of realising approximate versions of the weight unitary U_w with significantly lower implementation requirements in terms of circuit depth. Although most of the techniques that we will introduce below could in principle work equally well for the preparation of encoding states $|\psi_i\rangle$, it is important to stress already at this stage that such approaches cannot be interpreted as a way of solving the long standing issue represented by the loading of classical data into a quantum register. Instead, they are pursued here as an efficient way of *analysing* classical or quantum data presented in the form of a quantum state. Indeed, the variational approach proposed here requires ad-hoc training for every choice of the target vector \mathbf{w} whose U_w needs to be realised. To this purpose, we require access to many copies of the desired $|\psi_w\rangle$ state, essentially representing a quantum training set for our artificial neuron. As in our formulation a single node characterized by weight connections \mathbf{w} can be used as an elementary classifier recognising input data sufficiently close to \mathbf{w} itself [293], the variational procedure presented here essentially serves the double purpose of training the classifier upon input of positive examples $|\psi_w\rangle$ and of finding an efficient quantum realisation of such state analyser.

5.3.1 Global variational training

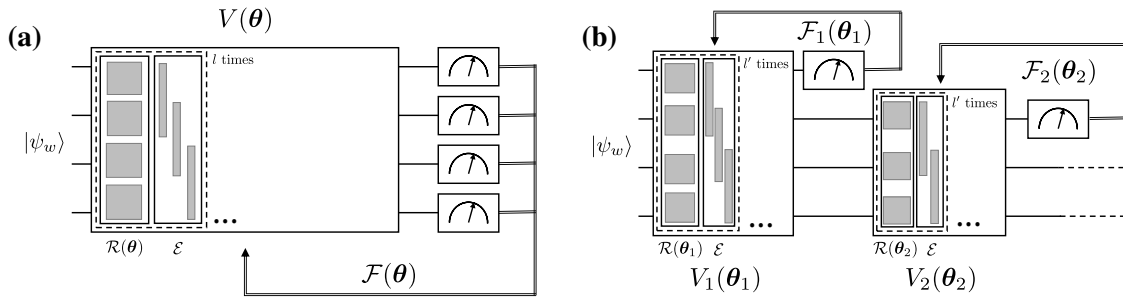


Figure 5.1: Variational learning via unsampling. (a) Global strategy, with optimization targeting all qubits simultaneously. (b) Local qubit-by-qubit approach, in which each layer is used to optimize the operation for one qubit at a time.

According to Eq. (5.4), the purpose of the transformation U_w within the quantum artificial neuron implementation is essentially to reverse the preparation of a non-trivial quantum state $|\psi_w\rangle$ back to the relatively simple product state $|1\rangle^{\otimes n}$. Notice that in general the qubits in the state $|\psi_w\rangle$ share multipartite entanglement [107]. Here we discuss a promising strategy for the efficient approximation of the desired transformation satisfying the necessary constraints based on variational techniques. Inspired by the well known variational quantum eigensolver (VQE) algorithm [230], and in line with a recently introduced unsampling protocol [54], we define the following optimization problem: given access to independent copies of $|\psi_w\rangle$ and to a variational quantum circuit, characterized by a unitary operation $V(\theta)$ and parametrized by a set of angles θ , we wish to find a set of values θ_{opt} that guarantees a good approximation of U_w . The heuristic circuit implementation typically consists of sequential blocks of single-qubit rotations followed by entangling gates, repeated up to a certain number that guarantees enough freedom for the convergence to the desired unitary [155].

Once the solution $V(\theta_{\text{opt}})$ is found, which in our setup corresponds to a fully trained artificial neuron, it would then provide a form of quantum advantage in the analysis of arbitrary input states

$|\psi_i\rangle$ as long as the circuit depth for the implementation of the variational ansatz is sub-linear in the dimension of the classical data, i.e. sub-exponential in the size of the qubit register. As it is customarily done in near-term VQE applications, the optimization landscape is explored by combining executions of quantum circuits with classical feedback mechanisms for the update of the $\boldsymbol{\theta}$ angles. In the most general scenario, and according to Eq. (5.4), a cost function can be defined as

$$\mathcal{F}(\boldsymbol{\theta}) = 1 - |\langle 11 \dots 1 | V(\boldsymbol{\theta}) | \psi_w \rangle|^2, \quad (5.9)$$

and the solution $\boldsymbol{\theta}_{\text{opt}}$ represented by

$$\boldsymbol{\theta}_{\text{opt}} = \arg \min_{\boldsymbol{\theta}} \mathcal{F}(\boldsymbol{\theta}) \quad (5.10)$$

which leads to $V(\boldsymbol{\theta}_{\text{opt}}) \simeq U_w$. We call this approach a *global* variational unsampling as the cost function in Eq. (5.9) requires all qubits to be simultaneously found as close as possible to their respective target state $|1\rangle$, without making explicit use of the product structure of the desired output state $|1\rangle^{\otimes n}$. It is indeed well known that VQE can lead in general to exponentially difficult optimization problems [202], however the characteristic feature of the problem under evaluation may actually allow for a less complex implementation of the VQE for unsampling purposes [54], as outlined in the following section. A schematic representation of the global variational training is provided in Fig. 5.1a.

5.3.2 Local variational training

An alternative approach to the global unsampling task, particularly suited for the case we are considering in which the desired final state of the quantum register is fully unentangled, makes use of a local, qubit-by-qubit procedure. This technique, which was recently proposed and tested on a photonic platform as a route towards efficient certification of quantum processors [54], is highlighted here as an additional useful tool within a general quantum machine learning setting.

In the local variational unsampling scheme, the global transformation $V(\boldsymbol{\theta})$ is divided into successive layers $V_j(\boldsymbol{\theta}_j)$ of decreasing complexity and size. Each layer is trained separately, in a serial fashion, according to a cost function which only involves the fidelity of a single qubit to its desired final state. More explicitly, every $V_j(\boldsymbol{\theta}_j)$ operates on qubits j, \dots, n and has an associated cost function

$$\mathcal{F}_j(\boldsymbol{\theta}_j) = 1 - \langle 1 | \text{Tr}_{j+1, \dots, n} [\rho_j] | 1 \rangle, \quad (5.11)$$

where the partial trace leaves only the degrees of freedom associated to the j -th qubit and, recursively, we define

$$\rho_j = \begin{cases} V_j(\boldsymbol{\theta}_j) \rho_{j-1} V_j^\dagger(\boldsymbol{\theta}_j) & j > 1 \\ |\psi_w\rangle\langle\psi_w| & j = 0 \end{cases}. \quad (5.12)$$

At step j , it is implicitly assumed that all the parameters $\boldsymbol{\theta}_k$ for $k = 1, \dots, j-1$ are fixed to the optimal values obtained by the minimisation of the cost functions in the previous steps. Notice that, operationally, the evaluation of the cost function \mathcal{F}_j can be automatically carried out by measuring the j -th qubit in the computational basis while ignoring the rest of the quantum register, as shown in Fig. 5.1b.

The benefits of local variational unsampling with respect to the global strategy are mainly associated to the reduced complexity of the optimisation landscape per step. Indeed, the local version always operates on the overlap between single-qubit states, at the relatively modest cost of adding $n-1$ smaller and smaller variational ansätze. In the specific problem at study, we thus envision the local approach to become particularly effective, and more advantageous than the global one, in the limit of large enough number of qubits, i.e. for the most interesting regime where the size of the quantum register, and therefore of the quantum computation, exceeds the current classical simulation capabilities.

5.3.3 Case study: pattern recognition

To show an explicit example of the proposed construction, let us fix $m = 16$, $n = 4$. Following Ref. [293], we can visualise a 16-bit binary vector \vec{b} , see Eq. (5.2), as a 4×4 binary pattern of black ($b_j = -1$) and white ($b_j = 1$) pixels. Moreover, we can assign to every possible pattern an integer label k_b corresponding to the conversion of the binary string $k_b = b_0 \dots b_{15}$, where $b_j = (-1)^{b_j}$. We choose as our target \mathbf{w} the vector corresponding to $k_w = 20032$, which represents a black cross on white background at the north-west corner of the 16-bit image, see Fig 5.2.

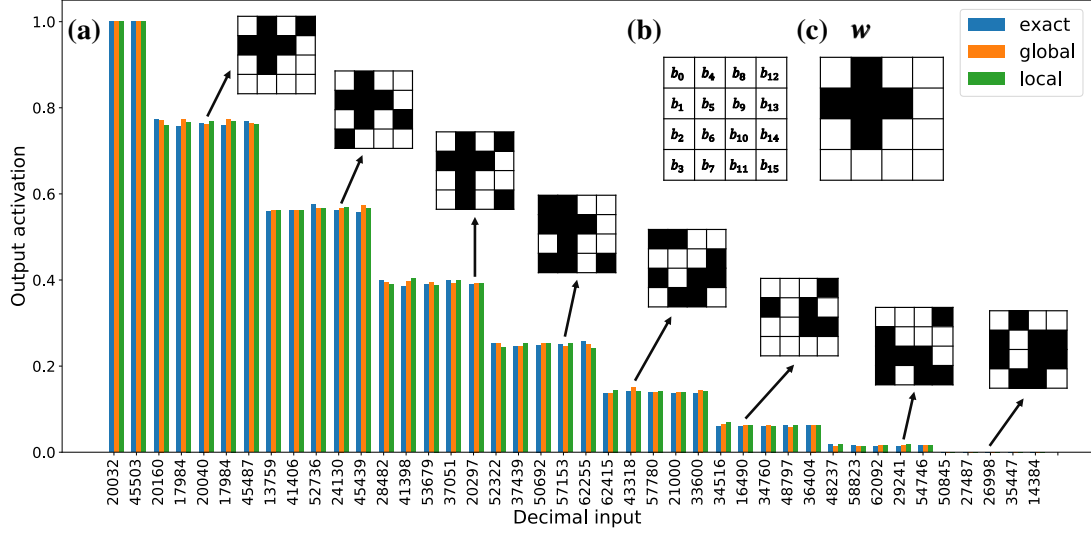


Figure 5.2: Comparison of output activation $p_{\text{out}} = |\langle \psi_w | \psi_i \rangle|^2$ among the exact (hypergraph states routine), global ($n = 3$) and local ($n' = 2$) approximate implementations of U_w . The inset shows the general mapping of any 16-dimensional binary vector \vec{b} onto the 4×4 binary image (b) and the cross-shaped \vec{w} used in this example (c). The selected inputs on which the approximations are tested were chosen to cover all the possible cases for p_{out} , and are labelled with their corresponding integer k_i (see main text).

Starting from the global variational strategy, we construct a parametrized ansatz for $V(\boldsymbol{\theta})$ as a series of entangling \mathcal{E} and rotation $\mathcal{R}(\boldsymbol{\theta})$ cycles:

$$V(\boldsymbol{\theta}) = \left(\prod_{c=1}^l \mathcal{R}(\theta_{c,1}, \dots, \theta_{c,4}) \mathcal{E} \right) \mathcal{R}(\theta_{0,1}, \dots, \theta_{0,4}), \quad (5.13)$$

where l is the total number of cycles which in principle can be varied to increase the expressibility of the ansatz by increasing its total depth. Rotations are assumed to be acting independently on the $n = 4$ qubits according to

$$\mathcal{R}(\theta_{c,1}, \dots, \theta_{c,4}) = \bigotimes_{q=1}^4 R_Y(\theta_{c,q}) = \bigotimes_{q=1}^4 \exp\left(-i \frac{\theta_{c,q}}{2} Y^{(q)}\right), \quad (5.14)$$

where $Y^{(q)}$ is the Pauli- Y matrix acting on qubit q . At the same time, the entangling parts promote

all-to-all interactions between the qubits according to²

$$\mathcal{E} = \prod_q \prod_{q'=q+1}^4 \text{CNOT}_{qq'}, \quad (5.15)$$

where $\text{CNOT}_{qq'}$ is the usual controlled NOT operation between control qubit q and target q' acting on the space of all 4-qubits. For l cycles, the total number of θ -parameters, including the initial rotation layer $\mathcal{R}(\theta_{0,1}, \dots, \theta_{0,4})$, is therefore $4 + 4l$.

A qubit-by-qubit version of the ansatz can be constructed in a similar way by using the same structure of entangling and rotation cycles, decreasing the total number of qubits by one after each layer of the optimization. Here we choose a uniform number l' of cycles per qubit (this condition will be relaxed afterwards, see Sec. 5.3.4), thus setting $\forall j \neq 4$

$$V_j(\boldsymbol{\theta}_j) = \left(\prod_{c=1}^{l'} \mathcal{R}(\theta_{c,j}, \dots, \theta_{c,4}) \mathcal{E} \right) \mathcal{R}(\theta_{0,j}, \dots, \theta_{0,4}) \quad (5.16)$$

For $j = 4$, we add a single general single-qubit rotation with three parameters

$$V_4(\alpha, \beta, \gamma) = \exp \left(-i \frac{(\alpha, \beta, \gamma) \cdot \boldsymbol{\sigma}^{(4)}}{2} \right) \quad (5.17)$$

where $\boldsymbol{\sigma} = (X, Y, Z)$ are again the usual Pauli matrices.

We implemented both versions of the variational training in Qiskit [5], combining exact simulation of the quantum circuits required to evaluate the cost function with classical Nelder-Mead [218] and Cobyla [235] optimizers from the `scipy` Python library. We find that the values $l = 3$ and $l' = 2$ allow the routine to reach total fidelities to the target state $|1\rangle^{\otimes n}$ well above 99.99%. As shown in Fig 5.2, this in turn guarantees a correct reproduction of the exact activation probabilities of the quantum artificial neuron with a quantum circuit depth of 19 (29) for the global (qubit-by-qubit) strategy, as compared to the total depth equal to 49 for the exact implementation of U_w using hypergraph states. This counting does not include the gate operations required to prepare the input state, i.e. it only evidences the different realisations of the U_w implementation assuming that each $|\psi_i\rangle$ is provided already in the form of a wavefunction. Moreover, the multi-controlled $C^P Z$ operations appearing in the exact version were decomposed into single-qubit rotations and CNOTs without the use of additional working qubits. Notice that these conditions are the ones usually met in real near-term superconducting hardware endowed with a fixed set of universal operations.

5.3.4 Structure of the ansatz and scaling properties

In many practical applications, the implementation of the entangling block \mathcal{E} could prove technically challenging, in particular for near term quantum devices based, e.g., on superconducting wiring technology, for which the available connectivity between qubits is limited. For this reason, it is useful to consider a more hardware-friendly entangling scheme, which we refer to as *nearest neighbours*. In this case, each qubit is entangled only with at most two other qubits, essentially assuming the topology of a linear chain

$$\mathcal{E}_{nn} = \prod_{q=1}^3 \text{CNOT}_{q,q+1} \quad (5.18)$$

²As discussed later in Chapter 7, this all-to-all entangling scheme is actually equivalent to nearest-neighbour linear chain of CNOTs in reversed order. Thus, differences in performances between this entangling strategy \mathcal{E} in Eq. (5.15) and the nearest-neighbour one \mathcal{E}_{nn} in Eq. (5.18) discussed in this chapter are likely to be attributed to the former one being more suited to the specific tasks analysed in this work, rather than to the creation of “more” entanglement, as one would reasonably expect. Although the creation of nontrivial entanglement is necessary to avoid classical simulability (see Ch. 7), these results confirm the importance of choosing problem-inspired variational ansätze for ensuring good performances.

This scheme may require even fewer two-qubit gates to be implemented with respect to the all-to-all scheme presented above. Moreover, this entangling unitary fits perfectly well on those quantum processors consisting of linear chains of qubits or heavy hexagonal layouts.

We implemented both global and local variational learning procedures with nearest neighbours entanglers in Qiskit [5], using exact simulation of the quantum circuits with classical optimizers to drive the learning procedure. In the following, we report an extensive analysis of the performances and a comparison with the all-to-all strategy introduced in Sec. 5.3.3 above. All the simulations are performed by assuming the same cross-shaped target weight vector \mathbf{w} depicted in Fig. 5.2.

In Figure 5.3 we show an example of the typical optimization procedure for three different choices of the ansatz depth (i.e. number of entangling cycles) $l = 1, 2, 3$, assuming a global cost function. Here we find that $l = 3$ allows the routine to reach a fidelity $\mathcal{F}(\boldsymbol{\theta})$ to the target state $|1\rangle^{\otimes n}$ above 99%.

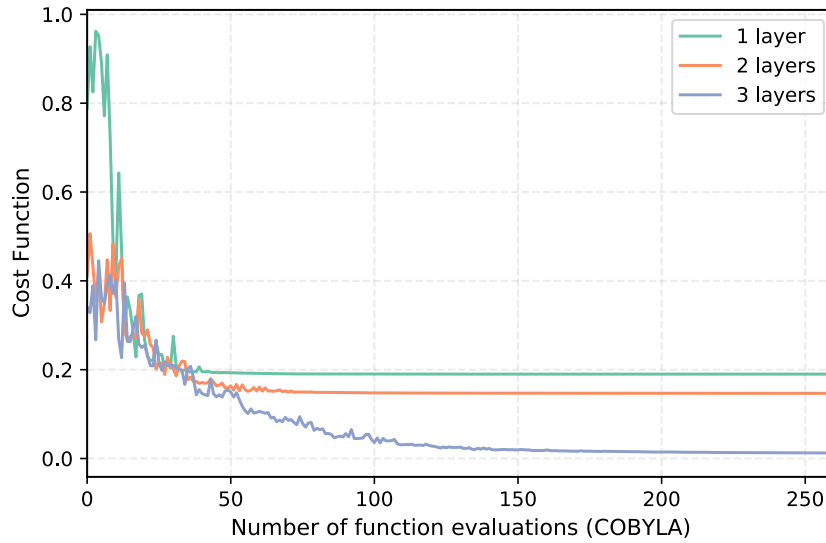


Figure 5.3: Optimisation of the global unitary with nearest neighbours entanglement for three different structures differing in the numbers of entangling blocks l . The cost function is $|\langle 11 \dots 1 | V(\boldsymbol{\theta}) | \psi_w \rangle|^2 = 1 - \mathcal{F}(\boldsymbol{\theta})$, see Eq. (5.9). Only for $l = 3$ the learning model has enough expressibility to reach a good final fidelity. The classical optimiser used in this case was COBYLA [235].

In the local qubit-by-qubit variational scheme, we can actually introduce an additional degree of freedom by allowing the number of cycles per qubit, l' , to vary between successive layers corresponding to the different stages of the optimization procedure. For example, we may want to use a deeper ansatz for the first unitary acting on all the qubits, and shallower ones for smaller subsystems. We thus introduce a different l'_j for each $V_j(\boldsymbol{\theta}_j)$ in Eq. (5.16) and we name *structure* the string ' $l_1 l_2 l_3$ '. The latter denotes a learning model consisting of three optimization layers: $V_1(\boldsymbol{\theta}_1)$ with l_1 entangling cycles, $V_2(\boldsymbol{\theta}_2)$ with l_2 and $V_3(\boldsymbol{\theta}_3)$ with l_3 , respectively. In the last step of the local optimization procedure, i.e. when a single qubit is involved, we always assume a single 3-parameter rotation, see Eq. (5.17). A similar notation will be also applied in the following when scaling up to $n > 4$ qubits.

The effectiveness of different structures is explored in Figure 5.4a. We see that, while the all-to-all entangling scheme typically performs better in comparison to the nearest neighbour one, this increase in performance comes at the cost of deeper circuits. Moreover, the stepwise decreasing structure '321' for the nearest neighbour entangler proves to be an effective solution to problem, achieving a good final accuracy (above 99%) with a low circuit depth. This trend is also confirmed for the higher dimensional case of $n = 5$ qubits, which we report in Fig. 5.4b. Here, the dimension

of the underlying pattern recognition task is increased by extending the original 16-bit weight vector \mathbf{w} with extra 0s in front of the binary representation \mathbf{k}_w . In fact, it can easily be seen that, assuming directly nearest neighbours entangling blocks, the decreasing structure ‘4321’ gives the best performance-depth trade-off.

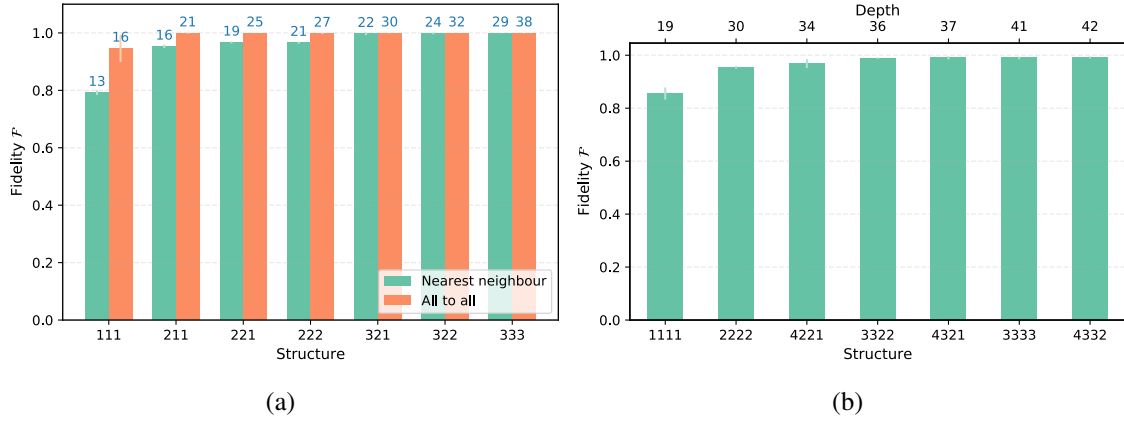


Figure 5.4: Final fidelity for different structures and number of qubits. (a) Final fidelity obtained for the local variational training and using both the all-to-all entangler \mathcal{E} (5.15) and nearest neighbour \mathcal{E}_{nn} (5.18). On top of each rectangle, in light blue, it is reported the depth of the corresponding quantum circuit to implement that given structure with that particular entangling scheme. For clarity, a structure ‘211’ corresponds to a variational model having two repetitions ($l'_1 = 2$) for the first layer acting on all 4 qubits, and 1 cycle ($l'_1 = l'_2 = 1$) for the remaining two layers acting on 3 and 2 qubits respectively. Each bar was obtained executing the optimization process 10 times, and then evaluating the means and standard deviations (shown as error bars). The optimization procedure was performed using COBYLA [235]. (b) Final fidelities for different structures of the local variational learning model with a nearest neighbour entangler, for the case of $n = 5$ qubits. Similarly to the case with $n = 4$ qubits portrayed in Figure 5.4a, the most depth-efficient structure is the one consisting of constantly decreasing number of cycles.

Such empirical fact, namely that the most efficient structure is typically the one consisting of decreasing depths, can be heuristically interpreted by recalling again that, in general, the optimization of a function depending on the state of a large number of qubits is a hard training problem [202]. Although we employ local cost functions, to complete our particular task each variational layer needs to successfully disentangle a single qubit from all the others still present in the register. It is therefore not totally surprising that the optimization carried out in larger subsystems requires more repetitions and parameters (i.e. larger n'_j) in order to make the ansatz more expressive.

By assuming that the stepwise decreasing structure remains sufficiently good also for larger numbers of qubits, we studied the optimization landscape of global (5.9) and local (5.11) cost functions by investigating how the hardness of the training procedure scales with increasing n . As commented above for $n = 5$ qubits, we keep the same underlying target \mathbf{w} , which we expand by appending extra 0s in the binary representation. To account for the stochastic nature of the optimization procedure, we run many simulations of the same learning task and report the mean number of iterations needed for the classical optimiser to reach a given target fidelity $\mathcal{F} = 95\%$, and we report simulation results in Figure 5.5.

The most significant message is that the use of the aforementioned local cost function seems to require higher classical resources to reach a given target fidelity when the number of qubits increases. This actually should not come as a surprise, since the number of parameters to be optimised in the two cases is different. In fact, in the global scenarios there are $n + n \cdot l$ (the first n

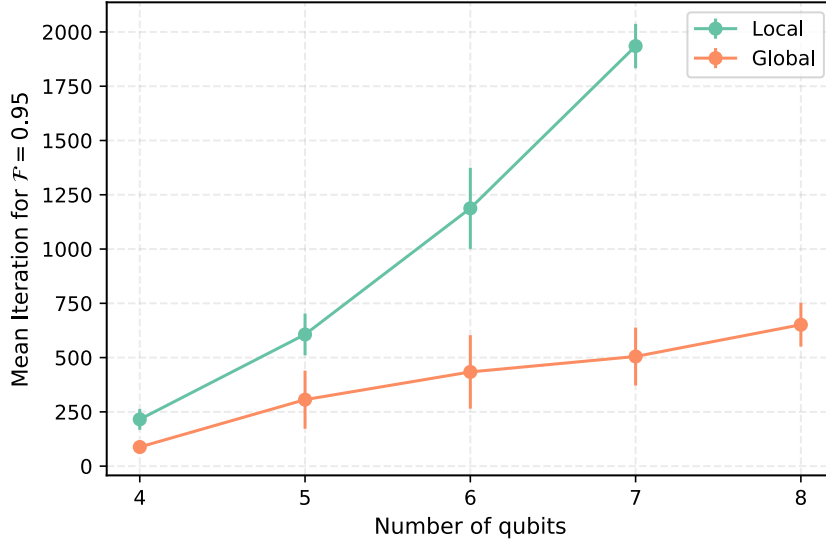


Figure 5.5: Number of iterations of the classical optimiser to reach a fidelity of $\mathcal{F} = 95\%$. Each point in the plot is obtained by running the optimization procedure 10 times and then evaluating the mean and standard deviation (shown as error bars in the plot). All results refer to exact simulations of the quantum circuits in the absence of statistical measurement sampling or device noise, performed with Qiskit `statevector_simulator`.

is due to the initial layer of rotations) parameters to be optimised, while in the local case there are $n + n \cdot l'_1$ for the first layer, $(n - 1) + (n - 1) \cdot l'_2$ for the second and so on, for a total of

$$\#_{\text{local}} = \sum_{q=2}^n q + q l'_q + 3, \quad (5.19)$$

where the final 3 is due to the fact that the last layer always consist of a rotation on the Bloch sphere with three parameters, see Eq. (5.17). Using the stepwise decreasing structure, that is $n'_q = q - 1$, we eventually obtain $\sum_{q=2}^n q + q(q - 1) = \sum_{q=2}^n q^2 \sim O(n^3)$, compared to $\#_{\text{global}} \sim O(n^2)$. Here we are assuming a number of layers $l = n - 1$, consistently with the $n = 4$ qubits case (see Figure 5.3). While in the global case the optimization makes full use of the available parameters to globally optimize the state towards $|1\rangle^{\otimes n}$, the local unitary has to go through multiple disentangling stages, requiring (at least for the cases presented here) more classical iteration steps. At the same time, it would probably be interesting to investigate other examples in which the number of parameters between the two alternative schemes remains fixed, as this would most likely narrow the differences and provide a more direct comparison.

In agreement with similar investigations [283], we can actually conclude that only modest differences between global and local layer-wise optimization approaches are present when dealing with exact simulations (i.e. free from statistical and hardware noise) of the quantum circuit. Indeed, both strategies achieve good results and a final fidelity $\mathcal{F}(\boldsymbol{\theta}) > 99\%$. At the same time, it becomes interesting to investigate how the different approaches behave in the presence of noise, and specifically statistical noise coming from measurements operations. For this reason, we implemented the measurement sampling using Qiskit `qasm_simulator` and employed a stochastic gradient descent (SPSA) classical optimization method. Each benchmark circuit is executed $n_{\text{shots}} = 1024$ times in order to reconstruct the statistics of the outcomes. Moreover, we repeat the stochastic optimization routine multiple times to analyse the average behaviour of the cost function.

In Figure 5.6 we show the optimization procedure for the local and global cost functions in the presence of measurement noise, with both of them reaching acceptable and identical final

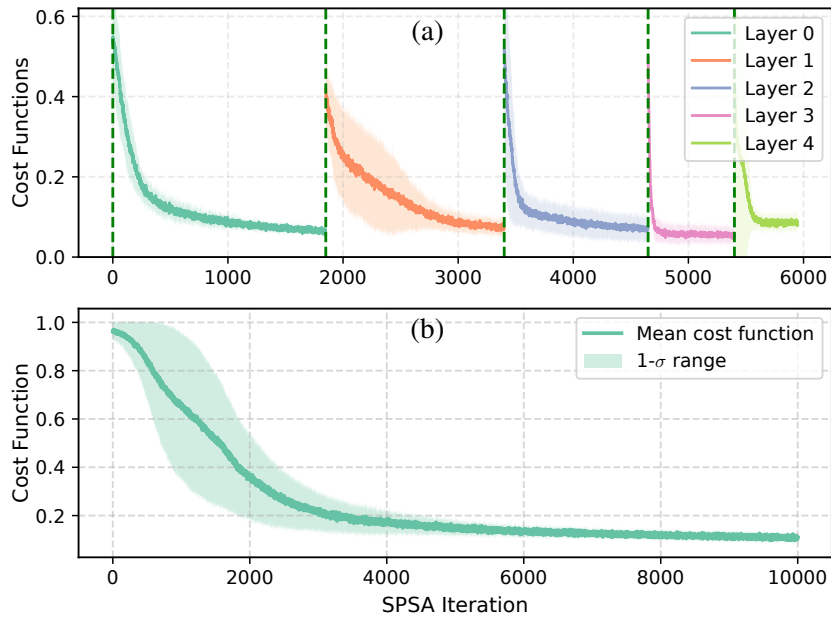


Figure 5.6: Optimisation of cost functions for the local **(a)** and global **(b)** case in the presence of measurement noise for $n = 5$ qubits. In each figure we plot the mean values averaged on 5 runs of the simulation. The shaded coloured areas denote one standard deviation. The number of measurement repetitions in each simulation was $n_{\text{shots}} = 1024$. The final fidelity at the end of the training procedure in this case were $\mathcal{F}_{\text{local}} = 0.87 \pm 0.02$ and $\mathcal{F}_{\text{global}} = 0.89 \pm 0.02$. Notice the difference in the horizontal axes bounds. **(a)** Optimisation of the local cost functions $V_j(\boldsymbol{\theta}_j)$ (see Eq. (5.11)), plotted with different colours for clarity. The vertical dashed lines denotes the end of the optimization of one layer, and the start of the optimization for the following one. **(b)** Optimisation of the global cost function $V(\boldsymbol{\theta})$ in Eq. (5.9)

fidelities $\mathcal{F}_{\text{local}} = 0.87 \pm 0.02$ and $\mathcal{F}_{\text{global}} = 0.89 \pm 0.02$. Notice that for the local case (Figure 5.6a) each coloured line indicates the optimization of a $V_j(\boldsymbol{\theta}_j)$ from Eq. (5.11). We observe that the training for the local model generally requires fewer iterations, with an effective optimization of each single layer. On the contrary, in the presence of measurement noise the global variational training struggles to find a good direction for the optimization and eventually follows a slowly decreasing path to the minimum. These findings look to be in agreement, e.g., with results from Refs. [57, 283]: with the introduction of statistical shot noise, the performances of the global model are heavily affected, while the local approach proves to be more resilient and capable of finding a good gradient direction in the parameters space [57]. In all these simulations, the parameters in the global unitary and in the first layer of the local unitary were initialised with a random distribution in $[0, 2\pi)$. All subsequent layers in the local model were initialised with all parameters set to zero in order to allow for smooth transitions from one optimization layer to the following. This strategy was actually suggested as a possible way to mitigate the occurrence Barren plateaus [115, 283].

We conclude the scaling analysis by reporting in Fig. 5.7 a summary of the quantum circuit depths required to implement the target unitary transformation with different strategies and for increasing sizes of the qubit register up to $n = 7$. As it can be seen, all the variational approaches scale much better when compared to the exact implementation of the target U_w , with the global ones requiring shallower depths in the specific case. In addition, we recall that the use of an all-to-all entangling scheme requires longer circuits due to the implementation of all the CNOTs, but generally needs less ansatz cycles (see Figure 5.4a). At last, while the global procedures seem to provide a better alternative compared to local ones in terms of circuit depth, they might be more

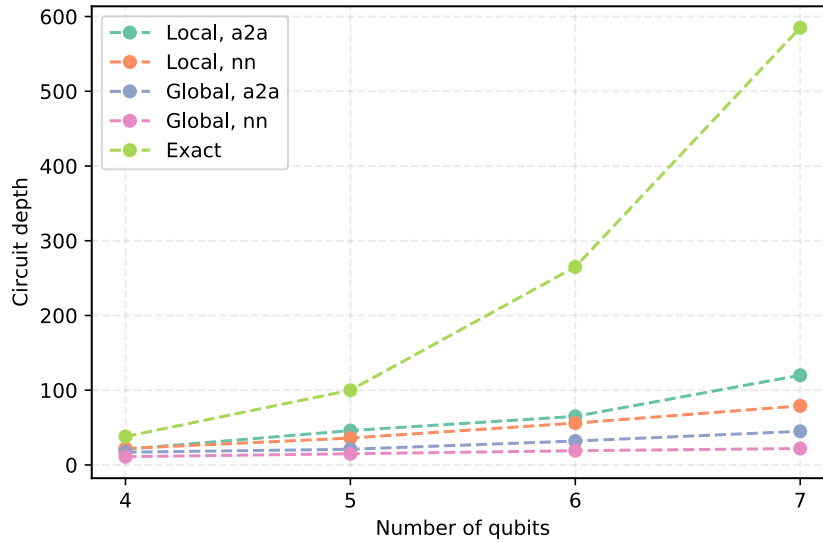


Figure 5.7: Scaling of circuit depth for the implementation of U_w computed with Qiskit. The labels *locals* and *global* refer to the local and global variational approaches, while *a2a* and *nn* refer to the all-to-all and nearest-neighbour entangling schemes respectively. The number of ansatz cycles used for both the global (l) and local/qubit-by-qubit (l') variational constructions and for each entangling structure are increased with the number of qubits up to the minimum value guaranteeing a fidelity of the approximations above 98%.

prone to suffering from classical optimization issues [202, 283] when trained and executed on real hardware, as suggested by the data reported in Fig. 5.6. The overall promising results confirm the significant advantage brought by variational strategies compared to the exponential increase of complexity required by the exact formulation of the algorithm.

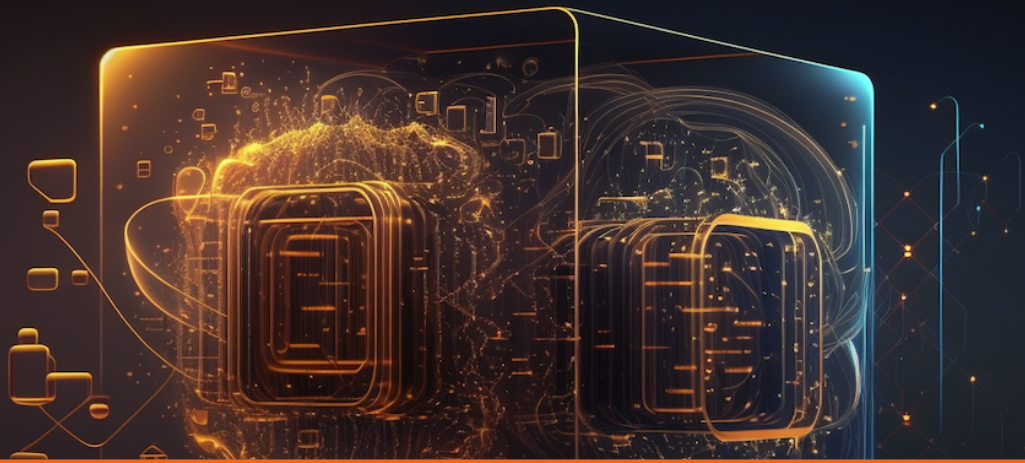
5.4 Conclusions

In this chapter we reviewed an exact model for the implementation of artificial neurons on a quantum processor and we introduced variational training methods for efficiently handling the manipulation of classical and quantum input data. Through extensive numerical analysis, we compared the effectiveness of different circuit structures and learning strategies, highlighting potential benefits brought by hardware-compatible entangling operations and by layerwise training routines. Our analysis suggests that quantum unsampling techniques represent a useful resource, upon input of quantum training sets, to be integrated in quantum machine learning applications.

From a theoretical perspective, our proposed procedure allows for an explicit and direct quantification of possible quantum computational advantages for classification tasks. It is also worth pointing out that such a scheme remains fully compatible with recently introduced architectures for quantum feed-forward neural networks [293], which are needed in general to deploy e.g. complex convolutional filters. Moreover, although the interpretation of quantum hypergraph states as memory-efficient carriers of classical information guarantees an optimal use of the available dimension of a n -qubit Hilbert space, the variational techniques introduced here can in principle be used to learn different encoding schemes designed, e.g., to include continuous-valued features or to improve the separability of the data to be classified [44, 131, 264].

In all envisioned applications, our proposed protocols are intended as an effective method for the analysis of quantum states as provided, e.g., by external devices or sensors, while it is worth stressing that the general problem of efficiently loading classical data into quantum registers still stands open. Finally, on a more practical level, a successful implementation on near-term quantum

hardware of the variational learning algorithm introduced in this work will necessarily rely on a deeper analysis of the impact of realistic noise effects both on the training procedure and on the final optimised circuit. In particular, we anticipate that the reduced circuit depth produced via the proposed method could critically lessen the quality requirements for quantum hardware, eventually leading to meaningful implementation of quantum neural networks within the near-term regime.



6. Quantum autoencoder and classifier for an industrial use case

6.1	Introduction	108
6.2	Case study	109
6.3	Neural network autoencoder	110
6.3.1	Classical Autoencoders	112
6.4	Quantum Data Compression	113
6.4.1	Quantum Autoencoder	113
6.5	Experiments and Results	115
6.5.1	Data compression	115
6.5.2	Classification	119
6.6	Conclusions	121

Quantum computing technologies are in the process of moving from academic research to real industrial applications, with the first hints of quantum advantage demonstrated in recent months. In these early practical uses of quantum computers it is relevant to develop algorithms that are useful for actual industrial processes. In this chapter¹, we propose a quantum pipeline, comprising a quantum autoencoder followed by a quantum classifier, which are used to first compress and then label classical data coming from a separator, i.e., a machine used in one of Eni's Oil Treatment Plants. This study represents one of the first attempts to integrate quantum computing procedures in a real-case scenario of an industrial pipeline, in particular using actual data coming from physical machines, rather than pedagogical data from benchmark datasets.

6.1 Introduction

In this chapter we test the use of quantum machine learning algorithms on a specific industrial use case. In particular, we propose the application of a newly formulated quantum pipeline comprising a quantum autoencoder algorithm [38, 161, 171, 253] followed by a quantum classifier, applied to real data coming from a first stage water/oil separator of one of Eni's oil treatment plant. This algorithm is compared to the performance of a classical autoencoder to compress the original data, which are then used to implement a classification task. It is particularly relevant to notice that

¹The content of this chapter is based on the author's work [196], and all the figures in this chapter are taken from, or are adaptations of, the figures present in such work.

these quantum autoencoding algorithms can be run on presently existing quantum hardware, thus making such quantum machine learning algorithm readily usable with actual input data coming from a realistic source of industrial interest. While various models of variational autoencoders in the quantum domain have been proposed in the literature, for example for generative modelling tasks [161] and for the study of entanglement in quantum states [62], our implementation of the quantum autoencoder directly follows the architecture proposed by authors in [253], which is often studied as a prototypical model in the quantum machine learning literature [57], and it was also even extended to feature input redundancy [229], as discussed in [38].

The chapter is organised as follows. In Sec. 6.2 we explain and give the specifics of the industrial case study considered in this work. In Sec. 6.3 we introduce the classical neural network model of the autoencoder, and also discuss the clustering algorithm used to create the two classes for the classification problem. In Sec. 6.4 we review the quantum algorithm developed for a continuously valued input neuron already discussed in Chapter 4 [195], from which the quantum algorithm for the quantum autoencoder is derived. In Sec. 6.5 we show the results obtained for the data compression task, comparing them with those obtained with the purely classical autoencoder. At last in Sec. 6.5.2, we use the compressed data to implement a quantum classifier used to label the original data in a binary classification problem.

6.2 Case study

The industrial case study discussed in this Chapter aims at testing classical and quantum machine learning approaches to analyse data coming from an industrial equipment within one of Eni's Oil Treatment plants, showed in Fig. 6.1. The equipment is a separator, i.e. a vessel receiving a stream

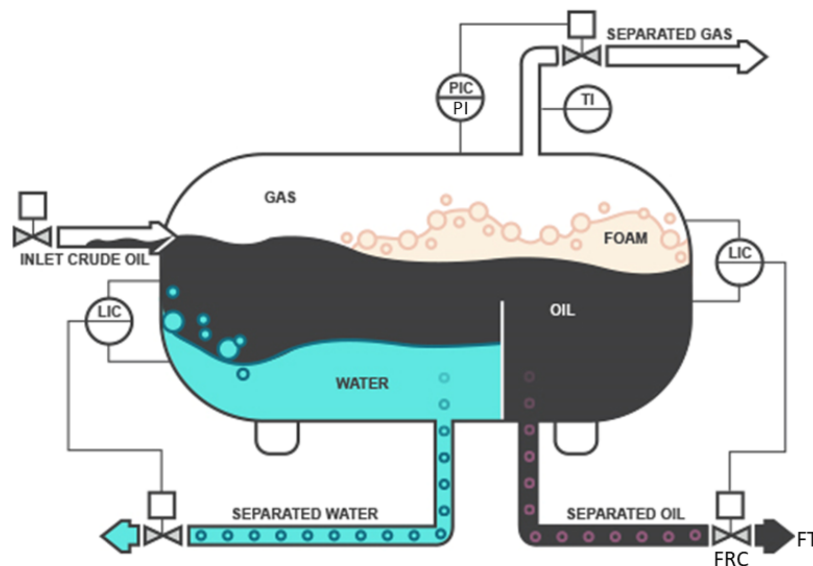


Figure 6.1: Snapshot of the separator. The separator is regulated with three controllers: a pressure controller for the output gas stream, and two-level controllers for the water and the oil stream. The controllers use PID controller equations to regulate the opening of valves on the output streams.

of high pressure, high temperature crude oil (left part of the figure, indicated with a black stream), and exploits gravity to separate three output streams: Water (the heaviest component), indicated in the figure with a light blue stream; Oil (intermediate component), in the lower part of the figure indicated with a black stream; and Gas (lightest component), indicated with a light grey stream. The separator is regulated with three controllers: a pressure controller for the output gas stream, and two-level controllers for the water and the oil stream. Notice that the controllers use PID

(proportional – integral – derivative) controller equations to regulate the opening of valves on the output streams.

In a realistic machine learning problem, we might wish to use all the measurements coming from the sensors installed on this component, as well as on some of the components installed upstream, in order to predict if the behavior of the equipment is normal or faulty (i.e. working in a degraded mode). However, due to the limitation in the complexity of the problems that can currently be faced with quantum computing, we will focus on a simplified problem, involving only 4 variables, that are: the oil level (LIC), the oil output flow (FT), the pressure (PI), and the opening of the oil output valve (FRC). Sensor measurements are sampled every 10 seconds and stored into data tables to be used for the training of the neural networks.

The first step of the case study is the implementation of a dimensionality reduction procedure to compress the 4-dimensional input vector $\mathbf{x} = (x_{\text{FRC}}, x_{\text{FT}}, x_{\text{LIC}}, x_{\text{PI}}) \in \mathbb{R}^4$ into a 2-dimensional vector. This is done both via a standard classical neural network autoencoder and a quantum autoencoder, introduced in Sec. 6.3 and Sec. 6.4 respectively.

The second step is the implementation a classifier using the 2-dimensional latent vector from the compression step to classify the status of the component. In order to do so, we need a labelled training dataset associating an input \mathbf{x}_i to a label $y_i = \{0, 1\}$ corresponding to the “ok” or “faulty” state respectively. However, since 4 variables are too few to label the working status of the separator as “ok” or “faulty”, we followed a different approach, as explained in the upper left panel of Fig. 6.2. We run a binary clustering algorithm on the initial variables, in order to identify two categorical states, named as “Class A” and “Class B”, and then used these categorical states as the labels for the classification task. So, the latent vector from the encoder is used as input for the classifier, that is trained to correctly predict the “Class A” and “Class B” states. The clustering algorithm used is the KMeans algorithm as implemented in the `scikit-learn` library [228]. This algorithm takes as input the desired number of clusters, in our case two, and tries to split the data in groups of equal variance. The centroids of the clusters were initialised uniformly at random. In Fig. 6.2 we show the result of the clustering procedure, where for ease of plotting we show only three of the four variables. This categorical dataset is then used to train a classical and quantum classifier, whose implementation details and results are discussed in Sec. 6.5.2.

In Table 6.1 we summarise the findings of our work, showing the key figures (compression error and classification accuracy) for the classical and quantum pipelines considered in the case study.

Table 6.1: Key figures for the compression and classification tasks for the classical and quantum procedures considered. The compression task is implemented with classical and quantum autoencoders; the classification task is implemented with a `KNeighborsClassifier` and with a single qubit variational classifier. Compression error refers to the average reconstruction error defined in Eq. (6.6). Classification accuracy is defined as the percentage of correctly classified data.

	Compression error	Classification accuracy
Classical	5%	89.7%
Quantum	5.4%	87.4%
Quantum hardware (<code>ibmq_x2</code>)	—	82.3%

6.3 Neural network autoencoder

As extensively discussed in Sec 3.2, the most common use case of artificial neural networks is supervised learning, where the network is asked to learn a mapping from an input to an output space, by having access to an example set of input-output pairs. Specifically, for classification tasks

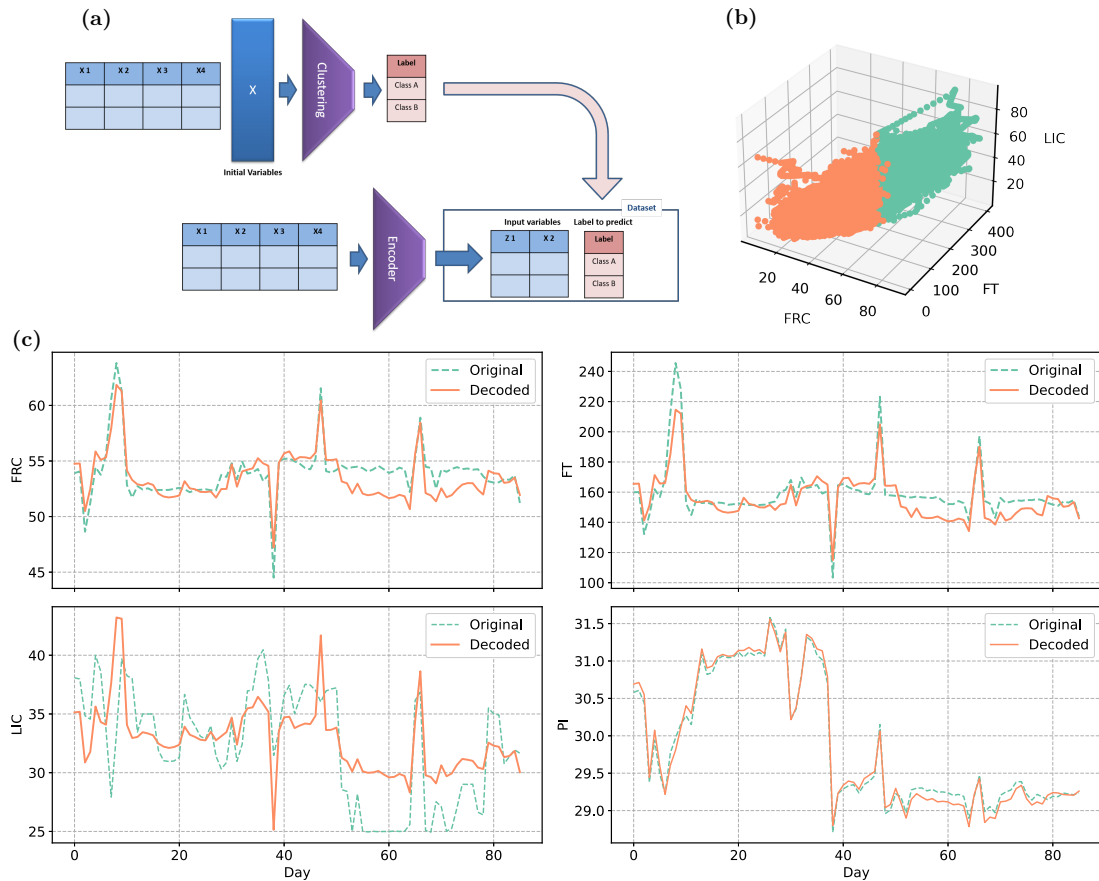


Figure 6.2: **(a)** The approach followed in this project: a clustering algorithm was used to define two categorical classes (Class A and Class B). Then, an autoencoder was used to reduce the dimensionality of the problem. Finally, a classifier was used to predict Class A and Class B identified with the clustering algorithm. **(b)** Results of the clustering algorithm KMeans on the input data. In particular, only the features FRC, FT and LIC are shown. The different colour indicates the different label (or class) assigned to the data. **(c)** Plot of the decoded data on top of the original validation data averaged by day. Here the features were rescaled to their original range.

the network is presented a labelled dataset $S = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{0, 1, \dots, c-1\}\}_{i=1}^m$ consisting of a set of inputs \mathbf{x}_i and the corresponding correct labels y_i , with c being the total number of classes the inputs can be divided into (see upper left side of Fig. 6.3). Using this dataset, called *training set*, a neural network can be trained in a supervised fashion to learn the relationship between the input variables and the expected classification results. When the training is complete, the neural network model can be used for *inference*, that is for labelling previously unseen data. This property of neural networks, called *generalisation*, is ultimately the key figure that distinguishes them from standard fitting techniques, making them incredibly powerful tools [112, 128, 178, 210].

When dealing with real world problems, such as classifying the operational status of a plant as “ok” or “faulty” based on the measurements from the sensors installed on the plant, it is often the case that a large number of input variables are available. In fact, measurements coming from tens of sensors need to be analyzed not only on their instantaneous values, but also on additional features computed on time intervals, such as moving averages, and minimal/maximal values trends. This leads to a situation where too many input variables are available in the dataset, and it is often ineffective to directly feed them into the neural network classifier. With such a large number of variables, correlation analysis and feature engineering are often performed to focus only on the

most influencing variables, and only after these preprocessing steps the neural network can be used effectively. Another strategy is to use a *dimensionality reduction* approach, consisting in computing a new set of variables, smaller than the initial one, incorporating most —ideally all— of the information contained in the original data. These new compressed data are then used as inputs to the classifier, as shown in Fig. 6.3a.

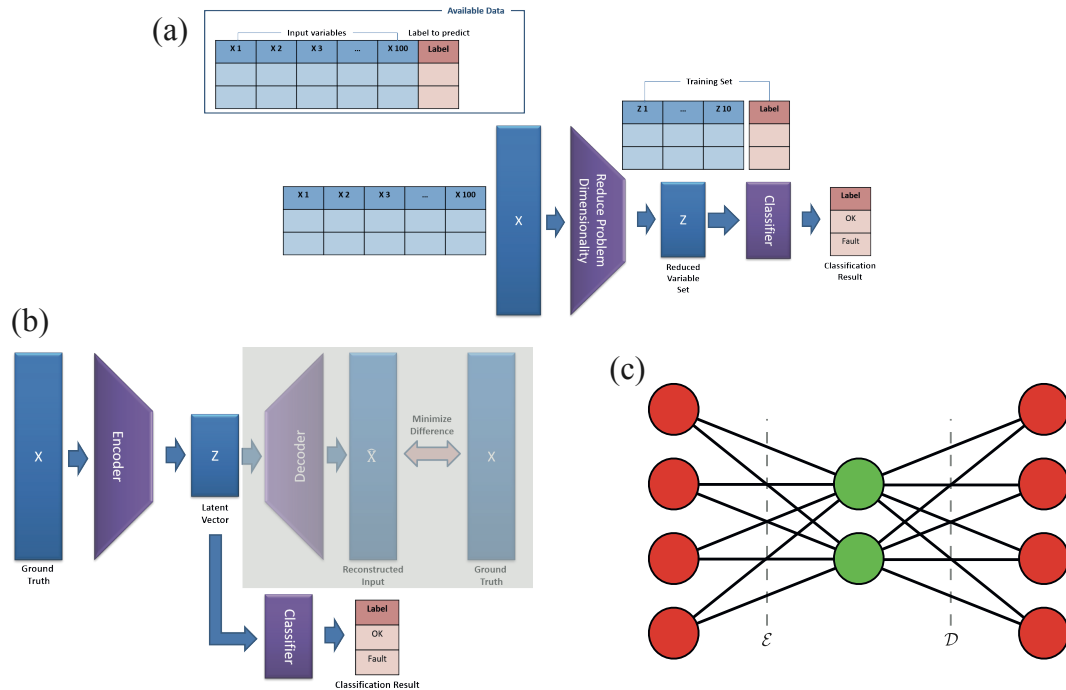


Figure 6.3: (a) Reducing the dimensionality of a classification problem. (b) Using Autoencoders to reduce the dimensionality of a problem and solving the classification problem on the reduced variable set. (c) Schematic representation of the neural network autoencoder architecture. The input neurons in red are mapped to an hidden layer (in green) of lower dimension, storing the compressed information. Then, an output layer with the same number of neurons as the input one, tries to restore the original data with low error.

In order to reduce the problem dimensionality, methods such as PCA (Principal Component Analysis) or SVD (Singular Value Decomposition) [128] are typically used. However, these methods are based on linear decomposition of the initial variable space, and they could not be suitable when nonlinear relationships between the variables need to be kept into account.

6.3.1 Classical Autoencoders

An alternative method to reduce the dimensionality of the problem is to use so-called Autoencoders [112], as shown in Fig 6.3c. An autoencoder is a neural network composed of two modules, called *encoder* and *decoder*, designed in such a way that the subsequent application of the encoder and the decoder to the input data results into an output that is as close as possible to the input, i.e. the discrepancy between output and input is minimised. With such an approach, the encoder builds a compressed representation of the input data to be eventually used by the decoder to fully, and as faithfully as possible, reconstruct the input. This means that the compressed representation built by the encoder (often referred to as *latent vector*) contains the same information of the initial input space, or at least that minimum information is lost. Once the autoencoder has been trained to reconstruct the input, the latent vector can be used as the input space for the classifier. Therefore, the classification problem can be described as shown in Fig. 6.3b.

In our case study, we consider a neural autoencoder as shown in Fig. 6.3c. The original input variables are fed to the input neurons, which are then passed to an intermediate hidden level (shown in green) consisting of a number of neurons much smaller than the input. Finally, there is an output layer (shown in red) with the same number of neurons as the input. The neural network is trained in an unsupervised fashion in order to generate an output that is as close as possible to the input. Thus, if it is possible to reconstruct the input (with a minimum loss of fidelity) starting from the inner layer, this means that the inner layer contains the same information as the input, and therefore we can use the compressed layer as an input for the classifier. The presence of non-linear activation functions within the neural network, such as the Rectified Linear Unit $\text{ReLU}(x) = \max(0, x)$, or sigmoid $s(x) = 1/(1 + e^{-x})$ (see Eqs. (3.39)), ensures that the network can better capture non-linear relationships in the input variables compared to PCA or SVD.

6.4 Quantum Data Compression

In order to use a quantum pipeline to analyse the classical data coming from the sensors, we need to encode such data on a quantum state to be used as the input of the quantum autoencoder. As discussed in Sec. 3.3.2 regarding the Fourier expansion of quantum circuits, recent literature points out the importance of choosing a good encoding scheme, even though no standard procedure is yet available [4, 109, 174, 189, 209, 269].

Given the relatively simple and low dimensional nature of the data sets to be analyzed, we choose to use the phase encoding strategy introduced earlier when discussing models of quantum perceptrons [195, 293]. This strategy provides an effective way to load classical data into a quantum state, and also already proved useful in other machine learning tasks such as pattern-recognition [195, 293, 295, 296]. In particular, given a data sample $\mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbb{R}^N$, this is encoded on the quantum state of $n = \log_2 d$ qubits as follows (see Eq. (5.3))

$$|\psi_{\mathbf{x}}\rangle = \frac{1}{2^{n/2}} \sum_{i=0}^{2^n-1} e^{i x_i} |i\rangle \quad (6.1)$$

where the data \mathbf{x} are first re-scaled to fit into an appropriate range, such as $x_i \in [0, \pi]$. We refer to Chapters 4 and 5 for an extended discussion on this class of states for variational quantum procedures.

6.4.1 Quantum Autoencoder

Having fixed a data encoding strategy, we now build a variational quantum algorithm for data compression. In particular, borrowing from the classical machine learning literature, our goal is to implement a quantum autoencoder [38, 171, 253]. In classical autoencoders, the compression is built in the geometric structure of the neural network, since the input layer is followed by a much smaller hidden layer consisting of a number of neurons equal to the desired reduced dimension. This bottleneck forces the NN to learn a low dimensional representation of the inputs, which is stored in the intermediate hidden layer(s) of the network. However, this procedure cannot be straightforwardly applied to the quantum domain, because quantum computations follow a unitary, thus reversible, evolution. In fact, while classically it is possible to perform fan-in(fan-out) operations, that is arbitrarily reducing (increasing) the number of classical bits in the computation, such operations are irreversible, which prevents their direct implementation on a quantum computer. Alternatively said, it is not possible to eliminate or create new qubits during the execution of a quantum computation.

Nonetheless, it is possible to circumvent this issue as follows. Consider two quantum systems, denoted as system A and system B , and be $|\psi\rangle_{AB}$ the quantum state of the composite quantum system AB . Our goal is to compress the information stored in the composite state in a lower dimensional representation, for example given by the state of subsystem A only, with system B

being safely discarded. We can formalise this intuition in the following way: denote with $\mathcal{E}(\boldsymbol{\theta})$ a quantum encoding (in the sense of *compressing*) operation depending on variational parameters $\boldsymbol{\theta}$, then the desired compression task consists in the operation

$$\mathcal{E}(\boldsymbol{\theta})|\psi\rangle_{AB} = |\phi\rangle_A \otimes |\text{trash}\rangle_B, \quad (6.2)$$

where the state $|\psi\rangle_{AB}$ of the composite system AB is compressed on the state $|\phi\rangle_A$ of subsystem A only, and the system B is mapped to a fixed reference state of choice, called *trash* state, for example being the ground state $|\text{trash}\rangle_B = |0\rangle^{\otimes |B|}$, with $|B| = \dim(\mathcal{H}_B)$ being the dimension of the Hilbert \mathcal{H}_B space associated to system B .

It is clear that the goal of the encoder is to *disentangle* the two systems in such a way that one of them, the trash system, goes to the fixed reference state, while the other contains all the original information of the full quantum state. In order to recover the original quantum state $|\psi\rangle_{AB}$, it is then possible to act with a *quantum decoder* operation $\mathcal{D}(\boldsymbol{\theta})$, defined as $\mathcal{D}(\boldsymbol{\theta}) := \mathcal{E}(\boldsymbol{\theta})^\dagger$. Indeed, acting with the decoder on the compressed state yields the original state, namely

$$\mathcal{D}(\boldsymbol{\theta})(|\phi\rangle_A \otimes |\text{trash}\rangle_B) = \mathcal{D}(\boldsymbol{\theta})(\mathcal{E}(\boldsymbol{\theta})|\psi\rangle_{AB}) = (\mathcal{E}(\boldsymbol{\theta})^\dagger \mathcal{E}(\boldsymbol{\theta}))|\psi\rangle_{AB} = |\psi\rangle_{AB}. \quad (6.3)$$

Thus, suppose having compressed the information stored in the quantum state of a composite system into one of its subsystems. Then, it is always possible to retrieve the original information by coupling such information-carrying system with some new qubits initialised in the $|\text{trash}\rangle$ state, and then act on them with the quantum decoder operator, as schematically represented in Fig. 6.4.

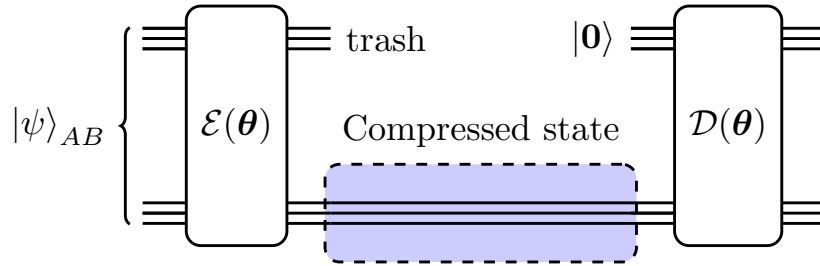


Figure 6.4: Schematic representation of the generic quantum autoencoder algorithm. The composite input quantum state $|\psi\rangle_{AB}$ is disentangled, so that system A carries the compressed information, and system B , called “trash” system, is mapped to a reference quantum state of choice like $|\text{trash}\rangle_B = |0\rangle_B$, and it can be discarded. Such procedure may then be reversed by coupling the information-carrying system A with a new set of clean qubits, and then applying a joint quantum decoder operation $\mathcal{D}(\boldsymbol{\theta}) := \mathcal{E}^\dagger(\boldsymbol{\theta})$ to retrieve the original state.

Of course, this only holds in the ideal case where the encoder perfectly manages to disentangle the subsystems A and B , by obtaining the product state in Eq.(6.2). In practice, this is never the case since the input state $|\psi\rangle_{AB}$ depends on the classical input data via the phase encoding, and these states cannot be exactly disentangled, in general. In fact, after discarding the trash system B , the compressed state A is no more a pure state, rather a mixed state given by the density matrix $\rho_A = \text{Tr}_B[(\mathcal{E}(\boldsymbol{\theta})|\psi_{AB}\rangle)(\langle\psi_{AB}|)\mathcal{E}(\boldsymbol{\theta})^\dagger]$. However, upon optimization of the variational parameters $\boldsymbol{\theta}$, the trained encoder creates a final state as close as possible to the target product state of Eq. (6.2).

Training the quantum autoencoder The initial quantum state $|\psi\rangle_{AB}$ is obtained by using phase encoding to load the classical information on the phase of the quantum state, with the following scheme. Be $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^d, i = 1, \dots, m\}$ the set containing the classical data to be analyzed, then the quantum autoencoder is trained using the quantum states obtained as $\{|\psi_{\mathbf{x}}\rangle = \sum_i e^{i\mathbf{x}_i} |i\rangle | \forall \mathbf{x} \in \mathcal{X}\}$. In our specific case, the classical data are four dimensional $d = 4$ and

thus we only need $n = \log_2 d = 2$ qubits to encode the data. This in turn implies that the compressed system A and the trash subsystem B consist of a single qubit each.

Given the input data, the variational parameters $\boldsymbol{\theta}$ of the encoder $\mathcal{E}(\boldsymbol{\theta})$ are optimised in order to rotate the trash qubit as close as possible to the target trash state, which we choose to be $|\text{trash}\rangle := |0\rangle$. This is achieved by means of a training procedure whose aim is to find optimal parameters $\boldsymbol{\theta}^*$ such that the loss function $L(\boldsymbol{\theta})$ characterising the task is minimised. That is, the goal of training is to find

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \quad \text{with} \quad L(\boldsymbol{\theta}) = \frac{1}{m} \sum_{j=1}^m \left| 1 - \langle Z_B \rangle_j \right|, \quad (6.4)$$

where we have defined

$$\langle Z_B \rangle_j := \langle \psi_{\mathbf{x}_j} | \mathcal{E}^\dagger(\boldsymbol{\theta})(\mathbb{I}_A \otimes Z_B) \mathcal{E}(\boldsymbol{\theta}) | \psi_{\mathbf{x}_j} \rangle \quad (6.5)$$

as the mean value of the Pauli-Z operator evaluated on the trash system B , after the encoder $\mathcal{E}(\boldsymbol{\theta})$ acted on the input quantum state $|\psi_{\mathbf{x}_j}\rangle$ depending on the j -th sample \mathbf{x}_j . The loss function used in Eq. (6.4) is referred to as *Mean Absolute Error* (MAE) in the classical machine learning literature, and together with the *Mean Squared Error* (MSE) is the one of the most commonly employed loss functions in supervised regression tasks, which is also our case. Note that the loss function is faithful, in the sense that it reaches its global minimum $L(\boldsymbol{\theta}^*) = 0$, only when $\langle Z_B \rangle_j = 1, \forall j = 1, \dots, m$, that is when the trash qubit is always and perfectly disentangled from the other qubit, and mapped to the target trash state $|0\rangle$. A schematic representation of the quantum circuit used for the training procedure is explicitly shown in Fig. 6.5a.

Variational ansatz The actual quantum circuit implementation of the encoder $\mathcal{E}(\boldsymbol{\theta})$, hence the decoder $\mathcal{D}(\boldsymbol{\theta})$, is arbitrary, and in fact different variational ansätze have been proposed in the quantum machine learning literature [29, 58, 192, 201], of which we gave an extended overview previously in Sec. 2.2.2. In our case, we are dealing with only two qubits, and the most general ansatz consists of repeated applications of single qubit rotations and two-qubits entangling gates. In fact, having in mind to keep the parameters count and the overall circuit complexity low, we hereby propose a minimal yet efficient variational autoencoder consisting of two layers of Pauli-Y rotations $R_Y(\theta) = e^{-iY\theta/2}$ (2.6) and a CNOT, followed by a final layer of rotations, as schematically depicted in Fig. 6.5b.

6.5 Experiments and Results

In this section we discuss the experiments implementing the classical and quantum data analysis approaches described above for the data compression and classification tasks.

6.5.1 Data compression

Classical autoencoder The classical neural network autoencoder was implemented with the Keras library of TensorFlow [2], and it consists of two dense layers in a 4-2-4 structure as in Fig. 6.3c, with sigmoid activation function.

The input data consists of a time series with 2873893 samples, 25% of which are used as validation data, and the rest for training. Before training, features were transformed with a MinMax scaler, which scaled each feature to fit in the range $[0, 1]$. After the learning phase, the average reconstruction error \bar{e} , defined as

$$\bar{e} := \frac{1}{m} \sum_{i=1}^m \left(\frac{1}{4} \sum_{j=1}^4 \frac{|x_{\text{decoder}}^{(i,j)} - x_{\text{original}}^{(i,j)}|}{|x_{\text{original}}^{(i,j)}|} \right) \quad (6.6)$$

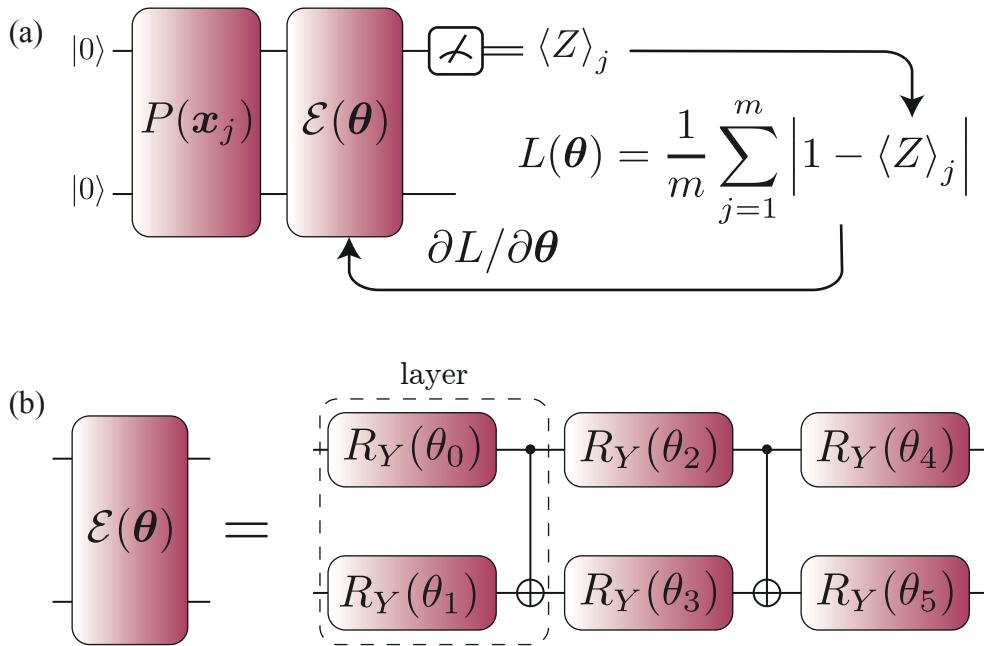


Figure 6.5: **(a)** Quantum circuit used to train the quantum autoencoder. A register initialised in the ground state $|00\rangle$ is first subject to the phase encoding operation denoted by $P(\mathbf{x})$, and then goes through the quantum encoder $\mathcal{E}(\boldsymbol{\theta})$. Then the trash qubit is measured, and the mean value of the Pauli operator $\langle Z \rangle$ is evaluated. Such value is then plugged into the loss function $\mathcal{L}(\boldsymbol{\theta})$ to drive the learning process. **(b)** Circuit representation of the quantum encoder $\mathcal{E}(\boldsymbol{\theta})$. Two layers of Pauli-Y rotations and a CNOT, are followed by a final layer of Pauli-Y rotations. In total, the circuit has 6 trainable parameters. The decoder $\mathcal{D}(\boldsymbol{\theta}) = \mathcal{E}^\dagger(\boldsymbol{\theta})$ is obtained by reversing the order of the operations, and changing the sign of the rotations angles.

amounts to 5%, and in Fig. 6.2 we show a comparison of the original against reconstructed data averaged by day, for the validation dataset. As we can see, the decoder shows quite good performance in the reconstruction of the input data for 3 of the 4 variables. For the ‘LIC’ variable, the median of the distribution of the reconstructed data coincides with the one of the original data, though the fluctuations are not very well described. There is no obvious a priori reason for the imperfect reconstruction of this particular variable, and this may well be a shortcoming of the autoencoding approach, which focuses more on the other variables to achieve a good-enough reconstruction scheme.

In the following step we used the two latent variables from the compressed layer as input for a supervised classification algorithm, to predict the class assigned at the beginning through the clustering algorithm. We expect that, if the compressed vector is a suitable representation of the input data, a classification algorithm would be able to achieve very good performances.

Quantum autoencoder The quantum autoencoder was simulated using a combination of PennyLane [27], TensorFlow [2] and Qiskit [5], and the optimisation was performed using the automatic differentiation techniques implemented by these libraries. While automatic differentiation is only possible when performing a classical simulation of the quantum algorithm, in realistic scenarios of optimising a quantum circuit on real quantum hardware one can resort to parameter-shift rules (2.55) to estimate gradients and optimise the variational parameters [209, 270].

The variational circuit was trained using the Adam optimiser [165] with learning rate set to $\eta = 0.001$, to update the six variational parameters $\boldsymbol{\theta} = (\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$. The training was

performed using mini-batches of size 20 for a total training set consisting of $m = 10040$ samples. In Fig. 6.6 it is shown the optimisation process across epochs of learning, both for the training loss, and for a validation set of 520 samples. Before the phase encoding process, the classical data $\{\mathbf{x}_i\}_i$ were normalised as $\mathbf{x}_i \leftarrow \pi \cdot \mathbf{x}_i / \|\mathbf{x}_i\|$. It is clear that the quantum encoder is effectively trained, with the loss reaching the minimum value of $L(\boldsymbol{\theta}^*) = 0.0058$.

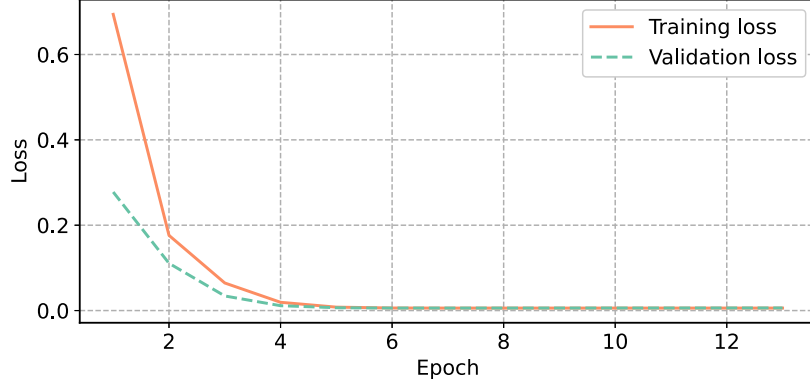


Figure 6.6: Optimisation of the quantum encoder, $\mathcal{E}(\boldsymbol{\theta})$, showing the training and validation loss evaluated with data sets containing 10040 and 520 samples, respectively. The minimum of the loss at the end of training amounts to $L(\boldsymbol{\theta}^*) = 0.0058$.

With a trained encoder, we can now proceed to investigate the quality of the data compression provided by the algorithm. The state of the qubits A and B after the quantum encoder operator consists of a general two-qubit state

$$|\Psi\rangle_{AB} = a|0\rangle_B \otimes |0\rangle_A + b|0\rangle_B \otimes |1\rangle_A + c|1\rangle_B \otimes |0\rangle_A + d|1\rangle_B \otimes |1\rangle_A, \quad (6.7)$$

where, if the encoder has been successfully trained, the probability of measuring qubit B in state $|1\rangle$, $p_1 = |c|^2 + |d|^2$, is much smaller, ideally zero, than the probability of finding it in $|0\rangle$, namely $p_1 \ll p_0 = |a|^2 + |b|^2$. Thus, in order to obtain a compressed pure state for qubit A rather than a mixed one, we could post-select state $|\Psi\rangle_{AB}$ on measuring the trash qubit in state $|0\rangle$. In this case, let $\hat{\Pi}_0^B = |0\rangle\langle 0|_B \otimes \mathbb{I}_A$ be the projector on state $|0\rangle$ for system B , then the composite state is projected to

$$|\Psi\rangle_{AB} \longrightarrow \frac{\hat{\Pi}_0^B |\Psi\rangle\langle\Psi| \hat{\Pi}_0^B}{\text{Tr}[\hat{\Pi}_0^B |\Psi\rangle\langle\Psi| \hat{\Pi}_0^B]} = |0\rangle\langle 0|_B \otimes |\psi_c\rangle\langle\psi_c|_A, \quad |\psi_c\rangle_A = \frac{a|0\rangle_A + b|1\rangle_A}{\sqrt{|a|^2 + |b|^2}}. \quad (6.8)$$

If we wish to retrieve the original information, now stored in compressed form in the state $|\psi_c\rangle_A$ of system A only, we can couple this system to a new qubit initialised in $|0\rangle$, and then apply the quantum decoder, as shown in Fig. 6.4. An example of this procedure is shown in Fig. 6.7, where the reconstruction performances of the quantum autoencoder are evaluated on a test set consisting of 1000 samples coming from the original dataset. In the case of Fig. 6.7, the average reconstruction error defined in Eq. (6.6) amounts to $\bar{\epsilon} = 5.4\%$, confirming that the quantum autoencoder can successfully compress and then retrieve information with low error.

It is important to note that the results discussed in this section were obtained with an exact simulation of the wavefunction of quantum systems, using Qiskit's `statevector_simulator`. This allows for a direct access to the amplitudes of the quantum states, and thus recover the final phases of the decoded state $|\varphi_{\text{decoder}}\rangle = \mathcal{D}(\boldsymbol{\theta})(|0\rangle \otimes |\psi_c\rangle_A)$. However, in a real case scenario with a quantum hardware, it is not possible to perfectly retrieve the phases of the decoded state $|\varphi_{\text{decoder}}\rangle$, since one would need to perform quantum tomography of such state, and even in that

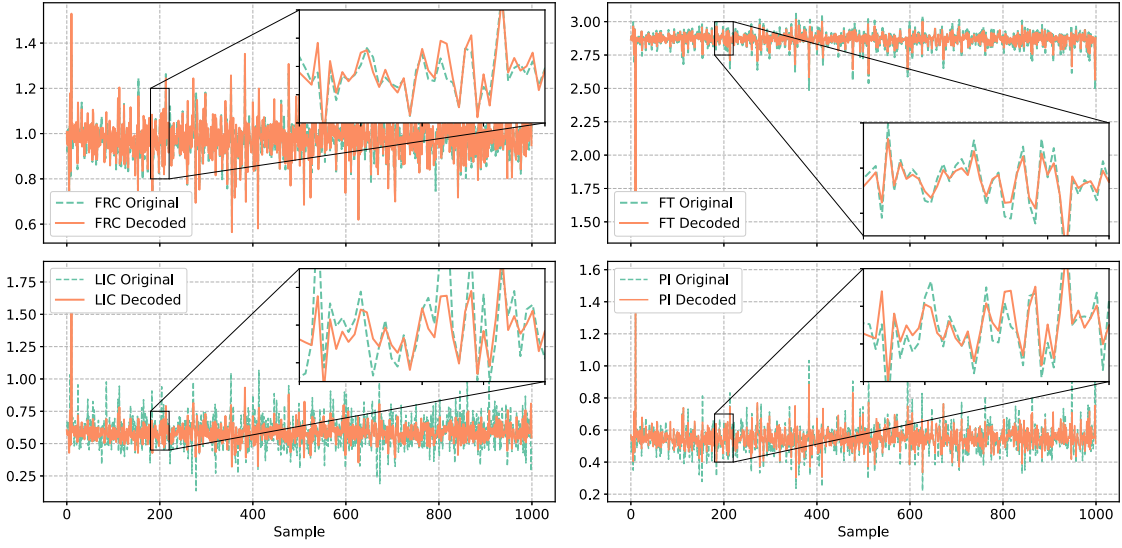


Figure 6.7: Performances of the quantum autoencoder in a compression and decoding task. Each plot shows one of the input features labelled ‘FRC’, ‘FT’, ‘LIC’, ‘PI’, as reconstructed by the quantum autoencoder (‘decoded’) confronted with the original sample (‘original’). This plots are evaluated on a test set consisting of $m = 1000$ samples. The average reconstruction error $\bar{\epsilon}$ as defined in the main text in Eq. (6.6), amounts to $\bar{\epsilon} = 5.4\%$. This results were obtained using the IBM Qiskit `statevector_simulator`.

case results could only be obtained up to an arbitrary constant, due to quantum measurement outcomes following Born’s rule. Thus, while such reconstruction test would prove much harder to be performed on a real device, the results in Fig. 6.7 obtained with the simulator are still relevant in checking the inner working of the quantum autoencoder, and that it is actually able to perform the task it was designed for, even if it is not currently accessible by a real experimenter.

We hereby discuss a second possible approach for measuring the faithfulness of the reconstruction, which albeit being indirect does not require state tomography and is thus more readily compatible with actual runs on quantum processors. The performances of the quantum autoencoder can be tested measuring the fidelity [323] $F(\rho_x, \eta_x^\theta) = \text{Tr}[\rho_x \eta_x^\theta]$ between the initial pure state $\rho_x = |\psi_x\rangle\langle\psi_x|$ obtained through phase encoding (6.1), and the generally mixed state obtained through the quantum circuit autoencoder of Fig. 6.4, defined as

$$\eta_x^\theta := D(\theta)[|0\rangle\langle 0| \otimes \text{Tr}_B[E(\theta)[\rho_x]]], \quad (6.9)$$

where $E(\theta)$ and $D(\theta)$ represent the superoperators corresponding to the encoder $\mathcal{E}(\theta)$ and decoder $\mathcal{D}(\theta)$ operators, respectively. Clearly, the larger the fidelity the better, since it corresponds to the quantum autoencoder being able to recreate states that are very close to the initial ones. Using this figure of merit, post-selecting on the trash subsystem B is not necessary since qubit A can be directly coupled to a new qubit initialised in $|0\rangle$, to then act with the decoder and with the evaluation of $\text{Tr}[\rho_x \eta_x^\theta]$. There are various techniques to evaluate state overlaps on quantum hardware [64, 195], the most common one being the SWAP test, and here we use the so-called *compute-uncompute* method, whose circuit is shown in Fig. 6.8.

Using a test set of $m = 1000$ samples, a simulation of the trained quantum autoencoder, even including stochastic measurement outcomes with $n_{\text{shots}} = 10^4$ shots, yields an average fidelity

$$\mathbb{E}[\text{Tr}[\rho_x \eta_x^\theta]] = \frac{1}{m} \sum_{j=1}^m \text{Tr}[\rho_{x_j} \eta_{x_j}^\theta] = 0.975 \pm 0.001,$$

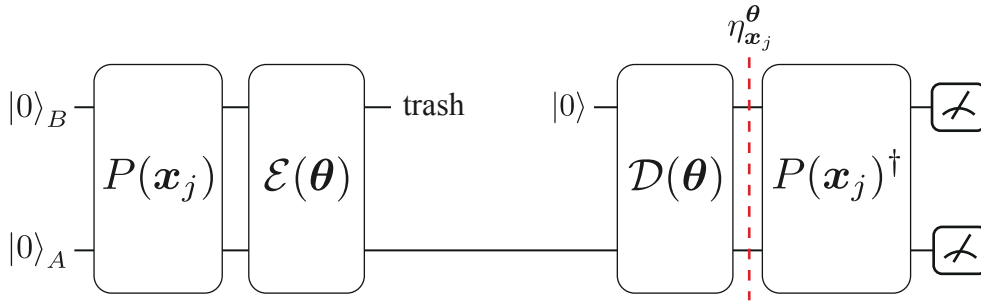


Figure 6.8: Circuit to evaluate the fidelity $F(\rho_x, \eta_x^\theta) = \text{Tr}[\rho_x \eta_x^\theta]$ between the initial pure state $\rho_x = |\psi_x\rangle\langle\psi_x|$ and the generally mixed state η_x^θ , obtained through the autoencoding procedure. The fidelity is obtained by counting the number of $|00\rangle$ outcomes at the end of the circuit. In fact, dropping the subscripts for simplicity, one has $p_0 = \text{Tr}[P(x)^\dagger \eta_x^\theta P(x) |0\rangle\langle 0|] = \text{Tr}[\eta_x^\theta P(x) |0\rangle\langle 0| P(x)^\dagger] = \text{Tr}[\eta_x^\theta |\psi_x\rangle\langle\psi_x|] = F(\rho_x, \eta_x^\theta)$.

which confirms again that the proposed variational quantum autoencoder is able to compress and later decode information.

6.5.2 Classification

Classical classifier The supervised classification algorithm used is the `KNeighborsClassifier` as implemented in `scikit-learn`. `KNeighborsClassifier` assigns the class to a point from a simple majority vote based on the k nearest neighbours of that point. The number of nearest neighbours is a parameter of the algorithm, and after some trials we fixed it at $k = 100$, which correspond to an optimal trade-off between performances and computational efficiency.

The lowest panel of Fig. 6.9 shows the results of the classification, which is now anticipated but discussed later in comparison with the quantum algorithm results. In red and blue are the points that have been correctly classified, while in yellow and green are those which were misclassified. The classification accuracy, evaluated as the percentage of correctly classified data, reach a remarkably high value of 89.7%, indicating that the compressed vector is able to summarise the information carried by the input data.

Single qubit quantum classifier Once the quantum autoencoder has been trained to learn a compressed representation of the original information, the compressed quantum state can be used as input for a classification task. We expect that, if the compressed information is a suitable representation of the input data, the classification algorithm would be able to learn the classes assigned to the full-size input data through the clustering algorithm described in Sec. 6.2.

To do so, we can use the information-carrying qubit obtained with the encoder $\mathcal{E}(\theta)$, as input to a quantum classifier which is trained to learn the desired clustering of the original data. A quantum classifier is made of two parts: (i) a trainable parametrized operation, $U(\boldsymbol{\varphi})$, which tries to map inputs belonging to different classes in two distant regions of the Hilbert space, and (ii) a final measurement, which is used to extract and assign the label. Since we are dealing with a single qubit classifier, the most general transformation on a qubit is represented by the unitary matrix (5.17)

$$U(\alpha, \beta, \gamma) = \begin{bmatrix} \cos(\alpha/2) & e^{-i\gamma} \sin(\alpha/2) \\ e^{i\beta} \sin(\alpha/2) & e^{i(\beta+\gamma)} \cos(\alpha/2) \end{bmatrix}. \quad (6.10)$$

Thus, it is reasonable to use such operation as the trainable block of the classifier, since it ensures the greatest flexibility. Actually, as discussed later, the angle β in Eq. (6.10) does not influence the measurement statistics of the qubit, hence it has no influence on the training of the classifier. For this reason, it is kept fixed at $\beta = 0$, and the actual trainable gate used is $U(\alpha, 0, \gamma) = U(\alpha, \gamma)$.

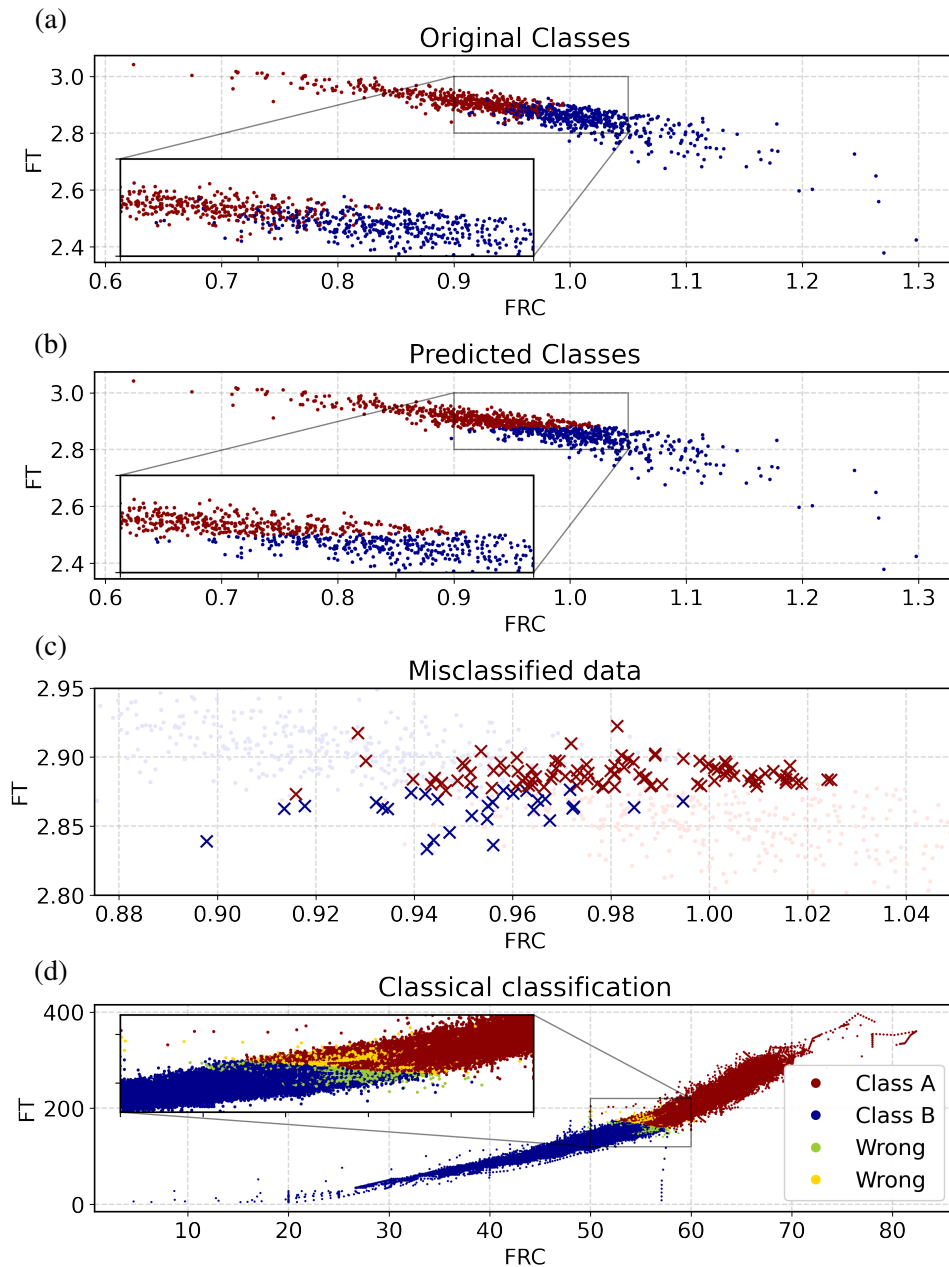


Figure 6.9: Results of the classification task performed by the quantum classifier, for a test set of size $m = 10^3$ samples. **(a)** Plot of the original data with the colour indicating the two different classes. Note that for simplicity, only the FRC and FT features are shown. **(b)** Label assigned by the trained quantum classifier. **(c)** Focus on the data that are mislabelled by the classifier. The colour indicates the label assigned by the quantum classifier, and the “cross” marker means that the data were misclassified. Note that these samples lay on the border of separating the two classes. The accuracy, evaluated as the percentage of correctly classified data, amounts to 87.4%. **(d)** Result of the classification using the classical autoencoder followed by a KNN clustering procedure. Note that the axis are different from the quantum case due to normalisation of the features. In this case the classification accuracy amounts to 89.7%

As for the label assignment, since the measurement process of a qubit has only two possible outcomes, these are interpreted to be the two possible values for the labels, namely “Class A” and “Class B” which were described earlier in Sec. 6.2. Specifically, a label is assigned based on a

majority vote on multiple shots of the same quantum circuit, that is an input is assigned to “Class A” if the majority of measurement gave $|0\rangle$ as outcome, and “Class B” otherwise. Formally, let $\rho_{\mathbf{x}}^A = \text{Tr}_B[\mathcal{E}(\boldsymbol{\theta})(|\psi_{\mathbf{x}}\rangle\langle\psi_{\mathbf{x}}|)\mathcal{E}(\boldsymbol{\theta})^\dagger]$ be the compressed quantum qubit, then the label is assigned based on the decision rule

$$\hat{y}_i = \begin{cases} 0 & \text{if } p_0 = \text{Tr}[|0\rangle\langle 0|U(\boldsymbol{\varphi})\rho_{\mathbf{x}_i}^AU(\boldsymbol{\varphi})^\dagger] \geq 0.5 \\ 1 & \text{otherwise} \end{cases}, \quad (6.11)$$

where p_0 denotes the probability that the measurement yields $|0\rangle$ outcome. As mentioned earlier, one can check easily that p_0 does not depend on the angle β of the unitary $U(\alpha, \beta, \gamma)$, and for this reason it is set to zero, yielding the variational unitary $U(\alpha, 0, \gamma) = U(\alpha, \gamma)$.

The loss function used to drive the training of the unitary $U(\alpha, \gamma)$ is the categorical cross entropy already introduced in Eq. (3.9), defined as

$$\mathcal{L}(y_i, \hat{y}_i) = -(1 - y_i)\log(1 - \hat{y}_i) - y_i\log(\hat{y}_i), \quad (6.12)$$

where y_i is the correct label, and \hat{y}_i is the label assigned by the quantum classifier, and the optimiser used is COBYLA [235] as implemented in SciPy’s Python package [308].

Figure 6.9 shows the results of the classification obtained after the optimization of the variational parameters $\boldsymbol{\varphi} = (\alpha, \gamma)$, for a test set of $m = 10^3$ samples. The accuracy, measured as the ratio of correctly classified to total samples, is measured to be 87.4% when evaluated with exact simulation of the quantum circuit. As clear from the figure, the misclassified data are only those located near the edge connecting the two classes. In fact, in this region, the samples are not neatly divided but rather a blurred border exists. On the contrary, given its relatively simple structure, the quantum classifier learns essentially a straight cut of the data in this region, thus committing some labelling errors.

This should not come as a surprise since, as seen when discussing the Fourier representation of variational circuits in Sec. 3.3.2.2, a single-qubit classifier can only learn simple functions (i.e. sine functions) of the input data if there is not enough input redundancy [109, 116, 174, 189, 229, 269]. However, note that the dependence of the classification on the original data is strictly non-linear, since the classical data first go through a classical preprocessing step, then are loaded onto the quantum states by means of rotations, and finally undergoes the encoding procedure which scrambles information even more.

It is interesting to notice that the classification performances remain stable even when including sources of noise, such as stochastic measurement outcomes. Indeed, a simulation of the circuit using $n_{\text{shots}} = 1024$ on $m = 10^3$ samples yields an accuracy of about 82.5%, which is only slightly lower than the exact case corresponding to an infinite number shots. In addition, the classifier proves robust even when tested on real quantum hardware. In fact, the circuit for the trained classifier was tested against IBM’s `ibmq_x2` quantum chip (accessed May 2021) but with a smaller test set of $m = 75$ samples, due to limitations in the device usage. In this case, using $n_{\text{shots}} = 1024$ shots per circuit, and averaging over 5 executions with different test samples, the classification accuracy (evaluated again as the percentage of correctly classified data) was found to be $(82.3 \pm 1.3)\%$, indeed very close to the simulation including only measurement noise, and not much different from the noiseless result.

6.6 Conclusions

We have presented a direct comparison between quantum and classical implementations of a neural network autoencoder, followed by a classifier algorithm, applied to sample real data coming from one of Eni’s plants, in particular from a first stage separator. While the achievement of a clear quantum machine learning advantage with variational algorithms is still disputed [4, 140, 141], this work sets a milestone in the field of quantum machine learning, since it is one of the first examples

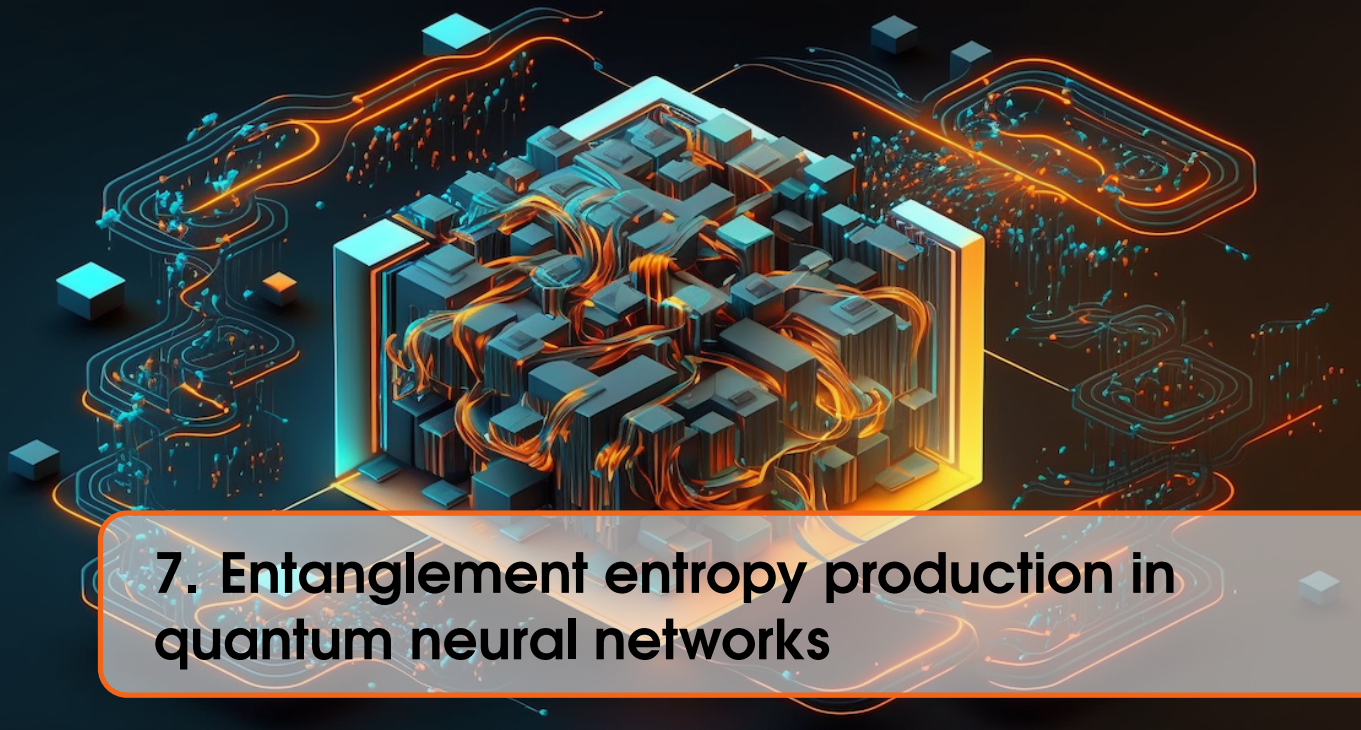
of direct application of quantum computing software and hardware to analyse real data sets from industrial sources.

As a first step, we have implemented and analyzed the performance of a variational quantum autoencoder to compress and subsequently recover the input data. We verified its performances using full simulation of the wavefunction, which allowed us to evaluate the average reconstruction error to about $\bar{\epsilon} = 5\%$ —essentially identical to the classical autoencoder— thus confirming the capability of the quantum autoencoder to effectively store a compressed version of the original data set, and then being able to recover it. In addition, we also checked the correctness of the quantum autoencoding procedure by evaluating the quantum fidelity between original and decoded quantum states, which were again found to be very similar to each other, even in the presence of simulated stochastic measurement noise.

Once the optimal parameters for the quantum autoencoder were determined during the training phase, we used the compressed quantum state as input to a quantum classifier, with the goal of performing a binary classification task. The algorithm achieved an accuracy above 87%, absolutely comparable to that achieved in the classical setting using the neural network autoencoder followed by a nearest-neighbours classifier, thus indicating again that the quantum algorithm is able to correctly compress the relevant information of the input data. We also tested the performance of the full quantum pipeline (given by the quantum autoencoder plus the classifier) on actual and currently available IBM superconducting quantum hardware, obtaining a classification accuracy of 82%, which is only slightly smaller than the ideal result.

The small size of current quantum devices and their relatively high noise levels make it hard to run actually relevant and large scale computations, thus making an effective quantum advantage out of reach. On the other hand, in this Chapter we provided a successful proof-of-concept demonstration that an original quantum autoencoder and a quantum classifier can actually reach the same level of accuracy as standard classical algorithms, on a data set that is sufficiently low dimensional to be handled on actual near-term quantum devices. In addition, it is worth emphasising that the quantum autoencoder allows to obtain results that are quantitatively comparable to the classical algorithm by using only 6 parameters instead of 16, thus displaying an increased efficiency in terms of number of trainable parameters already reached on NISQ devices. With continuing progress in quantum technologies and quantum information platforms, we envision the execution of the very same quantum algorithms on larger scales, possibly reaching the threshold for a classically intractable problem. We believe that these results take the first foundational steps towards the application of usable quantum algorithms on NISQ devices for industrial data.

Although the use of quantum resources may offer computational advantages over purely classical methods, the latter are incredibly versatile tools, not only capable of giving rise to incredibly successful machine learning models, but also to provide an effective description of quantum system themselves. Thus, in order for a (variational) quantum algorithm to deliver a meaningful advantage, it must be hard to simulate via classical methods. The topic of the next Chapter is more fundamental and unrelated to practical use-cases, but instead addresses the classical simulability of quantum circuits under the lens of the entanglement, by studying the entanglement produced inside common variational quantum circuits.



7. Entanglement entropy production in quantum neural networks

7.1	Introduction	124
7.2	Methods	125
7.2.1	Tensor Networks and Matrix Product States	125
7.2.2	Entanglement measure in Matrix Product States	126
7.2.3	Entanglement entropy in random quantum states	127
7.2.4	Quantum Neural Networks as Parameterised Quantum Circuits	128
7.2.5	Randomness, Entanglement and Trainability	129
7.3	Results	130
7.3.1	Alternating vs. Sequential data reuploading	130
7.3.2	Entanglement distribution across bonds	131
7.3.3	Entanglement scaling with increasing depth	134
7.3.4	Entanglement Speed	135
7.3.5	Expressibility	138
7.3.6	Distribution of the singular values	138
7.4	Discussion	139
7.5	Conclusion	141

In this chapter¹, we use tensor networks techniques to characterise the entanglement features of several recent proposals for Quantum Neural Networks. Specifically, we study the production of entanglement in random parameterised quantum circuits of up to fifty qubits, showing that their entanglement, measured in terms of entanglement entropy between qubits, tends to that of Haar distributed random states as the depth of the QNN is increased. We certify the randomness of the quantum states also by measuring the expressibility of the circuits, as well as using tools from random matrix theory. We show a universal behaviour for the rate at which entanglement is created in any given QNN architecture, and consequently introduce a new measure to characterise the entanglement production in QNNs: the entangling speed. These results characterise the entanglement properties of quantum neural networks, and provides new evidence of the rate at which these approximate random unitaries.

¹The content of this chapter is based on the author's work [18], and all the figures in this chapter are taken from, or are adaptations of, the figures present in such work.

7.1 Introduction

The topic of this chapter is the study of the entanglement properties of quantum neural networks when these initialised with random parameters. We employ methods from the tensor network literature, namely Matrix Product States (MPS), to study the entanglement generated in various QNNs architectures composed of up to 50 qubits. Since MPS are a very powerful tool for simulating quantum systems with bounded entanglement, if a quantum neural network can only access low entangled states, it can be easily simulated, which spoils any hope of achieving a concrete quantum advantage. Thus, using entanglement entropy among qubits as a figure of merit, we evaluate the entanglement capabilities of some of the most common and promising QNN architectures [4, 281]. We consider several QNNs with different combinations of feature maps \mathcal{F} and variational forms V and perform an extended numerical analysis varying: (i) the number of qubits n , (ii) the number of layers L in the network, (iii) the entangling topology of the circuit, (iv) the data re-uploading [229, 269] structure being either alternated or sequential. A summary of the circuit templates analysed in this study is shown in Fig. 7.1.

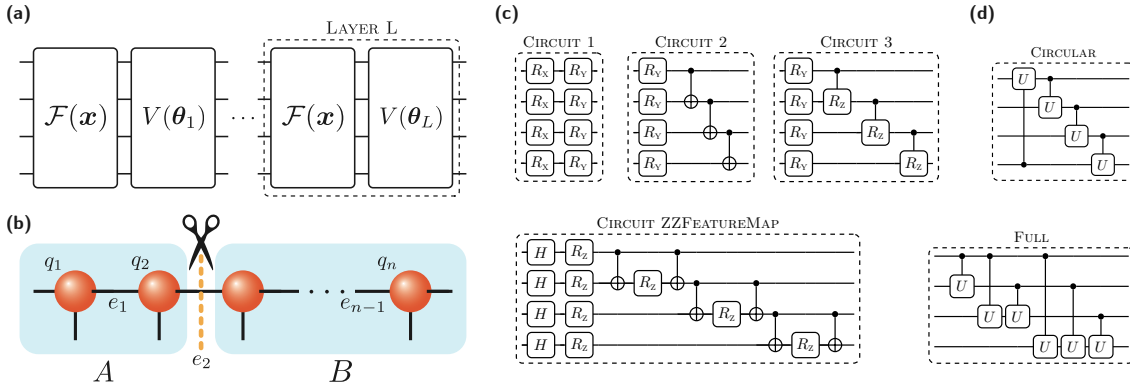


Figure 7.1: Graphical representation of QNN and MPS. **(a)** QNN structure with alternating feature map \mathcal{F} and variational ansatz V . Note that the ansatz parameters are different in each layer, while the feature map parameters are the same throughout the whole circuit. **(b)** MPS diagram. Each sphere is a tensor, representing a qubit q_j . The entanglement entropy between bi-partitions A and B is computed by "cutting" the connecting edge e_j . **(c)** Circuits analysed in the present study, depicted with a linear entanglement topology, i.e. entangling gates are only applied between nearest neighbours on a line. **(d)** Different entanglement topologies: circular, with the first and last qubit of the line connected, and full, where the entangling gates are applied between each pair of qubits (see Appendix D.3 for a clear definition and discussion). When using parameterized two qubits gates, like the controlled rotations in circuit 3, the entanglement maps are generalised to their parameterized version by using the corresponding parameterized operation. Note that the circuit templates 2 and ZZFEATUREMAP are those used in the QNN of [4], and also that circuits 1, 2 and 3 share similarities with circuits 1, 15, and 13 of [281], respectively.

For all the considered QNNs with nearest neighbour connectivity, as the number of layers L is increased, the entanglement generated inside the circuit grows, eventually reaching a plateau when $L \approx n$, where n is the number of qubits. This behaviour is associated with the typical entanglement of a random Haar-distributed quantum state. The choice of the entangling topology (nearest neighbours, circular, or all to all) clearly affects the rate of creation of entanglement in the circuit. We also point out that a careless definition of a full, i.e. all to all, connectivity map can effectively result in a linear nearest-neighbours interaction if unparameterized two qubits gates (CNOTs) are used, something apparently overlooked in the recent literature using this type of ansatz [4, 149, 296]. By bounding the entanglement generated by the circuit, we are able to simulate QNNs with MPS up to $n = 50$ qubits. It should be stressed that such simulations are exact up to a given number of layers,

after which a truncation of the entanglement via MPS is applied. By appropriately normalising the entanglement produced we show that all the points for a given QNN architecture follow the same curve, independently from the number of qubits. Thus, we exploit this behaviour to define a universal figure of merit given the QNN architecture, the entangling speed. This figure of merit characterises how fast the entanglement is produced by the QNN, with respect to the number of layers L .

In addition, we evaluate the expressibility measure of the considered QNNs as defined in [281] and argue that the optimality of the QNN introduced in [4] may be related to its good trade-off between mild entanglement production and high expressibility. Finally, we employ tools from random matrix theory, specifically convergence to the Marčenko-Pastur distribution, to further characterise the resemblance of the deep enough quantum neural networks to random unitary matrices. At last, we note that differently from [281] which bases their analysis on the Meyer-Wallach entanglement measure [207], in this study we make use of the entanglement entropy among subsystems, which allows for a more careful analysis of the entanglement distribution in the system, and it is also readily accessed in an MPS simulation with no computational overhead.

The chapter is organised as follows. In Sec. 7.2 we review the basis of tensor networks and MPS, and introduce the Von Neumann entropy as an entanglement measure. We then discuss the entanglement entropy properties of random quantum states. We proceed by discussing the most recent results on parameterized quantum circuits and QNNs, especially, on the relation between randomness, trainability, and entanglement found in these circuits. In Sec. 7.3 we show the results of our analysis for various QNN architectures, and discuss the results in Sec. 7.4. Finally, we discuss the implications of our analysis and possible routes for future investigations in Sec. 7.5.

7.2 Methods

7.2.1 Tensor Networks and Matrix Product States

An n -qubit quantum state is defined in a Hilbert space \mathcal{H} of dimension $\dim(\mathcal{H}) = 2^n$. The exponential scaling of \mathcal{H} with n makes the classical description of quantum states an exponentially expensive task. This problem is widely known in many-body quantum physics, and many different techniques have been developed to alleviate the issue, like the Density Matrix Renormalization Group (DMRG) or Tensor Network (TN) techniques [213, 280].

In this study, we use Tensor Network methods to efficiently describe the n -qubit state. In particular, we employ Matrix Product States (MPS), which are a specific tensor network ansatz particularly suited to represent 1-dimensional (i.e. like atoms on a chain, as in Fig. 7.1) quantum states [95]. The power of tensor networks lies in the assumption that we are only interested in a *tiny subspace* of the entire Hilbert space, namely the states that display a limited amount of entanglement.

An n -qubit pure state $|\psi\rangle \in \mathcal{H}$ can be written as a MPS as follows [95]

$$|\psi\rangle = \sum_{s_1, \dots, s_n=0}^1 \sum_{\alpha_1, \dots, \alpha_n=1}^{\chi} \mathbf{M}_{\alpha_1}^{[1], s_1} \mathbf{M}_{\alpha_2}^{[2], s_2} \dots \mathbf{M}_{\alpha_{n-2} \alpha_{n-1}}^{[n-1], s_{n-1}} \mathbf{M}_{\alpha_{n-1} 1}^{[n], s_n} |s_1 s_2 \dots s_n\rangle. \quad (7.1)$$

Each tensor $\mathbf{M}_{\alpha_i \alpha_{i+1}}^{[i], s_i}$ is a local description for the i -th site, which allows one to apply a *local* operator to a certain site without the need to change all the other coefficients. For a fixed s_i , $\mathbf{M}_{\alpha_i \alpha_{i+1}}^{[i], s_i}$ is a $\chi \times \chi$ complex matrix, meaning that Eq. (7.1) is the sum of basis elements weighted by matrix products. The integer χ is called the MPS *bond dimension*, and a sufficiently high χ is needed to express a general $|\psi\rangle$ in such form. However, MPS with a *lower* χ can still encode all the meaningful states, albeit clearly not *all* possible states. In particular, to correctly describe any quantum state the bond dimension needed is $\chi = d^{\lfloor \frac{n}{2} \rfloor}$, where d is the local dimension of the degrees of freedom ($d = 2$ for qubits). One can also efficiently evolve the state under the application of

2-qubit gates, using an approach known in the literature as time-evolving block decimation [223], and perform measurements. Simulations using MPS are not bounded by the number of qubits in the system, but by the amount of entanglement generated inside it, as we explain in detail in Section 7.2.2.

Nonetheless, while the use of an MPS simulation imposes some constraints on the maximum entanglement that it is possible to represent, this issue is relevant only for very deep circuits involving many qubits. Indeed, we reliably simulate circuit instances involving up to $n = 50$ qubits and moderate depth, which is already sufficient to provide clear insights on the entanglement entropy generated in such circuits. Moreover, as explained below in Sec. 7.2.2, during an MPS simulation one has constant access to the singular values of the quantum state, so the entanglement of the state can be calculated on the fly without any computational overhead. Thus, MPS are an effective tool to study the entanglement properties of quantum circuits, especially in regimes that cannot be easily accessed with a full-scale simulation of the statevector of the system.

7.2.2 Entanglement measure in Matrix Product States

Entanglement in quantum states can be evaluated using the so-called Von Neumann *entanglement entropy*. Let $\rho = |\psi\rangle\langle\psi|$ be the quantum state of a system of n qubits, and consider a bipartition A, B of such system of qubits n_A and $n_B = n - n_A$ respectively, like the one shown in Fig. 7.1b. The entanglement entropy of the subsystem A having reduced density matrix $\rho_A = \text{Tr}_B[\rho]$, is defined as

$$S(\rho_A) = -\text{Tr}[\rho_A \log \rho_A], \quad (7.2)$$

and quantifies the amount of entanglement shared between the parties A and its complement B^2 . If A and B are in a product state then $S(\rho_A) = 0$, while if the two subsystems share maximal entanglement one has $S(\rho_A) = n_A \log(2)$ [95]. An important property of Eq. (7.2) is that the entanglement entropy of the two subsystems is equal, namely $S(\rho_A) = S(\rho_B)$, as it can be easily checked using the Schmidt decomposition of the pure global state $\rho = |\psi\rangle\langle\psi|$ (see below).

It turns out that matrix product states are a natural tool to characterise the entanglement entropy of a quantum system. This can be illustrated by considering the simple case of a state of $n = 2$ qubits. Indeed, the statevector

$$|\psi\rangle = \sum_{i,j=0}^1 c_{ij} |ij\rangle \quad \text{with} \quad \sum_{i,j=0}^1 |c_{ij}|^2 = 1, \quad (7.3)$$

can be expressed in the Schmidt decomposition [220] as

$$|\psi\rangle = \sum_{\alpha=1}^{\chi_s} \lambda_{\alpha} |\xi_{\alpha}\rangle_1 \otimes |\eta_{\alpha}\rangle_2, \quad (7.4)$$

where χ_s is the *Schmidt rank*, λ_{α} are the Schmidt coefficients, and $\{|\xi_{\alpha}\rangle_1\}_{\alpha}, \{|\eta_{\alpha}\rangle_2\}_{\alpha}$ are orthonormal bases in the space of the first and second qubit respectively. Using the decomposition (7.4) in Eq. (7.2), the entanglement entropy between the two qubits then amounts to

$$S(\rho_A) = -\sum_{\alpha=1}^{\chi_s} \lambda_{\alpha}^2 \log \lambda_{\alpha}^2. \quad (7.5)$$

In an MPS simulation one always has access to a subset of the Schmidt coefficients, since such representation is built by iteratively applying the Singular Value Decomposition (SVD), a procedure equivalent to Schmidt-decomposing a quantum state. The reason why one has access only to subsets of the Schmidt coefficients is that the following conditions are imposed on them. Listing the coefficients in ascending order, i.e. $\lambda_0 \geq \lambda_1 \geq \dots \geq \lambda_{\chi_s}$, then

²Note that throughout the whole Chapter we consider logarithms in natural base e .

- Schmidt coefficients whose ratio with λ_0 is smaller than ε are discarded. The value of ε in this analysis is fixed at $\varepsilon = 10^{-9}$;
- only the first largest χ_{\max} coefficient are retained. The value χ_{\max} is called maximum *bond dimension*.

The approximation we are performing is the optimal one in terms of the represented entanglement. Then, the measure of entanglement for the MPS now becomes

$$S(\rho_A) = - \sum_{\alpha=1}^{\chi_{\max}} \lambda_{\alpha}^2 \log \lambda_{\alpha}^2. \quad (7.6)$$

As explained in detail in Appendix D.6, despite the approximations, the faithfulness of the simulation can be easily monitored. Finally, we remark that since we have constant access to the considered subset of Schmidt coefficients during the state evolution, we are able to compute the entanglement entropy of a quantum state without any computational overhead.

7.2.3 Entanglement entropy in random quantum states

In this section we briefly describe the entanglement features of uniformly distributed random pure quantum states, that is quantum states sampled according to the unique unitarily invariant probability distribution induced by the Haar measure. The Haar measure was already introduced and discussed previously in Chapter 2 for deriving the Barren Plateau phenomenon 2.2.4.2, and we hereby just recall some basic concepts necessary for the analysis presented in this Chapter.

Denoting by $\mathbb{U}(n)$ the group of $2^n \times 2^n$ unitary matrices, there is a unique unitarily invariant probability measure $\mu(U)$ defined on the group, and such measure is called *Haar measure* [93, 132, 205]. Unitary invariance corresponds to the requirement that the measure is invariant under translations in the space of unitary matrices, that is

$$\mu(MU) = \mu(UM) = \mu(U) \quad U, M \in \mathbb{U}(n). \quad (7.7)$$

The Haar measure induces a uniform probability distribution in the space of unitary matrices so that sampling a quantum state according to the Haar measure means randomly picking a state uniformly from the space of quantum states. We denote with $\mathcal{P}(n)$ such probability distribution.

We are interested in the entanglement features of random quantum states, particularly in the entanglement entropy. Let $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$ be a quantum state of n qubits sampled from the uniform distribution $|\psi\rangle \sim \mathcal{P}(n)$, and a bipartition of the n qubits system in two subsystems A and B , of size n_A and $n_B = n - n_A$ respectively. Then, for $n_A \leq n_B$, the expectation value of the entanglement entropy (7.2) corresponding to this cut amounts to the so-called Page value [132, 224]

$$\mathbb{E}[S(\rho_A)] = \sum_{j=d_B+1}^{d_A d_B} \frac{1}{j} - \frac{d_A - 1}{2d_B}, \quad (7.8)$$

where $d_B = 2^{n_B}$, $d_A = 2^{n_A}$ are the local dimensions of the two subsystems, and the expectation value is over the uniform probability distribution $\mathbb{E}(\cdot) = \mathbb{E}_{|\psi\rangle \sim \mathcal{P}(n)}(\cdot)$. One can check that the entanglement is highest whenever the two partitions have equal size $n_A = n_B = n/2$ (for n even, and similarly for n odd, $n_A = \lfloor \frac{n}{2} \rfloor$ and $n_B = \lceil \frac{n}{2} \rceil$).

From Eq. (7.8) it follows that $\mathbb{E}[S(\rho_A)] \geq \log d_A - d_A/2d_B$ [132], and since the maximum value of the entanglement entropy for such bi-partition is $\log d_A$, obtained if the subsystems A and B share maximal entanglement, one concludes that random states are generally highly entangled. Indeed, in ref. [132] it was shown that the probability that a random pure state has entanglement entropy lower than $\log d_A - d_A/2d_B$ is exponentially small. Thus, with very high probability, random quantum pure states are almost maximally entangled.

7.2.4 Quantum Neural Networks as Parameterised Quantum Circuits

In this section we briefly recall the main ideas and nomenclature of Variational Quantum Algorithms (VQAs) and Quantum Neural Networks (QNNs) to present the analysis in this Chapter in a self-consistent manner, but we refer to Section 2.2 and Section 3.2.2.2 for extended discussion on these topics.

Variational quantum algorithms are based on PQCs, which are quantum circuits in which some of the unitary operations are characterized by *variational* parameters to be adjusted in order to solve an optimization problem. The optimal parameters are found by minimizing a properly chosen cost (or loss) function encoding the task to be solved. Let U_{θ} be the unitary evolution implemented by a quantum circuit with tunable parameters θ , and O a Hermitian operator (an observable). The goal of variational quantum algorithms is to optimize the quantum circuit parameters θ in order to minimize the expectation value (or variations thereof, see Sec. 2.2)

$$f(\theta) = \langle O \rangle_{\theta} = \text{Tr} \left[O U_{\theta} \rho U_{\theta}^{\dagger} \right] \quad (7.9)$$

where ρ is an initial quantum state, generally set to the ground state $\rho = |\mathbf{0}\rangle\langle\mathbf{0}|$. This is achieved by means of an iterative hybrid quantum-classical approach where the quantum computer is used to estimate the cost function (7.9), and given such value, the classical computer proposes new variational parameters according to an optimization method, the most common one being gradient descent.

There is freedom in the choice of the gate sequence defining the parameterized unitary U_{θ} , and a choice of its structure is referred to as variational *ansatz*, of which we gave an extended overview in the dedicated Section 2.2.2. For example, the unitary could be composed of a layer of Pauli rotations around the X -axis on each qubit $R_X(\theta) = \exp(-i\theta X/2)$, followed by a layer of CNOTs acting on pairs of neighbouring qubits. This is in fact the general blueprint of variational quantum circuits, as they are generally created by repeating single-qubits parameterized rotations followed by multi-qubits operations which introduce entanglement into the computation. Examples of parameterized quantum circuits are shown in Fig. 7.1.

Quantum Neural Networks As it is often the case with learning tasks, either classical or quantum, the goal is to solve a problem given access to a dataset of inputs $\{\mathbf{x}_i\}_i$, $\mathbf{x}_i \in \mathcal{X}$, representative of the task to be solved. As outlined in Definition 3.2, whenever data is involved, variational quantum circuits are often referred to as quantum neural networks. In this case, the quantum circuit of the “neural network” depends on two sets of parameters \mathbf{x} and θ , the former being the input data to be analyzed, and the latter the variational parameters to be adjusted (i.e. the *weights* of the neural network). In the quantum machine learning jargon, the encoding scheme used to load the input data onto the quantum computer is known as *feature map*, and consists of a unitary operation parameterized by \mathbf{x} . We will denote such feature encoding gate with $\mathcal{F}(\mathbf{x})$, where $\mathbf{x} \in \mathcal{X}$. As with the variational unitary, there is no standard choice for a feature map, and one has to pick a specific *ansatz*, ideally biasing the choice towards architectures built using knowledge of the problem to be solved [208, 284]. Summing up, a general QNN can be then expressed as

$$U_{\text{QNN}}(\mathbf{x}; \theta) = \prod_{i=L}^1 V(\theta_i) \mathcal{F}(\mathbf{x}) = V(\theta_L) \mathcal{F}(\mathbf{x}) \cdots V(\theta_1) \mathcal{F}(\mathbf{x}), \quad (7.10)$$

where $\mathcal{F}(\mathbf{x})$ is the feature map *ansatz* depending on the input data \mathbf{x} ; $V(\theta_i)$ is a variational *ansatz* depending on trainable parameters $\theta_i \in \theta = (\theta_1, \dots, \theta_L)$ with $\theta \in \mathbb{R}^p$; and L is the number of repetitions (or *layers*) of the such layered structure.

As already discussed in Sec. 3.3.2, it was recently shown that uploading the input data multiple times throughout the circuit is essential for quantum neural networks to model higher-order functions of the inputs [109, 269], via a procedure now dubbed *data-reuploading* [229, 302]. Indeed, note that

the input data in the feature map in Eq. (7.10) is the same in every layer, while the variational blocks V use a different parameter vector in every layer. In Fig. 7.1a we give a graphical representation of the general structure of QNNs. As for the explicit implementation of \mathcal{F} and V , there is no fixed choice and these are usually composed of single qubit rotations followed by entangling operations, either fixed (e.g. CNOTs) or themselves parameterized (e.g. controlled rotations). See Fig. 7.1c for some prototypical examples of parameterized blocks proposed in the literature [4, 281], which we will consider throughout the presented analysis.

7.2.5 Randomness, Entanglement and Trainability

As we now well know from the discussion in Sec. 2.2.4.2, one of the hardest theoretical challenges affecting quantum machine learning models is the emergence of *barren plateaus* (BP). Different sources can lead to the unfolding of barren plateaus, and these can be summarised as follows³: randomness-induced BP [137, 202, 221, 258], BP induced by *global* cost functions defined with observables having support on a large number of qubits [57], and eventually noise-induced BP [312].

The analysis presented in this chapter concerns the former type of barren plateaus, that roughly occur when parameterized quantum circuits, when initialised with random parameters, resemble general random unitaries. Indeed, despite being quite limited in terms of qubits connectivity and gate operations, common instances of parameterized quantum circuits are often found to behave as unitary 2-designs [81], in which case, as proved in Th. 2.2, the variance of the gradients of any cost function $f(\boldsymbol{\theta})$ defined on the circuit will vanish exponentially with the number of qubits n , namely [137]

$$\text{Var}_{\boldsymbol{\theta}}[\partial_k f(\boldsymbol{\theta})] \in \mathcal{O}(c^{-n}) \quad c > 1, \quad (7.11)$$

where $f(\boldsymbol{\theta})$ is as in Eq. (7.9). Specifically, the cost function concentrates around its mean value and stays constant almost everywhere in parameter space [13], which makes training unfeasible.

Vanishing gradients are used as a *witness* to assess whether a parameterized quantum circuit resembles a unitary 2-designs. Of course, this is only *necessary* but not *sufficient* condition, as one can easily devise a circuit that is not a 2-design but has vanishing gradients, for example using a global cost with a shallow circuit [57], as the one presented in Appendix A.1.

In addition to vanishing gradients, another witness of randomness is the entanglement generated inside the circuit [221]. Indeed, as discussed previously in Sec. 7.2.3, random quantum states are almost maximally entangled, so one can use the maximality of entanglement generated by a parameterized circuit as an indicator of the resemblance to a random unitary evolution. As for vanishing gradients, the presence of large entanglement is however only a necessary but not sufficient condition for randomness, as a simple shallow circuit composed of Hadamards and CNOTs can create maximally entangled states (GHZ states), which are clearly not random. As discussed in [258], the so-called entanglement-induced BPs [221, 227] provide an alternative yet equivalent description of local cost barren plateaus (circuits with global costs always suffer of vanishing gradients [57], regardless of randomness), as they both stem from the proximity of parameterized quantum circuits to unitary 2-designs.

Indeed, if a circuit is a unitary 2-design, then the average entanglement entropy of any subsystem A of dimension d_A ($d_A \leq d_B$) will be already very close to its maximal value [184, 258]

$$\log d_A - 1 \leq \mathbb{E}_{\boldsymbol{\theta}}[S(\rho_A)] \leq \log d_A, \quad (7.12)$$

and approaches the Page value (7.8) for truly Haar-random states. For completeness, we provide a proof of Eq. (7.12) based on the Rényi 2-entropy in Appendix D.1.

³In Chapter 2 we anticipated that the four sources of BP shown in Fig.2.4 actually can be reduced to three, since, as clearly explained in this section, expressibility-Induced BP and entanglement-induced BP both derive from an abundance of randomness inside the quantum circuit. For this reason, we hereby refer to these types of BPs as randomness-induced BP.

To summarise, while the presence of entanglement is a necessary ingredient to avoid classical simulability, its uncontrolled growth is likely to signal the emergence of barren plateaus. The evaluation of the entangling capabilities of parameterized quantum circuits is then a valuable diagnostic tool to provide information both on the classical simulability and trainability issues of quantum machine learning models. At last, we remark that although various methods have been put forward to mitigate the occurrence of BPs [115, 283, 309], including proposals based on entanglement control [164, 227, 258], these remain a bottleneck for scaling up quantum machine learning computations based on variational circuits.

7.3 Results

We now proceed to analyse the entanglement production in various quantum neural network architectures with different feature maps and variational ansatz, obtained composing the circuit blocks shown in Fig. 7.1. In particular, we take as a prototypical example the QNN introduced in [4], argued as a good candidate for quantum machine learning applications in terms of capacity and expressibility, possibly achieving an advantage over classical counterparts.

Such QNN model uses as feature-map $\mathcal{F}(\mathbf{x})$ the so-called ZZFEATUREMAP firstly introduced in [131] as a classically-hard map to load classical data on a quantum state in a non-linear fashion. The variational block $V(\boldsymbol{\theta})$ is instead composed of single qubit rotations followed by entangling operations. In order to better understand the effect of every single operation in the quantum circuit, we also consider variations of the QNN introduced above, varying both the feature map, the variational form, and the entangling topology. All considered circuit blocks are graphically represented in Fig. 7.1.

Let $U_L(\mathbf{x}, \boldsymbol{\theta})$ be the unitary representing a specific quantum neural network with L layers with input data $\mathbf{x} = (x_1, \dots, x_d) \in \mathbb{R}^d$, and variational parameters $\boldsymbol{\theta} = (\theta_1, \dots, \theta_p) \in \mathbb{R}^p$, see Eq. (7.10). We consider random instances of such QNN by sampling both the inputs and the variational parameters according to the uniform distribution $x_i, \theta_i \sim \text{Unif}[0, \pi]$, hence obtaining a collection of QNNs $U_L = \{U_L(\mathbf{x}_i, \boldsymbol{\theta}_i), i = 1, \dots, M\}$. Then, we study the entanglement entropy properties of each of these instances and average the result over the M trials (unless stated otherwise, we take $M = 100$). Thus, when in the following we refer to the entanglement entropy of a quantum circuit, we are always denoting the average over M realisations of that circuit. In order to evaluate the influence of the depth on the entanglement, we repeat this analysis by increasing the number of layers in the quantum neural network $L = 1, \dots, L_{\max}$.

Note that although the total number of parameters (inputs and parameters) depends on the specific feature map and variational form used, for the considered circuits such difference generally amounts to a constant and does not have a relevant impact on the results. In Tab. 7.1 we report the number of parameters in each circuit template analyzed in this work. We anticipate that while the number of parameters in the considered quantum circuits only scales polynomially with the system size n , these are found to be sufficient to reproduce some entanglement features of random unitaries, which are instead characterized by an exponential number parameters. This is in agreement with results on random quantum circuits that states that polynomial resources are sufficient to approximate unitary designs [123, 126]. We refer to Sec. 7.4 for an extended discussion.

7.3.1 Alternating vs. Sequential data reuploading

As a first analysis, we study the difference in entanglement growth between a standard QNN using an *alternated* repetitions of feature maps and variational forms (as in Fig. 7.1), and one in which we have first L repetitions of the feature map followed by L repetitions of the variational form, which we refer to as a *sequential* structure. The former leverages an alternated evolution of the quantum state which is typical of quantum neural networks using a data reuploading scheme [109, 229, 269].

	Abbr.	Number of parameters		
		LINEAR	CIRCULAR	FULL
CIRCUIT 1	C_1	$2n$	$2n$	$2n$
CIRCUIT 2	C_2	n	n	n
CIRCUIT 3	C_3	$2n - 1$	$2n$	$\frac{n^2+n}{2}$
CIRCUIT ZZFEATUREMAP	C_{ZZ}	n	n	n

Table 7.1: Number of parameters for each considered circuit template and their relative entanglement topology. Notice that, while the number of parameters remains constant for C_{ZZ} as shown in the table, the number of parametric gates varies analogously to C_3 .

The latter instead uses an initial data-dependent evolution followed by a trainable unitary, thereby creating an architecture similar to quantum kernel machines [263].

While the two structures (alternated and sequential) may be mapped to each other using ancillary qubits [152], they can have rather different performances, and we hereby show how they also create entanglement in a different way. Specifically, given the two unitary evolutions, namely the fixed input-dependent feature map $\mathcal{F}(\mathbf{x})$ and the varying parameterised variational form $V(\boldsymbol{\theta}_i)$, one expects the alternated dynamics

$$U_{\text{alt}} = V(\boldsymbol{\theta}_L)\mathcal{F}(\mathbf{x}) \dots V(\boldsymbol{\theta}_1)\mathcal{F}(\mathbf{x}),$$

to introduce randomness at a faster rate than the sequential process

$$U_{\text{seq}} = \prod_{i=L}^1 V(\boldsymbol{\theta}_i) \prod_{i=L}^1 \mathcal{F}(\mathbf{x}),$$

and hence introduce more entanglement in the system. Such intuition is confirmed by the numerical results, and may be understood as a consequence of the universality of the alternating dynamics proved for example for QAOA circuits [187, 214].

Here we use $\mathcal{F} = C_{ZZ}$ and $V = C_2$, as defined in Fig. 7.1, both with linear topology. Be S^{alt} and S^{seq} the entanglement entropy of the bipartition with an equal number of qubits, which is generally the highest, for the alternated and sequential structure, respectively. We define the normalized difference as

$$\Delta\bar{S} = \frac{S^{\text{alt}} - S^{\text{seq}}}{(S^{\text{alt}} + S^{\text{seq}})/2}, \quad (7.13)$$

and study its behaviour as the depth of the quantum circuit is increased, as shown in Fig. 7.2.

The metric is always positive and features a maximum, implying that the alternated structure is creating entanglement *faster* (i.e. with fewer layers) than its non-alternated counterpart. Note that for $L = 1$ layers the two structures are identical, so the generated entanglement is the same up to the statistical error, which explains why all the curves start around zero. At a high number of repetitions, the two structures tend to the same value, showing a $\Delta\bar{S} \simeq 0$, which can be understood in light of the results presented in the following sections: as the number of layers of a QNN is increased, the entanglement rapidly converges to that of a Haar-distributed random state, thus the alternated and non-alternated structure eventually converge to the same value. Given the higher entanglement production rate of the alternated structure, in the following analysis, we shall focus on this structure only.

7.3.2 Entanglement distribution across bonds

It is natural to ask how the choice of the feature map, the variational form, and the entangling topology impact the growth of entanglement of the quantum state. In this section, we start to explore

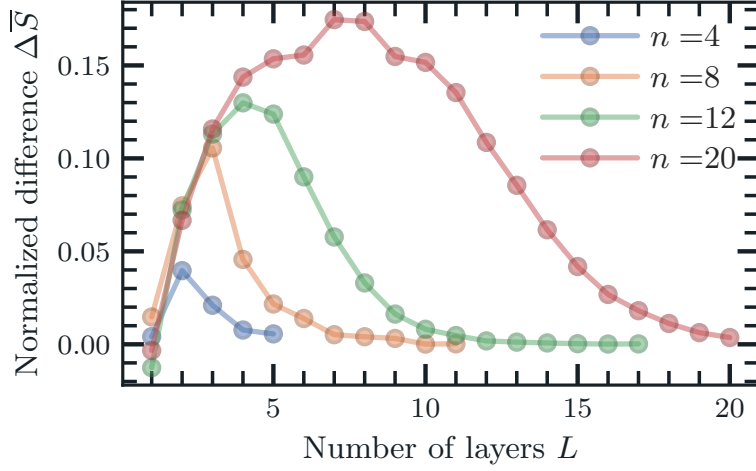


Figure 7.2: Normalised entanglement difference $\Delta\bar{S}$, as defined in Eq. (7.13) for different numbers of qubits. The used QNN is defined by $\mathcal{F} = C_{ZZ}$ and $V = C_2$. All the data points are obtained by averaging over 10^3 realisations.

this question by studying how entanglement is distributed across all possible ordered bi-partitions of the n qubits in the network. That is, given an MPS representation as in Fig. 7.1(b), we study the entanglement entropy corresponding to each bond in the linear chain. Denoting with e_i the bond connecting qubit q_i and q_{i+1} , the entanglement entropy of that bond is (see Eq. (7.2))

$$\begin{aligned} S(e_i) &= -\text{Tr}[\rho_{[1:i]} \log \rho_{[1:i]}] \\ \rho_{[1:i]} &= \text{Tr}_{i+1, \dots, n}[\rho] \end{aligned} \quad (7.14)$$

where $\rho_{[1:i]}$ is the reduced density matrix of all the qubits up to the i -th one, and ρ is the state obtained from the quantum neural network $\rho = U_L(\mathbf{x}, \boldsymbol{\theta}) |\mathbf{0}\rangle\langle\mathbf{0}| U_L(\mathbf{x}, \boldsymbol{\theta})^\dagger$.

In Fig. 7.3 we show the entanglement entropy distribution for the case of $n = 8$ qubits using three different quantum neural networks architectures: in panel (a) the one proposed in [4] with feature map $\mathcal{F} = C_{ZZ}$, variational ansatz $V = C_2$, both with linear entanglement; in (b) same as before but using a circular entanglement topology; and eventually in panel (c) a simpler circuit using a tensor product feature map $\mathcal{F} = C_1$ which encodes data independently on each qubit, followed by the same variational ansatz $V = C_2$, again with linear entanglement both. For reference, it is also shown the expectation value of the entanglement entropy for Haar-random quantum states evaluated with Eq. (7.8), as well as an upper bound given by the highest possible entanglement $\log(\min(d_A, d_B))$, obtained if the two partitions A and B were maximally entangled. Note that while we report only the simulation data for $n = 8$, the discussion has general validity as identical results hold for all tested numbers of qubits, $n = 2, \dots, 20$.

First of all, the findings agree with the intuition that deeper circuits are able to create higher entangled states with respect to shallower ones, in accordance with results from [281]. In particular, the entanglement entropy is higher at the centre of the chain. Clearly, depending on the specifics of the QNN, the entanglement grows faster in certain architectures with respect to others. Regarding the effect of the entangling topology, comparing panels (a) and (b) we see that circular connections produce greater entanglement compared to the nearest-neighbours interaction and that such entanglement grows at a faster rate as the number of layers is increased. As for the choice of the feature map, since the QNN in panel (c) produces entanglement only through the entangling gate in the variational blocks, its entanglement is lower and also grows slower with respect to the QNN in panel (a), even though it has twice the number of parameters in the feature map.

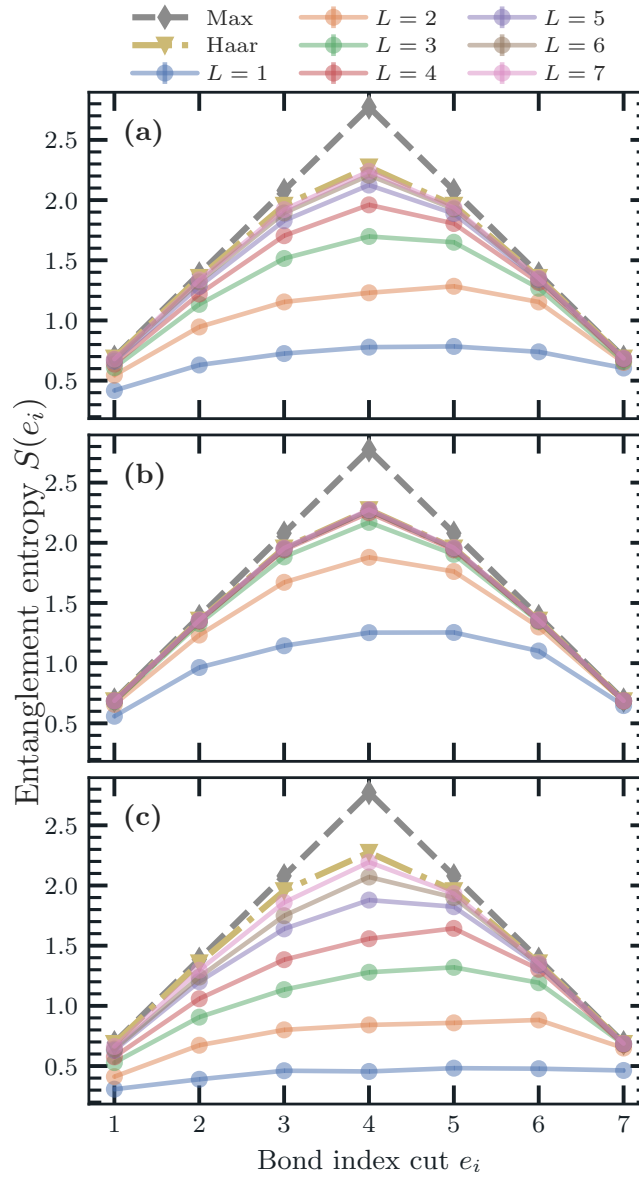


Figure 7.3: Average entanglement entropy across bonds for a system composed of $n = 8$ qubits, where e_i is the bond connecting qubit i and $i + 1$, as in Fig. 7.1. The curves represent different numbers of layers L in the quantum neural network. **(a)** QNN with structure: $\mathcal{F} = C_{ZZ}$, $V = C_2$, both with linear entanglement. **(b)** QNN with structure: same as in (a) but with circular entanglement. **(c)** QNN with structure: $\mathcal{F} = C_1$ which has no entangling gates, and $V = C_2$ with linear entanglement.

Interestingly, however, as the number of layers approaches the number of qubits $L \approx n$, all investigated QNNs converge to the same values, that is those obtained for random states sampled from the uniform Haar distribution. Deep enough QNNs are then flexible enough to reproduce the same entanglement spectrum of a random state, which, as discussed in section 7.2.3, are very highly entangled. Again, even though the measure of entanglement is different, this is in agreement with the results presented in [281], where the convergence to the Haar distribution is encountered for various parameterized quantum circuits, and also with other results in the literature regarding the properties of random quantum circuits to approximate the Haar distribution [37, 124]. We will discuss this more in detail in Sec. 7.4, while a more in-depth analysis of the convergence is the subject of the next section.

7.3.3 Entanglement scaling with increasing depth

In order to better understand the entanglement scaling properties of QNNs, we introduce a new quantity, defined as the total entanglement entropy S_{tot} created in the MPS chain

$$S_{\text{tot}} = \sum_{i=1}^{n-1} S(e_i), \quad (7.15)$$

which is the sum of the entanglement entropy of all the ordered bipartitions of the quantum state. We use this global measure to quantify how fast QNNs approach the Haar distribution in terms of overall entanglement production. In particular, we define a new figure of merit, the entangling layers \tilde{L} , defined as the number of layers needed by an architecture to reach 90% of the total entanglement of a Haar distributed state $S_{\text{tot}}^{\text{Haar}}$, namely

$$\tilde{L} = \text{min number of of layers} \quad \text{such that} \quad S_{\text{tot}} \geq 0.9 S_{\text{tot}}^{\text{Haar}}. \quad (7.16)$$

The choice of the 90% threshold allows to select states that are already very close to the Haar-random value, and avoids undesired oscillating behaviours obtained when higher thresholds are used, e.g. 99%, which are caused by statistical fluctuations (recall that every QNN is sampled multiple times with different parameters to calculate averages).

In Fig. 7.4 we show the behaviour of \tilde{L} for four different QNNs as the number of qubits is increased. Note that each QNN is considered with all the three possible entangling topologies (*linear*, *circular* and *full*, as defined in Fig. 7.1). At last, note that all QNNs leverage the same variational form $V = C_2$, while the feature map is changed, as reported in the legend.

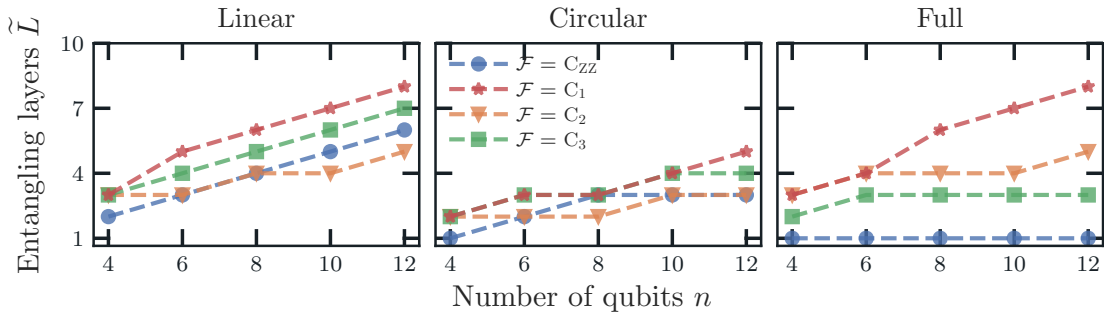


Figure 7.4: Entangling layers \tilde{L} , i.e number of layers to reach 90% of the Haar entanglement, versus the number of qubits. The analysis is carried out over four different QNN architectures, each evaluated with different entangling topologies (*linear*, *circular*, and *full*). The architectures leverage the same variational form V , while the feature map \mathcal{F} is changed, as reported in the legend.

First, we observe that the entangling layers display a linear behaviour when a linear entanglement topology is used. This means that the number of layers needed to entangle the system scales linearly with the size of the system. The behaviour changes abruptly when we move to a circular or full entangling topology. All architectures display a faster entanglement production when passing from a linear to a circular topology, as can be seen from the lower slope of the curves. The all-to-all connectivity speeds up entanglement production only for $\mathcal{F} = C_{ZZ}$, C_3 , while the circuits $\mathcal{F} = C_2$, C_1 show essentially the same behaviour of the linear case. We now proceed to discuss more in detail such results.

We start comparing the entangling capabilities of C_{ZZ} vs. C_2 . Both with linear and circular entangling topology, C_2 is able to produce entanglement essentially at the same rate as C_{ZZ} , despite C_2 being of a much simpler structure, with half the number of two-qubit gates. However, things change dramatically using a full entangling map, as the QNN reaches the 90% threshold already

at $\tilde{L} = 1$, while C_2 needs more layers, showing the same dependence of a linear connectivity. While counter-intuitive at first, it is easy to see that the entanglement generated by C_2 with a *full* architecture is indeed equivalent to the *linear* one. This is due to a simple circuit identity regarding networks of CNOTs reported in Fig. 7.5. Such circuitual identity holds for any number of qubits, which makes the full entangling map as shown in Fig. 7.1 just as a linear entangling map in disguise (in particular, it is the inverse of the linear entangling map). See Appendix D.3 for a more precise statement, discussion and proof. Such circuitual identity thus explains the equivalence of the yellow ($\mathcal{F} = C_2$) and red ($\mathcal{F} = C_1$) curves between the first and last plot of Fig. 7.4.

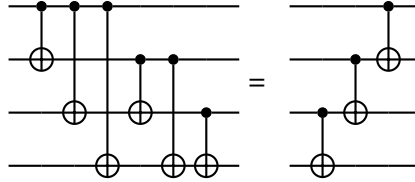


Figure 7.5: Circuitual identity between a full (all-to-all) entangling map made of only CNOTs and the inverse of a linear (nearest-neighbours) entangling map.

Such equivalence clearly does not hold if controlled rotations are used instead of CNOTs. Indeed, the feature map $\mathcal{F} = C_3$ uses controlled rotations with independent random parameters, and given that these gates do not cancel out, the entanglement is always increasing going from low to high connectivity. Note that such increase is mainly due to the feature map, as the variational ansatz $V = C_2$ is the same as other structures, suffering from the CNOTs cancellation issue described above.

For comparison, we also show the performances of a QNN with the tensor product feature encoding $\mathcal{F} = C_1$, using no entangling operations. Interestingly, even if this QNN uses two-qubit interactions only inside the variational blocks, these are sufficient to create entanglement similar to other considered QNNs, even at a slower yet comparable rate.

We report in Appendix D.5 the complete simulation results detailing the evolution of the entanglement with the depth of the circuit, for different numbers of qubits.

7.3.4 Entanglement Speed

So far we have presented numerical evidence for the entanglement production in QNNs up to a maximum of 20 qubits. In the following we extend the analysis leveraging MPS to simulate quantum systems of bigger size up to 50 qubits, with a maximum bond dimension of $\chi_{\max} = 4096$. More importantly, we show how the entanglement growth follows a behaviour that is specific to each particular QNN architecture and the number of layers considered, but independent of the number of qubits in the circuit. We can thus uniquely assign an *entanglement speed* value to each QNN, which, we stress again, only depends on the choice of the ansatz, and holds identically for any instantiation of that QNN with arbitrary number of qubits.

Taking into account the entanglement growth discussed in Sec. 7.3.3, we restrict the analysis to a linear architecture, to increase as much as possible the number of layers we can correctly simulate with tensor networks techniques. Indeed, the entanglement production with a circular or full topology is too fast to allow for a convergent simulation with MPS for deep circuits.

Furthermore, we introduce the maximum Haar entanglement entropy, defined as the maximum across all bond entropies for a given number of qubits, as

$$S_{n, \max}^{\text{Haar}} = \max_A (\mathbb{E}[S(\rho_A)]) \approx \frac{n}{2} \log 2 - \frac{1}{2} \quad \text{for } n_A = \frac{n}{2} \gg 1, \quad (7.17)$$

where the approximation in the second line has an error that scales as $\mathcal{O}(2^{-n/2})$, see Appendix D.2 for its derivation. Thus, for $n \geq 30$ qubits, when the exact computation of the Haar entanglement

entropy is unfeasible, we employ the approximated Eq. (7.17). Finally, we define the normalised entanglement entropy \tilde{S}_n as

$$\tilde{S}_n = \frac{\max_{e_i} [S(e_i)]}{S_{n,\max}^{\text{Haar}}}. \quad (7.18)$$

We stress that \tilde{S}_n is normalised to the maximum Haar entanglement for a fixed n , not to the real maximum of the entanglement, which would be $S = \frac{n}{2} \log 2$ for the equal size bipartition.

In Fig. 7.6 we show the evolution of \tilde{S}_n versus the normalised number of layers L/n for $n \in \{8, 12, 16, 20, 30, 50\}$ qubits, for the QNN defined with $\mathcal{F} = C_{zz}$, $V = C_2$ with linear connectivity. We note that all the points, independently of the system size n , follow the same curve: an initial linear growth of the entanglement is followed by a saturation to the Haar-random value for the entanglement entropy (7.8). In particular, we check this behaviour also at large system sizes with $n = 30, 50$ qubits and circuits with up to $L = 11$ layers, and confirm that such scaling is indeed size independent. See Appendix D.6 for a discussion on the errors introduced by truncation in the MPS representation for simulations with $n = 30, 50$ qubits.

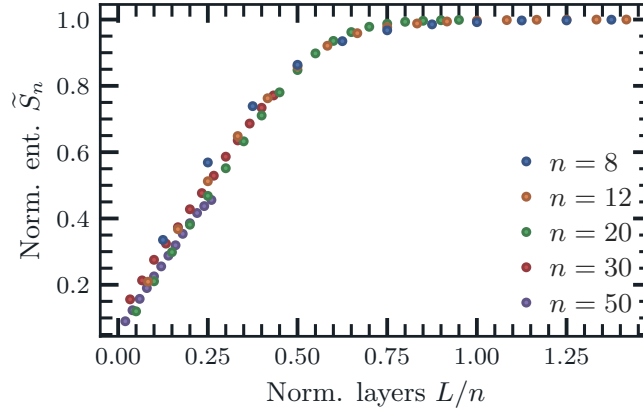


Figure 7.6: Normalised entanglement \tilde{S}_n (7.18) versus the normalised number of layers L/n , for different number of qubits n , and for the QNN defined by $\mathcal{F} = C_{zz}$, $V = C_2$ with linear connectivity. All the normalised entanglement points follow the same curve, independently of the system size n . The points for $n = 8, 12, 16, 20$ are obtained by averaging over 10^3 samples, while for $n = 30, 50$ the averages are over 10 samples.

Feature map \mathcal{F}	Variational ansatz V	Entangling speed v_s
C_{zz}	C_2	(1.8 ± 0.1)
C_{zz}	C_3	(0.59 ± 0.02)
C_1	C_3	(0.316 ± 0.006)

Table 7.2: Entangling speed, i.e. a measure of how fast the entanglement is created, for different QNN architectures. These results are obtained using up to 16 qubits.

Inspired by the behaviour of \tilde{S}_n , we introduce a measure for the entanglement production which is specific to a given QNN architecture (feature map plus variational ansatz) and independent of the number of qubits. Borrowing from the literature on random quantum circuits, it is known that the entanglement of a system undergoing random evolution initially grows linearly in time (depth of the circuit) before reaching the plateau of Haar random states [48, 164, 216, 332]. Indeed, as clear

from Fig. 7.6, we observe the same initial linear growth, and thus we define the *entangling speed* v_s as

$$\tilde{S}_n \propto v_s \cdot \left(\frac{L}{n}\right) \quad \text{for } \tilde{S}_n \leq 0.5, \quad (7.19)$$

where 0.5 is a threshold such that the linear behaviour holds.

The entangling speed can thus be obtained by fitting the curve in Fig. 7.6 with the linear model of Eq. (7.19) in the appropriate range. We report in Tab. 7.2 the entangling speed for a subset of the inspected architectures, and notice that entanglement is produced at sensibly different rates. In agreement with the findings of Sec. 7.3.3, we see that for a linear topology the circuit C_2 builds the entanglement at the fastest rate. Indeed, fixing the feature map to $\mathcal{F} = C_{ZZ}$, C_2 produces entanglement three times faster than C_3 .

To further characterise the applicability of the entangling speed, we shown that the behaviour of Fig. 7.6 evaluated for random circuits also holds when the input data $\mathbf{x} \in \mathbb{R}^n$ in the feature map $\mathcal{F}(\mathbf{x})$ are not drawn from the uniform distribution, but rather from real world datasets. In particular, we select two common dataset in the machine learning literature, the wine [324] and breast cancer [334] datasets, and calculate the entanglement generated in the circuit when these data are fed into the feature maps (variational blocks are still populated with random parameters as before). The results presented in Fig. 7.7 are obtained by rescaling all the features of the datasets in the interval $[0, \pi]$, and then averaging over all data points in the datasets. The wine dataset ($d = 13$ features, hence $d = n = 13$ qubits) follows perfectly the theoretical curve, and the breast cancer ($d = 9$ features, $n = 9$ qubits) only slightly deviates from it, producing entanglement at a smaller rate. We then conclude that the entangling speed depends primarily on the architecture of the circuit rather than the actual values of the parameters. Clearly this holds for reasonably distributed data features, that is excluding pathological cases of values being either zero or concentrating around it.

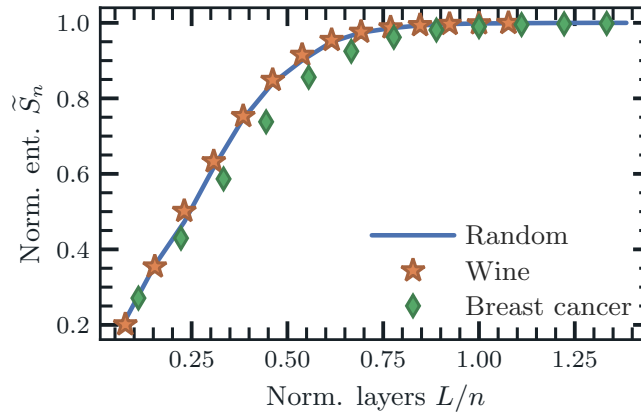


Figure 7.7: Normalised entanglement using inputs from real-world datasets in the feature map, compared to the one obtained using random input data sampled from the uniform distribution $\text{Unif}[0, \pi]$, as shown also in Fig. 7.6.

Thus, the entangling speed can be used as a good estimate of the entanglement generated in a QNN also in real use cases, especially at the start of optimisation, when trainable parameters are usually initialised at random. For example, one could measure the entangling speed of the architecture of interest on a random quantum circuit of just a few qubits, and then estimate the entanglement generated with the same architecture on an arbitrary number of qubits and circuit layers, especially in regimes where simulations are no longer computationally feasible.

7.3.5 Expressibility

In addition to entanglement, another useful quantity to characterise parametrized quantum circuits is the expressibility, as defined by authors in [281]. Such measure quantifies how well the QNN is able to explore the Hilbert space by comparing the distribution of fidelities of states generated by the QNNs with that of randomly Haar-distributed ones, and we refer to Appendix D.4 for a formal definition and explanation. Thus, in order to have a comprehensive understanding of the factors at play in the behaviour of QNNs, in Fig. 7.8 we show the expressibility measure for the QNNs analysed in Fig. 7.4 with a linear connectivity. As one would expect, the expressibility increases as the number of layers is increased, up until a plateau is reached.

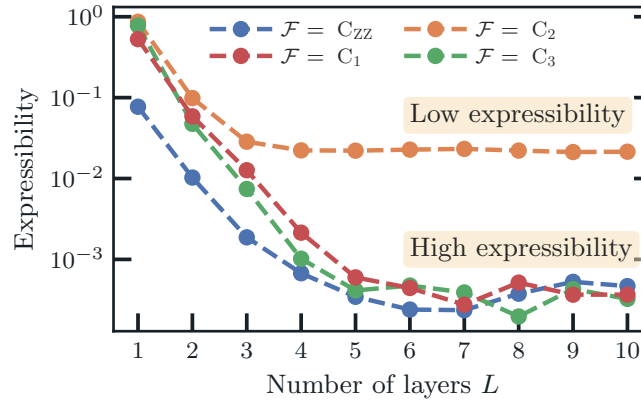


Figure 7.8: Expressibility of the QNNs analysed in Fig. 7.4, for $n = 8$ qubits with linear entanglement. The expressibility measures how well a variational circuit is able to address the unitary space (the lower, the better). All QNNs use the same variational form $V = C_2$, but with different feature maps. As the number of layers is increased, QNNs become more expressible, eventually reaching a plateau.

Interestingly, the structure with $\mathcal{F} = V = C_2$ turns out to be the least expressible of all the structures considered, even if it is the one producing entanglement at the fastest rate, in agreement with the results reported in [281], as such QNN is indeed very similar to the parameterized circuit labeled ‘15’ in [281]. On the contrary, the QNN with $\mathcal{F} = C_{zz}$, and $V = C_2$ proposed in [4] is able to reach high expressibility while producing entanglement at a controlled pace. As the presence of high entanglement is correlated with trainability issues [221], this QNN attains an optimal balance of mild entanglement with high expressibility even at low depth, which could be related to its good performances in quantum machine learning task [4, 131]. However, a similar, yet less favourable balance, is achieved by the other two architectures, so further investigation is needed to discriminate where the optimality comes from.

In this respect, the authors in [144] found the expressibility to be correlated with the classification accuracy of QNNs in supervised learning tasks, while weak correlation was found with the entanglement generated inside the circuit, in line with the observations regarding entanglement-induced barren plateaus [221]. As discussed earlier in Sec. 7.2.5, both expressibility and high entanglement are related to the resemblance of the circuit to a random unitary, but while the former provides a more direct evidence, the latter gives an indirect indication. Indeed, there are cases of circuits having low expressibility but high entanglement, indicating that such circuits selectively explore only some highly-entangled regions of the Hilbert space [281].

7.3.6 Distribution of the singular values

The randomness of a quantum state can also be probed using tools from random matrix theory. Specifically, this can be done by studying the distribution of the eigenvalues of the reduced density

matrices, which are known to follow the Marčenko-Pastur (MP) law when pure random quantum states are considered [150, 333]. More in detail, let $|\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B$ be an Haar-random bipartite quantum state with Schmidt decomposition $|\psi\rangle = \sum_{i=1}^d \lambda_i |\xi_i\rangle_A \otimes |\eta_i\rangle_B$, where $d = \min(d_A, d_B)$ and $d_{A,B}$ is the dimension of the Hilbert space $\mathcal{H}_{A,B}$. The reduced density matrix $\rho_A = \text{Tr}_B [|\psi\rangle\langle\psi|]$ has eigenvalues λ_i^2 given by the square of the singular values, and for large system size their distribution is described by the MP distribution [197, 239].

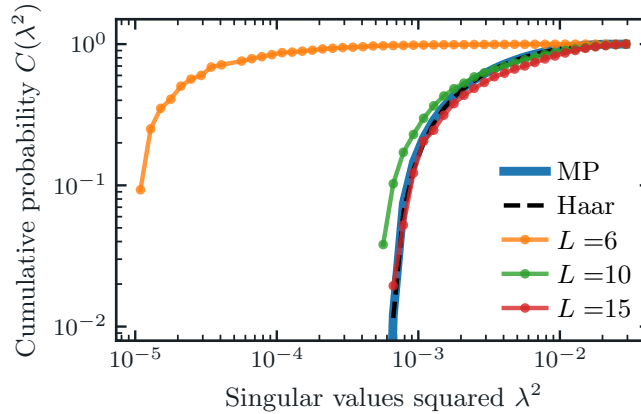


Figure 7.9: Convergence to the Marčenko-Pastur (MP) distribution of the eigenvalues. The cumulative distribution $C(\lambda)$ of the eigenvalues λ^2 corresponding to the central bond of a quantum state of $n = 15$ qubits, generated with $\mathcal{F} = C_{ZZ}$, $V = C_2$, and linear connectivity. We show the behaviour for different numbers of layers L , and for truly Haar-random states which, as expected, exactly follows the MP curve.

In Figure 7.9 we show the cumulative probability distribution of the eigenvalues $C(\lambda^2)$ of the reduced state of the first half of the qubits, obtained with a QNN with $n = 15$ qubits, feature map $\mathcal{F} = C_{ZZ}$, variational ansatz $V = C_2$, and linear connectivity. The distribution of the singular values for the QNN is obtained by running the circuit 10^2 times with different sets of parameters, and storing the singular values corresponding to the central cut. Then, we construct the cumulative distribution from the histogram of all the singular values obtained from the simulations. As the number of layers L is increased, the distribution of the eigenvalues approaches the theoretical MP distribution, eventually matching it when the number of layers is equal to the number of qubits. This behaviour is displayed also by other QNN architectures. For completeness, we also show the distribution of the eigenvalues of a truly Haar-random quantum state, generated by sampling its entries independently from a normal complex distribution and then normalising it [333], which, as expected, follows perfectly the MP curve.

7.4 Discussion

Moments of the Haar distribution can be approximated efficiently using local random quantum circuits of sufficient depth. Depending on the connectivity dimension D of the qubits, defined as the number of other qubits that are connected to each qubit, order $\mathcal{O}(\text{poly}(t) \cdot n^{1/D})$ -depth random circuits are sufficient to create approximate unitary t -designs [37, 123, 124, 126], that is circuits that generate a distribution of unitaries which approximately matches moments of the Haar distribution up to order t [81]. Numerical studies suggest that these results also hold for random parameterized quantum circuits of various forms [57, 137, 202, 281].

We extend these results by showing similar results also for quantum neural networks featuring data re-uploading, both for random instances using random inputs and parameters, and also for real-world dataset when these are used as inputs in the feature map. In particular, for a linear

connectivity, as the number of layers approaches to the number of qubits $L \approx n$, QNNs display the same entanglement entropy properties of Haar-distributed random states, a fact which can be taken as a proxy for QNNs approximating unitary designs. Such behaviour was also confirmed by studying the randomness of the circuits with other metrics, namely the expressibility and the convergence to the Marčenko-Pastur distribution of the eigenvalues of the reduced states. In both cases, we find strong evidence of the QNN reproducing the same features of random quantum states as the number of layers approaches the system size using a linear connectivity.

Our analysis also underlines the importance of the entangling operations, as careless use of an all-to-all connection can result in unwanted simplifications, making the effective connectivity identical to a nearest-neighbours one. Parameterised two-qubit interactions can solve the problem, even though they may be challenging to implement on real hardware. A good trade-off is achieved with a circular entangling topology, which is immune to simplifications and shows remarkable entangling capabilities. Indeed, from the results of Fig. 7.4, we see that such connectivity is able to create high multipartite entanglement between qubits already at shallow depth, and with only minor additional hardware resources compared to the linear connectivity. An all-to-all topology instead reaches typical values for entanglement of random states essentially at constant depth $L \in \mathcal{O}(1)$, independently of the system size, and the architecture used (when non-trivial feature maps and variational ansätze are used).

While limiting the entanglement inside a quantum neural network may be necessary to ensure its trainability [221], low entanglement makes the circuit prone to be simulated exactly with an MPS, as discussed in Sec. 7.3.4. Thus, we envision that a sweet spot should be found in order for QNNs to show signs of quantum advantage: not too high to preclude trainability, not too low to escape triviality.

At last, the introduction of the entangling speed v_s (7.19) can be used as a figure of merit for the entanglement production of a given QNN, independent of the size of the system. Indeed, the entangling speed can be studied and assigned to an architecture in the simulable regime (low number of qubits n), and then used to estimate the number of layers to achieve a well-determined quantity of the entanglement, for any system size. We also stress that v_s characterises the most interesting interval of layers in a circuit. As discussed earlier, a value of the entanglement too high might be connected to barren plateaus, underlying the importance of exploring the regime where the entanglement has not saturated yet, and the linear regime still holds.

We now briefly comment on future interesting investigation directions regarding entanglement and QNNs. The focus of this study was to carefully study the entanglement features of common quantum ansätze, specifically when they are initialised with random parameters and no optimisation has yet started. A natural followup is to ask whether entanglement plays any role also during the optimisation process, which is at core of variational quantum algorithms. While for some specific variational procedures like QAOA [90] or VQE-based ground state solvers [326] one has some knowledge of the structure of the target solution, and hence can infer the behaviour of the entanglement created in the circuit, this is not the case for quantum machine learning tasks, as they are usually very task-dependent. Indeed, current proposals for QML advocate for the use of constrained quantum ansätze specifically tailored to the problem under investigation [208, 246, 284], and then one expects the depth of the circuit and the entanglement generated inside it to highly depend on the specific task to be solved, and dataset to fit, either classical or quantum [275]. Moreover, while the use of deep QNN ansätze (with arguably more entanglement) could offer some optimisation advantages due to overparametrization [9, 162, 172], the emergence of barren plateaus suggests using shallow circuits instead [57, 312]. The characterisation of the role played by entanglement in QNNs, and how it may be leveraged to achieve a quantum advantage over classical methods definitely deserved further explorations. For sake of exploration and clarity, in Appendix D.7 we show some preliminary results on the study of the evolution of entanglement during training.

7.5 Conclusion

In this Chapter we discussed in detail the entanglement generated by different promising Quantum Neural Networks (QNNs) when these are initialised with random parameters, and showed that they reproduce the same properties of random quantum states under various measures.

We employed a Matrix Product States (MPS) simulation of the quantum circuits, which guarantees an easy computation of the entanglement in the circuits, and let us study systems of large system size composed of up to $n = 50$ qubits.

We showed that while all the architectures tend to a Haar entanglement distribution for a sufficiently high number of layers, the speed of convergence strongly depends on the specific circuit ansatz. This result highlights the universal behaviour of the normalised entanglement production (7.18) for a given architecture, so we introduced a new measure to characterise a QNN in terms of its entanglement production, namely the entangling speed (7.19).

Finally, we argued that a trade-off between expressibility and entanglement is the key to a better understanding of QNN performances and an auspicious target for the search of quantum advantage. While high entanglement is a necessary condition to avoid classical simulability, a too-large entanglement is detrimental to the training procedure due to its tight connection with barren plateaus, as discussed in Sec. 7.2.2. A promising future direction is to extend the entanglement analysis of QNNs not only at initialisation but also during the training procedure [164, 258, 326]. These tests would help to understand if QNNs really are a suitable platform for proving quantum advantage.

Whilst entanglement is a necessary feature to bestow the computational power offered by quantum mechanics, on the other end of the spectrum is quantum noise, undoubtedly one of —if not the most— compelling challenges to be tackled to ensure the adoption and applicability of quantum computers. More generally, noise arise in many quantum information processing tasks, and it is at the core the next Chapter, where we discuss a technique to remove a wide class of noises when performing arbitrary measurements on qubit systems.



8. Noise deconvolution

8.1	Introduction	142
8.2	Methods	144
8.2.1	Quantum channels	145
8.2.2	Qubit systems and Pauli Transfer Matrix formalism	145
8.2.3	Quantum tomographic reconstruction	146
8.3	Noise Deconvolution	147
8.4	Inversion of common noise maps	149
8.5	Experimental deconvolution	154
8.5.1	Decoherence noise model	154
8.5.2	Arbitrary Pauli channel	157
8.6	Conclusions	157

In this chapter¹ we present a noise deconvolution technique to remove a wide class of noises when performing arbitrary measurements on qubit systems. In particular, we derive the inverse map of the most common single qubit noisy channels, and exploit it at the data processing step to obtain noise-free estimates of observables evaluated on a qubit system subject to known noise. We illustrate a self-consistency check to ensure that the noise characterisation is accurate providing simulation results for the deconvolution of a generic Pauli channel, as well as experimental evidence of the deconvolution of decoherence noise occurring on Rigetti quantum hardware.

8.1 Introduction

Quantum noise is currently the largest limiting factor in the adoption of quantum computation and quantum technology. Their theoretical performances are in fact hindered by the intrinsic fragility of quantum systems, and over the last years many proposal have been put forward to mitigate, ideally correct, the effect of noise and recover reliable results. On the computing side, as fault-tolerant quantum computers remain out of reach at the moment [166, 236, 277, 289], various error mitigation techniques have been proposed to extend the capabilities of current small scale noisy quantum devices [49, 292, 307]. These ranges from correcting the readout noise via inversion of probability

¹The content of this chapter is based on the author's work [194], and all the figures in this chapter are taken from, or are adaptations of, the figures present in such work.

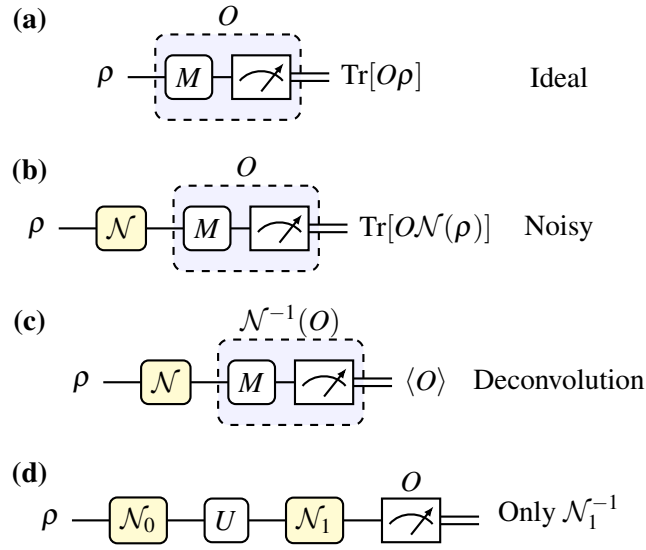


Figure 8.1: General scheme for the noise deconvolution process applied to a qubit. **(a)** Ideal estimation of an observable O on a single qubit in state ρ . The operator $M \in \{\mathbb{I}, H, HS^\dagger\}$, with H and S being the Hadamard and phase gate, used to select a measurement basis in $\{\sigma_z, \sigma_x, \sigma_y\}$ respectively, and thus reconstruct a generic observable O , using Eq. (8.13). **(b)** Noise (indicated with a yellow box) happening before measurement leads to noisy estimates of the expectation values. **(c)** Noise deconvolution approach: measurements of the noise-inverted observables $\mathcal{N}^{-1}(O)$ on the noisy state leads to the mitigated ideal result $\langle O \rangle$. **(d)** The noise deconvolution approach can be used to mitigate the effects of \mathcal{N}_1 only. However, the full noise (\mathcal{N}_0 and \mathcal{N}_1) can be mitigated either if the unitary can be easily inverted as well, or if the noise processes commutes with the interleaving unitary, as is the case for depolarizing noise.

assignment matrix [39], extrapolating the noise in the device to the zero error case [156, 199, 299], using a probabilistic sampling on specific circuits to approximate the noise free computation [97, 199, 299], to also using machine learning approaches to learn how to recover ideal results [190].

While these methods are concerned with mitigating noise occurring in a computation, here we instead focus on the more generic task of correcting the expectation value of arbitrary observables evaluated on a system which is subject to a known noise happening before the measurement stage. Such a scenario is relevant in quantum communication and quantum tomography tasks [278].

Noise in quantum systems is described by means of quantum channels [220]

$$\rho \longrightarrow \mathcal{E}(\rho) = \sum_k A_k \rho A_k^\dagger, \quad (8.1)$$

where ρ is the density matrix representing the state of a quantum system, and A_k are operators acting on the system, which are called Kraus operators. While the effect of unitary dynamics can be reversed using realisable operations, quantum channels cannot be undone, and one can only hope to find operations which only approximately invert the noise process at hand. Examples of this approach leverages for example Petz recovery maps [110, 176, 323], or unitaries which, on average, are able to best reverse the noise based on given distance measures [17, 158, 273].

Here instead we show that noise can be eliminated by means of a *deconvolution* process, provided that the noise map describing the process is known and invertible. In fact, we drop the requirements of the inverse transformation being itself a quantum channel, since the transformation is not applied to the quantum system itself, but to the outcome statistics as a classical post-processing step. We derive the inversion maps of the most common single-qubit noisy channels (both unital and non-unital), and show how to use these to remove the effect of noise from the expectation values

of general observables. In Figure 8.1 we schematically summarise the noise deconvolution idea. The mitigation is effectively obtained by multiplying the noisy estimates by a factor depending on the noise $\langle O \rangle_{\text{mitig}} \sim c \langle O \rangle_{\text{noisy}}$, which comes at the cost of increasing the variance of the estimation, as $\text{Var}[\langle O \rangle_{\text{mitig}}] \sim c^2 \text{Var}[\langle O \rangle_{\text{noisy}}]$, so one needs to gather more statistic to reach a target precision. A related post processing technique specialised for quantum many-body systems and quantum field theory is put forward in ref. [26].

In addition, we provide both numerical simulations of the noise deconvolution process, as well as evidence of deconvolution of decoherence noise occurring on the superconducting quantum computer “Aspen-9” provided by Rigetti, accessed using the Quantum Cloud Services (QCS) [157]. We show how simple self-consistency checks can test whether the known noise map is accurate and how a feedback scheme can be used to adjust the noise parameters.

Our contributions then include: (i) formalisation and discussion of CPTP (namely, *completely positive trace preserving*) noise deconvolution of expectation values through (mathematical) inversion of the noise map; (ii) explicit derivation of the inverse map of the most common single qubit noise channels; (iii) numerical and experimental application of the ideas introduced before.

Before continuing, we briefly describe the relation of the proposed noise deconvolution idea to probabilistic error mitigation (PEM) [97, 299], a quantum error mitigation technique aimed at correcting noisy operations during a quantum computation. Given a characterisation of the noise, PEM works by using the inverse noise map of the operations to build an ensemble of suitably generated quantum circuits. These are sampled according to specific weights, and the results combined to build an approximation of the action of the noise-free quantum circuit. In particular, the mitigation procedure is *active*, in the sense that the experimenter need to generate new quantum circuits and run them against the quantum device. On the contrary, we are instead concerned with the correction of expectation values evaluated on a noisy state, with no computation or dynamics involved. In addition, within our framework, the mitigation is *passive*, in the sense that the mitigation happens classically as a post-processing step, and no action on the quantum system is necessary. Appropriately limiting PEM to the specific case of measurement error mitigation, and realising that sampling on quantum circuits is no longer a necessary step, then one can recover the noise deconvolution procedure presented here, whose regime of application is not restricted to quantum computation, but applies to a general quantum mechanical measurement scenario. As such, some of the results presented here can be recovered also with the techniques proposed in [97, 299]. That said, the explicit calculations presented here for the general noise maps we analyse have not been presented elsewhere in full generality, e.g. see Table 8.1 below.

The chapter is organised as follows. In Sec. 8.2 we recall some basic concepts about quantum channels and the Pauli transfer matrix formalism, and the idea of noise deconvolution in Sec. 8.3. In Sec. 8.4 we leverage the Pauli transfer matrix formalism to explicitly derive the inverse map of the most common single qubit noise channels, and use inside the noise deconvolution procedure to obtain noise-free estimates. In Table 8.1 we summarise all the maps taken in consideration as well as their inverse. In Sec. 8.5 we show by means of simulations that the noise deconvolution process can be used to cancel out the effect of a general Pauli channel, and also provide experimental evidence of the deconvolution of decoherence noise as performed on a real quantum device by Rigetti.

8.2 Methods

In this section we introduce the notation and the theoretical tools used to derive the main results of the work. As in the previous chapters, we denote with \mathcal{H} the Hilbert space of the quantum system under investigation, and with $\mathcal{L}(\mathcal{H})$ the space of squared linear operators acting on \mathcal{H} . Also, in this chapter we use the standard quantum information —rather than computation— notation for Pauli matrices, that is we use $\{\sigma_0, \sigma_x, \sigma_y, \sigma_z\}$ instead of the previous $\{\mathbb{I}, X, Y, Z\}$ defined in Eq. (2.4), which is more convenient and appropriate for the topics treated in this chapter. In addition, we refer

to Appendix E.1 for a brief overview of quantum channels and Kraus decomposition [220].

8.2.1 Quantum channels

In general quantum channels cannot be physically inverted, as there is no quantum evolution capable of reversing their actions. Formally stated, let \mathcal{E} be a CPTP map, it is not possible to find another CPTP map $\mathcal{D} = \mathcal{E}^{-1}$, such that $(\mathcal{D} \circ \mathcal{E})(\rho) = \rho \forall \rho$. The only trivial case when this is possible, is for maps having only a single Kraus operator, in which case they reduce to standard unitary evolution $\mathcal{E}(\rho) = U\rho U^\dagger$, with the inverse given by $\mathcal{D}(\cdot) = U^\dagger(\cdot)U$.

The CPTP conditions impose hard constraints to the operatorial form that physically realisable evolutions must match, namely the Kraus representation. However, the requirement for admitting a more general operator-sum representation are looser. In fact, any Hermiticity preserving map, i.e. a map such that $\Phi(\rho)^\dagger = \Phi(\rho)$ for $\rho = \rho^\dagger$, admits an operator-sum representation as [36, 154]

$$\Phi(\rho) = \sum_k \lambda_k A_k \rho A_k^\dagger \quad \text{with } \lambda_k \in \{\pm 1\}. \quad (8.2)$$

Clearly, if all the coefficients are $\lambda_k = 1 \forall k$, then the map $\Phi(\cdot)$ is also completely positive, since it is in standard Kraus form (8.1). Another useful characterisation is the following.

Corollary 8.1 — Corollary II.2 of (36). Let \mathcal{C}_N be the space of complex $N \times N$ matrices. Suppose $\Phi : \mathcal{C}_N \rightarrow \mathcal{C}_N$ is a completely positive map having the form

$$\Phi(\rho) = \sum_k \beta_k A_k \rho A_k^\dagger, \quad (8.3)$$

where $\{A_k\}_k$ are linearly independent in \mathcal{C}_N , and $\beta_k \in \mathbb{R} \forall k$. Then $\beta_k \geq 0 \forall k$.

Conversely, if a map has the form (8.3) with linearly independent operators $\{A_k\}_k$ but has *some* of the coefficients $\beta_k < 0$, then the map is not completely positive. This result is readily applied to maps acting on qubit systems where $\mathcal{C}_N = \mathbb{C}^{2 \times 2}$. In fact, Pauli matrices σ_x, σ_y and σ_z together with the identity $\sigma_0 = \mathbb{I}_2$, form a linearly independent set in the space of 2×2 complex matrices, and then any map of the form

$$\mathcal{E}(\rho) = \beta_0 \sigma_0 \rho \sigma_0 + \beta_1 \sigma_x \rho \sigma_x + \beta_2 \sigma_y \rho \sigma_y + \beta_3 \sigma_z \rho \sigma_z, \quad (8.4)$$

having some negative coefficients is not a CP map, thus it is not a physically realisable channel. In the following we will derive many inverse maps having this form, for which this result holds. Of course, we already know that a quantum channel cannot be inverted (apart from the trivial unitary case), so that if an inversion map is found, then it is certainly not CP. Nonetheless, this result is still of interest because it gives a nice and clear condition that can be used to quickly assess the nature of the maps under investigation. In addition, as shown in ref. [153], if a CPTP map is invertible, then its inverse is Hermitian preserving (HP), and so it admits an operator-sum form of Eq. (8.2).

8.2.2 Qubit systems and Pauli Transfer Matrix formalism

Our analysis is focused on quantum systems made of qubits ($\mathcal{H} = \mathbb{C}^2$), and we now briefly review some useful results on qubit channels. The identity and the Pauli matrices $\{\mathbb{I}, \sigma_x, \sigma_y, \sigma_z\}$ form a basis on $\mathcal{L}(\mathcal{H}) = \mathbb{C}^{2 \times 2}$, and so any operator $O \in \mathcal{L}(\mathcal{H})$ can be expressed in this basis as

$$O = \sum_{i=0}^3 o_i \sigma_i = o_0 \mathbb{I} + \mathbf{o} \cdot \boldsymbol{\sigma}, \quad o_i = \text{Tr}[\sigma_i O] \quad (8.5)$$

where we have introduced the vector of Pauli matrices $\boldsymbol{\sigma} := (\sigma_x, \sigma_y, \sigma_z)$, and the vector of coefficients $\mathbf{o} := (o_1, o_2, o_3) \in \mathbb{C}^3$. Similarly, as discussed in Sec. 2.1.3, density operators are expressed in

this basis in terms of their Bloch vector as $\rho = (\mathbb{I} + \mathbf{r} \cdot \boldsymbol{\sigma})/2$, with $\mathbf{r} = (r_x, r_y, r_z) \in \mathbb{R}^3$, and $|\mathbf{r}| \leq 1$, where equality holds only for pure states $\rho = |\psi\rangle\langle\psi|$ [220].

Since any operator O is completely specified by its components in the Pauli basis, we define its vector of coefficients as the column vector $|O\rangle\rangle := (o_0, o_1, o_2, o_3)^\top$, but we refer to refs. [75, 254] for a detailed discussion on the correspondence between operators and vectors.

In addition, every linear map $\Phi : \mathcal{L}(\mathbb{C}^2) \rightarrow \mathcal{L}(\mathbb{C}^2)$ can be represented in this basis as a 4×4 matrix Γ , whose action is given by [28, 36, 45, 117]

$$\begin{aligned} \Phi(O) \longrightarrow \Gamma|O\rangle\rangle &= \begin{bmatrix} \gamma_0 & \boldsymbol{\gamma} \\ \mathbf{t} & T \end{bmatrix} \begin{bmatrix} o_0 \\ \mathbf{o} \end{bmatrix} = \begin{bmatrix} \gamma_0 o_0 + \boldsymbol{\gamma} \cdot \mathbf{o} \\ o_0 \mathbf{t} + T \mathbf{o} \end{bmatrix}, \\ \text{or } \Phi(O) &= (\gamma_0 o_0 + \boldsymbol{\gamma} \cdot \mathbf{o}) \mathbb{I} + (o_0 \mathbf{t} + T \mathbf{o}) \cdot \boldsymbol{\sigma}, \end{aligned} \quad (8.6)$$

where $\boldsymbol{\gamma}$ and \mathbf{t} are row and column vectors respectively, and T is a 3×3 matrix. The matrix Γ associated to the map Φ is called *Pauli Transfer Matrix* (PTM), and its elements are given by

$$\Gamma_{ij} = \frac{1}{2} \text{Tr}[\sigma_i \Phi(\sigma_j)] \quad i, j \in \{0, 1, 2, 3\}. \quad (8.7)$$

If we restrict to trace-preserving maps, then $\boldsymbol{\gamma} = \mathbf{0}$ and $\gamma_0 = 1$, so the Γ matrix reduces to the simpler form

$$\Gamma = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{t} & T \end{bmatrix}. \quad (8.8)$$

Furthermore, if the map is also unital, that is it preserves the identity matrix $\Phi(\mathbb{I}) = \mathbb{I}$, then also $\mathbf{t} = \mathbf{0}$. As an example, the quantum bit-flip channel described by the map

$$\mathcal{N}_x(\rho) = (1-p)\rho + p\sigma_x\rho\sigma_x, \quad (8.9)$$

has a corresponding PTM representation as

$$\mathcal{N}_x \longrightarrow \Gamma_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1-2p & 0 \\ 0 & 0 & 0 & 1-2p \end{bmatrix}. \quad (8.10)$$

8.2.3 Quantum tomographic reconstruction

Quantum tomography [70, 72, 73, 200] is a method to estimate the ensemble average of any arbitrary operator by using measurement outcomes of a quorum of observables. The goal of a tomographic reconstruction of an observable is to identify a set of observables $\{Q_\lambda\}$, called *quorum* [79], such that the mean value $\langle O \rangle = \text{Tr}[O\rho]$ of any observable $O \in \mathcal{L}(\mathcal{H})$, for all states ρ , can be reconstructed by using measurements outcomes of the quorum observables. A tomographic reconstruction formula for an operator O is obtained by using a spectral decomposition of the identity in the operator Hilbert space [33, 78, 79, 225], namely

$$O = \int_{\Lambda} d\lambda \text{Tr}[C_\lambda^\dagger O] C_\lambda, \quad (8.11)$$

where λ is a parameter living in either a continuous or discrete manifold Λ , and operators C_λ depend on the quorum observables. The term $\mathbb{E}[O](Q_\lambda) := \text{Tr}[C_\lambda^\dagger O] C_\lambda$ is called *quantum estimator* of the operator O , and given a quantum state ρ , the expectation value $\langle O \rangle$ on such state amounts to

$$\begin{aligned} \langle O \rangle = \text{Tr}[O\rho] &= \int_{\Lambda} d\lambda \text{Tr}[OC_\lambda^\dagger] \text{Tr}[C_\lambda\rho] = \int_{\Lambda} d\lambda \text{Tr}[\mathbb{E}[O](Q_\lambda)\rho] \\ &= \int_{\Lambda} d\lambda \langle \mathbb{E}[O](Q_\lambda) \rangle. \end{aligned} \quad (8.12)$$

For qubit systems, the most common choice (but non unique, e.g. [70]) for the quorum are the Pauli matrices $\{Q_\lambda\}_\lambda = \{\sigma_x, \sigma_y, \sigma_z\}$, and the tomographic reconstruction formula results in the standard expansion in the Pauli basis, albeit with a slightly different notation (see Appendix E.2 for the explicit derivation), that is

$$\begin{aligned} \langle O \rangle &= \sum_{\alpha=x,y,z} \frac{1}{3} \langle \mathbb{E}[O](\sigma_\alpha) \rangle, \\ \mathbb{E}[O](\sigma_\alpha) &= \left(\frac{3 \text{Tr}[O\sigma_\alpha]}{2} \sigma_\alpha + \frac{\text{Tr}[O]}{2} \mathbb{I} \right). \end{aligned} \quad (8.13)$$

Note that the quantum tomographic reconstruction can be straightforwardly applied to multipartite quantum systems by simply using as a quorum the tensor product of single-system quorums [79].

8.3 Noise Deconvolution

The tomographic reconstruction formula can be used whenever one has access to the quantum state ρ and measurements of the quorum observables. In practical scenarios however, estimations are performed in the presence of noise and one generally deals with noisy quantum states $\rho \rightarrow \tilde{\rho} = \mathcal{N}(\rho)$ which then leads to noisy estimates $\langle O \rangle_{\tilde{\rho}} = \text{Tr}[O\mathcal{N}(\rho)]$. The idea of noise deconvolution is to correct the errors by considering a new quorum of observables taking into account the noise, and then use a noise inverted quantum estimator to recover the ideal estimates, namely the ones that we would obtain in the absence of noise.

Suppose the noise map \mathcal{N} acting on the quantum system can be formally inverted, that is there exist a linear (not CP) map \mathcal{N}^{-1} such that $(\mathcal{N}^{-1} \circ \mathcal{N})(\rho) = \rho \forall \rho$. Then, we say that the noise can be *deconvolved* in the following sense: instead of measuring the original observable O , we can evaluate the expectation value of the noise-inverted operator $\hat{\mathcal{N}}^{-1}(O)$, thus obtaining as a result the desired noise-free ideal result $\langle O \rangle$, that is

$$\langle \hat{\mathcal{N}}^{-1}(O) \rangle_{\tilde{\rho}} = \text{Tr}[\hat{\mathcal{N}}^{-1}(O)\mathcal{N}(\rho)] \text{Tr}[O\mathcal{N}^{-1}(\mathcal{N}(\rho))] = \text{Tr}[O\rho] = \langle O \rangle, \quad (8.14)$$

where $\hat{\mathcal{N}}^{-1}(\cdot)$ denotes the adjoint of the inverse map $\mathcal{N}^{-1}(\cdot)$, and in the second line we made explicit use of the definition of the adjoint map (see Appendix E.4.4 for a formal definition of adjoint map). The conditions for *deconvolving* the effect of a noise channel \mathcal{N} are [79, 225]:

- the inverted noise map exists, that is there is a \mathcal{N}^{-1} such that $(\mathcal{N}^{-1} \circ \mathcal{N})(O) = O \forall O \in \mathcal{L}(\mathcal{H})$.
- the quantum estimator $\mathbb{E}[O](Q_\lambda)$ is in the domain of \mathcal{N}^{-1} .
- the map $\mathcal{N}^{-1}(\mathbb{E}[O](Q_\lambda))$ is a function of Q_λ .

If these hold, then one can substitute the quantum estimator in Eq. (8.12), with the deconvolved quantum estimator $\hat{\mathcal{N}}^{-1}(\mathbb{E}[O](Q_\lambda))$, yielding

$$\begin{aligned} \int_\Lambda d\lambda \text{Tr}[\hat{\mathcal{N}}^{-1}(\mathbb{E}[O](Q_\lambda))\mathcal{N}(\rho)] &= \int_\Lambda d\lambda \text{Tr}[\mathbb{E}[O](Q_\lambda)\mathcal{N}^{-1}(\mathcal{N}(\rho))] \\ &= \int_\Lambda d\lambda \text{Tr}[\mathbb{E}[O](Q_\lambda)\rho] = \text{Tr}[O\rho] = \langle O \rangle. \end{aligned} \quad (8.15)$$

This procedure yields the ideal expectation value of any observable O on the state ρ , even if having access only to a noisy version of it and provided that the noise map is known (and invertible). Note that this definition is similar to that recently reported in ref. [49], regarding invertible noise channels with non-CPTP inverse. Specialising it for qubits, using Eq. (8.15) in (8.13), leads to (see Appendix E.3 for further details)

$$\langle O \rangle = \frac{1}{2} \text{Tr}[O] + \frac{1}{2} \sum_{\alpha=x,y,z} \text{Tr}[O\sigma_\alpha] \langle \hat{\mathcal{N}}^{-1}(\sigma_\alpha) \rangle_{\tilde{\rho}}. \quad (8.16)$$

Similarly to standard tomographic reconstruction, noise deconvolution can be applied also to multi qubits systems [254], in which case the mitigated tomographic estimates can be obtained considering the tensor product of the deconvolved quantum estimator of each subsystem. In addition, generally non-invertible maps could still be deconvolved if one restricts the attention only to a subset of states of interest upon which the given map is invertible [76, 77].

As shown later, the correction of the expectation value of a Pauli matrix is obtained by multiplying the noisy estimate —the one the experimenter has access to— by a constant depending on the noise, i.e. $\langle \sigma_\alpha \rangle_{\text{mitig}} = c \langle \sigma_\alpha \rangle_{\text{noisy}}$. This clearly increases the variance of the estimation, since $\text{Var}[\langle \sigma_\alpha \rangle_{\text{mitig}}] = c^2 \text{Var}[\langle \sigma_\alpha \rangle_{\text{noisy}}] \sim c^2/M$, where M is the number of measurement shots performed on the system, and thus the experimenter need to increase the outcome statistics proportionally to c^2 to reach a desired target precision.

We now proceed discussing how the deconvolution behaves in the presence of multiple noise channels. Consider two noise processes \mathcal{N}_0 and \mathcal{N}_1 separated by a unitary gate $\mathcal{U}(\cdot) = U \cdot U^\dagger$, as shown in Fig. 8.1(d). The action of the circuit is

$$(\mathcal{N}_1 \circ \mathcal{U} \circ \mathcal{N}_0)(\rho) = \mathcal{N}_1 \left(U \mathcal{N}_0(\rho) U^\dagger \right) = \mathcal{N}_1(\tilde{\rho}_U), \quad (8.17)$$

with $\tilde{\rho}_U = U \mathcal{N}_0(\rho) U^\dagger$. Using (8.15), it is possible to deconvolve the outermost noise \mathcal{N}_1 with

$$\text{Tr}[\hat{\mathcal{N}}_1^{-1}(O) \mathcal{N}_1(\tilde{\rho}_U)], \quad (8.18)$$

but not \mathcal{N}_0 , since the unitary U is in the way. Actually, one could decide to deconvolve the unitary as well, using the trivial inverse $\mathcal{U}_1^{-1}(\cdot) = U_1^\dagger \cdot U_1$, and thus making it possible to deconvolve also the first noise channel \mathcal{N}_0 , as

$$\text{Tr}[\hat{\mathcal{N}}_1^{-1}((U \hat{\mathcal{N}}_0^{-1}(O) U^\dagger) \mathcal{N}_1(\tilde{\rho}_U))] = \text{Tr}[(U \hat{\mathcal{N}}_0^{-1}(O) U^\dagger) U \mathcal{N}_0(\rho) U^\dagger] \quad (8.19)$$

$$= \text{Tr}[\hat{\mathcal{N}}_0^{-1}(O) \mathcal{N}_0(\rho)] = \text{Tr}[O \rho]. \quad (8.20)$$

However, this procedure cannot be employed to invert the noise that happens before a generic unitary U , since it essentially offloads the computation from the quantum computer to the classical one, by simulating the inverse evolution of the quantum system.

A more interesting case is obtained when the error map happens to commute with all the operations in the computation, as is the case for the depolarizing noise, described by the map

$$\mathcal{N}_{\text{dep}}(\rho) = \frac{p\mathbb{I}}{2} + (1-p)\rho, \quad (8.21)$$

for which it is easy to see that $(\mathcal{N}_{\text{dep}} \circ \mathcal{U})(\rho) = (\mathcal{U} \circ \mathcal{N}_{\text{dep}})(\rho) \forall \mathcal{U}(\cdot) = U \cdot U^\dagger$. Suppose one is performing a quantum computation given by a sequence of operations U_i , each one followed by depolarizing noise

$$\rho = \left(\prod_{i=1}^d \mathcal{N}_{\text{dep}}^{(i)} \circ \mathcal{U}_i \right) (\rho_0) = \left(\prod_{i=1}^d \mathcal{N}_{\text{dep}}^{(i)} \circ \prod_{i=1}^d \mathcal{U}_i \right) (\rho_0) = \mathcal{N}_{\text{dep}}^{\text{tot}}(\rho_U), \quad (8.22)$$

with $\mathcal{N}_{\text{dep}}^{\text{tot}} = \prod_i \mathcal{N}_{\text{dep}}^{(i)}$ the composition of all the depolarizing channels, and $\rho_U = \prod \mathcal{U}_i(\rho_0)$ the state obtained by the ideal noise-free computation. Most importantly, one can check that the composition of multiple depolarizing channels is still a depolarizing channel with probability parameter $1 - p_{\text{tot}} = \prod_i (1 - p_i)$, where p_i is the probability associated with each depolarizing noise. In such a case, it is possible to deconvolve all noise at once, using the deconvolution formula for the depolarizing noise with the total noise parameter p_{tot} (see Eq. (8.37)).

Similarly, this also holds for computations involving multi qubits subject to *global* depolarizing errors. The authors in ref. [310] leverage this property to perform a simple yet effective error

mitigation technique for quantum computers, based on the assumption that noise in quantum circuits is well described by global depolarizing error channels. While exact depolarizing errors (either local or global) are hardly found in realistic quantum circuits where errors are both due to *coherent* (i.e. unitary) and *incoherent* noise (i.e. interaction), Pauli twirling and randomised compiling techniques [96, 127, 311, 313] can be used to approximately tailor noise to stochastic Pauli channels, preferably depolarizing noise, and then use the procedure above to mitigate it [306].

8.4 Inversion of common noise maps

We now proceed by explicitly evaluating the inverse maps of some of the most common noisy channels, leveraging the Pauli Transfer Matrix formalism introduced in Sec 8.2. The general method for finding the inverse map goes as follows: we first evaluate the matrix representation (8.6) of the channel, we then invert this matrix, and from this recover the operator sum representation of the inverse channel whenever this exists. We start from simpler cases to build some intuition on the construction of the inverse maps, and then proceed towards more complicated cases. In Table 8.1 we summarise the results obtained in this section, comprising all noise channels considered in this analysis together with their inverse maps.

	Noise Channel $\mathcal{N}(\rho)$	Inverse Map $\mathcal{N}^{-1}(O)$
Bit-Flip	$(1-p)\rho + p\sigma_x\rho\sigma_x$	$\frac{1-p}{1-2p}O - \frac{p}{1-2p}\sigma_x O\sigma_x$
Phase-Flip (<i>dephasing</i>)	$(1-p)\rho + p\sigma_z\rho\sigma_z$	$\frac{1-p}{1-2p}O - \frac{p}{1-2p}\sigma_z O\sigma_z$
Bit-Phase-Flip	$(1-p)\rho + p\sigma_y\rho\sigma_y$	$\frac{1-p}{1-2p}O - \frac{p}{1-2p}\sigma_y O\sigma_y$
Depolarizing	$(1-p)\rho + p\frac{\mathbb{I}}{2}$	$\frac{1}{1-p}\left(O - \frac{p}{2}\text{Tr}[O]\mathbb{I}\right)$
General Pauli Channel (see Eq. (8.39))	$p_0\rho + \sum_{k=x,y,z} p_k\sigma_k\rho\sigma_k$	$\beta_0 O + \sum_{k=x,y,z} \beta_k\sigma_k O\sigma_k$
Amplitude Damping	$V_0\rho V_0 + V_1\rho V_1^\dagger$ $V_0 = 0\rangle\langle 0 + \sqrt{1-\gamma} 1\rangle\langle 1 $ $V_1 = \sqrt{\gamma} 0\rangle\langle 1 $	$K_0 O K_0 - K_1 O K_1^\dagger$ $K_0 = 0\rangle\langle 0 + \sqrt{\frac{1}{1-\gamma}} 1\rangle\langle 1 $ $K_1 = \sqrt{\frac{\gamma}{1-\gamma}} 0\rangle\langle 1 $
2-Kraus Channel (see Eq. (8.48))	$A_0\rho A_0 + A_1\rho A_1^\dagger$ $A_0 = \cos\alpha 0\rangle\langle 0 + \cos\beta 1\rangle\langle 1 $ $A_1 = \sin\beta 0\rangle\langle 1 + \sin\alpha 1\rangle\langle 0 $	$B_0 O B_0^\dagger - B_1 O B_1^\dagger$ $B_0 = \sqrt{h_{\alpha\beta}}(\cos\beta 0\rangle\langle 0 + \cos\alpha 1\rangle\langle 1)$ $B_1 = \sqrt{h_{\alpha\beta}}(\sin\beta 0\rangle\langle 1 + \sin\alpha 1\rangle\langle 0)$

Table 8.1: Table summarising the results of the present analysis, consisting of some of the most common single-qubit noisy channels \mathcal{N} , along with their inverse noise maps \mathcal{N}^{-1} , defined as the map such that $(\mathcal{N}^{-1} \circ \mathcal{N})(\rho) = \rho \forall \rho$. Clearly, all noise channels are CPTP maps, while the inverse channels are not, yet they admit an operator-sum representation. All the noise maps except for amplitude damping and 2-Kraus channel have trivial adjoint channels, so one must pay attention in using the adjoint channel inside the deconvolution formula (8.16). As discussed in Eq. (8.48), the coefficient in the inverse 2-Kraus channel is $h_{\alpha\beta} = 2/(\cos 2\alpha + \cos 2\beta)$.

8.4.0.1 Bit-flip, phase-flip and bit-phase-flip

The bit-flip, phase-flip and bit-phase-flip channels are described by the Kraus operators, $A_0 = \sqrt{p}\mathbb{I}$ and $A_{1,\alpha} = \sqrt{1-p}\sigma_\alpha$, with $\sigma_\alpha \in \{\sigma_x, \sigma_z, \sigma_y\}$ respectively. For simplicity, in the following we focus only on the bit-flip channel (generated by σ_x), but the results hold equivalently also for the other two channels. The bit-flip channel acts as

$$\mathcal{N}_x(\rho) = (1-p)\rho + p\sigma_x\rho\sigma_x, \quad (8.23)$$

and its PTM is given by

$$\Gamma_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1-2p & 0 \\ 0 & 0 & 0 & 1-2p \end{bmatrix}. \quad (8.24)$$

In order to find an operator-sum expression for the inverse map \mathcal{N}_x^{-1} , consider the inverse matrix

$$\Gamma_x^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{(1-2p)} & 0 \\ 0 & 0 & 0 & \frac{1}{(1-2p)} \end{bmatrix}. \quad (8.25)$$

It's clear that Γ_x can be inverted provided that $p \neq 1/2$, since in that case $\det\Gamma_x = 0$. This is not a problem for real case scenarios, where the probability of errors are usually small, and can safely assume $0 < p < 1/2$. We now proceed using a derivation similar to that proposed in [36].

Note that Γ_x^{-1} is diagonal in the Pauli basis, thus has eigenvectors $\{|\mathbb{I}\rangle, |\sigma_x\rangle, |\sigma_y\rangle, |\sigma_z\rangle\}$ with eigenvalues $\boldsymbol{\lambda} = (1, 1, (1-2p)^{-1}, (1-2p)^{-1})$ respectively. Now, consider the generic map

$$\mathcal{E}(O) = \sum_{k=0}^3 \beta_k \sigma_k O \sigma_k. \quad (8.26)$$

Also this map has eigenvectors $\{\mathbb{I}, \boldsymbol{\sigma}\}$, but with eigenvalues $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3)$. Since two maps are equal if they have the same action on a basis, if we can find a way to match the two sets of eigenvalues $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}$, we would then recover the operator-sum representation for Γ_x^{-1} .

By evaluating the PTM $\Gamma_{\mathcal{E}}$ of $\mathcal{E}(\cdot)$ (8.26), we can relate the coefficients in the operator-sum representation (8.26), with those appearing in the expression for Γ_x^{-1} (see Appendix E.4 for a derivation). In particular, we want these to hold

$$(\Gamma_x^{-1})_{11} = \beta_0 + \beta_1 - \beta_2 - \beta_3, \quad (8.27)$$

$$(\Gamma_x^{-1})_{22} = \beta_0 - \beta_1 + \beta_2 - \beta_3, \quad (8.28)$$

$$(\Gamma_x^{-1})_{33} = \beta_0 - \beta_1 - \beta_2 + \beta_3, \quad (8.29)$$

plus the trace-preserving condition $1 = \beta_0 + \beta_1 + \beta_2 + \beta_3$, that the inverse map must satisfy because the direct map is trace-preserving. This condition is inherently satisfied by Γ_x^{-1} , since its first row has the form $(1, 0, 0, 0)$. This system has solutions $\beta_0 = (1-p)/(1-2p)$, $\beta_1 = -p/(1-2p)$, and $\beta_2 = \beta_3 = 0$, and substituting them back into Eq. (8.26), we obtain the operator-sum representation of the inverse bit-flip map

$$\mathcal{N}_x^{-1}(O) = \frac{1-p}{1-2p}O - \frac{p}{1-2p}\sigma_x O \sigma_x. \quad (8.30)$$

By virtue of Corollary II, and noticing that the coefficients appearing in the expression above have always opposite signs, we are sure that this map is not CP, as expected, yet it possesses an

operator-sum representation. Note how similar the direct and inverse map are, a feature which we will encounter in all the cases discussed here.

The same procedure can be applied to phase-flip (also referred to as *dephasing*, generated by σ_z), and bit-phase-flip (generated by σ_y) channels, yielding inverse maps

$$\mathcal{N}_z^{-1}(O) = \frac{1-p}{1-2p}O - \frac{p}{1-2p}\sigma_z O \sigma_z, \quad (8.31)$$

$$\mathcal{N}_y^{-1}(O) = \frac{1-p}{1-2p}O - \frac{p}{1-2p}\sigma_y O \sigma_y. \quad (8.32)$$

We can plug these inversion maps in the deconvolution formula (8.16) to obtain noise-free expectation values of observables. In particular, assume we are measuring a Pauli matrix $O = \sigma_\alpha$, and that the system is subject to one of the noise processes $\rho \rightarrow \rho_\beta = \mathcal{N}_\beta(\rho)$ with $\beta = \{x, y, z\}$. Then the ideal expectation values $\langle \sigma_\alpha \rangle_\rho = \text{Tr}[\sigma_\alpha \rho]$ can be expressed in compact form as (see Appendix E.4 for the explicit derivation)

$$\langle \sigma_\alpha \rangle = \delta_{\alpha\beta} \langle \sigma_\alpha \rangle_{\rho_\beta} + (1 - \delta_{\alpha\beta}) \frac{1}{1-2p} \langle \sigma_\alpha \rangle_{\rho_\beta}, \quad (8.33)$$

where $\delta_{\alpha\beta}$ is a Kronecker delta. It is then clear that if the noise happens along the measurement direction ($\alpha = \beta$), then the noise does not affect the measurement statistics, as the ideal and noisy value coincide. While for orthogonal directions ($\alpha \neq \beta$), these are equally contracted by a factor $1 - 2p$, thus recovering the usual pictorial representation of the contracting Bloch sphere on the plane orthogonal to the noise [220].

8.4.0.2 Depolarizing noise

The depolarizing noise channel is defined as

$$\mathcal{N}_{\text{dep}}(\rho) = (1-p)\rho + \frac{p}{2}\mathbb{I}, \quad (8.34)$$

whose action is to leave the state untouched with probability $1 - p$, and sends it to the completely mixed state $\mathbb{I}/2$ with probability p . The channel can be expressed in Kraus form in multiple ways, one of them being [220]

$$\mathcal{N}_{\text{depol}}(\rho) = \left(1 - \frac{3p}{4}\right)\rho + \frac{p}{4}\left(\sigma_x \rho \sigma_x + \sigma_y \rho \sigma_y + \sigma_z \rho \sigma_z\right), \quad (8.35)$$

with corresponding Kraus operators $A_0 = \sqrt{1-3p/4}\mathbb{I}$, $A_1 = \sqrt{p}\sigma_x/2$, $A_2 = \sqrt{p}\sigma_y/2$, $A_3 = \sqrt{p}\sigma_z/2$. Following the same procedure outlined before for the bit-flip channel, one can easily recover the inverse linear map (see Appendix E.4 for explicit derivation)

$$\mathcal{N}_{\text{depol}}^{-1}(O) = \frac{1}{1-p}\left(O - \frac{p}{2}\text{Tr}[O]\mathbb{I}\right). \quad (8.36)$$

While this is already a known result in the literature [33, 74, 79, 139, 299], it is presented without an explicit constructive derivation, as given here.

Using this formula in the deconvolution tomographic reconstruction (8.16), we find

$$\langle O \rangle = \frac{1}{2}\text{Tr}[O] + \sum_{\alpha} \frac{\text{Tr}[O\sigma_\alpha]}{1-p} \langle \sigma_\alpha \rangle_{\mathcal{N}_{\text{dep}}(\rho)}, \quad (8.37)$$

where it is clear that to counterbalance the effect of the depolarizing channel, whose effect on the Bloch sphere is to contract it uniformly, one needs perform an expansion of the same amount, obtained dividing by $1 - p$.

While our analysis is focused only on single qubit systems, it is worth noticing that a similar approach can be used to correct correlated and asymmetric depolarizing channels acting on multi-qubits systems [47], as recently shown in [254].

8.4.0.3 General Pauli channel

A more general and interesting case is that of general Pauli channels, where noise acts with different strength along the three Pauli axes, defined as

$$\mathcal{N}_{\mathbf{p}}(\rho) = p_0 O + p_x \sigma_x \rho \sigma_x + p_y \sigma_y \rho \sigma_y + p_z \sigma_z \rho \sigma_z. \quad (8.38)$$

The channel is parametrized by the probabilities $\mathbf{p} = (p_0, p_x, p_y, p_z)$, with the trace-preserving condition implying $p_0 = 1 - p_x - p_y - p_z$. Importantly, upon choosing appropriate values for \mathbf{p} , this channel reduces to all noise maps treated before. Though of considerable more general structure, the inverse map of this channel is derived using the same machinery developed above, and eventually one obtains

$$\begin{aligned} \mathcal{N}_{\mathbf{p}}^{-1}(O) &= \beta_0 O + \beta_1 \sigma_x O \sigma_x + \beta_2 \sigma_y O \sigma_y + \beta_3 \sigma_z O \sigma_z, \quad \text{with} \\ \beta_0 &= \frac{1}{4} \left(1 + \frac{1}{1-2(p_x+p_y)} + \frac{1}{1-2(p_x+p_z)} + \frac{1}{1-2(p_y+p_z)} \right), \\ \beta_1 &= \frac{1}{4} \left(1 - \frac{1}{1-2(p_x+p_y)} - \frac{1}{1-2(p_x+p_z)} + \frac{1}{1-2(p_y+p_z)} \right), \\ \beta_2 &= \frac{1}{4} \left(1 - \frac{1}{1-2(p_x+p_y)} + \frac{1}{1-2(p_x+p_z)} - \frac{1}{1-2(p_y+p_z)} \right), \\ \beta_3 &= \frac{1}{4} \left(1 + \frac{1}{1+2(p_x+p_y)} - \frac{1}{1-2(p_x+p_z)} - \frac{1}{1-2(p_y+p_z)} \right). \end{aligned} \quad (8.39)$$

One can check that varying \mathbf{p} it is possible to recover the inverse maps of all the cases treated before. For example, for $\mathbf{p} = (1-p, p, 0, 0)$ corresponding to the bit-flip channel in Eq. (8.23), one gets $\beta_0 = (1-p)/(1-2p)$ and $\beta_1 = -p/(1-2p)$, as in Eq. (8.30).

The noise deconvolution applied to measurements of Pauli matrices $O \in \{\sigma_x, \sigma_y, \sigma_z\}$, leads to the following relations

$$\begin{aligned} \langle \sigma_x \rangle &= \frac{1}{1-2(p_y+p_z)} \langle \sigma_x \rangle_{\mathcal{N}_{\mathbf{p}}(\rho)}, \\ \langle \sigma_y \rangle &= \frac{1}{1-2(p_x+p_z)} \langle \sigma_y \rangle_{\mathcal{N}_{\mathbf{p}}(\rho)}, \\ \langle \sigma_z \rangle &= \frac{1}{1-2(p_x+p_y)} \langle \sigma_z \rangle_{\mathcal{N}_{\mathbf{p}}(\rho)}, \end{aligned} \quad (8.40)$$

which can be used together with Eq. (8.16) to reconstruct the expectation value of a general observable O . As before, we see that the noise disturbs the estimation along orthogonal directions. Note that the explicit inversion of the general Pauli channel was also recently reported in ref. [292].

8.4.0.4 Amplitude Damping

The amplitude damping (AD) channel describes the energy loss of a quantum system, for example obtained through relaxation from the excited to the ground state. Its Kraus representation is

$$\mathcal{N}_{\text{AD}}(\rho) = V_0 \rho V_0^\dagger + V_1 \rho V_1^\dagger, \quad V_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, \quad V_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}, \quad (8.41)$$

where $\gamma \in [0, 1]$ is a parameter that encodes the strength of the energy loss process, which for real systems is often expressed in terms of characteristic decay times, as discussed in Sec. 8.5.

While still being trace preserving (TP), amplitude damping channel is not unital, since $\mathcal{N}_{\text{AD}}(\mathbb{I}) = \mathbb{I} + \gamma Z$. This in turn implies that the Pauli Transfer Matrix Γ_{AD} is not diagonal, but has an addition nonzero element in the last row of first column. This changes the derivation of the

inverse map with respect to the previous cases, but it can still be carried out without major changes, as shown in Appendix E.4.4. The inverse linear map in operator-sum representation is then found to be

$$\mathcal{N}_{\text{AD}}^{-1}(\rho) = K_0 O K_0^\dagger - K_1 O K_1^\dagger \quad K_0 = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\sqrt{1-\gamma}} \end{bmatrix}, \quad K_1 = \begin{bmatrix} 0 & \sqrt{\frac{\gamma}{1-\gamma}} \\ 0 & 0 \end{bmatrix}. \quad (8.42)$$

Up until now, all noisy channels (and their inverse maps) had trivial *adjoint* map, since all Kraus operators were Hermitian. However this is not the case for amplitude damping, since both $V_1 \neq V_1^\dagger$ and $K_1 \neq K_1^\dagger$. Thus, one must be careful in applying the adjoint inverse $\hat{\mathcal{N}}^{-1}$ in Eq. (8.16), and not just \mathcal{N}^{-1} of (8.42) (see Appendix E.4.4 for an extended discussion). Deconvolution of amplitude damping for measurements of the Pauli matrices leads to

$$\begin{aligned} \langle \sigma_x \rangle &= \frac{1}{\sqrt{1-\gamma}} \langle \sigma_x \rangle_{\mathcal{N}_{\text{AD}}(\rho)}, \\ \langle \sigma_y \rangle &= \frac{1}{\sqrt{1-\gamma}} \langle \sigma_y \rangle_{\mathcal{N}_{\text{AD}}(\rho)}, \\ \langle \sigma_z \rangle &= \frac{1}{1-\gamma} (\langle \sigma_z \rangle_{\mathcal{N}_{\text{AD}}(\rho)} - \gamma). \end{aligned} \quad (8.43)$$

Similarly, one can also obtain the inverse map of the *generalised amplitude damping* (GAD) channel, used to model the interaction of a qubit with an environment at a finite temperature [46, 220]. Such channel is parameterised by two parameters γ and p , and it is defined as

$$\begin{aligned} \mathcal{N}_{\text{GAD}}(\rho) &= A_0 \rho A_0^\dagger + A_1 \rho A_1^\dagger + A_2 \rho A_2^\dagger + A_3 \rho A_3^\dagger \\ A_0 &= \sqrt{p} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, \quad A_1 = \sqrt{p} \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}, \\ A_2 &= \sqrt{1-p} \begin{bmatrix} \sqrt{1-\gamma} & 0 \\ 0 & 1 \end{bmatrix}, \quad A_3 = \sqrt{1-p} \begin{bmatrix} 0 & \sqrt{\gamma} \\ \sqrt{\gamma} & 0 \end{bmatrix}. \end{aligned} \quad (8.44)$$

One can check that the following map is the inverse of the GAD channel

$$\begin{aligned} \mathcal{N}_{\text{GAD}}^{-1}(\rho) &= B_0 \rho B_0^\dagger - B_1 \rho B_1^\dagger + B_2 \rho B_2^\dagger - B_3 \rho B_3^\dagger \\ B_0 &= \sqrt{p} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{\frac{1}{1-\gamma}} \end{bmatrix}, \quad B_1 = \sqrt{p} \begin{bmatrix} 0 & \sqrt{\frac{\gamma}{1-\gamma}} \\ 0 & 0 \end{bmatrix}, \\ B_2 &= \sqrt{1-p} \begin{bmatrix} \sqrt{\frac{1}{1-\gamma}} & 0 \\ 0 & 1 \end{bmatrix}, \quad B_3 = \sqrt{1-p} \begin{bmatrix} 0 & \sqrt{\frac{\gamma}{1-\gamma}} \\ \sqrt{\frac{\gamma}{1-\gamma}} & 0 \end{bmatrix}, \end{aligned} \quad (8.45)$$

with corresponding noise deconvolved Pauli expectation values given by

$$\begin{aligned} \langle \sigma_x \rangle &= \frac{1}{\sqrt{1-\gamma}} \langle \sigma_x \rangle_{\mathcal{N}_{\text{GAD}}(\rho)}, \\ \langle \sigma_y \rangle &= \frac{1}{\sqrt{1-\gamma}} \langle \sigma_y \rangle_{\mathcal{N}_{\text{GAD}}(\rho)}, \\ \langle \sigma_z \rangle &= \frac{1}{1-\gamma} (\langle \sigma_z \rangle_{\mathcal{N}_{\text{GAD}}(\rho)} - \gamma(2p-1)). \end{aligned} \quad (8.46)$$

8.4.0.5 Two-Kraus channels

We conclude the analysis of single-qubit noise channels by considering the set of channels generated by two parametrized Kraus operators, namely

$$\mathcal{N}_{\text{two}}(\rho) = \sum_{i=1,2} A_i \rho A_i^\dagger, \quad (8.47)$$

with $A_1 = \cos \alpha |0\rangle\langle 0| + \cos \beta |1\rangle\langle 1|$, and $A_2 = \sin \beta |0\rangle\langle 1| + \sin \alpha |1\rangle\langle 0|$. This channel reduces to bit-flip for $\alpha = \beta$, and to amplitude damping for $\alpha = 0$. Following a procedure similar to the amplitude damping case, the inverse map of the two-Kraus channels is found to be

$$\begin{aligned} \mathcal{N}_{\text{two}}(O)^{-1} &= B_1 O B_1^\dagger - B_2 O B_2^\dagger, \\ B_1 &= \begin{bmatrix} \frac{\sqrt{2} \cos \beta}{\sqrt{\cos 2\alpha + \cos 2\beta}} & 0 \\ 0 & \frac{\sqrt{2} \cos \alpha}{\sqrt{\cos 2\alpha + \cos 2\beta}} \end{bmatrix}, \\ B_2 &= \begin{bmatrix} 0 & \frac{\sqrt{2} \sin \beta}{\sqrt{\cos 2\alpha + \cos 2\beta}} \\ \frac{\sqrt{2} \sin \alpha}{\sqrt{\cos 2\alpha + \cos 2\beta}} & 0 \end{bmatrix}. \end{aligned} \quad (8.48)$$

Similarly to amplitude damping, one of the generators (B_2) is not Hermitian, thus we must employ the adjoint inverse map when evaluating the deconvolved expectation values. By straightforward calculations the following holds

$$\begin{aligned} \langle \sigma_x \rangle &= \frac{1}{\cos(\alpha - \beta)} \langle \sigma_x \rangle_{\mathcal{N}_{\text{two}}(\rho)}, \\ \langle \sigma_y \rangle &= \frac{1}{\cos(\alpha + \beta)} \langle \sigma_y \rangle_{\mathcal{N}_{\text{two}}(\rho)}, \\ \langle \sigma_z \rangle &= h_{\alpha\beta} (\cos^2 \beta + \sin^2 \alpha - 1 + \langle \sigma_z \rangle_{\mathcal{N}_{\text{two}}(\rho)}), \end{aligned} \quad (8.49)$$

with $h_{\alpha\beta} = 2/(\cos(2\alpha) + \cos(2\beta))$. Note that upon varying the parameters α and β , the formulas above correctly reduce to the other limiting channels. For example, setting $\alpha = 0$ leads to amplitude damping channel in Eq. (8.43) with $\cos(\beta) := \sqrt{1 - \gamma}$.

8.5 Experimental deconvolution

In this section we provide some concrete applications of the noise deconvolution procedures for qubit tomography outlined above. In particular, we show both numerically and by experimentation on superconducting quantum hardware by Rigetti how to address a *decoherence* noise model, and we also provide numerical evidence for the deconvolution of the general Pauli channel (8.38). All simulations are performed using PyQuil and the real quantum device used is ‘‘Aspen-9’’, accessed via Rigetti’s Quantum Cloud Services (QCS) [157, 286].

8.5.1 Decoherence noise model

The concurrent action of a dephasing channel followed by amplitude damping is referred to as *decoherence* noise, which is an effective way to describe the noisy evolution a qubit undergoes due to uncontrolled interaction with its external environment. Using the definitions (8.23) and (8.41), one obtains

$$\begin{aligned} \mathcal{N}_{\text{dec}}(\rho) &= (\mathcal{N}_{\text{AD}}(\gamma) \circ \mathcal{N}_z(p)) \left(\begin{bmatrix} a & b \\ c & 1-a \end{bmatrix} \right) \\ &= \begin{bmatrix} 1 - (1-a)(1-\gamma) & (1-2p)\sqrt{1-\gamma}b \\ (1-2p)\sqrt{1-\gamma}c & (1-\gamma)(1-a) \end{bmatrix} \\ &= \begin{bmatrix} 1 - (1-a)e^{-t/T_1} & e^{-t/T_2} b \\ e^{-t/T_2} c & e^{-t/T_1} (1-a) \end{bmatrix}, \end{aligned} \quad (8.50)$$

where we have introduced the relaxation times T_1 and T_2 characterising the ‘‘quality’’ of the physical qubits. These are related to the noise parameters γ and p through the following relations

$$\gamma = 1 - e^{-t/T_1}, \quad p = \frac{1}{2} \left(1 - e^{-(t/T_2 - t/2T_1)} \right), \quad (8.51)$$

where t is a time parameter indicating the duration of the noise process.

Since the correction terms in the deconvolution formulas for dephasing (8.43) and amplitude damping (8.33) are multiplicative, for a decoherence channel these combine as

$$\begin{aligned}\langle\sigma_x\rangle &= \frac{1}{(1-2p)} \frac{1}{\sqrt{1-\gamma}} \langle\sigma_x\rangle_{\mathcal{N}_{\text{dec}}(\rho)}, \\ \langle\sigma_y\rangle &= \frac{1}{(1-2p)} \frac{1}{\sqrt{1-\gamma}} \langle\sigma_y\rangle_{\mathcal{N}_{\text{dec}}(\rho)}, \\ \langle\sigma_z\rangle &= \frac{1}{1-\gamma} (\langle\sigma_z\rangle_{\mathcal{N}_{\text{dec}}(\rho)} - \gamma).\end{aligned}\tag{8.52}$$

Additionally, if the quantum system under investigation is subject to repeated applications of a decoherence noise channel, i.e. $\mathcal{N}_{\text{dec}}^{\circ m}(\rho) = \mathcal{N}_{\text{dec}}^{(1)} \circ \mathcal{N}_{\text{dec}}^{(2)} \cdots \circ \mathcal{N}_{\text{dec}}^{(m)}(\rho)$, then the ideal expectation values are obtained through the following equations

$$\begin{aligned}\langle\sigma_x\rangle &= \frac{1}{((1-2p)\sqrt{1-\gamma})^m} \langle\sigma_x\rangle_{\mathcal{N}_{\text{dec}}(\rho)}, \\ \langle\sigma_y\rangle &= \frac{1}{((1-2p)\sqrt{1-\gamma})^m} \langle\sigma_y\rangle_{\mathcal{N}_{\text{dec}}(\rho)}, \\ \langle\sigma_z\rangle &= \frac{1}{(1-\gamma)^m} (\langle\sigma_z\rangle_{\mathcal{N}_{\text{dec}}(\rho)} - 1 + (1-\gamma)^m).\end{aligned}\tag{8.53}$$

In Figure 8.2 we show the application of these formulas to deconvolve the decoherence noise occurring on a qubit. The specific quantum circuit used for the experiments is shown in Figure 8.2a: first the system is prepared in the superposition state $|+\rangle = H|0\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, then we let qubit decohere for a certain amount of time dictated by the number D of (noisy) identities each of which supposedly takes a time t , and at last we measure the expectation value of the operator σ_x . Clearly, in a noise-free scenario, the result would always be $\langle\sigma_x\rangle = 1$, independent of the depth D . Figure 8.2c shows a simulation of these circuits with stochastic measurement outcomes for different values of D , and for a given choice of noise parameters p and γ . For comparison, the individual effect of dephasing and amplitude damping channels alone are also showed. Thanks to Eq. (8.53) we can invert the effect of the decoherence noise, and so retrieve the ideal noise-free results.

We also tested this procedure on real superconducting quantum hardware provided by Rigetti, in particular on the device ‘‘Aspen-9’’, whose topology is reported in Fig. 8.2b. The device comes with the calibration data reporting the T_1 and T_2 parameters for any qubit, as well as the time duration of a single gate. Identities in the circuits are used to introduce time delays, and thus let the qubit decohere for longer intervals of time, depending on the depth D . Differently from the previous simulations where only the identities are supposed to introduce (decoherence) noise, in the real case scenario noise happens along the whole computation, including state preparation, application of all gates in the circuit (both Hadamards and Identities), and finally measurement errors. Of these, the most detrimental are undoubtedly readout errors, and we addressed them by using the standard mitigation technique of calibrating the device and inverting the assignment probability matrix to recover readout mitigated results. Calibration data reports that the time it takes to execute a single qubit identity gate is $t = 40$ ns, and together with T_1 and T_2 , these are used to calculate the parameters p and γ of the decoherence noise, using relations (8.51). These are in turn used inside the deconvolution formulas to recover the noise-free results. Figures 8.2d and 8.2e show the results of the execution of circuit Fig. 8.2a on qubits 4 and 25, respectively.

The noise mitigation procedure on qubit 4 shown in panel 8.2d yields slightly unphysical results, in that the mitigated expectation values exceeds one at times. A naive solution to this problem could be to impose that the mitigation results are in the physical range $\langle\sigma_\alpha\rangle \in [-1, +1]$, so that if the result exceed the limits, it should be substituted with the appropriate physical bound. Though, assuming a gate time duration of $t = 35$ ns instead of standard 40 ns, yields results which are more

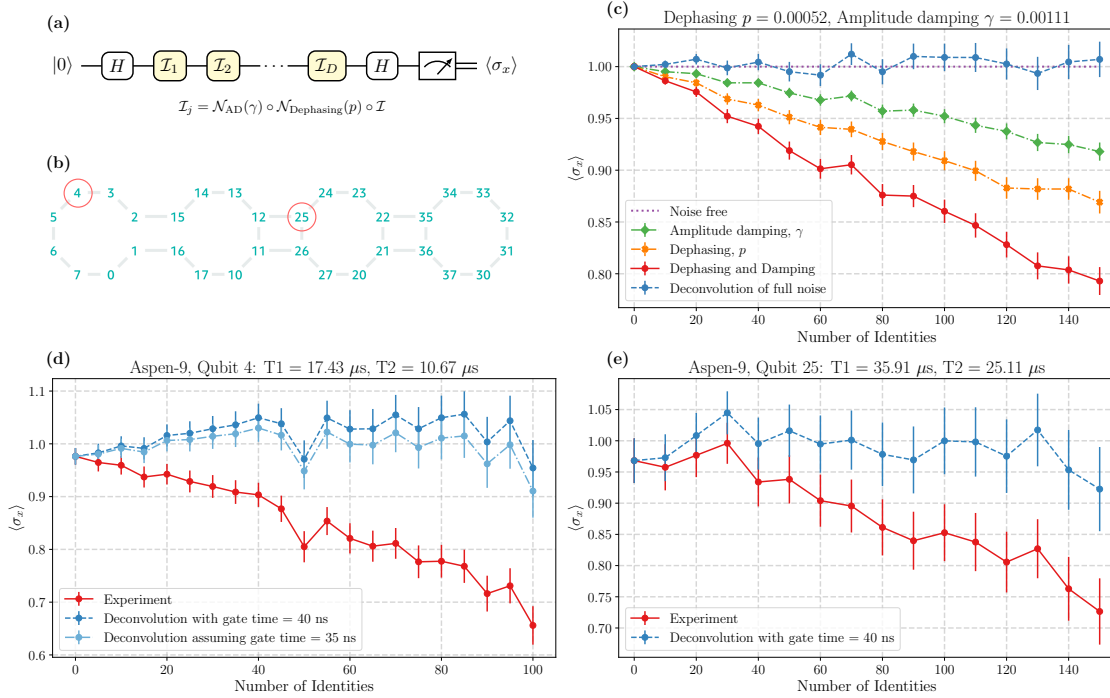


Figure 8.2: Deconvolution of decoherence noise both on a simulator and the real quantum device Aspen-9 by Rigetti. **(a)** Scheme of the quantum circuit used in the simulations and runs on the actual quantum device. A qubit is prepared in the superposition state and then it is left to decohere for a certain amount of time, dependent on the number D of identities in the circuit. Eventually the qubit is measured in the x -basis to estimate the expectation value σ_x . **(b)** Scheme of Aspen-9, the real quantum device by Rigetti used to run the quantum circuit. **(c)** Simulation of the decoherence noise for dephasing (p) and damping (γ) intensities equal to those characterising qubit 25 of Aspen-9, with gate duration of 40 ns. For comparison, the effect of the action of these channels alone is also showed. Using the deconvolution formulas for decoherence noise (8.53), it is possible to mitigate the decay caused by the noise, and recover the ideal result. Each expectation value is estimated evaluating the mean over $n_{\text{shots}} = 2048$ measurement outcomes, and the error bars showed are the statistical error of the mean. **(d)** Results obtained from running the circuit on qubit 4 of Aspen-9, characterised by relaxation times $T_1 = 17.43 \cdot 10^{-6}$ s and $T_2 = 10.67 \cdot 10^{-6}$ s, with $n_{\text{shots}} = 2048$, and the error bars are twice the error of the mean. See main text for comments on the results. **(e)** Results obtained from running the circuit on qubit 25 of Aspen-9, characterised by relaxation times $T_1 = 35.91 \cdot 10^{-6}$ s and $T_2 = 25.11 \cdot 10^{-6}$ s, with $n_{\text{shots}} = 1024$. Also in this case the error bars are equals to twice the error of the mean. See main text for comments on the results.

in agreement with the expected theoretical behaviour for decoherence noise, as the deconvoluted results are compatible with one, as expected. This hints that either the quality of the qubit is better then reported in the available calibration data (either due to shorter gate times t , or larger T_1 and T_2), or that the decoherence model alone poorly describes the noise happening on idle qubit 4 left interacting with the external environment. However, the good accordance between the deconvoluted results with $t = 35$ ns and the experiments suggests the first hypothesis to hold.

Such conclusion is also corroborated by the experimental results obtained with qubit 25. In fact, using the deconvolution formulas with reported T_1 , T_2 and standard gate time ($t = 40$ ns), we are able to mitigate the effect of noise with good accuracy, as showed in Fig. 8.2e, hinting that indeed the decay law of the qubit is well described through a decoherence noise model of Eq. (8.50). Also, note that the simulation in Fig. 8.2c is tuned with the same noise parameters p and γ characterising

qubit 25. Apart from fluctuations due to, e.g. imperfect readout, stochastic measurement outcomes, and noisy Hadamards, there is good agreement between the simulated (red curve in panel (c)) and experimental result (red curve in panel (e)). We do not report analogous experiments using other qubits in the device that produced obviously biased data.

8.5.2 Arbitrary Pauli channel

We implemented a simulation of the noise deconvolution of the general Pauli channel (8.38), using the quantum virtual machine (QVM) simulator provided with PyQuil [286]. The simulated circuit is showed on top of Figure 8.3. A qubit starting in the ground state is rotated in the Bloch sphere around the y axis via $R_Y(\theta) = e^{-i\theta\sigma_y/2}$, and then it is subject to the general Pauli noise (yellow box), simulated applying a Pauli transformation chosen randomly with probabilities p_x , p_y and p_z . At last, we estimate the expectation value of the three Pauli matrices by appending the appropriate change of basis gate, i.e. $M_j \in \{\mathbb{I}, H, HS^\dagger\}$ for $\{\sigma_z, \sigma_x, \sigma_y\}$ respectively.

The noise parameters (p_x, p_y, p_z) are used within the deconvolution formulas (8.40) to recover the mitigated results (green curve), which are, as expected, in perfect agreement with the ideal noise-free ones, obtained from executing the quantum circuit without the noisy channel (red curve).

8.6 Conclusions

In conclusion we have shown how mathematically invertible noise maps can always be removed from the final measurement stage, so that one can obtain unbiased expectation values of general observables provided that the noise process is known. We illustrated the method on most known qubit noise maps, and systematically derived their inverse maps (see Table 8.1). We simulated the noise deconvolution procedure for the case of a general Pauli channel (Fig. 8.3) and illustrated our method on noise on actual quantum hardware (Fig. 8.2).

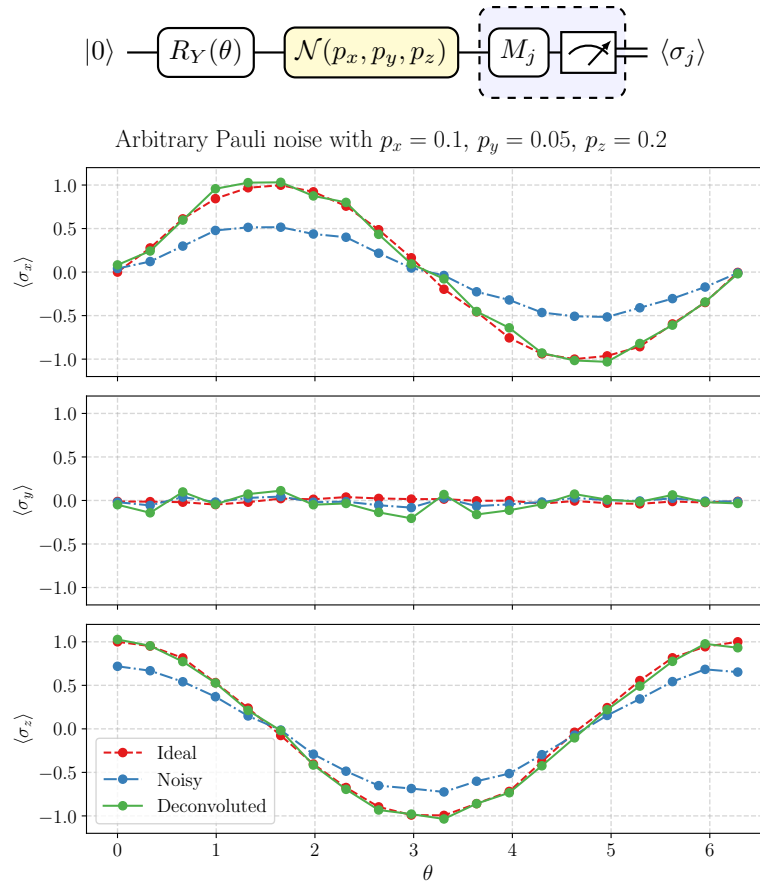


Figure 8.3: Simulation of the deconvolution process for the general Pauli channel $\mathcal{N}_{\mathbf{p}}$ (8.38). The noise parameters along the three Pauli axes are set to $p_x = 0.1$, $p_y = 0.05$, $p_z = 0.2$. The results are obtained simulating the circuit portrayed on top of the image for $n_{\text{shots}} = 1024$ shots and for multiple values of the angle θ . Then, the deconvolution formulas (8.40) are used to retrieve the ideal noise-free result. It is clear that the deconvolution effectively mitigates the Pauli noise yielding a final result which is much closer to the ideal noise-free one, up to differences due to stochastic measurement outcomes. In particular, the estimation of σ_y is dominated by the statistical error, which is amplified by the correction factor $1/(1 - 2(p_x + p_z)) = 2.5$.



9. Conclusions

The greatest success of our field [quantum computation and informatics] will not be to speed up calculations or communicate in secrecy, but to help people understand that this world is quantum-mechanical.

Charles Henry Bennett
as reported by Simone Severini in his book “Nella terra dei qubit” [272].

In this Thesis we have covered several topics regarding variational quantum algorithms and quantum machine learning, providing several compelling examples of how parameterized quantum circuits can be understood as machine learning models.

Indeed, we have started in Chapter 2 with a bird’s eye view on the new and exciting field of variational quantum algorithms, that is that ensemble of procedures that leverage parameterized quantum circuits and classical computing power in tandem to take full advantage of near-term quantum computing devices. Before future experimental and theoretical advancements pave the way toward the construction of large-scale noise-resilient quantum computers, near-term devices (NISQ) allow experimenting with quantum information processing, with an eye also on the possibility of achieving some sort of useful quantum advantage already with this new paradigm of hybrid quantum-classical computation.

The analysis of variational quantum algorithms then culminated in Chapter 3 where the field of Quantum Machine Learning was thoroughly characterised. Importantly, we have found how parameterized quantum models can be effectively described using tools from classical machine learning, for example when discussing kernel methods, expressing the output of data-dependent quantum circuits as truncated Fourier series, and last but not least how the classical statistical learning framework can be applied to derive statements about the generalisation performances of quantum neural networks.

The discussion then moved on to some concrete examples of quantum learning models, presenting some novel contributions to the field. In Chapter 4 we have reported on a novel quantum algorithm implementing a generalised perceptron model on a qubit-based quantum device that accepts and analyses continuously valued input data. The proposed algorithm can be readily run on existing quantum hardware, and it takes full advantage of the exponentially large Hilbert space available to encode input data on the phases of large superposition states, known as locally

maximally entanglable (LME) states. In addition, we saw how the proposed model can be used to implement classification tasks and pattern recognition involving grey-scale images.

In Chapter 5 we reviewed a discrete version of the continuous quantum neuron discussed in Ch. 4 and we introduced variational training methods for efficiently handling the manipulation of classical and quantum input data. Through extensive numerical analysis, we compared the effectiveness of different circuit structures and learning strategies, highlighting potential benefits brought by hardware-compatible entangling operations and by layerwise training routines that use local, instead of global, cost functions. In all envisioned applications, our proposed protocols are intended as an effective method for the analysis of quantum states as provided, e.g., by external devices or sensors, while it is worth stressing that the general problem of efficiently loading classical data into quantum registers still stands open.

Building on the phase encoding strategy introduced for the continuous quantum neuron, in Chapter 6 we have presented a toy model for a quantum pipeline comprising a quantum autoencoder and a classifier for analysing data coming from an industrial power plant. Specifically, we have implemented a variational quantum autoencoder that can compress information stored on a multipartite quantum state onto just some of its constituents. The compressed quantum states coming from the trained quantum autoencoder were then used as inputs to a quantum classifier to perform a binary classification task. For both tasks, the quantum procedures performed equally well to comparable classical counterparts, and they were also tested on real superconducting quantum hardware provided by IBM. While the achievement of a clear quantum advantage is still out of reach, this approach sets a milestone in the field of quantum machine learning, since it is one of the first examples of a direct application of quantum computing software and hardware to analyse real data sets from industrial sources.

Finally, in the last Chapters, we broadened our analysis to include more quantum information-related topics, discussing entanglement and noise. Specifically, in Chapter 7 we discussed in detail the Entanglement generated by different promising Quantum Neural Networks (QNNs) when these are initialised with random parameters, and showed that they reproduce the same properties of Haar random quantum states under various measures. Employing tensor network methods (MPS) we could simulate wide quantum circuits of up to 50 qubits, and introduced a new measure, the entangling speed, to characterise the rate of production of entanglement of a given circuit ansatz as its depth is increased.

At last, Chapter 8 moved our attention to discussing the impact of quantum noise on the estimation of expectation values of observables. Indeed, we have shown how mathematically invertible noise maps can always be removed from the final measurement stage so that one can obtain unbiased expectation values of general observables provided that the noise process is known. We illustrated the method on most known qubit noise maps, systematically derived their inverse maps, and also provided simulation and experimental application of the method on actual superconducting quantum hardware provided by Rigetti.

Quantum computing research is currently experiencing a surge of interest, with a significant amount of effort devoted to not only studying the long-term goal of universal fault-tolerant quantum devices, but also maximising the potential of current-generation quantum computers, both scientifically and technologically. In this thesis, we provided an extensive description of the state-of-the-art of variational quantum algorithms and machine learning, as well as several compelling original contributions to the field, some of which are more applied and others more fundamental.

Although the scientific value of near-term quantum computers is undeniable, the field is still too immature to confidently assess when and how a useful quantum advantage will be achieved. This is especially true when it comes to Quantum Machine Learning, which is the convergence of quantum physics and computing with artificial intelligence and deep learning, two notoriously complex and theoretically difficult fields.

Ultimately, only time will tell whether we will discover a useful and indisputable computational

advantage from quantum computing and/or quantum machine learning. In the meantime, we have the privilege of witnessing the evolution of this exciting field, and we should enjoy the journey as much as the destination.

References

Bibliography	164
---------------------------	------------

Bibliography

- [1] Scott Aaronson. “Read the fine print”. In: *Nat. Phys.* 11.4 (Apr. 2015), pages 291–293. ISSN: 1745-2481. DOI: <https://doi.org/10.1038/nphys3272> (cited on page 53).
- [2] Martin Abadi et al. “TensorFlow: A system for large-scale machine learning”. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pages 265–283. URL: <https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf> (cited on pages 115, 116).
- [3] Amira Abbas. *amyami187/effective_dimension*: version v1.0.1. May 2021. DOI: [10.5281/zenodo.4732856](https://doi.org/10.5281/zenodo.4732856). URL: <https://doi.org/10.5281/zenodo.4732856> (cited on pages 208, 210).
- [4] Amira Abbas et al. “The Power of Quantum Neural Networks”. In: *Nature Computational Science* 1.6 (June 2021), pages 403–409. ISSN: 2662-8457. DOI: [10.1038/s43588-021-00084-1](https://doi.org/10.1038/s43588-021-00084-1) (cited on pages 53, 79, 113, 121, 124, 125, 129, 130, 132, 138, 208, 210, 211).
- [5] Gadi Aleksandrowicz et al. *Qiskit: An Open-source Framework for Quantum Computing*. en. 2019. DOI: [10.5281/ZENODO.2562111](https://doi.org/10.5281/ZENODO.2562111). URL: <https://zenodo.org/record/2562111> (cited on pages 88, 93, 101, 102, 116).
- [6] *Amazon Braket*. <https://aws.amazon.com/braket/quantum-computers/>, 2023 (cited on page 34).
- [7] Mohammad H. Amin et al. “Quantum Boltzmann Machine”. In: *Phys. Rev. X* 8 (2 May 2018), page 021050 (cited on page 81).
- [8] Abhinav Anand et al. “A Quantum Computing View on Unitary Coupled Cluster Theory”. In: (2021). DOI: [10.48550/ARXIV.2109.15176](https://doi.org/10.48550/ARXIV.2109.15176). URL: <https://arxiv.org/abs/2109.15176> (cited on page 39).
- [9] Eric R. Anschuetz and Bobak T. Kiani. “Quantum variational algorithms are swamped with traps”. In: *Nature Communications* 13.1 (Dec. 2022). DOI: [10.1038/s41467-022-35364-5](https://doi.org/10.1038/s41467-022-35364-5). URL: <https://doi.org/10.1038/s41467-022-35364-5> (cited on pages 140, 207).
- [10] *Aria Quantum Processor, IonQ*. <https://ionq.com/posts/august-02-2022-ionq-aria-part-two-past-and-future>. Accessed: 05-01-2023 (cited on page 34).
- [11] Andrew Arrasmith et al. “Effect of barren plateaus on gradient-free optimization”. In: *Quantum* 5 (2021), page 558. DOI: <https://doi.org/10.22331/q-2021-10-05-558> (cited on page 45).
- [12] Andrew Arrasmith et al. “Effect of barren plateaus on gradient-free optimization”. In: *Quantum* 5 (Oct. 2021), page 558. ISSN: 2521-327X. DOI: [10.22331/q-2021-10-05-558](https://doi.org/10.22331/q-2021-10-05-558). URL: <https://doi.org/10.22331/q-2021-10-05-558> (cited on page 50).
- [13] Andrew Arrasmith et al. “Equivalence of quantum barren plateaus to cost concentration and narrow gorges”. In: (2021). arXiv: [2104.05868](https://arxiv.org/abs/2104.05868) (cited on pages 44, 50, 129).
- [14] Juan Miguel Arrazola et al. “Quantum-inspired algorithms in practice”. In: *Quantum* 4 (Aug. 2020), page 307. ISSN: 2521-327X. DOI: <https://doi.org/10.22331/q-2020-08-13-307> (cited on page 53).
- [15] Srinivasan Arunachalam and Ronald de Wolf. *A Survey of Quantum Learning Theory*. 2017. DOI: [10.48550/ARXIV.1701.06806](https://doi.org/10.48550/ARXIV.1701.06806). URL: <https://arxiv.org/abs/1701.06806> (cited on page 53).

- [16] Frank Arute et al. “Quantum supremacy using a programmable superconducting processor”. In: *Nature* 574.7779 (2019), pages 505–510. DOI: <https://doi.org/10.1038/s41586-019-1666-5> (cited on pages 33, 34, 81).
- [17] Erik Aurell, Jakub Zakrzewski, and Karol Życzkowski. “Time reversals of irreversible quantum maps”. In: *Journal of Physics A: Mathematical and Theoretical* 48.38 (Aug. 2015), 38FT01. DOI: [10.1088/1751-8113/48/38/38ft01](https://doi.org/10.1088/1751-8113/48/38/38ft01). URL: <https://doi.org/10.1088/1751-8113/48/38/38ft01> (cited on page 143).
- [18] Marco Ballarín et al. *Entanglement entropy production in Quantum Neural Networks*. Accepted for publication on *Quantum*. 2022. DOI: [10.48550/ARXIV.2206.02474](https://arxiv.org/abs/2206.02474). URL: <https://arxiv.org/abs/2206.02474> (cited on pages 7, 8, 38, 123).
- [19] Leonardo Banchi, Jason Pereira, and Stefano Pirandola. “Generalization in Quantum Machine Learning: A Quantum Information Standpoint”. In: *PRX Quantum* 2 (4 Nov. 2021), page 040321. DOI: [10.1103/PRXQuantum.2.040321](https://link.aps.org/doi/10.1103/PRXQuantum.2.040321). URL: <https://link.aps.org/doi/10.1103/PRXQuantum.2.040321> (cited on pages 42, 79).
- [20] Adriano Barenco et al. “Elementary gates for quantum computation”. In: *Phys. Rev. A* 52 (5 Nov. 1995), pages 3457–3467. DOI: [10.1103/PhysRevA.52.3457](https://link.aps.org/doi/10.1103/PhysRevA.52.3457). URL: <https://link.aps.org/doi/10.1103/PhysRevA.52.3457> (cited on page 27).
- [21] Kerstin Beer et al. “Training deep quantum neural networks”. In: *Nat. Commun.* 11.1 (2020), page 808. DOI: <https://doi.org/10.1038/s41467-020-14454-2> (cited on page 55).
- [22] Mikhail Belkin. *Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation*. 2021. DOI: [10.48550/ARXIV.2105.14368](https://arxiv.org/abs/2105.14368). URL: <https://arxiv.org/abs/2105.14368> (cited on page 62).
- [23] Adi Ben-Israel and Thomas N. E. Greville. *Generalized inverses : theory and applications / Adi Ben-Israel ; Thomas N.E. Greville*. eng. 2. ed. CMS books in mathematics. New York: Springer, 2003. ISBN: 0387002936 (cited on pages 62, 63).
- [24] F. Benatti, S. Mancini, and S. Mangini. “Continuous variable quantum perceptron”. In: *International Journal of Quantum Information* 17.08 (2019), page 1941009. DOI: [10.1142/S0219749919410090](https://doi.org/10.1142/S0219749919410090). eprint: <https://doi.org/10.1142/S0219749919410090>. URL: <https://doi.org/10.1142/S0219749919410090> (cited on page 7).
- [25] Marcello Benedetti et al. “Parameterized quantum circuits as machine learning models”. In: *Quantum Sci. Technol.* 5.1 (2020), page 019601. DOI: <https://doi.org/10.1088/2058-9565/ab4eb5> (cited on pages 53, 54).
- [26] Cédric Bény. “Quantum Deconvolution”. In: *Quantum Information Processing* 17.2 (Dec. 2017), page 26. ISSN: 1573-1332. DOI: [10.1007/s11128-017-1796-3](https://doi.org/10.1007/s11128-017-1796-3) (cited on page 144).
- [27] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2018. DOI: [10.48550/ARXIV.1811.04968](https://arxiv.org/abs/1811.04968). URL: <https://arxiv.org/abs/1811.04968> (cited on pages 116, 210).
- [28] Mary Beth Ruskai, Stanislaw Szarek, and Elisabeth Werner. “An Analysis of Completely-Positive Trace-Preserving Maps on M_2 ”. In: *Linear Algebra and its Applications* 347.1 (May 2002), pages 159–187. ISSN: 0024-3795. DOI: [10.1016/S0024-3795\(01\)00547-X](https://doi.org/10.1016/S0024-3795(01)00547-X) (cited on pages 146, 213).
- [29] Kishor Bharti et al. “Noisy intermediate-scale quantum algorithms”. In: *Rev. Mod. Phys.* 94 (1 Feb. 2022), page 015004. DOI: [10.1103/RevModPhys.94.015004](https://link.aps.org/doi/10.1103/RevModPhys.94.015004). URL: <https://link.aps.org/doi/10.1103/RevModPhys.94.015004> (cited on pages 22, 32, 35, 39, 43, 115).

- [30] R. Bhatia. *Matrix Analysis*. Graduate Texts in Mathematics. Springer New York, 1996. ISBN: 9780387948461 (cited on pages 77, 193).
- [31] Jacob Biamonte et al. “Quantum machine learning”. In: *Nature* 549.7671 (Sept. 2017), pages 195–202. ISSN: 1476-4687. DOI: <https://doi.org/10.1038/nature23474> (cited on pages 20, 53, 96).
- [32] M. Bilkis et al. *A semi-agnostic ansatz with variable structure for quantum machine learning*. 2021. DOI: [10.48550/ARXIV.2103.06712](https://arxiv.org/abs/2103.06712). URL: <https://arxiv.org/abs/2103.06712> (cited on page 39).
- [33] A. Bisio et al. “Optimal Quantum Tomography”. In: *IEEE Journal of Selected Topics in Quantum Electronics* 15.6 (Dec. 2009), pages 1646–1660. ISSN: 1558-4542. DOI: [10.1109/JSTQE.2009.2029243](https://doi.org/10.1109/JSTQE.2009.2029243) (cited on pages 146, 151).
- [34] Anselm Blumer et al. “Learnability and the Vapnik-Chervonenkis Dimension”. In: *J. ACM* 36.4 (Oct. 1989), pages 929–965. ISSN: 0004-5411. DOI: [10.1145/76359.76371](https://doi.org/10.1145/76359.76371). URL: <https://doi.org/10.1145/76359.76371> (cited on page 60).
- [35] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, Feb. 2013. ISBN: 9780199535255. DOI: [10.1093/acprof:oso/9780199535255.001.0001](https://doi.org/10.1093/acprof:oso/9780199535255.001.0001). URL: <https://doi.org/10.1093/acprof:oso/9780199535255.001.0001> (cited on page 60).
- [36] P. S. Bourdon and H. T. Williams. “Unital Quantum Operations on the Bloch Ball and Bloch Region”. In: *Physical Review A* 69.2 (Feb. 2004), page 022314. DOI: [10.1103/PhysRevA.69.022314](https://doi.org/10.1103/PhysRevA.69.022314) (cited on pages 145, 146, 150).
- [37] Fernando G. S. L. Brandão, Aram W. Harrow, and Michał Horodecki. “Local Random Quantum Circuits Are Approximate Polynomial-Designs”. In: *Communications in Mathematical Physics* 346.2 (Sept. 2016), pages 397–434. ISSN: 1432-0916. DOI: [10.1007/s00220-016-2706-8](https://doi.org/10.1007/s00220-016-2706-8) (cited on pages 133, 139).
- [38] Carlos Bravo-Prieto. “Quantum autoencoders with enhanced data encoding”. In: *Machine Learning: Science and Technology* 2.3 (July 2021), page 035028. DOI: [10.1088/2632-2153/ac0616](https://doi.org/10.1088/2632-2153/ac0616). URL: <https://dx.doi.org/10.1088/2632-2153/ac0616> (cited on pages 108, 109, 113).
- [39] Sergey Bravyi et al. “Mitigating measurement errors in multiqubit experiments”. In: *Phys. Rev. A* 103 (4 Apr. 2021), page 042605. DOI: [10.1103/PhysRevA.103.042605](https://doi.org/10.1103/PhysRevA.103.042605). URL: <https://link.aps.org/doi/10.1103/PhysRevA.103.042605> (cited on page 143).
- [40] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. DOI: [10.48550/ARXIV.2104.13478](https://arxiv.org/abs/2104.13478). URL: <https://arxiv.org/abs/2104.13478> (cited on page 39).
- [41] Michael Broughton et al. “Tensorflow quantum: A software framework for quantum machine learning”. In: *arXiv preprint arXiv:2003.02989* (2020) (cited on page 93).
- [42] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. DOI: [10.48550/ARXIV.2005.14165](https://arxiv.org/abs/2005.14165). URL: <https://arxiv.org/abs/2005.14165> (cited on page 56).
- [43] Colin D. Bruzewicz et al. “Trapped-ion quantum computing: Progress and challenges”. In: *Applied Physics Reviews* 6.2 (2019), page 021314. DOI: [10.1063/1.5088164](https://doi.org/10.1063/1.5088164). eprint: <https://doi.org/10.1063/1.5088164>. URL: <https://doi.org/10.1063/1.5088164> (cited on page 35).
- [44] H. Buhrman et al. “Quantum Fingerprinting”. en. In: *Physical Review Letters* 87.16 (Sept. 2001), page 167902. ISSN: 0031-9007, 1079-7114. DOI: [10.1103/PhysRevLett.87.167902](https://doi.org/10.1103/PhysRevLett.87.167902). (Visited on 08/19/2019) (cited on page 106).

- [45] C. King and M. B. Ruskai. “Minimal Entropy of States Emerging from Noisy Quantum Channels”. In: *IEEE Transactions on Information Theory* 47.1 (Jan. 2001), pages 192–209. ISSN: 1557-9654. DOI: [10.1109/18.904522](https://doi.org/10.1109/18.904522) (cited on pages 146, 213).
- [46] Carlo Cafaro and Peter van Loock. “Approximate quantum error correction for generalized amplitude-damping errors”. In: *Phys. Rev. A* 89 (2 Feb. 2014), page 022316. DOI: [10.1103/PhysRevA.89.022316](https://doi.org/10.1103/PhysRevA.89.022316). URL: <https://link.aps.org/doi/10.1103/PhysRevA.89.022316> (cited on page 153).
- [47] Carlo Cafaro and Stefano Mancini. “Quantum stabilizer codes for correlated and asymmetric depolarizing errors”. In: *Phys. Rev. A* 82 (1 July 2010), page 012306. DOI: [10.1103/PhysRevA.82.012306](https://doi.org/10.1103/PhysRevA.82.012306). URL: <https://link.aps.org/doi/10.1103/PhysRevA.82.012306> (cited on page 151).
- [48] Pasquale Calabrese and John Cardy. “Evolution of entanglement entropy in one-dimensional systems”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.04 (Apr. 2005), P04010. DOI: [10.1088/1742-5468/2005/04/p04010](https://doi.org/10.1088/1742-5468/2005/04/p04010). URL: <https://doi.org/10.1088/1742-5468/2005/04/p04010> (cited on page 136).
- [49] Ningping Cao et al. *NISQ: Error Correction, Mitigation, and Noise Simulation*. 2021. arXiv: [2111.02345](https://arxiv.org/abs/2111.02345) [quant-ph] (cited on pages 142, 147).
- [50] Yudong Cao, Gian Giacomo Guerreschi, and Alán Aspuru-Guzik. *Quantum Neuron: an elementary building block for machine learning on quantum computers*. 2017. DOI: [10.48550/ARXIV.1711.11240](https://doi.org/10.48550/ARXIV.1711.11240). URL: <https://arxiv.org/abs/1711.11240> (cited on pages 81, 96).
- [51] Giuseppe Carleo and Matthias Troyer. “Solving the quantum many-body problem with artificial neural networks”. In: *Science* 355.6325 (2017), pages 602–606 (cited on page 54).
- [52] Giuseppe Carleo et al. “Machine learning and the physical sciences”. In: *Rev. Mod. Phys.* 91 (4 Dec. 2019), page 045002. DOI: [10.1103/RevModPhys.91.045002](https://doi.org/10.1103/RevModPhys.91.045002). URL: <https://link.aps.org/doi/10.1103/RevModPhys.91.045002> (cited on pages 53, 54).
- [53] Matthias C. Caro et al. “Encoding-dependent generalization bounds for parametrized quantum circuits”. In: *Quantum* 5 (Nov. 2021), page 582. ISSN: 2521-327X. DOI: [10.22331/q-2021-11-17-582](https://doi.org/10.22331/q-2021-11-17-582). URL: <https://doi.org/10.22331/q-2021-11-17-582> (cited on pages 60, 72, 73, 78, 79, 198).
- [54] J. Carolan et al. “Variational quantum unsampling on a quantum photonic processor”. en. In: *Nature Physics* 16 (Jan. 2020), page 322. ISSN: 1745-2473, 1745-2481. (Visited on 01/16/2020) (cited on pages 98, 99).
- [55] Juan Carrasquilla. “Machine learning for quantum matter”. In: *Advances in Physics: X* 5.1 (2020), page 1797528. DOI: [10.1080/23746149.2020.1797528](https://doi.org/10.1080/23746149.2020.1797528). eprint: <https://doi.org/10.1080/23746149.2020.1797528>. URL: <https://doi.org/10.1080/23746149.2020.1797528> (cited on page 54).
- [56] M Cerezo and Patrick J Coles. “Higher order derivatives of quantum neural networks with barren plateaus”. In: *Quantum Science and Technology* 6.3 (June 2021), page 035006. DOI: [10.1088/2058-9565/abf51a](https://doi.org/10.1088/2058-9565/abf51a). URL: <https://doi.org/10.1088/2058-9565/abf51a> (cited on pages 41, 42).
- [57] M. Cerezo et al. “Cost function dependent barren plateaus in shallow parametrized quantum circuits”. In: *Nat. Commun.* 12.1 (2021). DOI: <https://doi.org/10.1038/s41467-021-21728-w> (cited on pages 39, 40, 43–46, 50, 105, 109, 129, 139, 140, 190, 191, 207).
- [58] M. Cerezo et al. “Variational quantum algorithms”. In: *Nature Reviews Physics* 3.9 (2021), pages 625–644. DOI: <https://doi.org/10.1038/s42254-021-00348-9> (cited on pages 22, 37, 39, 115).

- [59] M. Cerezo et al. “Challenges and Opportunities in Quantum Machine Learning”. In: *Nature Computational Science* 2.9 (Sept. 2022), pages 567–576. ISSN: 2662-8457. DOI: [10.1038/s43588-022-00311-3](https://doi.org/10.1038/s43588-022-00311-3) (cited on pages 22, 53, 55, 79).
- [60] *ChatGPT, OpenAI*. <https://openai.com/blog/chatgpt/>. Accessed: 13-01-2023 (cited on pages 20, 56, 67).
- [61] Yanzhu Chen et al. *How Much Entanglement Do Quantum Optimization Algorithms Require?* 2022. DOI: [10.48550/ARXIV.2205.12283](https://doi.org/10.48550/ARXIV.2205.12283). URL: <https://arxiv.org/abs/2205.12283> (cited on page 207).
- [62] Yiwei Chen et al. “Detecting quantum entanglement with unsupervised learning”. In: *Quantum Science and Technology* 7.1 (Nov. 2021), page 015005. DOI: [10.1088/2058-9565/ac310f](https://doi.org/10.1088/2058-9565/ac310f). URL: <https://doi.org/10.1088/2058-9565/ac310f> (cited on page 109).
- [63] Carlo Ciliberto et al. “Quantum machine learning: a classical perspective”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2209 (Jan. 2018), page 20170551. DOI: [10.1098/rspa.2017.0551](https://doi.org/10.1098/rspa.2017.0551). URL: <https://doi.org/10.1098/rspa.2017.0551> (cited on page 53).
- [64] Lukasz Cincio et al. “Learning the quantum algorithm for state overlap”. In: *New Journal of Physics* 20.11 (Nov. 2018), page 113022. DOI: [10.1088/1367-2630/aae94a](https://doi.org/10.1088/1367-2630/aae94a). URL: <https://doi.org/10.1088/1367-2630/aae94a> (cited on pages 118, 201).
- [65] Bob Coecke et al. *Foundations for Near-Term Quantum Natural Language Processing*. 2020. DOI: [10.48550/ARXIV.2012.03755](https://doi.org/10.48550/ARXIV.2012.03755). URL: <https://arxiv.org/abs/2012.03755> (cited on page 55).
- [66] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. “Quantum convolutional neural networks”. In: *Nature Physics* 15.12 (2019), pages 1273–1278. DOI: <https://doi.org/10.1038/s41567-019-0648-8> (cited on pages 50, 55, 81).
- [67] Jordan Cotler, Hsin-Yuan Huang, and Jarrod R. McClean. *Revisiting dequantization and quantum advantage in learning tasks*. 2021. DOI: [10.48550/ARXIV.2112.00811](https://doi.org/10.48550/ARXIV.2112.00811). URL: <https://arxiv.org/abs/2112.00811> (cited on page 53).
- [68] Gavin E. Crooks. *Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition*. 2019. DOI: [10.48550/ARXIV.1905.13311](https://doi.org/10.48550/ARXIV.1905.13311). URL: <https://arxiv.org/abs/1905.13311> (cited on page 42).
- [69] Andrew W. Cross et al. “Validating quantum computers using randomized model circuits”. In: *Phys. Rev. A* 100 (3 Sept. 2019), page 032328. DOI: [10.1103/PhysRevA.100.032328](https://doi.org/10.1103/PhysRevA.100.032328). URL: <https://link.aps.org/doi/10.1103/PhysRevA.100.032328> (cited on page 35).
- [70] G M D Ariano, L Maccone, and M Painsi. “Spin tomography”. In: *Journal of Optics B: Quantum and Semiclassical Optics* 5.1 (Jan. 2003), pages 77–84. ISSN: 1464-4266. DOI: [10.1088/1464-4266/5/1/311](https://doi.org/10.1088/1464-4266/5/1/311) (cited on pages 146, 147, 213, 214).
- [71] *D-Wave Systems, Inc.* <https://www.dwavesys.com/>, 2023 (cited on page 34).
- [72] G. M. D’Ariano and P. Lo Presti. “Quantum Tomography for Measuring Experimentally the Matrix Elements of an Arbitrary Quantum Operation”. In: *Phys. Rev. Lett.* 86 (19 May 2001), pages 4195–4198. DOI: [10.1103/PhysRevLett.86.4195](https://doi.org/10.1103/PhysRevLett.86.4195). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.86.4195> (cited on page 146).
- [73] G.M. D’Ariano, L. Maccone, and M.G.A. Paris. “Orthogonality relations in quantum tomography”. In: *Physics Letters A* 276.1 (2000), pages 25–30. ISSN: 0375-9601. DOI: [https://doi.org/10.1016/S0375-9601\(00\)00660-5](https://doi.org/10.1016/S0375-9601(00)00660-5). URL: <https://www.sciencedirect.com/science/article/pii/S0375960100006605> (cited on page 146).

- [74] G.M. D’Ariano and M.F. Sacchi. “Renormalized quantum tomography”. In: *Physics Letters A* 374.5 (2010), pages 713–724. ISSN: 0375-9601. DOI: <https://doi.org/10.1016/j.physleta.2009.11.081>. URL: <https://www.sciencedirect.com/science/article/pii/S037596010901514X> (cited on page 151).
- [75] Giacomo Mauro D’Ariano, Giulio Chiribella, and Paolo Perinotti. *Quantum Theory from First Principles: An Informational Approach*. Cambridge University Press, 2017. DOI: [10.1017/9781107338340](https://doi.org/10.1017/9781107338340) (cited on page 146).
- [76] Giacomo Mauro D’Ariano and Paoloplacido Lo Presti. “Imprinting Complete Information about a Quantum Channel on its Output State”. In: *Phys. Rev. Lett.* 91 (4 July 2003), page 047902. DOI: [10.1103/PhysRevLett.91.047902](https://doi.org/10.1103/PhysRevLett.91.047902). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.91.047902> (cited on page 148).
- [77] Giacomo Mauro D’Ariano, Lorenzo Maccone, and Paoloplacido Lo Presti. “Quantum Calibration of Measurement Instrumentation”. In: *Phys. Rev. Lett.* 93 (25 Dec. 2004), page 250407. DOI: [10.1103/PhysRevLett.93.250407](https://doi.org/10.1103/PhysRevLett.93.250407). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.93.250407> (cited on page 148).
- [78] G Mauro D’Ariano, Lorenzo Maccone, and Matteo G A Paris. “Quorum of observables for universal quantum estimation”. In: *Journal of Physics A: Mathematical and General* 34.1 (Jan. 2001), pages 93–103. ISSN: 0305-4470, 1361-6447. DOI: [10.1088/0305-4470/34/1/307](https://doi.org/10.1088/0305-4470/34/1/307) (cited on page 146).
- [79] Giacomo Mauro D’Ariano. “Universal quantum estimation”. In: *Physics Letters A* 268.3 (Apr. 2000), pages 151–157. ISSN: 03759601. DOI: [10.1016/S0375-9601\(00\)00164-X](https://doi.org/10.1016/S0375-9601(00)00164-X) (cited on pages 146, 147, 151).
- [80] Christoph Dankert et al. “Exact and Approximate Unitary 2-Designs and Their Application to Fidelity Estimation”. In: *Physical Review A* 80.1 (July 2009), page 012304. ISSN: 1050-2947, 1094-1622. DOI: [10.1103/PhysRevA.80.012304](https://doi.org/10.1103/PhysRevA.80.012304) (cited on page 47).
- [81] Christoph Dankert et al. “Exact and approximate unitary 2-designs and their application to fidelity estimation”. In: *Physical Review A* 80.1 (July 2009). DOI: [10.1103/physreva.80.012304](https://doi.org/10.1103/physreva.80.012304). URL: <https://doi.org/10.1103/physreva.80.012304> (cited on pages 129, 139).
- [82] Anna Dawid et al. *Modern applications of machine learning in quantum sciences*. 2022. DOI: [10.48550/ARXIV.2204.04198](https://doi.org/10.48550/ARXIV.2204.04198). URL: <https://arxiv.org/abs/2204.04198> (cited on pages 53, 54, 62–65, 67).
- [83] Jonas Degraeve et al. “Magnetic Control of Tokamak Plasmas through Deep Reinforcement Learning”. In: *Nature* 602.7897 (Feb. 2022), pages 414–419. ISSN: 1476-4687. DOI: [10.1038/s41586-021-04301-9](https://doi.org/10.1038/s41586-021-04301-9) (cited on pages 20, 36, 53).
- [84] Riccardo Di Sipio et al. “The Dawn of Quantum Natural Language Processing”. In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pages 8612–8616. DOI: [10.1109/ICASSP43922.2022.9747675](https://doi.org/10.1109/ICASSP43922.2022.9747675) (cited on page 7).
- [85] Carl Doersch and Andrew Zisserman. *Multi-task Self-Supervised Visual Learning*. 2017. DOI: [10.48550/ARXIV.1708.07860](https://doi.org/10.48550/ARXIV.1708.07860). URL: <https://arxiv.org/abs/1708.07860> (cited on page 56).
- [86] Yuxuan Du et al. “Efficient Measure for the Expressivity of Variational Quantum Algorithms”. In: *Physical Review Letters* 128.8 (Feb. 2022). DOI: [10.1103/physrevlett.128.080506](https://doi.org/10.1103/physrevlett.128.080506). URL: <https://doi.org/10.1103/physrevlett.128.080506> (cited on page 79).

- [87] Vedran Dunjko and Hans J Briegel. “Machine learning & artificial intelligence in the quantum domain: a review of recent progress”. In: *Rep. Prog. Phys.* 81.7 (2018), page 074001. DOI: <https://doi.org/10.1088/1361-6633/aab406> (cited on page 53).
- [88] Vedran Dunjko and Peter Wittek. “A non-review of Quantum Machine Learning: trends and explorations”. In: *Quantum* 4 (2020), page 32. DOI: <https://doi.org/10.22331/qv-2020-03-17-32> (cited on pages 20, 53).
- [89] Maxime Dupont et al. “Calibrating the Classical Hardness of the Quantum Approximate Optimization Algorithm”. In: *PRX Quantum* 3 (4 Dec. 2022), page 040339. DOI: [10.1103/PRXQuantum.3.040339](https://doi.org/10.1103/PRXQuantum.3.040339). URL: <https://link.aps.org/doi/10.1103/PRXQuantum.3.040339> (cited on page 207).
- [90] Maxime Dupont et al. “Entanglement perspective on the quantum approximate optimization algorithm”. In: *Phys. Rev. A* 106 (2 Aug. 2022), page 022423. DOI: [10.1103/PhysRevA.106.022423](https://doi.org/10.1103/PhysRevA.106.022423). URL: <https://link.aps.org/doi/10.1103/PhysRevA.106.022423> (cited on pages 140, 207).
- [91] F.L. Dyer and T.C. Martin. *Edison: His Life and Inventions*. Edison: His Life and Inventions v. 2. Harper & Brothers, 1910. URL: <https://books.google.it/books?id=B7A4AAAAAAAJ> (cited on page 52).
- [92] *Eagle Quantum Processor, IBM Quantum*. <https://research.ibm.com/blog/127-qubit-quantum-processor-eagle>. Accessed: 05-01-2023 (cited on page 34).
- [93] Alan Edelman and N. Raj Rao. “Random Matrix Theory”. In: *Acta Numerica* 14 (May 2005), pages 233–297. ISSN: 0962-4929, 1474-0508. DOI: [10.1017/S0962492904000236](https://doi.org/10.1017/S0962492904000236) (cited on page 127).
- [94] Daniel J. Egger et al. “Quantum Computing for Finance: State-of-the-Art and Future Prospects”. In: *IEEE Transactions on Quantum Engineering* 1 (2020), pages 1–24. DOI: [10.1109/TQE.2020.3030314](https://doi.org/10.1109/TQE.2020.3030314) (cited on page 55).
- [95] J. Eisert. *Entanglement and tensor network states*. 2013. arXiv: [1308.3318](https://arxiv.org/abs/1308.3318) [quant-ph]. URL: <https://arxiv.org/abs/1308.3318> (cited on pages 125, 126).
- [96] Joseph Emerson, Robert Alicki, and Karol Życzkowski. “Scalable noise estimation with random unitary operators”. In: *J. Opt. B: Quantum Semiclass. Opt.* 7.10 (Sept. 2005), S347–S352. DOI: [10.1088/1464-4266/7/10/021](https://doi.org/10.1088/1464-4266/7/10/021). URL: <https://doi.org/10.1088/1464-4266/7/10/021> (cited on page 149).
- [97] Suguru Endo, Simon C. Benjamin, and Ying Li. “Practical Quantum Error Mitigation for Near-Future Applications”. In: *Phys. Rev. X* 8 (3 July 2018), page 031027. DOI: [10.1103/PhysRevX.8.031027](https://doi.org/10.1103/PhysRevX.8.031027). URL: <https://link.aps.org/doi/10.1103/PhysRevX.8.031027> (cited on pages 143, 144).
- [98] Jesper E. van Engelen and Holger H. Hoos. “A survey on semi-supervised learning”. In: *Machine Learning* 109.2 (Nov. 2019), pages 373–440. DOI: [10.1007/s10994-019-05855-6](https://doi.org/10.1007/s10994-019-05855-6). URL: <https://doi.org/10.1007/s10994-019-05855-6> (cited on page 56).
- [99] Nic Ezzell et al. *Quantum Mixed State Compiling*. 2022. DOI: [10.48550/ARXIV.2209.00528](https://doi.org/10.48550/ARXIV.2209.00528). URL: <https://arxiv.org/abs/2209.00528> (cited on page 55).
- [100] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. DOI: [10.48550/ARXIV.1411.4028](https://doi.org/10.48550/ARXIV.1411.4028). URL: <https://arxiv.org/abs/1411.4028> (cited on pages 39, 207).
- [101] Edward Farhi and Hartmut Neven. *Classification with Quantum Neural Networks on Near Term Processors*. 2018. arXiv: [1802.06002](https://arxiv.org/abs/1802.06002) [quant-ph] (cited on pages 82, 93).

- [102] Richard P. Feynman. “Simulating physics with computers”. In: *International Journal of Theoretical Physics* 21.6-7 (June 1982), pages 467–488. DOI: [10.1007/bf02650179](https://doi.org/10.1007/bf02650179). URL: <https://doi.org/10.1007/bf02650179> (cited on page 19).
- [103] R.A. Fisher. *Iris*. UCI Machine Learning Repository. 1988 (cited on pages 208, 210).
- [104] Motohisa Fukuda, Robert König, and Ion Nechita. “RTNI—A symbolic integrator for Haar-random tensor networks”. In: *Journal of Physics A: Mathematical and Theoretical* 52.42 (Sept. 2019), page 425303. DOI: [10.1088/1751-8121/ab434b](https://doi.org/10.1088/1751-8121/ab434b). URL: <https://doi.org/10.1088/1751-8121/ab434b> (cited on page 46).
- [105] Juan Carlos Garcia-Escartin and Pedro Chamorro-Posada. *Equivalent Quantum Circuits*. 2011. arXiv: [1110.2998](https://arxiv.org/abs/1110.2998) [quant-ph] (cited on page 204).
- [106] Guillermo García-Pérez et al. “Learning to Measure: Adaptive Informationally Complete Generalized Measurements for Quantum Algorithms”. In: *PRX Quantum* 2 (4 Nov. 2021), page 040342. DOI: [10.1103/PRXQuantum.2.040342](https://doi.org/10.1103/PRXQuantum.2.040342). URL: <https://link.aps.org/doi/10.1103/PRXQuantum.2.040342> (cited on page 32).
- [107] M. Ghio et al. “Multipartite entanglement detection for hypergraph states”. en. In: *Journal of Physics A: Mathematical and Theoretical* 51.4 (Jan. 2018), page 045302. ISSN: 1751-8113, 1751-8121. DOI: [10.1088/1751-8121/aa99c9](https://doi.org/10.1088/1751-8121/aa99c9). (Visited on 08/22/2019) (cited on pages 97, 98).
- [108] Craig Gidney and Martin Ekerå. “How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits”. In: *Quantum* 5 (Apr. 2021), page 433. ISSN: 2521-327X. DOI: [10.22331/q-2021-04-15-433](https://doi.org/10.22331/q-2021-04-15-433). URL: <https://doi.org/10.22331/q-2021-04-15-433> (cited on page 35).
- [109] Francisco Javier Gil Vidal and Dirk Oliver Theis. “Input Redundancy for Parameterized Quantum Circuits”. In: *Front. Phys.* 8 (2020), page 297. ISSN: 2296-424X. DOI: <https://doi.org/10.3389/fphy.2020.00297> (cited on pages 72, 73, 113, 121, 128, 130, 210).
- [110] András Gilyén et al. “Quantum Algorithm for Petz Recovery Channels and Pretty Good Measurements”. In: *Phys. Rev. Lett.* 128 (22 June 2022), page 220502. DOI: [10.1103/PhysRevLett.128.220502](https://doi.org/10.1103/PhysRevLett.128.220502). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.128.220502> (cited on page 143).
- [111] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. “Quantum Random Access Memory”. In: *Phys. Rev. Lett.* 100 (16 Apr. 2008), page 160501. DOI: [10.1103/PhysRevLett.100.160501](https://doi.org/10.1103/PhysRevLett.100.160501). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.100.160501> (cited on pages 96, 200).
- [112] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016 (cited on pages 53, 62, 66, 67, 111, 112).
- [113] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016 (cited on pages 54, 61, 66).
- [114] *Google devices information*. <https://quantumai.google/cirq/google/devices>, 2022 (cited on page 33).
- [115] Edward Grant et al. “An initialization strategy for addressing barren plateaus in parametrized quantum circuits”. In: *Quantum* 3 (Dec. 2019), page 214. ISSN: 2521-327X. DOI: <https://doi.org/10.48550/arXiv.1903.05076> (cited on pages 50, 105, 130).
- [116] Aikaterini Gratsea and Patrick Huembeli. “Exploring quantum perceptron and quantum neural network structures with a teacher-student scheme”. In: *Quantum Machine Intelligence* 4.1 (Jan. 2022). DOI: [10.1007/s42484-021-00058-6](https://doi.org/10.1007/s42484-021-00058-6). URL: <https://doi.org/10.1007/s42484-021-00058-6> (cited on page 121).

- [117] Daniel Greenbaum. “Introduction to Quantum Gate Set Tomography”. In: *arXiv:1509.02921 [quant-ph]* (Sept. 2015). arXiv: [1509.02921](https://arxiv.org/abs/1509.02921) [quant-ph] (cited on page 146).
- [118] Harper R. Grimsley et al. “An Adaptive Variational Algorithm for Exact Molecular Simulations on a Quantum Computer”. In: *Nature Communications* 10.1 (July 2019), page 3007. ISSN: 2041-1723. DOI: [10.1038/s41467-019-10988-2](https://doi.org/10.1038/s41467-019-10988-2) (cited on page 39).
- [119] Lov Grover and Terry Rudolph. *Creating superpositions that correspond to efficiently integrable probability distributions*. 2002. DOI: [10.48550/ARXIV.QUANT-PH/0208112](https://arxiv.org/abs/quant-ph/0208112). URL: <https://arxiv.org/abs/quant-ph/0208112> (cited on page 201).
- [120] Sanjay Gupta and R.K.P. Zia. “Quantum Neural Networks”. In: *Journal of Computer and System Sciences* 63.3 (2001), pages 355–383. ISSN: 0022-0000. DOI: <https://doi.org/10.1006/jcss.2001.1769>. URL: <https://www.sciencedirect.com/science/article/pii/S0022000001917696> (cited on pages 20, 53).
- [121] Casper Gyurik, Dyon Vreumingen van, and Vedran Dunjko. “Structural risk minimization for quantum linear classifiers”. In: *Quantum* 7 (Jan. 2023), page 893. ISSN: 2521-327X. DOI: [10.22331/q-2023-01-13-893](https://doi.org/10.22331/q-2023-01-13-893). URL: <https://doi.org/10.22331/q-2023-01-13-893> (cited on page 79).
- [122] Stuart Hadfield et al. “From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz”. In: *Algorithms* 12.2 (2019). ISSN: 1999-4893. DOI: [10.3390/a12020034](https://www.mdpi.com/1999-4893/12/2/34). URL: <https://www.mdpi.com/1999-4893/12/2/34> (cited on page 39).
- [123] Jonas Haferkamp and Nicholas Hunter-Jones. “Improved spectral gaps for random quantum circuits: Large local dimensions and all-to-all interactions”. In: *Phys. Rev. A* 104 (2 Aug. 2021), page 022417. DOI: [10.1103/PhysRevA.104.022417](https://doi.org/10.1103/PhysRevA.104.022417). URL: <https://link.aps.org/doi/10.1103/PhysRevA.104.022417> (cited on pages 130, 139).
- [124] Aram Harrow and Saeed Mehraban. *Approximate unitary t -designs by short random quantum circuits using nearest-neighbor and long-range gates*. 2018. arXiv: [1809.06957](https://arxiv.org/abs/1809.06957) [quant-ph] (cited on pages 133, 139).
- [125] Aram W. Harrow, Avinandan Hassidim, and Seth Lloyd. “Quantum Algorithm for Linear Systems of Equations”. In: *Physical Review Letters* 103.15 (Oct. 2009). DOI: [10.1103/physrevlett.103.150502](https://doi.org/10.1103/physrevlett.103.150502). URL: <https://doi.org/10.1103/physrevlett.103.150502> (cited on page 53).
- [126] Aram W. Harrow and Richard A. Low. “Random Quantum Circuits Are Approximate 2-Designs”. In: *Communications in Mathematical Physics* 291.1 (Oct. 2009), pages 257–302. ISSN: 1432-0916. DOI: [10.1007/s00220-009-0873-6](https://doi.org/10.1007/s00220-009-0873-6) (cited on pages 130, 139).
- [127] Akel Hashim et al. “Randomized Compiling for Scalable Quantum Computing on a Noisy Superconducting Quantum Processor”. In: *Phys. Rev. X* 11 (4 Nov. 2021), page 041039. DOI: [10.1103/PhysRevX.11.041039](https://doi.org/10.1103/PhysRevX.11.041039). URL: <https://link.aps.org/doi/10.1103/PhysRevX.11.041039> (cited on page 149).
- [128] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics. Springer, 2009. ISBN: 978-0-387-84884-6 (cited on pages 61, 62, 66, 111, 112).
- [129] Trevor Hastie et al. “Surprises in high-dimensional ridgeless least squares interpolation”. In: *The Annals of Statistics* 50.2 (2022), pages 949–986. DOI: [10.1214/21-AOS2133](https://doi.org/10.1214/21-AOS2133). URL: <https://doi.org/10.1214/21-AOS2133> (cited on page 62).
- [130] Julian Havil. “Gamma: exploring Euler’s constant”. In: *The Australian Mathematical Society* (2003), page 250 (cited on page 203).

- [131] Vojtěch Havlíček et al. “Supervised learning with quantum-enhanced feature spaces”. In: *Nature* 567.7747 (Mar. 2019), pages 209–212. ISSN: 1476-4687. DOI: <https://doi.org/10.1038/s41586-019-0980-2> (cited on pages 70, 71, 81, 106, 130, 138).
- [132] Patrick Hayden, Debbie W. Leung, and Andreas Winter. “Aspects of Generic Entanglement”. In: *Communications in Mathematical Physics* 265.1 (July 2006), pages 95–117. ISSN: 1432-0916. DOI: [10.1007/s00220-006-1535-6](https://doi.org/10.1007/s00220-006-1535-6) (cited on pages 127, 202).
- [133] J. Helsen et al. “General Framework for Randomized Benchmarking”. In: *PRX Quantum* 3 (2 June 2022), page 020357. DOI: [10.1103/PRXQuantum.3.020357](https://doi.org/10.1103/PRXQuantum.3.020357). URL: <https://link.aps.org/doi/10.1103/PRXQuantum.3.020357> (cited on page 35).
- [134] Maxwell Henderson et al. “Quantum convolutional neural networks: powering image recognition with quantum circuits”. In: *Quantum Machine Intelligence* 2.1 (Feb. 2020). DOI: [10.1007/s42484-020-00012-y](https://doi.org/10.1007/s42484-020-00012-y). URL: <https://doi.org/10.1007/s42484-020-00012-y> (cited on page 81).
- [135] Loïc Henriot et al. “Quantum computing with neutral atoms”. In: *Quantum* 4 (Sept. 2020), page 327. ISSN: 2521-327X. DOI: [10.22331/q-2020-09-21-327](https://doi.org/10.22331/q-2020-09-21-327). URL: <https://doi.org/10.22331/q-2020-09-21-327> (cited on page 34).
- [136] Dylan Herman et al. *A Survey of Quantum Computing for Finance*. 2022. DOI: [10.48550/ARXIV.2201.02773](https://arxiv.org/abs/2201.02773). URL: <https://arxiv.org/abs/2201.02773> (cited on page 55).
- [137] Zoë Holmes et al. “Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus”. In: *PRX Quantum* 3 (1 2022), page 010313. DOI: <https://doi.org/10.1103/PRXQuantum.3.010313> (cited on pages 43, 46, 50, 51, 129, 139).
- [138] Hsin-Yuan Huang, Kishor Bharti, and Patrick Rebentrost. *Near-term quantum algorithms for linear systems of equations*. 2019. DOI: [10.48550/ARXIV.1909.07344](https://arxiv.org/abs/1909.07344). URL: <https://arxiv.org/abs/1909.07344> (cited on page 32).
- [139] Hsin-Yuan Huang, Richard Kueng, and John Preskill. “Predicting Many Properties of a Quantum System from Very Few Measurements”. In: *Nature Physics* 16.10 (Oct. 2020), pages 1050–1057. ISSN: 1745-2481. DOI: [10.1038/s41567-020-0932-7](https://doi.org/10.1038/s41567-020-0932-7) (cited on page 151).
- [140] Hsin-Yuan Huang, Richard Kueng, and John Preskill. “Information-Theoretic Bounds on Quantum Advantage in Machine Learning”. In: *Phys. Rev. Lett.* 126 (19 May 2021), page 190505. DOI: [10.1103/PhysRevLett.126.190505](https://doi.org/10.1103/PhysRevLett.126.190505). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.126.190505> (cited on pages 79, 121).
- [141] Hsin-Yuan Huang et al. “Power of Data in Quantum Machine Learning”. In: *Nature Communications* 12.1 (May 2021), page 2631. ISSN: 2041-1723. DOI: [10.1038/s41467-021-22539-9](https://doi.org/10.1038/s41467-021-22539-9) (cited on pages 46, 79, 121).
- [142] Hsin-Yuan Huang et al. “Provably efficient machine learning for quantum many-body problems”. In: *Science* 377.6613 (2022), eabk3333. DOI: [10.1126/science.abk3333](https://doi.org/10.1126/science.abk3333). eprint: <https://www.science.org/doi/pdf/10.1126/science.abk3333>. URL: <https://www.science.org/doi/abs/10.1126/science.abk3333> (cited on page 79).
- [143] Hsin-Yuan Huang et al. “Quantum advantage in learning from experiments”. In: *Science* 376.6598 (2022), pages 1182–1186. DOI: [10.1126/science.abn7293](https://doi.org/10.1126/science.abn7293). eprint: <https://www.science.org/doi/pdf/10.1126/science.abn7293>. URL: <https://www.science.org/doi/abs/10.1126/science.abn7293> (cited on page 79).

- [144] Thomas Hubregtsen et al. “Evaluation of Parameterized Quantum Circuits: On the Relation between Classification Accuracy, Expressibility, and Entangling Capability”. In: *Quantum Machine Intelligence* 3.1 (Mar. 2021), page 9. ISSN: 2524-4914. DOI: [10.1007/s42484-021-00038-w](https://doi.org/10.1007/s42484-021-00038-w) (cited on pages 38, 138).
- [145] *IBM Quantum*. <https://quantum-computing.ibm.com/>, 2022 (cited on pages 33–35, 88).
- [146] *IBM Quantum Roadmap*. <https://www.ibm.com/quantum/roadmap>, 2023 (cited on page 34).
- [147] *Il computer quantistico è “come un telescopio”, ma siamo ancora alla preistoria*, *Wired*. <https://www.wired.it/article/computer-quantistico-tecnologia-obiettivi-applicazioni/>, Feb. 2023 (cited on page 20).
- [148] *IonQ*. <https://ionq.com/>, 2023 (cited on pages 34, 35).
- [149] B Jaderberg et al. “Quantum self-supervised learning”. In: *Quantum Science and Technology* 7.3 (May 2022), page 035005. DOI: [10.1088/2058-9565/ac6825](https://doi.org/10.1088/2058-9565/ac6825). URL: <https://doi.org/10.1088/2058-9565/ac6825> (cited on page 124).
- [150] Daniel Jaschke and Simone Montangero. “Is quantum computing green? An estimate for an energy-efficiency quantum advantage”. In: *Quantum Science and Technology* 8.2 (Jan. 2023), page 025001. DOI: [10.1088/2058-9565/acae3e](https://doi.org/10.1088/2058-9565/acae3e). URL: <https://dx.doi.org/10.1088/2058-9565/acae3e> (cited on pages 139, 206).
- [151] Sofiene Jerbi et al. “Parametrized quantum policies for reinforcement learning”. In: *Advances in Neural Information Processing Systems* 34 (2021), pages 28362–28375 (cited on page 54).
- [152] Sofiene Jerbi et al. *Quantum machine learning beyond kernel methods*. 2022. arXiv: [2110.13162](https://arxiv.org/abs/2110.13162) [quant-ph] (cited on pages 70–72, 131).
- [153] Jiaqing Jiang, Kun Wang, and Xin Wang. “Physical Implementability of Linear Maps and Its Application in Error Mitigation”. In: *Quantum* 5 (Dec. 2021), page 600. ISSN: 2521-327X. DOI: [10.22331/q-2021-12-07-600](https://doi.org/10.22331/q-2021-12-07-600). URL: <https://doi.org/10.22331/q-2021-12-07-600> (cited on page 145).
- [154] John de Pillis. “Linear Transformations Which Preserve Hermitian and Positive Semidefinite Operators.” In: *Pacific Journal of Mathematics* 23.1 (Jan. 1967), pages 129–137 (cited on page 145).
- [155] Abhinav Kandala et al. “Hardware-Efficient Variational Quantum Eigensolver for Small Molecules and Quantum Magnets”. In: *Nature* 549.7671 (Sept. 2017), pages 242–246. ISSN: 1476-4687. DOI: [10.1038/nature23879](https://doi.org/10.1038/nature23879) (cited on pages 38, 98).
- [156] Abhinav Kandala et al. “Error Mitigation Extends the Computational Reach of a Noisy Quantum Processor”. In: *Nature* 567.7749 (Mar. 2019), pages 491–495. ISSN: 1476-4687. DOI: [10.1038/s41586-019-1040-7](https://doi.org/10.1038/s41586-019-1040-7) (cited on page 143).
- [157] Peter J Karalekas et al. “A quantum-classical cloud platform optimized for variational hybrid algorithms”. In: *Quantum Science and Technology* 5.2 (Apr. 2020), page 024003. DOI: [10.1088/2058-9565/ab7559](https://doi.org/10.1088/2058-9565/ab7559). URL: <https://doi.org/10.1088/2058-9565/ab7559> (cited on pages 144, 154).
- [158] Vahid Karimipour, Fabio Benatti, and Roberto Floreanini. “Quasi-inversion of qubit channels”. In: *Phys. Rev. A* 101 (3 Mar. 2020), page 032109. DOI: [10.1103/PhysRevA.101.032109](https://doi.org/10.1103/PhysRevA.101.032109). URL: <https://link.aps.org/doi/10.1103/PhysRevA.101.032109> (cited on page 143).

- [159] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. “Quantum Algorithms for Deep Convolutional Neural Networks”. In: (2019). DOI: [10.48550/ARXIV.1911.01117](https://doi.org/10.48550/ARXIV.1911.01117). URL: <https://arxiv.org/abs/1911.01117> (cited on page 82).
- [160] Iordanis Kerenidis and Alessandro Luongo. “Classification of the MNIST data set with quantum slow feature analysis”. In: *Phys. Rev. A* 101 (6 June 2020), page 062327. DOI: [10.1103/PhysRevA.101.062327](https://doi.org/10.1103/PhysRevA.101.062327). URL: <https://link.aps.org/doi/10.1103/PhysRevA.101.062327> (cited on pages 54, 93).
- [161] Amir Khoshaman et al. “Quantum variational autoencoder”. In: *Quantum Science and Technology* 4.1 (Sept. 2018), page 014001. DOI: [10.1088/2058-9565/aada1f](https://doi.org/10.1088/2058-9565/aada1f). URL: <https://doi.org/10.1088/2058-9565/aada1f> (cited on pages 108, 109).
- [162] Bobak Toussi Kiani, Seth Lloyd, and Reevu Maity. *Learning Unitaries by Gradient Descent*. 2020. DOI: [10.48550/ARXIV.2001.11897](https://doi.org/10.48550/ARXIV.2001.11897). URL: <https://arxiv.org/abs/2001.11897> (cited on pages 140, 207).
- [163] Nathan Killoran et al. “Continuous-variable quantum neural networks”. In: *Phys. Rev. Research* 1 (3 Oct. 2019), page 033063 (cited on pages 55, 81).
- [164] Joonho Kim and Yaron Oz. *Entanglement Diagnostics for Efficient Quantum Computation*. 2021. DOI: [10.48550/ARXIV.2102.12534](https://doi.org/10.48550/ARXIV.2102.12534). URL: <https://arxiv.org/abs/2102.12534> (cited on pages 130, 136, 141).
- [165] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: [10.48550/ARXIV.1412.6980](https://doi.org/10.48550/ARXIV.1412.6980). URL: <https://arxiv.org/abs/1412.6980> (cited on pages 40, 116, 211).
- [166] E. Knill. “Quantum Computing with Realistically Noisy Devices”. In: *Nature* 434.7029 (Mar. 2005), pages 39–44. ISSN: 1476-4687. DOI: [10.1038/nature03350](https://doi.org/10.1038/nature03350) (cited on page 142).
- [167] B. Kraus et al. “Preparation of entangled states by quantum Markov processes”. In: *Phys. Rev. A* 78 (4 Oct. 2008), page 042307. DOI: [10.1103/PhysRevA.78.042307](https://doi.org/10.1103/PhysRevA.78.042307). URL: <https://link.aps.org/doi/10.1103/PhysRevA.78.042307> (cited on page 200).
- [168] Lasse Bjørn Kristensen et al. “An artificial spiking quantum neuron”. In: *npj Quantum Information* 7.1 (Apr. 2021). DOI: [10.1038/s41534-021-00381-7](https://doi.org/10.1038/s41534-021-00381-7). URL: <https://doi.org/10.1038/s41534-021-00381-7> (cited on page 81).
- [169] C. Kruszynska and B. Kraus. “Local entanglability and multipartite entanglement”. In: *Phys. Rev. A* 79 (5 May 2009), page 052304. DOI: [10.1103/PhysRevA.79.052304](https://doi.org/10.1103/PhysRevA.79.052304). URL: <https://link.aps.org/doi/10.1103/PhysRevA.79.052304> (cited on pages 84, 87, 93, 200).
- [170] Jonas Kübler, Simon Buchholz, and Bernhard Schölkopf. “The Inductive Bias of Quantum Kernels”. In: *Advances in Neural Information Processing Systems*. Edited by M. Ranzato et al. Volume 34. Curran Associates, Inc., 2021, pages 12661–12673. URL: <https://proceedings.neurips.cc/paper/2021/file/69adc1e107f7f7d035d7baf04342e1ca-Paper.pdf> (cited on page 79).
- [171] L. Lamata et al. “Quantum autoencoders via quantum adders with genetic algorithms”. In: *Quantum Science and Technology* 4.1 (Oct. 2018), page 014007. DOI: [10.1088/2058-9565/aae22b](https://doi.org/10.1088/2058-9565/aae22b). URL: <https://doi.org/10.1088/2058-9565/aae22b> (cited on pages 81, 108, 113).
- [172] Martin Larocca et al. *Theory of overparametrization in quantum neural networks*. 2021. arXiv: [2109.11676](https://arxiv.org/abs/2109.11676) [quant-ph] (cited on pages 140, 207).

- [173] Martin Larocca et al. “Group-Invariant Quantum Machine Learning”. In: *PRX Quantum* 3 (3 Sept. 2022), page 030341. DOI: [10.1103/PRXQuantum.3.030341](https://doi.org/10.1103/PRXQuantum.3.030341). URL: <https://link.aps.org/doi/10.1103/PRXQuantum.3.030341> (cited on pages 39, 50, 207).
- [174] Ryan LaRose and Brian Coyle. “Robust data encodings for quantum classifiers”. In: *Phys. Rev. A* 102 (3 Sept. 2020), page 032420 (cited on pages 113, 121).
- [175] Jose I. Latorre. *Image compression and entanglement*. 2005. DOI: [10.48550/ARXIV.QUANT-PH/0510031](https://doi.org/10.48550/ARXIV.QUANT-PH/0510031). URL: <https://arxiv.org/abs/quant-ph/0510031> (cited on page 82).
- [176] Lea Lautenbacher, Fernando de Melo, and Nadja K. Bernardes. “Approximating invertible maps by recovery channels: Optimality and an application to non-Markovian dynamics”. In: *Phys. Rev. A* 105 (4 Apr. 2022), page 042421. DOI: [10.1103/PhysRevA.105.042421](https://doi.org/10.1103/PhysRevA.105.042421). URL: <https://link.aps.org/doi/10.1103/PhysRevA.105.042421> (cited on page 143).
- [177] Phuc Q. Le, Fangyan Dong, and Kaoru Hirota. “A flexible representation of quantum images for polynomial preparation, image compression, and processing operations”. In: *Quantum Information Processing* 10.1 (Apr. 2010), pages 63–84. DOI: [10.1007/s11128-010-0177-y](https://doi.org/10.1007/s11128-010-0177-y). URL: <https://doi.org/10.1007/s11128-010-0177-y> (cited on page 82).
- [178] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (May 2015), pages 436–444. ISSN: 1476-4687. DOI: <https://doi.org/10.1038/nature14539> (cited on pages 36, 53, 111).
- [179] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST handwritten digit database”. In: *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist> 2 (2010) (cited on page 92).
- [180] M. Ledoux. *The Concentration of Measure Phenomenon*. Mathematical surveys and monographs. American Mathematical Society, 2001. ISBN: 9780821837924. URL: https://books.google.it/books?id=mCX%5C_cWL6rqwC (cited on page 43).
- [181] Alexander LeNail. “NN-SVG: Publication-Ready Neural Network Architecture Schematics”. In: *Journal of Open Source Software* 4.33 (2019), page 747. DOI: [10.21105/joss.00747](https://doi.org/10.21105/joss.00747). URL: <https://doi.org/10.21105/joss.00747> (cited on page 66).
- [182] M. Lewenstein. “Quantum Perceptrons”. In: *Journal of Modern Optics* 41.12 (Dec. 1994), pages 2491–2501. DOI: [10.1080/09500349414552331](https://doi.org/10.1080/09500349414552331). URL: <https://doi.org/10.1080/09500349414552331> (cited on pages 20, 53).
- [183] Panchi Li and Hong Xiao. “Model and algorithm of quantum-inspired neural network with sequence input based on controlled rotation gates”. In: *Applied Intelligence* 40.1 (May 2013), pages 107–126. DOI: [10.1007/s10489-013-0447-3](https://doi.org/10.1007/s10489-013-0447-3). URL: <https://doi.org/10.1007/s10489-013-0447-3> (cited on page 82).
- [184] Zi-Wen Liu et al. “Entanglement, Quantum Randomness, and Complexity beyond Scrambling”. In: *Journal of High Energy Physics* 2018.7 (July 2018), page 41. ISSN: 1029-8479. DOI: [10.1007/JHEP07\(2018\)041](https://doi.org/10.1007/JHEP07(2018)041) (cited on pages 129, 202).
- [185] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. *A rigorous and robust quantum speed-up in supervised machine learning*. 2020. arXiv: [2010.02174](https://arxiv.org/abs/2010.02174) [quant-ph] (cited on page 79).
- [186] Seth Lloyd. *Quantum approximate optimization is computationally universal*. 2018. DOI: [10.48550/ARXIV.1812.11075](https://doi.org/10.48550/ARXIV.1812.11075). URL: <https://arxiv.org/abs/1812.11075> (cited on page 39).

- [187] Seth Lloyd. *Quantum approximate optimization is computationally universal*. 2018. DOI: [10.48550/ARXIV.1812.11075](https://doi.org/10.48550/ARXIV.1812.11075). URL: <https://arxiv.org/abs/1812.11075> (cited on page 131).
- [188] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. “Quantum algorithms for supervised and unsupervised machine learning”. In: (2013). arXiv: [1307.0411](https://arxiv.org/abs/1307.0411) (cited on pages 53, 54).
- [189] Seth Lloyd et al. 2020. arXiv: [2001.03622](https://arxiv.org/abs/2001.03622) [quant-ph] (cited on pages 113, 121).
- [190] Angus Lowe et al. “Unified approach to data-driven quantum error mitigation”. In: *Phys. Rev. Research* 3 (3 July 2021), page 033098. DOI: [10.1103/PhysRevResearch.3.033098](https://doi.org/10.1103/PhysRevResearch.3.033098). URL: <https://link.aps.org/doi/10.1103/PhysRevResearch.3.033098> (cited on page 143).
- [191] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. “Characterizing quantum gates via randomized benchmarking”. In: *Phys. Rev. A* 85 (4 Apr. 2012), page 042311. DOI: [10.1103/PhysRevA.85.042311](https://doi.org/10.1103/PhysRevA.85.042311). URL: <https://link.aps.org/doi/10.1103/PhysRevA.85.042311> (cited on page 35).
- [192] S. Mangini et al. “Quantum computing models for artificial neural networks”. In: *Europhysics Letters* 134.1 (2021), page 10002. DOI: <https://doi.org/10.1209/0295-5075/134/10002> (cited on pages 7, 22, 53, 115).
- [193] Stefano Mangini and Marcello Benedetti. In preparation (cited on pages 7, 78).
- [194] Stefano Mangini, Lorenzo Maccone, and Chiara Macchiavello. “Qubit Noise Deconvolution”. In: *EPJ Quantum Technology* 9.1 (Nov. 2022), page 29. ISSN: 2196-0763. DOI: [10.1140/epjqt/s40507-022-00151-0](https://doi.org/10.1140/epjqt/s40507-022-00151-0) (cited on pages 7, 8, 142).
- [195] Stefano Mangini et al. “Quantum computing model of an artificial neuron with continuously valued input data”. In: *Mach. Learn.: Sci. Technol.* 1.4 (Oct. 2020), page 045008 (cited on pages 7, 55, 81, 109, 113, 118).
- [196] Stefano Mangini et al. “Quantum neural network autoencoder and classifier applied to an industrial case study”. In: *Quantum Machine Intelligence* 4.2 (June 2022). DOI: [10.1007/s42484-022-00070-4](https://doi.org/10.1007/s42484-022-00070-4). URL: <https://doi.org/10.1007/s42484-022-00070-4> (cited on pages 7, 8, 108).
- [197] V A Marčenko and L A Pastur. “Distribution of eigenvalues for some sets of random matrices”. In: *Mathematics of the USSR-Sbornik* 1.4 (Apr. 1967), page 457. DOI: [10.1070/SM1967v001n04ABEH001994](https://doi.org/10.1070/SM1967v001n04ABEH001994). URL: <https://dx.doi.org/10.1070/SM1967v001n04ABEH001994> (cited on page 139).
- [198] Andrea Mari, Thomas R. Bromley, and Nathan Killoran. “Estimating the gradient and higher-order derivatives on quantum hardware”. In: *Phys. Rev. A* 103 (1 Jan. 2021), page 012405. DOI: [10.1103/PhysRevA.103.012405](https://doi.org/10.1103/PhysRevA.103.012405). URL: <https://link.aps.org/doi/10.1103/PhysRevA.103.012405> (cited on pages 40, 42).
- [199] Andrea Mari, Nathan Shammah, and William J. Zeng. “Extending quantum probabilistic error cancellation by noise scaling”. In: *Phys. Rev. A* 104 (5 Nov. 2021), page 052607. DOI: [10.1103/PhysRevA.104.052607](https://doi.org/10.1103/PhysRevA.104.052607). URL: <https://link.aps.org/doi/10.1103/PhysRevA.104.052607> (cited on page 143).
- [200] G. Mauro D’Ariano, Matteo G.A. Paris, and Massimiliano F. Sacchi. “Quantum Tomography”. In: edited by Peter W. Hawkes. Volume 128. *Advances in Imaging and Electron Physics*. Elsevier, 2003, pages 205–308. DOI: [https://doi.org/10.1016/S1076-5670\(03\)80065-4](https://doi.org/10.1016/S1076-5670(03)80065-4). URL: <https://www.sciencedirect.com/science/article/pii/S1076567003800654> (cited on page 146).

- [201] Jarrod R McClean et al. “The theory of variational hybrid quantum-classical algorithms”. In: *New J. Phys.* 18.2 (2016), page 023023. DOI: [10.1088/1367-2630/18/2/023023](https://doi.org/10.1088/1367-2630/18/2/023023) (cited on pages 43, 115).
- [202] Jarrod R. McClean et al. “Barren plateaus in quantum neural network training landscapes”. In: *Nat. Commun.* 9.1 (2018), page 4812. DOI: <https://doi.org/10.1038/s41467-018-07090-4> (cited on pages 39, 43, 44, 46, 99, 103, 106, 129, 139, 207).
- [203] Warren S. McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The Bulletin of Mathematical Biophysics* 5.4 (Dec. 1943), pages 115–133. DOI: [10.1007/bf02478259](https://doi.org/10.1007/bf02478259). URL: <https://doi.org/10.1007/bf02478259> (cited on pages 82, 95).
- [204] Elizabeth S. Meckes. *The Random Matrix Theory of the Classical Compact Groups*. Cambridge Tracts in Mathematics. Cambridge University Press, 2019. DOI: [10.1017/9781108303453](https://doi.org/10.1017/9781108303453) (cited on page 46).
- [205] Elizabeth S. Meckes. *The Random Matrix Theory of the Classical Compact Groups*. Cambridge Tracts in Mathematics. Cambridge: Cambridge University Press, 2019. ISBN: 978-1-108-41952-9. DOI: [10.1017/9781108303453](https://doi.org/10.1017/9781108303453) (cited on page 127).
- [206] Riccardo Mengoni and Alessandra Di Pierro. “Kernel methods in Quantum Machine Learning”. In: *Quantum Machine Intelligence* 1.3-4 (Nov. 2019), pages 65–71. DOI: [10.1007/s42484-019-00007-4](https://doi.org/10.1007/s42484-019-00007-4). URL: <https://doi.org/10.1007/s42484-019-00007-4> (cited on pages 54, 70).
- [207] David A. Meyer and Nolan R. Wallach. “Global Entanglement in Multiparticle Systems”. In: *Journal of Mathematical Physics* 43.9 (Sept. 2002), pages 4273–4278. ISSN: 0022-2488. DOI: [10.1063/1.1497700](https://doi.org/10.1063/1.1497700) (cited on page 125).
- [208] Johannes Jakob Meyer et al. *Exploiting symmetry in variational quantum machine learning*. 2022. DOI: [10.48550/ARXIV.2205.06217](https://doi.org/10.48550/ARXIV.2205.06217). URL: <https://arxiv.org/abs/2205.06217> (cited on pages 39, 50, 128, 140, 207).
- [209] K. Mitarai et al. “Quantum circuit learning”. In: *Phys. Rev. A* 98 (3 Sept. 2018), page 032309 (cited on pages 40, 41, 72, 113, 116).
- [210] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pages 529–533 (cited on pages 36, 111).
- [211] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. 2nd. The MIT Press, 2018. ISBN: 0262039400 (cited on pages 57, 61, 64, 65, 71).
- [212] Klaus Mølmer and Anders Sørensen. “Multiparticle Entanglement of Hot Trapped Ions”. In: *Phys. Rev. Lett.* 82 (9 Mar. 1999), pages 1835–1838. DOI: [10.1103/PhysRevLett.82.1835](https://doi.org/10.1103/PhysRevLett.82.1835). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.82.1835> (cited on page 33).
- [213] S. Montangero. *Introduction to Tensor Network Methods*. Cham, CH: Springer Nature Switzerland AG, 2018 (cited on page 125).
- [214] M. E. S. Morales, J. D. Biamonte, and Z. Zimborás. “On the Universality of the Quantum Approximate Optimization Algorithm”. In: *Quantum Information Processing* 19.9 (Aug. 2020), page 291. ISSN: 1573-1332. DOI: [10.1007/s11128-020-02748-9](https://doi.org/10.1007/s11128-020-02748-9) (cited on page 131).
- [215] Lorenzo Moro et al. “Quantum compiling by deep reinforcement learning”. In: *Communications Physics* 4.1 (Aug. 2021). DOI: [10.1038/s42005-021-00684-3](https://doi.org/10.1038/s42005-021-00684-3). URL: <https://doi.org/10.1038/s42005-021-00684-3> (cited on page 54).

- [216] Adam Nahum et al. “Quantum Entanglement Growth under Random Unitary Dynamics”. In: *Phys. Rev. X* 7 (3 July 2017), page 031016. DOI: [10.1103/PhysRevX.7.031016](https://doi.org/10.1103/PhysRevX.7.031016). URL: <https://link.aps.org/doi/10.1103/PhysRevX.7.031016> (cited on page 136).
- [217] *Native Gates Information, IonQ*. <https://ionq.com/docs/getting-started-with-native-gates#introducing-the-native-gates>, 2022 (cited on page 33).
- [218] J. A. Nelder and R. Mead. “A Simplex Method for Function Minimization”. In: *The Computer Journal* 7.4 (Jan. 1965), pages 308–313. ISSN: 0010-4620. DOI: [10.1093/comjnl/7.4.308](https://doi.org/10.1093/comjnl/7.4.308) (cited on pages 43, 101).
- [219] Quynh T. Nguyen et al. *Theory for Equivariant Quantum Neural Networks*. 2022. DOI: [10.48550/ARXIV.2210.08566](https://doi.org/10.48550/ARXIV.2210.08566). URL: <https://arxiv.org/abs/2210.08566> (cited on page 39).
- [220] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge, UK: Cambridge University Press, 2010. DOI: <https://doi.org/10.1017/CB09780511976667> (cited on pages 19, 25, 27, 34, 45, 126, 143, 145, 146, 151, 153, 213).
- [221] Carlos Ortiz Marrero, Mária Kieferová, and Nathan Wiebe. “Entanglement-Induced Barren Plateaus”. In: *PRX Quantum* 2 (4 2021), page 040316. DOI: <https://doi.org/10.1103/PRXQuantum.2.040316> (cited on pages 45, 129, 138, 140).
- [222] Mateusz Ostaszewski, Edward Grant, and Marcello Benedetti. “Structure optimization for parameterized quantum circuits”. In: *Quantum* 5 (Jan. 2021), page 391. ISSN: 2521-327X. DOI: [10.22331/q-2021-01-28-391](https://doi.org/10.22331/q-2021-01-28-391). URL: <https://doi.org/10.22331/q-2021-01-28-391> (cited on pages 39, 43).
- [223] Sebastian Paeckel et al. “Time-evolution methods for matrix-product states”. In: *Annals of Physics* 411 (Dec. 2019), page 167998. ISSN: 0003-4916. DOI: [10.1016/j.aop.2019.167998](https://doi.org/10.1016/j.aop.2019.167998). URL: <http://dx.doi.org/10.1016/j.aop.2019.167998> (cited on page 126).
- [224] Don N. Page. “Average entropy of a subsystem”. In: *Phys. Rev. Lett.* 71 (9 Aug. 1993), pages 1291–1294. DOI: [10.1103/PhysRevLett.71.1291](https://doi.org/10.1103/PhysRevLett.71.1291). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.71.1291> (cited on page 127).
- [225] Matteo Paris and Jaroslav Řeháček, editors. *Quantum State Estimation*. Lecture Notes in Physics 649. Berlin ; New York: Springer, 2004. ISBN: 978-3-540-22329-0 (cited on pages 146, 147).
- [226] German I. Parisi et al. “Continual lifelong learning with neural networks: A review”. In: *Neural Networks* 113 (May 2019), pages 54–71. DOI: [10.1016/j.neunet.2019.01.012](https://doi.org/10.1016/j.neunet.2019.01.012). URL: <https://doi.org/10.1016/j.neunet.2019.01.012> (cited on page 56).
- [227] Taylor L. Patti et al. “Entanglement devised barren plateau mitigation”. In: *Phys. Rev. Research* 3 (3 July 2021), page 033090. DOI: [10.1103/PhysRevResearch.3.033090](https://doi.org/10.1103/PhysRevResearch.3.033090). URL: <https://link.aps.org/doi/10.1103/PhysRevResearch.3.033090> (cited on pages 129, 130).
- [228] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pages 2825–2830 (cited on page 110).
- [229] Adrián Pérez-Salinas et al. “Data re-uploading for a universal quantum classifier”. In: *Quantum* 4 (Feb. 2020), page 226. ISSN: 2521-327X. DOI: <https://doi.org/10.22331/q-2020-02-06-226> (cited on pages 72, 73, 109, 121, 124, 128, 130).
- [230] Alberto Peruzzo et al. “A variational eigenvalue solver on a photonic quantum processor”. In: *Nat. Commun.* 5.1 (2014). DOI: <https://doi.org/10.1038/ncomms5213> (cited on pages 36, 39, 98).

- [231] Arthur Pesah et al. *Absence of Barren Plateaus in Quantum Convolutional Neural Networks*. 2020. arXiv: [2011.02966](https://arxiv.org/abs/2011.02966) [quant-ph] (cited on page 50).
- [232] Evan Peters and Maria Schuld. *Generalization despite overfitting in quantum machine learning models*. 2022. DOI: [10.48550/ARXIV.2209.05523](https://doi.org/10.48550/ARXIV.2209.05523). URL: <https://arxiv.org/abs/2209.05523> (cited on pages 76, 79).
- [233] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. Version 20121115. Nov. 2012. URL: <http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html> (cited on page 63).
- [234] Dénes Petz. “A survey of certain trace inequalities”. en. In: *Banach Center Publ.* 30.1 (1994), pages 287–298 (cited on page 193).
- [235] M. J. D. Powell. “Direct search algorithms for optimization calculations”. In: *Acta Numerica* 7 (1998), pages 287–336. DOI: <https://doi.org/10.1017/S0962492900002841> (cited on pages 43, 101–103, 121).
- [236] John Preskill. “Quantum Computing in the NISQ era and beyond”. In: *Quantum* 2 (Aug. 2018), page 79. ISSN: 2521-327X. DOI: [10.22331/q-2018-08-06-79](https://doi.org/10.22331/q-2018-08-06-79) (cited on pages 19, 34, 81, 142).
- [237] John Preskill. *Quantum computing 40 years later*. 2021. DOI: [10.48550/ARXIV.2106.10522](https://doi.org/10.48550/ARXIV.2106.10522). URL: <https://arxiv.org/abs/2106.10522> (cited on page 19).
- [238] Z. Puchała and J.A. Miszczak. “Symbolic integration with respect to the Haar measure on the unitary groups”. In: *Bulletin of the Polish Academy of Sciences: Technical Sciences* 65.No 1 (2017), pages 21–27. DOI: [10.1515/bpasts-2017-0003](https://doi.org/10.1515/bpasts-2017-0003). URL: <http://journals.pan.pl/Content/105697/PDF/10.1515bpasts-2017-0003.pdf> (cited on page 46).
- [239] Zbigniew Puchała, Łukasz Paweła, and Karol Życzkowski. “Distinguishability of generic quantum states”. In: *Physical Review A* 93.6 (2016), page 062112. DOI: <https://doi.org/10.1103/PhysRevA.93.062112> (cited on page 139).
- [240] *Qandela*. <https://www.quandela.com/>, 2023 (cited on page 34).
- [241] *Quantinuum* (cited on page 34).
- [242] *Quantinuum Sets New Record with Highest Ever Quantum Volume*. <https://www.quantinuum.com/news/quantinuum-sets-new-record-with-highest-ever-quantum-volume>, 2022 (cited on page 35).
- [243] *Quantum Volume of 512 announced by IBM Quantum on Twitter*. <https://twitter.com/jaygambetta/status/1529489786242744320>, 2022 (cited on page 35).
- [244] “Quasi-Newton Methods”. In: *Numerical Optimization*. New York, NY: Springer New York, 2006, pages 135–163. ISBN: 978-0-387-40065-5. DOI: [10.1007/978-0-387-40065-5_6](https://doi.org/10.1007/978-0-387-40065-5_6). URL: https://doi.org/10.1007/978-0-387-40065-5_6 (cited on page 40).
- [245] *QuEra* (cited on page 34).
- [246] Michael Ragone et al. *Representation Theory for Geometric Quantum Machine Learning*. 2022. DOI: [10.48550/ARXIV.2210.07980](https://doi.org/10.48550/ARXIV.2210.07980). URL: <https://arxiv.org/abs/2210.07980> (cited on page 140).
- [247] Aditya Ramesh et al. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. DOI: [10.48550/ARXIV.2204.06125](https://doi.org/10.48550/ARXIV.2204.06125). URL: <https://arxiv.org/abs/2204.06125> (cited on page 53).
- [248] Arthur G. Rattew et al. *A Domain-agnostic, Noise-resistant, Hardware-efficient Evolutionary Variational Quantum Eigensolver*. 2019. DOI: [10.48550/ARXIV.1910.09694](https://doi.org/10.48550/ARXIV.1910.09694). URL: <https://arxiv.org/abs/1910.09694> (cited on page 39).

- [249] P. Rebentrost et al. “Quantum Hopfield neural network”. en. In: *Physical Review A* 98.4 (Oct. 2018), page 042308. ISSN: 2469-9926, 2469-9934. DOI: [10.1103/PhysRevA.98.042308](https://doi.org/10.1103/PhysRevA.98.042308). (Visited on 05/30/2019) (cited on page 95).
- [250] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. “Quantum Support Vector Machine for Big Data Classification”. In: *Phys. Rev. Lett.* 113.13 (2014). DOI: [10.1103/PhysRevLett.113.130503](https://doi.org/10.1103/PhysRevLett.113.130503) (cited on page 53).
- [251] Mark Reid. “Generalization Bounds”. In: *Encyclopedia of Machine Learning*. Edited by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pages 447–454. ISBN: 978-0-387-30164-8. DOI: [10.1007/978-0-387-30164-8_328](https://doi.org/10.1007/978-0-387-30164-8_328). URL: https://doi.org/10.1007/978-0-387-30164-8_328 (cited on page 60).
- [252] Raúl Rojas. *Neural Networks*. Springer Berlin Heidelberg, 1996. DOI: [10.1007/978-3-642-61068-4](https://doi.org/10.1007/978-3-642-61068-4). URL: <https://doi.org/10.1007/978-3-642-61068-4> (cited on page 82).
- [253] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik. “Quantum autoencoders for efficient compression of quantum data”. In: *Quantum Science and Technology* 2.4 (Aug. 2017), page 045001. DOI: [10.1088/2058-9565/aa8072](https://doi.org/10.1088/2058-9565/aa8072). URL: <https://dx.doi.org/10.1088/2058-9565/aa8072> (cited on pages 81, 108, 109, 113).
- [254] Simone Roncallo, Lorenzo Maccone, and Chiara Macchiavello. “Multiqubit noise deconvolution and characterization”. In: *Phys. Rev. A* 107 (2 Feb. 2023), page 022419. DOI: [10.1103/PhysRevA.107.022419](https://doi.org/10.1103/PhysRevA.107.022419). URL: <https://link.aps.org/doi/10.1103/PhysRevA.107.022419> (cited on pages 146, 148, 151).
- [255] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological Review* 65.6 (1958), pages 386–408. DOI: [10.1037/h0042519](https://doi.org/10.1037/h0042519). URL: <https://doi.org/10.1037/h0042519> (cited on pages 65, 80, 82).
- [256] M. Rossi et al. “Quantum hypergraph states”. en. In: *New Journal of Physics* 15.11 (Nov. 2013), page 113022. ISSN: 1367-2630. DOI: [10.1088/1367-2630/15/11/113022](https://doi.org/10.1088/1367-2630/15/11/113022). (Visited on 08/22/2019) (cited on pages 95, 97).
- [257] Manuel S. Rudolph et al. *Synergy Between Quantum Circuits and Tensor Networks: Short-cutting the Race to Practical Quantum Advantage*. 2022. DOI: [10.48550/ARXIV.2208.13673](https://doi.org/10.48550/ARXIV.2208.13673). URL: <https://arxiv.org/abs/2208.13673> (cited on page 50).
- [258] Stefan H. Sack et al. *Avoiding barren plateaus using classical shadows*. 2022. DOI: [10.48550/ARXIV.2201.08194](https://doi.org/10.48550/ARXIV.2201.08194). URL: <https://arxiv.org/abs/2201.08194> (cited on pages 45, 129, 130, 141, 202).
- [259] Francesco Scala et al. “Quantum variational learning for entanglement witnessing”. In: *2022 International Joint Conference on Neural Networks (IJCNN)*. 2022, pages 1–8. DOI: [10.1109/IJCNN55064.2022.9892080](https://doi.org/10.1109/IJCNN55064.2022.9892080) (cited on page 7).
- [260] Bernhard Schölkopf, Alexander Johannes Smola, and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Reprint. Adaptive Computation and Machine Learning Series. Cambridge, Mass.: MIT Press, 2002. ISBN: 978-0-262-19475-4 (cited on pages 64, 71).
- [261] Franz J. Schreiber, Jens Eisert, and Johannes Jakob Meyer. *Classical surrogates for quantum learning models*. 2022. DOI: [10.48550/ARXIV.2206.11740](https://doi.org/10.48550/ARXIV.2206.11740). URL: <https://arxiv.org/abs/2206.11740> (cited on page 79).

- [262] M. Schuld, M. Fingerhuth, and F. Petruccione. “Implementing a distance-based classifier with a quantum interference circuit”. In: *Europhysics Letters* 119.6 (Dec. 2017), page 60002. DOI: [10.1209/0295-5075/119/60002](https://doi.org/10.1209/0295-5075/119/60002). URL: <https://dx.doi.org/10.1209/0295-5075/119/60002> (cited on pages 81, 95).
- [263] Maria Schuld. *Supervised quantum machine learning models are kernel methods*. 2021. DOI: [10.48550/ARXIV.2101.11020](https://arxiv.org/abs/2101.11020). URL: <https://arxiv.org/abs/2101.11020> (cited on pages 70, 131).
- [264] Maria Schuld and Nathan Killoran. “Quantum Machine Learning in Feature Hilbert Spaces”. In: *Phys. Rev. Lett.* 122 (4 Feb. 2019), page 040504 (cited on pages 64, 70, 71, 81, 106).
- [265] Maria Schuld and Nathan Killoran. “Is Quantum Advantage the Right Goal for Quantum Machine Learning?” In: *PRX Quantum* 3 (3 July 2022), page 030101. DOI: [10.1103/PRXQuantum.3.030101](https://link.aps.org/doi/10.1103/PRXQuantum.3.030101). URL: <https://link.aps.org/doi/10.1103/PRXQuantum.3.030101> (cited on page 80).
- [266] Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*. Springer International Publishing, 2018. ISBN: 978-3-319-96424-9 (cited on pages 53, 54, 201).
- [267] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. “The Quest for a Quantum Neural Network”. In: *Quantum Information Processing* 13.11 (Nov. 2014), pages 2567–2586. ISSN: 1573-1332. DOI: [10.1007/s11128-014-0809-8](https://doi.org/10.1007/s11128-014-0809-8) (cited on pages 53, 55, 82).
- [268] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. “An introduction to quantum machine learning”. In: *Contemporary Physics* 56.2 (2015), pages 172–185. DOI: [10.1080/00107514.2014.964942](https://doi.org/10.1080/00107514.2014.964942). eprint: <https://doi.org/10.1080/00107514.2014.964942>. URL: <https://doi.org/10.1080/00107514.2014.964942> (cited on pages 53, 81).
- [269] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. “Effect of data encoding on the expressive power of variational quantum-machine-learning models”. In: *Phys. Rev. A* 103 (3 Mar. 2021), page 032430. DOI: [10.1103/PhysRevA.103.032430](https://link.aps.org/doi/10.1103/PhysRevA.103.032430). URL: <https://link.aps.org/doi/10.1103/PhysRevA.103.032430> (cited on pages 72, 73, 76, 113, 121, 124, 128, 130, 210).
- [270] Maria Schuld et al. “Evaluating analytic gradients on quantum hardware”. In: *Phys. Rev. A* 99 (3 Mar. 2019), page 032331. DOI: <https://doi.org/10.1103/PhysRevA.99.032331> (cited on pages 40–42, 116).
- [271] Benjamin Schumacher. “Quantum coding”. In: *Phys. Rev. A* 51 (4 Apr. 1995), pages 2738–2747. DOI: [10.1103/PhysRevA.51.2738](https://link.aps.org/doi/10.1103/PhysRevA.51.2738). URL: <https://link.aps.org/doi/10.1103/PhysRevA.51.2738> (cited on page 23).
- [272] Simone Severini. *Nella terra dei qubit. La fisica quantistica e i confini dell’informatica*. Tréfolgie, 2022 (cited on page 159).
- [273] Fereshte Shahbeigi et al. “Quasi-inversion of quantum and classical channels in finite dimensions”. In: *Journal of Physics A: Mathematical and Theoretical* 54.34 (Aug. 2021), page 345301. DOI: [10.1088/1751-8121/ac13db](https://doi.org/10.1088/1751-8121/ac13db). URL: <https://doi.org/10.1088/1751-8121/ac13db> (cited on page 143).
- [274] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. DOI: [10.1017/CB09781107298019](https://doi.org/10.1017/CB09781107298019) (cited on pages 57, 58, 60, 61, 64, 65, 194, 195).

- [275] Kunal Sharma et al. “Reformulation of the No-Free-Lunch Theorem for Entangled Datasets”. In: *Phys. Rev. Lett.* 128 (7 Feb. 2022), page 070501. DOI: [10.1103/PhysRevLett.128.070501](https://doi.org/10.1103/PhysRevLett.128.070501). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.128.070501> (cited on pages 140, 207).
- [276] S. Shin, Y. S. Teo, and H. Jeong. “Exponential data encoding for quantum supervised learning”. In: *Phys. Rev. A* 107 (1 Jan. 2023), page 012422. DOI: [10.1103/PhysRevA.107.012422](https://doi.org/10.1103/PhysRevA.107.012422). URL: <https://link.aps.org/doi/10.1103/PhysRevA.107.012422> (cited on pages 72, 76, 79).
- [277] Peter W. Shor. “Scheme for reducing decoherence in quantum computer memory”. In: *Phys. Rev. A* 52 (4 Oct. 1995), R2493–R2496. DOI: [10.1103/PhysRevA.52.R2493](https://doi.org/10.1103/PhysRevA.52.R2493). URL: <https://link.aps.org/doi/10.1103/PhysRevA.52.R2493> (cited on page 142).
- [278] Vikesh Siddhu. “Maximum a posteriori probability estimates for quantum tomography”. In: *Phys. Rev. A* 99 (1 Jan. 2019), page 012342. DOI: [10.1103/PhysRevA.99.012342](https://doi.org/10.1103/PhysRevA.99.012342). URL: <https://link.aps.org/doi/10.1103/PhysRevA.99.012342> (cited on page 143).
- [279] David Silver et al. “Mastering the Game of Go with Deep Neural Networks and Tree Search”. In: *Nature* 529.7587 (Jan. 2016), pages 484–489. ISSN: 1476-4687. DOI: [10.1038/nature16961](https://doi.org/10.1038/nature16961) (cited on pages 20, 54, 56).
- [280] Pietro Silvi et al. “The Tensor Networks Anthology: Simulation techniques for many-body quantum lattice systems”. In: *SciPost Physics Lecture Notes* (Mar. 2019). ISSN: 2590-1990. DOI: [10.21468/scipostphyslectnotes.8](https://doi.org/10.21468/scipostphyslectnotes.8). URL: <http://dx.doi.org/10.21468/SciPostPhysLectNotes.8> (cited on page 125).
- [281] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. “Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms”. In: *Adv. Quantum Technol.* 2.12 (2019), page 1900070. DOI: <https://doi.org/10.1002/qute.201900070> (cited on pages 38, 50, 124, 125, 129, 132, 133, 138, 139, 205).
- [282] Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko. “Quantum agents in the gym: a variational quantum algorithm for deep q-learning”. In: *Quantum* 6 (2022), page 720 (cited on page 54).
- [283] Andrea Skolik et al. “Layerwise learning for quantum neural networks”. In: *Quantum Machine Intelligence* 3.1 (Jan. 2021). DOI: [10.1007/s42484-020-00036-4](https://doi.org/10.1007/s42484-020-00036-4). URL: <https://doi.org/10.1007/s42484-020-00036-4> (cited on pages 50, 104–106, 130).
- [284] Andrea Skolik et al. “Equivariant quantum circuits for learning on weighted graphs”. In: *arXiv preprint arXiv:2205.06109* (2022) (cited on pages 39, 50, 128, 140, 207).
- [285] Andrea Skolik et al. “Robustness of Quantum Reinforcement Learning under Hardware Errors”. In: *EPJ Quantum Technology* 10.1 (Feb. 2023), page 8. ISSN: 2196-0763. DOI: [10.1140/epjqt/s40507-023-00166-1](https://doi.org/10.1140/epjqt/s40507-023-00166-1) (cited on pages 7, 54, 85).
- [286] Robert S. Smith, Michael J. Curtis, and William J. Zeng. *A Practical Quantum Instruction Set Architecture*. 2016. arXiv: [1608.03355](https://arxiv.org/abs/1608.03355) [quant-ph] (cited on pages 154, 157).
- [287] James C Spall. “An overview of the simultaneous perturbation method for efficient optimization”. In: *Johns Hopkins apl technical digest* 19.4 (1998), pages 482–492 (cited on pages 40, 89).
- [288] Rishi Sreedhar et al. *The Quantum Approximate Optimization Algorithm performance with low entanglement and high circuit depth*. 2022. DOI: [10.48550/ARXIV.2207.03404](https://doi.org/10.48550/ARXIV.2207.03404). URL: <https://arxiv.org/abs/2207.03404> (cited on page 207).
- [289] A. M. Steane. “Error Correcting Codes in Quantum Theory”. In: *Phys. Rev. Lett.* 77 (5 July 1996), pages 793–797. DOI: [10.1103/PhysRevLett.77.793](https://doi.org/10.1103/PhysRevLett.77.793). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.77.793> (cited on page 142).

- [290] James Stokes et al. “Quantum Natural Gradient”. In: *Quantum* 4 (May 2020), page 269. ISSN: 2521-327X. DOI: <https://doi.org/10.22331/q-2020-05-25-269> (cited on pages 40, 43).
- [291] Edwin Stoudenmire and David J Schwab. “Supervised Learning with Tensor Networks”. In: *Advances in Neural Information Processing Systems*. Edited by D. Lee et al. Volume 29. Curran Associates, Inc., 2016. URL: <https://proceedings.neurips.cc/paper/2016/file/5314b9674c86e3f9d1ba25ef9bb32895-Paper.pdf> (cited on page 55).
- [292] Yasunari Suzuki et al. “Quantum Error Mitigation as a Universal Error Reduction Technique: Applications from the NISQ to the Fault-Tolerant Quantum Computing Eras”. In: *PRX Quantum* 3 (1 Mar. 2022), page 010345. DOI: [10.1103/PRXQuantum.3.010345](https://doi.org/10.1103/PRXQuantum.3.010345). URL: <https://link.aps.org/doi/10.1103/PRXQuantum.3.010345> (cited on pages 142, 152).
- [293] Francesco Tacchino et al. “An artificial neuron implemented on an actual quantum processor”. In: *npj Quantum Information* 5.1 (2019). DOI: <https://doi.org/10.1038/s41534-019-0140-4> (cited on pages 54, 81, 82, 95–98, 100, 106, 113).
- [294] Francesco Tacchino et al. “Quantum Computers as Universal Quantum Simulators: State-of-the-Art and Perspectives”. In: *Advanced Quantum Technologies* 3.3 (2020), page 1900052. DOI: <https://doi.org/10.1002/qute.201900052>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/qute.201900052>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/qute.201900052> (cited on page 34).
- [295] Francesco Tacchino et al. “Quantum implementation of an artificial feed-forward neural network”. In: *Quantum Sci. Technol.* 5.4 (2020), page 044010. DOI: <https://doi.org/10.1088/2058-9565/abb8e4> (cited on pages 87, 93, 95, 96, 113).
- [296] Francesco Tacchino et al. “Variational Learning for Quantum Artificial Neural Networks”. In: *IEEE Transactions on Quantum Engineering* 2 (2021), pages 1–10. DOI: [10.1109/TQE.2021.3062494](https://doi.org/10.1109/TQE.2021.3062494) (cited on pages 7, 94, 113, 124).
- [297] Ewin Tang. “A Quantum-Inspired Classical Algorithm for Recommendation Systems”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, 2019, pages 217–228. ISBN: 9781450367059. DOI: <https://doi.org/10.1145/3313276.3316310> (cited on pages 53, 55).
- [298] Ho Lun Tang et al. “Qubit-ADAPT-VQE: An Adaptive Algorithm for Constructing Hardware-Efficient Ansätze on a Quantum Processor”. In: *PRX Quantum* 2 (2 Apr. 2021), page 020310. DOI: [10.1103/PRXQuantum.2.020310](https://doi.org/10.1103/PRXQuantum.2.020310). URL: <https://link.aps.org/doi/10.1103/PRXQuantum.2.020310> (cited on page 39).
- [299] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta. “Error Mitigation for Short-Depth Quantum Circuits”. In: *Phys. Rev. Lett.* 119 (18 Nov. 2017), page 180509. DOI: [10.1103/PhysRevLett.119.180509](https://doi.org/10.1103/PhysRevLett.119.180509). URL: <https://link.aps.org/doi/10.1103/PhysRevLett.119.180509> (cited on pages 143, 144, 151).
- [300] Supanut Thanasilp et al. *Subtleties in the trainability of quantum machine learning models*. 2021. DOI: [10.48550/ARXIV.2110.14753](https://doi.org/10.48550/ARXIV.2110.14753). URL: <https://arxiv.org/abs/2110.14753> (cited on page 44).
- [301] Jules Tilly et al. “The Variational Quantum Eigensolver: A review of methods and best practices”. In: *Physics Reports* 986 (2022). The Variational Quantum Eigensolver: a review of methods and best practices, pages 1–128. ISSN: 0370-1573. DOI: <https://doi.org/10.1016/j.physrep.2022.08.003>. URL: <https://www.sciencedirect.com/science/article/pii/S0370157322003118> (cited on pages 22, 39, 42, 43).

- [302] E. Torrontegui and J. J. Garcia-Ripoll. “Unitary quantum perceptron as efficient universal approximator”. In: *EPL* 125.3 (Mar. 2019), page 30004. DOI: <https://doi.org/10.1209/0295-5075/125/30004> (cited on pages 81, 96, 128).
- [303] L. G. Valiant. “A Theory of the Learnable”. In: *Commun. ACM* 27.11 (Nov. 1984), pages 1134–1142. ISSN: 0001-0782. DOI: [10.1145/1968.1972](https://doi.org/10.1145/1968.1972). URL: <https://doi.org/10.1145/1968.1972> (cited on page 60).
- [304] V. N. Vapnik and A. Ya. Chervonenkis. “On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities”. In: *Theory of Probability & Its Applications* 16.2 (1971), pages 264–280. DOI: [10.1137/1116025](https://doi.org/10.1137/1116025). eprint: <https://doi.org/10.1137/1116025>. URL: <https://doi.org/10.1137/1116025> (cited on page 60).
- [305] Guillaume Verdon et al. *Learning to learn with quantum neural networks via classical neural networks*. 2019. arXiv: [1907.05415](https://arxiv.org/abs/1907.05415) [quant-ph] (cited on page 50).
- [306] Jean-Loup Ville et al. *Leveraging Randomized Compiling for the QITE Algorithm*. 2021. arXiv: [2104.08785](https://arxiv.org/abs/2104.08785) [quant-ph] (cited on page 149).
- [307] Lorenza Viola and Seth Lloyd. “Dynamical suppression of decoherence in two-state quantum systems”. In: *Phys. Rev. A* 58 (4 Oct. 1998), pages 2733–2744. DOI: [10.1103/PhysRevA.58.2733](https://doi.org/10.1103/PhysRevA.58.2733). URL: <https://link.aps.org/doi/10.1103/PhysRevA.58.2733> (cited on page 142).
- [308] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pages 261–272. DOI: [10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2) (cited on page 121).
- [309] Tyler Volkoff and Patrick J Coles. “Large gradients via correlation in random parameterized circuits”. In: *Quantum Sci. Technol.* 6 (2021). URL: <http://iopscience.iop.org/article/10.1088/2058-9565/abd891> (cited on pages 50, 130).
- [310] Joseph Vovrosh et al. “Simple mitigation of global depolarizing errors in quantum simulations”. In: *Phys. Rev. E* 104 (3 Sept. 2021), page 035309. DOI: [10.1103/PhysRevE.104.035309](https://doi.org/10.1103/PhysRevE.104.035309). URL: <https://link.aps.org/doi/10.1103/PhysRevE.104.035309> (cited on page 148).
- [311] Joel J. Wallman and Joseph Emerson. “Noise tailoring for scalable quantum computation via randomized compiling”. In: *Phys. Rev. A* 94 (5 Nov. 2016), page 052325. DOI: [10.1103/PhysRevA.94.052325](https://doi.org/10.1103/PhysRevA.94.052325). URL: <https://link.aps.org/doi/10.1103/PhysRevA.94.052325> (cited on page 149).
- [312] Samson Wang et al. “Noise-Induced Barren Plateaus in Variational Quantum Algorithms”. In: *Nature Communications* 12.1 (Nov. 2021), page 6961. ISSN: 2041-1723. DOI: [10.1038/s41467-021-27045-6](https://doi.org/10.1038/s41467-021-27045-6) (cited on pages 45, 129, 140).
- [313] Matthew Ware et al. “Experimental Pauli-frame randomization on a superconducting qubit”. In: *Physical Review A* 103.4 (Apr. 2021). ISSN: 2469-9934. DOI: [10.1103/PhysRevA.103.042604](https://doi.org/10.1103/PhysRevA.103.042604). URL: <http://dx.doi.org/10.1103/PhysRevA.103.042604> (cited on page 149).
- [314] Michael L. Waskom. “seaborn: statistical data visualization”. In: *Journal of Open Source Software* 6.60 (2021), page 3021. DOI: [10.21105/joss.03021](https://doi.org/10.21105/joss.03021). URL: <https://doi.org/10.21105/joss.03021> (cited on pages 210, 211).
- [315] John Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018. DOI: [10.1017/9781316848142](https://doi.org/10.1017/9781316848142) (cited on page 77).
- [316] Zak Webb. “The Clifford Group Forms a Unitary 3-Design”. In: *Quantum Info. Comput.* 16.15–16 (Nov. 2016), pages 1379–1400. ISSN: 1533-7146 (cited on page 47).

- [317] Dave Wecker, Matthew B. Hastings, and Matthias Troyer. “Progress towards practical quantum variational algorithms”. In: *Phys. Rev. A* 92 (4 Oct. 2015), page 042303. DOI: [10.1103/PhysRevA.92.042303](https://doi.org/10.1103/PhysRevA.92.042303). URL: <https://link.aps.org/doi/10.1103/PhysRevA.92.042303> (cited on page 39).
- [318] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *Journal of Big Data* 3.1 (May 2016). DOI: [10.1186/s40537-016-0043-6](https://doi.org/10.1186/s40537-016-0043-6). URL: <https://doi.org/10.1186/s40537-016-0043-6> (cited on page 56).
- [319] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. *Quantum Perceptron Models*. 2016. DOI: [10.48550/ARXIV.1602.04799](https://doi.org/10.48550/ARXIV.1602.04799). URL: <https://arxiv.org/abs/1602.04799> (cited on page 81).
- [320] Nathan Wiebe et al. *Quantum Language Processing*. 2019. DOI: [10.48550/ARXIV.1902.05162](https://doi.org/10.48550/ARXIV.1902.05162). URL: <https://arxiv.org/abs/1902.05162> (cited on page 55).
- [321] David Wierichs, Christian Gogolin, and Michael Kastoryano. “Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer”. In: *Phys. Rev. Research* 2 (4 Nov. 2020), page 043246. DOI: [10.1103/PhysRevResearch.2.043246](https://doi.org/10.1103/PhysRevResearch.2.043246). URL: <https://link.aps.org/doi/10.1103/PhysRevResearch.2.043246> (cited on page 40).
- [322] David Wierichs et al. “General parameter-shift rules for quantum gradients”. In: *Quantum* 6 (Mar. 2022), page 677. ISSN: 2521-327X. DOI: [10.22331/q-2022-03-30-677](https://doi.org/10.22331/q-2022-03-30-677). URL: <https://doi.org/10.22331/q-2022-03-30-677> (cited on pages 40, 72).
- [323] Mark M. Wilde. 2nd edition. Cambridge University Press, 2017 (cited on pages 118, 143).
- [324] *Wine*. UCI Machine Learning Repository. 1991 (cited on page 137).
- [325] P. Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Elsevier Science, 2014. ISBN: 9780128009536. URL: <https://books.google.it/books?id=PwUongEACAAJ> (cited on pages 53, 55).
- [326] Andreas J. C. Woitzik et al. “Entanglement production and convergence properties of the variational quantum eigensolver”. In: *Phys. Rev. A* 102 (4 Oct. 2020), page 042402. DOI: [10.1103/PhysRevA.102.042402](https://doi.org/10.1103/PhysRevA.102.042402). URL: <https://link.aps.org/doi/10.1103/PhysRevA.102.042402> (cited on pages 140, 141, 207).
- [327] Michael M. Wolf. *Mathematical Foundations of Supervised Learning*. 2021 (cited on page 195).
- [328] K. Wright et al. “Benchmarking an 11-Qubit Quantum Computer”. In: *Nature Communications* 10.1 (Nov. 2019), page 5464. ISSN: 2041-1723. DOI: [10.1038/s41467-019-13534-2](https://doi.org/10.1038/s41467-019-13534-2) (cited on page 33).
- [329] *Xanadu*. <https://www.xanadu.ai/>, 2023 (cited on page 34).
- [330] Tailong Xiao et al. “Quantum Boltzmann machine algorithm with dimension-expanded equivalent Hamiltonian”. In: *Phys. Rev. A* 101 (3 Mar. 2020), page 032304. DOI: [10.1103/PhysRevA.101.032304](https://doi.org/10.1103/PhysRevA.101.032304). URL: <https://link.aps.org/doi/10.1103/PhysRevA.101.032304> (cited on page 81).
- [331] Leo Zhou et al. “Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices”. In: *Phys. Rev. X* 10 (2 June 2020), page 021067. DOI: [10.1103/PhysRevX.10.021067](https://doi.org/10.1103/PhysRevX.10.021067). URL: <https://link.aps.org/doi/10.1103/PhysRevX.10.021067> (cited on page 50).

-
- [332] Tianci Zhou and Adam Nahum. “Emergent statistical mechanics of entanglement in random unitary circuits”. In: *Phys. Rev. B* 99 (17 May 2019), page 174205. DOI: [10.1103/PhysRevB.99.174205](https://doi.org/10.1103/PhysRevB.99.174205). URL: <https://link.aps.org/doi/10.1103/PhysRevB.99.174205> (cited on page 136).
- [333] Marko Žnidarič. “Entanglement of random vectors”. In: *Journal of Physics A: Mathematical and Theoretical* 40.3 (Dec. 2006), F105. DOI: [10.1088/1751-8113/40/3/F04](https://doi.org/10.1088/1751-8113/40/3/F04). URL: <https://dx.doi.org/10.1088/1751-8113/40/3/F04> (cited on page 139).
- [334] Matjaz Zwitter and Milan Soklic. *Breast Cancer*. UCI Machine Learning Repository. 1988 (cited on page 137).
- [335] Karol Życzkowski and Hans-Jürgen Sommers. “Average fidelity between random quantum states”. In: *Phys. Rev. A* 71 (3 Mar. 2005), page 032313. DOI: [10.1103/PhysRevA.71.032313](https://doi.org/10.1103/PhysRevA.71.032313). URL: <https://link.aps.org/doi/10.1103/PhysRevA.71.032313> (cited on page 205).

Appendices

A	Variational Quantum Algorithms	190
A.1	Global and local cost functions	190
A.2	Variance of gradients	191
B	Quantum Machine Learning	194
B.1	Generalisation bound for data-reuploading quantum neural networks	194
C	Continuous Quantum Neuron	199
C.1	Proof of the activation function of the quantum neuron	199
C.2	Noise resilience	199
C.3	Alternative schemes for the data encoding operations	200
D	Entanglement of Quantum Neural Networks	202
D.1	Lower bound on entanglement entropy for unitary 2-designs	202
D.2	Details on Haar entanglement	203
D.3	Triviality of the full entangling map	204
D.4	Expressibility of Parameterised Quantum Circuits	205
D.5	Entanglement scaling with increasing depth	205
D.6	Convergence of MPS simulations	206
D.7	Entanglement evolution during training	207
E	Noise Deconvolution	213
E.1	Kraus Decomposition	213
E.2	Tomographic reconstruction formula for qubits	213
E.3	Noise deconvolution for qubits	214
E.4	Inverse maps of Noise channels	214

A. Variational Quantum Algorithms

A.1 Global and local cost functions

In this Appendix we report the toy model introduced in [57] to show that the global cost functions can lead to the emergence of vanishing gradients, hence the use of local costs is advisable to ensure trainability of the parameterised quantum circuit. Consider a simple tensor-product parameterised ansatz

$$U(\boldsymbol{\theta}) = \bigotimes_{i=1}^n e^{-i\theta_i X/2}, \quad (\text{A.1})$$

and the global and local observables

$$O_G = \mathbb{I}^{\otimes n} - |\mathbf{0}\rangle\langle\mathbf{0}|, \quad O_L = 1 - \frac{1}{n} \sum_{i=1}^n \mathbb{I}^{(1)} \otimes \dots \otimes |0\rangle\langle 0|^{(i)} \otimes \dots \otimes \mathbb{I}^{(n)}, \quad (\text{A.2})$$

where $|\mathbf{0}\rangle\langle\mathbf{0}| = |0\rangle\langle 0|^{\otimes n}$ is the ground state of quantum system of n qubits, and the local cost is given by the sum of terms composed of single-qubit operators $|0\rangle\langle 0|^{(i)}$ acting on the i -th site, and trivially on the others. Then, consider the corresponding cost functions

$$C_G(\boldsymbol{\theta}) = \text{Tr}[O_G V(\boldsymbol{\theta}) |\mathbf{0}\rangle\langle\mathbf{0}| V(\boldsymbol{\theta})^\dagger], \quad C_L(\boldsymbol{\theta}) = \text{Tr}[O_L V(\boldsymbol{\theta}) |\mathbf{0}\rangle\langle\mathbf{0}| V(\boldsymbol{\theta})^\dagger], \quad (\text{A.3})$$

which has the same vanish on the same solution, that is $C_G(\boldsymbol{\theta}) = 0 \iff C_L(\boldsymbol{\theta}) = 0$. By direct calculation the cost functions can be easily shown to be

$$C_G(\boldsymbol{\theta}) = 1 - \prod_{i=1}^n \cos^2 \frac{\theta_i}{2}, \quad C_L(\boldsymbol{\theta}) = 1 - \frac{1}{n} \sum_{i=1}^n \cos^2 \frac{\theta_i}{2}. \quad (\text{A.4})$$

whose partial derivatives amount to

$$\frac{\partial C_G(\boldsymbol{\theta})}{\partial \theta_k} = \frac{\sin \theta_k}{2} \prod_{i \neq k} \cos^2 \frac{\theta_i}{2}, \quad \frac{\partial C_L(\boldsymbol{\theta})}{\partial \theta_k} = \frac{\sin \theta_k}{2n}. \quad (\text{A.5})$$

Assuming that each variational angle is sampled independently from the uniform distribution $\theta_i \sim \text{Unif}[0, 2\pi]$, then the expectation value of the partial derivative vanishes

$$\mathbb{E} \left[\frac{\partial C_G(\boldsymbol{\theta})}{\partial \theta_k} \right] = \mathbb{E} \left[\frac{\sin \theta_k}{2} \right] \prod_{i \neq k} \mathbb{E} \left[\cos^2 \frac{\theta_i}{2} \right] = \int_0^{2\pi} \frac{d\theta_k}{2\pi} \frac{\sin \theta_k}{2} \cdot \left(\int_0^{2\pi} \frac{d\theta}{2\pi} \cos^2 \frac{\theta}{2} \right)^{n-1} = 0, \quad (\text{A.6})$$

and similarly for the local cost $\mathbb{E}[\partial_k C(\boldsymbol{\theta})] = 0$. The variance of the gradients can be calculated in a similar way, thus obtaining

$$\text{Var} \left[\frac{\partial C_G(\boldsymbol{\theta})}{\partial \theta_k} \right] = \mathbb{E} \left[\left(\frac{\partial C_G(\boldsymbol{\theta})}{\partial \theta_k} \right)^2 \right] = \int_0^{2\pi} \frac{d\theta_k}{2\pi} \frac{\sin^2 \theta_k}{4} \cdot \left(\int_0^{2\pi} \frac{d\theta}{2\pi} \cos^4 \frac{\theta}{2} \right)^{n-1} \quad (\text{A.7})$$

$$= \frac{1}{8} \cdot \left(\frac{3}{8} \right)^{n-1} \xrightarrow{n \rightarrow \infty} 0. \quad (\text{A.8})$$

which is *exponentially* vanishing with the number of qubits, the hallmark of barren plateaus. On the contrary, the local cost has a variance which vanish only *polynomially* with the number of qubits, since

$$\text{Var} \left[\frac{\partial C_L(\boldsymbol{\theta})}{\partial \theta_k} \right] = \mathbb{E} \left[\left(\frac{\sin \theta_k}{2n} \right)^2 \right] = \int_0^{2\pi} \frac{d\theta_k}{2\pi} \frac{\sin^2 \theta_k}{4n^2} = \frac{1}{8n^2}. \quad (\text{A.9})$$

This example makes it clear that a careful choice of the cost function is necessary to ensure trainability of the circuit. Indeed, in this case we showed that even a depth one circuit suffer of exponentially vanishing gradients if a global cost function is used. Thus, the use of a local cost function, that is one that only uses local measurements on a subset of the qubits, can ameliorate the barren plateau phenomenon, at least for shallow circuits whose depth L scales logarithmically with the system size n , that is $L \sim \mathcal{O}(\log n)$ [57].

A.2 Variance of gradients

We are interested in evaluating the following expectation values

$$\text{Var}[\partial_k C(\boldsymbol{\theta})] = \mathbb{E}[(\partial_k C(\boldsymbol{\theta}))^2] = -\frac{1}{4} \mathbb{E} \left[\text{Tr} \left[U_A^\dagger O U_A \left[P_k, U_B \rho U_B^\dagger \right] \right]^2 \right], \quad (\text{A.10})$$

under the assumption that the unitary ensembles $\mathbb{U}_{A,B}$ generated by $U_{A,B}$ are 2-designs, so that we can apply the explicit formula for integration over random unitary matrices reported in Eqs. (2.65), (2.67), and (2.66), which we recall also here for $d = 2^n$

$$\mathbb{E}_U[U A U^\dagger] = \int d\mu(U) U A U^\dagger = \frac{\text{Tr}[A] \mathbb{I}}{d} \quad (\text{A.11})$$

$$\begin{aligned} \mathbb{E}_U[A U B U^\dagger C U D U^\dagger] &= \int d\mu(U) A U B U^\dagger C U D U^\dagger \\ &= \frac{\text{Tr}[BD] \text{Tr}[C]A + \text{Tr}[B] \text{Tr}[D]AC}{d^2 - 1} \\ &\quad - \frac{\text{Tr}[BD]AC + \text{Tr}[B] \text{Tr}[C] \text{Tr}[D]A}{d(d^2 - 1)} \end{aligned} \quad (\text{A.12})$$

$$\begin{aligned} \mathbb{E}_U[\text{Tr}[U A U^\dagger B] \text{Tr}[U C U^\dagger D]] &= \int d\mu(U) \text{Tr}[U A U^\dagger B] \text{Tr}[U C U^\dagger D] \\ &= \frac{\text{Tr}[A] \text{Tr}[B] \text{Tr}[C] \text{Tr}[D] + \text{Tr}[AC] \text{Tr}[BD]}{d^2 - 1} \\ &\quad - \frac{\text{Tr}[AC] \text{Tr}[B] \text{Tr}[D] + \text{Tr}[A] \text{Tr}[C] \text{Tr}[BD]}{d(d^2 - 1)} \end{aligned} \quad (\text{A.13})$$

\mathbb{U}_A is a 2-design First, we start with the case when \mathbb{U}_A is a 2-design. Then, by direct application of (A.13) to (A.10), and setting $Q = [P_k, U_B \rho U_B^\dagger]$, one has

$$\begin{aligned} -4\text{Var}_{\mathbb{U}_A}[\partial_k C(\boldsymbol{\theta})] &= \mathbb{E}_{\mathbb{U}_A} \left[\text{Tr} \left[U_A^\dagger O U_A Q \right] \text{Tr} \left[U_A^\dagger O U_A Q \right] \right] \\ &= \frac{\text{Tr}[O]^2 \text{Tr}[Q]^2 + \text{Tr}[O^2] \text{Tr}[Q^2]}{2^{2n} - 1} - \frac{\text{Tr}[O^2] \text{Tr}[Q]^2 + \text{Tr}[O]^2 \text{Tr}[Q^2]}{2^n(2^{2n} - 1)} \\ &= \frac{\text{Tr}[O^2] \text{Tr}[Q^2]}{2^{2n} - 1} - \frac{\text{Tr}[O]^2 \text{Tr}[Q^2]}{2^n(2^{2n} - 1)} \\ &= \frac{1}{2^{2n} - 1} \left(\text{Tr}[O^2] - \frac{\text{Tr}[O]^2}{2^n} \right) \text{Tr} \left[[P_k, U_B \rho U_B^\dagger]^2 \right], \end{aligned} \quad (\text{A.14})$$

where in the second line we made use of Eq. (A.13), and in the third line the terms having $\text{Tr}[Q]^2$ vanish because $\text{Tr}[Q] = 0$ since Q is a commutator.

\mathbb{U}_B is a 2-design The procedure is very similar if \mathbb{U}_B is a 2-design. First, we rearrange the partial derivative as

$$\partial_k C(\boldsymbol{\theta}) = -\frac{i}{2} \text{Tr} \left[U_A^\dagger O U_A [P_k, U_B \rho U_B^\dagger] \right] = -\frac{i}{2} \text{Tr} \left[U_B \rho U_B^\dagger [U_A^\dagger O U_A, P_k] \right] \quad (\text{A.15})$$

and then apply again Eq. (2.67). Setting $Q = [U_A^\dagger O U_A, P_k]$, one then obtains

$$\begin{aligned} -4\text{Var}_{\mathbb{U}_B}[\partial_k C(\boldsymbol{\theta})] &= \mathbb{E}_{\mathbb{U}_B} \left[\text{Tr} \left[U_B \rho U_B^\dagger Q \right] \text{Tr} \left[U_B \rho U_B^\dagger Q \right] \right] \\ &= \frac{\text{Tr}[\rho]^2 \text{Tr}[Q]^2 + \text{Tr}[\rho^2] \text{Tr}[Q^2]}{2^{2n} - 1} - \frac{\text{Tr}[\rho^2] \text{Tr}[Q]^2 + \text{Tr}[\rho]^2 \text{Tr}[Q^2]}{2^n(2^{2n} - 1)} \\ &= \frac{1}{2^{2n} - 1} \left(\text{Tr}[\rho^2] - \frac{1}{2^n} \right) \text{Tr} \left[\left[U_A^\dagger O U_A, P_k \right]^2 \right], \end{aligned} \quad (\text{A.16})$$

where as before terms with $\text{Tr}[Q]$ vanish because Q is a commutator, and in the last line we used $\text{Tr}[\rho] = 1$ by definition of density matrix.

$\mathbb{U}_{A,B}$ are both 2-designs Finally, we analyse what happens when both \mathbb{U}_A and \mathbb{U}_B are 2-designs. First, by setting again $O_A = U_A^\dagger O U_A$, we rewrite explicitly the squared commutator as

$$\text{Tr} \left[[O_A, P_k]^2 \right] = \text{Tr} \left[O_A P_k O_A P_k - O_A P_k^2 O_A - P_k O_A^2 P_k + P_k O_A P_k O_A \right] \quad (\text{A.17})$$

$$= 2 \text{Tr} [O_A P_k O_A P_k] - 2 \text{Tr} [O_A^2 P_k^2], \quad (\text{A.18})$$

and then use Eq. (A.12) to calculate the expectation values of these quantities. The first term amounts to

$$\begin{aligned} \mathbb{E}_{\mathbb{U}_A} \left[\text{Tr} \left[U_A^\dagger O U_A P_k U_A^\dagger O U_A P_k \right] \right] &= \text{Tr} \left[\mathbb{E}_{\mathbb{U}_A} \left[U_A^\dagger O U_A P_k U_A^\dagger O U_A \right] P_k \right] \\ &= \text{Tr} \left[\frac{\text{Tr}[O^2] \text{Tr}[P_k] \mathbb{I} + \text{Tr}[O]^2 P_k}{2^{2n} - 1} P_k \right] - \text{Tr} \left[\frac{\text{Tr}[O^2] P_k + \text{Tr}[O]^2 \text{Tr}[P_k] \mathbb{I}}{2^n(2^{2n} - 1)} P_k \right] \\ &= \frac{\text{Tr}[O^2] \text{Tr}[P_k]^2 + \text{Tr}[O]^2 \text{Tr}[P_k^2]}{2^{2n} - 1} - \frac{\text{Tr}[O^2] \text{Tr}[P_k^2] + \text{Tr}[O]^2 \text{Tr}[P_k]^2}{2^n(2^{2n} - 1)}, \end{aligned} \quad (\text{A.19})$$

and the second term can be calculated using the formula for first-order moments of Haar-random matrices (A.11)

$$\mathbb{E}_{\mathbb{U}_A} \left[\text{Tr} [O_A^2 P_k^2] \right] = \text{Tr} \left[\mathbb{E}_{\mathbb{U}_A} \left[U_A^\dagger O^2 U_A \right] P_k^2 \right] = \text{Tr} \left[\frac{\text{Tr}[O^2] \mathbb{I}}{2^n} P_k^2 \right] = \frac{\text{Tr}[O^2] \text{Tr}[P_k^2]}{2^n}. \quad (\text{A.20})$$

Thus, one can eventually compute the expectation value over $\mathbb{U}_{A,B}$ by combining Eqs. (A.19) and (A.20) with (A.17), and plugging these into Eq. (A.16), obtaining

$$\begin{aligned} -4\text{Var}_{\mathbb{U}_{A,B}}[\partial_k C(\boldsymbol{\theta})] &= -4\mathbb{E}_{\mathbb{U}_A} \left[\mathbb{E}_{\mathbb{U}_B} \left[(\partial_k C(\boldsymbol{\theta}))^2 \right] \right] \\ &= \frac{1}{2^{2n} - 1} \left(\text{Tr}[\rho^2] - \frac{1}{2^n} \right) \mathbb{E}_{\mathbb{U}_A} \left[\text{Tr} \left[\left[U_A^\dagger O U_A, P_k \right]^2 \right] \right] \\ &= \frac{2}{2^{2n} - 1} \left(\text{Tr}[\rho^2] - \frac{1}{2^n} \right) \left(\frac{\text{Tr}[O^2] \text{Tr}[P_k]^2 + \text{Tr}[O]^2 \text{Tr}[P_k^2]}{2^{2n} - 1} \right. \\ &\quad \left. - \frac{\text{Tr}[O^2] \text{Tr}[P_k^2] + \text{Tr}[O]^2 \text{Tr}[P_k]^2}{2^n(2^{2n} - 1)} - \frac{\text{Tr}[O^2] \text{Tr}[P_k^2]}{2^n} \right) \end{aligned} \quad (\text{A.21})$$

Summing up, we summarise Equations (A.14), (A.16), and (A.21), as follows.

If \mathbb{U}_A is at least a 2-design, then $\forall k$:

$$\text{Var}_{\mathbb{U}_A}[\partial_k C(\boldsymbol{\theta})] = -\frac{1}{4} \frac{1}{2^{2n}-1} \left(\text{Tr}[O^2] - \frac{\text{Tr}[O]^2}{2^n} \right) \text{Tr} \left[\left[P_k, U_B \rho U_B^\dagger \right]^2 \right] \quad (\text{A.22})$$

If \mathbb{U}_B is at least a 2-design, then $\forall k$:

$$\text{Var}_{\mathbb{U}_B}[\partial_k C(\boldsymbol{\theta})] = -\frac{1}{4} \frac{1}{2^{2n}-1} \left(\text{Tr}[\rho^2] - \frac{1}{2^n} \right) \text{Tr} \left[\left[U_A^\dagger O U_A, P_k \right]^2 \right] \quad (\text{A.23})$$

If $\mathbb{U}_{A,B}$ are both at least 2-designs, then $\forall k$:

$$\begin{aligned} \text{Var}_{\mathbb{U}_{A,B}}[\partial_k C(\boldsymbol{\theta})] = & -\frac{1}{4} \frac{2}{2^{2n}-1} \left(\text{Tr}[\rho^2] - \frac{1}{2^n} \right) \left(\frac{\text{Tr}[O^2] \text{Tr}[P_k]^2 + \text{Tr}[O]^2 \text{Tr}[P_k^2]}{2^{2n}-1} \right. \\ & \left. - \frac{\text{Tr}[O^2] \text{Tr}[P_k^2] + \text{Tr}[O]^2 \text{Tr}[P_k]^2}{2^n(2^{2n}-1)} - \frac{\text{Tr}[O^2] \text{Tr}[P_k^2]}{2^n} \right) \end{aligned} \quad (\text{A.24})$$

Note that despite the minus signs in front of the expressions, one can verify that all these variances are correctly positive. Indeed, for Eq. (A.22), by Cauchy–Schwarz it holds

$$\text{Tr}[O\mathbb{I}]^2 \leq \text{Tr}[O^\dagger O] \text{Tr}[\mathbb{I}] = 2^n \text{Tr}[O^2] \implies \text{Tr}[O^2] \geq \frac{\text{Tr}[O]^2}{2^n}$$

and similarly for the term involving the purity $\text{Tr}[\rho^2]$ in (A.23), where the latter can also be seen as a consequence of the minimum of the purity of a quantum state being achieved on the completely mixed state. On the contrary, the trace of squared commutators are always negative values, as for any matrices A, B it holds $|\text{Tr}[ABAB]| \leq \text{Tr}[A^\dagger A B B^\dagger]$ [30, 234]. Applying such inequality for the specific case of A, B being Hermitian matrices, yields

$$\text{Tr}[A, B]^2 = 2(\text{Tr}[ABAB] - \text{Tr}[A^2 B^2]) \leq 2(\text{Tr}[A^2 B^2] - \text{Tr}[A^2 B^2]) = 0, \quad (\text{A.25})$$

where in our case A, B are the Hermitian operators $P_k, U_B \rho U_B^\dagger$, and $U_A^\dagger O U_A$, as both P_k and O are Hermitian.

The expressions for the variance take a simpler form when evaluated for the most common case of parameterised gates generated by Pauli rotations P_k , when the observable O is a Pauli string, and the initial state is a pure state, for example $\rho = |0\rangle\langle 0|$. In this case, one has $\text{Tr}[P_k] = \text{Tr}[O] = 0$, and $\text{Tr}[P_k^2] = \text{Tr}[O^2] = \text{Tr}[\mathbb{I}] = 2^n$, because Pauli matrices are traceless and involutory, and also $\text{Tr}[\rho^2] = \text{Tr}[\rho] = 1$, because ρ is a pure state. Then, the equations above greatly simplifies to

If \mathbb{U}_A is at least a 2-design, then $\forall k$:

$$\text{Var}_{\mathbb{U}_A}[\partial_k C(\boldsymbol{\theta})] = \frac{1}{2} \frac{2^n}{2^{2n}-1} \left(1 - \text{Tr} \left[P_k U_B \rho U_B^\dagger P_k U_B \rho U_B^\dagger \right] \right) \quad (\text{A.26})$$

If \mathbb{U}_B is at least a 2-design, then $\forall k$:

$$\text{Var}_{\mathbb{U}_B}[\partial_k C(\boldsymbol{\theta})] = \frac{1}{2} \frac{1}{2^n(2^n+1)} \left(2^n - \text{Tr} \left[P_k U_A^\dagger O U_A P_k U_A^\dagger O U_A \right] \right) \quad (\text{A.27})$$

If $\mathbb{U}_{A,B}$ are both at least 2-designs, then $\forall k$:

$$\text{Var}_{\mathbb{U}_{A,B}}[\partial_k C(\boldsymbol{\theta})] = \frac{2^{2n}}{2(2^n+1)(2^{2n}-1)}. \quad (\text{A.28})$$

For all the cases treated above it is clear that the variance of the gradients vanish exponentially with the number of qubits, namely $\text{Var}[\partial_k C(\boldsymbol{\theta})] \in \mathcal{O}(2^{-n})$.

B. Quantum Machine Learning

B.1 Generalisation bound for data-reuploading quantum neural networks

In this Appendix we show how to derive the generalisation bound for reuploading quantum neural networks as the one shown in Eq. (3.72), using Rademacher complexity as a measure of uniform convergence. In particular, we first start introducing definitions and tools on Rademacher complexity and show how to apply them to linear models with features in Sec. B.1.2. Then, in Sec. B.1.3 we show how to apply these results for the case of data reuploading quantum neural networks.

B.1.1 Rademacher complexity and generalisation error

We now proceed introducing the Rademacher Complexity measure and the so-called Concentration Lemma, needed for the derivation of the generalisation bound. Then, we state the main theorem connecting the Rademacher complexity of an hypothesis class with its generalisation error.

Definition B.1 — Empirical Rademacher Complexity. Let \mathcal{F} a family of functions mapping a data space \mathcal{Z} to $[a, b]$, and $S = (z_1, \dots, z_m) \subset \mathcal{Z}^m$ a fixed sample of size m with elements in \mathcal{Z} . The empirical Rademacher complexity of class \mathcal{F} with respect to the sample S is defined as

$$\mathcal{R}_S(\mathcal{F}) := \frac{1}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^m \sigma_i f(z_i) \right], \quad (\text{B.1})$$

where $\boldsymbol{\sigma} = (\sigma_1, \dots, \sigma_m) \in \{\pm 1\}^m$, with $\sigma_i \sim \text{Unif}\{+1, -1\}$ are independent uniformly distributed binary random variables. The random variables σ_i are called *Rademacher* variables.

More generally, given a set of vectors $A = \{\mathbf{a}_1, \mathbf{a}_2, \dots \mid \mathbf{a}_i \in \mathbb{R}^m\} \subset \mathbb{R}^m$, one defines the Rademacher complexity of the set A as

$$\mathcal{R}(A) := \frac{1}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\mathbf{a} \in A} \sum_{i=1}^m \sigma_i a_i \right]. \quad (\text{B.2})$$

We now state the Contraction lemma regarding the Rademacher complexity of composed functions.

Lemma B.1 — Contraction Lemma (lemma 26.9 in ref. (274)). For each $i \in 1, \dots, m$, let $\ell_i : \mathbb{R} \rightarrow \mathbb{R}$ be a L -Lipschitz function, namely for all $\alpha, \beta \in \mathbb{R}$ we have $|\ell_i(\alpha) - \ell_i(\beta)| \leq L|\alpha - \beta|$. For $\mathbf{a} \in \mathbb{R}^m$ let $\boldsymbol{\ell}(\mathbf{a})$ denote the vector $\boldsymbol{\ell}(\mathbf{a}) = (\ell_1(a_1), \dots, \ell_m(a_m))$. Let $\boldsymbol{\ell} \circ A = \{\boldsymbol{\ell}(\mathbf{a}) \mid \mathbf{a} \in A\}$. Then,

$$\mathcal{R}(\boldsymbol{\ell} \circ A) \leq L \mathcal{R}(A) \quad (\text{B.3})$$

Let $S = \{z_1, \dots, z_m\} \subset \mathcal{Z}^m$ be a collection of *iid* variables sampled from a distribution \mathcal{D} on \mathcal{Z} . In supervised learning scenarios one has $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where elements $z_i = (x_i, y_i)$ are pairs of input data $x_i \in \mathcal{X}$ and corresponding output $y_i \in \mathcal{Y}$. Let \mathcal{M} denote an hypothesis class containing mappings from input to output space $\mathcal{M} \subset \{h \mid h : \mathcal{X} \rightarrow \mathcal{Y}\}$, and let ℓ denote a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$. In addition, let

$$\mathcal{G} = \ell \circ \mathcal{M} := \{z = (x, y) \mapsto \ell(h(x), y) \mid h \in \mathcal{M}\} \quad (\text{B.4})$$

be the class of mappings from the data space \mathcal{Z} to \mathbb{R} , obtained by combining the action of the models $h \in \mathcal{M}$ in the hypothesis class with the loss function ℓ . With $g \in \mathcal{G}$, we define the true risk $L_{\mathcal{D}}(g)$ and the empirical risk $L_S(g)$, as

$$L_{\mathcal{D}}(g) := \mathbb{E}_{z \sim \mathcal{D}}[g(z)] \quad , \quad L_S(g) := \frac{1}{m} \sum_{i=1}^m g(z_i), \quad (\text{B.5})$$

which are defined in terms of the probability distribution \mathcal{D} on data space \mathcal{Z} , the sample set S consisting of m data points, the hypothesis class \mathcal{M} representing the parameterised learning model, and finally the loss function ℓ used to train the model. With these definitions, we are now ready to state the well-known theorem in classical statistical learning theory which bounds the generalisation error of a parameterised model with its Rademacher complexity.

Theorem B.1 — Generalisation Bound via Rademacher Complexity (th. 26.5 in Ref. (274), th. 1.15 in Ref. (327)). Be $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ a data space with arbitrary inputs and outputs spaces, \mathcal{X} and \mathcal{Y} . Consider an hypothesis class $\mathcal{M} \subset \{h : \mathcal{X} \rightarrow \mathcal{Y}\}$, a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, c]$, and define $\mathcal{G} := \{z \mapsto \ell(h(x), y) \mid h \in \mathcal{M}\}$. For any $\delta > 0$, and probability measure \mathcal{D} over \mathcal{Z} , with probability at least $1 - \delta$ over the draw of a training set $S \in \mathcal{Z}^m$ of size m , for all $g \in \mathcal{G}$:

$$L_{\mathcal{D}}(g) - L_S(g) < 2\mathcal{R}_S(\mathcal{G}) + 3c \sqrt{\frac{\log 2/\delta}{2m}} \quad (\text{B.6})$$

B.1.2 Rademacher complexity of Linear Classes

In the following we derive the Rademacher complexity of linear — with respect to the trainable parameters — models where we allow the inputs to be transformed with a feature map. The derivation follows that found in ref. [274], with the difference that we here allow inputs to go through a feature map, and we use a different final bound on the norm. Consider the hypothesis class of parametric linear models

$$\mathcal{M} = \{\mathbf{x} \mapsto \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) \mid \|\mathbf{w}\|_2 \leq B\}, \quad (\text{B.7})$$

where $\boldsymbol{\phi} : \mathbb{R}^d \rightarrow \mathbb{R}^M$ is a general feature map mapping inputs to feature vectors $\mathbf{x} \in \mathbb{R}^d \mapsto \boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^M$, and $\|\cdot\|_2$ denotes the standard Euclidean norm.

Let $S = ((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)) \in (\mathbb{R}^d \times \mathbb{R})^m$ be a training set, and $\ell : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ a L -lipschitz loss function. Then, consider the class

$$\mathcal{G} = \{(\mathbf{x}_i, y_i) \mapsto \ell(h(\mathbf{x}_i), y_i) \mid h \in \mathcal{M}\}. \quad (\text{B.8})$$

The empirical Rademacher complexity of class \mathcal{G} on set S is defined as (see Definition B.1)

$$\mathcal{R}_S(\mathcal{G}) = \frac{1}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{g \in \mathcal{G}} \sum_{i=1}^m \sigma_i g(z_i) \right] \quad (\text{B.9})$$

$$= \frac{1}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{h \in \mathcal{M}} \sum_{i=1}^m \sigma_i \ell(h(\mathbf{x}_i), y_i) \right] \quad (\text{B.10})$$

$$\leq \frac{L}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{h \in \mathcal{M}} \sum_{i=1}^m \sigma_i h(\mathbf{x}_i) \right], \quad (\text{B.11})$$

where in the last line we made use of the Contraction lemma B.1 to remove the dependence on the loss function, by defining the map $\boldsymbol{\ell}(\mathbf{a}) \mapsto (\ell_1(a_1), \dots, \ell_m(a_m)) = (\ell(a_1, y_1), \dots, \ell(a_m, y_m))$ with $\mathbf{a} = (h(\mathbf{x}_1), \dots, h(\mathbf{x}_m))$, and assuming that $\ell(a_i, y_i)$ is L -Lipschitz for all $a_i, y_i, i = 1, \dots, m$.

Substituting the definition of the linear model (B.8) $h(\mathbf{x}_i) = \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i)$, we then have

$$\mathcal{R}_S(\mathcal{G}) \leq \frac{L}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\|\mathbf{w}\|_2 \leq B} \sum_{i=1}^m \sigma_i \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}_i) \right] \quad (\text{B.12})$$

$$= \frac{L}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\|\mathbf{w}\|_2 \leq B} \mathbf{w} \cdot \sum_{i=1}^m \sigma_i \boldsymbol{\phi}(\mathbf{x}_i) \right] \quad (\text{B.13})$$

$$\text{(Cauchy-Schwartz)} \leq \frac{L}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\sup_{\|\mathbf{w}\|_2 \leq B} \|\mathbf{w}\|_2 \left\| \sum_{i=1}^m \sigma_i \boldsymbol{\phi}(\mathbf{x}_i) \right\|_2 \right] \quad (\text{B.14})$$

$$\leq \frac{BL}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\left\| \sum_{i=1}^m \sigma_i \boldsymbol{\phi}(\mathbf{x}_i) \right\|_2 \right] \quad (\text{B.15})$$

$$= \frac{BL}{m} \mathbb{E}_{\boldsymbol{\sigma}} \left[\left(\left\| \sum_{i=1}^m \sigma_i \boldsymbol{\phi}(\mathbf{x}_i) \right\|_2^2 \right)^{1/2} \right] \quad (\text{B.16})$$

$$\text{(Jensen's inequality)} \leq \frac{BL}{m} \left(\mathbb{E}_{\boldsymbol{\sigma}} \left[\left\| \sum_{i=1}^m \sigma_i \boldsymbol{\phi}(\mathbf{x}_i) \right\|_2^2 \right] \right)^{1/2}, \quad (\text{B.17})$$

where in (B.14) we used Cauchy-Schwartz inequality $\mathbf{w} \cdot \mathbf{q} \leq \|\mathbf{w}\|_2 \|\mathbf{q}\|_2$ followed by $\|\mathbf{w}\|_2 \leq B$, and in (B.17) we used Jensen's inequality to move the square root out of the expectation value.

Now, since the Rademacher variables σ_i are independent, the expectation can be calculated as follows

$$\mathbb{E}_{\boldsymbol{\sigma}} \left[\left\| \sum_{i=1}^m \sigma_i \boldsymbol{\phi}(\mathbf{x}_i) \right\|_2^2 \right] = \mathbb{E}_{\boldsymbol{\sigma}} \left[\sum_{i,j=1}^m \sigma_i \sigma_j \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) \right] \quad (\text{B.18})$$

$$= \sum_{i \neq j}^m \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_j) \underbrace{\mathbb{E}_{\boldsymbol{\sigma}}[\sigma_i \sigma_j]}_{=0} + \sum_{i=1}^m \boldsymbol{\phi}(\mathbf{x}_i) \cdot \boldsymbol{\phi}(\mathbf{x}_i) \underbrace{\mathbb{E}_{\boldsymbol{\sigma}}[\sigma_i^2]}_{=1} \quad (\text{B.19})$$

$$= \sum_{i=1}^m \|\boldsymbol{\phi}(\mathbf{x}_i)\|_2^2 \leq m \max_i \|\boldsymbol{\phi}(\mathbf{x}_i)\|_2^2 \quad (\text{B.20})$$

$$\leq mM \max_i \|\boldsymbol{\phi}(\mathbf{x}_i)\|_2^2 \quad (\text{B.21})$$

where in the last line we used that for any vector $\boldsymbol{\phi} \in \mathbb{R}^M$ one can bound the 2-norm with the infinity norm as $\|\boldsymbol{\phi}\|_2 \leq \sqrt{M} \|\boldsymbol{\phi}\|_\infty$, where the infinity norm of a vector is defined as the maximum absolute value of its components, namely $\|\boldsymbol{\phi}\|_\infty = \max(|\phi_1|, \dots, |\phi_M|)$.

Putting it all together, substituting Eq. (B.21) into Eq. (B.17), we have shown that the linear model class (B.8) has

$$\mathcal{R}_S(\mathcal{G}) \leq BL \max_i \|\boldsymbol{\phi}(\mathbf{x}_i)\|_2^2 \sqrt{\frac{M}{m}}. \quad (\text{B.22})$$

B.1.3 Generalisation bound of Quantum Neural Networks

Using the result on linear models with features, we are now ready to prove the generalisation bound shown in Eq. (3.72) for quantum neural networks.

We first show that a data reuploading quantum model can be written in terms of a ‘‘linear’’ model with trigonometric features. A quantum model can be expressed as

$$\text{Tr}[\rho(\mathbf{x}; \boldsymbol{\theta}) O] = f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{\boldsymbol{\omega} \in \Omega} c_{\boldsymbol{\omega}}(\boldsymbol{\theta}) e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \quad (\text{B.23})$$

where the coefficients are such that $c_{-\boldsymbol{\omega}} = c_{\boldsymbol{\omega}}^*$, and the spectrum the spectrum Ω consists of positive and negative frequencies

$$\Omega = \{\Omega^-\} \cup \{\mathbf{0}\} \cup \{\Omega^+\} \quad \text{with} \quad \Omega^- = \{-\boldsymbol{\omega} \mid \boldsymbol{\omega} \in \Omega^+\} \quad (\text{B.24})$$

so that $|\Omega| = 2|\Omega^+| + 1$. Dropping the dependence of the coefficients on $\boldsymbol{\theta}$ for ease of notation, the model can be expressed in terms of real coefficients and trigonometric features as

$$f(\mathbf{x}; \boldsymbol{\theta}) = \sum_{\boldsymbol{\omega} \in \Omega} c_{\boldsymbol{\omega}} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \quad (\text{B.25})$$

$$= \sum_{\boldsymbol{\omega} \in \Omega^-} c_{\boldsymbol{\omega}} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} + c_0 + \sum_{\boldsymbol{\omega} \in \Omega^+} c_{\boldsymbol{\omega}} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \quad (\text{B.26})$$

$$= \sum_{\boldsymbol{\omega} \in \Omega^+} c_{\boldsymbol{\omega}}^* e^{i\boldsymbol{\omega} \cdot \mathbf{x}} + c_0 + \sum_{\boldsymbol{\omega} \in \Omega^+} c_{\boldsymbol{\omega}} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}} \quad (\text{B.27})$$

$$= c_0 + \sum_{\boldsymbol{\omega} \in \Omega^+} (c_{\boldsymbol{\omega}}^* e^{i\boldsymbol{\omega} \cdot \mathbf{x}} + c_{\boldsymbol{\omega}} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}}) \quad (\text{B.28})$$

$$= c_0 + \sum_{\boldsymbol{\omega} \in \Omega^+} ((a_{\boldsymbol{\omega}} - ib_{\boldsymbol{\omega}}) e^{i\boldsymbol{\omega} \cdot \mathbf{x}} + (a_{\boldsymbol{\omega}} + ib_{\boldsymbol{\omega}}) e^{-i\boldsymbol{\omega} \cdot \mathbf{x}}) \quad (\text{B.29})$$

$$= c_0 + \sum_{\boldsymbol{\omega} \in \Omega^+} 2a_{\boldsymbol{\omega}} \cos(\boldsymbol{\omega} \cdot \mathbf{x}) + \sum_{\boldsymbol{\omega} \in \Omega^+} 2b_{\boldsymbol{\omega}} \sin(\boldsymbol{\omega} \cdot \mathbf{x}) \quad (\text{B.30})$$

$$= \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}), \quad (\text{B.31})$$

where we have introduced a Fourier-like feature map $\boldsymbol{\phi} : \mathbb{R}^d \rightarrow \mathbb{R}^{|\Omega|}$ and coefficients vector defined as

$$\mathbf{x} \mapsto \boldsymbol{\phi}(\mathbf{x}) = [1, \cos(\boldsymbol{\omega}_1 \cdot \mathbf{x}), \dots, \cos(\boldsymbol{\omega}_{|\Omega^+|} \cdot \mathbf{x}), \sin(\boldsymbol{\omega}_1 \cdot \mathbf{x}), \dots, \sin(\boldsymbol{\omega}_{|\Omega^+|} \cdot \mathbf{x})], \quad (\text{B.32})$$

and the vector of coefficients $\mathbf{w} \in \mathbb{R}^{|\Omega|}$

$$\mathbf{w} = [c_0, 2a_{\boldsymbol{\omega}_1}, \dots, 2a_{\boldsymbol{\omega}_{|\Omega^+|}}, 2b_{\boldsymbol{\omega}_1}, \dots, 2b_{\boldsymbol{\omega}_{|\Omega^+|}}]. \quad (\text{B.33})$$

Denote with $\mathbf{c} = [c_{\boldsymbol{\omega}}, \boldsymbol{\omega} \in \Omega]$ the vector of complex coefficients in Eq. (B.25), then the norm of the coefficient vectors \mathbf{c} and \mathbf{w} amounts to

$$\|\mathbf{c}\|_2^2 = c_0^2 + 2 \sum_{\boldsymbol{\omega} \in \Omega^+} a_{\boldsymbol{\omega}}^2 + b_{\boldsymbol{\omega}}^2 \quad (\text{B.34})$$

$$\|\mathbf{w}\|_2^2 = c_0^2 + 4 \sum_{\boldsymbol{\omega} \in \Omega^+} a_{\boldsymbol{\omega}}^2 + b_{\boldsymbol{\omega}}^2 = 2\|\mathbf{c}\|_2^2 - c_0^2 \leq 2\|\mathbf{c}\|_2^2. \quad (\text{B.35})$$

Now, let $\mathbf{x} \in [0, 2\pi]^d$, and assume that the frequency spectrum is made of integer frequencies only, that is $\boldsymbol{\omega} \in \mathbb{Z}^d$, $\forall \boldsymbol{\omega} \in \Omega$. In this case, one can prove the so-called *Parseval's equality*, which connects the integral of a function to the the norm of the coefficients in the Fourier expansion of the function. Indeed, by dropping again the dependence on $\boldsymbol{\theta}$ for ease of notation, one can show that

$$\int_{[0, 2\pi]^d} d\mathbf{x} |f(\mathbf{x})|^2 = \int_{[0, 2\pi]^d} d\mathbf{x} \sum_{\boldsymbol{\omega}, \mathbf{v} \in \Omega} c_{\mathbf{v}}^* c_{\boldsymbol{\omega}} e^{-i(\boldsymbol{\omega} - \mathbf{v}) \cdot \mathbf{x}} \quad (\text{B.36})$$

$$= \sum_{\boldsymbol{\omega}, \mathbf{v} \in \Omega} c_{\mathbf{v}}^* c_{\boldsymbol{\omega}} \int_{[0, 2\pi]^d} d\mathbf{x} e^{-i(\boldsymbol{\omega} - \mathbf{v}) \cdot \mathbf{x}} \quad (\text{B.37})$$

$$= \sum_{\boldsymbol{\omega}, \mathbf{v} \in \Omega} c_{\mathbf{v}}^* c_{\boldsymbol{\omega}} \prod_{j=1}^d \int_0^{2\pi} dx^{(j)} e^{-i(\omega^{(j)} - v^{(j)})x^{(j)}} \quad (\text{B.38})$$

$$= \sum_{\boldsymbol{\omega}, \mathbf{v} \in \Omega} c_{\mathbf{v}}^* c_{\boldsymbol{\omega}} (2\pi)^d \delta_{\boldsymbol{\omega}, \mathbf{v}} = (2\pi)^d \sum_{\boldsymbol{\omega} \in \Omega} |c_{\boldsymbol{\omega}}|^2 = (2\pi)^d \|\mathbf{c}\|_2^2, \quad (\text{B.39})$$

where $\delta_{\boldsymbol{\omega}, \mathbf{v}}$ is a Kronecker delta which is one if $\boldsymbol{\omega} = \mathbf{v}$, and zero otherwise.

Since the output of the quantum model is the expectation value of the observable O , the maximum value it can attain is bounded by the operator norm (i.e. the largest eigenvalue) of the observable, namely

$$|f(\mathbf{x}; \boldsymbol{\theta})| = |\text{Tr}[\rho(\mathbf{x}; \boldsymbol{\theta}) O]| \leq \|\rho(\mathbf{x}; \boldsymbol{\theta})\|_1 \|O\|_\infty = \|O\|_\infty. \quad (\text{B.40})$$

where $\|\rho(\mathbf{x}; \boldsymbol{\theta})\|_1 = \text{Tr}[\rho(\mathbf{x}; \boldsymbol{\theta})] = 1$ because $\rho(\mathbf{x}; \boldsymbol{\theta})$ is a density matrix. Eventually, one can bound the 2-norm of the Fourier coefficient vector \mathbf{c} , hence \mathbf{w} , as follows

$$\int_{[0, 2\pi]^d} d\mathbf{x} |f(\mathbf{x})|^2 \leq \int_{[0, 2\pi]^d} d\mathbf{x} \|O\|_\infty^2 = (2\pi)^d \|O\|_\infty^2 \quad (\text{B.41})$$

$$\implies \|\mathbf{c}\|_2 \leq \|O\|_\infty \quad \text{and} \quad \|\mathbf{w}\|_2 \leq 2\|O\|_\infty. \quad (\text{B.42})$$

Summing up, we have shown how to write a quantum reuploading model as a linear model with trigonometric features $f(\mathbf{x}) = \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x})$ (B.31), whose parameter have bounded 2-norm $\|\mathbf{w}\|_2 \leq 2\|O\|_\infty$, so that the results from the previous section on linear models readily apply also to the case of quantum neural networks.

Let \mathcal{M}_{QNN} denote the model class implemented by a quantum neural network with an encoding scheme generating an integer-valued spectrum Ω , obtained by measuring an observable O . Such class and its composition with a L -lipschitz loss function ℓ can then be expressed respectively as (see B.1.2)

$$\mathcal{M}_{\text{QNN}} = \left\{ \mathbf{x} \mapsto \mathbf{w} \cdot \boldsymbol{\phi}(\mathbf{x}) \mid \|\mathbf{w}\|_2 \leq 2\|O\|_\infty, \mathbf{w} \in \mathbb{R}^{|\Omega|} \text{ as in (B.33)}, \right. \\ \left. \boldsymbol{\phi} : [0, 2\pi]^d \rightarrow \mathbb{R}^{|\Omega|} \text{ as in (B.32)} \right\}, \quad (\text{B.43})$$

and

$$\mathcal{G}_{\text{QNN}} = \ell \circ \mathcal{H}_{\text{QNN}} = \{(x_i, y_i) \mapsto \ell(h(\mathbf{x}_i), y_i) \mid h \in \mathcal{M}_{\text{QNN}}\}. \quad (\text{B.44})$$

Since the feature map $\boldsymbol{\phi}$ consists of trigonometric functions of the input data, it holds that $\boldsymbol{\phi}(\mathbf{x}_i) \in [-1, 1]^{|\Omega|} \forall i = 1, \dots, m$. Hence $\max_i \|\boldsymbol{\phi}(\mathbf{x}_i)\|_\infty = 1$, achieved on the first element of the vector, it being the constant feature 1. Thus, finally, using (B.22) on class \mathcal{G}_{QNN} , one has

$$\mathcal{R}_S(\mathcal{G}_{\text{QNN}}) \leq 2\|O\|_\infty L \sqrt{\frac{|\Omega|}{m}}, \quad (\text{B.45})$$

and plugging this in the generalisation theorem (B.6), one finally has the desired generalisation bound.

Theorem B.2 — Generalisation Bound for Quantum Neural Networks (see also Theorem 6 in ref. (53)). Be $\mathcal{Z} = [0, 2\pi]^d \times \mathbb{R}$ a data space of inputs and outputs. Consider a data reuploading quantum circuit whose encoding scheme generates an integer-valued spectrum Ω , whose model class is $\mathcal{M}_{\text{QNN}} := \{\mathbf{x} \mapsto \text{Tr}[\rho(\mathbf{x}; \boldsymbol{\theta}) O] = \sum_{\boldsymbol{\omega} \in \Omega} c_{\boldsymbol{\omega}} e^{-i\boldsymbol{\omega} \cdot \mathbf{x}}\}$. Be $\ell : \mathbb{R} \times \mathbb{R} \rightarrow [0, c]$ an L -Lipschitz loss function and define $\mathcal{G}_{\text{QNN}} := \{z = (x, y) \mapsto \ell(h(x), y) \mid h \in \mathcal{M}_{\text{QNN}}\}$. For any $\delta > 0$ and probability measure \mathcal{D} over \mathcal{Z} , with probability at least $1 - \delta$ over the drawn of a training set $S \in \mathcal{Z}^m$ of size m , for all $g \in \mathcal{G}_{\text{QNN}}$:

$$L_{\mathcal{D}}(g) - L_S(g) < 4\|O\|_\infty L \sqrt{\frac{|\Omega|}{m}} + 3c \sqrt{\frac{\log 2/\delta}{2m}} \quad (\text{B.46})$$

C. Continuous Quantum Neuron

C.1 Proof of the activation function of the quantum neuron

Consider a collection of complex numbers $z_i = r_i e^{i\gamma_i} \in \mathbb{C}$, $i = 1, \dots, N$. The squared modulus of their sum can be explicitly calculated as follows

$$\left| \sum_{i=1}^N z_i \right|^2 = \left(\sum_{i=1}^N z_i \right) \left(\sum_{j=1}^N z_j^* \right) = \sum_{i,j=1}^N z_i z_j^* = \sum_{i=j}^N |z_i|^2 + \sum_{i \neq j}^N r_i r_j e^{i(\gamma_i - \gamma_j)} \quad (\text{C.1})$$

$$= \sum_{i=j}^N r_i^2 + 2 \sum_{i < j}^N r_i r_j \cos(\gamma_j - \gamma_i), \quad (\text{C.2})$$

where in the last line the following relation has been applied

$$e^{ix} + e^{-ix} = 2 \cos(x). \quad (\text{C.3})$$

Setting $r_i = 1/N$ and $\gamma_i = \theta_i - \phi_i$, one obtains

$$\left| \sum_{i=1}^N \frac{e^{i(\theta_i - \phi_i)}}{N} \right|^2 = \frac{1}{N} + \frac{2}{N^2} \sum_{i < j}^N \cos((\theta_j - \phi_j) - (\theta_i - \phi_i)), \quad (\text{C.4})$$

which correctly reduces to Eq. (4.6) in the main text upon substituting $N = 2^n$ and shifting the summation indices to start from zero.

C.2 Noise resilience

Consider an input vector θ equal to the weight vector $\varphi = \theta$, and suppose that the input is corrupted by some noise and transformed into $\theta \rightarrow \theta + \Delta = \varphi + \Delta$, with $\Delta = (\Delta_0, \Delta_1, \dots, \Delta_{2^n-1})$. The activation function in Eq. (4.6) thus reduces to

$$f(\Delta) = \frac{1}{2^n} + \frac{1}{2^{2n-1}} \sum_{i < j}^{2^n-1} \cos(\Delta_j - \Delta_i). \quad (\text{C.5})$$

Assuming that the noise values Δ_i are sampled from the uniform distribution

$$p(\Delta_i) = \frac{1}{a} \quad \text{for } \Delta_i \in \left[-\frac{a}{2}, \frac{a}{2} \right], \quad a \in \mathbb{R}, \quad (\text{C.6})$$

it is possible to evaluate the *average* activation function as

$$\begin{aligned} \mathbb{E}_\Delta[f(\Delta)] &= \frac{1}{2^n} + \frac{1}{2^{2n-1}} \sum_{i < j}^{2^n-1} \mathbb{E}_\Delta[\cos(\Delta_j - \Delta_i)] \\ &= \frac{1}{2^n} + \frac{1}{2^{2n-1}} \frac{2^n(2^n-1)}{2} \langle \cos(\Delta_j - \Delta_i) \rangle \end{aligned} \quad (\text{C.7})$$

where in the last line used that $\mathbb{E}_\Delta[\cos(\Delta_j - \Delta_i)]$ is the same for all i, j . Averaging then yields

$$\begin{aligned} \mathbb{E}_\Delta[\cos(\Delta_j - \Delta_i)] &:= \int_{-\frac{a}{2}}^{\frac{a}{2}} \int_{-\frac{a}{2}}^{\frac{a}{2}} \cos(\Delta_j - \Delta_i) \frac{d\Delta_i d\Delta_j}{a^2} \\ &= 2 \left(\frac{1 - \cos(a)}{a^2} \right). \end{aligned} \quad (\text{C.8})$$

Substituting back into (C.7) one obtains

$$\mathbb{E}_{\Delta}[f(\Delta)] = \frac{1}{2^n} + \frac{2^n - 1}{2^{2n-1}} \left(\frac{1 - \cos(a)}{a^2} \right). \quad (\text{C.9})$$

Consider now the case where the input and weight vectors are different $\theta \neq \phi$, and we ask again how much does the activation function change if the input is corrupted by the presence of noise. As before, considering an input $\theta \rightarrow \theta + \Delta$, the activation function reads

$$f(\theta, \phi, \Delta) = \frac{1}{2^n} + \frac{1}{2^{2n-1}} \sum_{i < j}^{2^n - 1} \cos(A_{ij} + D_{ij}). \quad (\text{C.10})$$

with $A_{ij} = (\theta_j - \phi_j) - (\theta_i - \phi_i)$, and $D_{ij} = \Delta_j - \Delta_i$. Since $\cos(A_{ij} + D_{ij}) = \cos(A_{ij})\cos(D_{ij}) + \sin(A_{ij})\sin(D_{ij})$, and $\mathbb{E}_{\Delta}[\sin(D_{ij})] = 0$ for uniformly distributed noise (C.6), the activation function finally results in

$$\langle f(\theta, \phi, \Delta) \rangle = \frac{1}{2^n} + \frac{D}{2^{2n-1}} \sum_{i < j}^{2^n - 1} \cos(A_{ij}). \quad (\text{C.11})$$

with $D = 2(1 - \cos(a))/a^2$. A more realistic noise model would consist of a Gaussian distribution centred in zero with width $\sigma = a/2$, namely

$$p(\Delta_i) = \frac{1}{\sqrt{2\pi}(a/2)^2} e^{-\frac{\Delta_i^2}{2(a/2)^2}}. \quad (\text{C.12})$$

By repeating the same procedure above one obtains

$$\mathbb{E}_{\Delta}[\cos(\Delta_i - \Delta_j)] = \frac{2}{\pi a^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-\sqrt{2}(\Delta_i + \Delta_j)^2/a^2} \cos(\Delta_i - \Delta_j) d\Delta_i d\Delta_j = e^{-a^2/4}, \quad (\text{C.13})$$

which is comparable to the uniform distribution noise model, but less effective in terms of noise resilience due to the Gaussian tails having the net effect of lowering the mean activation. Nonetheless, this qualitative behaviour proves the quantum neuron to have some internal degree of noise resilience.

C.3 Alternative schemes for the data encoding operations

Several strategies can be envisioned to reduce the time complexity of the proposed quantum neuron algorithm. First, the encoding of input data could be effectively replaced by a direct call to a quantum memory, such as a qRAM [111]. In this case, the information to be analyzed would be directly stored in the form of quantum states coming, e.g. from quantum internet applications or quantum simulators.

Alternatively, one could make use of some specific properties of the LME states arising from the phase encoding procedure used in the main text. Indeed, let U_{ψ} be the unitary operation whose action is to create an LME state from a blank register, i.e $U_{\psi} |\mathbf{0}\rangle = |\psi\rangle$. It is then easy to check that, for $W_k = U_{\psi} Z_k U_{\psi}^{\dagger}$, where Z_k is the Pauli-Z operator acting on the k -th qubit, it holds that $W_k |\phi\rangle = |\phi\rangle \forall k$ if and only if $|\phi\rangle = |\psi\rangle$. This means that the operators $\{W_1, W_2, \dots, W_n\}$ stabilise the state $|\psi\rangle$. Depending on the values of the phases $\{\alpha_i\}_{i=0}^{2^n-1}$, these operators $\{W_k\}_{k=0}^{n-1}$ may be quasi-local, meaning that they only act on a smaller subsystem of the whole n -qubit register. In this case, it can be shown [167, 169] that there exists a quantum dissipative process for which $|\psi\rangle$ is the only stationary state. Of course, this property strongly depends on the nature of the phases α_i of the target LME state, i.e., correlations in the phases are directly related to a specific preparation scheme. However, it might be the case that for some special classes of incoming inputs, a clear a priori

correlation exists between the phases, which would allow to replace the “brute force” approach used in the main text with a more efficient preparation scheme.

Finally, it is worth mentioning that more general strategies to load probability distributions and classical datasets on quantum states are known in the literature [119, 266], whose application could be investigated also in the present case.

Additional room for improvement could be represented by a more efficient implementation of the inner product operation U_w 4.2. In this case, instead of simply inverting the preparation circuit, one could devise a variational circuit optimised to output the desired result (4.5) using the approach shown in Figure C.1. There $V_w(\boldsymbol{\phi}, \boldsymbol{\omega})$ is an operator approximating the unitary U_w depending on variational parameters $\boldsymbol{\omega}$ and the weights of neuron $\boldsymbol{\phi}$, and $\mathcal{L}(\boldsymbol{\omega}; \langle \psi_w | \psi_i \rangle)$ is a cost function evaluating the distance between the output of a the circuit using $V_w(\boldsymbol{\phi}, \boldsymbol{\omega})$, and the desired inner product $\langle \psi_w | \psi_i \rangle$. Optimisation of the trainable parameters $\boldsymbol{\omega}$ could yield an approximate yet efficient implementation for evaluating the inner product gate U_w .

It is also interesting to notice that such optimization procedure could in principle be carried out in combination with a supervised learning approach in order to simultaneously train the value of the weight vector and the actual quantum circuit realisation of the required operation. Finally, if an efficient preparation scheme exists for both the quantum input state $|\psi_i\rangle$ and the quantum weight state $|\psi_w\rangle$, their inner product could also be evaluated by means of specialised algorithms, such as the SWAP test or the Bell basis algorithm [64].

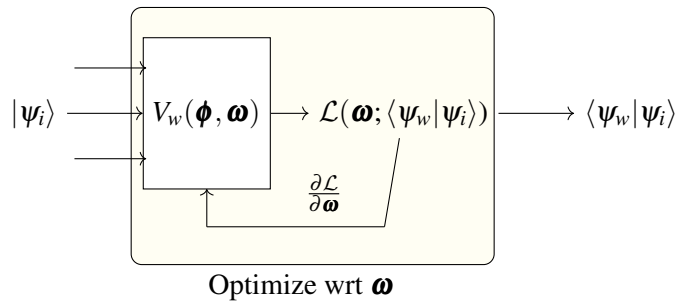


Figure C.1: Scheme of a variational approach for implementing the U_w operation in the quantum neuron model shown in Fig. 4.2.

D. Entanglement of Quantum Neural Networks

D.1 Lower bound on entanglement entropy for unitary 2-designs

The presented derivation is a straightforward application of known results on the entanglement of random states and properties of Rényi-entropies [184, 258]. The Rényi α -entropies of a density operator ρ are defined as

$$S_\alpha(\rho) := \frac{1}{1-\alpha} \log \text{Tr}[\rho^\alpha], \quad (\text{D.1})$$

where $\lim_{\alpha \rightarrow 1} S_\alpha(\rho) = S(\rho)$ is the Von Neumann entropy of Eq. (7.2), and it holds that $S_\beta(\rho) \leq S_\alpha(\rho)$ for $\beta \geq \alpha$. Of particular interest is the Rényi 2-entropy $S_2(\rho) = -\log \text{Tr}[\rho^2]$ depending on the purity $\text{Tr}[\rho^2]$ of the system, which is much easier to compute and it can be used to lower bound the Von Neumann entropy via $S(\rho) > S_2(\rho)$.

Indeed, let $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$ be the state of a composite system made of subsystems A and B with dimensions $d_A = 2^{n_A}$ and $d_B = 2^{n-n_A}$ respectively, and suppose that $|\psi\rangle$ is a random state $|\psi\rangle = U |\psi_0\rangle$, where U is sampled from an ensemble of unitaries that constitutes at least a unitary 2-design. Then, the average value of the purity of the reduced density matrix $\rho_A = \text{Tr}_B[|\psi\rangle\langle\psi|]$ amounts to [184, 258]

$$\mathbb{E}_{2\text{-design}} \text{Tr}[\rho_A^2] = \frac{d_A + d_B}{d_A d_B + 1}. \quad (\text{D.2})$$

By the convexity of Rényi-entropies with respect to $\text{Tr}[\rho^\alpha]$, and using Jensen's inequality (that is, $\mathbb{E} f \geq f \mathbb{E}$), one can lower bound the average Rényi 2-entropy as

$$\mathbb{E}_{2\text{-design}}[S_2(\rho_A)] \geq -\log \mathbb{E}_{2\text{-design}}[\rho_A^2] \quad (\text{D.3})$$

and consequently

$$\mathbb{E}_{2\text{-design}}[S_2(\rho_A)] \geq -\log \frac{d_A + d_B}{d_A d_B + 1} > \log d_A - \log \frac{d_A + d_B}{d_B} > \log d_A - 1. \quad (\text{D.4})$$

Then, since $S(\rho) = S_1(\rho) \geq S_2(\rho) \forall \rho$, taking the expectation value on both sides yields a lower bound on the average Von Neumann entropy of ρ_A , namely

$$\log d_A - 1 < \mathbb{E}_{2\text{-design}}[S_2(\rho_A)] \leq \mathbb{E}_{2\text{-design}}[S(\rho_A)] \leq \log d_A. \quad (\text{D.5})$$

which is the bound shown in Eq. (7.12) in the main text.

If the state $|\psi\rangle$ is instead a truly Haar-random state, that is U is sampled from the uniform Haar distribution and not just from a 2-design, the entanglement entropy is given by the Page value of Eq. (7.8) in the main text, which is itself lower bounded by [132]

$$\mathbb{E}_{\text{Haar}}[S(\rho_A)] > \log d_A - \frac{1}{2} \frac{d_A}{d_B} > \log d_A - \frac{1}{2} \quad (d_A < d_B). \quad (\text{D.6})$$

Summarising, for $d_A < d_B$, putting together the bounds (D.5) and (D.6) one has

$$\log d_A - 1 < \mathbb{E}_{2\text{-design}}[S(\rho_A)] < \log d_A \quad (\text{D.7})$$

$$\log d_A - \frac{1}{2} < \mathbb{E}_{\text{Haar}}[S(\rho_A)] < \log d_A, \quad (\text{D.8})$$

Alternatively, in the limit when the subsystem B is much larger than A , $d_B \gg d_A$, then by approximating the logarithm $\log(1+x) \approx x$ in (D.4) one also has

$$\log d_A - \frac{d_A}{d_B} < \mathbb{E}_{2\text{-design}}[S(\rho_A)] < \log d_A \quad (\text{D.9})$$

$$\log d_A - \frac{1}{2} \frac{d_A}{d_B} < \mathbb{E}_{\text{Haar}}[S(\rho_A)] < \log d_A, \quad (\text{D.10})$$

Thus, the entanglement entropy of a state sampled from a 2-design is close to that of a truly Haar-random state, with both achieving near-maximal entanglement. Of course, one also expects the Von Neumann entropy of a general t -design to be upper bounded by the Page value, $\mathbb{E}_{t\text{-design}}[S(\rho_A)] < \mathbb{E}_{\text{Haar}}[S(\rho_A)]$, with equality obtained in the limit $t \gg 1$.

D.2 Details on Haar entanglement

While Eq. (7.8) is the theoretical definition of the Haar entanglement entropy, it is not possible to exactly compute it due to the exponential number of terms in the sum. However, it is possible to exploit the similarity of the sum with the harmonic series to obtain a good approximation.

Indeed, first denote with H_n the truncated harmonic series

$$H_n := \sum_{k=1}^n \frac{1}{k}. \quad (\text{D.11})$$

Then, rewrite the sum in Eq. (7.8) in a more convenient form

$$\sum_{j=d_B+1}^{d_A d_B} \frac{1}{j} = \sum_{j=1}^{d_A d_B} \frac{1}{j} - \sum_{j=1}^{d_B} \frac{1}{j} = H_{d_A d_B} - H_{d_B}, \quad (\text{D.12})$$

where each term can be approximated effectively using a well-known result for truncated Harmonic series [130], namely

$$H_n = \log n + \gamma + \frac{1}{2n} - \varepsilon_n, \quad (\text{D.13})$$

where $\gamma \simeq 0.5772$ is the Euler-Mascheroni constant, and $0 \leq \varepsilon_n \leq 1/8n^2$. Thus, the correction ε_n goes to zero as the number of terms in the sum n increases, allowing for a meaningful approximation of the value. Using this technique, we are able to estimate the Haar entanglement entropy of a 50-qubit state with an error of the order 10^{-16} .

We now proceed to compute the maximum and average of the distribution with a fixed number of qubits n . Using Eq. 7.8 and recalling $d_{A(B)} = 2^{n_{A(B)}}$, $n_B = n - n_A$, $n_A \in [1, n/2]$ we can write:

$$\mathbb{E}[S(\psi_A)] = H_{d_A d_B} - H_{d_B} - \frac{d_A - 1}{2d_B} \quad (\text{D.14})$$

$$= H_{2^n} - H_{2^{n-n_A}} - \frac{2^{n_A} - 1}{2^{n-n_A+1}} \quad (\text{D.15})$$

$$= \log 2^{n_A} - \frac{2^{n_A} - 1}{2^{n-n_A+1}} + \mathcal{O}\left(\frac{1}{2^{n-n_A}}\right). \quad (\text{D.16})$$

We are now interested in the maximum and average of the distribution. It is easy to see that the maximum is achieved for $n_A = n/2$. In this scenario, when $2^{n_A} \gg 1$, one then has

$$\max_A (\mathbb{E}[S(\psi_A)]) = \frac{n}{2} \log 2 - \frac{1}{2} + \mathcal{O}\left(\frac{1}{2^{n/2}}\right). \quad (\text{D.17})$$

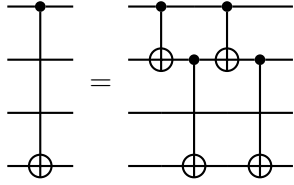
Taking into account that for an n -qubit system the maximum of the entanglement entropy is $S = \frac{n}{2} \log 2$ we can state that, in the large n limit, a Haar state presents a maximally entangled bond.

Algorithm 3: Full (or all-to-all) entangling map

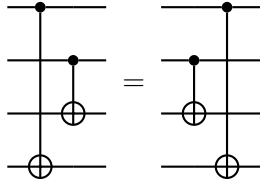
Data: q_1, \dots, q_n , qubits
Result: Quantum circuit
for $i = 1, \dots, n$ **do**
 for $j = i, \dots, n$ **do**
 CNOT $_{q_i, q_j}$;
 end
end

D.3 Triviality of the full entangling map

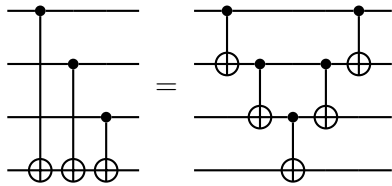
The full (or all-to-all) entangling map defined in Alg. 3 can be shown to be equivalent to a nearest neighbours entangling map with the gates in reversed order, see Fig. D.1. The proof is straightforward and obtained by direct evaluation, making use of some circuit identities for networks of CNOTs [105]. In particular, (i) a CNOT can be distributed into four CNOTs acting on an additional intermediate qubit



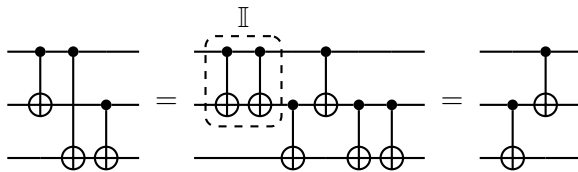
(ii) CNOTs having different controls and targets commute with each other



(iii) a cascade of CNOTs can be decomposed as



The full entangling map can be highly simplified using these three rules, reducing it to a simple sequence of nearest-neighbors interactions. For example, for $n = 3$ qubits, using (i) to distribute the long-range CNOT, one obtains



The simplification process can be iterated for a higher number of qubits by first commuting long range CNOTs at the end of the circuit to create a final cascade, and then making use of the result

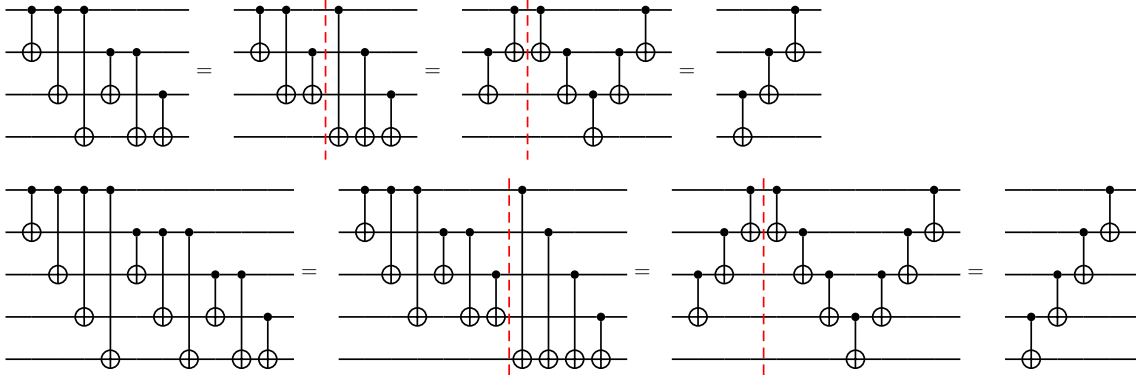


Figure D.1: Equivalence of the full entangling map with a nearest-neighbours scheme. Using the circuit identities discussed in the main text, it is straightforward to check that the all-to-all entangling scheme as defined in Alg. 3 is equivalent to a nearest-neighbours interaction.

from the lower dimension case. In Fig. D.1 the simplification process for $n = 4, 5$ qubits is explicitly shown, and it is directly generalised for all numbers of qubits.

Clearly, these results only hold for networks composed of plain CNOTs, and do not apply for general two-qubit interactions made of controlled unitaries.

D.4 Expressibility of Parameterised Quantum Circuits

The expressibility introduced in [281] quantifies how well the QNN is able to explore the unitary space by comparing the distribution of fidelities of states generated by the QNN with that of randomly Haar-distributed ones.

Let $U(\boldsymbol{\varphi})$ be the unitary operation implemented by a parameterized quantum circuit with parameters $\boldsymbol{\varphi}$, and let $|\psi_{\boldsymbol{\varphi}}\rangle = U(\boldsymbol{\varphi})|\mathbf{0}\rangle$. Given two realizations of the quantum circuit with parameters $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$, consider the fidelity $F = |\langle\psi_{\boldsymbol{\varphi}_1}|\psi_{\boldsymbol{\varphi}_2}\rangle|^2$. By repeatedly sampling two sets of parameters and evaluating the corresponding fidelity F , one can construct a histogram approximating the probability distribution $\hat{\mathcal{P}}(F)$ of the fidelity for states generated by the considered parameterised quantum circuit.

For truly Haar random quantum states, the probability density function of fidelity is known and amounts to $P_{\text{Haar}}(F) = (N-1)(1-F)^{N-1}$, where $N = 2^n$ is the dimension of the Hilbert space [335]. The expressibility is then defined as the Kullback–Leibler divergence D_{KL} between the estimated fidelity distribution and that of a Haar-distributed ensemble, namely

$$\text{Expressibility} := D_{KL}(\hat{\mathcal{P}}_{\text{PQC}}(F) || P_{\text{Haar}}(F)). \quad (\text{D.18})$$

D.5 Entanglement scaling with increasing depth

In Figure D.2 we show the behaviour of the total entanglement S_{tot} defined in Eq. (7.15) for four different QNNs, and as the depth of the quantum circuit is increased. Note that each QNN is considered with all the three possible entangling topologies (*linear*, *circular* and *full* as defined in Fig. 7.1), and the results are shown for several numbers of qubits $n = 4, 6, 8, 10, 12$. At last, note that all QNNs leverage the same variational form $V = C_2$, while the feature map is changed, $\mathcal{F} = C_{ZZ}, C_2, C_3, C_1$ for panels (a), (b), (c) and (d), respectively. See main text for comments on results.

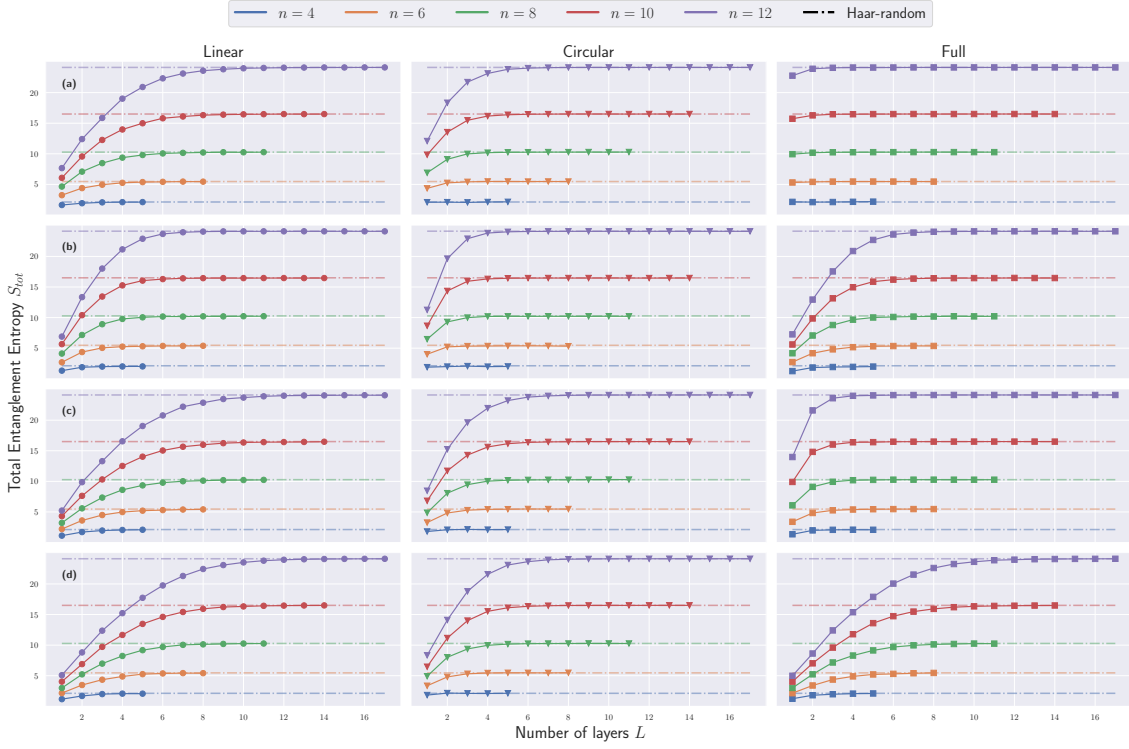


Figure D.2: Total entanglement S_{tot} (7.15) for four different QNN architectures, each evaluated with different entangling topologies (*linear*, *circular* and *full*), shown for increasing number of layers L and for several numbers of qubits n . QNNs architecture given by: (a) $\mathcal{F} = C_{ZZ}$, $V = C_2$; (b) $\mathcal{F} = C_2$, $V = C_2$; (c) $\mathcal{F} = C_3$, $V = C_2$; (d) $\mathcal{F} = C_1$, $V = C_2$. Note that QNNs leverage the same variational form V , while the feature map \mathcal{F} is changed. See the main text for a discussion of the results.

D.6 Convergence of MPS simulations

Using tensor network methods, MPS in this case, it is possible to approximately simulate large qubit systems, and in our analysis we go up to simulating circuits of $n = 50$ qubits.

The error introduced by the approximations can be monitored and so one has always an estimate of the faithfulness of the tensor network simulation [150]. Let $|\psi_{\text{exact}}\rangle$ be the true state of the quantum system after the i -th two qubit gates in the circuit is applied (one qubit gates do not imply approximation errors), and let $|\psi_{\text{trunc}}\rangle$ denote the truncated quantum state represented by the MPS. The fidelity between these two states evaluated on the i -th step of the computation is

$$F_i = |\langle \psi_{\text{exact}} | \psi_{\text{trunc}} \rangle|^2 = \left| \sum_{\alpha=1}^{\chi_{\text{exact}}} \lambda_{\alpha} \langle \xi_{\alpha} |_{1} \otimes \langle \eta_{\alpha} |_{2} \sum_{\beta=1}^{\chi_s} \lambda_{\beta} | \xi_{\beta} \rangle_{1} \otimes | \eta_{\beta} \rangle_{2} \right|^2 \quad (\text{D.19})$$

$$= \left| \sum_{\alpha=1}^{\chi_s} \lambda_{\alpha}^2 \right|^2 = \left| 1 - \sum_{\alpha=\chi_s+1}^{\chi_{\text{exact}}} \lambda_{\alpha}^2 \right|^2, \quad (\text{D.20})$$

where we represented the states in the Schmidt decomposition with respect to the bond where the i -th two-qubit gate was applied, and χ_s is the bond dimension of the MPS state. The fidelity F_i of the simulation after application of the t -th two-qubit gate is lower bounded by the product of the previous fidelities F_i , as [150]

$$F_t \geq \prod_{i=1}^{t-1} F_i. \quad (\text{D.21})$$

where we note that the single step fidelities F_i are readily accessed during the MPS simulation, since one calculates the fidelity before the truncation of the singular values takes place. Equation (D.21) gives a lower bound to the error introduced by truncation in terms of the fidelity between the true state and the one evolved using an MPS simulation, and one can then control the faithfulness of the simulation at any given time step of the circuit.

In Figure D.3 we show the infidelity $1 - F$ of the final state from the circuit for $n = 30, 50$ with a maximum bond dimension $\chi_s = 4096$. The plotted result is the average over $M = 10$ realisation of the quantum circuit with different sets of parameters. Defining reliable results with the infidelity of at most $1 - F = 10^{-4}$ we observe that, for $n = 50$, we describe reliably circuits up to $L = 11$ layers, while for $n = 30$ we can reach $L = 12$ layers.

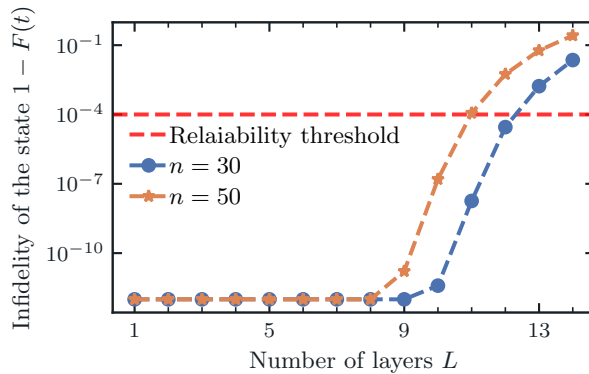


Figure D.3: We show the infidelity of the state as a function of the number of layers for $n = 30, 50$ qubits. The results are reliable up to $L = 11$ layers for $n = 50$ and up to $L = 12$ for $n = 30$.

D.7 Entanglement evolution during training

In this section we discuss some preliminary investigation regarding the evolution of entanglement entropy while training a quantum neural network. Indeed, the focus of Chapter 7 was the study of the relationship between entanglement growth and depth in common parameterized quantum ansätze, specifically when they are initialised with uniform random parameters and no optimization has yet started. Though the key ingredient in variational quantum algorithms is the learning procedure, and it is natural to ask what role plays the entanglement — if any — during the optimization process.

In combinatorial problems based on QAOA [100] the final state of the quantum circuit should correspond to the specific bitstring solving the binary optimization problem at hand. One then expects that as the number of layers in QAOA is increased, the entanglement may first grow but then decrease when the circuit is deep enough to find the correct solution [90]. While theoretically sound, this situation is not always met in real instances [61, 90], and the role of entanglement for the performance and classical simulability of QAOA is still an active area of research [89, 288].

However, differently from binary optimization problems or ground state solver [326], in general, there is no *a priori* structure that can be used to assess the entanglement properties of states coming from optimised quantum neural networks, independently of their depth. While the use of deep QNNs could offer some optimization advantages due to overparametrization [9, 162, 172], randomness-induced barren plateaus can hinder the training of these circuits altogether [57, 202]. Current proposals then advocate for the use of constrained quantum ansätze specifically tailored to the problem under investigation [173, 208, 284], and then one expects the impact of depth to be highly dependant on the specific task to be solved, and dataset to fit, either classical or quantum [275]. The general impact of circuit depth on the accuracy quantum machine learning model is still not fully established, let alone the entanglement features of the quantum states

generated by the model.

As we are specifically focused on the entanglement properties of quantum neural networks, in the following we start to shed light on the relation between entanglement and optimization by considering a fixed depth circuit and studying how entanglement entropy evolves during training. Specifically, we reproduce the classification task of the well-known IRIS dataset [103] proposed by Abbas et al. in [4] to study the expressivity of quantum machine learning models, but instead focus our attention on its entanglement features.

In Fig. D.4 we show the results of training the QNNs shown in panel (c), to classify the first two classes of the IRIS dataset, consisting of $m = 100$ samples of normalised four-dimensional inputs vectors, whose features distribution is shown in panel (a). We refer to Sec. D.7.1 for extended details on the preprocessing of the dataset, the choice of the ansatz, and the optimization process. The training procedure is run 100 times starting from different initialisations of the parameters, thus obtaining multiple training trajectories which are plotted in Fig. D.4b.

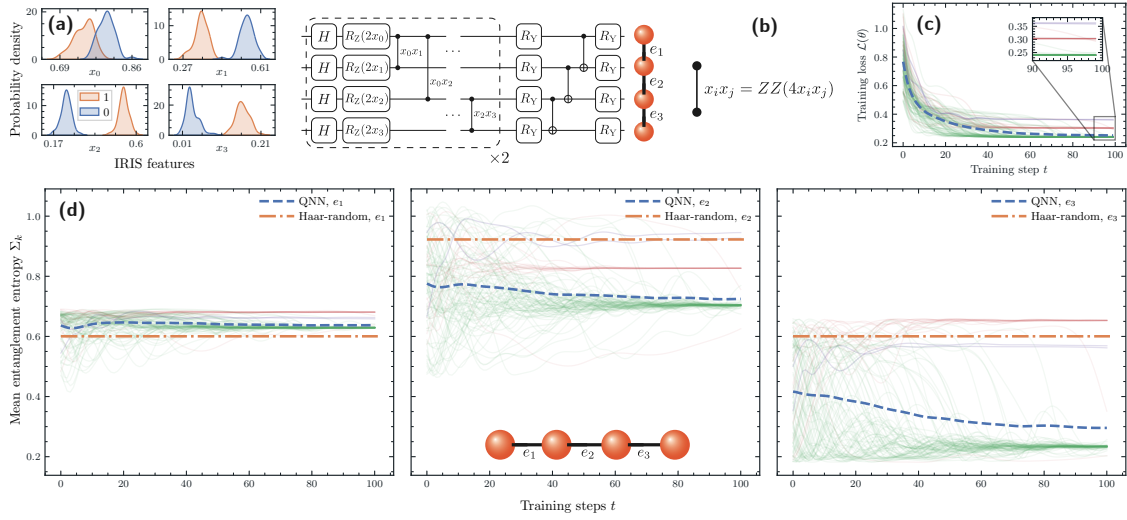


Figure D.4: Evolution of entanglement entropy while training a parameterized quantum circuit to classify the IRIS dataset, using the same setup proposed in [3, 4]. Extended details on the preprocessing of the data, the optimiser, the quantum circuit, and the loss function for training the classifier can be found in Appendix D.7.1. **(a)** Plot of the distribution of the IRIS features after normalisation of the dataset. **(b)** Parameterised quantum circuit used in [4], which uses one qubit per input data feature. Two repetitions of a ZZFEATUREMAP with an all-to-all connectivity are used to encode the data, followed by a variational ansatz with 8 trainable parameters. The two-qubit operations in the feature map are shown in Eq. D.23. **(c)** Training loss for 10^2 training runs starting with different initialisation of the variational parameters. The dashed line is the average over such training trajectories. Curves are grouped into three classes denoted by the colours (green, red, purple), depending on the final loss they achieve at the end of training. Such colour-coding is used in panel (d) to distinguish the training trajectories and study how the final loss relates to the entanglement created in the circuit. **(d)** Mean entanglement entropy (D.22) averaged over the full training dataset, for the bipartitions occurring at bonds e_1 , e_2 , and e_3 . The curves are coloured according to the final loss obtained at the end of the training, as shown in panel (c). Dashed lines are obtained by averaging over all the trajectories.

At the end of the training, trajectories cluster around three possible values of the final loss, and each curve is coloured depending on the cluster they belong to, that is the final loss they achieve at the end of training. Such colour coding scheme is then used to distinguish trajectories in panel (d), which shows the evolution of the mean (over the dataset) entanglement entropy Eq. (D.22) during training, across the three bipartitions corresponding to bonds e_1 , e_2 , and e_3 , indicated in the Figure.

Finally, dashed lines in panels (b) and (d) are obtained by taking the mean over all curves in the corresponding plots.

The mean entanglement entropy $\Sigma_k(t)$ is defined as the average over the full training dataset of the entanglement entropy corresponding to bond e_k , when the parameter vector is $\boldsymbol{\theta}_t$ at training step t , namely

$$\Sigma_k(t) = \frac{1}{m} \sum_{i=1}^m S(e_k; \mathbf{x}_i, \boldsymbol{\theta}_t), \quad (\text{D.22})$$

where $m = 100$ is the size of the dataset, \mathbf{x}_i is an input vector from the dataset, and $S(e_k; \mathbf{x}_m, \boldsymbol{\theta}_t)$ is the entanglement entropy of the bipartition corresponding to cut e_k of the quantum state $\rho = U(\mathbf{x}_i, \boldsymbol{\theta}_t) |\mathbf{0}\rangle\langle\mathbf{0}| U(\mathbf{x}_i, \boldsymbol{\theta}_t)^\dagger$, see Eq. (7.14).

Entanglement at $t = 0$ Various observations can be drawn from the simulation results reported in Fig. D.4. First of all, the mean entanglement at the start of training $\Sigma_k(t = 0)$, when the circuit is initialized with random parameters, is very different on single qubit cuts corresponding to bonds e_1 and e_3 . While the first qubit starts with a highly entangled state with the rest of the system, even more than a typical Haar-random state, qubit four has instead a much lower initial entanglement. This can be understood as a consequence of the structure of the quantum circuit, as well as the actual numerical values of the features. Indeed, the two-qubits gates used in the feature map are parameterized ZZ-interactions of the form

$$\text{ZZ}_{ij}(\phi_{ij}) = \exp(-i\phi_{ij} Z_i \otimes Z_j / 2), \quad (\text{D.23})$$

where the rotation amount is a function on the numerical value of the features, in our case $\phi_{ij} = \phi(x_i, x_j) = 2x_i \cdot 2x_j$.

If the rotation angle is small $\phi \ll 1$, then $\text{ZZ}(\phi) \sim \mathbb{I}$, and so little entanglement is created between the interested interacting qubits. Since the feature x_3 is particularly smaller than the other features (especially for class zero, see Fig. D.4a), then the ZZ interactions involving the fourth qubit will be closer to the identity, and hence the entanglement generated in the feature map between the fourth qubit and the rest of the system will be consequently smaller.

In addition, the variational part of the circuit consists of a reversed linear network of CNOT gates, which has a different impact on different qubits. For example, only one CNOT is inside the past light cone of the fourth qubit (corresponding to the feature x_3), while the state of the first qubit (corresponding to feature x_0) is affected by all the CNOTs.

Thus, overall due to the specific structure of the feature map and the variational ansatz, one can expect the fourth qubit to have little entanglement with the rest of the system, which is indeed confirmed by the numerical results. At last, it is worth noticing that while the circuit uses two repetitions of an all-to-all feature map, these are arranged in a sequential manner, and thus entanglement at the central bond e_2 has not yet reached the Haar-random value, as one would expect from a two-layered all-to-all circuit, see App. D.5.

Entanglement at $t > 0$ We now move our attention to the dynamics of entanglement during training. First, looking at the average dashed lines in panel (d) of Fig. D.4, entanglement among qubits generally decreases as training progresses, especially for cut e_3 , but also for e_2 to a lesser extent. The training trajectories that reach better solutions (lower loss, indicated in green) are those that correspond to lowest entanglement, thus indicating that for this specific setup a decrease in entanglement is beneficial to reach good performances. This is particularly true for the fourth qubit, as the training procedure drives the system towards states where the qubit is further disentangled from the rest of the system.

Specifically, in Sec. D.7.1 we show how the entanglement evolves for the two classes separately, that is evaluating Σ_k in Eq. (D.22) not on the full dataset but on each class separately. We see that

elements in class 0 are almost disentangled from the rest of the system as training proceeds towards good minima (green curves). The specific choice of the variational ansatz impacts the shape of the loss landscape, and it is such that continuing to disentangle the fourth qubit is the most effective strategy to ensure good performance.

A few concluding remarks The dependence between the entanglement generated in the data encoding part of the quantum circuit and the numerical values of the features underlines the importance of preprocessing the classical data, which is known to have a profound impact on the class of functions the parameterized quantum model have access to, so its performances [109, 269]. Moreover, the trainable part of the circuit has a profound impact on how training can increase or diminish entanglement among qubits.

Finally, we stress that as the quantum model is trained to minimize the loss function, the entanglement entropy in the circuit must not be a monotone function of the training step, and in fact, we check numerically that varies a lot both within a single training run, as well as across several training trajectories. It would be interesting to study whether the exploration of different regimes of entanglement is related to the performances of a quantum machine learning model.

D.7.1 Details on the classification procedure

For the optimization task of the IRIS dataset we use a custom PennyLane [27] implementation based on the code [3], accompanying the results presented in Abbass et al. [4].

IRIS dataset The IRIS dataset [103] consists of 150 samples of four dimensional real input vectors $\mathbf{x}_i = (x_0^{(i)}, x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) \in \mathbb{R}^4$, each one paired with a corresponding class label $\lambda_i \in \{0, 1, 2\}$. We consider a classification task involving only the first two classes $\lambda_i \in \{0, 1\}$, so the actual dataset considered for our analysis consists of $m = 100$ samples¹, namely

$$S = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^4 \times \{0, 1\} \mid \|\mathbf{x}_i\| = 1, i = 1, \dots, 100\}, \quad (\text{D.24})$$

where each input vector was normalised $\mathbf{x}_i \rightarrow \mathbf{x}_i / \|\mathbf{x}_i\|$, and we defined the desired output label as $y_i = 1 - \lambda_i$, that is $y_i = 1$ if the sample is in class $\lambda_i = 0$, and zero otherwise.

In Fig. D.5 we summarise the properties of the IRIS dataset used in our simulations. Note that the plots on the diagonal are those shown in Fig. D.4a, and represent the continuous probability density of the corresponding features, obtained by smoothing the histogram of the frequencies [314].

Quantum neural network circuit and optimisation The quantum neural network ansatz, denoted with $U(\mathbf{x}_i; \boldsymbol{\theta})$, acts on $n = 4$ qubits and it is showed in Fig. D.4c. It consists of two layers of the ZZFEATUREMAP with an all-to-all (full) entangling connectivity which encodes the inputs \mathbf{x}_i into the circuit, where one qubit per feature is used.

The encoding part of the circuit is followed by a variational ansatz with eight trainable parameters $\boldsymbol{\theta} \in \mathbb{R}^8$, and a network of CNOT gates. Note that we used a reversed linear network of CNOTs, as this is equivalent to the full entangling connectivity originally used in [3, 4], as shown previously in Appendix D.3.

Given an input \mathbf{x}_i and trainable parameters $\boldsymbol{\theta}$, the output of the circuit is obtained by measuring a Pauli-Z operator on all qubits $\langle O \rangle_{\mathbf{x}_i; \boldsymbol{\theta}} = \langle \mathbf{0} | U(\mathbf{x}_i; \boldsymbol{\theta})^\dagger Z^{\otimes 4} U(\mathbf{x}_i; \boldsymbol{\theta}) | \mathbf{0} \rangle$, and by constructing the predicted class probability

$$\tilde{y}_i(\boldsymbol{\theta}) = \frac{1 + \langle O \rangle_{\mathbf{x}_i; \boldsymbol{\theta}}}{2} \in [0, 1], \quad (\text{D.25})$$

which is equivalent to measuring the parity of the bitstrings at the output of the circuit.

¹The full IRIS dataset comprises 50 samples per class, for a total of 150 samples.

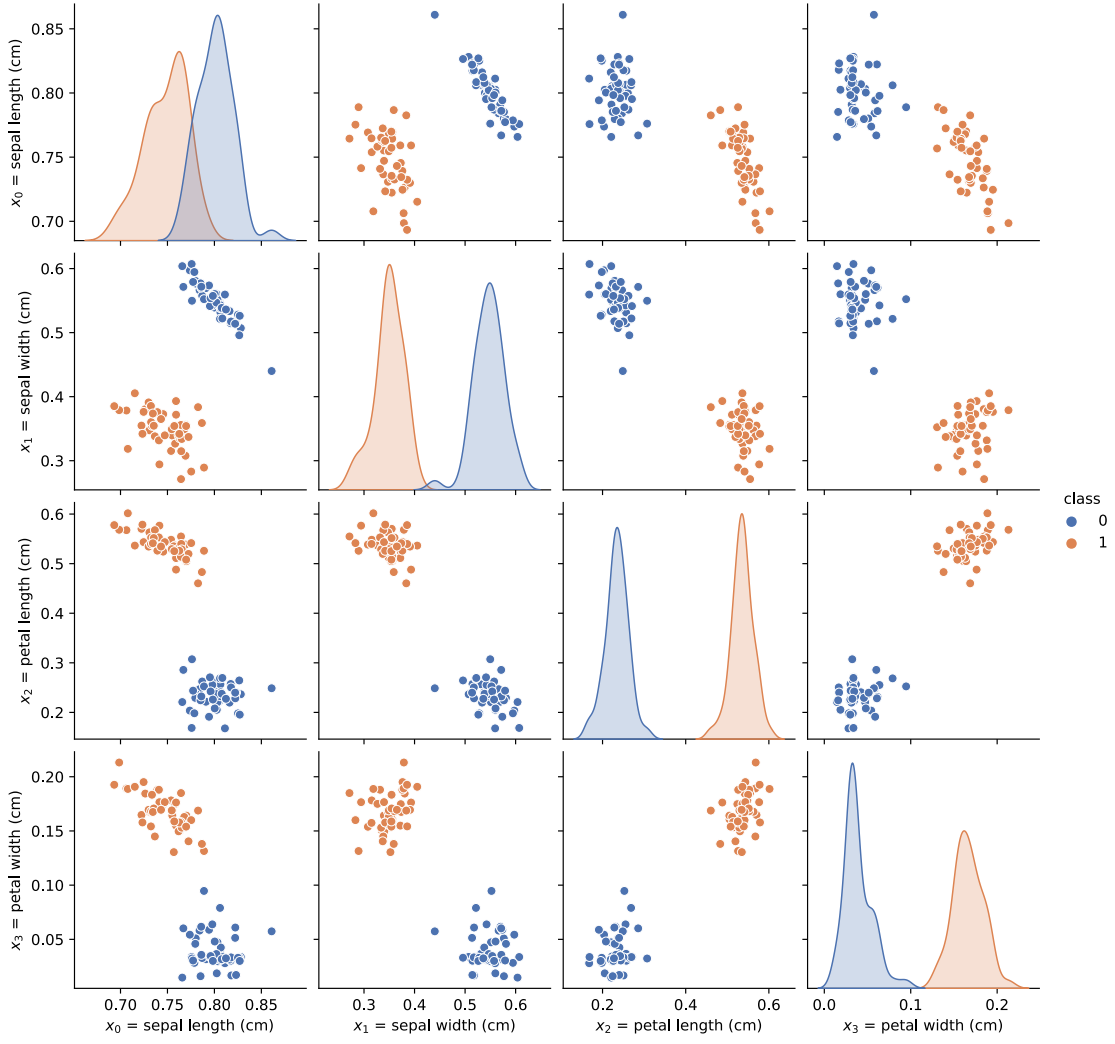


Figure D.5: Summary of the preprocessed IRIS dataset used in our simulations to reproduce the classification task proposed in [4]. The plots on the diagonal of the grid correspond to a smoothed version of the histogram of the frequencies of each feature built with `seaborn` [314], and are the same plots shown in Fig. D.4 in the main text.

The circuit is optimized through gradient descent with Adam optimiser [165] with learning rate set to $\eta = 0.1$, which was used to minimize the cross-entropy loss function averaged over the whole training dataset

$$L(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \ell_i(y_i, \hat{y}_i(\boldsymbol{\theta})) = -\frac{1}{100} \sum_{i=1}^{100} (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)). \quad (\text{D.26})$$

We run the training process 100 times, each time initialising the circuit with variational parameters sampled randomly from the uniform distribution $\theta_i \sim \text{Unif}[0, 2\pi)$.

Mean entanglement entropy per class In Fig. D.6 we report the mean entanglement entropy $\Sigma_k(t)$ defined in Eq. (D.22), where the average is taken separately over elements belonging to the

two different classes, namely

$$\Sigma_k^{(0)}(t) = \frac{1}{50} \sum_{m=1}^{50} S(e_k; \mathbf{x}_m, \boldsymbol{\theta}_t) \quad \text{for } \lambda_m = 0, \quad (\text{D.27})$$

$$\Sigma_k^{(1)}(t) = \frac{1}{50} \sum_{m=1}^{50} S(e_k; \mathbf{x}_m, \boldsymbol{\theta}_t) \quad \text{for } \lambda_m = 1. \quad (\text{D.28})$$

As discussed in the main text, the entanglement entropy for cut e_3 for elements belonging to class 0 is smaller in general, due to the feature x_3 for Class 0 being roughly one order of magnitude smaller than the remaining features, see Fig. D.5.

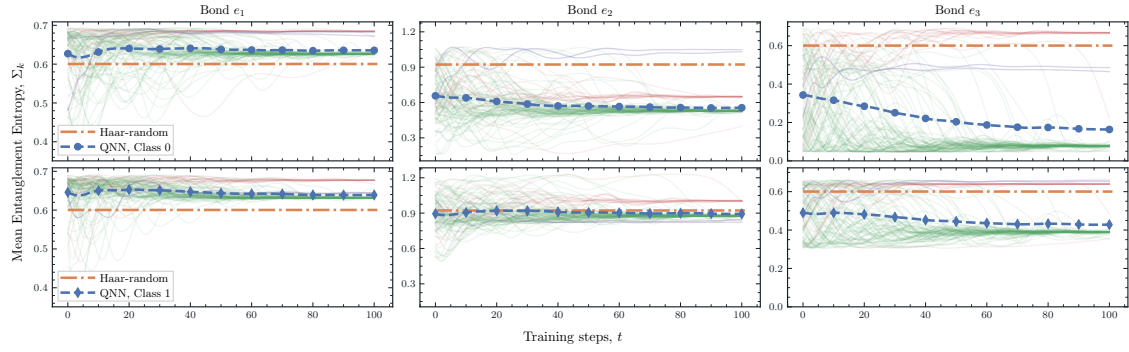


Figure D.6: Mean entanglement entropy Σ_k (D.22) averaged over elements belonging to the different classes, for the three bonds e_1 , e_2 , e_3 . Dashed lines represent averages over the various training trajectories.

E. Noise Deconvolution

E.1 Kraus Decomposition

A quantum physical evolution is represented by (i) *linear*, (ii) *completely-positive* and (iii) *trace-preserving* (CPTP) maps taking quantum density operators to quantum density operators. A map satisfying these three properties is called a *quantum channel*, and can be interpreted as a quantum evolution obtained through the interaction of the system with an external environment. A map is a quantum channel if and only if it admits a Kraus (or *operator-sum*) representation as

$$\rho \longrightarrow \mathcal{E}(\rho) = \sum_k A_k \rho A_k^\dagger, \quad (\text{E.1})$$

with the trace preserving condition requiring

$$\text{Tr}[\mathcal{E}(\rho)] = \text{Tr}[\rho] \implies \sum_k A_k^\dagger A_k = \mathbb{I}. \quad (\text{E.2})$$

The operators $\{A_k\}_k$ are called the Kraus operators of the channel, which are however non-unique [220]. Such channels are often referred to as *stochastic* channels [28, 45], and if they also preserve identity ($\mathcal{E}(\mathbb{I}) = \mathbb{I}$), they are called *unital* (or *bistochastic*). Unitality corresponds to the requirement that also $\sum_k A_k A_k^\dagger = \mathbb{I}$, from which it is clear that a sufficient condition for a CPTP map to be unital is that its Kraus operators be self-adjoint $A_k = A_k^\dagger \forall k$.

E.2 Tomographic reconstruction formula for qubits

In this Appendix, we show how the tomographic reconstruction formula for systems made of qubits $\mathcal{H} = \mathbb{C}^2$ can be recovered starting from the standard basis expansion in terms of the Pauli matrices [70]. The set of matrices $\{\mathbb{I}, \sigma_x, \sigma_y, \sigma_z\}$ forms an orthonormal set and constitutes a basis for the space of 2×2 complex matrices $\mathcal{L}(\mathcal{H}) = \mathbb{C}^{2 \times 2}$. So, given an operator $O \in \mathcal{L}(\mathcal{H})$, it can be expressed as

$$\begin{aligned} O &= \frac{\mathbb{I} \text{Tr}[O] + \sigma_x \text{Tr}[O\sigma_x] + \sigma_y \text{Tr}[O\sigma_y] + \sigma_z \text{Tr}[O\sigma_z]}{2} \\ &= \frac{\text{Tr}[O]}{2} \mathbb{I} + \sum_{\alpha=x,y,z} \frac{\text{Tr}[O\sigma_\alpha]}{2} \sigma_\alpha \\ &= \sum_{\alpha=x,y,z} \frac{1}{3} \left(\frac{3 \text{Tr}[O\sigma_\alpha]}{2} \sigma_\alpha + \frac{\text{Tr}[O]}{2} \mathbb{I} \right) \\ &= \sum_{\alpha=x,y,z} \frac{1}{3} \mathcal{E}[O](\sigma_\alpha), \end{aligned} \quad (\text{E.3})$$

where we defined

$$\mathcal{E}[O](\sigma_\alpha) = \left(\frac{3 \text{Tr}[O\sigma_\alpha]}{2} \sigma_\alpha + \frac{\text{Tr}[O]}{2} \mathbb{I} \right) \quad (\text{E.4})$$

which is the desired *quantum estimator*, with $\{\sigma_x, \sigma_y, \sigma_z\}$ constituting the *quorum* of observables of the tomographic reconstruction. The equation (E.3) has the same form of the tomographic reconstruction formula in Eq. (8.11), with substitutions

$$\int_\lambda \longrightarrow \sum_\lambda, \quad d\lambda \longrightarrow 1/3, \quad \lambda \longrightarrow \{x,y,z\}, \quad \{Q_\lambda\} \longrightarrow \{\sigma_x, \sigma_y, \sigma_z\},$$

which account for the fact that we are dealing with a discrete, and not continuous, basis expansion.

Also, note that (E.3) is not the unique choice for the tomographic formula. In fact, one could use a continuous parameterisation of the group $SU(2)$ given by the operator $\mathcal{R}(\vec{n}, \psi) = e^{i\vec{s}\cdot\vec{n}\psi}$, where \vec{s} is the spin of the particle ($\vec{s} = \vec{\sigma}/2$ for qubits), $\vec{n} = (\cos\phi \sin\theta, \sin\phi \sin\theta, \cos\phi)$, $\theta \in [0, \pi]$ and $\phi, \psi \in [0, 2\pi]$ [70].

E.3 Noise deconvolution for qubits

In this appendix, we derive the noise deconvolution formula for qubits. Let ρ be a quantum state, and \mathcal{N} be a noise channel that admits an inverse map \mathcal{N}^{-1} , and $\hat{\mathcal{N}}^{-1}$ its adjoint map. Then, using Eq. (8.15) in (8.13), one obtains

$$\begin{aligned}
\langle O \rangle &= \sum_{\alpha} \frac{1}{3} \text{Tr}[\hat{\mathcal{N}}^{-1}(\mathbb{E}(O)[\sigma_{\alpha}])\mathcal{N}(\rho)] \\
&= \sum_{\alpha} \frac{1}{3} \text{Tr} \left[\left(\frac{3}{2} \text{Tr}[O\sigma_{\alpha}]\hat{\mathcal{N}}^{-1}(\sigma_{\alpha}) + \frac{1}{2} \text{Tr}[O]\hat{\mathcal{N}}^{-1}(\mathbb{I}) \right) \mathcal{N}(\rho) \right] \\
&= \sum_{\alpha} \frac{1}{3} \left(\frac{3}{2} \text{Tr}[O\sigma_{\alpha}]\langle \hat{\mathcal{N}}^{-1}(\sigma_{\alpha}) \rangle_{\mathcal{N}(\rho)} + \frac{1}{2} \text{Tr}[O] \underbrace{\text{Tr}[\hat{\mathcal{N}}^{-1}(\mathbb{I})\mathcal{N}(\rho)]}_{=\text{Tr}[\mathbb{I}\rho]=1} \right) \\
&= \frac{1}{2} \text{Tr}[O] + \frac{1}{2} \sum_{\alpha=x,y,z} \text{Tr}[O\sigma_{\alpha}]\langle \hat{\mathcal{N}}^{-1}(\sigma_{\alpha}) \rangle_{\mathcal{N}(\rho)}.
\end{aligned} \tag{E.5}$$

This equation (E.5) lets us deconvolve the effect of noise by evaluating the expectation value of the noise-inverted Pauli matrices σ_{α} on the noisy state $\mathcal{N}(\rho)$. In particular, note that the formula remains valid whether the noise is unital or not. In fact, in the second line we can always move the adjoint inverse noise $\hat{\mathcal{N}}^{-1}$ from the identity to the noisy state $\mathcal{N}(\rho)$, thus obtaining $\text{Tr}[\hat{\mathcal{N}}^{-1}(\mathbb{I})\mathcal{N}(\rho)] = \text{Tr}[\mathbb{I}\mathcal{N}^{-1}(\mathcal{N}(\rho))] = \text{Tr}[\rho] = 1$.

E.4 Inverse maps of Noise channels

We hereby report the explicit calculations to derive the inverse maps of the noise channels considered in the main text.

E.4.1 Bit-flip, phase-flip, and bit-phase-flip channels

We focus on the bit-flip channel, but the calculations are identical for the phase-flip and bit-phase-flip channels as well. The bit-flip channel is described by Kraus operators $A_0 = \sqrt{1-p}\mathbb{I}$ and $A_1 = \sqrt{p}\sigma_x$, so that its action is given by

$$\mathcal{N}_x(\rho) = (1-p)\rho + p\sigma_x\rho\sigma_x. \tag{E.6}$$

The Pauli Transfer Matrix Γ_x is defined as

$$(\Gamma_x)_{ij} = \frac{1}{2} \text{Tr}[\sigma_i \mathcal{N}_x(\sigma_j)], \tag{E.7}$$

and by straightforward calculation one obtains

$$\begin{aligned}
(\Gamma_x)_{11} &= (1-p) + p = 1, \\
(\Gamma_x)_{22} &= (1-p) - p = 1 - 2p, \\
(\Gamma_x)_{33} &= (1-p) - p = 1 - 2p, \\
(\Gamma_x)_{ij} &= 0, \quad \text{for } i \neq j,
\end{aligned}$$

thus yielding

$$\Gamma_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & (1-2p) & 0 \\ 0 & 0 & 0 & (1-2p) \end{bmatrix}, \quad (\text{E.8})$$

whose inverse is trivially

$$\Gamma_x^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{(1-2p)} & 0 \\ 0 & 0 & 0 & \frac{1}{(1-2p)} \end{bmatrix}. \quad (\text{E.9})$$

The eigenvectors of such Pauli Transfer Matrix are clearly the Pauli matrices \mathbb{I} , $|\sigma_x\rangle$, $|\sigma_y\rangle$, $|\sigma_z\rangle$, with eigenvalues $\boldsymbol{\lambda} = (1, 1, 1/(1-2p), 1/(1-2p))$, respectively.

The operator sum representation of \mathcal{N}_x^{-1} can be reconstructed by noticing that the map

$$\mathcal{E}(O) = \sum_{j=0}^3 \beta_j \sigma_j O \sigma_j. \quad (\text{E.10})$$

has also the Pauli matrices as eigenvectors, but with eigenvalues $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3)$. Since two maps are equal if they have the same action on a basis, we can find the operator-sum representation of \mathcal{N}_x^{-1} by finding those β_j such that $\boldsymbol{\lambda} = \boldsymbol{\beta}$. If we can find such a mapping, then inserting those values into (E.10), we recover the operator sum of the inverse map. The PTM matrix $\Gamma_{\mathcal{E}}$ of \mathcal{E} amounts to

$$\Gamma_{\mathcal{E}} = \text{diag}(\beta_0 + \beta_1 + \beta_2 + \beta_3, \beta_0 + \beta_1 - \beta_2 - \beta_3, \quad (\text{E.11})$$

$$\beta_0 - \beta_1 + \beta_2 - \beta_3, \beta_0 - \beta_1 - \beta_2 + \beta_3), \quad (\text{E.12})$$

The equality $\Gamma_x^{-1} = \Gamma_{\mathcal{E}}$ correspond to the system of equations

$$\begin{cases} 1 = \beta_0 + \beta_1 + \beta_2 + \beta_3 \\ 1 = \beta_0 + \beta_1 - \beta_2 - \beta_3 \\ \frac{1}{1-2p} = \beta_0 - \beta_1 + \beta_2 - \beta_3 \\ \frac{1}{1-2p} = \beta_0 - \beta_1 - \beta_2 + \beta_3 \end{cases} \quad (\text{E.13})$$

where the first equation is the trace-preserving condition, dictated by the fact that the direct map is TP, and so the inverse map has to be. This condition is also evident from the expression of Γ_x^{-1} and $\Gamma_{\mathcal{E}}$, since the first row is of the form $(1, 0, 0, 0)$. The system of equations (E.13) has solutions

$$\beta_0 = \frac{1-p}{1-2p}, \quad \beta_1 = -\frac{p}{1-2p}, \quad \beta_2 = \beta_3 = 0,$$

and substituting these values in Eq. (E.10) leads to the desired operator-sum representation

$$\mathcal{N}_x^{-1}(O) = \frac{1-p}{1-2p} O - \frac{p}{1-2p} \sigma_x O \sigma_x. \quad (\text{E.14})$$

Similarly, the same procedure can be carried out for the dephasing (generated by σ_z) and bit-phase-flip channel (generated by σ_y), leading to

$$\mathcal{N}_z^{-1}(O) = \frac{1-p}{1-2p} O - \frac{p}{1-2p} \sigma_z O \sigma_z, \quad (\text{E.15})$$

$$\mathcal{N}_y^{-1}(O) = \frac{1-p}{1-2p} O - \frac{p}{1-2p} \sigma_y O \sigma_y. \quad (\text{E.16})$$

Note that for all these three cases, the adjoint channels are equal to the direct ones, i.e. $\hat{\mathcal{N}}^{-1} = \mathcal{N}^{-1}$, since the generating operators are all Hermitian (see Appendix E.4.4 for a case where this is not true).

We now proceed to evaluate the explicit form of the deconvolution formula. Let $\beta \in \{x, y, z\}$ index one of the noise channels $\mathcal{N}_\beta \in \{\mathcal{N}_x, \mathcal{N}_y, \mathcal{N}_z\}$, the action of the inverse map on a Pauli matrix σ_α amounts to

$$\mathcal{N}_\beta^{-1}(\sigma_\alpha) = \frac{1}{1-2p} \left((1-p)\sigma_\alpha - p\sigma_\beta\sigma_\alpha\sigma_\beta \right) = \frac{1-2\delta_{\alpha\beta}p}{1-2p}\sigma_\alpha, \quad (\text{E.17})$$

where in the second line we made use of the fact that $\sigma_\beta\sigma_\alpha\sigma_\beta = (2\delta_{\alpha\beta} - 1)\sigma_\alpha$. Substituting this in Eq (E.5), one obtains

$$\langle O \rangle_\beta = \frac{1}{2} \text{Tr}[O] + \frac{1}{2(1-2p)} \sum_{\alpha=x,y,z} \text{Tr}[O\sigma_\alpha] (1-2\delta_{\alpha\beta}p) \langle \sigma_\alpha \rangle_{\mathcal{N}_\beta(\rho)}, \quad (\text{E.18})$$

where the subscript β in $\langle O \rangle_\beta$ is just used to denote that we are deconvolving with respect to noise \mathcal{N}_β , but remember that it correspond to the mitigated noise-free result. Clearly, when the observable to be measured is itself a Pauli matrix $O = \sigma_\gamma$, this further simplifies to

$$\langle \sigma_\gamma \rangle_\beta = \frac{1}{2(1-2p)} \sum_{\alpha=x,y,z} \underbrace{\text{Tr}[\sigma_\gamma\sigma_\alpha]}_{=2\delta_{\gamma\alpha}} (1-2\delta_{\alpha\beta}p) \langle \sigma_\alpha \rangle_{\mathcal{N}_\beta(\rho)} \quad (\text{E.19})$$

$$= \frac{1-2\delta_{\gamma\beta}p}{1-2p} \langle \sigma_\gamma \rangle_{\mathcal{N}_\beta(\rho)}. \quad (\text{E.20})$$

E.4.2 Depolarizing channel

The depolarizing noise channel is represented by the map

$$\mathcal{N}_{\text{dep}}(\rho) = \left(1 - \frac{3p}{4}\right)\rho + \frac{p}{4} \left(\sigma_x\rho\sigma_x + \sigma_y\rho\sigma_y + \sigma_z\rho\sigma_z \right),$$

having Kraus operators $A_0 = \sqrt{1-3p/4}\mathbb{I}$, $A_1 = \sqrt{p}\sigma_x/2$, $A_2 = \sqrt{p}\sigma_y/2$, and $A_3 = \sqrt{p}\sigma_z/2$. By straightforward calculation, the Pauli Transfer Matrix of the depolarising channel as well as its inverse, amounts to

$$\Gamma_{\text{dep}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-p & 0 & 0 \\ 0 & 0 & 1-p & 0 \\ 0 & 0 & 0 & 1-p \end{bmatrix}, \quad (\text{E.21})$$

$$\Gamma_{\text{dep}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{1-p} & 0 & 0 \\ 0 & 0 & \frac{1}{1-p} & 0 \\ 0 & 0 & 0 & \frac{1}{1-p} \end{bmatrix}. \quad (\text{E.22})$$

Following the same procedure used for the bit-flip channel, one arrives at the system of equations

$$\begin{cases} 1 = \beta_0 + \beta_1 + \beta_2 + \beta_3 \\ \frac{1}{1-p} = \beta_0 + \beta_1 - \beta_2 - \beta_3 \\ \frac{1}{1-p} = \beta_0 - \beta_1 + \beta_2 - \beta_3 \\ \frac{1}{1-p} = \beta_0 - \beta_1 - \beta_2 + \beta_3 \end{cases}, \quad (\text{E.23})$$

which has solutions $\beta_0 = (4-p)/4(1-p)$ and $\beta_1 = \beta_2 = \beta_3 = -p/4(1-p)$. Substituting these values in (E.10), and using the relation $2\text{Tr}[O]\mathbb{I} = O + \sigma_x O \sigma_x + \sigma_y O \sigma_y + \sigma_z O \sigma_z$, one obtains

$$\mathcal{N}_{\text{depol}}^{-1}(O) = \frac{1}{1-p} \left(O - \frac{p}{2} \text{Tr}[O]\mathbb{I} \right). \quad (\text{E.24})$$

Plugging this in the tomographic deconvolution formula (E.5), leads to:

$$\langle O \rangle = \frac{1}{2} \text{Tr}[O] + \frac{1}{2} \sum_{\alpha} \frac{\text{Tr}[O\sigma_{\alpha}]}{1-p} \langle \sigma_{\alpha} \rangle_{\mathcal{N}_{\text{depol}}(\rho)}, \quad (\text{E.25})$$

from which it is clear that whenever a Pauli matrix is to be measured, $O = \sigma_k$, then the expectation values are contracted by a factor $1-p$, i.e. $\langle \sigma_k \rangle = \langle \sigma_k \rangle_{\text{depol}} / (1-p)$.

E.4.3 General Pauli channel

The most general channel involving only Pauli operators is the channel given by

$$\mathcal{N}_{\mathbf{p}}(\rho) = p_0 \rho + p_x \sigma_x \rho \sigma_x + p_y \sigma_y \rho \sigma_y + p_z \sigma_z \rho \sigma_z \quad (\text{E.26})$$

characterized by probabilities $\mathbf{p} = (p_0, p_x, p_y, p_z)$, with the trace-preserving condition implying $p_0 = 1 - p_x - p_y - p_z$. The PTM of this map is diagonal

$$\Gamma_{\mathbf{p}} = \text{diag}(1, p_0 + p_x - p_y - p_z, p_0 - p_x + p_y - p_z, p_0 - p_x - p_y + p_z), \quad (\text{E.27})$$

and has trivial inverse

$$\Gamma_{\mathbf{p}}^{-1} = \text{diag}(1, (p_0 + p_x - p_y - p_z)^{-1}, (p_0 - p_x + p_y - p_z)^{-1}, (p_0 - p_x - p_y + p_z)^{-1}). \quad (\text{E.28})$$

Again, using the same procedure as before, one arrives at the system of equations:

$$\begin{cases} 1 = \beta_0 + \beta_1 + \beta_2 + \beta_3 \\ \frac{1}{p_0 + p_x - p_y - p_z} = \beta_0 + \beta_1 - \beta_2 - \beta_3 \\ \frac{1}{p_0 - p_x + p_y - p_z} = \beta_0 - \beta_1 + \beta_2 - \beta_3 \\ \frac{1}{p_0 - p_x - p_y + p_z} = \beta_0 - \beta_1 - \beta_2 + \beta_3 \end{cases}, \quad (\text{E.29})$$

whose solution is reported in Eq. (8.39) in the main text. The action of the inverse map on the Pauli matrix σ_x is

$$\mathcal{N}_{\mathbf{p}}^{-1}(\sigma_x) = \beta_0 \sigma_x + \beta_1 \sigma_x \sigma_x \sigma_x + \beta_2 \sigma_y \sigma_x \sigma_y + \beta_3 \sigma_z \sigma_x \sigma_z \quad (\text{E.30})$$

$$= (\beta_0 + \beta_1 - \beta_2 - \beta_3) \sigma_x \quad (\text{E.31})$$

$$= \frac{1}{1 - 2(p_y + p_z)} \sigma_x, \quad (\text{E.32})$$

and a similar expression also hold for σ_y and σ_z , from which one obtains the deconvolution formulas in Eq. (8.40).

E.4.4 Amplitude Damping

The amplitude damping channel is given by the map

$$\begin{aligned} \mathcal{N}_{\text{AD}}(\rho) &= K_0 \rho K_0^\dagger + K_1 \rho K_1^\dagger, \\ K_0 &= \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix} \quad K_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}. \end{aligned} \quad (\text{E.33})$$

Differently from all the other cases treated above, this channel is not generated by coupled sigma matrices, and in addition one of its generators is not Hermitian. This has two consequences: first, we cannot straightforwardly apply the same eigenvalue matching procedure used above, second one must consider the adjoint channel when deconvolving.

The PTM of the amplitude damping channel is

$$\Gamma_{\text{AD}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{1-p} & 0 & 0 \\ 0 & 0 & \sqrt{1-p} & 0 \\ p & 0 & 0 & 1-p \end{bmatrix} \quad (\text{E.34})$$

whose inverse is

$$\Gamma_{\text{AD}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{1-p}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{1-p}} & 0 \\ \frac{-p}{1-p} & 0 & 0 & \frac{1}{1-p} \end{bmatrix} \quad (\text{E.35})$$

In this case the eigenvalues of Γ_{AD} and Γ_{AD}^{-1} are not the Pauli matrices, and so we cannot use the eigenvalue matching with the general map in (8.2). However, the two PTMs have the same structure, so one may easily guess that the operator-sum representation of the two maps share the same operators, something that also always happened in all previous cases. Let us then suppose that the inverse map $\mathcal{N}_{\text{AD}}^{-1}$ has the form

$$\mathcal{N}_{\text{AD}}^{-1}(\cdot) = \tilde{K}_0 \cdot \tilde{K}_0^\dagger - \tilde{K}_1 \cdot \tilde{K}_1^\dagger \quad (\text{E.36})$$

with $\tilde{K}_0 = |0\rangle\langle 0| + \kappa|1\rangle\langle 1|$, and $\tilde{K}_1 = \tau|0\rangle\langle 1|$, with κ, τ free parameters to be determined. This map has corresponding PTM

$$\Gamma(\kappa, \tau) = \begin{bmatrix} \frac{1+\kappa^2-\tau^2}{2} & 0 & 0 & 0 \\ 0 & \kappa & 0 & 0 \\ 0 & 0 & \kappa & 0 \\ \frac{1-\tau^2-\kappa^2}{2} & 0 & 0 & \frac{1+\tau^2+\kappa^2}{2} \end{bmatrix}, \quad (\text{E.37})$$

and by requiring that $\Gamma(\kappa, \tau) = \Gamma_{\text{AD}}^{-1}$, we obtain

$$\kappa = \frac{1}{\sqrt{1-\gamma}}, \quad \tau = \sqrt{\frac{\gamma}{1-\gamma}}, \quad (\text{E.38})$$

thus recovering the inverse map

$$\mathcal{N}_{\text{AD}}^{-1}(O) = \tilde{K}_0 O \tilde{K}_0^\dagger - \tilde{K}_1 O \tilde{K}_1^\dagger \quad \tilde{K}_0 = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\sqrt{1-\gamma}} \end{bmatrix}, \quad \tilde{K}_1 = \begin{bmatrix} 0 & \sqrt{\frac{\gamma}{1-\gamma}} \\ 0 & 0 \end{bmatrix}. \quad (\text{E.39})$$

In order to evaluate the deconvolution formula, we first need to calculate the adjoint of the inverse channel. Let $\Phi(\cdot)$ be a linear map acting on the space of operators $\mathcal{L}(\mathcal{H})$, its adjoint $\hat{\Phi}$ is defined as the unique map satisfying the following relation

$$\langle A, \Phi(B) \rangle = \langle \hat{\Phi}(A), B \rangle_{\text{HS}}. \quad (\text{E.40})$$

where $\langle \cdot, \cdot \rangle_{HS}$ denotes the Hilbert-Schmidt inner product $\langle A, B \rangle \equiv \text{Tr}[A^\dagger B]$. Let's consider a generic linear map of the form

$$\Phi(A) = \sum_k \alpha_k V_k A V_k^\dagger, \quad \alpha_k \in \mathbb{R}. \quad (\text{E.41})$$

which is, in general, neither CP nor TP, since we make no further hypothesis on α_k and V_k . By direct application of the definition of adjoint map, we obtain

$$\langle A, \Phi(B) \rangle \equiv \text{Tr}[A^\dagger \Phi(B)] = \text{Tr}\left[A^\dagger \sum_k \alpha_k V_k B V_k^\dagger\right] = \text{Tr}\left[\sum_k \alpha_k V_k^\dagger A^\dagger V_k B\right] \quad (\text{E.42})$$

$$= \text{Tr}\left[\left(\sum_k \alpha_k V_k^\dagger A V_k\right)^\dagger B\right] = \left\langle \sum_k \alpha_k V_k^\dagger A V_k, B \right\rangle \quad (\text{E.43})$$

$$\Rightarrow \hat{\Phi}(A) = \sum_k \alpha_k V_k^\dagger A V_k, \quad (\text{E.44})$$

where we used the linearity and cyclic property of the trace, as well as the fact that the coefficients are real, $\alpha_k^* = \alpha_k \in \mathbb{R}$. We see that for any map of the form (E.41), its adjoint is obtained by simply substituting the operators with their adjoint, i.e. $V_k \rightarrow V_k^\dagger$. If the map $\Phi(\cdot)$ leverages only Hermitian operators $V_k = V_k^\dagger$, as it happens with every Pauli noise channel, then the adjoint and the direct map of course coincides, $\hat{\Phi}(\cdot) = \Phi(\cdot)$. However, the Amplitude Channel uses non Hermitian generators V_k , thus has a non-trivial, yet simple, adjoint map.

Straightforward application of the deconvolution formula then leads to the deconvolved expectation values

$$\begin{aligned} \langle \sigma_x \rangle &= \frac{1}{\sqrt{1-\gamma}} \langle \sigma_x \rangle_{\mathcal{N}_{AD}(\rho)}, \\ \langle \sigma_y \rangle &= \frac{1}{\sqrt{1-\gamma}} \langle \sigma_y \rangle_{\mathcal{N}_{AD}(\rho)}, \\ \langle \sigma_z \rangle &= \frac{1}{1-\gamma} (\langle \sigma_z \rangle_{\mathcal{N}_{AD}(\rho)} - \gamma). \end{aligned} \quad (\text{E.45})$$

E.4.5 2-Kraus channel

The set of channels considered here is generated by two parametrized Kraus operators

$$\mathcal{N}_{\text{two}}(\rho) = \sum_{i=1,2} A_i \rho A_i^\dagger, \quad (\text{E.46})$$

with $A_1 = \cos \alpha |0\rangle\langle 0| + \cos \beta |1\rangle\langle 1|$, and $A_2 = \sin \beta |0\rangle\langle 1| + \sin \alpha |1\rangle\langle 0|$. The PTM of this channel and its inverse are respectively

$$\Gamma_{\text{two}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha - \beta) & 0 & 0 \\ 0 & 0 & \cos(\alpha + \beta) & 0 \\ \frac{\cos(2\alpha) - \cos(2\beta)}{2} & 0 & 0 & \frac{\cos(2\alpha) + \cos(2\beta)}{2} \end{bmatrix}, \quad (\text{E.47})$$

$$\Gamma_{\text{two}}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\cos(\alpha - \beta)} & 0 & 0 \\ 0 & 0 & \frac{1}{\cos(\alpha + \beta)} & 0 \\ \frac{\cos(2\beta) - \cos(2\alpha)}{\cos(2\alpha) + \cos(2\beta)} & 0 & 0 & \frac{2}{\cos(2\alpha) + \cos(2\beta)} \end{bmatrix}. \quad (\text{E.48})$$

Using the trigonometric relation

$$\cos(2\alpha) + \cos(2\beta) = 2 \cos\left(\frac{2\alpha - 2\beta}{2}\right) \cos\left(\frac{2\alpha + 2\beta}{2}\right) \quad (\text{E.49})$$

$$= 2 \cos(\alpha - \beta) \cos(\alpha + \beta) \quad (\text{E.50})$$

we can rewrite the elements of Γ_{two}^{-1} as

$$(\Gamma_{\text{two}}^{-1})_{11} = h_{\alpha\beta} \cos(\alpha + \beta), \quad (\text{E.51})$$

$$(\Gamma_{\text{two}}^{-1})_{22} = h_{\alpha\beta} \cos(\alpha - \beta), \quad (\text{E.52})$$

$$(\Gamma_{\text{two}}^{-1})_{33} = h_{\alpha\beta}^2 \frac{\cos(2\alpha) + \cos(2\beta)}{2}, \quad (\text{E.53})$$

$$(\Gamma_{\text{two}}^{-1})_{30} = h_{\alpha\beta} \frac{\cos(2\beta) - \cos(2\alpha)}{2}, \quad (\text{E.54})$$

with $h_{\alpha\beta} = 2/(\cos(2\alpha) + \cos(2\beta))$.

Expressed in this manner, these matrix elements are very similar to those in the Pauli transfer matrix of the direct channel Γ_{two} in Eq. (E.47). The differences are in the presence of the pre-factor $h_{\alpha\beta}$, as well as in the signs of the angles in elements $(\Gamma_{\text{two}}^{-1})_{11}$ and $(\Gamma_{\text{two}}^{-1})_{22}$, and in the sign in the difference in element $(\Gamma_{\text{two}}^{-1})_{30}$. This suggests that the operator-sum representation of the inverse map can be obtained starting from the direct one with some small changes, as it happened with the amplitude damping channel. First of all, we can multiply the Kraus operators by $\sqrt{h_{\alpha\beta}}$ to introduce the pre-factor, then, to account for the difference in elements $(\Gamma_{\text{two}}^{-1})_{11}$ and $(\Gamma_{\text{two}}^{-1})_{22}$, we can subtract the two operators instead of summing them. At last, element $(\Gamma_{\text{two}}^{-1})_{30}$ can be fixed by changing $\alpha \leftrightarrow \beta$ in the first Kraus operator A_1 . Incidentally, these changes also fix the $(\Gamma_{\text{two}}^{-1})_{33}$ element to the correct value. Thus, eventually, implementing these changes leads to the definition of the operators

$$B_1 = \sqrt{h_{\alpha\beta}} \cos(\beta) |0\rangle\langle 0| + \sqrt{h_{\alpha,\beta}} \cos(\alpha) |1\rangle\langle 1|, \quad (\text{E.55})$$

$$B_2 = \sqrt{h_{\alpha\beta}} \sin(\beta) |0\rangle\langle 1| + \sqrt{h_{\alpha,\beta}} \sin(\alpha) |1\rangle\langle 0|, \quad (\text{E.56})$$

$$h_{\alpha\beta} = \frac{2}{\cos(2\alpha) + \cos(2\beta)}, \quad (\text{E.57})$$

to be used within the inverse map

$$\mathcal{N}_{\text{two}}^{-1}(\cdot) = B_1 \cdot B_1^\dagger - B_2 \cdot B_2^\dagger. \quad (\text{E.58})$$

One can check that this map has the desired Pauli Transfer Matrix Γ_{two}^{-1} . As with the amplitude damping case, one the generators (B_2) is not Hermitian, thus one must be careful in considering the adjoint inverse map when evaluating the deconvolved mean values. By explicit calculations the following holds

$$\begin{aligned} \langle \sigma_x \rangle &= \frac{1}{\cos(\alpha - \beta)} \langle \sigma_x \rangle_{\mathcal{N}_{\text{two}}(\rho)}, \\ \langle \sigma_y \rangle &= \frac{1}{\cos(\alpha + \beta)} \langle \sigma_y \rangle_{\mathcal{N}_{\text{two}}(\rho)}, \\ \langle \sigma_z \rangle &= h_{\alpha\beta} (\cos^2(\beta) + \sin^2(\alpha) - 1) + \langle \sigma_z \rangle_{\mathcal{N}_{\text{two}}(\rho)}. \end{aligned} \quad (\text{E.59})$$