UNIVERSITÀ
DI PAVIA

Department of Electrical, Computer and Biomedical Engineering

Ph.D. program in Electronics, Computer Science and Electrical Engineering

Ph.D. Thesis

LSTM neural networks for industrial and biomedical applications

Advisor:
Prof. Chiara Toffanin

Candidate:
Francesca Iacono

Cycle XXXV

"All models are wrong,
but some are useful."
George Box

Abstract

In the last years, following the principles of Industry 4.0, companies have adapted to the evolution of the market in order to stay competitive. The innovation of the design processes is one of the milestones of the Industry 4.0 and mathematical models of the machines have a key role in the improvement of this process. Similarly, the biomedical field has undergone to an evolution due to the availability of smart devices able to improve the daily life of the patients. Combining data availability and improved hardware capability, neural network techniques had a fast growth and have spread to different fields and also in the system identification and control one.
In this thesis, a particular neural network architecture, the Long Short-Term Memory (LSTM) is employed for system identification. It has being proved that the LSTM is able to take into account the data temporal dependencies and that has good performances in terms of prediction, giving also the possibility to ensure stability properties. The reasons for the choice of this network are presented, together with a comparison with other architectures and the conditions to ensure stability properties.
Four different applications are then presented and analyzed: three cases belong to the industrial field, the latter to the biomedical one. The industrial applications regard the modeling of an industrial autoclave, an industrial plant for coffee roasting and a wastewater treatment plant. The LSTM networks are used to model the processes, comparing the black-box models to more traditional white-box ones, highlighting from case to case pros and cons of the two different approaches. The obtained results have been evaluated in terms of FIT, having very good performances with values ranging between 60% and 86% in the different case studies. The biomedical application instead is related to the research on the artificial pancreas. The LSTM networks are used to predict future glucose values in type 1 diabetes patients, employing then the predictions in an alarm system for hypoglycemia and hyperglycemia prevention, obtaining in silico between 80% and 86% of sensitivity and precision in the two cases.

Sommario

Negli ultimi anni, in seguito all'avvento dell'Industria 4.0, le aziende si sono dovute adattare all'evoluzione del mercato per restare competitive. L'innovazione dei processi produttivi è uno degli obiettivi primari dell'Industria 4.0 e i modelli matematici delle macchine hanno un ruolo chiave nel miglioramento di questi processi. Allo stesso modo si sta assistendo ad un'evoluzione nel campo biomedico, grazie alla disponibilità di dispositivi smart che permettono di aiutare la vita quotidiana dei pazienti. Combinando una maggiore disponibilità di dati e una migliorata capacità hardware, nuove tecniche che utilizzano reti neurali si sono diffuse in molti ambiti, tra cui anche l'identificazione di sistemi dinamici e il controllo.
In questa tesi, una particolare architettura di reti neurali, la Long Short-Term Memory (LSTM), viene usata per l'identificazione di sistemi dinamici, poiché è stato dimostrato che queste reti riescono a tener conto delle dipendenze temporali dei dati e che hanno buone prestazioni in termini di predizione, dando anche la possibilità di garantire proprietà di stabilità.
Quattro diverse applicazioni vengono presentate e analizzate: le prime tre appartengono all'ambito industriale, la quarta a quello biomedico. Le applicazioni industriali riguardano la modellizzazione di un'autoclave industriale, di un impianto industriale per la tostatura del caffè e di un impianto per la depurazioni di acque reflue. Delle reti LSTM vengono utilizzate per modellizzare questi processi, confrontando questi modelli a scatola nera con modelli a scatola bianca, ottenuti con metodi più tradizionali, evidenziando di caso in caso i vantaggi e gli svantaggi dei due diversi approcci. I risultati ottenuti sono stati valutati utilizzando l'indice di FIT, mostrando ottime prestazioni, con valori che variano tra il 60& e l'86% nei diversi casi studio.
L'applicazione biomedica invece è relativa alla ricerca sul pancreas artificiale. Delle reti LSTM vengono usate per predire futuri valori di glucosio in pazienti diabetici di tipo 1, utilizzando poi le predizioni all'interno di un sistema di allarme per la prevenzione di eventi ipoglicemici e iperglicemici, ottenendo per i dati in silico valori di sensitività e precisione tra l'80% e l'86%.

Contents

List of Figures

List of Tables

List of Acronyms

Neural Networks

BRNN	Bi-directional Recurrent Neural Network
CNN	Convolutional Neural Network
ESN	Echo State Network
FFNN	Feed-Forward Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short Term-Memory
ML	Machine Learning
MLP	Multi Layer Perceptron
NN	Neural Network
NNARX	Neural Nonlinear AutoRegressive eXogenous
RNN	Recurrent Neural Network
TCN	Temporal Convolution Network

Control Field

δISS	Incremental Input-to-State Stability
ISS	Input-to-State Stability
MPC	Model Predictive Control
PH	Prediction Horizon

Industrial Field

ASM	Activated Sludge Model
COD	Chemical Oxygen Demand

FHA	Fedegari Horizontal Autoclave
WWTP	WasteWater Treatment Plant

Biomedical Field

AP	Artificial Pancreas
AS	Alarm System
BG	Blood Glucose
CGM	Continuous Glucose Monitoring
DW	Detection Window
FDA	Food and Drug Administration
PW	Prediction Window
SAP	Sensor-Augmented Pump
SMBG	Self-Monitoring Blood Glucose
T1D	Type 1 Diabetes

Chapter 1

Introduction

In the last years the growth and diffusion of new Machine Learning (ML) techniques has influenced both the industrial and the biomedical fields.
The industrial field has been impacted by the start of the so-called Industry 4.0 [1, 2], the 4th industrial revolution, started less than ten years ago. The novelties brought in the industrial field can be synthesized in the so-called "Nine pillars of Industry 4.0" [3, 4], reported in Figure 1.1 and here listed:

1. *Internet of Things*: networks of connected objects via a common protocol, involving immediate response and location information;

2. *Simulation*: virtual models of plants operations, machines and setup, to help in decision making, reduce failures, analyze energy consumption and production time;

3. *Big Data and Analytics*: collection of large datasets, derived from several sources, directly from the production plant, and their analysis;

4. *Cyber Security*: particular attention to the protection of the communication, since everything is connected;

5. *Cloud*: backbone of the connections and acquisitions of the data, to store, access and share data easily;

6. *Additive Manufacturing*: new production methods based on the production of small quantities, for customization to satisfy particular costumer needs;

7. *Augmented Reality*: interface technology between real and virtual world, to help for human-machine interaction, remote control, visual inspection;

8. *Horizontal and Vertical System Integration*: creation of smart factories, for optimization and automation of the entire manufacturing processes,

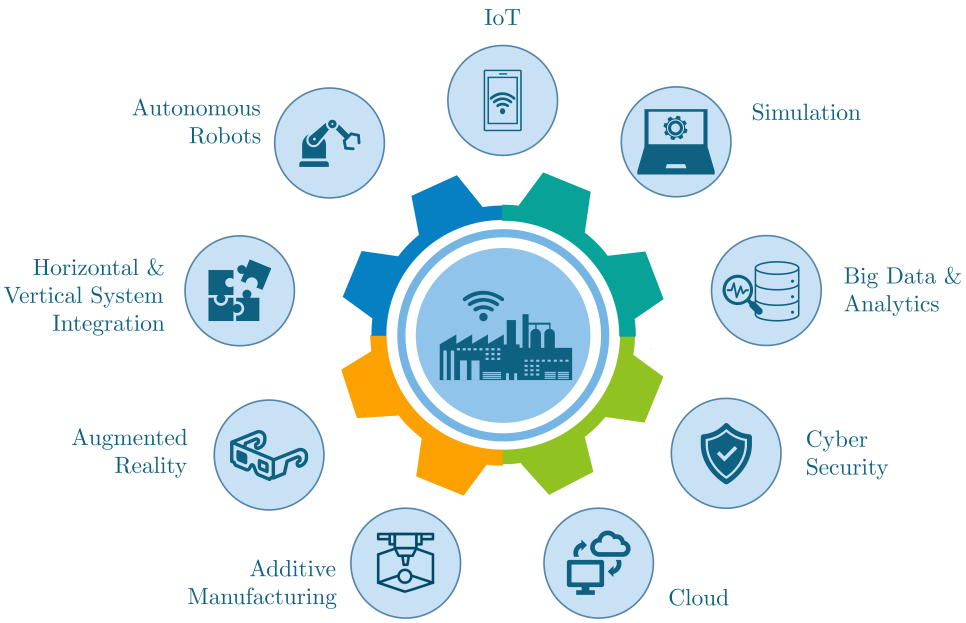


Figure 1.1: The nine pillars of Industry 4.0.

both horizontally (integration between the different part of the production process) and vertically (integration of the production process with the higher-levels businesses);

9. *Autonomous Robots*: robots executing tasks autonomously, with precision and safety, cooperating with humans or with other robots as well.

Analyzing what the pillars have in common, it is clear that data, interconnections and virtualization are essential parts of Industry 4.0: thanks to these innovations, a great amount of industrial data is now available. In general, an increased availability of data, also thanks to the diffusion of internet, together with an improved hardware capability that allows an efficient parallel computing, have brought to the improvement of ML techniques. ML receives data from the industrial world, on the other way the industrial world receives several useful tools to implement simulations and cyber-physical systems. ML can be applied in industry in a variety of tasks [5]: computer vision [6], fault detection [7], predictive maintenance [8], production planning and control [9], hybrid modeling [10] and so on. To summarize, it is clear that the industrial and the ML fields are strictly connected to each other and are growing and evolving together.
Another field into which ML techniques are spreading more and more is the biomedical one. The main advantages of ML in this area is the capability to analyze complex data, collected from intelligent sensors, such as the biomedical ones. For example, in the last years, several special issues of scientific

journals featured ML applications in the biomedical fields [11–13], presenting works such as classification of diseases, classification of images from different diagnostic equipment (magnetic resonance, endoscopic, ultrasound), early detection of illnesses, epidemic predictions, human-machine interfaces for people with disabilities, etc. Thanks to the employment of ML techniques, it is possible to help patients dealing with their diseases during daily life and also to improve the scientific research through the results obtained with these new smart techniques.

1.1 Thesis contribution

Starting from these considerations, the aim of this thesis is to use ML techniques and in particular Neural Networks (NNs), for the identification of dynamical systems, exploiting the advantages brought by ML in the industrial and biomedical fields.
Four practical applications are presented, three of them belonging to the industrial world, while the latter to the biomedical one, where the considered systems are identified also by mean of NNs. In the industrial applications the models obtained using NNs are compared with white-box models identified using more traditional techniques, highlighting the advantages and the limitations of the two different techniques, analyzing when it is better to use one or another. Then in the biomedical application, the identified NN model is used as reference signal for the design of an alarm system.
A particular NN architecture called Long Short Term-Memory (LSTM) network is employed, that belongs to the family of Recurrent Neural Networks (RNNs) and it is well known for its ability to learn long-term dependencies when dealing with temporal data. For this reason, the LSTM is particularly suitable to represent and model dynamical systems. Moreover, the LSTM is clearly defined by mathematical equations that explain its working mechanism: it works well and it is known why. These equations can also be written in a state-space form, giving the possibility to analyze the network using tools of the system identification field, in addition to the classical ML ones. For control-oriented applications, LSTMs are attractive and the architecture more similar to state-space models.

1.2 Thesis structure

This thesis is structured as follows:

- In Chapter 2 an introduction to the NN topic is presented. First, the NNs background is described, focusing on the RNNs architectures in general and the LSTMs in particular. Then, the link between the NNs and the system identification and control fields is depicted, with a brief

historical overview up to nowadays, where LSTMs are the state-of-the-art for this application. A study on the stability analysis of RNNs and LSTMs is reported and lastly these networks are compared with other architectures that can be used to identify dynamical systems.

- In Chapter 3 the first industrial application is presented, that is the modeling of the sterilization process of an industrial autoclave. Two models are proposed: a physical model based on the equations that describe the temperature and pressure behaviors of the machine, to be used for simulation purposes; a LSTM based model, to be used for the design of a control system. Both models are evaluated on real data collected on an industrial autoclave.
  This chapter is based on the results published in "F. Iacono, J. L. Presti, I. Schimperna, S. Ferretti, A. Mezzadra, L. Magni, and C. Toffanin, "Improvement of manufacturing technologies through a modelling approach: an air-steam sterilization case-study", *Procedia Computer Science*, vol. 180, pp. 162–171, 2021".

- In Chapter 4 the modeling of an industrial coffee roaster is carried out. The goal is to define a scalable model to be identified on small machines and then applied also to bigger ones, for economical and environmental sustainability. Firstly, a physical scalable model is presented, based on the state-of-the-art and modified linking some parameters to the machine geometry. Then, the same idea is proposed designing a model based on a LSTM network, trying to enforce the portability. Both solutions have been identified and evaluated on real data of two different machines with different dimensions.
  The formulation and the results of the physical model have been published in "F. Di Palma, F. Iacono, C. Toffanin, A. Ziccardi, and L. Magni, "Scalable model for industrial coffee roasting chamber", *Procedia Computer Science*, vol. 180, pp. 122–131, 2021".

- In Chapter 5 the modeling of the biological reactor of a wastewater treatment plant is described. Also in this case a white-box model and a LSTM based one are proposed. The first one is formulated taking into account both the chemical and biological reactions that occur inside the reactor and the involved flows. However, with this model some dynamics are not correctly described, so the LSTM network is designed, leveraging the availability of a great amount of input/output data, to overcome the problems met with the first model.
  The results of this chapter are presented in "C. Toffanin, F. Di Palma, F. Iacono, and L. Magni, "LSTM network for the oxygen concentration modeling of a wastewater treatment plant", in *Applied Sciences*, Submitted".

- In Chapter 6 the biomedical application is presented. The LSTM network is used to model the glucose dynamic in type 1 diabetes patients, with the goal to design an alarm system for hypoglycemia and hyperglycemia prevention. Firstly, Personalized LSTMs (P-LSTMs) are proposed to predict the glucose level of 100 different in silico patients with a Prediction Horizon (PH) of 40 minutes, considering in input insulin, carbohydrate intake and past glucose values. Then, an improvement of this model is proposed, called Enhanced Personalized LSTMs (EP-LSTMs), to meet some performance requirements so that the EP-LSTMs are used as reference signals in an alarm system. Lastly, this technique is also tested on real data of the OhioT1DM Dataset. This chapter is based on the results published in:
  - "F. Iacono, L. Magni, and C. Toffanin, "Patient-tailored LSTM model for hypoglycemia prevention: an in-silico case study", in *Mathematical Modelling and Control for Healthcare and Biomedical Systems (MCHBS 2021), Virtual Online Conference*, 2021";
  - "F. Iacono, L. Magni, and C. Toffanin, "Personalized LSTM models for glucose prediction in Type 1 diabetes subjects", in *2022 30th Mediterranean Conference on Control and Automation (MED)*, 2022, pp. 324–329";
  - "C. Toffanin, F. Iacono, and L. Magni, "Personalized LSTM-Based Alarm Systems for Hypoglycemia Prevention", in *2023 31th Mediterranean Conference on Control and Automation (MED)*, Accepted";
  - "F. Iacono, L. Magni, and C. Toffanin, "Personalized LSTM-based alarm systems for hypoglycemia and hyperglycemia prevention", in *Biomedical Signal Processing and Control*, Submitted".
- Lastly in Chapter 7 the concluding remarks are gathered, to summarize the results obtained in this thesis, proposing some possible future developments.

Chapter 2

Neural Network Background

In the last decades, a growing availability of data, strictly connected to a more powerful computational capability of the machines, has lead to a new era in the diffusion of neural networks in a lot of different application fields. In recent years, the idea to use neural networks also in the field of identification and control of dynamical system became more consistent [21, 22]. Among all the different type of NNs, the RNNs showed promising results on their performance in this task and in particular the LSTM networks showed particularly interesting results in both industrial and biomedical fields.

2.1 What is a Neural Network

A NN is a non linear mathematical model composed by several connected units, called neurons, that resemble the characteristics of the neural networks in the human brain. A schematic representation of the neuron structure is presented in Figure 2.1. Each neuron receives m inputs (x_i) from other units, with different weights (w_i) and bias (b), and produces an output (y), which value can be expressed as:

$$y = f\left(\sum_{i=1}^{m} w_i x_i + b\right) \tag{2.1}$$

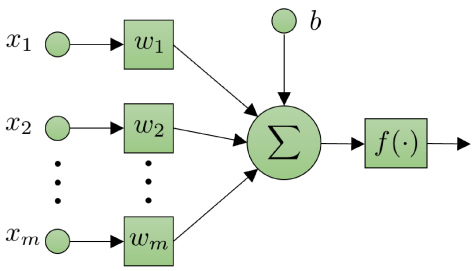


Figure 2.1: Neuron structure.

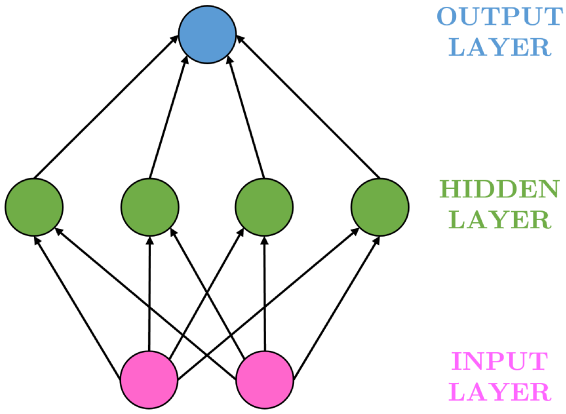


Figure 2.2: Scheme of a NN.

The output is obtained elaborating the inputs through a transformation, determined by the learning parameters w (weight) and b (bias) and an activation function $f(\cdot)$. The activation function can be both linear or non linear, some of the most used are the Rectified Linear unit (ReLu), the sigmoid function or the hyperbolic tangent.
The neurons are organized in different layers, as shown in Figure 2.2. The first one is the input layer, that receives the values of the data from the outside; then one or more hidden layers can be present, each one composed by a variable number of neurons and which number determines the depth of the NN. Lastly, the output layer contains the result of the network.
Depending on the task performed by the NN, the main goal is to define the value of the weights and biases of the hidden layer in order to create a relationship between the inputs and the outputs of the network, through a learning algorithm. This phase is known as training of the NN. For example, two of the most common NN tasks are classification and regression, where the learning algorithm has to classify the given inputs into specific categories or to predict a numerical value related to the inputs, respectively.
The learning algorithm can be seen as an optimization procedure, where a loss function L that measures the difference between the real data and the data predicted by the network is minimized. One of the most employed optimization algorithm is the gradient descent: at each iteration (j) the gradient of L with respect to the weights (w) is calculated to update the weights in the opposite direction with respect to the gradient:

$$w(j+1) = w(j) - \alpha \frac{\partial}{\partial w} L\left(w(j)\right) \tag{2.2}$$

where α is the so-called learning rate, that determines the learning step size at each iteration.

2.1.1 Neural networks classification

The NNs can be divided in different classes, depending on how the neurons are connected and how the information flows through the layers. In this section, only few classes will be presented, in order to give a general overview of the different NN architectures, considering the most relevant ones nowadays.

Feed-Forward Neural Networks

The Feed-Forward Neural Networks (FFNNs) are the simpler existing NN architectures and the first that have been defined. The FFNNs are also called Multi Layer Perceptron (MLP), since they are composed by multiple layers of interconnected units: the general scheme shown in Figure 2.2 represents a FFNN. The main characteristic of this architecture is that in a FFNN the information flows directly from the input to the output and the neurons of a layer are connected only to the neurons of the next layer, without feedback connections.

Recurrent Neural Networks

The Recurrent Neural Networks (RNNs) [23] are a class of NNs derived from the FFNNs, where, unlike the latter, a feedback connection is present between the neurons in the same layer. These networks are characterized by the presence of an internal state, making possible to take into account temporal dependencies. Consequently, this architecture is mainly used to process sequential data and in application like time series prediction, natural language processing, translation, handwriting recognition, and so on.
This thesis is mainly focused on the use of a particular RNN architecture, so this class will be explained more in detail in the following in Section 2.2.

Convolutional Neural Networks

The Convolutional Neural Networks (CNNs) [24, 25] are derived from the FFNNs too and are the most diffused and important architectures used for images classification.
The CNNs are inspired by the works of two neurophysiologists, Hubel and Wiesel, about the neurons connections of the animal visual cortex, where each neuron responds only to a small portion of the visual field, called receptive field. Similarly, in the CNNs, each neuron receives input only from a restricted area of the previous layer, called receptive field too. These networks take their name from the mathematical operation of the convolution, since they use convolution instead of matrix multiplication in at least one layer [26]. The input features of a CNN, typically images, are transformed through convolution, providing a feature map i.e. an abstraction of the input image, that is then passed in input to the subsequent CNN layer, where each

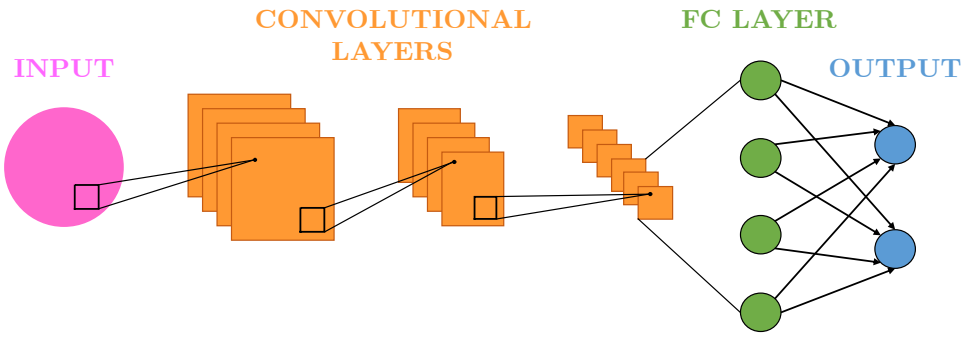


Figure 2.3: CNN architecture.

neuron receives an input only from its receptive field. After several CNN layers and transformations, the last layer of a CNN is a Fully-Connected layer (FC layer), like in regular FFNNs, where the final classification is performed. In Figure 2.3 an example of a CNN with 3 convolutional layers is shown.

Transformers

Transformers [27] are one of the most promising and recent NN architecture, developed by a Google Brain team in 2017. They are designed to process sequential data, so are mostly used in natural language processing and computer vision, where are fast replacing RNNs in time series tasks [28]. Transformers are based on an encoder-decoder structure, where the input is received as a sequence, converted in an encoded vector that is then decoded back in another sequence. Moreover, they are entirely based on self-attention mechanisms. The attention mechanism [29] has already been implemented in RNNs to improve their performances, especially in translation tasks, enabling the network to focus on certain part of the sequence input, deciding for the encoding of a given word how much important are the other words of the input sequence.
The Transformer architecture is similarly based on an encoder-decoder structure but without recurrence or convolution to produce an output. The self-attention mechanism of the Transformers instead computes a representation of the inputs sequence, with respect to the other elements of the same sequence. The attention is composed by a query and a key-value pair for each word in the sequence, the output is then a weighted sum of the values computed as a scaled-dot product. For each attention unit there are three weight matrices (query weights, key weights, value weights), called attention head, that are multiple for each layer.
The main advantage of the Transformers with respect to the RNNs is their capability to process the input sequence all together, so it is possible to parallelize the job and reduce the training time. Moreover, the short and long term dependencies are clearly modeled.

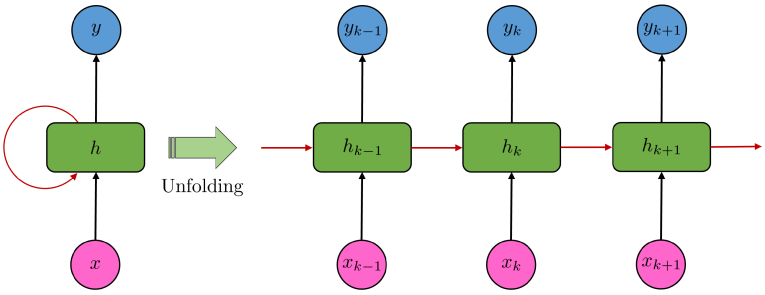


Figure 2.4: Temporal unfolding of a RNN.

2.2 Recurrent Neural Network

In this section, the RNNs are described in depth, since this architecture is the one used in the practical applications of this thesis.
The RNNs, firstly presented in [23], are a specific class of NNs characterized by a recurrent loop where the output of a neuron is fed-back in input to its replica at the next time instant. It means that at each time instant the output of a neuron is function of its previous outputs, that have been computed using the same update rules. This dependence can be mathematically formulated as:

$$h(k) = W_{rec}h(k-1) + W_{in}x(k) + b \tag{2.3}$$

where the value of the hidden layer at instant k, $h(k)$, depends on the inputs at the current instant, $x(k)$, multiplied by the weights, W_{in}, and a bias, b, but also on its own value at the previous time instant, $h(k-1)$, multiplied by the recurrent weight matrix, W_{rec}.
To better understand this process, the RNNs can be seen using the concept of temporal unfolding, as in Figure 2.4, where the computational graph of the network, on the left, can be seen as a repeated graph that shows the temporal dependencies of the network layers, on the right. In Figure 2.4 it is shown that at time instant k the hidden layer receives in input both the input $x(k)$ and the value of the hidden layer at time instant $k-1$, giving a visual idea of the information flow through the time.
Since this structure is able to take into account time dependencies, RNNs are particularly suitable in processing temporal data. However, their recurrent structure leads to some problems during the gradient calculation. In fact, it is well known that the RNNs suffer during the training phase of the vanishing and exploding gradient problem [30, 31], that happens when the same operations is performed repeatedly on the same element at each time instant of a temporal sequence. Since the training of a RNN is based on the backpropagation algorithm [32] and a gradient descent method, at each iteration the weights are updated considering the gradient of the loss function with respect to the weight itself, so as explained in [33], the long term

components can grow exponentially or can go exponentially to 0, leading in a severe worsening of the optimization. In particular, with a vanishing gradient it is difficult to understand in which direction the parameters have to move to reduce the cost function, while with an exploding gradient the training phase may be unstable.
To avoid the vanishing gradient problem, some techniques can be applied during the training of a network, like a proper weight initialization, gradient clipping, using a linear activation function, and so on. Moreover, some particular variations of the RNNs have been designed to solve this problem, called gated RNNs. The successful idea behind the gated RNNs is the substitution of the RNN internal state with an internal self-loop, regulated by a gating system that manages the information flow, deciding also when an information is no more necessary and can be forgotten. The Long Short-Term Memory (LSTM) networks belong to this group.

2.2.1 LSTM network

The LSTMs, firstly introduced in [34, 35], are a particular type of RNNs, that thanks to their specific units architecture, called memory cells, are able to learn the information dependencies better than the simple RNNs. The LSTM memory cell presents an internal self-loop, in addition to the outer feedback present in all the RNNs, and a gating system that regulates the flow of information. In this way, the gradient flows during time, even for long periods, and its derivative do not explode nor vanish.
The mathematical formulation of a single memory cell, represented in Figure 2.5, is expressed as follows:

$$c(k) = f(k) \times c(k-1) + i(k) \times \tilde{c}(k) \tag{2.4a}$$

$$h(k) = o(k) \times tanh\,(c(k)) \tag{2.4b}$$

where $c(k) \in \mathbb{R}^n$ is the internal cell state at time k, $x(k) \in \mathbb{R}^m$ is the input at time k and $h(k) \in \mathbb{R}^n$ is the hidden state at time k, with n the cell state dimension, that corresponds to the number of LSTM neurons, m the number of features, $\times$ is the Hadamard product and $tanh$ is the hyperbolic tangent, used as activation function.
Four structures, called gates, regulates the internal loop of the cell adding or removing information to modify the value of the cell state at each time instant. The first gate is the forget gate, $f(k)$, that decides which information is removed from the cell state $c(k)$, multiplying it by its past value $c(k-1)$, as can be seen in the first part of (2.4a). Then the input gate, $i(k)$, chooses which value of the cell state is updated, while the input activation gate, $\tilde{c}(k)$, creates a new candidate value of the cell state. The last gate is the output gate, $o(k)$, that determines which information contained in the cell state is going in output. The hidden state is then defined in (2.4b), where the output

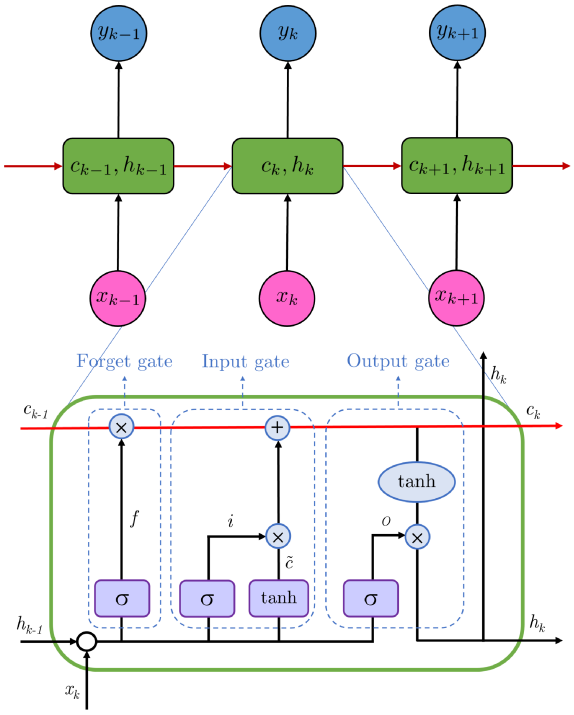


Figure 2.5: Scheme of a single LSTM cell.

gate is multiplied by a transformation of the cell state through hyperbolic tangent function. These gates are expressed as:

$$f(k) = \sigma(W_f x(k) + U_f h(k-1) + b_f) \tag{2.5a}$$
$$i(k) = \sigma(W_i x(k) + U_i h(k-1) + b_i) \tag{2.5b}$$
$$\tilde{c}(k) = tanh(W_{\tilde{c}} x(k) + U_{\tilde{c}} h(k-1) + b_{\tilde{c}}) \tag{2.5c}$$
$$o(k) = \sigma(W_o x(k) + U_o h(k-1) + b_o) \tag{2.5d}$$

where W_f, W_i, W_o, $W_{\tilde{c}} \in \mathbb{R}^{n \times m}$ are the input weight matrices, U_f, U_i, U_o, $U_{\tilde{c}}$ $\in \mathbb{R}^{n \times n}$ are the previous output weight matrices and $b_f, b_i, b_o, b_{\tilde{c}} \in \mathbb{R}^{n \times 1}$ are the biases. These quantities are specific of each gate and obtained during the training of the network. The chosen activation functions are the hyperbolic tangent *tanh* and σ, the sigmoid function, equal to:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

Several LSTM cells are combined in the so-called LSTM layer and the LSTM network can be composed by one or more stacked layers, representing the hidden layers. In regression applications it is common use that the output of the (last, if more) LSTM layer is passed to a fully connected layer without activation function, that computes a linear combination, according to the

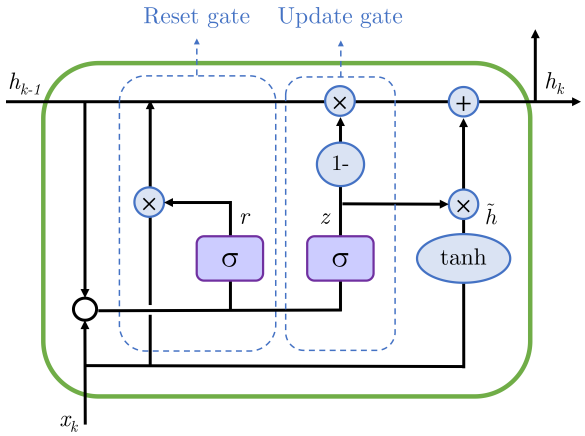


Figure 2.6: Scheme of a GRU cell.

following equation:

$$y(k) = W_y h(k) + b_y \tag{2.7}$$

where $W_y \in \mathbb{R}^{p \times n}$ is the matrix of the output weights, with p the number of outputs of the network, and $b_y \in \mathbb{R}^{p \times 1}$ the bias. W_y and b_y are determined during the training of the network as well.
Thanks to this structure, LSTMs show their strength especially when dealing with temporal data and when temporal lags must be taken into account [33, 35–38].

2.2.2 Other relevant RNNs

Gated Recurrent Unit

Gated Recurrent Units (GRUs) [39], are RNNs with a gating architecture, similarly to the LSTMs. A single GRU memory cell, represented in Figure 2.6, is described by the following mathematical equations:

$$z(k) = \sigma(W_z x(k) + U_z h(k-1) + b_z) \tag{2.8a}$$
$$r(k) = \sigma(W_r x(k) + U_r h(k-1) + b_r) \tag{2.8b}$$
$$\tilde{h}(k) = tanh(W_h x(k) + U_h \left(r(k) \times h(k-1)\right) + b_h) \tag{2.8c}$$
$$h(k) = z(k) \times h(k-1) + (1 - z(k)) \times \tilde{h}(k) \tag{2.8d}$$

where $z(k)$ is the update gate, $r(k)$ is the reset gate, $\tilde{h}(k)$ is the candidate activation, $h(k)$ is the output, $x(k)$ is the input. The matrices W and U and the biases b are the weights to be tuned.
Unlike the LSTM case, the cell state and the output gate are not present in a GRU memory cell; there are only an update and a reset gate, that decide the information to pass to the output. Since it lacks of an output gate, there are less parameters to tune with respect to the LSTM, so the training can be

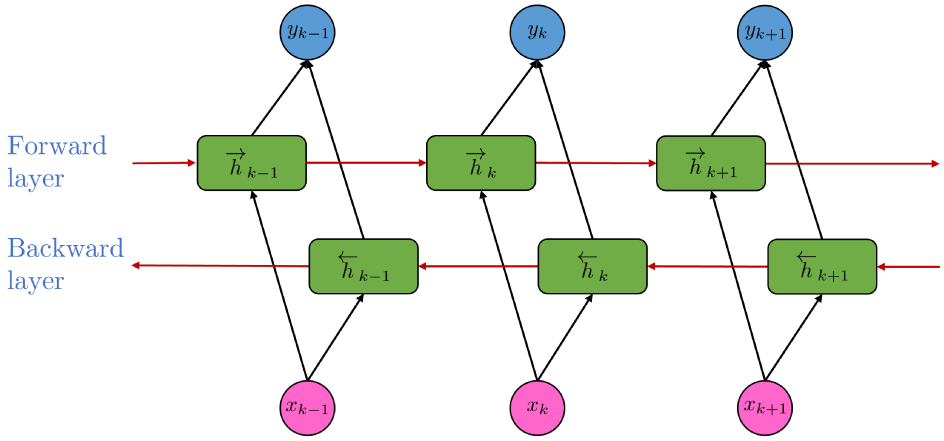


Figure 2.7: Temporal unfolding of a BRNN.

faster. It has also been proved that the performances of the two architectures are comparable and that the preference of one architecture over the other depend on the task and on the specific dataset [40].

Bidirectional Recurrent Neural Network

The Bidirectional Recurrent Neural Networks (BRNNs) [41] are a variation of the classical RNNs, more effective when the input's context is important (handwriting recognition, translation, speech recognition, and so on). A BRNN is a combination of two RNNs that moves in two different directions: a "regular" RNN, that moves forward from the start to the end of the time sequence, and a second RNN that moves backward, from the end to the start. The two flows of information are independent and do not interact. In this way, the BRNN is trained using all the available input information for the current time frame, both from the past and from the future. In Figure 2.7, the temporal unfolding of a BRNN is shown for three time instants, highlighting how the two flows are independent from each other. The red arrows represent the internal feedback loops of the BRNN.
Another existing variation is the Bidirectional LSTM that works in the same way, considering two LSTMs, one forward and one backward.

2.3 NNs in system identification and control fields

In 1989 the so-called universal approximation theorem was proved for FFNNs [42, 43], saying that there always exists a neural network capable to approximate any function $f(x)$, no matter its complexity. As said in [26]:

> "The universal approximation theorem means that regardless of what function we are trying to learn, we know that a large MLP will be able

> to represent this function. However, we are not guaranteed that the training algorithm will be able to learn that function."

In [43] the theorem was proved in particular for a sigmoid activation function, then during years the result was extended also to other classes of activation functions and recently also to RNNs [44].
In the early 90s, in [21, 22] NNs were used for the first time for identification and control of dynamical systems. The main idea was to use FFNNs as one step-ahead predictors, like in [45], in order to predict future outputs, given past inputs and outputs. Since the FFNNs are static and memoryless, they were not particularly suitable for this goal and the results were not accurate, especially in learning long-term dependencies. RNNs instead seemed to be more appropriate for these applications. Thanks to their specific structure, the RNNs are able to take into account the temporal dependencies of the data, being in that way particularly able to describe dynamical systems [46]. Moreover, they are also able to represent non linear behaviours, very useful when more classical linear identification methods cannot be used [45].
The use of NNs in identification and control fields was then extended more specifically to RNNs: for example in [47] a RNN was used to identify a non linear plant, then controlled with a standard PID, in 1995. In the same years, to find a solution to the vanishing and exploding gradient problem, some new architectures were proposed, like the LSTMs [34] in 1997. The LSTMs rapidly increased their popularity in several fields of application, becoming soon the state-of-the-art for recurrent models, thanks to their very powerful learning ability, especially when working with time-series data and it is important to consider long-term dependencies [37]. In [48] a detailed review about LSTMs is presented, outlining the most diffused and interesting applications of these networks, analyzing more than 400 peer-reviewed papers. As result, the most relevant applications regard time series prediction, natural language processing, sentiment analysis, image and video captioning, computer vision, text recognition, but also traffic analyses, healthcare applications, computer science, sound recognition. Even if this is a recent review, the application of LSTMs in system identification and control problems is never mentioned, confirming how in this field their popularity is still very marginal.
One of the first works in which LSTMs are used for dynamic system identification is [49], in 2017. Some previous works probably exist, but this one is particularly relevant: the identification process is designed minimizing the error between the LSTM model and the real plant model, relying on the Universal Approximation Theorem. The resulting performances are highly better than classical RNNs. In 2020, links between deep learning and system identification are shown in [50]. The goal of both topics is "to infer models from observed data". On one hand, a key decision in system identification is the choice of the model structure, estimating the parameters and then val-

idating the model: if this decision needs to be changed, the procedure may become a loop. Moreover, the parameters estimation is essentially carried out minimizing the error between real and estimated outputs. This procedure seems not so different from the training of a NN. On the other hand, the LSTM equations can be seen as a general NonLinear State-Space model:

$$x(t+1) = f(x(t), y(t), u(t), \theta) \tag{2.9a}$$
$$\hat{y}(t|\theta) = h(x(t), \theta) \tag{2.9b}$$

where x is the state, y the output, u the input and θ the parameters of the model. Considering this duality, the conclusion of [49] is that

> "Deep learning is a task that fits very well into the system identification framework. What is special is that the choice of model structure $\mathcal{M}$ is done within the family of deep networks. But it is still an (statistical) estimation problem".

Starting from first few works in 2012, the presence of LSTMs in this field has considerably grown and this can be easily demonstrated: searching on the *Web of Science* the terms "LSTM AND system AND identification", the obtained result is shown in Figure 2.8 (citation report graphic is derived from Clarivate *Web of Science*, Copyright Clarivate 2023. All rights reserved. Visited on 06/01/2023). A clear growing trend can be observed and it will probably further increase in the next future. Moreover, considering the most cited works in this list, many different topics are present, highlighting how this approach can be applied in different fields: fault detection [51–54], nonlinear system identification [55], diagnostic algorithms in nuclear plants [56, 57], parameters identification for robot manipulators [58], state-of-charge estimation of Li-ion batteries [59], nonlinear structural seismic response prediction [60], and so on.
In a recent survey [61], the potentiality of RNNs in the control field are shown, classifying the different approaches in the use of NNs for control in six groups:

1. *NN as model of a plant*: the NN is trained on plant data as a black-box model of the considered system. Then this model can be used as reference for control purposes in model-based control, like in Model Predictive Control (MPC).

2. *NN as a part of a grey-box model*: the NN is used to model unknown or complex terms present in physical models based on first-principle equations. This approach includes the so-called physics-based models, that seek to include physical knowledge of the system in the network, directly in the structure of the RNN or in the loss function. An interesting survey about this kind of networks can be found in [62].

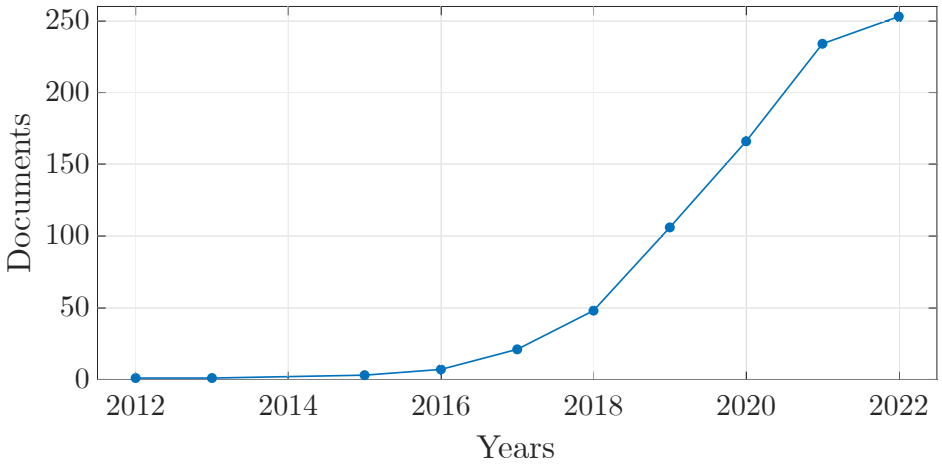


Figure 2.8: Number of publications in *Web of Science* of "LSTM AND system AND identification" from 2012.

3. *NN as model of the uncertainty*: the NN is trained to model the uncertainty of an existent known model, in order to improve its accuracy. The modelled uncertainty can be used to design the control algorithm.

4. *NN as approximator of computationally-intensive control laws*: given an existent trustworthy model with a control law that has a high computational cost, the NN can be used to approximate it offline, thanks to its approximation properties and low computational load.

5. *NN as controller directly synthesized from data*: the NN is not used to obtain a model but to obtain instead the control law directly from the plant data.

6. *NN for Reinforcement Learning*: the NN learns the control law of a system with reinforcement learning techniques, exploiting several closed-loop simulations, ideal when a simulator of the real system exists.

The first approach, in which the NNs are used as models of dynamical systems with the goal of control design, is probably the most common nowadays and also the one more emphasized in [61]. Moreover, this approach is also the main concept that drives this thesis.
Lastly, in [61] a discussion of some unresolved problems on this topic is carried out, highlighting the open issues that make difficult the diffusion of RNNs in the control field, with respect to their fast diffusion in other fields of study: robustness of the identified model, safety verification, interpretability and consistency of the model with respect to the physical laws characterizing the system. These issues are mainly related to the generalization capabilities of the network and to its connection to the real plant. If these problems may

be solved, it will be possible in the future to test the models and their control schemes on the real plants, comparing NN models with more traditional ones in order to assess and better understand advantages and disadvantages.

2.3.1 RNN stability analysis

In order to be successfully employed in system identification and control, RNNs need to satisfy some stability requirements. In [61] stability results for the main RNNs families are shown: Neural Nonlinear AutoRegressive eXogenous (NNARX), Echo State Networks (ESN), LSTM and Gated Recurrent Units (GRU). In particular, sufficient conditions to guarantee the Input-to-State Stability (ISS) and Incremental Input-to-State Stability (δISS) of the networks are provided. These properties are used to design an observer, then used in a MPC scheme, guaranteeing the asymptotic stability of the closed-loop system.
To better understand the stability properties, firstly some useful definitions are recalled in the following.

Definition 1 ($\mathcal{K}$-function)**.** *A continuous function $\gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{K}$ function if $\gamma(s) > 0$ for all $s > 0$, it is strictly increasing and $\gamma(0) = 0$.*

Definition 2 ($\mathcal{K}_\infty$-function)**.** *A continuous function $\gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{K}_\infty$ function if it is a $\mathcal{K}$ function and $\gamma(s) \to \infty$ for $s \to \infty$.*

Definition 3 ($\mathcal{KL}$-function)**.** *A continuous function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{KL}$ function if $\beta(s, k)$ is a $\mathcal{K}$ function with respect to s for all k, it is strictly decreasing in k for all $s \geq 0$, and $\beta(s, k) \to 0$ as $k \to \infty$ for all $s > 0$.*

Definition 4 (ISS [63])**.** *The dynamical system $x(k+1) = f(x(k), u(k))$ is Input-to-State Stable (ISS) if there exist functions $\beta \in \mathcal{KL}$ and $\gamma_u \in \mathcal{K}_\infty$ such that for any $k \geq 0$, any initial condition $\bar{x}$ and any input sequence $\{u(0), u(1), ..., u(\tau)\}$, it holds that:*

$$\|x(k)\| \leq \beta(\|\bar{x}\|, k) + \gamma_u(\max_{h \geq 0} \|u(h)\|). \tag{2.10}$$

Definition 5 (δISS [64])**.** *The dynamical system $x(k+1) = f(x(k), u(k))$ is Incrementally Input-to-State Stable (δISS) if there exist functions $\beta \in \mathcal{KL}$ and $\gamma_u \in \mathcal{K}_\infty$ such that for any $k \geq 0$, any pair of initial conditions $\bar{x}_a$ and $\bar{x}_b$, any pair of input sequences $\{u_a(0), u_a(1), ..., u_a(\tau)\}$ and $\{u_b(0), u_b(1), ..., u_b(\tau)\}$, it holds that:*

$$\|x_a(k) - x_b(k)\| \leq \beta(\|\bar{x}_a - \bar{x}_b\|, k) + \gamma_u(\max_{h \geq 0} \|u_a(h) - u_b(h)\|). \tag{2.11}$$

The ISS of a system guarantees the boundedness of the state, given a bounded input. The δISS of a system guarantees that the smaller is the distance between two inputs, the smaller is the distance between the corresponding state

trajectories, asymptotically, independently on the initial states. In this way the modeling is independent from the initialization, particularly useful with NNs. Moreover, δISS imply ISS, since it is a stronger condition.
Given these definitions, in [61] the following Proposition is enunciated:

Proposition 1 ([61]). *Under suitable conditions on their weights Φ, NNARXs [65], ESNs [66], LSTMs [67], and GRUs [68] are guaranteed to be ISS and δISS. These conditions can be generally regarded as nonlinear inequalities on the weights of the network, denoted by*

$$\nu(\Phi) < 0 \tag{2.12}$$

If the condition on the weights of the network in (2.12) holds, then the RNN is ISS and/or δISS. This condition can be easily implemented during the training phase of the network as a penalization of the loss function.

2.3.2 LSTM stability analysis

In this thesis, the aim is mainly focused on LSTMs so in the following their stability property is more deeply described. A first condition is defined in [69], then extended in [67].

Remark. In order to be more consistent with the typical system identification notation, in this section the input is referred as u and the state as $\chi = \begin{bmatrix} c^T & h^T \end{bmatrix}^T \in \mathbb{R}^{2n}$.

First of all, Equation (2.4) is rewritten in state space form, considering in this way the network as a discrete-time, invariant, non linear dynamical system. The output transformation of the system is instead defined by Equation (2.7). In particular:

$$c(k+1) = f(k) \times c(k) + i(k) \times \tilde{c}(k) \tag{2.13a}$$
$$h(k+1) = o(k) \times \tanh(c(k+1)) \tag{2.13b}$$
$$y(k) = W_y h(k) + b_y \tag{2.13c}$$

In the same way Equation (2.5) can be rewritten as:

$$f(k) = \sigma(W_f u(k) + U_f h(k) + b_f) \tag{2.14a}$$
$$i(k) = \sigma(W_i u(k) + U_i h(k) + b_i) \tag{2.14b}$$
$$\tilde{c}(k) = tanh(W_{\tilde{c}} u(k) + U_{\tilde{c}} h(k) + b_{\tilde{c}}) \tag{2.14c}$$
$$o(k) = \sigma(W_o u(k) + U_o h(k) + b_o) \tag{2.14d}$$

In [69], the inputs are assumed bounded, with

$$u \in \mathcal{U} = [-u_{max}, u_{max}]^m \tag{2.15}$$

and this holds considering for example a physical saturation of the input or can be obtained through normalization techniques of the data. Consequently,

considering the activation functions bounds, also the output state is bounded, so

$$h \in [-1, 1]^n \tag{2.16}$$

Considering the Definitions 1, 2, 3 and 4, the following Definitions and Theorems are valid.

Definition 6 (ISS Lyapunov function [70]). *A continuous function $V : \mathbb{R}^n \to \mathbb{R}_+$ is called an ISS-Lyapunov function if there exist functions $\psi_1, \psi_2, \psi \in \mathcal{K}_\infty$ and $\sigma_u, \sigma_b \in \mathcal{K}$ such that for all $\chi(k) \in \mathbb{R}^{2n}$, for all $b_c \in \mathbb{R}^n$ and $u(k) \in \mathbb{R}^m$, it holds that:*

$$\psi_1(|\chi(k)|_2) \leq V(\chi(k)) \leq \psi_2(|\chi(k)|_2) \tag{2.17a}$$

$$V\big(\chi(k+1)\big) - V\big(\chi(k)\big) \leq -\psi_2(|\chi(k)|_2) + \sigma_u(|u(k)|_2) + \sigma_b(|b_c|_2) \tag{2.17b}$$

Theorem 1 ([70]). *If the system admits a time invariant ISS Lyapunov function such that* (2.17) *hold, then it is ISS in the sense specified in Definition 4.*

Based on these results, the Theorem regarding the LSTM stability is formulated in [69] as follows:

Theorem 2 ([69]). *Given the LSTM network (2.13), if*

$$\begin{aligned} (1 + \sigma_g\left(|[W_o\, U_o\, b_o]|_\infty\right))\ \sigma_g\left(|[W_f\, U_f\, b_f]|_\infty\right) &< 1 \\ (1 + \sigma_g(|[W_o\, U_o\, b_o]|_\infty))\, \sigma_g(|[W_i\, U_i\, b_i]|_\infty)\, |U_{\tilde{c}}|_1 &< 1 \end{aligned} \tag{2.18}$$

then (2.13) *is Input-to-State stable with respect to $u \in \mathcal{U}$ and to b_c.*

Since δISS is a stronger condition than ISS, in a more recent work [67] some more stability conditions for LSTMs are presented. Starting from the same assumptions regarding the boundedness of the inputs and the states of the network expressed in Equations (2.15) and (2.16), the following theorems are valid:

Theorem 3 ([67]). *The LSTM network (2.13) is ISS with respect to the input u and bias b_c if $\rho(A) < 1$, where*

$$A = \begin{bmatrix} \bar{\sigma}^f & \bar{\sigma}^i \|U_c\| \\ \bar{\sigma}^o \bar{\sigma}^f & \bar{\sigma}^o \bar{\sigma}^i \|U_c\| \end{bmatrix}. \tag{2.19}$$

denoting

$$\bar{\sigma}^f = \sigma(\|[W_f u_{max}\ U_f\ b_f]\|_\infty) \tag{2.20}$$

$$\bar{\sigma}^i = \sigma(\|[W_i u_{max}\ U_i\ b_i]\|_\infty) \tag{2.21}$$

$$\bar{\sigma}^o = \sigma(\|[W_o u_{max}\ U_o\ b_o]\|_\infty) \tag{2.22}$$

where $\bar{\sigma}^f$, $\bar{\sigma}^i$ and $\bar{\sigma}^o$ are upper bounds of the gate vectors (2.14a), (2.14b) and (2.14d) respectively.
Applying the Jury criterion, this condition on the eigenvalues of the matrix A can be transformed into a condition on the weights of the LSTM, as follows:

Proposition 2 ([67]). *The Schur stability of the matrix A defined in* (2.19) *is ensured if the following inequality holds:*

$$\bar{\sigma}^f + \bar{\sigma}^o \bar{\sigma}^i \|U_c\| < 1. \tag{2.23}$$

In a similar way, this approach can be used also to establish a sufficient condition for δISS.

Theorem 4 ([67]). *The LSTM network (2.13) is δISS if $\rho(A_\delta) < 1$, where*

$$A_\delta = \begin{bmatrix} \bar{\sigma}^f & \alpha \\ \bar{\sigma}^o \bar{\sigma}^f & \alpha \bar{\sigma}^o + \frac{1}{4} \bar{\sigma}^x \|U_o\| \end{bmatrix}. \tag{2.24}$$

denoting

$$\alpha = \frac{1}{4} \|U_f\| \frac{\bar{\sigma}^i \bar{\sigma}^c}{1 - \bar{\sigma}^f} + \bar{\sigma}^i \|U_c\| + \frac{1}{4} \|U_c\| \bar{\sigma}^c \tag{2.25}$$

$$\bar{\sigma}^c = \tanh(\|[W_c u_{max} \; U_c \; b_c]\|_\infty) \tag{2.26}$$

$$\bar{\sigma}^x = \tanh\left(\frac{\bar{\sigma}^i \bar{\sigma}^c}{1 - \bar{\sigma}^f}\right) \tag{2.27}$$

where $\bar{\sigma}^c$ is an upper bound for the gate (2.14c) and $\bar{\sigma}^x$ is an upper bound for the term $\tanh(c(k+1))$ in the state equation (2.13b).
Also in this case, using the Jury criterion, it is possible to obtain a condition on the weights of the network:

Proposition 3 ([67]). *The Schur stability of matrix A_δ is ensured if the following inequality holds:*

$$-1 + \bar{\sigma}^f + \alpha \bar{\sigma}^o + \frac{1}{4} \bar{\sigma}^x \|U_o\| < \frac{1}{4} \bar{\sigma}^f \bar{\sigma}^x \|U_o\| < 1. \tag{2.28}$$

Moreover, in [67] is observed that the satisfaction of the sufficient condition for δISS (2.28) implies the satisfaction of the sufficient condition for ISS (2.23).
The conditions presented in Theorem 2, Propositions 2 and 3 are sufficient conditions, explicitly dependent from the LSTM weights and can be imposed as constraints during the training of the network or used to check a posteriori its stability properties.

2.4 Comparison with other NNs in system identification

Besides the advantages of using RNNs and in particular LSTMs depicted in the previous sections, other type of NN architectures have been employed recently in the system identification and control fields. In [71] the relationship between CNNs and system identification model structures is shown, proving that CNNs are as able as RNNs in applications that employ temporal data, in particular considering a CNN architecture called Temporal Convolution Network (TCN) [72].

2.4.1 Temporal Convolutional Network

Considering the structure of a CNN, as described in Section 2.1.1, it is possible to formalize the mathematical operation of the convolution also when dealing with time series data. Formally for a sequence input $x = x_0, \ldots, x_T$ and a filter $f\colon \{0, ..., k-1\} \to \mathbb{R}$, the convolution F on the element s of the sequence is:

$$F(s) = (\mathbf{x} * f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s \cdot i} \tag{2.29}$$

where $\mathbf{x}$ is the input, f is a filter (also called kernel) that represents the learnable parameters of the layer, k is the size of the filter, $s \cdot i$ the direction of the past. The asterisk $*$ denotes the convolution operation.
The TCNs are able to consider long-term dependencies through temporal convolutional filters, according to two principles [73]:

1. the input sequence can be of any length and transformed to an output of the same length; this is done using a fully convolutional layer where the hidden layer have the same dimensions of the input layer, through zero padding;

2. the convolutions are causal so there are no leaks of information between future and past.

Given an input sequence $x_0, \ldots, x_T$, the goal of the sequence modeling task is to find the corresponding outputs $y_0, \ldots, y_T$, and to fulfill the causality only the previous inputs already observed are considered. To do this in practice, the TCN employs causal convolutions where, at time t, the output is convolved with elements of the lower layers only at time t or before. The disadvantage of this technique is that the network can consider only temporal dependencies with size equal to the network's depth and for longer task the required depth of the network would be too big.
To overcome this problem, two solutions are employed in TCNs from an architectural point of view: dilated convolutions and residual blocks. With a dilated convolution the filter is applied to a larger region with respect to the standard one, using a fixed step to skip input values, like in [74]. Considering the convolution equation in (2.29), the dilated convolution F_d on the element s of the input sequence is:

$$F_d(s) = (\mathbf{x} *_d f)(s) = \sum_{i=0}^{k-1} f(i) \cdot \mathbf{x}_{s - d \cdot i} \tag{2.30}$$

where d is the dilation factor and $*_d$ denotes the dilated convolution. In this case the direction of the past is $s - d \cdot i$. Notice that with $d = 1$ the dilated convolution becomes a standard convolution and Equation (2.30) is equivalent to Equation (2.29). In Figure 2.9 an example of a dilated

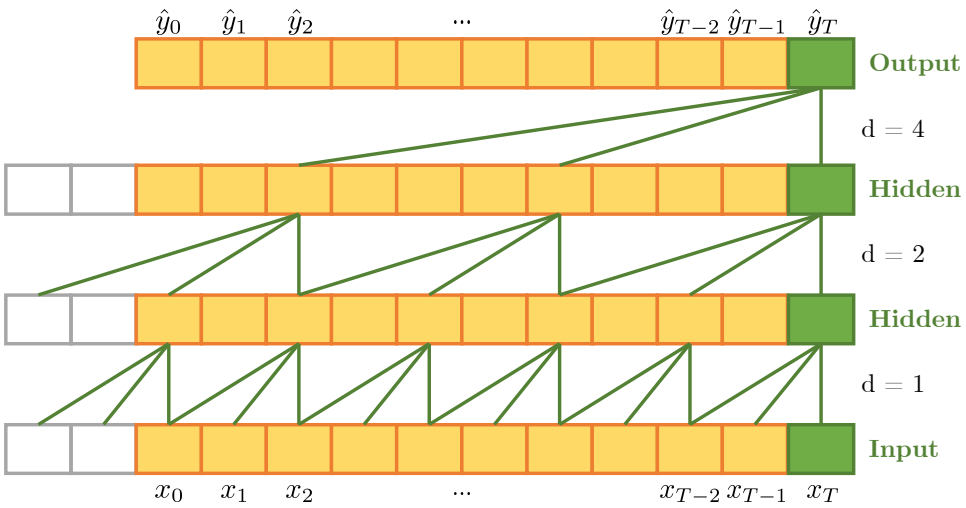


Figure 2.9: Dilated causal convolution example.

convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$ is shown. The second solution is the usage of a residual block [75] that replaces in TCNs the standard convolutional layer. A residual block is a block that learns the residual functions with respect to the input layer. In practice, the residual block output is calculated adding the residual of a transformation $\mathcal{F}(\mathbf{x})$ to the input itself $\mathbf{x}$, as shown in Figure 2.10. As can be seen in the left branch of Figure 2.10, the transformation $\mathcal{F}(\mathbf{x})$ is resulting from different layers and one or more activation functions. In the case of the TCN, the block Weight Layer is a dilated causal convolutional layer, with usually a ReLU activation function and eventual weight normalization and dropout layers. Unlike the standard residual mechanism, where the input $\mathbf{x}$ is summed directly to the output of $\mathcal{F}(\mathbf{x})$, in the TCN an additional 1D convolutional layer can be added if the two quantities (input and output of the residual function) have different lengths, as can be seen in the right branch in Figure 2.10. Lastly, in Figure 2.11 it is shown how in a TCN these two architectural elements are combined, in a case with $k = 3$ and $d = 1$, where in blue the additional 1D convolution of the residual block is added to the output, as in Figure 2.10. Figures 2.9, 2.10 and 2.11 are adapted and redrawn from [73] and [75].

2.4.2 TCNs and LSTMs comparison

Thanks to their structure, it is possible to employ TCNs for sequence modeling. In [73], the advantages and disadvantages for the use of TCNs, with respect to RNNs, are listed. The most relevant advantages concern parallelism, since different convolutions can be executed in parallel because the layers use the same filter; the presence of a stable gradient that avoids the vanishing or exploding gradient problem; low memory requirements for train-

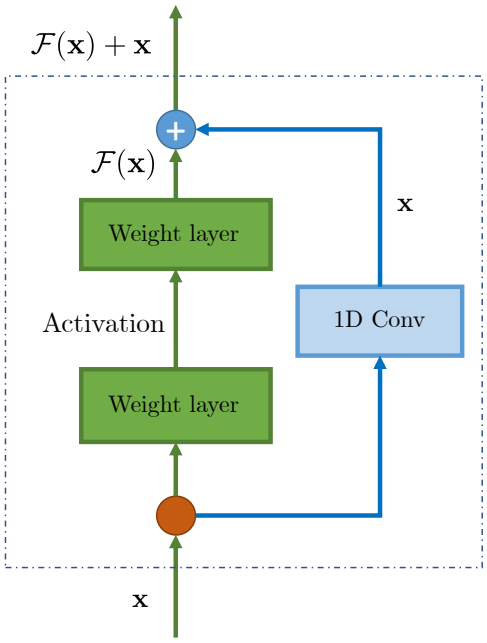


Figure 2.10: Residual block example.

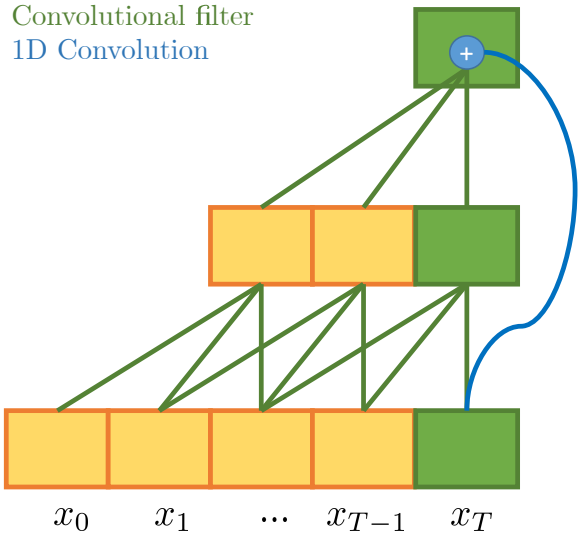


Figure 2.11: Residual connection in TCN, with k=3 and d=1.

ing and the possibility to have variable length inputs. On the other hand, TCNs need more data storage during evaluation and are subject to potential parameters change for a transfer of domain, i.e. when a model is transferred to a domain that needs a longer memory. Moreover, in [73] the TCN is compared to canonical recurrent architectures like RNN, LSTM and GRU, on classical sequencing benchmarks (synthetic stress tests, polyphonic music modeling, character-level language modeling, word-level language modeling) showing outperforming results.
Some studies have also been conducted considering TCNs for system identification. In [71], TCNs are applied on three system identification problems: a nonlinear toy problem, a Silverbox benchmark and a F-16 ground vibration test; the obtained results are compared with the ones obtained with a LSTM network. The results of the two networks are very similar, showing good performances; the long memory offered by the dilation factor of the TCN does not give any advantages, probably for the relatively short memory of these dynamical systems. The authors' conclusion is that both architecture can work well for system identification when long-term memory is needed, with interesting applications in a MPC.
Anyway, the application of both the LSTMs and the TCNs is still very limited in this field, even if they look promising. In [76] further disadvantages of the use of CNNs for system identification and control are listed: the need of large-scale datasets and high computational power, given their deeper and deeper structures; the need of skills and experience for the choice of the hyperparameters; finally, the lack of a solid theory. In fact, they reach good performance but it is still not completely clear the cause of that, which is one of the most important point for our analysis.

2.5 Neural networks recap

In these years, the interest of the system identification community in NNs has continuously grown. At the same time, the desire to understand their working principles, to better suit the system identification task, is leading to a new branch of literature, as shown in the previous sections, trying to apply system identification tools to NNs in order to better exploit them for the desired task. However, the interest of the entire scientific community in ML is leading to a continuous growth of this field, where NNs are being studied from several different point of views and for many different applications. For example, considering that the Transformers are fast replacing RNNs and LSTMs for processing sequential data, it would not be surprising if in the next years the Transformers will be able to outclass LSTMs also in the system identification task.
In this chapter, an overview of the most diffused architectures has been presented, showing their characteristics, their main field of applications and

their key strengths. Moreover, it has been shown how these networks can be employed in the system identification and control fields. Even if other architectures can achieve interesting results too, as things stand at present, LSTMs represents a very promising NN solution for system identification, also thanks to their clear composition from a mathematical point of view. The possibility to rewrite the equations of the LSTM in a state-space form gives the possibility to analyze the network's properties using also system identification tools and not only the ML ones. This thesis aims to highlight the link between these two fields, presenting practical applications in which LSTMs have been successfully employed for the identification of dynamical systems.

Chapter 3

Modeling of an industrial autoclave

In this chapter a physical model and a LSTM network are used to model the sterilization process of an industrial autoclave. The two models are built with different goals: the physical model is a novelty of the state of the art, designed ad-hoc on the specific process, to build a simulator; the LSTM is trained considering the ISS properties enunciated in Theorem 2 (Section 2.3.2), with control purposes. Pros and cons of both approaches are presented and analyzed, discussing the results obtained in this specific application.

3.1 Motivation and state of the art

In order to meet the principles of Industry 4.0, the modeling of complex processes is required and now achievable, thanks to the availability of a large amount of data. The continuous evolution of the market, due to a growing technological innovation, has led companies to adapt their design processes and leverage their capabilities to remain competitive.
An interesting case study is represented by the autoclaves, machines constituted by pressurized chambers used for the sterilization through the control of temperature and pressure, when it is required they assume values higher than the environment ones. The presence of multiple outputs and control variables, interacting with each other, complicates the process making its modeling not trivial. Moreover, the use of these machines in different fields of applications leads to a large variety of autoclaves available on the market with different size, toolkits, and components, requiring a variety of different models. However, even if each machine requires a specific physical model, starting from a common basic model representing the main processes occurring in an autoclave, the new model can be obtained adapting the basic one to the specific case. In this chapter an industrial air-steam autoclave is con-

sidered, where the sterilization is performed using a mix of air and steam: in order to balance the pressure inside the machine air injection is used, while to reach the sterilization temperature target through steam injection is applied. This machine is called FHA (Fedegari Horizontal Autoclave) and is produced by Fedegari Group (Albuzzano, Pavia), a company specialized in the production of autoclaves [77], that provided the data.
Previously, few works in the literature faced this modeling problem: in each case, the proposed model was specific of the analyzed process and could not be easily adapted to others. For example, in [78] an autoclave for curing of composites, and in [79] one for chemical leaching process were proposed; the heat transfer between the autoclave and the product was analyzed in [80], [81]. A purely physical approach to a similar autoclave used for sterilization was proposed in [82], showing a not optimal results for this kind of processes. Despite the affinities with some of these works, where temperature and pressure inside the autoclaves are modeled, the differences in the model characteristics and the considered processes require a new model for each particular application.
In the following, two models of an industrial air-steam sterilizer are proposed. The first one is a physical model, to be used for simulation, obtained extending a previous model of a lab equipment sterilizer proposed in [83], considering the new components of the industrial machine. The second model exploits a LSTM network to obtain a model for control applications, investigating also the stability properties of the system.
The physical model is the result of an hybrid approach, similar to the one proposed in [83], that involves a physical structure with data-driven adaptation of the parameters. In this way, the parameters take into account the model uncertainties derived by the physical modeling, while the main structure is based on the physical laws that affect the temperature and pressure behaviors. This model allows to simulate the effect of possible design changes before applying them. The second solution is a black-box approach, where a LSTM is used to identify the system. This approach requires a greater amount of data but provides in short time a good model of the machine on which the data were acquired. Moreover, the stability property of this network can be ensured applying a constrained training process, so the model can be used inside model-based controllers. These results have been published in [14].

3.2 Autoclave description

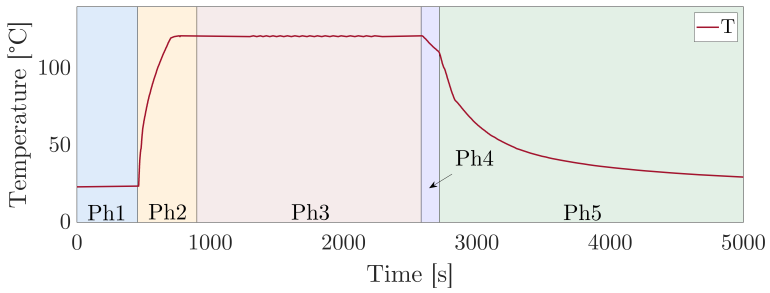
An autoclave is a pressurized chamber used in industrial applications to sterilize different products, exploiting pressurized steam, when high temperatures and pressures other than ambient air pressure are required. It is often used for sterilization, for example in pharmaceutical, food or chemicals in-

Figure 3.1: FHA autoclave.

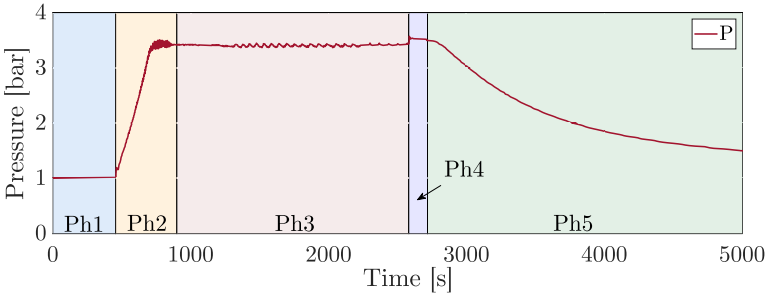
dustries, in hospitals and research centers, microbiology and analytical labs and so on.
The particular autoclave studied in the following is named FHA and is shown in Figure 3.1). The FHA is an industrial autoclave, composed by a single chamber for the sterilization, made by stainless steel as well as pneumatic valves and hydraulic components, with internal heat exchangers (plates) and a jacket surrounding the chamber, to preheat and cool down the chamber through steam and cold water. Inside the chamber a fan to distribute homogeneously air and steam is also present.

3.2.1 Air-steam cycle

The machine considered in this thesis can perform sterilization cycles using a mixture of air and steam, typical for the sterilization of liquid in sealed containers. Under these conditions, when the product is heated up, the liquid expands, exerting pressure. Consequently the pressure inside the chamber has to balance the liquid one, through the regulation of air and steam as described in [84]. The cycle is composed by four different phases, during which the temperature (Figure 3.2a) and pressure (Figure 3.2b) profiles undergo several variations. The cycle starts with the preparation of the autoclave (Ph1) during which the initial conditions are set and checked. Then, the chamber is heated (Ph2) through steam injection with the possibility to activate the auxiliary elements (plates and jacket) to speed up the process. This phase lasts until the target temperature and pressure are reached and maintained throughout the sterilization phase (Ph3). At the end of the sterilization, the cooling phase starts. Firstly, there is a pressurization using



(a) Temperature



(b) Pressure

Figure 3.2: Temperature and pressure profiles during the air-steam cycle.

only compressed air (Ph4) and then a controlled rate cooling (Ph5) during which the auxiliary elements can be activated to cool down the chamber.

3.3 Physical model

In this Section the physical model (PhM) is explained in detail. The sterilization process can be described through the definition of five different processes:

1. chamber filling and emptying;
2. chamber pressure regulation;
3. chamber heating and cooling;
4. plates heating and cooling;
5. jacket heating and cooling.

Processes 1-3 were already present in [83] and are integrated in this model adding the effects of plates and jacket that were not present in the lab sterilizer. The outputs of the system are the pressure and the temperature of the chamber. The temperature of the jacket is also measured and can be a variable of interest, so an additional output has been introduced in the model.

A discrete state is added to take into account different conditions of the chamber, that imply different physical laws for temperature and pressure.

3.3.1 Description

The dynamic of the system is described by 7 states, 14 inputs and 3 outputs, that are described in the following.
Among the seven states of the system, the first one is a discrete state, while the remaining are continuous ones. In details, the states of the system are:

- $x_1(k) = CC(k)$: chamber conditions;
- $x_2(t) = Q_a(t)$: air quantity in the chamber;
- $x_3(t) = Q_s(t)$: steam quantity in the chamber;
- $x_4(t) = P_c(t)$: chamber pressure;
- $x_5(t) = T_c(t)$: chamber temperature;
- $x_6(t) = T_p(t)$: plates temperature;
- $x_7(t) = T_j(t)$: jacket temperature.

The inputs of the system may be divided in three different groups, depending on the part of the system that they influence, i.e. the chamber, the plates and the jacket. The chamber inputs are:

- $u_1(t) = Q_{ca}(t)$: ingoing compressed air flow;
- $u_2(t) = Q_{st}(t)$: ingoing steam flow;
- $u_3(t) = a_f(t)$: fan activation state (0,1);
- $u_4(t) = d_1(t)$: contact surface of drain 1;
- $u_5(t) = d_2(t)$: contact surface of drain 2;
- $u_6(t) = d_3(t)$: drain 3 activation state.

The plates inputs are:

- $u_7(t) = Q_{st,p}(t)$: ingoing steam flow in the plates;
- $u_8(t) = Q_{H_2O,p}(t)$: ingoing water flow in the plates;
- $u_9(t) = d_4(t)$: drain 4 activation state.

The jacket inputs are:

- $u_{10}(t) = Q_{st,c}(t)$: ingoing steam flow in the jacket;
- $u_{11}(t) = Q_{H_2O,c}(t)$: ingoing water flow in the jacket;
- $u_{12}(t) = d_5(t)$: drain 5 activation state;
- $u_{13}(t) = d_6(t)$: drain 6 activation state
- $u_{14}(t) = d_7(t)$: drain 7 activation state.

Lastly, the outputs of the system are:

- $y_1(t) = P_c(t)$: chamber pressure;
- $y_2(t) = T_c(t)$: chamber temperature;
- $y_3(t) = T_j(t)$: jacket temperature.

In this machine, some valves can work both as on/off and as modulated valves. In particular, their working mode is regulated by the activation, a_v and a_w, of the analogue signals v and w, respectively. These signals modulate the opening of the valves: when the signals is active ($a_v = 1, a_w = 1$) the valves are modulated, otherwise they are in the on/off mode and their state is determined by the activation state a_n of the specific valve n. Inputs $u_1(t)$, $u_2(t)$, $u_4(t)$, $u_5(t)$ and $u_{11}(t)$ depend on the signal v, input $u_8(t)$ depends on the signal w, following a non linear relation that can be approximated with sigmoid functions, expressed in Equations (3.1) and (3.2):

$$u_n(t) = \left(\frac{L}{1 + Ae^{\,c\,(v-8)}} Q_n \; a_v + Q_n(1 - a_v) \right) \; a_n(t) \tag{3.1}$$

with $n = 1, 2, 4, 5, 11$ and

$$u_8(t) = \left(\frac{L}{1 + Be^{\,c\,(w-8)}} Q_8 \; a_w + Q_8(1 - a_w) \right) \; a_8(t) \tag{3.2}$$

where $L = 1, c = -3, A = 0.2$, $B = 200$, Q_n is the ingoing or outgoing quantity modulated by the valve n, a_n is its activation state and a_v, a_w are the activation states of the analogue signals. Both the signals work in a range of $[4 - 20]$ mA. When they control an input flow (steam, air or cooling water), the valve is completely open when the signal is high. Vice versa, when they control a drain, the valve is completely open when the signal is low. In this way it is possible to control two opposite effects with a unique signal. In the following, when a valve n is modulated, its input is indicated as u_n, while when it functions as on/off, its input is indicated as a_n, referring to the activation of the valve. If different valves are used to control the ingoing/outgoing flows, it is possible to adapt the model studying the new component and changing the expression of $u_n(t)$.

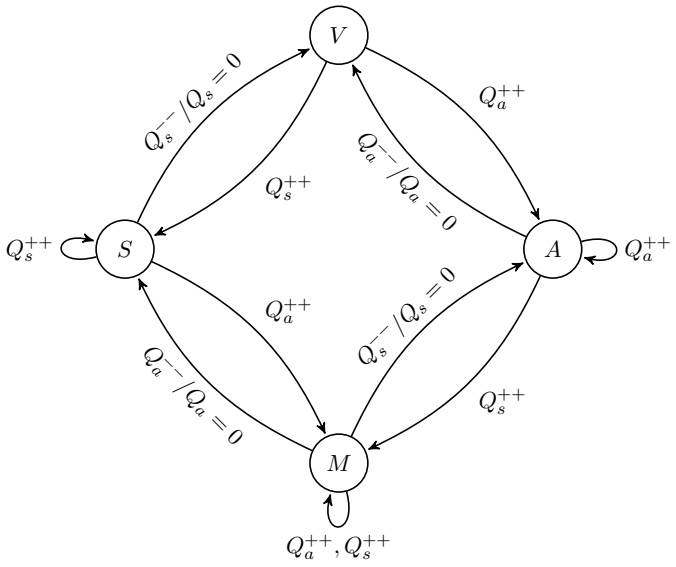


Figure 3.3: Finite state machine of the chamber conditions

3.3.2 Equations

Chamber conditions

The first state that needs to be explained is the discrete state $x_1(k)$ that describes the chamber conditions. Since temperature and pressure follow different physical laws based on the quantities of air, $x_2(t)$, and steam, $x_3(t)$, inside the chamber, the value of $x_1(k)$ determines the values of the other continuous states. For this reason, the finite state machine (Figure 3.3) proposed in [83] is still valid and it is used to determine the different chamber condition between:

- Vacuum (V): no gas in the chamber ($x_1(k) = 0$);
- Mixed (M): both air and steam in the chamber ($x_1(k) = 1$);
- Saturated steam (S): only steam in the chamber ($x_1(k) = 2$);
- Air (A): only air in the chamber ($x_1(k) = 3$).

So the state $x_1(k)$ can be expressed by the following equations:

$$x_1(k) = \begin{cases} 0 & \text{if} \quad x_2(t) = x_3(t) = 0 \\ 1 & \text{if} \quad x_2(t), x_3(t) > 0 \\ 2 & \text{if} \quad x_2(t) = 0, x_3(t) > 0 \\ 3 & \text{if} \quad x_2(t) > 0, x_3(t) = 0 \end{cases} \tag{3.3}$$

It can be observed that since the state $x_1(k)$ is function of the continuous states $x_2(t)$ and $x_3(t)$, the overall state system described in the following is in a non-minimal state space representation. However, it has been chosen to represent it in this way in order to better understand the working mechanism of the above explained finite state machine and how the other states depend on the values assumed by $x_1(k)$ during the different phases of the process. Lastly, since in the air-steam cycle performed by the FHA there is always air inside the chamber, it always holds that $x_2(t) > 0$ and so only the cases with $x_1(k) = 1$ and $x_1(k) = 3$ are considered in the following.

Gas quantities

The second and third continuous states represent the quantities of the gases present in the chamber:

$$\begin{aligned}\dot{x}_2(t) = {} & u_1(t) \\ & - k_{d_1} u_4(t)(x_4(t) - P_a) P_{ot} \cdot \left(1 - \frac{x_3(t)}{x_2(t) + x_3(t)} u_3(t)\right) q_a \\ & - k_{d_2} u_5(t)(x_4(t) - P_a) P_{ot} \cdot \left(1 - \frac{x_3(t)}{x_2(t) + x_3(t)} u_3(t)\right) q_a\end{aligned} \tag{3.4}$$

$$\begin{aligned}\dot{x}_3(t) = {} & u_2(t) \\ & - k_{d_1} u_4(t)(x_4(t) - P_a) \frac{x_3(t)}{x_2(t) + x_3(t)} P_{ot} \cdot \big(1 - q_a(1 - u_3(t))\big) q_s \\ & - k_{d_2} u_5(t)(x_4(t) - P_a) \frac{x_3(t)}{x_2(t) + x_3(t)} P_{ot} \cdot \big(1 - q_a(1 - u_3(t))\big) q_s \\ & - k_1(x_5(t) - T_a) q_s T_{ot}\end{aligned} \tag{3.5}$$

where P_{ot} is an auxiliary logic variable active if the pressure $x_4(t)$ is above the threshold P_a, similarly T_{ot} is active if the temperature $x_5(t)$ is above the threshold T_a, while q_a and q_s detect the presence of air ($x_2(t) > 0$) and steam ($x_3(t) > 0$), respectively.
The quantities $x_2(t)$ and $x_3(t)$ depend on the ingoing flow of compressed air, $u_1(t)$, or steam, $u_2(t)$, and on the outgoing flow through the chamber drains, $u_4(t)$ and $u_5(t)$. The outgoing flow is proportional to the difference between the pressure inside the chamber and the atmospheric one, P_a, if the pressure inside is greater than it. The machine is equipped with a fan, $u_3(t)$, in order to have an homogeneous distribution of gases inside the chamber. If the fan is active, the two gases will be expelled proportionally to their quantities in the chamber, otherwise the air will be expelled firstly because the drains are located on the bottom. A term to take into account the condensation effect of the steam is also added in Equation (3.5).

Chamber pressure

The fourth state describes the pressure evolution and depends on the chamber condition. In the Mixed case ($x_1(k) = 1$) the pressure equation is:

$$\begin{aligned}\dot{x}_4(t) = &+ k_{p,u_{1m}} u_1(t)\big(x_4(t) - P_a\big) \\ &+ k_{p_1} u_1(t)\big(x_5(t) - T_a\big) \\ &- k_{p,d_{2m}} u_5(t)\big(x_4(t) - P_a\big) \\ &- k_{p_m}\Big(x_4(t) - P_{st}\big(x_5(t)\big)\Big)\end{aligned} \tag{3.6}$$

It increases due to the injection of compressed air and decreases because of the opening of the drains. A condensation effect is present due to the injection of air with a lower temperature (atmospheric one, T_a) with respect to the temperature of the gas inside the chamber. The fourth term of Eq. (3.6) represents the one-to-one correspondence between temperature in Kelvin (T_c^k) and pressure (P_{st}) inside the chamber:

$$P_{st}\left(T_c^k(t)\right) = f\left(T_c^k(t)\right) = P_{wv}\left(T_c^k(t)\right) + P_{as}\left(T_c^k(t)\right) \tag{3.7}$$

where P_{wv} is given by water vapour pressure law [85]

$$P_{wv}\big(T_c^k(t)\big) = \frac{e^{73.649 - \frac{7258.2}{T_c^k(t)} + 4.1653\cdot 10^{-6}\cdot T_c^k(t)^2}}{e^{7.3037}\cdot T_c^k(t)} \tag{3.8}$$

and P_{as} is proportional to the temperature T_c^k following the law related to the overpressure inside sealed container with aqueous solution [84]. Note that a similar relation stands in the saturated steam case so that Equation (3.6) can be easily adapted to that new case. In the Air case ($x_1(k) = 3$) we obtain:

$$\begin{aligned}\dot{x}_4(t) = &+ k_{p,u_1} u_1(t)\big(x_4(t) - P_a\big) \\ &- k_{p,d_1} u_4(t)\big(x_4(t) - P_a\big) \\ &- k_{p,d_2} u_5(t)\big(x_4(t) - P_a\big) \\ &- k_{p,cool_{pl}} a_8(t)\big(x_4(t) - P_a\big) \\ &- k_{p,cool_j} a_{11}(t)\big(x_4(t) - P_a\big)\end{aligned} \tag{3.9}$$

The pressure of the chamber is increased by the injection of compressed air and is decreased by the activation of drain 1, $u_4(t)$, or drain 2, $u_5(t)$. There is a further decreasing when jacket and/or plates are filled with cooling fluid.

Chamber temperature

The fifth state represents the temperature inside the chamber, and as the pressure, it depends on the chamber conditions.

In the Mixed case ($x_1(k) = 1$) the temperature is:

$$\begin{aligned}\dot{x}_5(t) = &+ k_{c,u_{2m}}(T_{ot0})\, u_2(t)\, (T_{stmax} - x_5(t)) \\ &- k_{c,d_{2m}} u_5(t)(x_5(t) - T_a) \\ &- k_{c,d_3} a_6(t)(x_5(t) - T_a) \\ &+ k_{c,pl_m} a_7(t)\big(x_6(t) - x_5(t)\big)\end{aligned} \tag{3.10}$$

It increases due to the injection of steam until T_{stmax} is reached, that is the maximum temperature that can be reached through steam injection. It is decreased by the opening of drains 1 and 3. An heat exchange between chamber and plates increases or decreases the temperature, proportionally to the temperature difference of the two elements. Note that, the parameter $k_{c,u_{2m}}$ depends on the initial temperature of the chamber $x_5(0)$; in particular, T_{ot0} is an auxiliary logic variable active if the initial temperature is above the threshold $T_{th} = 30°$. In the Air case ($x_1(k) = 3$) we obtain:

$$\begin{aligned}\dot{x}_5(t) = &- k_{c,d_1} u_4(t)(x_5(t) - T_a) \\ &- k_{c,d_2} u_5(t)(x_5(t) - T_a) \\ &- k_{c,u_1} u_1(t)(x_5(t) - T_a) \\ &- k_{c,d_{1pl}}\big(a_4(t) + a_8(t)\big)(x_5(t) - T_a) \\ &- k_{c,d_{2pl}}\big(a_5(t) + a_8(t)\big)(x_5(t) - T_a) \\ &- k_{c,pl} a_8(t)(x_5(t) - T_p) \\ &- k_{c,j} a_{11}(t)(x_5(t) - T_a)\end{aligned} \tag{3.11}$$

The temperature is decreased by drain 1 or 2, the injection of compressed air and the cooling effect of the jacket. The cooling effect of the plates is also present and described by the following relationships:

$$k_{c,d_{1pl}} = \begin{cases} \bar{k}_{c,d_{1pl}} & \bar{t}_p < t < \bar{t}_p + \tau_1 \\ 0 & otherwise \end{cases}$$

$$k_{c,d_{2pl}} = \begin{cases} \bar{k}_{c,d_{2pl}} & \bar{t}_p < t < \bar{t}_p + \tau_1 \\ 0 & otherwise \end{cases}$$

$$k_{c,pl} = \begin{cases} 0 & \bar{t}_p < t < \bar{t}_p + \tau_1 \\ \bar{k}_{c,pl} & otherwise \end{cases}$$

At the beginning, it depends on $k_{c,d_{1pl}}$ or $k_{c,d_{2pl}}$, after τ_1 it depends on $k_{c,pl}$ and tends to the plates temperature, T_p, with $\bar{t}_p$ the time of the activation of the cooling fluid in the plates. This happens because after this time interval the cooling fluid of the plates has reached a thermal equilibrium with the chamber, so its cooling effect is slightly reduced.

Plates temperature

The sixth state is the plates temperature:

$$\begin{aligned}\dot{x}_6(t) = &+ k_{pl,u_7} a_7(t)\,(T_{stmax} - x_6(t)) \\ &- k_{pl,d_4} a_9(t)(x_6(t) - T_a) \\ &- k_{pl,c} a_7(t)\,(x_6(t) - x_5(t))\end{aligned} \tag{3.12}$$

It is increased by the injection of steam in the plates through $u_7(t)$, until the sterilization temperature is reached, and decreased through the drain 4, $u_9(t)$, proportionally to the atmospheric temperature. The temperature of the plates also depends on the heat exchange with the chamber, proportionally to the temperature difference of the two elements.

Jacket temperature

The seventh state represents the jacket temperature.

$$\begin{aligned}\dot{x}_7(t) = &+ k_{j,c}\big(x_5(t) - x_7(t)\big) \\ &+ k_{j,u_2} u_2(t)\big(T_{steam} - x_7(t)\big) \\ &+ k_{j,u_{10}} a_{10}(t-\tau_2)\big(T_{stmax} - x_7(t)\big) \\ &- k_{j,d_5} a_{12}(t)(x_7(t) - T_a) \\ &- k_{j,d_6} a_{13}(t)(x_7(t) - T_a) \\ &- k_{j,d_7} a_{14}(t)(x_7(t) - T_a) \\ &- k_{j,pl} a_8(t-\tau_1)(x_7(t) - T_a) \\ &- k_{j,cool} a_{11}(t-\tau_2)(x_7(t) - T_a) \\ &+ k_{j,fl} a_{11}(t)\big(T_{steam} - x_7(t)\big) \\ &- k_{j,disp} a_{12}(t)\big(x_7(t) - T_{steam}\big)\end{aligned} \tag{3.13}$$

Since the chamber is completely surrounded by the jacket, there is always an heat exchange between them. The temperature of the jacket increases when steam is injected inside the chamber through $u_2(t)$ and when steam is directly injected through $u_{10}(t)$, until the sterilization temperature is reached, considering a delay τ_2 due to the waiting time for the complete filling of the jacket. It decreases through the drains $(u_{12}(t), u_{13}(t), u_{14}(t))$ and when the cooling fluid is present in the plates, $u_8(t)$, and in the jacket, $u_{11}(t)$, considering the delay τ_1 due to the waiting time for the cooling of the plates. The parameter $k_{j,fl}$ takes into account the delayed effect of the injection of cooling fluid in the jacket when the jacket is used for cooling. On the contrary, $k_{j,disp}$ considers the delayed effect of the drain 5 when the jacket is used for heating. These relationships are described by the following equations:

$$k_{j,fl} = \begin{cases} \bar{k}_{j,fl} & \bar{t}_c + \epsilon < t < \bar{t}_c + \tau_2 \\ 0 & otherwise \end{cases}$$

$$k_{j,disp} = \begin{cases} \bar{k}_{j,disp} & \bar{t}_d < t < \bar{t}_d + \tau_3 \\ 0 & otherwise \end{cases}$$

with $\bar{t}_c$ the time of the activation of the cooling fluid in the jacket delayed by a constant ϵ and $\bar{t}_d$ the time of the activation of the drain 5. All the delays reported in this section have been experimentally estimated.

Output transformation

Finally, the output transformation is given by:

$$\begin{aligned} y_1(t) &= x_4(t) \\ y_2(t) &= x_5(t) \\ y_3(t) &= x_7(t) \end{aligned} \tag{3.14}$$

3.4 LSTM model

The current FHA system is equipped with simple controllers, like PID, that do not require a model. Considering in particular the modulated valves described in Section 3.3.1, their behaviour can be further improved applying a more complex control approach such as model-based solutions. In this case, models with a lower complexity with respect to the one used in simulation are required to perform predictions. A possible solution to this problem is constituted by a LSTM model.

3.4.1 Description

A single layer LSTM network with 23 inputs, 300 neurons and 2 outputs is considered in this thesis. The outputs are chamber temperature and pressure, the jacket temperature is not considered in this model since it is not a variable of interest for the controller.
The training has been performed using the Matlab environment, exploiting the DeepLearning Toolbox that provides a framework to design and train NNs. Moreover, thanks to the Parallel Computing Toolbox it is possible to use a GPU NVIDIA to fasten the computational time. In practice, a computer equipped with GPU NVIDIA GeForce GTX 1050 was used to train the LSTM network for 600 epochs, considering a mini-batch size of 2. The Adam optimizer is used with initial learning rate $\alpha = 0.001$ (with 0.8 drop factor after 70 epochs), decay rate $\beta_1 = 0.9$ and squared decay rate $\beta_2 = 0.99$.
The ISS property of the LSTM is enforced during the training of the network considering Theorem 2. In particular, Condition (2.18) is included as soft

constraints in the loss function J_{LSTM}, modified as follows:

$$J_{LSTM} = \frac{1}{2L}\sum_{i=1}^{L}(\hat{Y}(i) - Y(i))^2 + \mu(\rho_1 A_1 + \rho_2 A_2) \tag{3.15}$$

considering

$$\begin{cases} A_1 = 0 & \text{if } \psi_1 < 1 \\ A_1 = (1 + \epsilon - \psi_1)^2 & \text{if } \psi_1 \geq 1 \end{cases} \tag{3.16}$$

and

$$\begin{cases} A_2 = 0 & \text{if } \psi_2 < 1 \\ A_2 = (1 + \epsilon - \psi_2)^2 & \text{if } \psi_2 \geq 1 \end{cases} \tag{3.17}$$

having fixed

$$\begin{array}{rcl} \psi_1 & = & (1 + \sigma_g(|[W_o\, U_o\, b_o]|_\infty))\, \sigma_g(|[W_f\, U_f\, b_f]|_\infty) \\ \psi_2 & = & (1 + \sigma_g(|[W_o\, U_o\, b_o]|_\infty))\, \sigma_g(|[W_i\, U_i\, b_i]|_\infty)\, |U_c|_1 \end{array} \tag{3.18}$$

where L is the mini-batch size considered at each iteration, $\hat{Y}$ is the prediction, Y the real data, μ, ρ_1, ρ_2 and ϵ are the parameters to be tuned.
The training sequences are 2000 samples long, with initial state set to 0. The values of the hyperparameters are initially obtained through a trial and error procedure. The value of μ is set equal to 1000, in this way a violation of the constraints is heavily weighted. Note that without regularization, so with $\mu = 0$, it would be possible anyway to have a network that satisfies Condition (2.18), but not having any guarantee; instead, with the considered loss function this condition is forced.

3.5 Datasets description

The data used for the two models have been ad-hoc acquired by the company on the FHA. The high complexity of this machine and the optional use of the auxiliary components for the heating and cooling processes lead to several possible configurations of the sterilization cycle. In particular, they are discriminated by:

1. the initial temperature (high, low);
2. the auxiliary heating method (none, plates, jacket, both plates and jacket);
3. the cooling method (plates, both plates and jacket);
4. the drain (1 or 2) used during the cooling phase.

Run	Init. Temperature	Aux Heating	Aux Cooling	Drain
1	Low	None	Plates & Jacket	1
2	Low	None	Plates	1
3	Low	Plates	Plates & Jacket	1
4	Low	Plates	Plates	1
5	Low	Jacket	Plates	2
6	High	Plates	Plates & Jacket	1
7	High	Plates	Plates	1
8	High	Jacket	Plates	2
9	High	Jacket	Plates & Jacket	2
10	High	Plates & Jacket	Plates	1

Table 3.1: Datasets description.

To cover most of the possible combinations over the total 32 possible, 10 runs were collected on the FHA, as shown in Table 3.1. In particular, the data collection was focused on obtaining at least all the possible combinations of the heating and cooling methods, in accordance with the company possibilities. The cycles using jacket and plates both in heating and cooling presented some problems during data collection and have to be excluded from this analysis.
In order to assess the performances of the two models described in the previous sections, the 10 runs listed in Table 3.1 are equally split in two different datasets, each one containing 5 runs, one for training and one for testing. Considering C_n^k, the k-combination of a set with n elements, expressed as:

$$C_n^k = \frac{n!}{k!(n-k)!} \tag{3.19}$$

then, the combination of the 10 available runs, taken 5 at a time without repetition, is $C_5^{10} = 252$. To discriminate between the 252 possible combinations, in order to have balanced characteristics of the runs between the two datasets, the following criteria are formalized. In each dataset there should be:

- half runs with low initial temperature, half with high initial temperature;
- one without auxiliary heating;
- at least two runs with plates as auxiliary heating;
- at least one run with jacket as auxiliary heating;
- at least one run with plates and jacket as auxiliary cooling, with low initial temperature;

- at least one run with plates and jacket as auxiliary cooling, with high initial temperature;
- at least one run with plates as auxiliary cooling, with low initial temperature;
- at least one run with plates as auxiliary cooling, with high initial temperature;

Out of the 252 possible combinations, 8 satisfy the above criteria and the following one is chosen:

1. *Training dataset*: runs 2, 3, 5, 6, 8;
2. *Testing dataset*: runs 1, 4, 7, 9, 10.

3.6 Discussion

3.6.1 Physical model results

The optimization procedure proposed in [83], for the modeling of the lab sterilizer, required a collection of repeated runs for each configuration of the considered air-steam cycle, that are not currently available for the FHA. So the parameters tuning for this model is performed via a trial and error procedure and the complete parameters optimization presented in [83] is demanded to a future work when further data collection of repeated tests will be acquired. In particular, a preliminary case-study is introduced here to validate the contributions of auxiliary heating and cooling components added to the previous state-space model.
The tuning of the parameters is performed on the *Training dataset* and then the validation on the *Testing dataset*. The parameters specifically connected to the physical valves of the FHA have been set under the supervision of the technical staff of Fedegari company. The values of the parameters are reported in Table 3.2, together with the constants used in the physical model. The goodness of the model is computed for temperature and pressure in the chamber and for the temperature in the jacket. The plates temperature is excluded from this validation study since no direct measure is possible for this quantity. Two performance indexes are considered:

- Index of fitting (FIT): normalized index that indicates how much the prediction matches the real data. For a perfect prediction it is equal to 100% and it can also be negative. It is expressed as:

$$FIT = 100\left(1 - \frac{\sum_{i=1}^{S}\|\hat{y}_i - y_i\|}{\sum_{i=1}^{S}\|y_i - \overline{y}_i\|}\right) \tag{3.20}$$

	Parameter	Value
Air/st	k_{d_1}	0.01
	k_{d_2}	0.001
	k_1	0.1
Chamber pressure	$k_{p,u_{1m}}$	0.3
	k_{p_1}	0.01
	$k_{p,d_{2m}}$	$15 \cdot 10^{-4}$
	k_{p_m}	0.022
	k_{p,u_1}	0.05
	k_{p,d_1}	$5 \cdot 10^{-4}$
	k_{p,d_2}	$2 \cdot 10^{-4}$
	$k_{p,cool_{pl}}$	$8 \cdot 10^{-4}$
	$k_{p,cool_j}$	$2 \cdot 10^{-4}$
Chamber temperature	$k_{c,u_{2m}}$	0.55 if $T_{ot0} = 0$ 0.65 if $T_{ot0} = 1$
	$k_{c,d_{2m}}$	$1 \cdot 10^{-6}$
	k_{c,d_3}	$2 \cdot 10^{-4}$
	k_{c,pl_m}	$5 \cdot 10^{-4}$
	k_{c,d_1}	0.002
	k_{c,d_2}	0.001
	k_{c,u_1}	0.001
	$\bar{k}_{c,d_{1pl}}$	$15 \cdot 10^{-4}$
	$\bar{k}_{c,d_{2pl}}$	$25 \cdot 10^{-4}$
	$\bar{k}_{c,pl}$	0.001
	$k_{c,j}$	0.001

	Parameter	Value
Plates	k_{pl,u_7}	0.02
	k_{pl,d_4}	$3 \cdot 10^{-7}$
	$k_{pl,c}$	$35 \cdot 10^{-4}$
Jacket temperature	$k_{j,c}$	$48 \cdot 10^{-5}$
	k_{j,u_2}	0.024
	$k_{j,u_{10}}$	0.01
	k_{j,d_5}	$3 \cdot 10^{-5}$
	k_{j,d_6}	$5 \cdot 10^{-5}$
	k_{j,d_7}	$3 \cdot 10^{-5}$
	$k_{j,pl}$	$5 \cdot 10^{-5}$
	$k_{j,cool}$	0.005
	$\bar{k}_{j,fl}$	0.05
	$\bar{k}_{j,disp}$	0.02
Constants	T_p	45 [$^\circ C$]
	T_a	20 [$^\circ C$]
	P_a	1.01325 [bar]
	T_{stmax}	130 [$^\circ C$]
	T_{steam}	100 [$^\circ C$]
	τ_1	180 [s]
	τ_2	120 [s]
	τ_3	180 [s]
	ϵ	60 [s]

Table 3.2: Parameters of the physical model of the FHA.

- Pearson correlation coefficient (ρ): measures the linear correlation between two variables and has a value between -1 (total negative correlation) and +1 (total positive correlation).

$$\rho = \frac{\sum_{i=1}^{S}(y_i - \overline{y})(\hat{y}_i - \overline{\hat{y}})}{\|y_i - \overline{y}\| \cdot \|\hat{y}_i - \overline{\hat{y}}\|} \tag{3.21}$$

where y is the vector of the real data, $\hat{y}$ of the predicted ones, containing S elements each, and $\overline{y}$, $\overline{\hat{y}}$ are their respective mean values. The results of the index calculation are shown in Table 3.3. The mean of the performance indexes is reported in the last row of the table; these mean values are referred as $\overline{FIT}$ and $\overline{\rho}$.
The overall performance is satisfactory, with $\overline{FIT} = 94.26\%$ and $\overline{\rho} = 0.998$ for the temperature and $\overline{FIT} = 91.55\%$ and $\overline{\rho} = 0.998$ for the pressure.

	Temperature		Pressure		Jacket	
Run	FIT	ρ	FIT	ρ	FIT	ρ
1	95.08	0.999	91.92	0.997	86.54	0.992
4	93.51	0.998	90.60	0.998	73.52	0.897
7	93.00	0.998	89.25	0.999	98.23	0.997
9	95.34	0.999	93.05	0.998	96.42	0.998
10	94.34	0.998	92.93	0.998	97.53	0.996
Av	94.26	0.998	91.55	0.998	90.45	0.976

Table 3.3: Physical model performances.

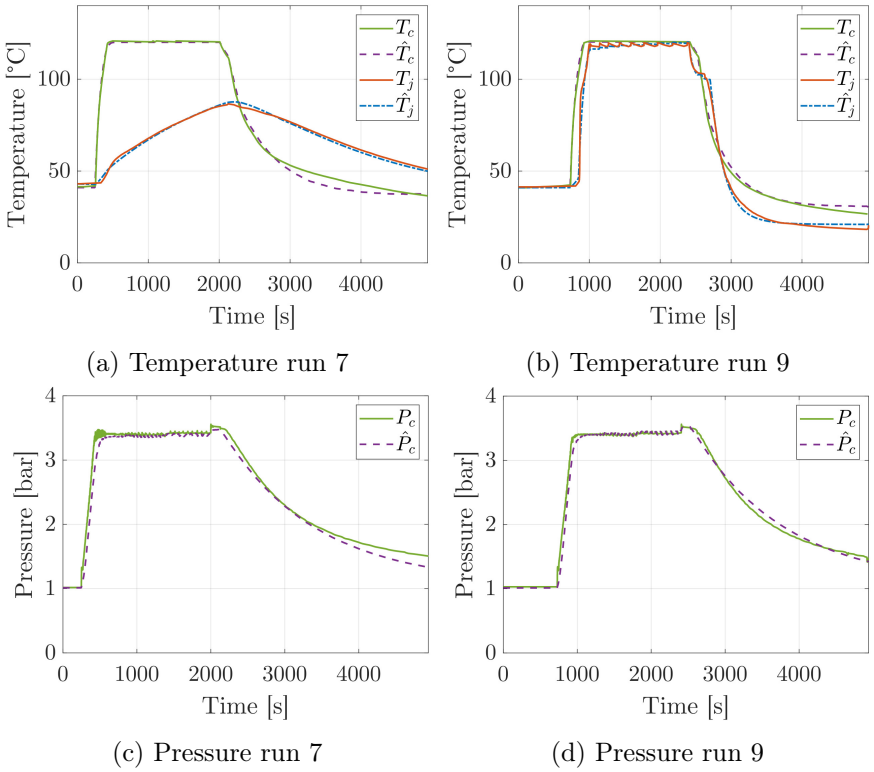
	Temperature		Pressure	
Run	FIT	ρ	FIT	ρ
1	81.90	0.984	73.88	0.966
4	87.44	0.992	82.72	0.989
7	84.42	0.988	78.34	0.985
9	34.38	0.836	16.72	0.830
10	51.88	0.898	54.50	0.936

Table 3.4: LSTM model performances.

Contrarily to the results obtained in [83], no particular difference has been noticed with respect to the initial temperature, only the parameter $k_{c,u_{2m}}$ required to be differentiated. The validation results of the temperature in the jacket, reported in the same table, obtained great results with $\overline{FIT} = 90.45\%$ and $\overline{\rho} = 0.976$.
A graphical example of two different runs is reported: in Figure 3.4a and 3.4c temperature and pressure of run 7 are reported, respectively, in Figure 3.4b and 3.4d the same signals for run 9. T_c and T_j are the real temperatures in the chamber and in the jacket, P_c (bottom) is the real pressure in the chamber, while $\hat{T}_c$, $\hat{T}_j$ and $\hat{P}_c$ are the correspondent predictions. In particular, it is interesting to notice how the jacket temperature is well represented both when it is used for heating and cooling (Figure 3.4b) or not (Figure 3.4a).

3.6.2 LSTM model results

The training of the LSTM network is performed on the *Training dataset* and then evaluated on the *Testing dataset*, as done before for the physical model, in order to have also comparable results. The results are shown in Table 3.4, considering FIT and ρ also in this case. Looking at the results, it



(a) Temperature run 7

(b) Temperature run 9

(c) Pressure run 7

(d) Pressure run 9

Figure 3.4: Temperature and pressure profiles with the physical model.

can be observed that good performances are obtained for the first three runs while the results are not satisfactory for runs 9 and 10. From an analysis of the characteristics reported in Table 3.1, it can be noticed that these runs have some singular characteristics: run 9 uses both plates and jacket for auxiliary cooling with drain 2 and run 10 uses both plates and jacket for auxiliary heating. These particular combinations are present only in these two runs, so the network is not able to recognize them during testing. For the moment, in first analysis, these two runs are not considered. Hence, looking at the performances of runs 1, 4 and 7, reported in Table 3.4, the overall performance is satisfactory with $\overline{FIT} = 84.59\%$ and $\overline{\rho} = 0.99$ for the temperature and $\overline{FIT} = 78.31\%$ and $\overline{\rho} = 0.98$ for the pressure. If the controller would have to manage cycles with features similar to the ones that generated runs 9 and 10, an update of the model will be required.

3.7 Models comparison

Two models for an industrial autoclave have been developed following two different approaches: a white-box model based on the physical knowledge of

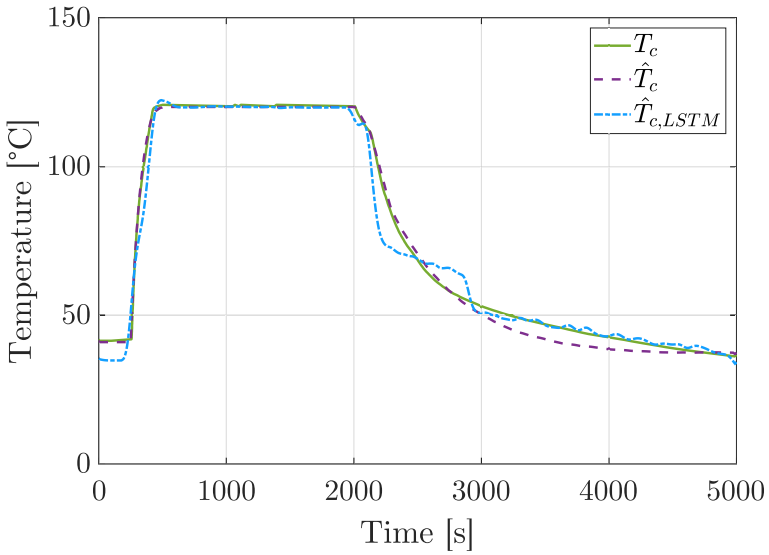


Figure 3.5: Temperature profile comparing the physical model and the LSTM one.

the process for simulation and a black-box one based on a LSTM network to obtain predictions necessary for the application of advanced control systems. Both the models obtain satisfactory results, considering their own specific target, highlighting at the same time their limitations.
In Figure 3.5 a comparison between the two models is shown, in particular plotting the temperature profile of run 7. As already seen from the performances indexes the physical model shows a better accuracy, anyway both the models have satisfying performances especially during the sterilization plateau that is the most important phase, where pressure and temperature must be kept constant.
The physical model gives excellent results but, requiring an accurate study of the physical transformations involved in the process, it is not trivial and requires a lot of time. Even if several physical aspects of the process have been neglected in the modeling procedure, the main behavior of temperatures and pressure are satisfactorily represented as proved by high FIT values. With additional data coming from different machines, a portability study of this model could be carried out. In principle, the model is generic enough to be easily adapted modifying the ingoing/outgoing flow and the time delays typical of the considered machine. Of course, the definition of the physical equation and the identification of the parameters are not trivial tasks but the quality reached is high enough to use this model as a reliable simulator. It is a powerful tool to be used for simulation but does not fit well control requirements.
On the other hand, considering the mathematical complexity of the physical

model and the high number of involved parameters that need to be tuned, the LSTM model is a good alternative to reduce modeling time and effort. This model does not require a deep knowledge of the physical processes and of the functioning of the machine, provides satisfactory predictions and also ensures ISS properties. A limitation of this approach is that the network is not able to deduce an unseen behavior such as the physical model, so all the possible combination have to be present in the *Training dataset*. In order to improve the network performances more data are needed, in this way it is possible to cover all the possible machine configurations and to enforce the generalization capability of the model. Unfortunately, the possibility of a further data collection is currently limited. Moreover, since this is a black-box approach, the effect of each component of the machine is not visible and, depending on the specific application that requires the usage of a model, this can be a limitation.

In summary, the physical model can be used to perform simulation and analyze the effect of components change even before the machine building, requiring time and effort to acquire the specific knowledge of the physical behavior of the machine. The LSTM model can be used for control applications, however since it does not reflect the physical composition of the machine, if a component is changed, a new data acquisition and a new training phase are required.

Chapter 4

Modeling of an industrial coffee roaster

In this chapter an industrial plant for coffee roasting is taken into consideration comparing the LSTM model with a scalable physical model. The modeling of this process is a well-known case study in the scientific literature and several physical models have been proposed with very satisfying results. It will be discussed if it is convenient to use a NN model under these conditions.

4.1 Motivation and state of the art

The roasting of the coffee beans is the industrial process mainly responsible for forming the flavor and aroma of a cup of coffee [86]. This process is suitable to be object of different research projects, so that the growing interest in it is not surprising: energy consumption [87], roasting temperature control [88], taste prediction [89] and even smart design with connected devices [90] are some examples of the current research applications. Several studies were conducted to model the behavior of an industrial roasting chamber since all these applications require a model for their design and testing. A solid physical model of the whole roasting plant was provided in [91]. Some physical parameters of this model, like specific heats or transfer coefficients, are closely related to the particular coffee bean quality (like Robusto or Arabica) and the particular size of the plant (e.g. 120 kg, 360 kg or 600 kg) used to identify the model. Further studies investigated with a greater detail the coffee bean reaction in the chamber providing a better description of the coffee beans parameters [92], [93]. Nevertheless, the strong connection between the coffee beans quality, the size of the plant and the whole model still stands. Such connections strongly limited the application of the model to different plants, requiring de facto a new parameter identification phase each time. This is particularly relevant in food industry processes where data collection

requires the consumption of a considerable amount of resources (e.g. several kg of coffee). In order to avoid food waste, several strategies have been investigated, considering also advanced process control techniques [94].
In this chapter, the modeling of an industrial coffee roaster is described: firstly, a scalable model, based of on the physics of the system is proposed. This model has the peculiarity of being used on plants of different size: starting from the model proposed in [91] and exploiting some considerations from [95], a first group of parameters is defined as function of the chamber geometry, while the others are identified through non linear identification. In this way the model identified with data collected on one plant can be used on plants of different sizes simply scaling the first group of parameters, without requiring a new identification phase, saving time and resources. The methodology is validated identifying the model with data collected on a 120 kg plant and simulating the behavior on a 360 kg one, obtaining satisfactory results. These results have been published in [15].
The second contribution consists in a LSTM network, trained considering the same inputs and output of the scalable model. The usage of machine learning techniques is not a novelty in the food industry, even if they are often used for classification purposes to improve the process performances. In the coffee industry in particular, as said before, the quality of the beverage mainly depends on the temperature reached during the roasting process, that in practice, can be associated to the coffee beans color during roasting. NNs can be successfully employed for classification of the coffee roasting degree [96, 97], or before the roasting to assess the quality of the green coffee beans [98]. Computer vision can also be employed to better understand some difficult process dynamics, like in [99], where two NN models are used to predict brightness and bean surface areas during coffee roasting, helping to understand heat and mass transfers, that are the critical elements in physical models. A usage of NNs for modeling purposes in this field is presented in [100] where an hybrid model is built, considering regression trees and a FFNN. In that work the physical model proposed in the following could not be used since the drum rotation speed can not be taken into account and it is the variable of interest for the desired controller. This is a perfect example of the advantages brought by NN models when well-known white-box models are present in literature but are not suitable for the desired application.
The two models have been developed in this thesis based on data provided by Brambati SpA (Codevilla, Pavia), a company specialized in plants designing, building and installation for food, coffee and plastic industries [101].

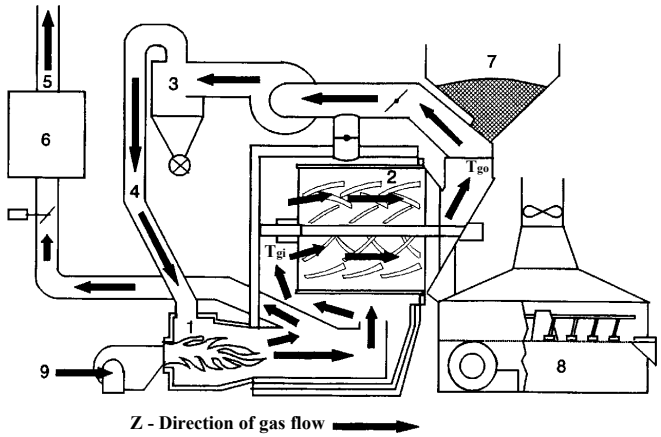


Figure 4.1: Rotating-drum roaster with solid wall: 1 furnace, 2 roaster drum, 3 cyclone, 4 gas recycle line, 5 gas discharge stack, 6 catalytic afterburner, 7 green bean bin, 8 cooler, 9 fresh air. Figure from [91].

4.2 Roasting process description

The roasting of green coffee beans is a complex process that involves several chemical reactions fundamental to determinate the coffee color, favor and aroma. In particular, these characteristics are determined by the temperature profile of the coffee beans during the roasting.
The roasting process is composed by three major phases: drying, roasting and cooling. During these phases the coffee bean is subjected to heat and mass transfers. The heat transfer occurs both by convection and conduction, and increases the bean temperature with consequent physical and chemical changes, such as a mass transfer due to the evaporation of water inside the bean and exothermic reactions.
Several roaster architectures are available on the market, this thesis considers a batch roaster: a plant that treats only a fixed amount of coffee, called batch, throughout a single operating cycle, shown in Figure 4.1. Each cycle is characterized by an initial phase where no coffee is loaded in the machine and both the air stream and the drum walls are heated up to the desired temperature. The process starts with a batch of green coffee beans at the environmental temperature loaded to the roaster drum through a conical funnel (7 in Fig. 4.1). Then, the drying phase starts when the air flow (9 in Fig. 4.1) heated by the furnace (1 in Fig. 4.1) is aspirated in the drum chamber (2 in Fig. 4.1) via a fan. The drum rotates at a uniform speed to ensure a uniform effect and to avoid the beans to adhere to the drum walls. The chamber is equipped with spiral blades in the internal surface to mix the beans in the axial direction. During this phase, the hot air flow dries the beans, then the beans are heated up until exothermic reactions near the

end of roasting cause a rapid increase in the bean temperature rise (roasting phase). The gases leave the chamber through a cyclone (3 in Fig. 4.1) that removes the chaff released by the beans during the roasting process. These gases can be either collected in a stack after be passed in a afterburner (6 in Fig. 4.1) to be discharged (5 in Fig. 4.1) or in part sent back to the roaster furnace (4 in Fig. 4.1). Once the end-of-roast temperature is reached, the gas supply is turned off and the roasting process is stopped by spraying cool water on the beans to cool them thanks to the evaporation of the water (8 in Fig. 4.1). The cooling phase proceeds in the cooling tank, where the beans are transferred to be stirred and further cooled through cold air input. Finally, the beans are unloaded and the system is prepared for the next roasting cycle. It is worth to be noted that the machine warming up phase particularly influences the roasting of the first batch [93].

4.3 Physical model

In recent years, several models have been proposed to investigate the roasting process of the green coffee beans. In the following, the one proposed by [91] is considered and extended using considerations published in [95]. The model is adapted for a batch roaster and some of the model parameters are defined on the base of the chamber geometry in order to create a new scalable model. Starting from the machine where the data were collected on, this model will be able to describe the behavior of new different unseen machines.

4.3.1 Equations

Hot gas heat transfer

During the roasting process hot gas is introduced in the roasting chamber. Considering a uniform flow in a single direction and a convective heat transfer between gas and beans, a temperature balance in the Z-direction (see Figure 4.1) can be expressed as:

$$-G_g(t)c_g(t)\frac{dT_g(t)}{dZ} = h_e(t)\frac{dA_{gb}}{dZ}\big(T_g(t) - T_b(t)\big) \tag{4.1}$$

where $G_g(t)$ is the gas mass-flow rate, $c_g(t)$ is the specific heat capacity of drying air, $h_e(t)$ is the gas to beans heat transfer coefficient, A_{gb} is the gas to beans heat transfer area, $T_g(t)$ and $T_b(t)$ are known gas and beans temperatures.
In particular, $c_g(t)$ is defined in [102] considering the thermophysical properties of drying air obtained from [103] as:

$$c_g(t) = \sum_{i=0}^{6} \alpha_i \big(T_{gi}(t) + 273.15\big)^i \tag{4.2}$$

where $\alpha_0 = 1.0839 \cdot 10^3$, $\alpha_1 = -7.2075 \cdot 10^{-1}$, $\alpha_2 = +2.1034 \cdot 10^{-3}$, $\alpha_3 = -2.3267 \cdot 10^{-6}$, $\alpha_4 = 1.3621 \cdot 10^{-9}$, $\alpha_5 = -4.1550 \cdot 10^{-13}$, $\alpha_6 = 5.3091 \cdot 10^{-17}$. In [91], h_e is considered a fixed parameter, on the contrary in this work it is time variant and depends on the moisture quantity $X(t)$, as defined in [95]:

$$h_e(t) = 0.49 - 0.443exp^{-0.206X(t)} \tag{4.3}$$

Integrating Equation (4.1) between the gas inlet and outlet temperatures, $T_{gi}(t)$ and $T_{go}(t)$, and rearranging, Equation (4.4) is obtained:

$$T_{gi}(t) - T_{go}(t) = \big(T_{gi}(t) - T_b(t)\big)\left(1 - exp^{-\frac{h_e(t)A_{gb}}{G_g(t)c_g(t)}}\right) \tag{4.4}$$

where the effects due to the heat transfer from gas to the metal part of the chamber are not considered. So an average value of the metal temperature, $T_m(t)$, is introduced and Equation (4.4) is refined taking this heat transfer into account as follows:

$$T_{gi}(t) - T_{go}(t) = \left(T_{gi}(t) - \frac{T_b(t) + F(t)T_m(t)}{1 + F(t)}\right)\left(1 - exp^{-\frac{h_e(t)A_{gb}}{G_g(t)c_g(t)}}\right) \tag{4.5}$$

The first contribution of this work is the definition of the new parameter $F(t)$, that is the ratio between the gas-metal and gas-beans thermal resistances:

$$F(t) = \frac{h_{gm}A_{gm}}{h_e(t)A_{gb}} \tag{4.6}$$

It depends on the gas to metal and gas to beans heat transfer coefficients, h_{gm} and $h_e(t)$, and the respective contact areas, A_{gm} and A_{gb}. In [91] the term $F(t)$ is negligible since for the mentioned roasters (rotating-bowl, scoop-wheel, spouted-bed, swirling bed roasters) this term is small. On the contrary in this application the term $F(t)$ is significant and so it has to be considered. Moreover, it is one of the parameters related to the chamber geometry so it is an important term in order to make the model scalable. Of course it requires the knowledge of both chamber and coffee beans dimensions in order to reach our purpose.
The contact area between gas and metal, A_{gm}, is then defined as the sum of the inner surface of the chamber and the total surface of the flaps inside it:

$$A_{gm} = \pi D_{ch}(L_{ch} + (H_{\text{flap}}L_{ch})/S_{\text{flap}} + D_{ch}/2) \tag{4.7}$$

where D_{ch} and L_{ch} are the diameter and the length of the chamber respectively, S_{flap} and H_{flap} are the step and the height of the flap.
The gas to beans heat transfer area, A_{gb}, depends on the dimensions of the beans, assumed having an average dimension determined experimentally in [102]. The total surface area of the beans is assumed to be $A_b =$

$(M_b/m_b)\pi D_b^2$, where M_b is the total weight of the beans loaded in the chamber, m_b is the weight of a single bean and D_b is the bean diameter. Since the model considers a rotating drum, it is necessary to define new parameters to determine the contact area between beans-metal and beans-gas. At each time instant a portion of the beans is in contact with the metal on the bottom of the drum, while the remaining part is in contact with the gas, pushed by the rotatory movement of the drum. So, calling P_{bm} the percentage of contact area of a single bean to the metal, the contact area between metal and beans A_{bm} is defined as $A_{bm} = A_b P_{bm}$ and consequently the contact area between gas and beans as $A_{gb} = A_b(1 - P_{bm})$.

Bean temperature

According to [91] the bean temperature variation is given by the following energy balance:

$$\dot{T}_b(t) = \frac{Q_{gb}(t) - Q_{gm}(t) + Q_{bm}(t) + M_{bd}(t)\big(Q_r(t) + \lambda \dot{X}(t)\big)}{M_{bd}(t)\big(1 + X(t)\big)c_b(t)} \tag{4.8}$$

Briefly, the heat is mainly transferred from the gas to the beans by convection, $Q_{gb}(t)$, while a small part is transferred from the gas to the metal of the chamber $Q_{gm}(t)$, which in turn transfers heat to the beans by conduction, Q_{bm}(t). The final term of (4.8) represents the heat produced due to exothermic reactions inside the beans: part of energy is lost, representing the latent heat of vaporization of the moisture inside the bean.
In the following, each element in Equation (4.8) is further described. The heat transfer rate between gas and beans is defined as:

$$Q_{gb}(t) = G_g(t)c_g(t)\big(T_{gi}(t) - T_{go}(t)\big) \tag{4.9}$$

the heat transfer rate between gas and metal is:

$$Q_{gm}(t) = \frac{F(t)\Big(h_e(t)A_{gb}\big(T_b(t) - T_m(t)\big) + Q_{gb}(t)\Big)}{1 + F(t)} \tag{4.10}$$

and the heat transfer rate between metal and beans is:

$$Q_{bm}(t) = h_{bm}A_{bm}\big(T_m(t) - T_b(t)\big) \tag{4.11}$$

where h_{bm} is the metal to beans heat transfer coefficient. Moreover, $M_{bd}(t)$ is the mass of dry beans in the chamber, $Q_r(t)$ is the exothermic heat production, λ is the latent heat of vaporization of beans moisture and

$$c_b(t) = \frac{c_s(t) + c_w X}{1 + X(t)} \tag{4.12}$$

is the specific heat capacity of coffee beans, as expressed in [91], where $c_s(t) = 1.099 + 0.007T_b(t)$ is the partial heat capacity of bean solids, c_w is the partial heat capacity of water and $X(t)$ is the beans moisture content.

Metal temperature

The metal temperature variation is defined as [91]:

$$\dot{T}_m(t) = \frac{Q_{gm}(t) - Q_{bm}(t) + Q_e(t)}{M_m c_m} \tag{4.13}$$

Basically, $T_m(t)$ increases thanks to the heat transfer from the gas while it decreases transferring heat to the beans. The heat transfer from sources external to the chamber, $Q_e(t)$, in this case is negligible since the model assumes that there is no leak in the roasting chamber. M_m and c_m are the mass and specific heat capacity of the metal, respectively.

Moisture loss

A semi-empirical relation between $X(t)$ and $T_b(t)$ is defined to model water evaporation during the roasting process, through an Arrhenius-type equation [91], where k_1 and k_2 are semi-empirical parameters:

$$\dot{X}(t) = -\frac{k_1}{D_b^2} exp^{-\frac{k_2}{T_b(t)+273.15}} \tag{4.14}$$

Exothermic roasting reactions

After the evaporation, heat is generated by exothermic reactions as reported in [104]. This effect is modelled as follows [91]:

$$Q_r(t) = A\frac{H_{et} - H_e(t)}{H_{et}} exp^{-\frac{H_a}{R(T_b(t)+273.15)}} \tag{4.15}$$

where H_{et} is the total reaction heat, $H_e(t)$ is the reaction heat produced thus far, H_a is the reaction activation energy and R is the gas costant. Reactants are consumed during the process and the concentration of the remaining ones is proportional to $\bar{H}(t) = \big(H_{et} - H_e(t)\big)/H_{et}$. The rate of the reactions is proportional to $\bar{H}(t)$ and to the coefficient of the Arrhenius equation, called A.

Final model equations

Starting from these considerations, the model dynamic is represented by four states, two inputs and one output, as listed below:

- $x_1(t) = T_b(t)$: temperature of the coffee bean inside the roasting chamber in Celsius;
- $x_2(t) = T_m(t)$: temperature of the metal chamber;
- $x_3(t) = X(t)$: moisture content of the coffee bean;

- $x_4(t) = H_e(t)$: amount of heat produced per kilogram of dry coffee thus far;
- $u_1(t) = G_g(t)$: mass flow rate of the gas at the inlet of the roasting chamber;
- $u_2(t) = T_{gi}(t)$: temperature of the gas at the inlet of the roasting chamber;
- $y(t) = T_b(t)$: temperature of the coffee bean inside the roasting chamber in Celsius;

The differential equations representing the model are:

$$\dot{x}_1(t) = \frac{Q_{gb}(t) - Q_{gm}(t) + Q_{bm}(t) + M_{bd}(t)\big(\dot{x}_4(t) + \lambda\dot{x}_3(t)\big)}{M_{bd}(t)\big(1 + x_3(t)\big)c_b(t)} \tag{4.16a}$$

$$\dot{x}_2(t) = \frac{Q_{gm}(t) - Q_{bm}(t) + Q_e(t)}{M_m c_m} \tag{4.16b}$$

$$\dot{x}_3(t) = \frac{k_1}{D_b^2} exp^{-\frac{k_2}{x_1(t)+273.15}} \tag{4.16c}$$

$$\dot{x}_4(t) = A\frac{H_{et} - x_4(t)}{H_{et}} exp^{-\frac{H_a}{R(x_1(t)+273.15)}} \tag{4.16d}$$

$$y(t) = x_1(t) \tag{4.16e}$$

It is important to notice that $M_{bd}(t)$ has been defined is this work by $M_b/\big(1 + x_3(t)\big)$, where M_b is the weight of the green beans coffee batch, so that also this parameter contributes to the scalability of the model.
Lastly, in Table 4.1 a comprehensive list of the parameters used in the physical model is reported for clarity.

4.3.2 Experimental setup

The standard equipment of an industrial roasting plant can provide only one of the signals described by the proposed model: the inlet gas temperature $T_{gi}(t) \equiv u_2(t)$. In order to collect the data required for the model identification, the inlet gas mass flow rate $G_g(t) \equiv u_1(t)$ has to be measured.
The bean temperature is the main measure of the whole process, so every plant is equipped with a temperature sensor that tries to measure the bean temperature. Of course this should be modeled to consider delays due to the sensor. So a well known sensor model proposed in [91] is included in order to allow an input-output identification.

Flow sensor

The measure of $G_g(t)$ was originally not available so that a Pitot tube and a thermocouple were placed in the center of the inlet pipe of the roasting

Acronym	Parameter
A	Arrhenius equation pre-factor
A_{gb}	gas to beans heat transfer area
A_{gm}	gas to metal heat transfer area
A_{bm}	metal to beans heat transfer area
c_b	specific heat capacity of coffee beans
c_g	specific heat capacity of drying air
c_m	specific heat capacity of the metal
D_b	bean diameter
D_{ch}	chamber diameter
G_g	gas mass-flow rate
h_e	gas to beans heat transfer coefficient
h_{gm}	gas to metal heat transfer coefficient
h_{bm}	metal to beans heat transfer coefficient
H_a	activation energy
H_e	reaction heat produced thus far
H_{et}	total reaction heat
H_{flap}	flap height
k_1, k_2	Schwartzberg's semi-empirical parameters
K_t	bean temperature sensor time constant
L_{ch}	chamber length
m_b	bean mass
M_b	green beans coffee batch mass
M_{bd}	dry beans coffee batch mass
M_m	metal mass
P_{bm}	percentage of bean metal contact area
Q_e	external sources heat transfer rate
Q_{gb}	gas to beans heat transfer rate
Q_{gm}	gas to metal heat transfer rate
Q_{bm}	metal to beans heat transfer rate
Q_r	exothermic heat production
R	gas constant
S_{flap}	flap step
T_b	beans temperature
T_g	gas temperature
T_{gi}	gas inlet temperature
T_{go}	gas outlet temperature
X	beans moisture content
λ	latent heat of vaporization of beans moisture

Table 4.1: List of the parameters of the physical model.

chamber to obtain the needed measure. Further detail of the placement of the sensor can be found in [105] and [106]. As described in [107], the required measure is given by:

$$G_g(t) = S\sqrt{2\rho_0 \Delta p \frac{273.15}{T(t) + 273.15}} \tag{4.17}$$

where S is the section of the pipe, ρ_0 is the air density (assumed 1.275 kg/m^3), Δp is the pressure difference measured by the Pitot tube and $T(t)$ is the thermocouple measurement expressed in Celsius degrees.

Measured bean temperature

The bean temperature is usually measured through thermocouples. Since coffee beans are not good conductors, there is a difference between the effective bean temperature $T_b(t)$ and the measured one $T_a(t)$.In [91] this difference is modeled as:

$$\dot{T_a}(t) = K_t\big(T_b(t) - T_a(t)\big) \tag{4.18}$$

In the following, $T_a(t)$ will be considered the output of the physical model.

4.4 LSTM model

Considering the process complexity and the great amount of parameters that need to be tuned, a NN model can be an interesting approach for the modeling task of the roasting process.
The proposed NN model is trained using the same inputs and output of the physical model: $G_g(t)$ and $T_{gi}(t)$, the mass flow rate and the temperature of the gas at the inlet of the roasting chamber in input, and $T_a(t)$, the measured bean temperature obtained through thermocouples, in output. Both input and output data are scaled using mean normalization.
The implemented architecture consists in a single layer LSTM. Since the input sequences have different lengths (the length of each sequence, n, is listed in Table 4.2), the data needs to be preprocessed. Firstly, the data are padded, adding at the end of the vectors NaN values, to have vectors of uniform length. Then, a Keras masking layer is employed. This layer receives in input the input shape and a masking value: at each time step, the values of the input vector that are equal to the masking value are masked and skipped in all the subsequent layers. In this particular case, when the LSTM layer receives a mask, ignores the padded values. Moreover, since the masking layer is not able to mask NaN values, NaN were first converted to a value outside the data range, in this case 10.
The training is performed considering Adam optimizer to perform the back-propagation algorithm, considering the Mean Squared Error (MSE) as loss

Batch	Roaster Size	n	Batch	Roaster Size	n
Batch 1	120 kg	26	*Batch 6*	360 kg	28
Batch 2	120 kg	26	*Batch 7*	360 kg	28
Batch 3	120 kg	26			
Batch 4	120 kg	26			
Batch 5	120 kg	33			

Table 4.2: Datasets description

function, defined as:

$$MSE = \frac{1}{S}\sum_{i=1}^{S}(\hat{y}_i - y_i)^2 \tag{4.19}$$

where $\hat{y}_i$ is the predicted value, y_i is the real value and S is the number of samples. The LSTM is trained for 200 epochs on a computer Intel i9-10920X CPU with 3.50 GHz, equipped with graphics processing unit GPU NVIDIA GeForce RTX 2080Ti. The training has been written in Python 3.9, using TensorFlow [108] and Keras API [109].
The hyperparameters of the network, i.e. the number of neurons n_c and the learning rate α, have been optimized using KerasTuner [110] with the `RandomSearch` algorithm, minimizing the MSE loss calculated on the validation dataset. The chosen exploration ranges for the hyperparameters optimization are:

$$32 < n_c < 512 \quad \text{and} \quad 0.001 < \alpha < 0.01$$

The optimal values obtained from this optimization are then presented in the following in Section 4.6.2, together with the results obtained from the LSTM training. Moreover, a detailed description of TensorFlow, Keras and KerasTuner is reported in Appendix A.

4.5 Datasets description

A data collection is necessary to realize the two models: to validate the physical model and optimize its parameters, and to train and test the LSTM one.
Through the setup described in Section 4.3.2, two datasets are collected from two plants of different sizes. The first dataset is collected on a 120 kg roaster and composed of 5 batches; the second one is collected on a 360 kg roaster and composed of 2 batches. Notice that the term "Batch" refers in this case to the coffee roasting industry, where it represents the coffee quantity roasted in a cycle, considering that the modeled machine is a batch roaster, and not to the meaning assumed in the machine learning applications. In Table 4.2

a description of the datasets is presented: each batch is listed with the name that will be referred to in the following, the size of the roaster where it is collected and its length n.
Several signals are present in the datasets but not all of them are considered in the models or mentioned in this chapter so their description is neglected. The signals of interest are the inlet gas mass flow rate, $G_g(t)$, and the inlet gas temperature, $T_{gi}(t)$, used as input signals for both the models; the measured bean temperature obtained through thermocouples, $T_a(t)$, used as the output signal.
It is important to highlight that this data collection is ad hoc performed for research and modeling purposes, so to avoid coffee waste, only few datasets are available, especially those of the 360 kg machine. It can be a limitation of this work but some solutions are currently under study to overcome this limit.

4.6 Discussion

4.6.1 Physical model results

Parameters estimation

Most of the model parameters described in Section 4.3 are specific of the roasting plant and directly measurable on it or can be obtained from well known physical expressions. On the contrary, three parameters, h_{gm}, h_{bm} and P_{bm}, are not measurable and have to be identified from the data so in the following an automatic identification procedure to define the optimal values of these parameters, h^*_{gm}, h^*_{bm} and P^*_{bm}, is proposed.
The extended model (4.16a)-(4.16d), (4.18) is used to generate the bean temperature prediction $\hat{T}_a(t)$ in all the batches of the two datasets. In particular, all the simulations share the same initialization: $x_1(0) = 30$ since the beans are at environmental temperature, $x_2(0) = 121$ as defined in [91], $x_3(0) = 0.1$ by hypothesis, $x_4(0) = 0$ since at the beginning there is no evaporation heat and $\hat{T}_a(0) = T_a(0)$.
An optimization procedure is performed to find the optimal parameters to match as much as possible the real measure of the coffee bean temperature with the simulated one. In order to do this, the cost function is defined as the Sum of Square Residuals (SSR) between the simulated data vector ($\hat{Y} = \hat{T}_a$) and the real one ($Y = T_a$). SSR is a function of the vector $\theta = [h_{gm}\ h_{bm}\ P_{bm}]'$ used to generate the predictions:

$$SSR^j(\theta) = \sum_{i=1}^{n^j} (\widehat{Y_i^j}(\theta) - Y_i^j)^2 \tag{4.20}$$

Parameter	Unit	θ_{LB}	θ_{UB}	Optimized value
h_{gm}	$[W/m^2K]$	0.01	0.35	0.0100
h_{bm}	$[W/m^2K]$	0.01	0.35	0.0254
P_{bm}	[%]	0.5	0.8	0.5793

Table 4.3: Parameters identified on *Dataset-I* with the boundary conditions used in the optimization.

where $\widehat{Y_i^j}(\theta)$ and Y_i^j are the i^{th} samples of the predicted data obtained with a specific θ and of the real measurements respectively, and n^j is the length of the j-th batch of the *Dataset-I*.
The optimization problem is then defined as:

$$\begin{aligned} \theta^* \quad = \quad & \arg\min_\theta \sum_{j=1}^{N} SSR^j(\theta) \\ & \text{subject to } \theta_{LB} \leq \theta \leq \theta_{UB} \end{aligned} \tag{4.21}$$

with $\theta^*, \theta \in \mathbb{R}^{1\times 3}$, where $N = 5$ is the number of datasets used in identification, $\theta_{LB}, \theta_{UB} \in \mathbb{R}^{1\times 3}$ are the boundary conditions (see Table 4.3) defined by practical experience interviewing company experts.
During the optimization, the SSR is minimized through an optimization algorithm solved in MATLAB using the `GlobalSearch` function initialized with $\theta_0 = [0.01\ 0.01\ 0.5]'$.

Results

The optimization is carried out on the 120 kg dataset, called in the following *Dataset-I*. The validation is then performed simulating the model and comparing the obtained results with the real data of the 360 kg dataset, called in the following *Dataset-V*. The optimized parameters are reported in Table 4.3 along their boundaries. The quality of the model is evaluated through FIT and ρ indexes (Equations 3.20 and 3.21, in Section 3.6.1) and the obtained results are reported in Table 4.4. For each batch in *Dataset-I* on the left and *Dataset-V* on the right, the length of the batch n, SSR (Equation 4.21), FIT and ρ are reported. The last column $Av.$ of each dataset reports the average values of the performance indexes. The model obtained very good results in validation, with an average FIT of 75.49% and comparing them with the identification ones it can be seen that the two sets showed similar performances. A graphical representation of the identification results can be observed in Figure 4.2a and 4.2b, where the best and worst cases are shown, respectively. Similarly, in Figure 4.2c and 4.2d the validation results are shown. In all the plots the simulated values $\hat{T}_a(t)$ in dotted purple are compared with the real ones $T_a(t)$ in green.
The predictions are able to correctly reproduce the real data, a result that

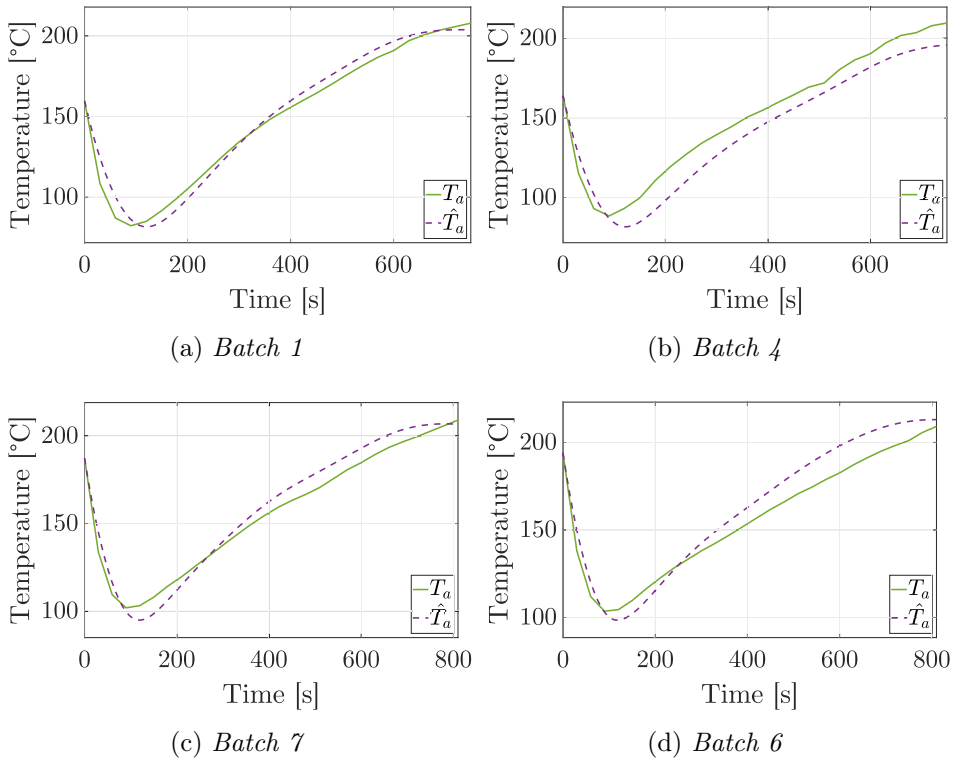
	Dataset-I						*Dataset-V*		
#	1	2	3	4	5	*Av.*	6	7	*Av.*
n	26	26	26	26	33	/	28	28	/
SSR	53.48	60.62	70.06	89.22	98.57	74.39	74.97	57.63	66.30
FIT	85.82	81.34	77.01	69.18	70.95	76.86	70.25	80.73	75.49
ρ	0.991	0.986	0.987	0.982	0.990	0.987	0.990	0.991	0.991

Table 4.4: Physical model performances on *Dataset-I* and on *Dataset-V*

is particularly interesting in validation since it proves that the model can be successfully applied to a machine with a different capacity than the one used in identification. To further investigate the portability of the proposed model, the absolute prediction error (ε) observed in the two datasets can be considered. Figure 4.3 reports its distribution along the batches in hand. Even if the variability of the identification set seems bigger (as expected taking in hand the different number of batches) the average values look definitely close. A possible way to address this empirical consideration is to compare the overall error distribution occurred over the *Dataset-V* (ε_V) with the one over *Dataset-I* (ε_I). The first sample moments are really close ($\overline{\varepsilon_I} = 7.4$ and $\overline{\varepsilon_V} = 7.2$) while the second ones show some distance ($s^2(\varepsilon_I) = 31$ and $s^2(\varepsilon_V) = 17$) that can be likely due to the different number of batches.
Although the limited number of batches used in validation, it can be reasonably assumed that the proposed model, identified on the 120 kg plant, produced satisfying results once scaled on a 360 kg plant via the parameters reported in Table 4.5.

4.6.2 LSTM model results

A first LSTM model, called *Model A*, is obtained with the goal to reproduce the portability of the physical model using a NN approach, trying to stress the robustness of the network. For this reason *Dataset-I* is employed to train (*Batch 1* - *Batch 4*) and validate (*Batch 5*) the model, while the performances of the network are then tested on *Dataset-V* (*Batch 6* and *Batch 7*). The hyperparameters of the model are obtained through an optimization via KerasTuner [110], minimizing the validation loss. The resulting values are a number of neurons $n_c = 352$ and a learning rate $\alpha = 0.009$. Also in this case the performances are evaluated in terms of FIT and ρ, so the results can be compared with the ones of the physical model.
The results obtained from the training of *Model A* are not satisfactory: evaluating the LSTM network on data completely different from those used in training bring to very poor performances, as can be seen in Table 4.6 on the left. In the first two rows the testing results are reported, with a FIT of 12.98% for *Batch 6* and 20.73% for *Batch 7*, while in the latter the validation



(a) *Batch 1* (b) *Batch 4*

(c) *Batch 7* (d) *Batch 6*

Figure 4.2: Physical model results.

results on *Batch 5* are reported for completeness. It can be observed that evaluating the network on data coming from the same machine of those used to train the network gives good results with a FIT of 60.31%, as can be seen in Figure 4.4a, where the real data in green are compared with the predicted ones in dashed purple on *Batch 5*. On the other hand, the network is not able to understand well the behavior of a bigger machine: even if the inputs are higher in the 360 kg machine, this is not sufficient to correctly determine the output, as can be seen in Figure 4.4b where the prediction is way lower than the real data. However, an interesting result can be observed looking at the ρ values on the testing batches in Table 4.6. The obtained values are very high, with 0.990 for *Batch 6* and 0.993 for *Batch 7*, meaning that real data and predicted ones are correlated, the results are simply not correctly calibrated on the machine dimensions.
However, this result is not unexpected: considering the currently available data, it is not possible to have a valid scalable LSTM model, able to model a new machine whose data are unseen during the training phase. So a second one, called *Model B*, is built as a global model valid for both the machines, adding *Batch 7* to the training dataset. In this way the network is trained considering data of both the 120 kg and the 360 kg coffee roasters. *Batch*

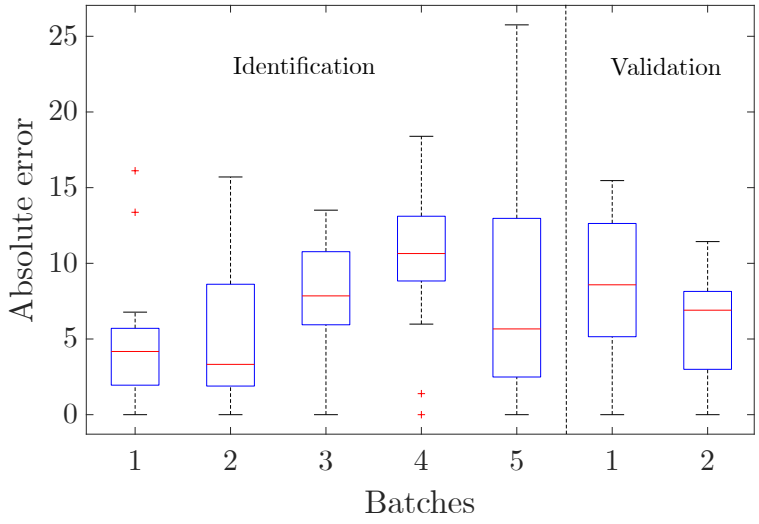


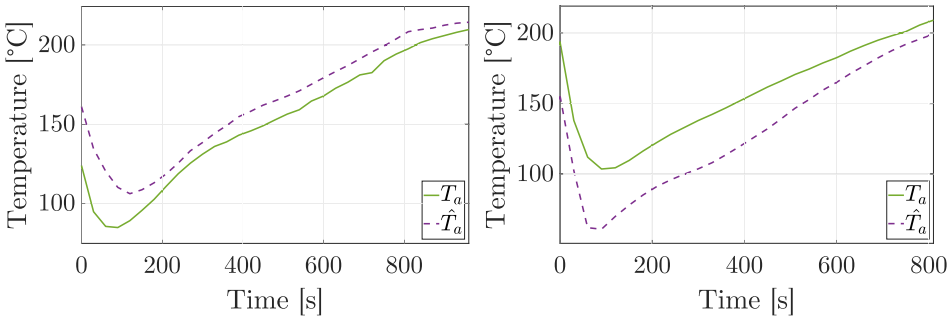
Figure 4.3: Absolute error distribution over identification and validation datasets.

Fixed param	Value	Unit	Scalable param	Value		Unit
A	116200	[kJ/kg]	M_b	120	360	[kg]
c_m	0.418	[kJ/(kg °C)]	D_{ch}	1.24	1.90	[m]
c_w	5	[kJ/(kg °C)]	H_{flap}	0.3	0.3	[m]
D_b	$7.65 \cdot 10^{-3}$	[m]	L_{ch}	1.335	2.04	[m]
H_a/R	5500	[K]	M_m	2000	7000	[kg]
H_{et}	232	[kJ/kg]	Stp_{flap}	0.1	0.1	[m]
k_1	$4.32 \cdot 10^{-9}$					
k_2	9889					
K_t	0.01	[1/s]				
m_b	$1.5 \cdot 10^{-4}$	[kg]				
λ	2790	[kJ/kg]				

Table 4.5: Parameters of the model. On the left, fixed parameters depending on the process. On the right, scalable parameters depending on the machine geometry.

Batch	*Model A*		*Model B*	
	FIT	ρ	FIT	ρ
Batch 6	12.98	0.990	87.94	0.994
Batch 7	20.73	0.993	–	–
Batch 5	60.31	0.979	70.99	0.981

Table 4.6: LSTM models results.



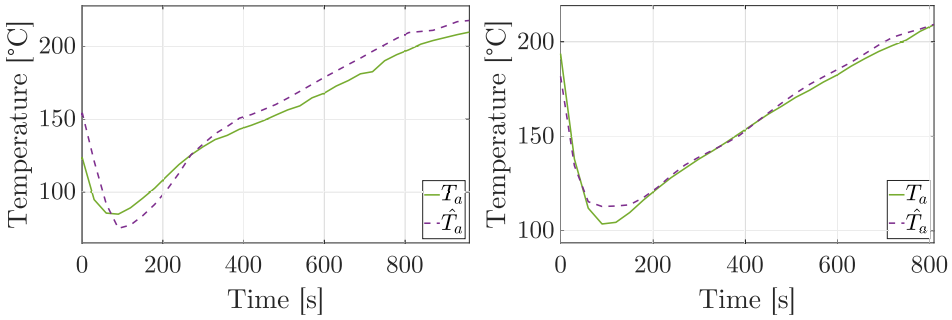
(a) *Batch 5* (b) *Batch 6*

Figure 4.4: *Model A* results.

5 is still used as validation dataset and *Batch 6* as testing dataset. Also in this case the hyperparameters are obtained through KerasTuner, resulting in number of neurons $n_c = 352$ and learning rate $\alpha = 0.007$.
The obtained results are presented in the last two columns of Table 4.6. It is clear that in this case the performances have a considerable improvement, with a FIT of 87.94% on the testing dataset, shown in Figure 4.5b, and 70.99% on the validation one, shown in Figure 4.5a. The addition of *Batch 7* in the training dataset has improved not only the result obtained on the 360 kg machine, since it gave the lacking information that were missing in *Model A*, but also the result on the 120 kg one, that can be seen in the validation dataset. For sure having more data helps in the learning phase of a NN.

4.7 Models comparison

In this chapter the modeling of an industrial coffee roaster is investigated, comparing the results obtained with a physical scalable model and with a model based on a LSTM network.
The physical model proves to be usable on plants of different size by scaling only geometrical parameters directly measurable on the roasting plant. The proposed model was obtained merging two detailed models into a well-known physical framework and defining new parameters in order to correlate



(a) *Batch 5* (b) *Batch 6*

Figure 4.5: *Model B* results.

the model to geometrical characteristics of the plant, making it scalable. The model parameters are identified from a 5-batches dataset collected on a 120 kg plant with reasonable performance. The portability is addressed by predicting the behavior of a different size plant. In particular, the scaled model is able to predict a 2-batches dataset collected on a 360 kg plant with a good performance ($FIT = 75\%$).

On the other hand, the model based on the LSTM network is not able to guarantee the portability properties of the model on plants of different sizes. By their nature, NNs are capable to reproduce only what was learned during the training phase and so a model trained on the 120 kg plant data gives poor performances once tested on the 360 kg ones. This problem is not irrelevant considering the importance assumed by the temperature during the coffee roasting process [111]. As found in [112], the coffee-like aroma can be obtained between 180 °C and 190 °C, it becomes stronger around 220 °C and 230 °C, while the coffee is over-roasted beyond this temperature. Having a drop in the accuracy of the temperature modeling involves in this application also a drop in the quality of the obtained product.

Under these conditions, it is only possible to train a global model with data of both the 120 kg and 360 kg machines. The results are satisfying with FIT of 88% for the 360 kg machine and FIT of 71% for the 120 kg one. To have a scalable LSTM model, an idea for a future development is to use a physics-based LSTM, linking the structure of the network to the physical knowledge of the plant and in particular to its size.

In this particular application the data collection is not a trivial aspect due to the huge dimension of the plant that requires not negligible time and resources, in this case coffee, that do not have to go to waste. If the data are ad hoc collected only for modeling purposes, it is simpler and more convenient to collect data from smaller machines and to build models that are still valid also for larger plants, not only from an economical point of view but also in a sustainability perspective, to avoid food waste. Recently, thanks to more ad-

vanced machines that are equipped with intelligent sensors, it is possible to easily collect a great amount of data directly during the production phase. In this way it is not necessary to perform experiments only for data collection, minimizing wastes, and it is also possible to collect data from machines of various sizes. Anyway, not all the signals are available from these sensors, especially those obtained through the special setup described in Section 4.3.2, that is the main reason for which only few datasets were used in this work. This approach is for sure ideal for the LSTM model, that thanks to a bigger data availability can have better performances, but anyway it is necessary to rethink the model, probably considering different inputs or outputs, that can also be useful to develop the final model of the machine. In fact, future developments currently under study include the modeling of the other components of the plant that influence the chamber process. Once the whole plant is modeled in detail, new intelligent control approaches (e.g. hybrid control) could be explored in order to optimize the roasting process both in terms of efficiency (ecological and productive), predictive maintenance and analytic. The final goal is to build a simulator in order to synthesize and test new complex control approaches [113].

Chapter 5

Modeling of a wastewater treatment plant

In this chapter, a third industrial case study is presented, comparing a white-box model and a LSTM one of a wastewater treatment plant. In this case, a well-known model is present in literature but it can not be applied for lack of data. So starting from it, a model is built exploiting the company's expertise. Along with this model also a LSTM one is presented, taking advantage of the availability of a vast amount of data.

5.1 Motivation and state of the art

A WasteWater Treatment Plant (WWTP) is a plant where the wastewater is treated to remove pollutants, exploiting biological and chemical reactions before being released back in the environment. The modeling and control of this process is not an easy task, since the treatment depends on the nature and the characteristics of the wastewater. Wastewater usually firstly undergoes to chemical-physical treatments, in order to remove the solid part of the wastes, and then to biological treatments for the organic components. One of the most common biological treatment used in WWTPs is the Conventional Activated Sludge (CAS) process, where bacteria are used to nitrify and denitrify wastewater.
Considering the complexity of the process, controlling a WWTP can be a challenging task: several flows may be involved, among with a huge variety of different biological and chemical reactions; it is necessary to provide a sufficient oxygen quantity but without an excess of aeration, reaching a trade-off between energy consumption and process demand, with a growing attention to the environmental related problems. To design an optimal control strategy it is necessary to have an accurate model of the process [114]. In [115] a review on the current state of the art regarding modeling of activated sludge WWTPs is presented, considering both white-box models and black-

box ones. The model of the entire WWTP is usually composed by two main components: the hydraulic model that takes into account the different flows in input and output in the reactor, and the Activated Sludge Model (ASM) that models the biological and chemical reactions inside the tank due to the activated sludge process. The control and design of the ASM is a further control problem in literature [116], strictly linked to environmental issues and new severe regulations. In 1983 the International Association on Water Quality (IAWQ) organized a task group to develop a mathematical model of the activated sludge problem with low complexity but accurate from a biological point of view. The first model created by the task group was the so called Activated Sludge Model No. 1 (ASM1) [117], also known as IAWQ model, extended in the years to consider the phosphorus dynamics in ASM2 [118] and including storage of organic substrates as a new process with an easier calibration in ASM3 [119]. Anyway, ASM1 is still the most used model and it can be considered the state of the art since several successive works are based on it. For example, in [120] the ASM1 is used in order to derive the steady-state behavior of the system and find the values of some model's parameters; then in a successive work [121] the authors use the ASM1 not fixing the value of the dissolved oxygen concentrations like in many other works, including also the recycle flow of the settling unit.
In [122], one of the authors of the ASMs recalls the developments of these models during the years, highlighting the high number of contributions in this field. Anyway, his feeling is that the models are more and more complex and specialized, requiring the identification of parameters always more specific and subject to high sensitivity, with expensive and time consuming procedures, far from practical applications. As said also in [123], these specialized models are not the best for process design and practical applications, so the authors propose to leverage these modeling tasks to a next step, combining process knowledge with new artificial intelligence techniques. Also in [115], in addition to the most diffused WWTP white-box modeling, some black-box methodologies based on NN techniques were already presented. In view of all these considerations and regarding the complexity of the process and its high non-linearity, a NN approach is interesting, especially when the available input-output data are not describing the biological and chemical reactions, but only flows and concentrations.
In this chapter, the modeling of a WWTP located in Mortara (Pavia), managed by the company ASMortara S.p.A. [124], is presented. The main goal is the development of a model for the biological reactor of the plant to improve several critical aspects of the process such as the oxygen concentration regulation. The data used in this chapter, in fact, have been collected with a simple switching controller that enables/disables the input flow if the oxygen concentration inside the reactor is below/above a certain threshold. The effect of this control strategy is an on/off oxygen flow rate, that generates important oscillations in the oxygen concentration evolution, making the

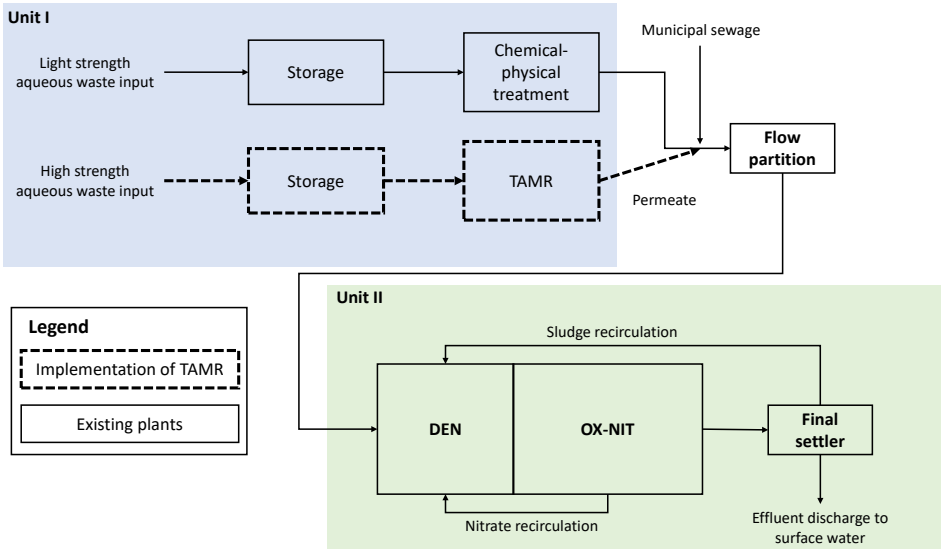


Figure 5.1: Flow diagram of the WWTP.

modeling of this process a tricky task. Even if sub-optimal, the oxygen oscillating dynamic was considered useful by the company to have an estimation of the health of the biomass. In fact, biomass and oxygen quantities inside the reactor are strictly linked since the biomass is composed by aerobic bacteria, i.e. bacteria that need oxygen to grow and can use it to oxidize substrates. In this kind of treatments the aerobic bacteria can be stimulated by increasing the oxygen quantity in the reactor to keep them alive. Anyway, it is not needed to do it constantly, considering also that this procedure can not guarantee good process performances: the plant can be better controlled, leaving the biomass check only to periodic laboratory analysis that can be done when needed.
In the following, two different modeling approaches are proposed for this plant. The first one is a white-box model, composed by two contributions: the biological component, developed starting from the ASM1 and employing the available input/output data, and the hydraulic component that models the flows ingoing and outgoing the reactor. The second one is a LSTM based model, developed exploiting the availability of a large amount of data.
The results of this chapter are part of the work presented in [16].

5.2 Plant description

The considered WWTP is an industrial plant used to treat both municipal and industrial wastewater. The plant is composed by two different units, as shown in Figure 5.1. The treatment is usually divided in two stages, called

primary and secondary treatment, that corresponds to the two units of the plant.
The industrial wastewater needs to be pre-treated because of its characteristics before being accepted in municipal WWTP. So the industrial wastewater is pre-treated in Unit I to remove the solid part of the waste, exploiting chemical reactions. Firstly, the flow goes into a storage-equalization tank, with the goal to transform a variable flow into a steady-state one that goes in input to the treatment plant, optimizing in this way the process. The flow is then divided in two chemical-physical treatment lines: one dedicated to the light strength aqueous wastes and the second for the high strength ones. The light strength wastes are subjected to a chemical-physical treatment with classic procedures like flocculation, coagulation and sedimentation. The wastes are put in sedimentation tanks where the solid part is separated by gravity; chemicals can be added to help coagulation, while with flocculation the small colloidal particles are separated from the wastes and settle in form of flocks. The high strength aqueous waste instead are fed into a Thermophilic Aerobic Membrane Reactor (TAMR). The plant has been recently equipped with this innovative reactor, built ad-hoc to treat high strength wastes. The advantages of the TAMR technology is the exploiting of biological reactions even during the pre-treating phase, considering that biological solutions are more affordable and sustainable of chemical ones. In practice, the waste undergoes to aerobic reactions in thermophilic conditions, exploiting oxygen with temperatures greater than 45 °C [125, 126].
After this phase, the pre-treated industrial wastewater is mixed with the municipal wastewater and flow into the biological reactor (Unit II), where the CAS comprehends denitrification (DEN), oxidation and nitrification (OX-NIT) stages: this is the part of the plant of interest and that will be modeled in the following. The biological reactor is filled with oxygen, where aerobic microorganisms are introduced to react with wastewater and reduce its organic compounds. The overall flow goes to a settling tank (Final settler) where the activated sludge settles, since the micro-organisms create a biological flock, producing a liquid mostly free from suspended solid. The sludge is separated from the clarified water: the majority of the sludge is reintroduced in the CAS to treat the new wastewater, the remaining part is removed, while the cleared water is released in the environment. In the biological reactor the wastewater is subjected also to the denitrification and nitrification processes to remove the nitrogen components from the wastewater. This is done before releasing back the water in the environment since these components are dangerous for aquatic organisms and plants. The nitrification occurs in aerobic conditions and is the oxidation of ammonia into nitrite and nitrate by nitrifying autotrophic bacteria; the denitrification instead happens in anaerobic conditions to reduce nitrite to nitrogen [127]. For this reason nitrate recirculates from OX-NIT to DEN in Figure 5.1.

5.3 Physical model

The current state-of-the-art for the modeling of the biological reactor is the ASM1. The dynamic of this model is composed by 13 state variables, that describe the concentrations of different materials inside the reactor:

- S_I, inert soluble organic material;
- S_S, readily biodegradable soluble substrate;
- X_I, particulate inert organic matter;
- X_S, slowly biodegradable particulate substrate;
- $X_{B,H}$, active heterotrophic particulate biomass;
- $X_{B,A}$, active autotrophic particulate biomass;
- X_P, inert particulate products arising from biomass decay;
- S_O, soluble oxygen;
- S_{NO}, soluble nitrate and nitrite nitrogen;
- S_{NH}, soluble ammonium nitrogen;
- S_{ND}, soluble biodegradable organic nitrogen;
- X_{ND}, particulate biodegradable organic nitrogen;
- S_{ALK}, alkalinity.

These quantities can be grouped as: the carbonaceous components, divided in biodegradable (soluble S_S and particulate X_S), non-biodegradable (soluble S_I and particulate X_I, X_P) and active biomass (heterotrophs $X_{B,H}$ and autotrophs $X_{B,A}$); the nitrogenous components, divided in ammonia (S_{NH}), biodegradable (soluble S_{ND} and particulate X_{ND}), nitrate and nitrite (S_{NO}); oxygen concentration (S_O) and alkalinity (S_{ALK}). The inert components S_I and X_I are considered not affected by the biological reactions and are usually removed from the system when excessive sludge is removed [128].
These state variables are considered to model the different processes that are included in the ASM:

1. aerobic growth of heterotrophs;
2. anoxic growth of heterotrophs;
3. aerobic growth of autotrophs;
4. decay of heterotrophs;

5. decay of autotrophs;

6. ammonification of soluble organic nitrogen;

7. hydrolisis of entrapped organics;

8. hydrolisis of entrapped organic nitrogen.

Given the complexity of this model, a reduced version of the ASM1 described in [120] is investigated, where out of the 13 differential equations that compose the model, only 9 are taken into consideration, removing S_I, X_I, X_P and S_{ALK}, since these quantities are uncoupled from the others and so do not affect the system dynamics. However, even considering these simplifications, the application of the reduced ASM1 is not easy because of the lack of the required data. In order to use this model, further assumptions are needed but in this way too many dynamics of the model are neglected and the results would not be satisfying. For example, it is not known the percentage of the nitrogenous components or the division between biodegradable and non-biodegradable ones. Considering the data provided by the company, it is not possible to apply neither the ASM1 nor the reduced one, since there are not information about the chemical reactions occurring inside the reactor.
To overcome these difficulties, a lumped-parameter model is defined, where only the principal processes are modeled (1-5), neglecting the remaining ones, being aware that the resulting model will be suboptimal but sufficient to achieve the company's goals. Consequently, only the quantities affecting the oxygen consumption, O, are considered. Starting from the ASM1 state variables, two new ones are defined:

- an overall biomass component:

$$X = X_{B,H} + X_{B,A}$$

- an overall substrate component:

$$S = X_S + S_S + X_{ND} + S_{ND}$$

Lastly, in addition to the biological dynamic, also the hydraulic one is modeled, through the state variable h, the level inside the reactor, dependent on the input and output flow rates specific for this plant. So this lumped-parameter model is characterized by four state variables, that are deeply described in the following.

5.3.1 Equations

Biomass concentration

The first state of the model is the biomass concentration inside the reactor, $X(t)$, that is given by

$$\dot{X}(t) = X_s(t) - X_d(t) - \tilde{X}_m(t) \tag{5.1}$$

where $X_s(t)$ is the biomass growth rate with respect to the substrate, $S(t)$; $X_d(t)$ is the natural biomass decay rate; $\tilde{X}_m(t)$ is a biomass quantity daily removed from the reactor. $X_s(t)$ is expressed as:

$$X_s(t) = \hat{\mu}\frac{X(t)S(t)}{K_s + S(t)} \tag{5.2}$$

In $X_s(t)$ the presence of the dissolved oxygen, $O(t)$, that influences the kinetic in aerobic processes is considered, as represented in the Monod equation [129], since $\hat{\mu}$ is the maximum specific growth rate and contains the oxygen dependence:

$$\hat{\mu} = \hat{\mu}_1\frac{O(t)}{K_c + O(t)} \tag{5.3}$$

The coefficient K_s is the half-velocity constant i.e. the substrate concentration where the maximum specific growth rate is half of the maximum velocity, while the coefficient K_c is similarly the half-velocity constant referred to the oxygen.
The second component of the equation, $X_d(t)$ is expressed as:

$$X_d(t) = bX(t) \tag{5.4}$$

where b is the endogenous decay coefficient.
The third component of the equation, $\tilde{X}_m(t)$, is empirically defined with company's experts as:

$$\tilde{X}_m(t) = \frac{X(t)}{Ah(t)}q_{om}(t) \tag{5.5}$$

Each day a quantity of biomass and substrate, $q_{om}(t)$, is manually removed from the reactor because the natural decay rate of the biomass is not fast enough to keep the process under control and it could saturate the process. Since A is the tank surface and $h(t)$ the level, the concentration of biomass manually removed is considered proportional to the concentration of the biomass contained in the volume ($Ah(t)$) of the reactor at that moment.
Finally, making explicit all the components, the biomass concentration is defined as:

$$\dot{X}(t) = \hat{\mu}_1\frac{O(t)}{K_c + O(t)}\frac{X(t)S(t)}{K_s + S(t)} - bX(t) - \frac{X(t)}{Ah(t)}q_{om}(t) \tag{5.6}$$

Substrate concentration

The second state is the substrate concentration inside the reactor, $S(t)$, that is modeled by

$$\dot{S}(t) = -S_s(t) - \tilde{S}_m + S_{in}(t) - S_{out}(t) \tag{5.7}$$

where $S_s(t)$ is the substrate removal rate; $\tilde{S}_m(t)$ is a substrate quantity daily removed from the reactor; $S_{in}(t)$ is the quantity of substrate in input to the reactor; $S_{out}(t)$ is the quantity of substrate in output from the reactor. $S_s(t)$ is defined as:

$$S_s(t) = \frac{1}{Y} X_s(t) \tag{5.8}$$

that is directly proportional to the biomass growth rate in Equation (5.2), $X_s(t)$, according to a constant of proportionality $1/Y$.
The second component, $\tilde{S}_m(t)$, is equivalent to $\tilde{X}_m(t)$ in Equation (5.5), considering in this case the quantity of substrate that is manually removed and is defined as:

$$\tilde{S}_m(t) = \frac{S(t)}{Ah(t)} q_{om}(t) \tag{5.9}$$

The third component, $S_{in}(t)$, is defined as:

$$S_{in}(t) = \frac{COD_{in}(t)}{Ah(t)} q_i(t) \tag{5.10}$$

It is computed in terms of Chemical Oxygen Demand (COD) that the input substrate flow rate, $q_i(t)$, requires. Equivalently, $S_{out}(t)$ is expressed as:

$$S_{out}(t) = \frac{COD_{out}(t)}{Ah(t)} q_{ol}(t) \tag{5.11}$$

Also this term is computed as the COD that the output liquid substrate flow rate, $q_{ol}(t)$, requires. Since $q_{ol}(t)$ is the outgoing quantity from the reactor, resulting from the filtering process, its oxygen demand is lower than the one required by $q_i(t)$. For this reason two different quantities are defined, called $COD_{in}(t)$ and $COD_{out}(t)$ respectively. These quantities are divided by the volume of the tank ($Ah(t)$), in order to represent the concentration variation. Also these components have been defined according with company's experts. Considering all the mentioned contributions, the substrate concentration is defined by:

$$\begin{aligned} \dot{S}(t) = &- \frac{\hat{\mu}_1}{Y} \frac{O(t)}{K_c + O(t)} \frac{X(t)S(t)}{K_s + S(t)} - \frac{S(t)}{Ah(t)} q_{om}(t) \\ &+ \frac{COD_{in}(t)}{Ah(t)} q_i(t) - \frac{COD_{out}(t)}{Ah(t)} q_{ol}(t) \end{aligned} \tag{5.12}$$

Oxygen concentration

The third state is the oxygen concentration inside the reactor, $O(t)$, and is modeled by

$$\dot{O}(t) = -O_s(t) - O_d(t) + O_{in}(t) - c_{oss}O(t) \tag{5.13}$$

where $O_s(t)$ represents the oxygen consumption due to the substrate removal rate, $S_s(t)$; $O_d(t)$ represents the oxygen consumption due to the biomass decay rate, $X_d(t)$; $O_{in}(t)$ represents the concentration of the oxygen flow rate in input in the reactor; the last component models an oxygen leak from the reactor, that have been empirically noticed, proportional to a coefficient c_{oss}. $O_s(t)$ is defined as:

$$O_s(t) = \alpha S_s(t) \tag{5.14}$$

indeed it is proportional to $S_s(t)$, defined in Equation (5.8), according to a constant of proportionality α.
The second component, $O_d(t)$, is expressed as:

$$O_d(t) = \beta X_d(t) \tag{5.15}$$

being so proportional to $X_d(t)$, defined in Equation (5.4), according to a constant of proportionality β.
The third component, $O_{in}(t)$, is defined as:

$$O_{in}(t) = c_{molg}\frac{q_{oss}(t)}{Ah(t)} \tag{5.16}$$

It depends on the oxygen flow rate in input to the reactor, $q_{oss}(t)$, divided by the volume of the reactor ($Ah(t)$), and on a conversion factor, c_{molg}, needed to convert the volumetric flow rate into a mass flow rate. In particular, c_{molg} is defined as

$$c_{molg} = \frac{M \cdot P_o}{R \cdot T_o} \tag{5.17}$$

where M is the molar mass of the oxygen, P_o is the oxygen pressure, R is the universal gas constant and T_o the oxygen temperature.
To conclude, the oxygen concentration is explicitly modeled by:

$$\begin{aligned}\dot{O}(t) = &- \alpha\left(\frac{\hat{\mu}_1}{Y}\frac{O(t)}{K_c + O(t)}\frac{X(t)S(t)}{K_s + S(t)}\right) - \beta\left(bX(t)\right) \\ &- c_{oss}O(t) + c_{molg}\frac{q_{oss}(t)}{Ah(t)}\end{aligned} \tag{5.18}$$

Level

The fourth state is the level inside the reactor, $h(t)$, that depends on the hydraulic component of the model and is defined as:

$$\dot{h}(t) = \frac{q_i(t) - q_{ol}(t) - q_{om}(t)}{A} \tag{5.19}$$

The level variation depends on the influent flow rate $q_i(t)$ and the effluent ones $q_{ol}(t)$ and $q_{om}(t)$, divided by the reactor surface, A.

Final model equations

Considering the above considerations about the system dynamic, the lumped-parameter model with 4 states, 6 inputs and 2 outputs can be defined. The states of the system are:

- $x_1(t) = X(t)$: biomass concentration;
- $x_2(t) = S(t)$: substrate concentration;
- $x_3(t) = O(t)$: oxygen concentration;
- $x_4(t) = h(t)$: level inside the reactor.

The inputs of the system are:

- $u_1(t) = q_{oss}(t)$: ingoing oxygen flow rate;
- $u_2(t) = q_{in}(t)$: ingoing substrate flow rate;
- $u_3(t) = q_{ol}(t)$: outgoing liquid substrate flow rate;
- $u_4(t) = q_{om}(t)$: quantity of biomass and substrate manually removed from the reactor;
- $u_5(t) = COD_{in}(t)$: Chemical Oxygen Demand of the influent;
- $u_6(t) = COD_{out}(t)$: Chemical Oxygen Demand of the effluent.

Lastly, the outputs of the system are:

- $y_1(t) = O(t)$: oxygen concentration;
- $y_2(t) = h(t)$: level inside the reactor.

5.4 LSTM model

The WWTP's dynamic is a complex process, where several contributions need to be taken into account and it is hard to obtain the data needed to model the chemical and biological reactions. Instead, it is possible to easily collect a great amount of input/output data, that can be employed in a black-box model.
In view of these considerations, a NN model based on a LSTM network is developed considering 3 inputs: the ingoing oxygen flow rate $q_{oss}(t)$, the ingoing substrate flow rate $q_{in}(t)$ and the outgoing liquid substrate flow rate $q_{ol}(t)$, and one output, the oxygen concentration $O(t)$. In this case the other inputs

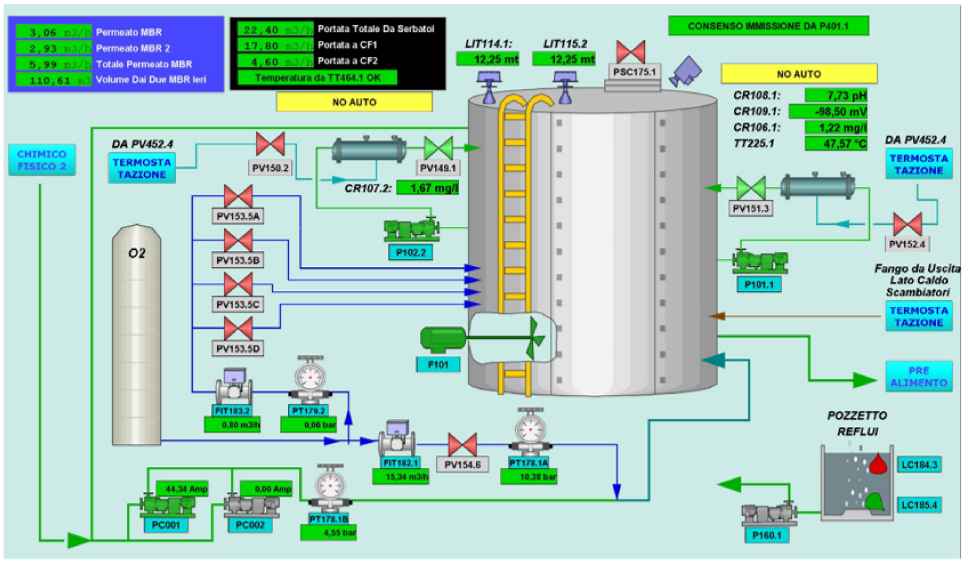


Figure 5.2: SCADA interface of the biological reactor.

of the physical model $q_{om}(t)$, $COD_{in}(t)$ and $COD_{out}(t)$ are not considered in input since they are daily measurements, with very little variations, that can be considered almost constant signals that do not give contribution in a NN model. All the signals have been rescaled using mean normalization before being used to train the LSTM network.
The used architecture is a single layer LSTM with 160 neurons, trained for 1000 epochs, using Adam optimizer with learning rate $\alpha = 0.002$ and MSE loss function (Equation 4.19 in Section 4.4). The values of the hyperparameters have been obtained through an optimization with KerasTuner [110], employing the `RandomSearch` algorithm in a range of $32 < n_c < 512$ for the number of neurons and a range of $0.001 < \alpha < 0.01$ for the learning rate.
The training of the network is performed on a computer Intel i9-10920X CPU with 3.50 GHz, with graphics processing unit GPU NVIDIA GeForce RTX 2080Ti; it has been written in Python 3.9, using TensorFlow [108], Keras API [109] and KerasTuner [110], thoroughly described in Appendix A.

5.5 Datasets description

The WWTP is monitored through a SCADA system used to manage the control protocols, alarm handling and also for data collection. From the SCADA interface the user can have access to sensors equipped in the chemical-physical reactor where the wastewater is pretreated, in the biological reactor and in the two membrane bioreactors, called respectively MBR1 and MBR2. An example of the interface is presented in Figure 5.2 where the biological reactor interface is shown.

Dataset name	Start date	End date
Dataset 1	06/12/2021	11/12/2021
Dataset 2	15/12/2021	20/12/2021
Dataset 3	21/12/2021	26/12/2021
Dataset 4	27/12/2021	01/01/2022
Dataset 5	01/03/2022	06/03/2022
Dataset 6	01/05/2022	06/05/2022

Table 5.1: Datasets description.

In this way, the signals of interest can be downloaded. The value of $q_{in}(t)$ is obtained from the chemical-physical reactor, from which it goes in input into the biological reactor, after the preprocessing of the wastewater. In the reactor are then present sensors that measure the ingoing oxygen flow rate, $q_{oss}(t)$, the oxygen concentration, $O(t)$, measured at a height of 4 meters, and the total level, $h(t)$, obtained through two sensors located on the top of the reactor. Then, the measure of $q_{ol}(t)$ is obtained from MBR1 and MBR2, as the sum of the two components that from the biological reactor flow into the two membrane bioreactors.
Lastly, COD_{in} and COD_{out} are obtained through laboratory analysis, while $q_{om}(t)$ is acquired daily by the operators that remove this quantity from the reactor. These quantities are not considered in the LSTM model, not only for their constant values, but also for the difficulty in their acquisition, since to train the NN several datasets are needed.
Six different datasets are collected, each one containing six days of data, as in Table 5.1. All the datasets contain the three inputs and the two outputs previously described; *Dataset 1* contains also $COD_{in}(t)$, $COD_{out}(t)$ and $q_{om}(t)$, since it was the first dataset provided directly by the company. Unluckily, these values are not obtainable from the SCADA interface and so they are not available for the remaining datasets. For these reasons, *Dataset 1* is used to assess the lumped-parameter model performances and as testing dataset in the LSTM model, while the remaining 5 datasets are used to train and validate the LSTM model.

5.6 Discussion

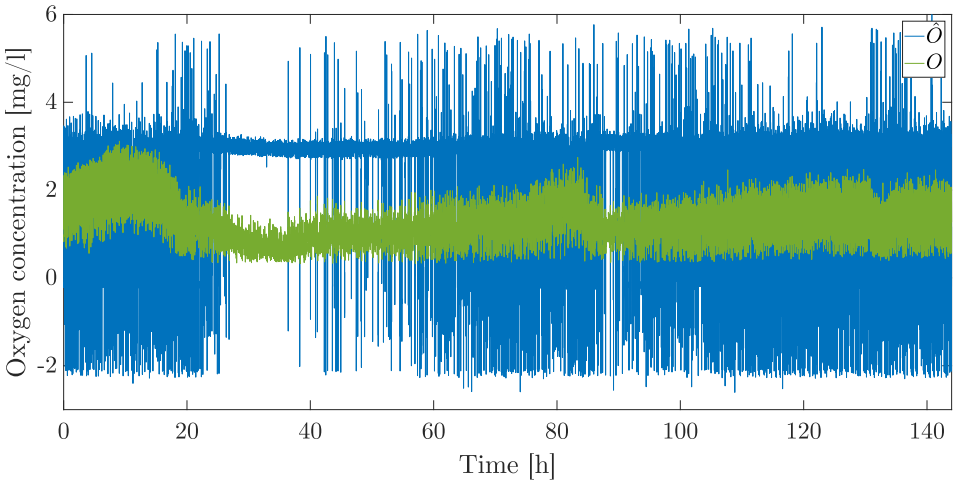
5.6.1 Physical model results

The lumped-parameter model, described by Equations (5.1), (5.7), (5.13) and (5.19), is used to predict the oxygen concentration and the level inside the biological reactor on *Dataset 1*. The values of the parameters are listed in Table 5.2. Some of these values, like K_s, K_c, Y are taken from literature [117, 120, 121], the others are obtained from company's know-how.

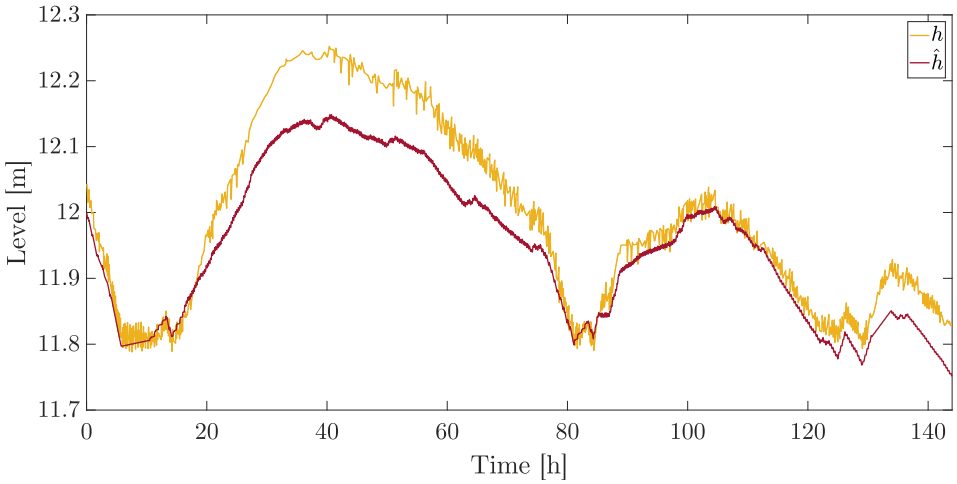
Parameter	Value	Unit
$\hat{\mu}_1$	$1.3 \cdot 10^{-4}$	[h$^-$1]
K_s	20	[mg/l]
K_c	0.4	[mg/l]
b	$1.06 \cdot 10^{-5}$	[h$^-$1]
A	113	[m^2]
Y	0.11	
α	6.12	
β	1.22	
c_{oss}	185	[h$^-$1]
M	32	[g/mol]
P_o	$8 \cdot 10^5$	[Pa]
R	8.31	[J/mol K]
T_o	292.65	[K]

Table 5.2: Parameters of the lumped-parameter model.

In Figure 5.3, the real data collected for *Dataset 1* and the prediction of the physical model are presented. The oxygen concentration is presented in Figure 5.3a, where the real data in green are compared with the prediction in blue. The obtained result is poor; it is clear that the modeling phase did not lead to a good model formulation and a good choice of the parameters values. The behavior of the prediction is sometimes even opposite to the real one, and the tricky oscillating dynamic is hard to catch. On the other hand, the prediction of the level inside the reactor is pretty good, with a FIT of 59%. In Figure 5.3b, the real data, in yellow, are compared with the prediction, in red, emphasizing the model's ability to understand the level variation. Between 20 and 60 [h] the level variation is less accurate, but the error is about 10 cm, that is considered acceptable by the company for their goals.
As said before, the lumped-parameter model consists of two components: the biological component, from which the oxygen concentration is derived and the hydraulic one, from which instead the level is determined. The hydraulic component is way simpler, defined by the input/output flows in the reactor; these signals can be easily acquired and their measurements are accurate. The biological component instead depends on a big variety of elements and a specialized knowledge in the field is needed to correct apply it, and probably this knowledge was lacking in the modeling phase. Better results can be achieved with a deeper understanding of the process: it is necessary to know the biological and chemical components of both biomass and substrate, how they interact and how to collect these kind of data. In such a way it will be possible to apply both the ASM1 or its reduced version.



(a) Oxygen concentration



(b) Level

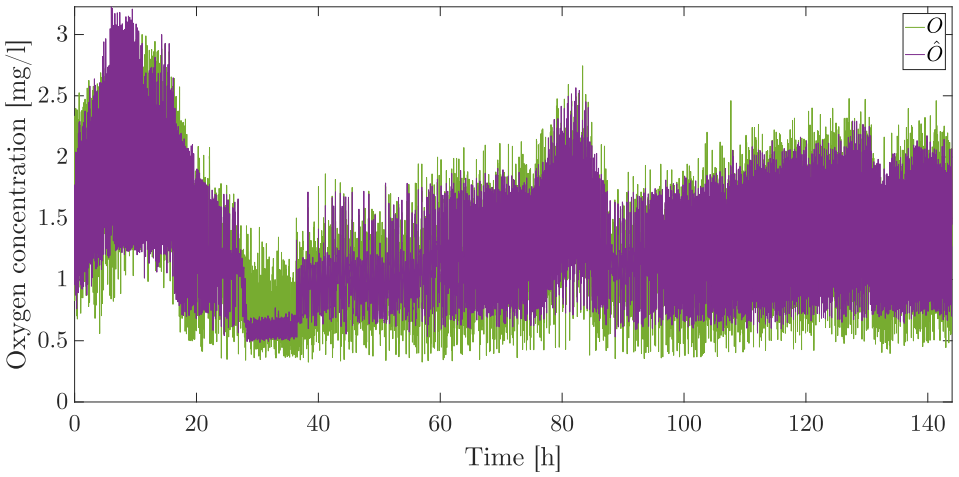
Figure 5.3: Results of the lumped-parameter model.

5.6.2 LSTM model results

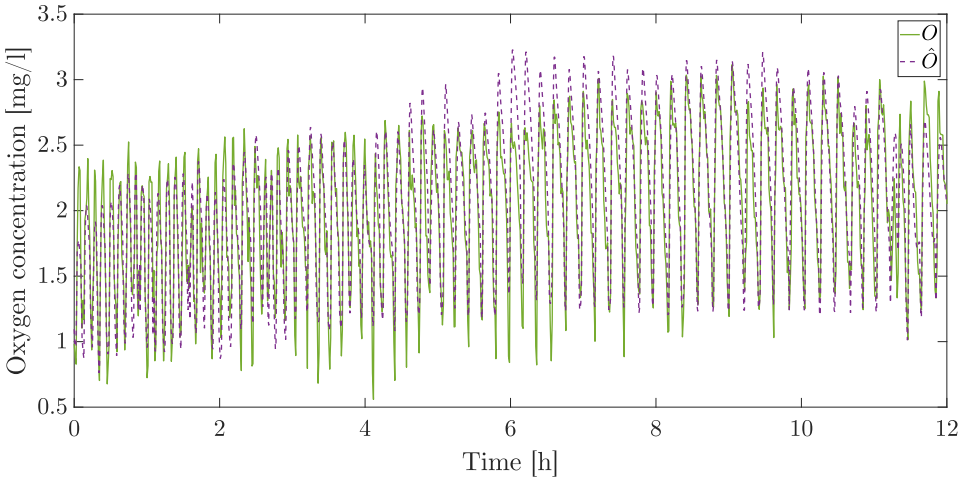
A model based on a LSTM network is defined in order to predict the oxygen concentration inside the reactor. Since it was difficult to model this quantity with a white-box approach and a big amount of input/output data is available, the use of a NN model looks promising. The modeling task using the LSTM model is focused only on the oxygen concentration since it was the trickiest output to model with the white-box approach, while the level performances were already good enough for the company's goals.
The training and validation of the network are performed on five datasets (*Dataset 2* - *Dataset 6*) for a total amount of thirty days, while it is tested on *Dataset 1*, so that the model can be comparable with the physical one. The results are shown in Figure 5.4, with a global overview in Figure 5.4a and a closer detail of the first 12 hours in Figure 5.4b, to better emphasize the accuracy of the result. It can be observed that both the global dynamic of the signal and the oscillations are correctly reproduced, with an overall satisfying outcome. Calculating the performance indexes (FIT and ρ, described in Equations (3.20) and (3.21) in Section 3.6.1) on the signals, the results are quite satisfying, with $FIT = 60.56\%$ and $\rho = 0.921$. It is clear that a straightforward improvement is obtained with respect to the result of the lumped-parameter model.

5.7 Models comparison

The modeling task of a WWTP is described in this chapter with both a white-box model and a black-box one.
The overall WWTP modeling and the ASM, regarding the reactions inside the biological reactor of the plant, are well-known case studies in literature. However, the ASM could not be applied to this plant because of the lack of both data and knowledge of the chemical and biological reactions. So a lumped-parameter model is presented in this thesis, built starting from the state-of-the-art and exploiting the company's expertise and the available input/output data, in order to model the oxygen concentration and the level inside the reactor. The results are not outstanding: the simulated level is quite good, with a FIT of 59%, being able to reproduce the behavior of the original signal; on the other hand, the simulated oxygen concentration is totally wrong and the model is not able to recreate the model's dynamic. The considered process is not easy to model, it requires a deep knowledge of the biological and chemical reactions that are involved, there exists a rich literature about it and the adaptation of an existent working model can not be done easily. In order to propose and use a white-box model of this process, a deeper study is required, together with more specific data and possibly involving experts of the related fields. Moreover, the considered process has some control issues, since the actual control system produces an



(a) Oxygen concentration



(b) First 12 h zoom

Figure 5.4: Results of the LSTM model.

undesired oscillating dynamic in the oxygen input flow that is propagated also to the oxygen concentration. The company is currently working on the use of a proportional valve to replace the current switching one. In this way, the input of the system will have a smoother profile and so also the oxygen concentration. A data collection after this substantial change in the machine can be useful in order to analyze and improve the existing model.
Currently, considering the difficulties experienced trying to use a white-box model and the availability of input/output data that can be easily acquired, a NN model seems instead the most promising approach. A LSTM model is trained on thirty days of data in order to reproduce the oxygen concentration inside the reactor. The result is promising, with a FIT of around 61%, being able to represent even the switching behavior of the output variable. For sure also in this case a better control of the process can help during the training phase of the network. A pros of the NN approach with respect to the white-box one is that a new model can be easily trained again if a new data collection is required, at most a new search of the optimal hyperparameters should be needed.
On the other hand, a limitation of the LSTM model is the inability to have a clear view of the interaction between the physical quantities involved in the process. As seen in Section 5.3, the oxygen concentration plays an essential role in this process, considering that the other quantities depend on it. So it is important that the network is able to correctly reproduce how the oxygen concentration influences the other quantities, but currently, this is not known. To deeper investigate this correlation, a future development of this work is the implementation of a physics-based LSTM that takes into account the physical dependencies of the different involved quantities.
To conclude, this application represents a perfect example of when is preferable to use a NN approach: difficulty to model a process considering only physical equations; availability of input/output data that well represents the characteristics that is needed to model, that can be easily collected and in big quantity; need of a simple model for future control purposes in model-based controllers.

Chapter 6

LSTM networks for glucose prediction

In this chapter, an application of the LSTM networks related to the artificial pancreas is presented. The modeling of the glucose-insulin and glucose-meal dynamics is a challenging task, due to the non-linearity and complexity of the process. So, a NN approach can be appropriate for this kind of problems. The LSTM networks are used in this thesis to predict the glucose values in diabetes patients, in order to create an alarm system for hypoglycemia and hyperglycemia prevention.

6.1 Motivation and state of the art

Diabetes refers to a group of common metabolic conditions that share the phenotype of hyperglycemia, i.e. when Blood Glucose (BG) reaches high values (BG>180 mg/dl). Type 1 Diabetes (T1D) is the result of complete or near-total insulin deficiency due to an autoimmune process against the β-pancreatic cells, cells that produce the insulin. Indeed, T1D is defined as an insulin-dependent disease, i.e. it can only be treated with exogenous insulin injections to decrease the BG levels, that involve risks of hypoglycemia (BG<70 mg/dl) if excessive. In the next years T1D incidence in the population is supposed to largely increase. The International Diabetes Federation reports approximately 537 million adults with diabetes and an expected increase to 643 million of people living with diabetes by 2030 [130]. Then, the T1D optimal treatment is a challenging and interesting control problem.
The T1D standard treatment is the basal-bolus therapy constituted of a piecewise constant amount of insulin delivered all along the day during fasting periods, called basal, and some meal boluses. A meal bolus is an impulse-like amount of insulin delivered to compensate the glucose rise due to a meal intake, computed using the estimation of the carbohydrate intake of the meal [131]. Along with the conventional therapy, in the past years the use

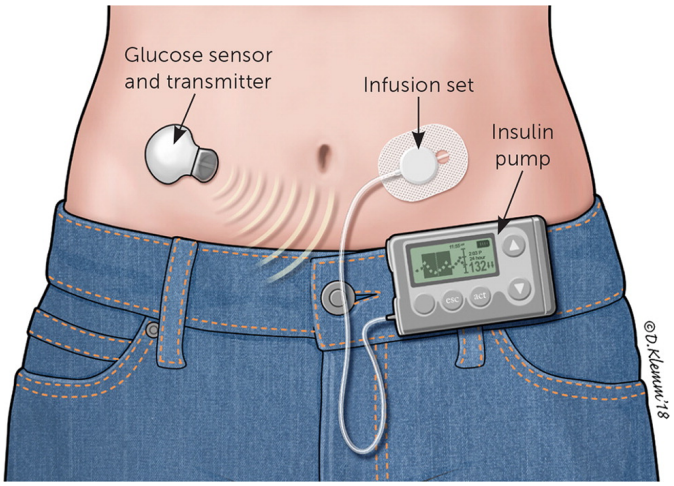


Figure 6.1: Constitutive elements of the artificial pancreas.

of Continuous Glucose Monitoring (CGM) sensors and insulin pumps spread around in the so-called Sensor-Augmented Pump (SAP) system. The use of SAP systems brought an improvement in the glucose control, reducing the incidence of adverse episodes and improving at the same time treatment satisfaction and diabetes related distress [132–134]. Moreover, at the same time systems called Artificial Pancreas (APs) have been developed to correctly administrate insulin, in order to prevent both hypoglycemia and hyperglycemia phenomena in T1D patients. An AP is a closed-loop system that computes and directly delivers the optimal insulin quantity through an insulin pump, exploiting CGM measurements and other additional information provided by the patient to estimate this optimal quantity. In Figure 6.1 the constitutive elements of the AP are shown: the glucose CGM sensor on the left and the insulin pump on the right. The core of the AP is the control algorithm devoted to the estimation of the optimal insulin therapy. One of the most promising algorithm, widely used in this kind of application, is the MPC [135–141]. This algorithm forecasts future BG levels exploiting a patient model in order to estimate the optimal quantity of insulin that keeps BG within the safe range ([70-180] mg/dl). Thanks to the creation of the UVA/Padova simulator [142], a metabolic simulator approved by Food and Drug Administration (FDA) as a substitute to animal experiments, the AP's development had a notable acceleration.
Since hypoglycemia and hyperglycemia prevention is one of the key point of an effective diabetes management, to preserve the patients safety, CGM devices, SAP systems and AP can be equipped with Alarm Systems (ASs) that alert the patient of an upcoming critical event. Depending on the method that generates the alert, the ASs can be categorized into two types: threshold detection and prediction-based. Threshold detection alarms are

activated when particular thresholds are crossed, in this case when critical BG levels are reached [143], meaning in this particular application when critical BG levels are reached (70 [mg/dl] for hypoglycemia, 180 [mg/dl] for hyperglycemia); while prediction-based alarms attempt to evaluate the risk beforehand, relying on patient models, which goodness determines the AS performance, since the quality of the predictions plays a key role in the performance of the entire AS [144, 145].

6.1.1 Neural network models for glucose prediction

In recent years, thanks to a growing availability of both in silico and in vivo data [146], different neural network techniques have been used for glucose prediction. For example multilayer perceptron [147], reinforcement learning techniques [148] and above all the deep learning models [149], were object of recent study. Also in this field, the LSTM is one of the most employed deep learning architecture and some interesting results can be found in literature [150–155]. In [150], a population model is developed for both in silico and in vivo patients, to predict future CGM values at different PHs, given only past CGM in input and leaving the use of additional features as future developments. In [151] an LSTM model is trained on CGM signals of different patients and then used to predict future BG values of an unseen new patient. LSTM obtained very good results both with short and long PHs, overcoming other previous methods like feed-forward neural networks [156], autoregressive models [147] and RNN [157].
A model using only past CGM information in input is easier to obtain, since the patients have to simply wear a sensor to collect these data, with respect to models that require other quantities like assumed meals and injected insulin, that have to be recorded manually or acquired automatically by the devices. However, as stated in [158], adding meals and insulin information as inputs helps BG prediction when PH>30 [min]. In [158] the authors also found that meals information is more significant than insulin. According to this result, in [159] a first LSTM network is trained using CGM as unique feature and then, adding the insulin in input, the obtained improvement was only of 1%. Other works present more complex LSTM architectures with multiple inputs. In [154] a Memory-Augmented LSTM with a neural attention module is trained on both in silico and in vivo data, comparing the results obtained from different datasets and with different inputs: only BG; adding meals and insulin; adding time of the day. A stacked LSTM is presented in [152], having two different input channels: one with past CGM values, meals and insulin quantities and the second one with estimated future meals and expected insulin therapy. Different PHs, number of layers and hidden units in each layer were analysed and the population models were trained on in silico patients and then tested on a real patient. A similar analysis is performed in [153], where meal, insulin and past CGM values

are used for CGM forecasting, with a particularly interesting analysis on different PHs and their clinical implications. In [155] a two layers stacked LSTM is proposed, having in input past CGM, meal, insulin and step count information. Real data are considered and so CGM is preprocessed using Kalman filtering to correct inaccurate readings due to sensor fault.

6.1.2 Contribution

All these considerations suggest that NNs can be a good approach to the BG prediction problem. However, taking into account the inter-patient and intra-patient variability that characterize this system, i.e. that each patient is different from another and that their behavior changes over the time, population and time-invariant models can result less effective: personalized and time-variant models represent a more promising choice for this type of applications.
At first, a case study on one single in silico patient is carried out, giving very good results and presented in [17]. In view of this, Personalized LSTM models (P-LSTMs) are developed in this thesis and will be presented in Section 6.3. One LSTM for each of the 100 in silico adult patients of the UVA/Padova simulator [142] is trained, using current meals and insulin quantities together with past CGM values to predict current CGM values with a PH of 40 minutes, exploiting a single layer LSTM. A different network is trained for each patient, using the same set of hyperparameters for the entire population, and very satisfying results have been achieved, also comparing them with the state of the art. These results have been published in [18]. A preliminary analysis where the P-LSTMs are employed to design personalized ASs for hypoglycemia and hyperglycemia detection is presented in [19].
The P-LSTMs are then improved to achieve better performance and meet some accuracy requirements. The so obtained Enhanced Personalized LSTM models (EP-LSTMs) are proposed in Section 6.4, together with the personalized ASs for hypoglycemia and hyperglycemia detection, showing good predictive performances. These results are presented in [20].
Lastly, the work is extended to the in vivo data of the OhioT1DM Dataset [160], to prove the robustness of the methodology. The choice of this particular dataset is made because it was created for ML applications in the diabetes field, making it one of the most employed dataset for this purpose. A first analysis on the in vivo data is introduced in Section 6.5 for the use of the predictions in an hyperglycemia AS.

6.2 Data

In order to carry out the development of a LSTM model for glucose prediction, both in silico and in vivo data have been used.
In silico datasets have been generated through the UVA/Padova simulator

Dataset	Meals	CHO (g)	Time
tr-dataset (8 days of data)	Breakfast	59 ± 22	7:08 ± 81
	Lunch	63 ± 11	12:45 ± 60
	Dinner	60 ± 17	20:26 ± 56
	Snacks	25 ± 7	
v-dataset (4 days of data)	Breakfast	50 [50-52.5]	7:30 ± 24
	Lunch	66 ± 9	13:22 ± 29
	Dinner	72 ± 6	20:15 ± 52
	Snacks	23 ± 7	
ts-dataset (4 days of data)	Breakfast	57 ± 6	7:22 ± 75
	Lunch	65 ± 4	13:15 ± 39
	Dinner	86 ± 12	19:52 ± 29
	Snacks	19 ± 4	

Table 6.1: In silico datasets descriptions.

[142], a simulator approved by the FDA and used to test insulin therapies in silico, as an alternative to animal experiments, being demonstrated that its simulations reflect the glucose patterns observed in human studies on T1D patients [161]. A detailed description of the simulator is reported in Appendix B, together with an in-depth analysis of some settings that have been employed for data generation.
The in vivo dataset is the OhioT1DM Dataset [160], publicly released for research purposes by the Ohio University, being one of the most employed for glucose prediction models. Both datasets are deeply described in the following.

6.2.1 In silico datasets

In this work, the 100 adult subjects of the most advanced version of the UVA/Padova simulator are considered for data generation.
The datasets include CGM, insulin and meals; the insulin quantity is computed by the MPC described in [162]. Since the final goal is the correct detection of hypoglycemia and hyperglycemia events, in order to have a significant amount of critical episodes in all the datasets, the MPC is tuned to be sub-optimal, as described in Appendix B. In this way the glucose dynamic of patients with regulation problems that will mainly benefit of a personalized model can be reproduced.
Three scenarios are employed to generate three different datasets: a training dataset (*tr-dataset*), used to train the network; a validation dataset (*v-dataset*) for the hyperparameters tuning; a testing dataset (*ts-dataset*), used to assess the network's performance. All the datasets include three meals

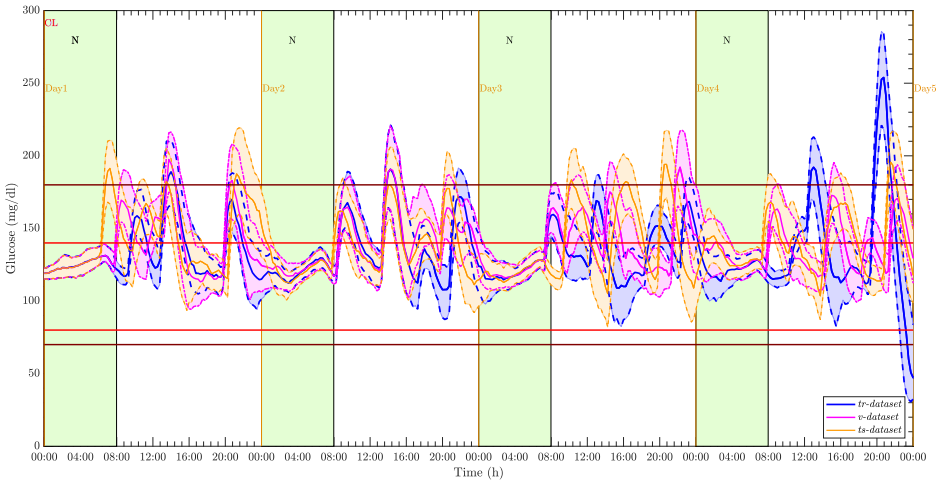
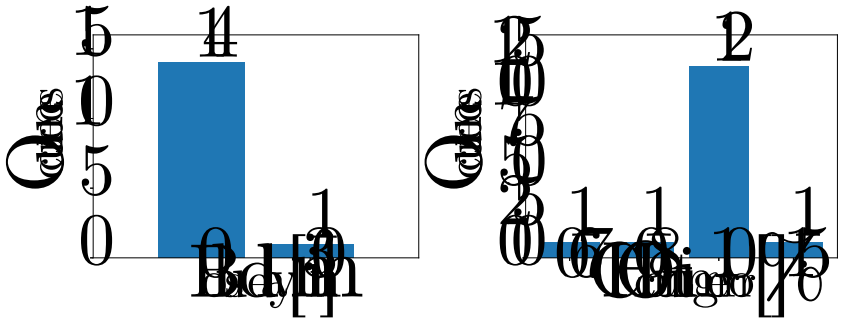
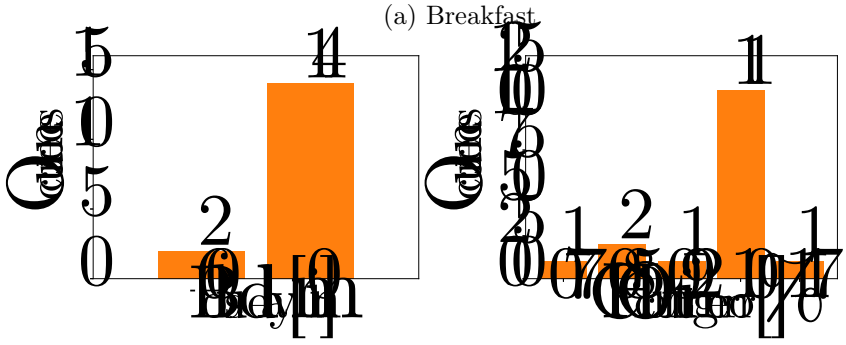


Figure 6.2: Blood glucose profiles of 100 in silico patients with *tr-dataset* (blue), *v-dataset* (magenta) and *ts-dataset* (orange).

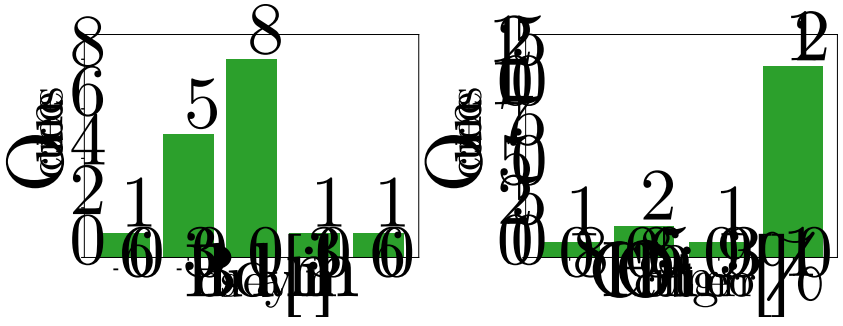
per day and up to two additional snacks, in different quantities and times of the day. The number of meals per day and the time distribution are inspired by the ones observed on the real patients of the Padova center [163]. To ensure that the datasets are uncorrelated, there are no repetitions in the days that compose them. A complete description of the meal distribution is presented in Table 6.1: times and quantities of the meals in each dataset are reported in terms of mean ($\pm$ SD) or median [25^{th} - 75^{th}] percentiles, if the data are normally or not normally distributed, respectively. Note that, since the snacks can be assumed throughout the day, only their quantities are reported, seen their time distribution is not significant.
In view of this meal variability, the *tr-dataset*, *v-dataset* and *ts-dataset* present significant differences both from one day to another and between the datasets as shown in Figure 6.2. In the figure the median [25^{th} - 75^{th}] percentiles of the glucose profiles of the entire population are reported for all the 3 datasets, since they are not normally distributed. To provide more realistic settings during the data generation phase, the meal announcement presents inaccuracies in terms of amount (CHO) and time of the carbohydrate intake. In Figure 6.3 the distributions of the bolus delays and the CHO counting errors introduced in all the datasets are reported for each type of meal. Note that these inaccuracies in the meal management, as well as making the scenario more realistic, also increment the number of critical episodes due to a not optimal insulin therapy.



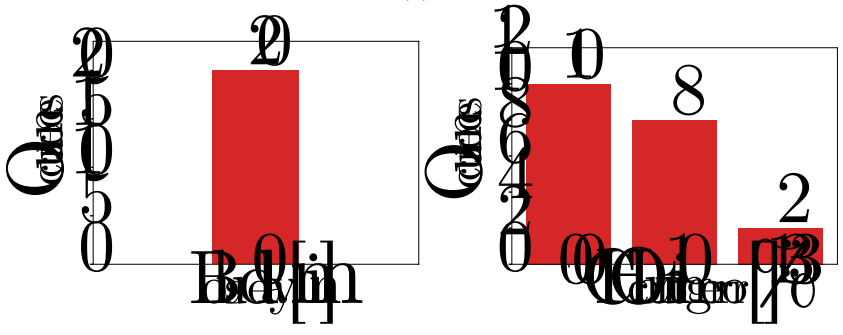
(a) Breakfast



(b) Lunch



(c) Dinner



(d) Snack

Figure 6.3: Histograms representing the time delay and the CHO counting error of the insulin boluses delivered to compensate different type of meals.

	Time	CHO (g)	Insulin Bolus
Day 1	09:00	40	Bolus on time
	12:30	70	Bolus on time for 60 g
	19:30	60	Bolus at 19:00
Day 2	08:00	50	Bolus at 7:30
	13:00	80	Bolus on time
	17:00	30	Bolus on time for 40 g
	20:30	60	Bolus on time
Day 3	07:00	45	Bolus at 7:30
	12:00	60	Bolus on time for 70 g
	18:00	25	No bolus
	22:00	55	Bolus on time
Day 4	10:00	20	Bolus on time
	12:00	65	Bolus on time for 60 g
	19:00	90	Bolus at 19:30
Day 5	07:00	40	Bolus on time for 60 g
	12:30	65	Bolus on time
	20:00	70	Bolus at 19:30
Day 6	01:00	25	Bolus on time
	08:00	60	Bolus on time for 40 g
	13:00	60	Bolus on time
	20:30	55	Bolus at 21:00
Day 7	01:00	15	Bolus on time for 20 g
	06:00	100	Bolus on time for 80 g
	15:00	60	Bolus on time
	21:00	30	Bolus on time
Day 8	05:00	75	Bolus on time
	12:00	40	Bolus on time
	17:00	35	No bolus
	21:00	60	Bolus at 20:00

Table 6.2: Training Scenario.

	Time	CHO (g)	Insulin Bolus
Day 1	07:30	55	Bolus on time
	11:00	25	No bolus
	13:00	65	Bolus on time
	19:30	75	Bolus on time for 70 g
	22:30	15	Bolus on time
Day 2	08:00	50	Bolus on time
	13:00	80	Bolus on time
	16:00	30	No bolus
	19:30	70	Bolus at 19:00 for 60 g
	22:00	15	No bolus
Day 3	07:00	50	Bolus on time
	09:30	25	No bolus
	13:30	60	Bolus on time
	16:00	25	Bolus on time
	21:00	80	Bolus at 20:30 for 70 g
	23:30	15	No bolus
Day 4	07:30	50	Bolus on time
	14:00	60	Bolus on time
	18:00	30	Bolus on time
	21:00	65	Bolus on time

Table 6.3: Validation Scenario.

Scenarios

The scenarios used to generate the *tr-dataset*, *v-dataset* and *ts-dataset* are presented in detail in Tables 6.2-6.4. The number of meals per day and the time distribution are inspired by real data distribution [163] and to provide more realistic settings during the data generation phase, the meal announcement presents inaccuracies in terms of amount (CHO) and time of the carbohydrate intake. For each dataset, time and quantities of meals are reported in the tables for each day in the first columns, then the last one reports if the insulin bolus is delivered at the same time of the meal and if the meal quantity used for its computation is correct or not.

Data preprocessing

The glucose concentration measured by CGM sensors can be affected by sensor measurement noise and calibration errors typical of the devices used to measure the glucose subcutaneously. In order to minimize the effect of these problems, the data used to train the LSTM network are prepro-

	Time	CHO (g)	Insulin Bolus
Day 1	06:00	60	Bolus on time
	09:30	25	No bolus
	12:30	65	Bolus on time
	15:30	15	Bolus on time
	20:00	100	Bolus at 19:30 for 80 g
Day 2	07:30	50	Bolus on time
	13:00	70	Bolus on time
	16:00	20	No bolus
	19:30	70	Bolus on time
Day 3	09:00	65	Bolus on time
	11:00	20	Bolus on time
	14:00	65	Bolus at 13:00 for 50 g
	19:30	90	Bolus on time
	23:30	20	No bolus
Day 4	07:00	55	Bolus on time
	10:00	20	Bolus on time
	13:30	60	Bolus at 12:30 for 50 g
	20:30	85	Bolus on time

Table 6.4: Testing Scenario.

cessed. To have a more accurate signal, a preprocessing algorithm, called "retrofitting" [164], is used. This algorithm reconstructs the BG profile starting from CGM measurements and using sporadic BG values, if available, and Self-Monitoring Blood Glucose (SMBG) measurements, obtained by the patients from finger-stick. In the following, CGM will refer to the CGM after retrofitting. Furthermore, because the involved quantities have different ranges, CGM, meals and insulin quantities are all rescaled using mean normalization. This is a typical approach to improve neural network training performance since unscaled data can cause a slow learning phase, non-convergence, or exploding gradient problems, particularly when working with RNNs. Lastly, the output of the network is filtered through robust quadratic regression to eliminate the prediction's noise.

6.2.2 In vivo dataset

The OhioT1DM dataset [160], made publicly available by the Ohio University to support the research in the field of BG prediction, is the first dataset containing CGM, insulin, physiological sensor, and self-reported life-event data, to be publicly and freely accessible. It is one of the most employed datasets in this field, especially for ML applications, making the comparison

ID	Gender	Age	Pump Model	Sensor Band	Cohort
540	Male	20-40	630G	Empatica	2020
544	Male	40-60	530G	Empatica	2020
552	Male	20-40	630G	Empatica	2020
567	Female	20-40	630G	Empatica	2020
584	Male	40-60	530G	Empatica	2020
596	Male	60-80	530G	Empatica	2020
559	Female	40-60	530G	Basis	2018
563	Male	40-60	530G	Basis	2018
570	Male	40-60	530G	Basis	2018
575	Female	40-60	530G	Basis	2018
588	Female	40-60	530G	Basis	2018
591	Female	40-60	530G	Basis	2018

Table 6.5: OhioT1DM Dataset description.

with the state of the art easy and reliable. It can be obtained only for research purposes through a data use agreement between the Ohio University and researcher institution.
To collect data, the Ohio University run five clinical trials, involving more than 50 patients affected by T1D, subject to the same insulin therapy. The dataset contains eight weeks data of 12 different patients using insulin pump and CGM sensors, released in two different moments. The data of the first 6 patients were released after the first Blood Glucose Level Prediction Challenge (BGLP Challenge) held during the 3rd International Workshop on Knowledge Discovery in Healthcare Data, at IJCAI-ECAI 2018, in Stockholm. The participants of the challenge had to develop an algorithm for glucose prediction, based on the first cohort of real patient data. A second edition of the BGLP Challenge was held at the 5th International Workshop on Knowledge Discovery in Healthcare Data, at ECAI 2020, in Spain, leading to the release of the second cohort of 6 patients.
The dataset contains CGM, SMBG, insulin (bolus and basal), meals with carbohydrates quantity, exercise, sleep data and various life-event data collected from a fitness band, like heart rate, skin conductance, skin temperature, air temperature, step count and others. The patients wore Medtronic 530G or 630G insulin pump, Medtronic Enlite CGM sensor, Basis Peak fitness band for the 2018 cohort and Empatica Embrace fitness band for the 2020 one. Moreover, in order to ensure privacy of the participants in the clinical trials, the data have been de-identified: a random ID number is assigned to each patient and the data are randomly shifted in the future. Some information about the different contributors are listed in Table 6.5: each patient in the dataset is described with its ID, gender, age range, pump model, sensor band

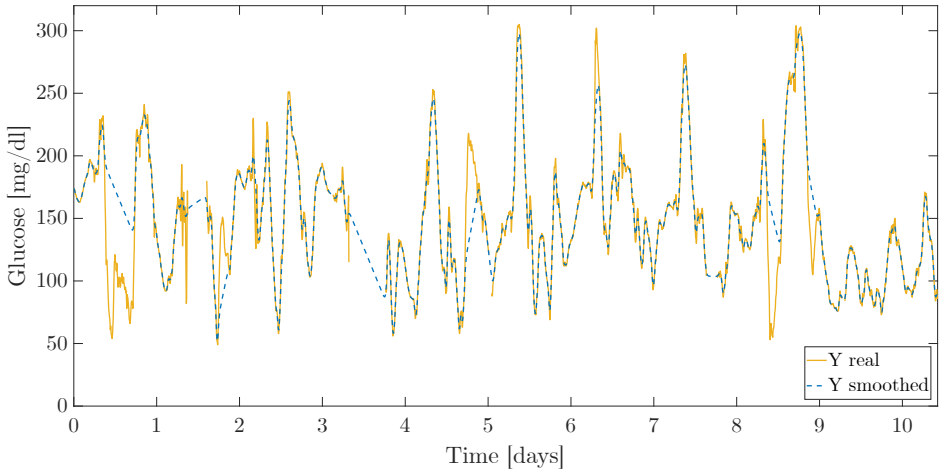


Figure 6.4: CGM Kalman filtering and smoothing for a patient of the OhioT1DM dataset.

and the cohort to which it belongs. The dataset contains for each patient a XML file for training and one for testing; in the 2020 cohort the first hour of the test dataset is not considered to be unbiased and more distant from the training dataset. In this work, the training dataset is further divided in training and validation dataset, considering 30 days of the original training dataset to train the network and the remaining for validation.

Data preprocessing

It was not possible to use the retrofitting algorithm described in Section 6.2.1 to preprocess the OhioT1DM Dataset because, in addition to the sensor measurement noises present in the in silico CGM signal, the in vivo data present some missing CGM data. So in this case the Kalman filtering and smoothing technique proposed in [165] is used to filter and smooth the CGM data, removing noise and interpolating in case of missing data, exploiting also SMBG data. The MATLAB implementation of the technique is publicly available [166].
This algorithm firstly employs Kalman filtering at each time step to compute a state estimation and a state covariance matrix, and uses a dynamic model to predict the state of the subsequent time steps. In this way an a priori estimate of the state is generated. Then it is updated considering measurement noises, obtaining an a posteriori estimate. If a measure is not available in a certain time step, the a posteriori estimate is set equal to the a priori one. The estimates can be further improved through Kalman smoothing: a backward pass computes the smoothed estimates, having in input the a priori and a posteriori estimates from the Kalman filtering. The Rauch-Tung-Striebel

algorithm [167] is used to do this. The mathematical implementation of this algorithm is described in Appendix C.
To obtain the state predictions, a dynamical model is exploited. In the available MATLAB implementation, it is possible to choose among two different models and in this thesis the second one is chosen: a central-remote rate model, where the glucose dynamic is represented through a compartmental model. The two models are further described in Appendix C.3.
It is also possible to choose the outlier suppression and removal methods since the outlier detection can be based on the filter estimate or on the smoother estimate. Considering that the process noise is high, the first method is able to detect only major outlier and so it is chosen in this work because it is more conservative of the original data. The second method removes several hypoglycemia or hyperglycemia peaks that were considered outliers, while it is important to keep them in order to build the AS.
Considering these settings, the CGM signals of all the 12 patients are pretreated. In Figure 6.4 the original signal, in yellow, is compared with the filtered and smoothed one, in blue, representing the testing dataset of patient 563. It is interesting to notice how the used technique imputes missing data, like it can be observed in day 3. However, this technique presents also some disadvantages because sometimes the outlier suppression is too aggressive, like for example during the first day where too many data are removed. In a further development of this work an analysis of this undesired behavior is needed, together with an improvement of the technique.

6.3 P-LSTMs training procedure

A first model, called Personalized LSTM (P-LSTM) is trained, using the in silico datasets described in Section 6.2.1. One different P-LSTM is trained for each of the 100 adult patients of the UVA/Padova simulator.
The considered network architecture is a single layer LSTM with 96 neurons ($n_c = 96$), three inputs (n_x=3) and one output. The inputs are the injected insulin through a subcutaneous insulin pump, $I(t)$, the carbohydrate intake provided by the patient, $M(t)$, and the past CGM values of PH minutes ago, $CGM(t - PH)$. The output is the glucose value obtained through CGM at the current time, $CGM(t)$. The training is performed considering Adam optimizer to perform the backpropagation algorithm, with learning rate $\alpha = 0.01$, number of epochs fixed at 500 and MSE (Equation 4.19 in Section 4.4) chosen as loss function to be minimized. The training procedure has been implemented using TensorFlow [108] and Keras API [109]. The hyperparameters of the network, such as n_c and α, have been optimized using KerasTuner [110], where the optimization criterion is the minimization of the loss computed using the *v-dataset*. The optimization was performed on a subset of the patients and then the results have been extended to the entire

population. The description of these software is presented in Appendix A.
In order to define the PH, the results presented in [153] are considered, where several LSTM networks have been trained to predict future glucose values, considering the same inputs and output used in this work. The authors compared the results obtained varying n_c, number of LSTM layers, input dimension and PHs. In particular, the authors found that the networks with small PH obtain better performances, but an anticipation of at least 30 minutes are needed in a clinical scenario to allow the patients to react and avoid hypoglycemia. The best trade-off between performances and clinical needs is obtained with a PH of 30 [min], even if a PH of 45 [min] has performances still acceptable. Considering that models with these PHs were good enough to be used in clinical practice, in the following PH is set equal to 40 [min].

6.3.1 Results

The 100 P-LSTM models are evaluated through several performance indexes, in order to facilitate the comparison with the literature.
In addition to the FIT, already defined in Equation 3.20 in Section 3.6.1, also the Root Mean Squared Error ($RMSE$) and two additional indexes specific of this application, Downward Delay (DD) and Upward Delay (UD), are considered. Being y the real data, $\hat{y}$ the predicted data and S the number of samples, they are calculated as:

$$RMSE = \sqrt{\frac{1}{S}\sum_{i=1}^{S}(y_i - \hat{y}_i)^2} \tag{6.1}$$

$$\begin{aligned} DD = \underset{j\in[0,PH]}{\arg\min}\left[\frac{1}{S}(\hat{y}(t|t-PH+j) - y(t))^2\right] \\ \forall t \in [t_P, t_{N_{75}}] \end{aligned} \tag{6.2}$$

$$\begin{aligned} UD = \underset{j\in[0,PH]}{\arg\min}\left[\frac{1}{S}(\hat{y}(t|t-PH+j) - y(t))^2\right] \\ \forall t \in [t_N, t_{P_{75}}] \end{aligned} \tag{6.3}$$

The $RMSE$ quantifies the variance of the prediction error; the smaller it is, the better is the prediction. DD and UD estimate the delay of the prediction with respect to the real data [168, 169]. In the above equations DD and UD are computed as the time shifts that minimize the mean squared error between real and predicted data [168]. As observed in [169], including Peaks (P) and Nadirs (N) in the computation would give a too pessimistic result, since in these particular cases the delays are bigger. So the delays are calculated only on positive and negative trends, identifying the time instant at which certain thresholds are crossed ($t_N, t_P, t_{N_{75}}, t_{P_{75}}$). For positive trends the threshold (P_{75}) is placed at the 75% of the nadir-to-peak distance,

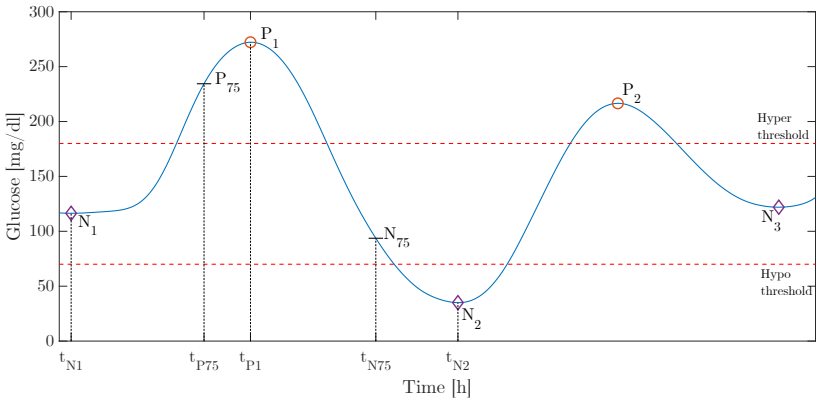


Figure 6.5: Example of glucose profile where nadirs, peaks and the respective thresholds are highlighted.

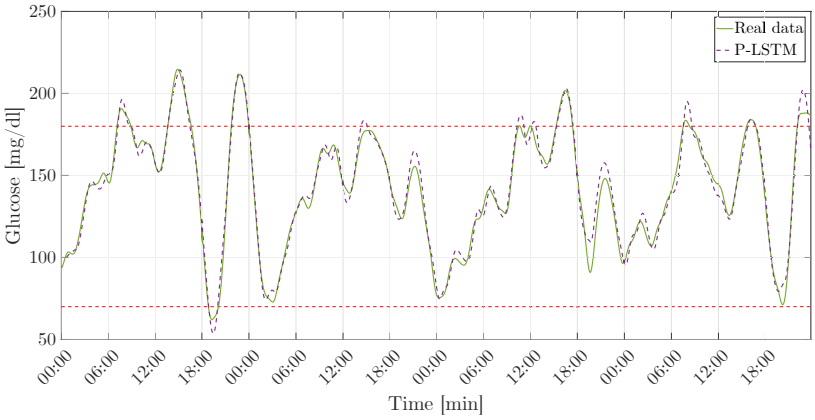


Figure 6.6: Glucose profile of an in silico subject with P-LSTM model.

for negative ones the threshold (N_{75}) is placed at the 75% of the peak-to-nadir distance. For clarity, an example is shown in Figure 6.5, where peaks are pointed with red circles, nadirs with purple diamonds and the above-mentioned thresholds are shown. Red dashed lines show the hypoglycemia and hyperglycemia thresholds. DD is considered on negative trends and UD on positive ones, calculated as the average delays of the total thresholds crossings.

All the performance indexes are computed for each patient independently, then the mean ($\pm$ SD) or the median [$25^{th} - 75^{th}$ percentiles] is reported for the entire population if the results are normally or not normally distributed, respectively. The 100 P-LSTM models, evaluated on the *ts-dataset*, obtained a $RMSE$ of 7.67 [6.44-9.07] and a FIT of 75.86 [70.52-79.57]. These results are satisfying considering the particularly challenging application. The glucose profiles are predicted with a DD of 9 [8-12] and a UD of 9 ($\pm$3) minutes,

Model	RMSE	FIT	PH	# subj	Data
P-LSTM	7.67	75.86	40	100	Silico
Sun [150]	30.21	42.56	45	20	Silico/ Vivo
Aiello [152]	11.68 31.01	58.84 41.41	[5, ..., 60]	100 1	Silico Vivo
Carrillo [153]	20.76	-	45	8	Vivo
Mirshekarian [154]	2.93 18.07	- -	30	30 6	Silico Vivo
Aliberti [151]	7.18	88.79	45	451	Vivo
Rabby [155]	5.89 18.96	- -	30	6	Vivo

Model	LSTM Architecture	Inputs
P-LSTM	Single Layer LSTM	CGM, M, I
Sun [150]	LSTM + Bi-directional LSTM	CGM
Aiello [152]	2 branches of stacked LSTMs	CGM, M, I
Carrillo [153]	Stacked LSTMs	CGM, M, I
Mirshekarian [154]	Memory-Augmented LSTM	CGM, M, I
Aliberti [151]	Single Layer LSTM	CGM
Rabby [155]	Stacked LSTMs w/ filtered CGM Single Layer LSTM	CGM, M, I, steps

Table 6.6: P-LSTM results and comparison with the state-of-the-art.

that make the models useful for applications like alarms or model-based AP that require to predict specific critical events in advance.
A graphical example of the glucose profile of one patient is reported in Figure 6.6, where it can be noticed that the trend is followed by the P-LSTM simulation and the peaks are well tracked: in few cases the peaks are overestimated, but the hypoglycemia and hyperglycemia events are present also in the real case.

6.3.2 Comparison with the state-of-the-art

In order to evaluate the goodness of the P-LSTMs results, a comparison with the most relevant works in the literature is reported in Table 6.6. These works present different characteristics but they are the most similar results

found in literature to have an estimation of the goodness of the proposed models. In Table 6.6, the performance indexes, the considered PH, number of subjects, the nature of the data (silico or vivo), a brief description of the LSTM architecture and the used inputs are reported.
The P-LSTMs outperform [150, 152, 153], where population models were tested on in vivo data. It is interesting to notice how a simpler architecture is capable to understand better the glucose dynamics. It is also shown how personalized models are more effective then population ones. A better result is obtained in [154], on in silico data using a previous version of the UVA/Padova simulator [170]. In this version of the simulator the intra-day variability was not modeled and in [154] fewer patients are considered with a smaller PH and a more complex architecture. Our limited loss in the performance is a good trade-off considering the simpler architecture and the more challenging population. However, an interesting result obtained in this work is the robustness of the LSTM to noise and sensors data loss, also thanks to the implementation of an attention mechanism. Apart [151] and [152] all the other works considered less patients. Lastly, the result is comparable with [151], where a very large dataset with 451 real patients is used and [155], where single and stacked LSTMs with Kalman filtering or without are used on the OhioT1DM Dataset [160]. In this latter, considering the result with only single layer LSTM, similarly to this work, the obtained results were worsened, while with a more complex architecture the improvement is negligible.

6.4 EP-LSTM based Alarm System

6.4.1 EP-LSTM description

Since the P-LSTM models obtained satisfying results for glucose prediction, these models can be successfully employed to design personalized ASs. However, for this kind of application the models must meet some performance requirements. In particular, the delay with respect to the original signal has to be limited in order to catch the glucose dynamic. In this prospective, a criterion to improve the P-LSTM quality is proposed. Each P-LSTM is firstly evaluated on the *v-dataset*, if $DD \geq 10$ [min] or $UD \geq 10$ [min], the prediction could not be good enough to be used in the alarm system and an ad-hoc hyperparameters tuning is performed via KerasTuner on the specific patient. These new models are called Enhanced Personalized LSTM models (EP-LSTMs).

6.4.2 Alarm system description

ASs are powerful tools that can be used to avoid harmful situations like hypoglycemia and hyperglycemia for subjects with T1D. The ASs described

Dataset	Meals	CHO (g)	Time
AS-dataset (28 days of data)	Breakfast	55 [50-60]	7:30 [7:00-7:45]
	Lunch	65 [60-65]	13:00 [12:30-13:45]
	Dinner	68 ± 12	19:30 [19:30-20:00]
	Snacks	20 [15-30]	

Table 6.7: *AS-dataset* description.

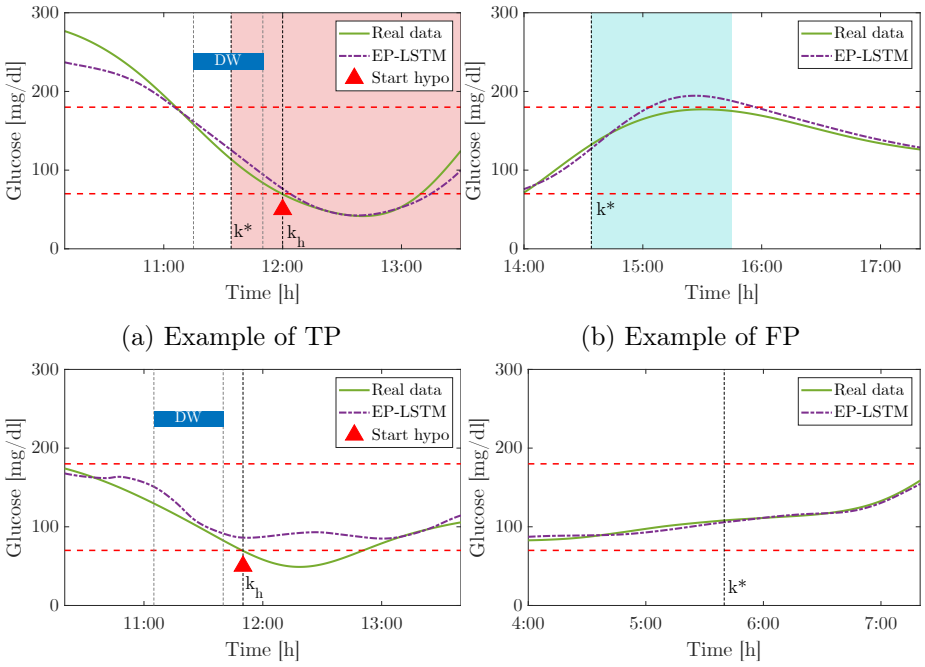
in the following are prediction-based alarms.
The AS considers the EP-LSTM predictions at time k^* over a Prediction Window (PW) from $k^* + 1$ to $k^* + PW$: if the predicted BG levels exceeds the threshold G_{hypo} (or G_{hyper}) for at least S minutes, an hypoglycemia (or hyperglycemia) alarm is raised. In this way the patient is alerted of an incoming critical event and can take some action to avoid it. Note that $PW \leq PH$ since the EP-LSTMs require CGM in the previous PH minutes to predict future CGM, so it is chosen to exploit the longest PH which the EP-LSTM was trained for.

6.4.3 Data generation for AS evaluation

In order to test the AS performances in silico, a new dataset is generated, called *AS-dataset*. *AS-dataset* contains three meals per day with the possibility to have snacks, uncertainties in the meal announcement are present accordingly to the other datasets. There are not repetitions of the days that are uncorrelated with the scenarios of *tr-dataset*, *v-dataset* and *ts-dataset* too. *AS-dataset* is described in detail in Table 6.7. Since a relevant amount of hypoglycemia and hyperglycemia events distributed over this time period is needed to effectively test the AS performances, *AS-dataset* is much longer than the other datasets used so far and the MPC used in the UVA/Padova simulator is tuned accordingly to the training dataset, as described in Appendix B.

6.4.4 Performance metrics for alarm systems

A preliminary definition of the events that can occur during event prediction is needed to describe the performance metrics for the evaluation of the alarm goodness. Firstly, k_h is defined as the time instant at which a generic hypoglycemia or hyperglycemia event occurs; DW is the Detection Window determined by the event, starting DW_s (Detection Window start) and ending DW_e (Detection Window end) minutes before the considered time, with $DW_s > DW_e > 0$ to guarantee the alarm diagnostic, where DW_e is the minimum interval to detect the event; k^* the time instant at which a generic hypoglycemia or hyperglycemia alarm is raised.



(a) Example of TP

(b) Example of FP

(c) Example of FN

(d) Example of TN

Figure 6.7: Example of TP, FP, FN and TN events.

These events are shown in Figure 6.7 where the real data are in green, prediction through EP-LSTM in dotted purple, red triangle points the start of an hypoglycemia event and its detection window. The alarm activation is represented by the light-red region in the hypoglycemia case and by light-blue region in the hyperglycemia case.
The following events are valid both for hypoglycemia and hyperglycemia detection:

- True Positive (TP): at time k_h an event occurs and an alarm is raised in the Detection Window, $DW = [k_h - DW_s, k_h - DW_e]$, as shown in Figure 6.7a for hypoglycemia. In practice, the alarm must be activated at least DW_e minutes before the event to allow the patient to prevent it and also not too distant in time, at most DW_s, to be considered related to the current event.

- False Positive (FP): at time k^* an alarm is raised but there is no event in $[k^*, k^* + DW_s]$, as shown in Figure 6.7b for hyperglycemia. It is important to notice that if an alarm is raised too late, after DW_e, it is not considered a FP event even if it is not a TP. In fact, it is a correct detection of the event (TP) that is not raised enough in advance to take an action to avoid hyper/hypoglycemia. In this prospect, those events are not taken into account in the results.

- False Negative (FN): at time k_h there is an event but the alarm is not activated in the DW, as in Figure 6.7c for hypoglycemia.
- True Negative (TN): at time k^* the alarm is not activated and there is no event in the interval $[k^*, k^* + (DW_s - DW_e)]$, as in Figure 6.7d.

Considering these events, the following metrics to evaluate the AS performances are defined:

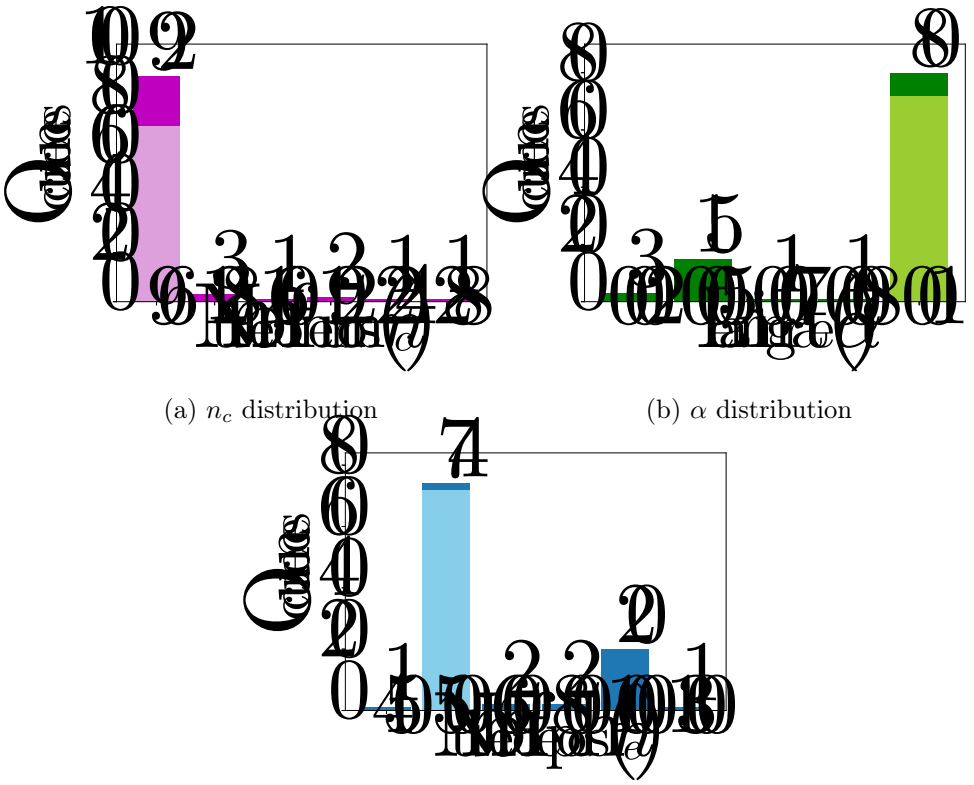
- True Positive Rate (TPR), also called recall or sensitivity, measures how many positive events are properly detected over the total amount of events:
$$TPR = \frac{TP}{TP + FN} \tag{6.4}$$
- Positive Predicted Value (PPV), or precision, measures how many alarms are correctly activated over the total amount of alarms rising:
$$PPV = \frac{TP}{TP + FP} \tag{6.5}$$
- F1 score ($F1$), the harmonic mean of TPR and PPV:
$$F1 = \frac{2 * TPR * PPV}{TPR + PPV} \tag{6.6}$$

It should be noted that only the metrics related to the positive events are here considered because the dataset is very unbalanced, having few hypoglycemia and hyperglycemia events with respect to the overall amount of data in the safe range. In fact in this case the number of TN events is definitely higher than the others and all the metrics including it would be saturated and misleading. This is proven in [171], showing that for unbalanced sets precision and recall are more informative than ROC plots, where TPR is compared with its counterpart, False Positive Rate ($FPR = FP/(FP + TN)$).

6.4.5 Results

EP-LSTM results

A computer equipped with graphics processing unit (GPU) was used to carry out the training processes: an Intel i7-7700HQ CPU with 2.80 GHz, 32.0 GB memory with GPU NVIDIA GeForce GTX 1050 was used in the experiments. The program was written in Python 3.6, using CUDA 9.0.
Among the 100 patients of the UVA/Padova simulator, in 28 cases the P-LSTMs do not satisfied the conditions related to DD or UD on the *v-dataset*. So, a specific hyperparameters tuning is performed for these problematic patients via KerasTuner. The total distributions of the hyperparameters are



(a) n_c distribution

(b) α distribution

(c) n_e distribution

Figure 6.8: Hyperparameters distributions in the 100 in silico patients.

presented in Figure 6.8, where three bar plots show the occurrences of the values for each hyperparameter, where the lighter colors represent the 72 unvaried cases and the darker colors represent the retuned hyperparameters: n_c in Figure 6.8a, α in Figure 6.8b and n_e in Figure 6.8c. It can be noticed that n_c is the hyperparameter with less impact, 20 patients out of 28 have $n_c = 96$, that is the original value used in the P-LSTM, then only the other 8 needed an higher number of neurons. For what concern the learning rate, 15 patients out of 28 need a smaller $\alpha = 0.005$, then 8 keep $\alpha = 0.01$ and the other 5 have different values. The distribution of n_e is even diverse, only 3 patients have $n_e \leq 500$, while the others need an higher number of epochs, with a peak of 20 patients that need 1000 epochs.
After the retuning of the hyperparameters, evaluating the delays on this subset of the patients, significant improvements can be noticed: on the *v-dataset* DD improved from 11 [min] to 8.84 [min] ($p < 0.01$) and UD from 13.03 [min] to 9.78 [min] ($p < 0.001$); on the *ts-dataset* DD improved from 11.54 [min] to 10.17 [min] ($p < 0.05$) and UD from 12.85 [min] to 10.37 [min] ($p < 0.001$). The p-values are computed with the appropriate statistical test based on the data distribution characteristics. The gaussianity and ho-

Index	EP-LSTM	P-LSTM
$RMSE$	7.47 [6.02 - 8.69]	7.67 [6.44 - 9.07]
FIT	76.47 [71.16 - 80.47]	75.86 [70.52 - 79.57]
DD	9.02 [7.27 - 11.53]	9.49 [7.81 - 12.12]
UD	8.74 ($\pm$3.04)	9.65 ($\pm$4.15)

Table 6.8: Results of the prediction on the *ts-dataset*, comparing EP-LSTM and P-LSTM.

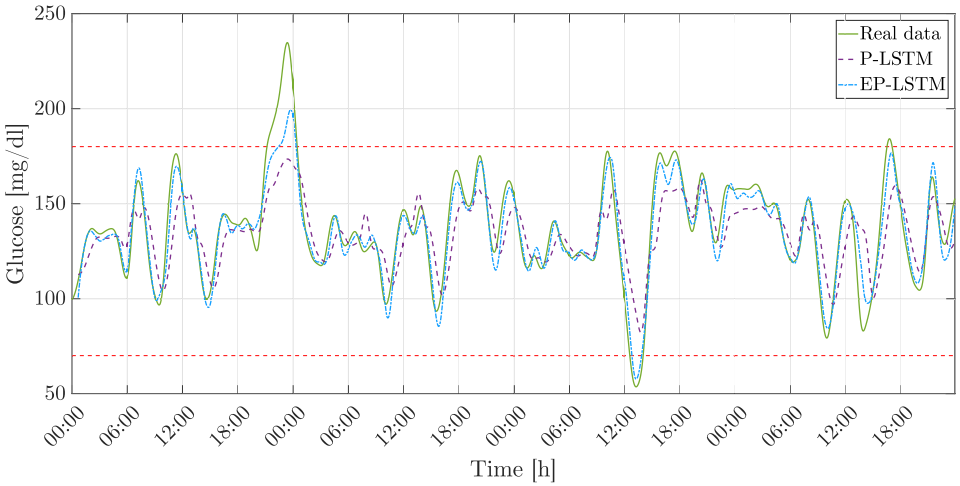


Figure 6.9: Glucose profile of an in silico subject, comparing P-LSTM and EP-LSTM predictions.

moscedasticity of the data distributions are assessed by the Lilliefors test and two-sample F-test, respectively. If at least one distribution is non-Gaussian, the Wilcoxon rank sum test is used to test the significance of the differences; if both distributions are Gaussian and homoscedastic, a two-sample t-test is performed; otherwise, the two-sample t-test with Satterthwaites approximation is used.
The 100 EP-LSTMs are then evaluated on all the patients of the UVA/ Padova simulator through $RMSE$, FIT, DD and UD on the *ts-dataset*. The results are shown in Table 6.8, compared with the P-LSTM ones [18]. Mean ($\pm$ SD) or median [25^{th} - 75^{th} percentiles] are reported for the entire population, if the results are normally or not normally distributed, respectively. The EP-LSTMs obtain an overall improvement with respect to the previous results, especially the two delays are reduced, guaranteeing more reliability of the models for their employment in the alarm system. A graphical example is reported in Figure 6.9 for a patient that required the model hyper-

Index	Hypo Alarm	Hyper Alarm
TPR	80.00 [57.42 - 92.98]	85.71 [66.67 - 97.28]
PPV	85.79 [66.67 - 100]	80.43 [66.67 - 89.61]
F1	75.00 [66.67 - 87.61]	80.00 [66.33 - 90.68]
# episodes	12 ± 9	39 ± 26
# alarms	12 ± 9	40 ± 27

Table 6.9: Hypoglycemia and hyperglycemia alarm systems performances on the in silico dataset.

parameters retuning: P-LSTM and EP-LSTM models are compared on the *ts-dataset*. The real data are represented in green, the P-LSTM prediction in purple dashed, the EP-LSTM prediction in blue dotted. The red dashed lines are hyperglycemia and hypoglycemia thresholds. It can be noticed that the EP-LSTM prediction is more accurate, especially in the hypoglycemia and hyperglycemia regions, and the delays are reduced: for this patient, DD goes from 18.57 [min] to 6.27 [min] and UD from 22.60 [min] to 14.73 [min].

AS results

The AS performances are evaluated on the *AS-dataset* for all the 100 patients of the UVA/Padova simulator. In this work $DW_s = 45$ [min] and $DW_e = 10$ [min], in accordance with previous literature [144, 145], and PW=40 [min], $G_{hypo} = 70$ [mg/dl], $G_{hyper} = 180$ [mg/dl], S=10 [min]. Among the 100 patients, 3 subjects did not present hyperglycemia events in the *AS-dataset* so for the hyperglycemia alarm the results are calculated on 97 patients. The median $[25^{th} - 75^{th}$ percentiles] is then calculated on the entire population since the results are not normally distributed.
The performances of the AS are shown in Table 6.9: the EP-LSTM AS obtained satisfactory performances with the 80% of hypoglycemia events correctly detected ($TPR = 80\%$) over an average number of episodes equal to 12 and with the 86% of hyperglycemia events correctly detected ($TPR = 86\%$) over an average number of episodes equal to 39. The precision of the alarm is also adequate with an 86% of the hypoglycemia alarms raised correctly ($PPV = 86\%$) over an average number of alarms of 12 and an 80% of the hyperglycemia alarms raised correctly ($PPV = 80\%$) over an average number of alarms of 39.
A graphical example of the AS performances is shown in Figure 6.10, where the first four days of the *AS-dataset* on a specific patient are shown. In the middle panel the real data in green and the predictions in dashed purple are represented, together with hypoglycemia and hyperglycemia events indicated by a red triangle and a blue diamond, respectively. The activation of the

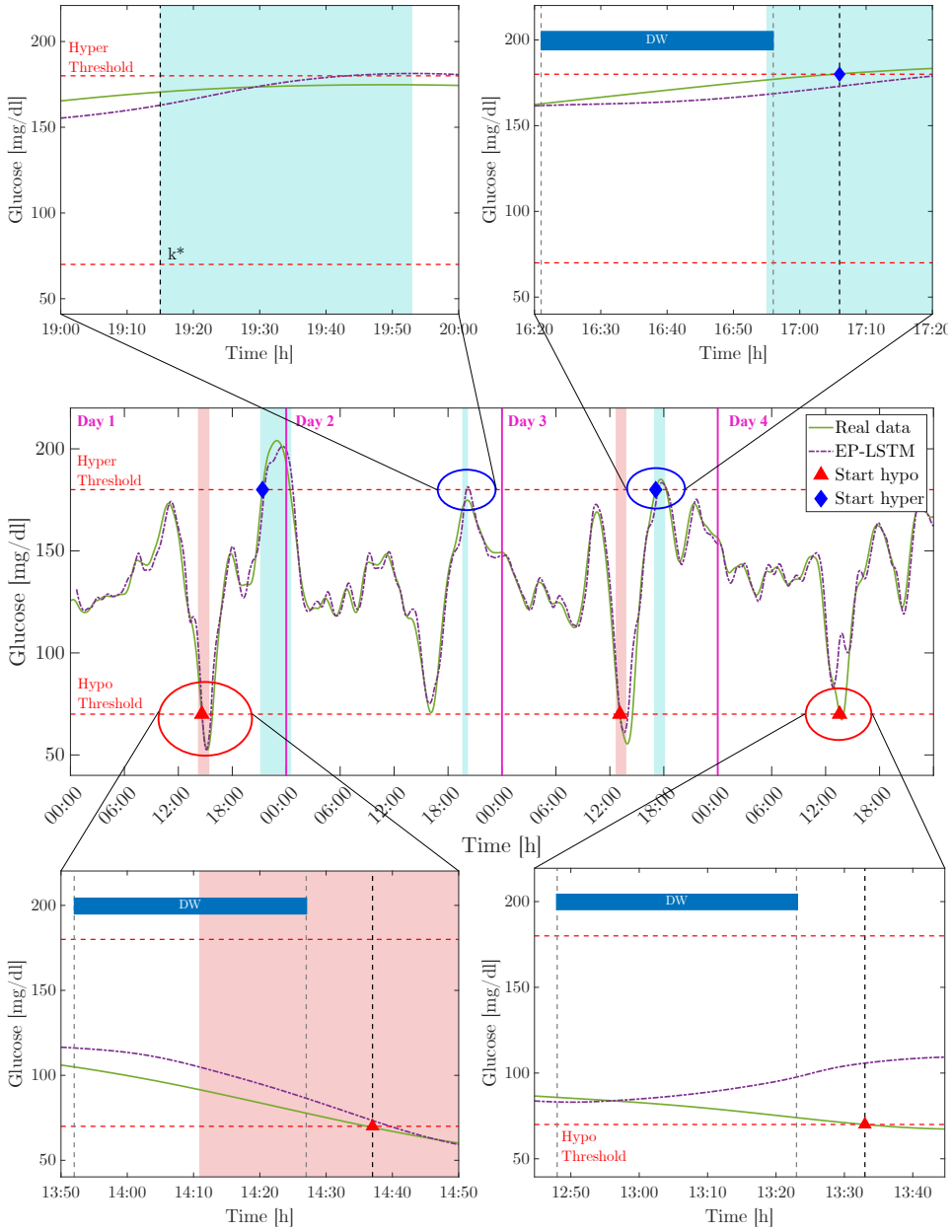


Figure 6.10: Personalized alarm system example. In the middle panel the glucose profile in the first 4 days. In the top panels a *FP* and a *TP* case for hyperglycemia; in the bottom panels a *TP* and a *FN* case for hypoglycemia.

alarms is represented by the light-red regions for hypoglycemia and by the light-blue ones for hyperglycemia. During the four days, three hypoglycemia and two hyperglycemia episodes are present: four of them are correctly detected, while around 14:00 of Day 4 a FN event for hypoglycemia can be noticed and around 19:00 of Day 2 a FP event for hyperglycemia is present. In the bottom panel two zooms in of two hypoglycemia episodes are shown, highlighting how the alarm is correctly raised in the DW on the left (TP) while the event is not detected on the right (FN). In the top panel a wrong hyperglycemia detection is presented on the left (FP) while a corrected one is represented on the right (TP). From two of the zoomed panels, it can be observed that these false events are not severe or harmful for the patient. In fact, the hypoglycemia event that is not detected is not a severe event (BG>60 [mg/dl]), while the hyperglycemia alarm is wrongly raised when the glucose was closer to the hyperglycemia threshold (BG>170 [mg/dl]) as shown in Figure 6.7. In general the results obtained from the proposed AS are really satisfying.

6.5 Ohio EP-LSTM

Given the very good results obtained by the EP-LSTM AS on in silico patients, the methodology is extended to in vivo data. Data directly collected from diabetes patients are more challenging and tricky, subject to noise and possibility of missing data, they need to be analyzed and pretreated, but let to test the effectiveness of the proposed methodology, representing a significant step towards new therapies.
A different network is trained for each of the 12 patients of the OhioT1DM Dataset [160]. The network architecture is kept equal to the one used in the in silico case, as described in Section 6.3: a single layer LSTM having in input $I(t)$, $M(t)$, $CGM(t-PH)$ and $CGM(t)$ in output, with PH of 40 minutes. Then, the goal is to design also in this case an AS for hypoglycemia and hyperglycemia prevention based on the EP-LSTM predictions, keeping the same structure used in the in silico case and described in Section 6.4.2.

6.5.1 Results

EP-LSTM results

The 12 personalized LSTM networks are trained on 30 days for each patient of the OhioT1DM Dataset and then evaluated on around 10 days of testing data. The performance indexes used to evaluate performances of the the networks are also in this case FIT (Equation (3.20) in Section 3.6.1), $RMSE$, DD and UD (Equations (6.1), (6.2) and (6.3) in Section 6.4.5). The results are reported in Table 6.10, where the above mentioned indexes are calculated for each patient. In the Table, the first six patients belong to

ID	$RMSE$	FIT	DD	UD
540	6.77	87.25	25.75	22.07
544	4.46	89.08	16.16	8.89
552	4.87	91.03	17.79	14.95
567	7.15	83.93	18.74	22.54
584	10.31	85.41	29.49	25.20
596	5.60	87.16	20.72	16.95
559	11.17	75.17	33.05	32.51
563	4.16	87.07	15.40	14.77
570	8.12	85.47	26.53	22.97
575	5.47	90.27	15.36	17.79
588	10.73	89.28	16.46	12.85
591	10.19	76.96	34.97	29.64
Av.	7.41 ± 2.61	85.67 ± 4.96	22.23 ± 7.06	20.40 ± 7.16

Table 6.10: Results on the testing data of the OhioT1DM.

the 2020 cohort, the other half the 2018 cohort. In the last row mean ± SD of all the patients is calculated, since the results are normally distributed. The obtained average values of $RMSE = 7.41$ and $FIT = 85.67\%$ are very satisfying and, comparing them with the in silico results in Table 6.8, it can be observed that the $RMSE$ is almost equal while the FIT is even higher in the real case. Another interesting comparison can be made also with some works mentioned in Section 6.3.2 and reported in Table 6.6, in particular [154] and [155], that are based on the 2018 cohort of the OhioT1DM dataset. The result of this thesis outperforms [154], where however the data are not pretreated and missing data are simply not considered. In [155] the same pretreating algorithm used in this thesis is applied, training a two layered stacked LSTM with a prediction horizon of 30 and 60 minutes and having in input also the step count. Considering the results with PH=30, in [155] a mean $RMSE$ of 5.89 is obtained. In this thesis, where a simpler architecture, a longer PH and less inputs are employed, the mean $RMSE$ for the 2018 cohort is 8.31, so even if slightly worse the result is still very satisfying. On the other hand, the DD and UD indexes are for certain high, with average values of 22 and 20 minutes respectively, exceeding the criteria stated in Section 6.4.1 to use the predictions for the design of the AS.
Moreover, a graphical result is reported in Figure 6.11 where the first 4 days of the testing dataset of two patients are shown, with the real data in blue and the prediction in black dashed. In particular patient 552 is shown in Figure 6.11a and patient 559 in Figure 6.11b, being respectively the best and the worst result. It can be observed that the predictions are accurate, presenting a trend similar to the real data. The main difference between the

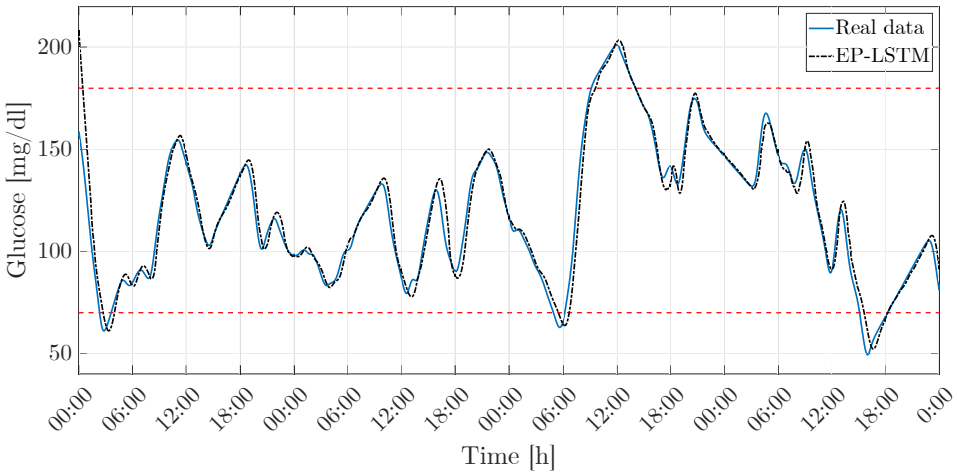
ID	TPR	PPV	$F1$	hyper	alarms
540	20.00	25.00	22.22	5	6
544	77.78	100.00	87.50	9	9
552	50.00	66.67	57.14	4	4
567	33.33	30.77	32.00	12	14
584	0	0	NaN	6	6
596	76.92	76.92	76.92	13	13
559	0	0	NaN	11	11
563	100.00	73.08	84.44	19	20
570	80.00	80.00	80.00	10	10
575	33.33	50.00	40.00	6	7
588	100.00	80.00	88.89	8	9
591	9.09	8.33	8.70	11	11
Av.	48.37 ± 37.53	49.23 ± 35.11	48.15 ± 35.22	9.50 ± 4.17	10.00 ± 4.33

Table 6.11: Hyperglycemia alarm systems performances on the OhioT1DM Dataset.

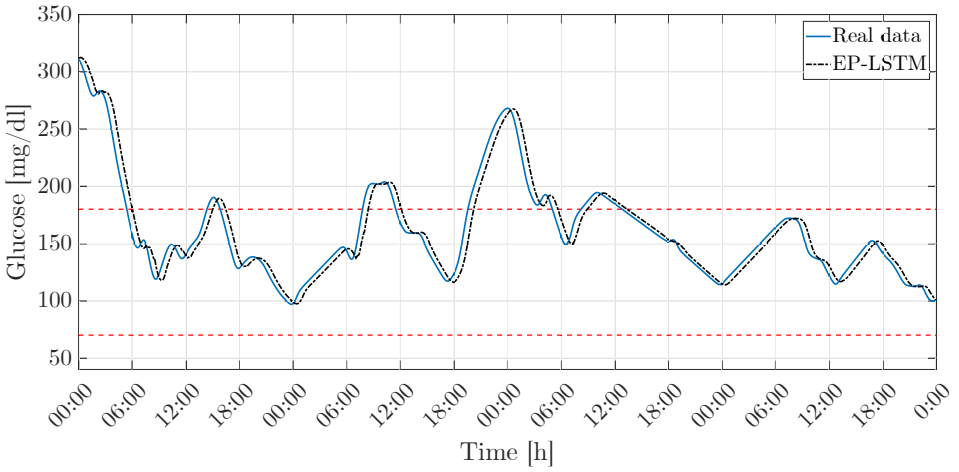
best and the worst case is represented by the prediction delay that can be observed in Figure 6.11b and that is deduced even more by the analysis of the performance indexes in Table 6.10.

AS results

Despite the high values of the delays of the Ohio EP-LSTMs, an attempt to design an AS is made. It is important to notice that the data of the OhioT1DM Dataset are collected from patients subject to a well balanced insulin therapy, while the main target of the in silico models are patients with regulation problems, obtained by tuning accordingly the MPC in the simulator. For this reason, both in the training and the testing data of the OhioT1DM Dataset, few hypoglycemia events are present and so the LSTM model is not able to correctly detect them. Consequently, only an AS for hyperglycemia prevention is presented in the following and evaluated with the performance indexes listed in Section 6.4.4.
The obtained results are very different among the various patients, as can be deduced by the very high values of the standard deviations in all the indexes: TPR of 48±37, PPV of 49±35, $F1$ of 48±35. Comparing the alarm results in Table 6.11 with the prediction results in Table 6.10, it can be observed that the worst alarm performances are those of the patients (584, 559 and 591) that have UD and DD of around 30 minutes. With such delays it is not possible to raise correctly alarms inside the DW, resulting in TPR and



(a) Glucose profile of patient 552



(b) Glucose profile of patient 559

Figure 6.11: LSTM predictions of two patients of the OhioT1DM Dataset.

PPV below 10%, having even 0% in two cases for both indexes. Another group of patients (540, 552, 567, 575) has intermediate results, with values of TPR and PPV between 20% and below 70%. The remaining 5 patients instead (544, 596, 563, 570 and 588) have very good results, with values of TPR and PPV between 70% and 100%. Moreover, it can be noticed that for each patient the number of alarms raised and effective hyperglycemia events is almost equal and that the bad performances depend on the delays of the prediction.
Considering this model as a preliminary study, carried out to test the defined methodology also on real data, it can be affirmed that the results are promising. For sure a deep analysis, in order to understand the common characteristics between the different patients and the group of patients is necessary and to have better AS results, an improvement in the predictions is needed.

6.6 Discussion

In this chapter, LSTM networks are successfully employed to predict glucose levels in T1D patients and used as models in personalized ASs for hypoglycemia and hyperglycemia prevention. The methodology is firstly applied on in silico patients of the UVA/Padova simulator and then on in vivo ones of the OhioT1DM Dataset. The LSTM networks are trained individually for each patient, in order to predict CGM values, with a PH of 40 minutes, considering carbohydrate intakes, insulin quantities and past CGM values in input.
The study considers the entire population of the UVA/Padova simulator. A first in silico result is reported, where a personalized LSTM, called P-LSTM, with population hyperparameters is proposed for each different patient. The results, compared with the state-of-the-art, are promising with a FIT of 76%, DD and UD of around 9 minutes.
Considering the already satisfying results, the P-LSTMs are improved in order to meet some performance requirements to be successfully employed in an AS. The predictions of each patient are analyzed on the validation dataset: if the prediction delay is excessive, the specific patient hyperparameters are ad-hoc optimized, leading to a new model, called EP-LSTM. The obtained results show an improvement with respect to the previous models and so the EP-LSTMs are used in an AS for both hypoglycemia and hyperglycemia prevention. The performances of the AS are satisfying, with TPR of 80% and PPV of 86% for hypoglycemia detection and TPR of 86% and PPV of 80% for hyperglycemia detection.
In order to prove the goodness of the methodology, this has been extended on the in vivo data of the OhioT1DM Dataset, containing data of 12 different patients. First of all, a different EP-LSTM is trained for each patient,

keeping the same network architecture employed in the in silico case. The results are satisfying, with a medium FIT of around 86%, even if the prediction delays are pretty high, with a medium DD of 22 minutes and UD of 20 minutes. Moreover, since the patients are treated with a correct insulin therapy, few hypoglycemia events are present in this dataset and so the AS is designed only for hyperglycemia prevention. The results are very different from patient to patient, with TPR of 48±37 and PPV of 49±35; some patients have very high performances, while with others the AS is not able to detect the critical hyperglycemia events in time.
The results are promising but obviously an improvement of the predictions is necessary in order to reduce the delay and consequently have better AS performances. A first idea is to consider the effect of other inputs in addition to insulin, carbohydrate intake and glucose levels, thanks to the presence of several clinical and physiological data in the OhioT1DM Dataset. In this way it is possible to study how other factors, that can not be obtained from the in silico simulators, can influence the glucose dynamic. In the works cited in Section 6.3.2, that employ the OhioT1DM Dataset, some analysis are already presented: in [155] step count is considered, while in [154] heart rate, skin temperature, skin conductance and time of the day are analyzed. So, in a future development of this work, exercise and step count can be considered along with the current inputs, considering that the duration and the intensity of physical activity can cause a glucose consumption [172]. The presence of a great variety of quantities in the OhioT1DM Dataset is a big advantage of this dataset, however the lack of a sufficient number of hypoglycemia events makes it unsuitable for the design of an hypoglycemia AS. Probably it is better for this purpose to look for a more adequate dataset.
A cluster analysis of the patients can be performed both on in silico and in vivo data, in order to define groups of patients that share the same clinical parameters and could share also the same control and the network hyperparameters. This can be a demanding application but, if successful, it can facilitate the customization of this kind of models and ASs.
Lastly, a further future development can be the implementation of LSTM networks with attention mechanisms, considering the results obtained in [154], since it could help when dealing with sensors data loss. Even if this problem has been solved in this work with an ad-hoc data preprocessing (using retrofitting for in silico data and Kalman smoothing and filtering for in vivo ones), this other approach can be useful to reduce the preprocessing workload.
Anyway it is important to also highlight the limitation shown by a NN approach when applied to this biomedical process. Considering its high complexity, even if some dynamics have already been implemented in the model, like the inter-day and intra-day variability, it is difficult to take into account all the possible events that can occur. If an unexpected event not present in the training dataset happens, it is not known how the black-box model will

respond and the first goal is to avoid everything that can hurt the patients. Since this is a biomedical application, it is necessary to be very careful, even more than what has been already observed in the industrial applications.
In this chapter, it has been proved that LSTM networks can be successfully used to describe the glucose dynamic of T1D patients. Considering the complexity of the process and, consequently, that also the dynamic models that are present in the UVA/Padova simulator [142] are very complicated, a further development is to employ a LSTM network as reference model directly inside the simulator. A slightly different LSTM is currently under study for this purpose, having in input only the carbohydrate intakes and the insulin quantities. Some accurate preliminary analysis are necessary and this project is still in its early stages.

Chapter 7

Concluding remarks

This thesis aims to present modeling strategies of dynamical systems based on ML techniques, showing in particular results obtained adopting a specific NN architecture, the LSTM. The use of ML based techniques in disparate areas has notably increased in recent years thanks to the outstanding performances that can be achieved with this approach. The industrial and the biomedical worlds have been notably impacted by the innovation brought by ML, benefiting of new smart tools, in particular for collection, management and analysis of data, that are now the core of their technological improvements. In the system identification and control fields, the use of ML techniques is still not so diffused, nonetheless thanks to particular architectures that can deal with temporal data, it is possible to use NNs to model dynamical systems.
Four different applications are considered in this thesis. The first three are industrial applications, where the process is modeled both with white-box models and with LSTM based ones. The latter is a biomedical application, where the LSTM network is used for glucose prediction.

Industrial applications

In the first application, the modeling problem is the sterilization process performed by an industrial autoclave. The white-box model is a novelty in the literature; it is built starting from the model of a smaller machine produced by the same company, where the additional components present in the industrial one are considered. The goal is to use this model to build a simulator, given the modular design of the machine that can be easily reproduced with a physics-based model. The LSTM model is instead defined for control purposes, since the physical model is characterized by a high complexity and involves a lot of parameters. Moreover, stability properties are enforced during the training the network. In this first case, both the solutions gave very good results and the choice of one model over the other depends mainly on the specific purpose. The main limitation of the LSTM approach in this ap-

plication is the lack of generality noticed for certain machine configurations due to a dataset not rich enough.

The second application is related to the modeling of an industrial coffee roaster. The goal of the project is to create a scalable model to be identified on a small machine, that can be used also on bigger ones. This problem is common in the food industry, where the data collection causes the waste of resources, that companies want to avoid for economical and environmental reasons. A scalable white-box model is proposed, starting from two models present in literature, that have been modified linking some parameters to the machine geometry, to be portable on machines with different dimensions. This model is validated on data collected from two coffee roasters of different size. Similarly, a LSTM model is trained on the data of the smaller machine to be tested on the data of the bigger one. In this case, the white-box model gives excellent results on both machines while the LSTM model presents criticalities since it is not able to reproduce a trend is not present in the training dataset.

The third application refers to the modeling of the biological reactor of a WWTP. The modeling problem is a well-known case study in literature and also in this case the existent models are not exploitable because of lacking of the required data. So a lumped-parameter model is created, starting from the models present in literature to describe the biological and chemical reactions inside the reactor, considering only the available quantities and exploiting the company's knowledge to add the hydraulic dynamic of the specific plant. The LSTM model is defined similarly, using the same datasets. In this case, the white-box model is not able to correctly reproduce the biological and chemical reactions, probably for a lack of knowledge in the modeling phase, while the hydraulic component of the model is instead quite satisfying. On the other hand the LSTM model gives very good results, leveraging the availability of a good amount of input/output data.

These industrial applications have in common the key role of the data: the goodness of the ML models depends mostly on the quality and the quantity of the used data. The improvements brought in the industrial field by the advent of the Industry 4.0 are trying to make things better in this direction. Analyzing the three applications also from a temporal point of view, the technological innovation can be observed. The modeling of the WWTP is the most recent project, in this case data were easily collected and not only their quantity is adequate for the desired application, but also their quality, considering that they are input/output data able to correctly describe the process; consequently the LSTM model is satisfying. Considering instead the coffee roaster modeling, where the LSTM model is not able to accomplish the goal of the project, the available data are not good enough. Few datasets

were provided and their collection was performed ad-hoc for research purposes, implying so the use of 1240 kg of coffee. The quantity of the data can be enough to identify the physical model but for a ML application it is not sufficient. Anyway, in this application it can be observed how the Industry 4.0 revolution is actually changing industrial world, since the new machines produced by the company are equipped with intelligent sensors, connected to a cloud, where data can be acquired directly during the production phase. In this way a great amount of data can be used to obtain a valid LSTM model. For the modeling of the industrial autoclave, the data collection was ad-hoc performed as well and considering the variety of configurations that can be assumed by the machine and the absence of repeated runs, the results can be further improved with a richer dataset. Also in this case, a big effort was required to the company that must invest time, employees and resources, to carry out a data collection only for research purposes. Another advantage of the Industry 4.0 is the reduction of this investment, considering that the data collection and the consequent data analysis can be performed directly during the production, being also more realistic. However, the availability of lot of data brings new problems and challenges: dealing with real data is not easy, but this subject is now well-known in the ML field, where specialized figures and tools are developed.

Biomedical application

The latter application of this thesis, unlike the previous ones, belongs to the biomedical field. In this case the LSTM networks are used for glucose prediction in T1D patients, with the goal to design a prediction-based AS for hypoglycemia and hyperglycemia prevention.
The methodology is at first defined on the in silico patients of the UVA/ Padova simulator, predicting glucose values with a PH of 40 minutes, considering in input meals, insulin and past glucose values. A different network is trained for each patient, firstly keeping the same hyperparameters for the entire population, then optimizing them for the patients that have a too high prediction delay. The LSTM predictions are so used in an AS obtaining promising results. Moreover, the same methodology is applied to the real data of the OhioT1DM Dataset. The predictions gave very good results and are employed to design an hyperglycemia AS with mixed results, very good for some patients, unsuccessful for others.

The usage of ML techniques in this second field has the purpose to improve the lives of the patients and to help the scientific research. The complexity of the mathematical models in the biomedical field can be reduced thanks to the application of ML, together with a better handling of the challenging biomedical data. In the case study reported in this thesis this goal is achieved, even if there is still a long way to go because a practical appli-

cation of these techniques is obviously more difficult in the biomedical field with respect to the industrial one. For sure, the development of simulators, database, sensors and medical devices is already a big step ahead and can help to better understand the processes that regulate the human body.

In conclusion, the goal of this thesis to analyze the connection between the ML and the system identification fields has been deeply reached. The usage of ML in diverse areas is widely justified by the excellent results that can be obtained, sometimes also with less effort with respect to more traditional instruments. However, this can also have a drawback because ML can be wrongly applied, having anyway a good outcome. With proper attention, study and understanding, ML is a very powerful tool that can simplify different tasks, as in this case for the identification of dynamical systems.

Appendix A

TensorFlow and Keras

A.1 TensorFlow

TensorFlow [108] is an open-source library developed by Google for ML. The TensorFlow framework includes several machine learning algorithms and architectures, that can be used in the most common programming languages (Python, Javascript, Java, C++ and so on). It works on data flow graphs with nodes and edges, where the nodes are mathematical operations and the edges are multidimensional data arrays, called tensors: this working mechanism gives the name TensorFlow. The graph structure also allows to optimize the code execution on GPU; anyway the applications can be executed also on common CPU or even on TPU, Tensor Processing Units, that are custom devices created by Google itself, that can be used on their cloud to further accelerate TensorFlow jobs.
In this thesis, TensorFlow has been implemented in Python, thanks also to the availability of several Python libraries that are very useful in ML implementations, like Numpy, Scipy, Pandas and Matplotlib.

A.2 Keras

Keras [109] is an open-source deep learning library that can be used in different ML libraries like TensorFlow, Theano and PyTorch. In the actual version of TensorFlow, Keras is directly integrated as the high-level API of TensorFlow, implemented as user interface, being more user-friendly and flexible, referred as "`tf.keras`". It provides the most common ML blocks: first of all layers and models, to build the ML architecture, but also optimizers, activation functions, loss functions, to train the model. It allows to preprocess data before the training, and to evaluate and test the model once trained.
Keras simplifies and fastens the building process of a neural network model, in five easy steps:

1. `Model`: this command builds the model, defines the model architecture,

choosing between predefined models, layers, or creating a new custom one. The Sequential model is the most used one, defined as a sequence of layers, and the one used in this thesis;

2. `model.compile`: it compiles the code, choosing the loss function, the optimizer and the metrics that define the model accuracy;

3. `model.fit`: it trains the model to fit the training data for a fixed number of epochs. Training and validation data are here passed to the model, choosing the number of epochs, eventual callbacks and several training options, like `batch_size`, `validation_split` and `shuffle`;

4. `model.evaluate`: it evaluates the model error after training, returning the values of the loss and of the chosen metrics on the training data, to have an idea of how well the training data have been modeled;

5. `model.predict`: it generates the output predictions of the input data passed to the function. It is usually used on new data to make predictions and test the model performances.

Keras contains also sample datasets and pre-trained models that can be freely employed by the users for learning or research purposes.

A.2.1 KerasTuner

A Keras toolbox that was used and helped in the development of the projects presented in this thesis is KerasTuner [110], an hyperparameters optimization framework that exploits search algorithms to automatically find the best hyperparameter values for each model. It can be used to optimize the configuration of a NN model, finding the optimal values of the architecture choices, layer sizes, number of neurons and so on.
The model is wrapped in a `build_model` function, with a `hp` input argument, that calls inside the function each hyperparameter that must be tuned, together with its relative search space. This function returns a compiled model that is called by the tuner.
KerasTuner has three classes of built-in algorithms: `BayesianOptimization`, `Hyperband` and `RandomSearch`, but it can also be used with custom algorithms. The tuner object is called defining the search algorithm, the hyperparameters (called by the `build_model` function), the objective to optimize, the number of trials to run the tuner, path and name of the project. The tuning starts with the `search` method, that requires the same arguments as `model.fit`: the model-building function is run during the search with new hyperparameter values in each trial, the model is fit and evaluated, the tuner gradually analyzes the search space until it identifies a suitable set of hyperparameters values. At the end, the function returns the best model with its relative hyperparameters.

In this work, KerasTuner was employed with `RandomSearch` algorithm, minimizing the validation loss as objective function, to tune the number of neurons of the LSTM layer and the learning rate of the Adam optimizer.

Appendix B

The UVA/Padova simulator

The availability of glucose-insulin models that simulate the glucose response to insulin and meal intake is a key point for the design and evaluation of glucose sensors, control algorithms, alarms and decision support systems. In fact, computer simulations allow to perform several in silico tests with relevant time- and cost- savings, and they also allow to simulate experiments potentially dangerous for patient safety.
In the last decades, several simulation tools have been developed (see [173] for a review), each one based on a comprehensive mathematical model and equipped with an in silico population. In 2008 the US Food and Drug Administration (FDA) accepted the UVA/Padova simulator, a T1D simulator developed by Universities of Virginia (UVA) and Padova, as a substitute for preclinical trials for closed-loop algorithms tests. This simulator is equipped with 300 in silico subjects generated from a joint distribution of model parameters, obtained by identifying a complex model [174] from a unique multiple-tracer dataset. This simulator is able to span the variability observed in the real T1D population representing the inter-individual variability that characterizes this population. Moreover, the new version of the simulator [142] incorporates a nonlinear model to describe the glucose response to hypoglycemia and the counter-regulation [170], and a model with time-varying parameters to describe the intra-day variability of the Insulin Sensitivity (SI) and the meal intestinal absorption rate, taking into account the circadian variability of SI and the dawn phenomena [175]. The complete model equations can be found in [142].
This simulator has been used by more than 30 sites in academia and companies involved in T1D research, more than 70 articles were published in peer-reviewed journals [142]. The Epsilon Group (TEG) offers a commercial version of this simulator, the TEG's Diabetes Mellitus Metabolic Simulator for Research (DMMS.R). It provides a unique in silico environment for testing diabetes treatment and monitoring interventions, and it is ideal for modeling new devices or examining treatment protocols and dosing algo-

rithms. For this reason, the data generated by the simulator can not be made publicity available.

B.1 CGM noise model

The measurement error model used in this work to simulate the CGM measurements on the UVA/Padova simulator is the autoregressive model proposed in [162], which was identified exploiting 141 datasets relative to the 47 patients enrolled in the CAT AP@home trial [176]. This model describes the total measurement error, including wearing issues in addition to noise and drift usually considered. It can be described by the formula:

$$\varepsilon_{PV}(k) = a_1 \cdot \varepsilon_{PV}(k-1) + a_2 \cdot \varepsilon_{PV}(k-2) + v(k)$$

where the error $v(k)$ is a Gaussian White Noise with mean μ_v and variance σ_v^2; the parameters a_1, a_2 are coefficients of the AR model and the distribution of the initial states of the process is normal with mean μ_{is} and variance σ_{is}^2. The parameters used in this work are: $a_1 = 1.5458$, $a_2 = -0.5708$, $\mu_v = 0.0017$ [mmol/L], $\sigma_v^2 = 0.0283$ [mmol^2/L^2], $\mu_{is} = [-0.1766 \quad -0.1566]$ [mmol/L] and

$$\sigma_{is}^2 = \begin{bmatrix} 0.7759 & 0.7895 \\ 0.7895 & 0.8603 \end{bmatrix} \quad [\text{mmol}^2/\text{L}^2].$$

B.2 Pavia MPC controller

The insulin therapy in all datasets is computed by the MPC developed in [162] tuned in a sub-optimal way in order to have a significant amount of critical episodes. In fact, the goal of this work is the detection of hypoglycemia and hyperglycemia events, so we need patients with regulation problems. The MPC algorithm [162] is a Linear-Model-Predictive-Control (LMPC) that uses an approximate linear model of the insulin-glucose dynamics in order to predict the future glucose profile given the carbohydrates and insulin intakes. This model is obtained by the linearization around a suitable working point of the more complex nonlinear Dalla Man model [177] of the average in silico patients of the UVA/Padova simulator. This linearized model can be written as:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + Md(k) \\ y(k) = Cx(k) \end{cases} \tag{B.1}$$

where

- $x(k) \in R^{13}$, is the 13-state vector;
- $y(k) = CGM(k) - G_b$ [mg/dl] is the difference between the subcutaneous glucose CGM and the basal value (G_b);

- $u(k) = i(k) - u_b(k)$ [pmol/kg] is the difference between the injected insulin i and its reference basal value u_b, normalized by the patient weight;
- $d(k)$ [mg], represents the meal.

The triplet (A, B, C) is assumed both stabilizable and detectable. The MPC cost function is a quadratic penalty function J defined as follows:

$$J\left(x\left(k\right), u\left(\cdot\right), k\right) = \sum_{i=0}^{N-1} \left(q\left(y\left(k+i\right) - y_o\left(k+i\right)\right)^2 + \left(u\left(k+i\right) - u_o\left(k+i\right)\right)^2\right) + \|x(k+N)\|_P^2 \tag{B.2}$$

where q is the positive scalar weight and N is the prediction horizon. Moreover, $||\ x(k+N)\ ||_P = x(k+N)'Px(k+N)$, where P is the unique nonnegative solution of the discrete time Riccati equation

$$P = A'PA + qC'C - A'PB(1 + B'PB)B'PA$$

The reference signals are defined as:

- $y_o(k) = \widetilde{y}(k) - G_b$ [mg/dl], is the difference between the reference value $(\widetilde{y})$ of the subcutaneous glucose and the glucose basal value;
- $u_o(t) = \widetilde{u}(k) - u_b(k)$ [pmol/kg], is the difference between the reference value $(\widetilde{u})$ of the insulin profile and the insulin basal value, normalized by the patient weight.

The proposed algorithm does not include explicit constraints, so it is possible to calculate the closed form solution as follows

$$u^{MPC}(k) = \begin{bmatrix}1 & 0 & \cdots & 0\end{bmatrix}\left(-K_x x(k) - K_d D(k) + K_{Y_o} Y_o(k) + K_{U_o} U_o(k)\right)$$

where $K_x \in R^{Nx13}$, $K_d \in R^{NxN}$, $K_{Y_o} \in R^{NxN}$, $K_{U_o} \in R^{NxN}$ are derived as described in [178] and

$$D(k) = \begin{bmatrix}d(k) & \ldots & d(k+N-2) & d(k+N-1)\end{bmatrix}'$$

is the vector of future meals and

$$\begin{aligned} Y_o(k) &= [y_o(k+1) \quad \cdots \quad y_o(k+N-1) \quad 0]' \\ U_o(k) &= [u_o(k) \quad \cdots \quad u_o(k+N-2) \quad u_o(k+N-1)]' \end{aligned}$$

Since the state $x(k)$ of the model is not accessible, a Kalman Filter is used to estimate it. The linear system B.1 can be written considering the noises as:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) + Md(k) + v_x(k) \\ y(k) = Cx(k) + v_y(k) \end{cases}$$

where $v = [v_x \;\; v_y]$ is a multivariate zero-mean white Gaussian noise with covariance matrix:

$$V = \begin{bmatrix} Q_{KF} & 0 \\ 0 & R_{KF} \end{bmatrix}, \; Q_{KF} > 0, \; R_{KF} > 0$$

and the initial state $x_0 = x(0)$ is assumed to be a zero mean Gaussian random variable independent of v.
The control weight q in Equation B.2 is individualized using body weight, BW, and CR ratio following the formula:

$$q = Q_m \; e^{(-0.0366 * BW \; -0.2149 * CR + \; 2.5444)}$$

This algorithm with $Q_m = 1$ was successfully used in 3 outpatient trials lasted 1-2 months [179–181], while in this work the parameter Q_m is set equal to 10 to make sub-optimal the control action, in order to have a significant amount of critical episodes (hypo/hyperglycemia events) in all the datasets.

Appendix C

Kalman filtering and smoothing for glucose preprocessing

In [166] a method for glucose preprocessing is presented, based on a Kalman filtering and smoothing technique, exploiting both CGM and SMBG data.

C.1 Kalman filtering

The Kalman filter technique is used to estimate the optimal value of the state in a linear dynamic system.
Considering a system expressed as:

$$\begin{aligned} x_k &= \Phi x_{k-1} + w_{k-1} \\ y_k &= C x_k + v_k \end{aligned} \tag{C.1}$$

where x is the state of the system, u the input, y the output, $\Phi = e^{A\Delta t}$ is the matrix of the dynamics of the system, discretized with sample time Δt. The process noise w and the measurement noise v are assumed white Gaussian noises, with covariance defined my matrices Q and R:

$$w_k \sim \mathcal{N}(0, Q) \qquad v_k \sim \mathcal{N}(0, R)$$

Firstly, an a priori estimate of the state $\bar{x}$ and of its covariance matrix $\bar{P}$ is calculated, using the model (C.1):

$$\begin{aligned} \bar{x}_k &= \Phi \hat{x}_{k-1} \\ \bar{P}_k &= \Phi \hat{P}_{k-1} \Phi^T + Q \end{aligned}$$

Then, an a posteriori estimate of $\hat{x}$ and $\hat{P}$ is calculated by updating the a priori estimate with a measurement, considering a gain K. If the measurement is not available in a certain time step, the a posteriori estimate is set

equal to the a priori one ($\hat{x}_k = \bar{x}_k$, $\hat{P}_k = \bar{P}_k$).

$$K_k = \bar{P}_k C^T (R + C\bar{P}_k C^T)^{-}1)$$
$$\hat{x}_k = K_k(y_k - C\bar{x}_k) + \bar{x}_k$$
$$\hat{P}_k = (I - K_k C)\bar{P}_k$$

C.2 Kalman smoothing

The Kalman smoothing technique is used improve the result obtained with the Kalman filter.
The algorithm is firstly initialized considering the last state and covariance matrix estimations of the Kalman filter, $\hat{x}^S_{k+1} = \hat{x}_{k+1}$ and $\hat{P}^S_{k+1} = \hat{P}_{k+1}$. The gain F is updated and a backward pass is performed to update the state such that the next estimation is as close as possible to the real value. In this way, in case of missing data the backward pass can consider the contribution of the measurements of the subsequent time steps to reconstruct the signal.

$$F_k = \hat{P}_k \Phi^T \bar{P}^{-1}_{k+1}$$
$$\hat{x}^s_k = \hat{x}_k + F_k(\hat{x}^s_{k+1} - \bar{x}_{k+1})$$
$$\hat{P}^s_k = \hat{P}_k + F_k(\hat{P}^s_{k+1} - \bar{P}_{k+1})F^T_k$$

C.3 Dynamic models

The Kalman filter requires a dynamic model to obtain the state predictions and in [166] two different dynamic models are proposed.
In order to be used for this technique, a model must be observable, to compute the internal state from the measurements, and without external inputs, to avoid having necessarily the measures of insulin and meals. So since meals and insulin injections are not considered inputs of the model but as unknown disturbances, the variance of the process noise must be set large enough to include their effects. The two proposed models satisfy the requirements.

C.3.1 Model 1: Rate-Only Model

Model 1 is described by two states representing the plasma glucose and its rate, $x = \begin{bmatrix} G_p & \dot{G}_p \end{bmatrix}^T$, with $\dot{x} = Ax$, where:

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -a \end{bmatrix}$$

The parameter a is the decay of an observed rate of change rate. The process noise is set to:

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & q_{m1} \end{bmatrix} \Delta t$$

The system is discretized, with $\Delta t = 10\,[s]$; the values of the parameters are $a = 0.05$ and $q_{m1} = 0.005\,[mmol^2/L^2]$.

C.3.2 Model 2: Central-Remote Rate Model

Model 2 is inspired by [182] and further simplified. The glucose rate G_p of Model 1, is divided in two states in Model 2: C_c, that represents a central compartment, and C_r, a remote compartment. The inputs affect the central compartment, from which are diffused to the remote one, considering a certain delay, and from which are then diffused to the glucose.
The model is described by three states, $x = [G_p \quad C_c \quad C_r]^T$:

$$\begin{aligned}\dot{G}_p &= Cr \\ \dot{C}_c &= -\frac{1}{T_d}C_c \\ \dot{C}_r &= \frac{1}{T_d}(C_c - C_r)\end{aligned}$$

T_d is a time constant that represents the flow rate between the compartments, set equal to $600\,[s]$. The process noise is:

$$Q = \begin{bmatrix}0 & 0 & 0 \\ 0 & q_{m2} & 0 \\ 0 & 0 & 0\end{bmatrix} \Delta t$$

where $q_{m2} = 0.02\,[mmol^2/L^2]$.

Bibliography

[1] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues", *Journal of industrial information integration*, vol. 6, pp. 1–10, 2017.

[2] V. Roblek, M. Meško, and A. Krapež, "A complex view of industry 4.0", *Sage open*, vol. 6, no. 2, p. 2 158 244 016 653 987, 2016.

[3] S. Vaidya, P. Ambad, and S. Bhosle, "Industry 4.0–a glimpse", *Procedia manufacturing*, vol. 20, pp. 233–238, 2018.

[4] G. Erboz, "How to define industry 4.0: Main pillars of industry 4.0", *Managerial trends in the development of enterprises in globalization era*, vol. 761, p. 767, 2017.

[5] R. Rai, M. K. Tiwari, D. Ivanov, and A. Dolgui, *Machine learning in manufacturing and industry 4.0 applications*, 2021.

[6] V. C. Coffey, "Machine vision: the eyes of industry 4.0", *Optics and photonics news*, vol. 29, no. 7, pp. 42–49, 2018.

[7] A. Angelopoulos, E. T. Michailidis, N. Nomikos, P. Trakadas, A. Hatziefremidis, S. Voliotis, and T. Zahariadis, "Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects", *Sensors*, vol. 20, no. 1, p. 109, 2019.

[8] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni, and J. Loncarski, "Machine learning approach for predictive maintenance in industry 4.0", in *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, IEEE, 2018, pp. 1–6.

[9] J. P. Usuga Cadavid, S. Lamouri, B. Grabot, R. Pellerin, and A. Fortin, "Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0", *Journal of Intelligent Manufacturing*, vol. 31, no. 6, pp. 1531–1558, 2020.

[10] J. Sansana, M. N. Joswiak, I. Castillo, Z. Wang, R. Rendall, L. H. Chiang, and M. S. Reis, "Recent trends on hybrid modeling for Industry 4.0", *Computers & Chemical Engineering*, vol. 151, p. 107 365, 2021.

[11] C. Park, C. C. Took, and J.-K. Seong, "Machine learning in biomedical engineering", *Biomedical Engineering Letters*, vol. 8, no. 1, pp. 1–3, 2018.

[12] C. Turchetti and L. Falaschetti, *Machine Learning in Electronic and Biomedical Engineering*, 2022.

[13] M. Strzelecki and P. Badura, *Machine Learning for Biomedical Application*, 2022.

[14] F. Iacono, J. L. Presti, I. Schimperna, S. Ferretti, A. Mezzadra, L. Magni, and C. Toffanin, "Improvement of manufacturing technologies through a modelling approach: an air-steam sterilization case-study", *Procedia Computer Science*, vol. 180, pp. 162–171, 2021.

[15] F. Di Palma, F. Iacono, C. Toffanin, A. Ziccardi, and L. Magni, "Scalable model for industrial coffee roasting chamber", *Procedia Computer Science*, vol. 180, pp. 122–131, 2021.

[16] C. Toffanin, F. Di Palma, F. Iacono, and L. Magni, "LSTM network for the oxygen concentration modeling of a wastewater treatment plant", in *Applied Sciences*, Submitted.

[17] F. Iacono, L. Magni, and C. Toffanin, "Patient-tailored LSTM model for hypoglycemia prevention: an in-silico case study", in *Mathematical Modelling and Control for Healthcare and Biomedical Systems (MCHBS 2021), Virtual Online Conference*, 2021.

[18] F. Iacono, L. Magni, and C. Toffanin, "Personalized LSTM models for glucose prediction in Type 1 diabetes subjects", in *2022 30th Mediterranean Conference on Control and Automation (MED)*, 2022, pp. 324–329.

[19] C. Toffanin, F. Iacono, and L. Magni, "Personalized LSTM-Based Alarm Systems for Hypoglycemia Prevention", in *2023 31th Mediterranean Conference on Control and Automation (MED)*, Accepted.

[20] F. Iacono, L. Magni, and C. Toffanin, "Personalized LSTM-based alarm systems for hypoglycemia and hyperglycemia prevention", in *Biomedical Signal Processing and Control*, Submitted.

[21] S. N. Kumpati, P. Kannan, *et al.*, "Identification and control of dynamical systems using neural networks", *IEEE Transactions on neural networks*, vol. 1, no. 1, pp. 4–27, 1990.

[22] P. J. Werbos, "Neural networks for control and system identification", in *Proceedings of the 28th IEEE Conference on Decision and Control*, IEEE, 1989, pp. 260–265.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation", California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[24] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network", *Advances in neural information processing systems*, vol. 2, 1989.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need", *Advances in neural information processing systems*, vol. 30, 2017.

[28] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun, "Transformers in time series: A survey", *arXiv preprint arXiv:2202.07125*, 2022.

[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate", *arXiv preprint arXiv:1409.0473*, 2014.

[30] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[31] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult", *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[32] P. J. Werbos, "Backpropagation through time: what it does and how to do it", *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[33] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks", in *International conference on machine learning*, PMLR, 2013, pp. 1310–1318.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM", *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[36] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks", *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[37] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey", *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[38] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition", in *2014 14th international conference on frontiers in handwriting recognition*, IEEE, 2014, pp. 285–290.

[39] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", *arXiv preprint arXiv:1409.1259*, 2014.

[40] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling", *arXiv preprint arXiv:1412.3555*, 2014.

[41] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks", *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[42] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators", *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[43] G. Cybenko, "Approximation by superpositions of a sigmoidal function", *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[44] A. M. Schäfer and H. G. Zimmermann, "Recurrent neural networks are universal approximators", in *International Conference on Artificial Neural Networks*, Springer, 2006, pp. 632–640.

[45] K. S. Narendra, "Neural networks for control theory and practice", *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1385–1406, 1996.

[46] D. Mandic and J. Chambers, *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley, 2001.

[47] A. Delgado, C. Kambhampati, and K. Warwick, "Dynamic recurrent neural network for system identification and control", *IEE Proceedings-Control Theory and Applications*, vol. 142, no. 4, pp. 307–314, 1995.

[48] G. Van Houdt, C. Mosquera, and G. Nápoles, "A review on the long short-term memory model", *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5929–5955, 2020.

[49] Y. Wang, "A new concept using LSTM neural networks for dynamic system identification", in *2017 American control conference (ACC)*, IEEE, 2017, pp. 5324–5329.

[50] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön, "Deep learning and system identification", *IFAC-Papers OnLine*, vol. 53, no. 2, pp. 1175–1181, 2020.

[51] P. Park, P. Di Marco, H. Shin, and J. Bang, "Fault Detection and Diagnosis Using Combined Autoencoder and Long Short-Term Memory Network", *Sensors*, vol. 19, no. 21, Nov. 2019. DOI: 10.3390/s19214612.

[52] X. Zhang, Z. Zhao, Z. Wang, and X. Wang, "Fault Detection and Identification Method for Quadcopter Based on Airframe Vibration Signals", *Sensors*, vol. 21, no. 2, Jan. 2021. DOI: 10.3390/s21020581.

[53] X. Li, F. Duan, P. Loukopoulos, I. Bennett, and D. Mba, "Canonical variable analysis and long short-term memory for fault diagnosis and performance estimation of a centrifugal compressor", *Control Engineering Practice*, vol. 72, pp. 177–191, Mar. 2018, ISSN: 0967-0661. DOI: 10.1016/j.conengprac.2017.12.006.

[54] X. Chen, B. Zhang, and D. Gao, "Bearing fault diagnosis base on multi-scale CNN and LSTM model", *Journal of intelligent manufacturing*, vol. 32, no. 4, pp. 971–987, Apr. 2021. DOI: 10.1007/s10845-020-01600-2.

[55] A. Brusaferri, M. Matteueei, P. Portolani, and S. Spinelli, "Nonlinear system identification using a recurrent network in a Bayesian framework", in *2019 IEEE 17th International Conference On Industrial Informatics (INDIN)*, 2019, pp. 319–324.

[56] H. Kim, A. M. Arigi, and J. Kim, "Development of a diagnostic algorithm for abnormal situations using long short-term memory and variational autoencoder", *Annals of nuclear energy*, vol. 153, Apr. 2021, ISSN: 0306-4549. DOI: 10.1016/j.anucene.2020.108077.

[57] J. Yang and J. Kim, "An accident diagnosis algorithm using long short-term memory", *Nuclear Engineering and Technology*, vol. 50, no. 4, SI, pp. 582–588, May 2018. DOI: 10.1016/j.net.2018.03.010.

[58] S. Wang, X. Shao, L. Yang, and N. Liu, "Deep Learning Aided Dynamic Parameter Identification of 6-DOF Robot Manipulators", *IEEE Access*, vol. 8, pp. 138 102–138 116, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3012196.

[59] E. Chemali, P. J. Kollmeyer, M. Preindl, R. Ahmed, and A. Emadi, "Long Short-Term Memory Networks for Accurate State-of-Charge Estimation of Li-ion Batteries", *IEEE Transactions On Industrial Electronics*, vol. 65, no. 8, pp. 6730–6739, Aug. 2018, ISSN: 0278-0046. DOI: 10.1109/TIE.2017.2787586.

[60] R. Zhang, Z. Chen, S. Chen, J. Zheng, O. Buyukozturk, and H. Sun, "Deep long short-term memory networks for nonlinear structural seismic response prediction", *Computers & Structures*, vol. 220, pp. 55–68, Aug. 2019, ISSN: 0045-7949. DOI: 10.1016/j.compstruc.2019.05.006.

[61] F. Bonassi, M. Farina, J. Xie, and R. Scattolini, "On Recurrent Neural Networks for learning-based control: recent results and ideas for future developments", *Journal of Process Control*, vol. 114, pp. 92–104, 2022.

[62] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating scientific knowledge with machine learning for engineering and environmental systems", *ACM Computing Surveys (CSUR)*, 2021.

[63] Z.-P. Jiang and Y. Wang, "Input-to-state stability for discrete-time nonlinear systems", *Automatica*, vol. 37, no. 6, pp. 857–869, 2001.

[64] F. Bayer, M. Bürger, and F. Allgöwer, "Discrete-time incremental ISS: A framework for robust NMPC", in *2013 European Control Conference (ECC)*, IEEE, 2013, pp. 2068–2073.

[65] F. Bonassi, M. Farina, and R. Scattolini, "Stability of discrete-time feed-forward neural networks in NARX configuration", *IFAC-Papers OnLine*, vol. 54, no. 7, pp. 547–552, 2021.

[66] L. B. Armenio, E. Terzi, M. Farina, and R. Scattolini, "Model predictive control design for dynamical systems learned by echo state networks", *IEEE Control Systems Letters*, vol. 3, no. 4, pp. 1044–1049, 2019.

[67] E. Terzi, F. Bonassi, M. Farina, and R. Scattolini, "Learning model predictive control with long short-term memory networks", *International Journal of Robust and Nonlinear Control*, vol. 31, no. 18, pp. 8877–8896, 2021.

[68] F. Bonassi, M. Farina, and R. Scattolini, "On the stability properties of gated recurrent units neural networks", *Systems & Control Letters*, vol. 157, p. 105 049, 2021.

[69] F. Bonassi, E. Terzi, M. Farina, and R. Scattolini, "LSTM neural networks: Input to state stability and probabilistic safety verification", in *Learning for Dynamics and Control*, PMLR, 2020, pp. 85–94.

[70] Z.-P. Jiang and Y. Wang, "Input-to-state stability for discrete-time nonlinear systems", *Automatica*, vol. 37, no. 6, pp. 857–869, 2001.

[71] C. Andersson, A. H. Ribeiro, K. Tiels, N. Wahlström, and T. B. Schön, "Deep convolutional networks in system identification", in *2019 IEEE 58th conference on decision and control (CDC)*, IEEE, 2019, pp. 3670–3676.

[72] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 156–165.

[73] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling", *arXiv preprint arXiv:1803.01271*, 2018.

[74] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio", *arXiv preprint arXiv:1609.03499*, 2016.

[75] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[76] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, *et al.*, "Recent advances in convolutional neural networks", *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[77] *Fedegari Group*. [Online]. Available: `https://www.fedegari.com/`.

[78] V. K. Pillai, A. N. Beris, and P. S. Dhurjati, "Implementation of Model-Based Optimal Temperature Profiles for Autoclave Curing of Composites Using a Knowledge-Based System", *Industrial & Engineering Chemistry Research*, vol. 33, no. 10, pp. 2443–2452, 1994.

[79] C. Dorfling, G. Akdogan, S. Bradshaw, and J. Eksteen, "Modelling of an autoclave used for high pressure sulphuric acid/oxygen leaching of first stage leach residue. Part 1: Model development", *Minerals Engineering*, vol. 53, pp. 220–227, 2013, ISSN: 0892-6875.

[80] M. Telikicherla, M. Altan, and F. Lai, "Autoclave curing of thermosetting composites: Process modeling for the cure assembly", *International Communications in Heat and Mass Transfer*, vol. 21, no. 6, pp. 785–797, 1994.

[81] W. L. Lau, J. Reizes, V. Timchenko, S. Kara, and B. Kornfeld, "Heat and mass transfer model to predict the operational performance of a steam sterilisation autoclave including products", *International Journal of Heat and Mass Transfer*, vol. 90, pp. 800–811, 2015, ISSN: 0017-9310.

[82] A. Preglej, R. Karba, I. Steiner, and I. Škrjanc, "Mathematical model of an autoclave", *Strojniški vestnik-Journal of Mechanical Engineering*, vol. 57, no. 6, pp. 503–516, 2011.

[83] F. Iacono, S. Ferretti, A. Mezzadra, L. Magni, and C. Toffanin, "Industry 4.0: Mathematical model for monitoring sterilization processes", in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 610–615.

[84] D. Pistolesi and V. Mascherpa, "F0 A technical note", 2015. [Online]. Available: `https://www.fedegari.com/wp-content/uploads/2019/03/ST19_EBook_F0-What-it-means-How-to-calculate-it-How-to-use-it.pdf`.

[85] D. W. Green, "Perry's Chemical Engineers' Handbook/edición Don W", *Green y Robert H. Perry*, vol. 100, pp. 660–28, 1973.

[86] J. Baggenstoss, L. Poisson, R. Luethi, R. Perren, and F. Escher, "Influence of Water Quench Cooling on Degassing and Aroma Stability of Roasted Coffee", *Journal of Agricultural and Food Chemistry*, vol. 55, no. 16, pp. 6685–6691, 2007.

[87] M. Milani, L. Montorsi, and S. Terzi, "Numerical analysis of the heat recovery efficiency for the post-combustion flue gas treatment in a coffee roaster plant", *Energy*, vol. 141, pp. 729–743, 2017, ISSN: 0360-5442.

[88] F. Winjaya, M. Rivai, and D. Purwanto, "Identification of cracking sound during coffee roasting using neural network", in *2017 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 2017, pp. 271–274.

[89] Y. Thazin, T. Pobkrut, and T. Kerdcharoen, "Prediction of Acidity Levels of Fresh Roasted Coffees Using E-nose and Artificial Neural Network", in *2018 10th International Conference on Knowledge and Smart Technology (KST)*, 2018, pp. 210–215.

[90] Y. Xu, J. Shaull, T. Bavar, and L. Tan, "Smart coffee roaster design with connected devices", in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, 2018, pp. 1–5.

[91] H. Schwartzberg, "Modeling bean heating during batch roasting of coffee beans", in *Engineering and Food for the 21st Century*, CRC Press, 2002, pp. 901–920.

[92] W. Hernández-Díaz, I. Ruiz-López, M. Salgado-Cervantes, G. Rodríguez-Jimenes, and M. García-Alvarado, "Modeling heat and mass transfer during drying of green coffee beans using prolate spheroidal geometry", *Journal of Food Engineering*, vol. 86, no. 1, pp. 1–9, 2008, ISSN: 0260-8774.

[93] D. Bottazzi, S. Farina, M. Milani, and L. Montorsi, "A numerical approach for the analysis of the coffee roasting process", *Journal of Food Engineering*, vol. 112, no. 3, pp. 243–252, 2012, ISSN: 0260-8774.

[94] T. A. Haley and S. J. Mulvaney, "Advanced process control techniques for the food industry", *Trends in Food Science and Technology*, vol. 6, no. 4, pp. 103–110, 1995, ISSN: 0924-2244. DOI: https://doi.org/10.1016/S0924-2244(00)88992-X.

[95] L. Pérez-Alegría and H. Ciro-Velasquez, "Mathematical simulation of parchment coffee drying in a deep bed with airflow reversal", *Transactions of the American Society of Agricultural Engineers*, vol. 44, no. 5, pp. 1229–1234, 2001.

[96] D. S. Leme, S. A. da Silva, B. H. G. Barbosa, F. M. Borém, and R. G. F. A. Pereira, "Recognition of coffee roasting degree using a computer vision system", *Computers and electronics in agriculture*, vol. 156, pp. 312–317, 2019.

[97] M. Okamura, M. Soga, Y. Yamada, K. Kobata, and D. Kaneda, "Development and evaluation of roasting degree prediction model of coffee beans by machine learning", *Procedia Computer Science*, vol. 192, pp. 4602–4608, 2021.

[98] E. M. de Oliveira, D. S. Leme, B. H. G. Barbosa, M. P. Rodarte, and R. G. F. A. Pereira, "A computer vision system for coffee beans classification based on computational intelligence techniques", *Journal of Food engineering*, vol. 171, pp. 22–27, 2016.

[99] J. Hernández, B. Heyd, and G. Trystram, "Prediction of brightness and surface area kinetics during coffee roasting", *Journal of Food Engineering*, vol. 89, no. 2, pp. 156–163, 2008.

[100] C. E. Bolt and P. L. de Vaal, "A Practical Guide to Coffee Roaster Modelling", in *Computer Aided Chemical Engineering*, vol. 51, Elsevier, 2022, pp. 145–150.

[101] *Brambati S.p.A.* [Online]. Available: https://www.brambati.it/.

[102] J. Vosloo, "Heat and mass transfer model for a coffee roasting process", M.S. thesis, North-West University (South Africa), Potchefstroom Campus, 2017.

[103] F. P. Incropera, A. S. Lavine, T. L. Bergman, and D. P. DeWitt, *Principles of heat and mass transfer*. Wiley, 2013.

[104] A. Raemy, "Differential thermal analysis and heat flow calorimetry of coffee and chicory products", *Thermochimica Acta*, vol. 43, no. 2, pp. 229–236, 1981, ISSN: 0040-6031.

[105] R. Draghi, "Model Identification of a Coffee Roasting Plant for the creation of a Virtual Sensor", M.S. thesis, University of Pavia, Italy, 2018.

[106] Zhiqiang Sun, Zhiyong Li, and Jianwu Zheng, "Influence of improper installation on measurement performance of Pitot tube", in *2009 International Conference on Industrial Mechatronics and Automation*, 2009, pp. 53–56.

[107] P. R. Thottathikudy, "Model Identification of an Industrial Coffee Roasting Plant", M.S. thesis, University of Pavia, 2019.

[108] Martín Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[109] F. Chollet *et al.*, *Keras*, `https://keras.io`, 2015.

[110] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, *et al.*, *KerasTuner*, `https://github.com/keras-team/keras-tuner`, 2019.

[111] S. Schenker, C. Heinemann, M. Huber, R. Pompizzi, R. Perren, and R. Escher, "Impact of roasting conditions on the formation of aroma compounds in coffee beans", *Journal of food science*, vol. 67, no. 1, pp. 60–66, 2002.

[112] R. Silwar and C. Lullmann, "Investigation of aroma formation in robusta coffee during roasting", *Cafe Cacao The (France)*, 1993.

[113] T. Chai, "Intelligent feedback control for operation of complex industrial processes", in *2015 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2015, pp. 1–3.

[114] R. Hamitlon, B. Braun, R. Dare, B. Koopman, and S. A. Svoronos, "Control issues and challenges in wastewater treatment plants", *IEEE control systems magazine*, vol. 26, no. 4, pp. 63–69, 2006.

[115] K. V. Gernaey, M. C. Van Loosdrecht, M. Henze, M. Lind, and S. B. Jørgensen, "Activated sludge wastewater treatment plant modelling and simulation: state of the art", *Environmental modelling & software*, vol. 19, no. 9, pp. 763–783, 2004.

[116] R. Hreiz, M. Latifi, and N. Roche, "Optimal design and operation of activated sludge processes: State-of-the-art", *Chemical Engineering Journal*, vol. 281, pp. 900–920, 2015.

[117] M. Henze, C. L. Grady Jr, W. Gujer, G. Marais, and T. Matsuo, "A general model for single-sludge wastewater treatment systems", *Water Research*, vol. 21, no. 5, pp. 505–515, 1987. DOI: `https://doi.org/10.1016/0043-1354(87)90058-3`.

[118] W. Gujer, M. Henze, T. Mino, T. Matsuo, M. Wentzel, and G. Marais, "The Activated Sludge Model No. 2: Biological phosphorus removal", *Water Science and Technology*, vol. 31, no. 2, pp. 1–11, 1995, Modelling and Control of Activated Sludge Processes. DOI: https://doi.org/10.1016/0273-1223(95)00175-M.

[119] W. Gujer, M. Henze, T. Mino, and M. Van Loosdrecht, "Activated sludge model No. 3", *Water Science and Technology*, vol. 39, no. 1, pp. 183–193, 1999, Modelling and microbiology of activated sludge processes, ISSN: 0273-1223. DOI: https://doi.org/10.1016/S0273-1223(98)00785-9.

[120] M. Nelson and H. S. Sidhu, "Analysis of the activated sludge model (number 1)", *Applied Mathematics Letters*, vol. 22, no. 5, pp. 629–635, 2009.

[121] M. Nelson, H. S. Sidhu, S. Watt, and F. I. Hai, "Performance analysis of the activated sludge model (number 1)", *Food and Bioproducts Processing*, vol. 116, pp. 41–53, 2019.

[122] W. Gujer, "Activated sludge modelling: past, present and future", *Water Science and Technology*, vol. 53, no. 3, pp. 111–119, 2006.

[123] G. Sin and R. Al, "Activated sludge models at the crossroad of artificial intelligence—A perspective on advancing process modeling", *Npj Clean Water*, vol. 4, no. 1, pp. 1–7, 2021.

[124] *ASMortara S.p.A.* [Online]. Available: https://www.asmortara.eu.

[125] M. Collivignarelli, A. Abbà, A. Frattarola, S. Manenti, S. Todeschini, G. Bertanza, and R. Pedrazzani, "Treatment of aqueous wastes by means of Thermophilic Aerobic Membrane Reactor (TAMR) and nanofiltration (NF): Process auditing of a full-scale plant", *Environmental monitoring and assessment*, vol. 191, no. 12, pp. 1–17, 2019.

[126] M. C. Collivignarelli, A. Abbà, and G. Bertanza, "Why use a thermophilic aerobic membrane reactor for the treatment of industrial wastewater/liquid waste?", *Environmental Technology*, vol. 36, no. 16, pp. 2115–2124, 2015.

[127] P. Frącz, "Nonlinear modeling of activated sludge process using the Hammerstein-Wiener structure", in *E3S Web of Conferences*, EDP Sciences, vol. 10, 2016, p. 00 119.

[128] U. Jeppsson, "A general description of the IAWQ activated sludge model No. 1", *IEA, Lund*, 1997.

[129] J. Monod, "The growth of bacterial cultures", *Annual review of microbiology*, vol. 3, no. 1, pp. 371–394, 1949.

[130] "International diabetes federation", *IDF Diabetes Atlas, 10th edn. Brussels, Belgium: International Diabetes Federation*, 2021.

[131] A. Janež, C. Guja, A. Mitrakou, N. Lalic, T. Tankova, L. Czupryniak, A. G. Tabák, M. Prazny, E. Martinka, and L. Smircic-Duvnjak, "Insulin therapy in adults with type 1 diabetes mellitus: a narrative review", *Diabetes Therapy*, vol. 11, no. 2, pp. 387–409, 2020.

[132] R. M. Bergenstal, W. V. Tamborlane, A. Ahmann, J. B. Buse, G. Dailey, S. N. Davis, C. Joyce, T. Peoples, B. A. Perkins, J. B. Welsh, *et al.*, "Effectiveness of sensor-augmented insulin-pump therapy in type 1 diabetes", *New England Journal of Medicine*, vol. 363, no. 4, pp. 311–320, 2010.

[133] J. Hermanides, K. Nørgaard, D. Bruttomesso, C. Mathieu, A. Frid, C. M. Dayan, P. Diem, C. Fermon, I. Wentholt, J. Hoekstra, *et al.*, "Sensor-augmented pump therapy lowers HbA1c in suboptimally controlled Type 1 diabetes; a randomized controlled trial", *Diabetic Medicine*, vol. 28, no. 10, pp. 1158–1167, 2011.

[134] K. Nørgaard, A. Scaramuzza, N. Bratina, N. M. Lalić, P. Jarosz-Chobot, G. Kocsis, E. Jasinskiene, C. De Block, O. Carrette, J. Castañeda, *et al.*, "Routine sensor-augmented pump therapy in type 1 diabetes: the INTERPRET study", *Diabetes technology & therapeutics*, vol. 15, no. 4, pp. 273–280, 2013.

[135] S. Del Favero, C. Toffanin, L. Magni, and C. Cobelli, "Deployment of modular MPC for type 1 diabetes control: the Italian experience 2008–2016", in *The Artificial Pancreas*, Elsevier, 2019, pp. 153–182.

[136] P. Abuin, P. S. Rivadeneira, A. Ferramosca, and A. H. González, "Artificial pancreas under stable pulsatile MPC: Improving the closed-loop performance", *Journal of Process Control*, vol. 92, pp. 246–260, 2020.

[137] B. Kovatchev, "Automated closed-loop control of diabetes: the artificial pancreas", *Bioelectronic Medicine*, vol. 4, no. 1, pp. 1–12, 2018.

[138] R. Hovorka, V. Canonico, L. J. Chassin, U. Haueter, M. Massi-Benedetti, M. O. Federici, T. R. Pieber, H. C. Schaller, L. Schaupp, T. Vering, *et al.*, "Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes", *Physiological measurement*, vol. 25, no. 4, p. 905, 2004.

[139] D. Boiroux, V. Bátora, M. Hagdrup, S. L. Wendt, N. K. Poulsen, H. Madsen, and J. B. Jørgensen, "Adaptive model predictive control for a dual-hormone artificial pancreas", *Journal of Process Control*, vol. 68, 105–117, 2018.

[140] I. Hajizadeh, M. Rashid, and A. Cinar, "Plasma-insulin-cognizant adaptive model predictive control for artificial pancreas systems", *Journal of process control*, vol. 77, pp. 97–113, 2019.

[141] D. Shi, E. Dassau, and F. J. Doyle, "Adaptive zone model predictive control of artificial pancreas based on glucose-and velocity-dependent control penalties", *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 4, pp. 1045–1054, 2018.

[142] R. Visentin, E. Campos-Náñez, M. Schiavon, D. Lv, M. Vettoretti, M. Breton, B. P. Kovatchev, C. Dalla Man, and C. Cobelli, "The UVA/Padova type 1 diabetes simulator goes from single meal to single day", *Journal of diabetes science and technology*, vol. 12, no. 2, pp. 273–281, 2018.

[143] E. Zijlstra, T. Heise, L. Nosek, L. Heinemann, and S. Heckermann, "Continuous glucose monitoring: quality of hypoglycaemia detection", *Diabetes, Obesity and Metabolism*, vol. 15, no. 2, pp. 130–135, 2013.

[144] C. Toffanin, S. Del Favero, E. Aiello, M. Messori, C. Cobelli, and L. Magni, "Glucose-insulin model identified in free-living conditions for hypoglycaemia prevention", *Journal of Process Control*, vol. 64, pp. 27–36, 2018.

[145] C. Toffanin, E. M. Aiello, C. Cobelli, and L. Magni, "Hypoglycemia prevention via personalized glucose-insulin models identified in free-living conditions", *Journal of Diabetes Science and Technology*, vol. 13, no. 6, pp. 1008–1016, 2019.

[146] A. Weisman, J.-W. Bai, M. Cardinez, C. K. Kramer, and B. A. Perkins, "Effect of artificial pancreas systems on glycaemic control in patients with type 1 diabetes: a systematic review and meta-analysis of outpatient randomised controlled trials", *The lancet Diabetes & endocrinology*, vol. 5, no. 7, pp. 501–512, 2017.

[147] C. Pérez-Gandía, A. Facchinetti, G. Sparacino, C. Cobelli, E. Gómez, M. Rigla, A. de Leiva, and M. Hernando, "Artificial neural network algorithm for online glucose prediction from continuous glucose monitoring", *Diabetes technology & therapeutics*, vol. 12, no. 1, pp. 81–88, 2010.

[148] J. Nordhaug Myhre, M. Tejedor, I. Kalervo Launonen, A. El Fathi, and F. Godtliebsen, "In-Silico Evaluation of Glucose Regulation Using Policy Gradient Reinforcement Learning for Patients with Type 1 Diabetes Mellitus", *Applied Sciences*, vol. 10, no. 18, p. 6350, 2020.

[149] T. Zhu, K. Li, P. Herrero, and P. Georgiou, "Deep learning for diabetes: a systematic review", *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 7, pp. 2744–2757, 2020.

[150] Q. Sun, M. V. Jankovic, L. Bally, and S. G. Mougiakakou, "Predicting blood glucose with an lstm and bi-lstm based deep neural network", in *2018 14th Symposium on Neural Networks and Applications (NEUREL)*, IEEE, 2018, pp. 1–5.

[151] A. Aliberti, I. Pupillo, S. Terna, E. Macii, S. Di Cataldo, E. Patti, and A. Acquaviva, "A multi-patient data-driven approach to blood glucose prediction", *IEEE Access*, vol. 7, pp. 69 311–69 325, 2019.

[152] E. M. Aiello, G. Lisanti, L. Magni, M. Musci, and C. Toffanin, "Therapy-driven deep glucose forecasting", *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103 255, 2020.

[153] J. Carrillo-Moreno, C. Pérez-Gandía, R. Sendra-Arranz, G. García-Sáez, M. E. Hernando, and Á. Gutiérrez, "Long short-term memory neural network for glucose prediction", *Neural Computing and Applications*, vol. 33, no. 9, pp. 4191–4203, 2021.

[154] S. Mirshekarian, H. Shen, R. Bunescu, and C. Marling, "LSTMs and neural attention models for blood glucose prediction: Comparative experiments on real and synthetic data", in *2019 41st annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, IEEE, 2019, pp. 706–712.

[155] M. F. Rabby, Y. Tu, M. I. Hossen, I. Lee, A. S. Maida, and X. Hei, "Stacked LSTM based deep recurrent neural network with kalman smoothing for blood glucose prediction", *BMC Medical Informatics and Decision Making*, vol. 21, no. 1, pp. 1–15, 2021.

[156] A. Gani, A. V. Gribok, Y. Lu, W. K. Ward, R. A. Vigersky, and J. Reifman, "Universal glucose models for predicting subcutaneous glucose concentration in humans", *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 1, pp. 157–165, 2009.

[157] F. Allam, Z. Nossai, H. Gomma, I. Ibrahim, and M. Abdelsalam, "A recurrent neural network approach for predicting glucose concentration in type-1 diabetic patients", in *Engineering applications of neural networks*, Springer, 2011, pp. 254–259.

[158] C. Zecchin, A. Facchinetti, G. Sparacino, and C. Cobelli, "How much is short-term glucose prediction in type 1 diabetes improved by adding insulin delivery and meal content information to CGM data? A proof-of-concept study", *Journal of diabetes science and technology*, vol. 10, no. 5, pp. 1149–1160, 2016.

[159] C. Mosquera-Lopez, R. Dodier, N. Tyler, N. Resalat, and P. Jacobs, "Leveraging a Big Dataset to Develop a Recurrent Neural Network to Predict Adverse Glycemic Events in Type 1 Diabetes", *IEEE Journal of Biomedical and Health Informatics*, pp. 1–1, 2019. DOI: 10.1109/JBHI.2019.2911701.

[160] C. Marling and R. Bunescu, "The OhioT1DM dataset for blood glucose level prediction: Update 2020", in *CEUR workshop proceedings*, NIH Public Access, vol. 2675, 2020, p. 71.

[161] R. Visentin, C. Dalla Man, B. Kovatchev, and C. Cobelli, "The university of Virginia/Padova type 1 diabetes simulator matches the glucose traces of a clinical trial", *Diabetes technology & therapeutics*, vol. 16, no. 7, pp. 428–434, 2014.

[162] C. Toffanin, M. Messori, F. Di Palma, G. De Nicolao, C. Cobelli, and L. Magni, *Artificial pancreas: model predictive control design from clinical experience*, 2013.

[163] E. M. Aiello, C. Toffanin, L. Magni, and G. De Nicolao, "Model-based identification of eating behavioral patterns in populations with type 1 diabetes", *Control Engineering Practice*, vol. 123, p. 105 128, 2022.

[164] S. Del Favero, A. Facchinetti, G. Sparacino, and C. Cobelli, "Improving accuracy and precision of glucose sensor profiles: retrospective fitting by constrained deconvolution", *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 4, pp. 1044–1053, 2013.

[165] O. M. Staal, S. Sælid, A. Fougner, and Ø. Stavdahl, "Kalman smoothing for objective and automatic preprocessing of glucose data", *IEEE journal of biomedical and health informatics*, vol. 23, no. 1, pp. 218–226, 2018.

[166] O. M. Staal, *Kalman-smoothing-glucose, GitHub repository*, `https://github.com/omstaal/kalman-smoothing-glucose`, 2017.

[167] F. Gustafsson, *Statistical sensor fusion*. Studentlitteratur, 2010.

[168] F. Prendin, S. Del Favero, M. Vettoretti, G. Sparacino, and A. Facchinetti, "Forecasting of Glucose Levels and Hypoglycemic Events: Head-to-Head Comparison of Linear and Nonlinear Data-Driven Algorithms Based on Continuous Glucose Monitoring Data Only", *Sensors*, vol. 21, no. 5, p. 1647, 2021.

[169] G. Sparacino, F. Zanderigo, S. Corazza, A. Maran, A. Facchinetti, and C. Cobelli, "Glucose concentration can be predicted ahead in time from continuous glucose monitoring sensor time-series", *IEEE Transactions on biomedical engineering*, vol. 54, no. 5, pp. 931–937, 2007.

[170] C. Dalla Man, F. Micheletto, D. Lv, M. Breton, B. Kovatchev, and C. Cobelli, "The UVA/Padova type 1 diabetes simulator: new features", *Journal of diabetes science and technology*, vol. 8, no. 1, pp. 26–34, 2014.

[171] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets", *PloS one*, vol. 10, no. 3, e0118432, 2015.

[172] A. D. Association, "Physical activity/exercise and diabetes mellitus", *Diabetes care*, vol. 26, no. suppl_1, s73–s77, 2003.

[173] C. Cobelli, C. Dalla Man, G. Sparacino, L. Magni, G. De Nicolao, and B. P. Kovatchev, "Diabetes: models, signals, and control", *IEEE reviews in biomedical engineering*, vol. 2, pp. 54–96, 2009.

[174] C. Dalla Man, R. A. Rizza, and C. Cobelli, "Meal simulation model of the glucose-insulin system", *IEEE Transactions on biomedical engineering*, vol. 54, no. 10, pp. 1740–1749, 2007.

[175] C. Toffanin, R. Visentin, M. Messori, F. Di Palma, L. Magni, and C. Cobelli, "Toward a run-to-run adaptive artificial pancreas: In silico results", *IEEE Transactions on Biomedical Engineering*, vol. 65, no. 3, pp. 479–488, 2017.

[176] Y. M. Luijf, J. H. DeVries, K. Zwinderman, L. Leelarathna, M. Nodale, K. Caldwell, K. Kumareswaran, D. Elleri, J. M. Allen, M. E. Wilinska, *et al.*, "Day and night closed-loop control in adults with type 1 diabetes: a comparison of two closed-loop algorithms driving continuous subcutaneous insulin infusion versus patient self-management", *Diabetes care*, vol. 36, no. 12, pp. 3882–3887, 2013.

[177] L. Magni, D. M. Raimondo, L. Bossi, C. Dalla Man, G. De Nicolao, B. Kovatchev, and C. Cobelli, *Model predictive control of type 1 diabetes: an in silico trial*, 2007.

[178] P. Soru, G. De Nicolao, C. Toffanin, C. Dalla Man, C. Cobelli, L. Magni, A. H. Consortium, *et al.*, "MPC based artificial pancreas: strategies for individualization and meal compensation", *Annual Reviews in Control*, vol. 36, no. 1, pp. 118–128, 2012.

[179] J. Kropff, S. Del Favero, J. Place, C. Toffanin, R. Visentin, M. Monaro, M. Messori, F. Di Palma, G. Lanzola, A. Farret, *et al.*, "2 month evening and night closed-loop glucose control in patients with type 1 diabetes under free-living conditions: a randomised crossover trial", *The lancet Diabetes & endocrinology*, vol. 3, no. 12, pp. 939–947, 2015.

[180] E. Renard, A. Farret, J. Kropff, D. Bruttomesso, M. Messori, J. Place, R. Visentin, R. Calore, C. Toffanin, F. Di Palma, *et al.*, "Day-and-night closed-loop glucose control in patients with type 1 diabetes under free-living conditions: results of a single-arm 1-month experience compared with a previously reported feasibility study of evening and night at home", *Diabetes Care*, vol. 39, no. 7, pp. 1151–1160, 2016.

[181] M. Messori, J. Kropff, S. Del Favero, J. Place, R. Visentin, R. Calore, C. Toffanin, F. Di Palma, G. Lanzola, A. Farret, *et al.*, "Individually adaptive artificial pancreas in subjects with type 1 diabetes: a one-month proof-of-concept trial in free-living conditions", *Diabetes Technology & Therapeutics*, vol. 19, no. 10, pp. 560–571, 2017.

[182] N. Magdelaine, L. Chaillous, I. Guilhem, J.-Y. Poirier, M. Krempf, C. H. Moog, and E. Le Carpentier, "A long-term model of the glucose–insulin dynamics of type 1 diabetes", *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 6, pp. 1546–1552, 2015.