



UNIVERSITY OF PAVIA

Department of Economics and Management

UNIVERSITY OF BERGAMO

Department of Economics

Deep Learning and Evolutionary Optimization for Financial Forecasting with Echo State Networks

AUTHOR: SOFIA GIANTSIDI

SUPERVISOR: CLAUDIA TARANTOLA

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy - September 2025

Abstract

Financial time series are inherently noisy, nonlinear, and non-stationary, shaped by a wide range of economic and societal factors, which makes accurate forecasting exceptionally challenging. Traditional statistical approaches, relying on econometric assumptions, often struggle to capture these complex dynamics. In contrast, deep learning (DL) methods, particularly recurrent architectures, have demonstrated superior predictive power. However, these models face challenges such as high training complexity and vanishing/exploding gradients. Echo State Networks (ESNs), a more recent recurrent paradigm, address many of these issues through a lightweight and computationally efficient design, showing promising results in financial forecasting.

This thesis advances the application of ESNs for financial time series forecasting by focusing on their reservoir, the core element projecting inputs into a high-dimensional dynamic state space. The reservoir is inherently black-box and susceptible to chaotic dynamics, which, if not properly managed, can undermine stability and predictive accuracy. To improve ESN performance, this research develops a principled optimization strategy that accounts for these dynamics. Specifically, a novel hybrid metaheuristic, the Self-Adaptive Cuckoo Genetic Algorithm (SACGA), is proposed. SACGA combines genetic algorithms with adaptive mutation and Lévy flight mechanisms, enabling efficient exploration of the ESN hyperparameter space while balancing forecasting accuracy with reservoir stability.

A major contribution of this work is a systematic review of 187 Scopus-indexed studies (2020–2024) on DL applications in financial forecasting. The review categorizes approaches across domains, compares key architectures, and highlights persistent challenges, providing a consolidated knowledge base for researchers and practitioners.

Extensive experiments on both synthetic chaotic benchmarks and real-world financial datasets demonstrate that SACGA-optimized ESNs consistently outperform standard models, exhibiting improved stability and interpretability. Chaos- and dynamical-system-inspired diagnostics, including Lyapunov exponents, input separability, and kernel PCA projections, show that optimized reservoirs form well-defined, low-dimensional structures, reflecting stable, organized internal dynamics with good generalizability. In contrast, random (unoptimized) reservoirs display scattered, disordered patterns (blobs), with unstable behavior. These findings support the central hypothesis of the thesis: that harnessing, rather than overlooking

or mitigating intrinsic reservoir chaos can improve forecasting performance.

Overall, this thesis contributes (i) a comprehensive review of deep learning in financial forecasting, (ii) a chaos-informed perspective on ESN reservoirs, (iii) the development of SACGA as a novel hybrid optimization algorithm tailored to ESNs, and (iv) diagnostic tools for reservoir analysis. Together, these contributions advance both the theoretical understanding and the practical application of ESNs in financial prediction.

Acknowledgements

I would like to express my sincere gratitude to my project supervisor, Professor Claudia Tarantola, for her invaluable intellectual guidance, encouragement, and for consistently suggesting new avenues of professional development, including opportunities at conferences and academic networking.

I am also deeply grateful to Professor Petros Dellaportas for his insightful suggestions, constructive feedback on analyses, and engaging discussions throughout this study. I am particularly thankful to him for facilitating my research stay at the Athens University of Economics and Business (AUEB). I would also like to sincerely thank him for introducing me to Jacopo Capra, who generously provided the high-frequency data that was crucial for my study, enabling me to conduct a more comprehensive and rigorous analysis.

I wish to thank the Department of Statistics at the AUEB, including the Head of Department, faculty members, administrative staff, technical staff, and fellow PhD students. Their support, guidance, and hospitality during my period abroad were a tremendous source of inspiration and greatly enriched my research experience.

Finally, I am profoundly grateful to my family. Their unwavering love, support, and encouragement have provided me with the strength and motivation to fulfill my academic obligations and complete this work.

Contents

Abstract	4
Acknowledgements	5
1 Introduction	13
1.1 Background and motivation	13
1.2 Research objectives and contributions	14
1.3 Structure of the Thesis	15
2 Deep Learning for Financial Forecasting: A Review of Recent Trends	17
2.1 Motivation, Scope, and Research Design.	17
2.2 Existing Surveys	20
2.3 Deep Learning (DL)	26
2.3.1 Deep Multi-Layer Perceptron (DMLP)	27
2.3.2 Recurrent neural network (RNN)	28
Long Short-Term Memory (LSTM)	29
Bidirectional Long Short-Term Memory (BiLSTM)	30
Gated Recurrent Units (GRUs)	30
Reservoir Computing (RC)	32
2.3.3 Convolutional Neural Networks (CNNs)	33
Temporal Convolutional Network (TCN)	35
2.3.4 Autoencoder (AE)	35
2.3.5 Other deep structures	37
2.3.6 Hybrids	38
2.4 Hyperparameters Tuning	39
2.5 Financial Time series Forecasting.	39
2.5.1 Stock Price Forecasting	40
Standalone Architectures	41
Comparative Performance of RNN Variants	41
Novel Model Architectures	41
Integrating Technical and Sentiment Analysis	41
Other Use Cases	42

	Hybrid Architectures	42
	Fusion of CNNs, RNNs, and Autoencoders	42
	Novel Model Architectures	42
	Enhancing Prediction via Dimensionality Reduction	43
	Optimization-Integrated Models	43
	Integrating Technical and Sentiment Analysis	44
	Financial Forecasting in Pandemic Contexts	45
	Addressing Forecasting Challenges with Specialized Techniques	45
	Summary	46
	Cross-study Observations	46
2.5.2	Index Forecasting	49
	Standalone Architectures	49
	LSTM-Based Approaches	49
	Beyond LSTM Architectures	49
	Integrating Technical Analysis	50
	Hybrid Models	50
	Fusions of LSTM and Beyond	50
	Dimensionality Reduction for Forecast Enhancement	50
	Novel Model Architectures	51
	Signal Decomposition and Denoising Architectures	53
	Optimization-based Architectures	53
	Integrating Technical and Sentiment Analysis	54
	Financial Forecasting in Crisis and Pandemic Contexts	55
	Advanced Modeling Techniques and Approaches	55
	Summary	56
	Cross-study Observations	57
2.5.3	Forex Price Forecasting	59
	CNN-Based Architectures	59
	Fusions of LSTM and Beyond	59
	Chaos-based Advanced Modeling Techniques	59
	Summary	60
	Cross-study Observations	60
2.5.4	Commodity Price Forecasting	61
	Leveraging Recurrent Architectures	62
	Integrating Reinforcement Learning and Interpretability	62
	Chaos-based Advanced Modeling Techniques	62

2.5.5	Bond Price Forecasting	63
2.5.6	Cryptocurrency Price Forecasting	63
	GRU-Based Forecasting	64
	LSTM and Temporal Modeling	64
	Alternative Models: FB-Prophet and ARIMA	64
	Hybrid Architectures and Advanced Modeling Techniques	64
	Novel Hybrid Approaches	65
	Summary	66
	Cross-study Observations	66
2.5.7	Volatility Forecasting	68
	Standalone LSTM for Market Volatility	68
	Hybrid Architectures for Stock Volatility	68
	Hybrid Architectures for Index Volatility	69
	Hybrid Architectures to Commodity Volatility	69
2.5.8	Other Works	70
2.6	Current snapshot of DL research for financial applications	71
2.7	Discussion and Open Issues	76
2.7.1	Dominance and Underutilization of RNN Variants	76
2.7.2	Feature Selection and Multimodal Data Integration	77
2.7.3	Asset Focus and Emerging Domains	77
2.7.4	Applications Beyond Forecasting	77
2.7.5	Handling Market Shocks and Global Crises	78
2.7.6	Quantum Computing: A Futuristic Promise	78
2.7.7	Interpretability and Explainability Challenges	79
2.7.8	Lightweight Models for Real-World Deployment	79
2.7.9	Need for Standards in Evaluation	80
2.7.10	A Design-Principles Taxonomy	80
2.7.11	Critical Methodological Gaps	81
2.7.12	Initial Questions Answered	81
2.8	Concluding Remarks	84
3	Methodology	85
3.1	Echo State Networks:	
	Architecture and Dynamic Behavior	85
3.1.1	Overview of ESNs	85
3.1.2	Mathematical Formulation	87
3.1.3	Algorithm	89
3.1.4	Open-Source ML Frameworks for ESN Implementation	89
3.1.5	Hyperparameters in ESN Models	89
3.1.6	Fundamental ESN Properties	92
3.1.7	Probing Inner Chaos in Reservoir Dynamics	92
	Attractors: Order Amidst Chaos	93
	Edge of Chaos	95
	Estimation of the largest Lyapunov exponent	97
	Theoretical Basis	99
	Practical Estimation	100

	Algorithm	101
	Separation Ratio	101
	Controlled Chaos	102
	Interpreting Reservoir Dynamics through Kernel PCA	
	Visualization	104
	Related Work	104
	Theoretical Basis	106
	Implementation.	107
	Implications	108
3.1.8	Limitations in ESN Design	109
3.1.9	Existing Optimization Approaches for ESNs	111
3.2	Self-Adaptive-Cuckoo-Genetic Algorithm –	
	A Novel Optimization Algorithm.	114
3.2.1	Motivation for SACGA	114
3.2.2	Theoretical foundation	115
	Genetic Algorithm	115
	Self-Adaptation.	119
	Cuckoo Search Algorithm	122
3.2.3	Methodology of SACGA	124
	Overview of SACGA	124
	Cuckoo Offspring	126
	Separation Metric.	127
	GA and Self-Adaptive Mutation Mechanism	127
	Fitness Evaluation Process.	128
	Integration with Grid Search	129
	Computational Complexity of SACGA Compared to Genetic Algorithms and Cuckoo Search.	130
3.2.4	Algorithm	131
3.2.5	Expected advantages over existing approaches	131
4	Findings and Discussion	137
4.1	Experimental Validation on Benchmark Datasets	137
4.1.1	Experimental Design	137
4.1.2	Description of Benchmark Datasets	139
4.1.3	Comparative Methods	140
4.1.4	Performance Analysis	141
4.1.5	Discussion.	143
4.1.6	Concluding Remarks	146
4.2	Real-World Application in Financial Forecasting	147
4.2.1	Proposed Methodology	147
	Experimental Design	150
	Model Performance Metrics	151
	Hardware and Software Environment.	152
4.2.2	Dataset Description	152
	Feature Description	155
	Description of Selected Features	156

4.2.3	Comparative Method	158
4.2.4	Performance Analysis	159
	Best Individual Characterization	159
	Regression & Classification Metrics	160
	Reservoir Stability: Probing Inner Chaos	162
	Quantitative Analysis	162
	Qualitative Analysis	165
4.2.5	Concluding Remarks	168
5	Conclusion and Future Directions	171
	Appendices	180
.1	Technical Summary of Reviewed Studies for Chapter 2	180
.2	Derivation of Ridge Regression Solution	212
.3	Technical Features	213
.4	Summary of the Random Parameter Sets for ESN Initialization for Chapter 4	215
.5	Glossary	218
	List of Figures	222
	List of Tables	226

Contents

Chapter 1

Introduction

This chapter provides an overview of the research context, beginning with the background and motivation for the study, followed by a detailed statement of the research objectives and contributions, and concluding with an outline of the thesis structure.

1.1 Background and motivation

Financial markets are like living, breathing ecosystems, messy, unpredictable, and constantly in motion. They're shaped by a tangle of factors: economic signals, investor psychology, breaking news, and raw speculation, all generating massive streams of high-frequency, noisy data. Trying to forecast these time series remains a critical challenge with profound implications for investment decisions, risk management, and economic policy formulation. Traditional statistical models often fall short, unable to capture the intricate temporal dependencies and chaotic behaviors exhibited by market data. Consequently, the emergence of deep learning (DL) techniques has revolutionized financial forecasting by enabling models to learn complex patterns directly from raw data.

Among various DL architectures, Recurrent Neural Networks (RNNs) and their variants, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs), have dominated research due to their ability to model sequential dependencies. However, these models are often associated with substantial training complexity, including challenges related to gradient descent optimization issues, as well as considerable computational demands. Another recurrent architecture, Echo State Networks (ESNs), a form of reservoir computing, offers a compelling alternative. ESNs employ a fixed, sparsely connected, and randomly initialized reservoir, with only the output weights subject to training, typically through ridge regression. This structural simplicity enables faster convergence and significantly reduces computational overhead, rendering ESNs an efficient solution for capturing temporal dynamics in sequential data.

Despite these advantages, ESNs face significant challenges, particularly due to their black-box nature, sensitivity to hyperparameter tuning, and the complexity

of reservoir design. These issues can have a critical impact on both forecasting accuracy and model stability. Most existing optimization methods tend to focus on only a subset of hyperparameters, thereby reducing the complexity of the search space but often at the cost of overlooking important interactions within the reservoir. Additionally, the standard cost functions commonly used prioritize predictive error minimization without accounting for the inherently chaotic dynamics within the reservoir. This can result in suboptimal configurations that compromise performance. Effectively managing the chaotic behavior of the reservoir's inner workings remains essential for maintaining consistent and reliable predictive outcomes.

This thesis is motivated by the need to enhance ESN optimization through advanced metaheuristic algorithms that adaptively navigate the complex hyperparameter space. Drawing inspiration from the strengths of nature-inspired optimization techniques, this work proposes a novel Self-Adaptive Cuckoo-Genetic Algorithm (SACGA). The proposed approach integrates the global search strengths of Genetic Algorithms, with a self-adaptive mutation rate mechanism, dynamically adjusted based on predictive performance and reservoir quality metrics. Additionally, the incorporation of Levy flights, inspired by the foraging behavior of cuckoos, facilitates extensive exploration of the search space boundaries. This hybrid strategy aims to deliver robust reservoir optimization, ultimately enhancing the accuracy and stability of financial forecasting models.

In essence, this work emerges from the recognition of the untapped potential within the relatively simple yet powerful ESN architecture. The enigmatic, black-box nature of ESNs presents a compelling challenge, highlighting the urgent need to transcend current optimization limitations and to develop chaos-informed methodologies that unravel the reservoirs' complex nonlinear dynamics. At the intersection of evolutionary optimization and deep learning paradigms, this work seeks to bridge the gap between theory and application, enhancing understanding and delivering diagnostic tools for ESN behavior, thereby advancing their use in the demanding domain of financial forecasting.

1.2 Research objectives and contributions

This section outlines the key objectives driving this research and highlights the main contributions made toward advancing the understanding and application of deep learning methods, particularly Echo State Networks (ESNs), in financial forecasting.

Research Objectives The primary objectives of this research are to:

- Conduct a systematic review of deep learning methods in financial forecasting, establishing a clear understanding of the current landscape.
- Explore the complex dynamics of ESNs, focusing on chaos and interpretability of reservoir behavior.

- Develop and implement an advanced optimization algorithm tailored for ESNs to enhance their forecasting capabilities.
- Validate the proposed optimized-predictive framework on benchmark datasets as well as real-world financial data.

Research Contributions This work makes several significant contributions:

- **Comprehensive Review:** A comprehensive and systematic literature review of deep learning approaches for financial forecasting (2020–2024), categorizing 187 Scopus-indexed studies by forecasting domain. The review presents comparative insights into key DL models (e.g., MLP, RNN, LSTM, GRU, RC/ESN, CNN, AE, GAN), highlights domain-specific challenges and model preferences, and identifies open issues and future research directions. This work offers a consolidated knowledge base for researchers and practitioners alike.
- **Reservoir Chaos Perspective:** A novel perspective on ESN architecture through the lens of chaos theory. Rather than suppressing chaotic behavior, the study explores how intrinsic chaos in the reservoir can be manifested and harnessed to enhance input encoding, enrich dynamic representations, and support stable and reliable predictive performance.
- **Optimization Innovation:** The introduction of a novel hybrid metaheuristic, the Self-Adaptive Cuckoo Genetic Algorithm (SACGA), specifically designed to optimize ESNs by balancing predictive accuracy with dynamic reservoir quality.
- **Forecasting Performance:** Anticipated improvements in forecasting performance, evaluated on both synthetic chaotic benchmarks and real-world financial time series, to demonstrate the effectiveness and robustness of the proposed approach.
- **Reservoir Diagnostics:** An interpretability framework leveraging chaos and topological tools (e.g., Lyapunov Exponents, Separation Ratio, kernel PCA) to analyze ESN internal dynamics, in terms of system stability, memory capacity, and separability, offering deeper understanding of ESN behavior in financial contexts.

1.3 Structure of the Thesis

This thesis adopts a monographic format, consisting of thematically interconnected chapters. Chapter 2 presents a comprehensive literature review of deep learning applications in financial forecasting, based on an analysis of 187 Scopus-indexed papers published between 2020 and 2024. Chapter 3 outlines the methodological framework, detailing the fundamentals of Echo State Networks, and introducing a novel hybrid optimization algorithm specifically designed for ESNs, termed the

Self-Adaptive-Cuckoo-Genetic Algorithm (SACGA).

The empirical findings and discussion are encompassed within the subsequent part of the thesis, divided into two complementary contexts to provide a thorough evaluation of the proposed approach. Chapter 4.1 focuses on experimental validation using well-established synthetic benchmark datasets. In this chapter, the SACGA optimized ESN is rigorously assessed and benchmarked against other state-of-the-art ESN optimization methods, demonstrating its effectiveness in controlled environments.

Building upon these findings, Chapter 4.2 applies the proposed methodology to a real-world financial forecasting scenario involving stock market data. Here, the SACGA-optimized ESN model is evaluated against the Random Walk model, which serves as a fundamental yet robust baseline in financial forecasting. This chapter offers both quantitative and qualitative analyses, delivering critical insights into the role of intrinsic chaotic dynamics for achieving reliable, stable, and accurate predictive performance.

The thesis concludes by summarizing the key contributions and outlining directions for future research. A glossary is also included to assist readers with important terms and concepts, complemented by appendices containing supplementary information.

Chapter 2

Deep Learning for Financial Forecasting: A Review of Recent Trends

This chapter presents a systematic literature review on the application of Deep Learning (DL) techniques to financial time series forecasting. It aims to synthesize the rapid advancements in DL model design, application domains, and methodological practices between 2020 and 2024. The review begins by outlining the motivation, scope, and research design (Section 2.1), followed by a critical examination of prior surveys on DL in financial forecasting to contextualize the need for this study. Section 2.3 introduces core DL architectures, including DMLPs, RNN variants (LSTM, GRU, RC), CNNs, Autoencoders, and hybrid models, followed by an overview of the most common hyperparameter optimization strategies. Subsequently, the chapter systematically evaluates DL applications across major financial forecasting domains: stock, index, forex, commodity, bond, cryptocurrency, and volatility forecasting. Each domain-specific section outlines key modeling innovations, hybrid architectures, data integration strategies, and emerging challenges, concluding with summary insights and cross-study observations. The latter part of the chapter offers a snapshot of recent DL research trends in financial forecasting (Section 2.6), followed by an in-depth discussion of key challenges and open issues (Section 2.7), including a design-principles taxonomy. The chapter concludes with the final remarks.

Through this structured synthesis, the chapter identifies critical areas for improvement and provides guidance for future DL research in financial forecasting.

2.1 Motivation, Scope, and Research Design

Motivation The field of financial forecasting has witnessed a transformative shift with the advent of DL techniques, which offer powerful tools for analyzing complex patterns within financial time series data. Traditional forecasting methods relying on econometric assumptions often struggle to capture the nonlinear relationships and intricate dynamics in financial markets. However, DL algorithms have emerged as robust alternatives. This chapter provides a comprehensive and up-to-date review of DL model implementations across a wide range of financial forecasting do-

mains.

The fundamental motivation was to answer the following:

- Which DL models are preferred (and more successful) in different applications?
- What challenges do DL models face when applied to financial time series data?
- What is the future direction for DL research in financial forecasting?

To address these questions, we define the scope of our review as follows.

Scope Despite the growing adoption of DL in finance, existing surveys have several limitations. Most reviews focus on studies published prior to 2020, concentrate on specific asset classes or model families, or primarily provide descriptive overviews without systematic evaluation. As a result, there remains a need for a comprehensive, up-to-date synthesis that maps recent advancements, highlights prevailing trends, and identifies methodological challenges across financial forecasting domains.

This paper provides a systematic review of 187 Scopus-indexed studies on DL applications in financial time series forecasting, published between 2020 and 2024. The goal is to offer a comprehensive and holistic overview of recent advancements in DL-based financial forecasting. To the best of our knowledge, this study represents one of the first systematic surveys dedicated exclusively to literature from this period. The primary objectives of this review are the following:

- **Comprehensive coverage of financial domains:** We analyze and categorize 187 Scopus-indexed studies spanning stocks, indices, forex, commodities, bonds, cryptocurrencies, and volatility forecasting, providing a detailed overview of DL adoption in each domain.
- **Systematic taxonomy of models and methodologies:** In each financial forecasting domain, DL models are organized into standalone and hybrid architectures. We further examine feature integration (technical indicators, sentiment analysis, and alternative data), preprocessing techniques (dimensionality reduction, decomposition, denoising), optimization strategies, and novel methodological contributions.
- **Cross-study quantitative synthesis:** We tabulate representative studies by model type, dataset, feature set, performance metrics, and development environment, enabling clear comparisons and highlighting patterns in data usage, feature engineering, and methodological choices. Cross-study statistics summarize the findings within each forecasting domain.

- **Identification of trends, gaps, and actionable insights:** We identify prevailing trends, persistent challenges, methodological gaps, and emerging opportunities in financial DL research. Building on a design-principles taxonomy, this analysis shifts the focus from model labels to architectural rationale, providing actionable recommendations for developing robust, interpretable, and efficient context-aware DL models. The review also outlines clear directions for future research to advance the field.

Research Design To systematically collect relevant studies on DL models in financial forecasting, a structured search is conducted in Scopus database. The search was performed within the fields of *Article Title*, *Abstract*, and *Keywords* to ensure the retrieval of publications explicitly addressing the intersection of DL and financial forecasting.

The query included the term "*deep learning*" in conjunction with financial and predictive terms ("*financ**", "*forecast**", "*predict**") to capture studies within the broader domain of financial forecasting. To focus on DL architectures commonly used in time series forecasting, we included a set of specific model keywords: "*ANN*", "**MLP*", "*RNN*", "*LSTM*", "*RC*", "*ESN*", "*CNN*", "*Autoencoders*", "*GRU*", and "*GAN*". Furthermore, the query was restricted to studies applied to financial instruments, incorporating terms such as "*stock*", "*index*", "*forex*", "*commodity*", "*bond*", and "*cryptocurrency*".

The search was limited to publications from 2020 to 2024. Additionally, to maintain relevance to the study's focus, only articles classified under the subject areas of *Computer Science*, *Engineering*, *Mathematics*, *Economics*, *Econometrics*, and *Finance* were considered. The final dataset was filtered to include only *English language journal articles*, excluding conference papers, reviews, and other document types.

A total of 211 research articles were initially collected. After excluding two review papers and filtering out studies that did not align with the scope of financial forecasting, 187 papers were selected, summarized, and categorized based on the application areas, including stock price forecasting, index forecasting, forex forecasting, commodity forecasting, bond forecasting, cryptocurrency forecasting, and volatility forecasting. The exclusions were due to factors such as restricted access, a focus on non-forecasting financial tasks (e.g., asset allocation, pricing, credit risk assessment, and corporate financial performance), non-financial forecasting tasks (e.g., NLP tasks, fraud detection, material demand forecasting, and agricultural exports prediction), or the use of non-relevant data types (e.g., accounting data, loan data, systemic risk indicators, sales demand forecast data, and material information). Although survey papers were not included in this analysis, two review publications, Sezer et al. [335] and Ma'arif et al. [250], were identified during the paper collection process and are discussed in the following section on existing surveys.

It is important to note that this study's inclusion and exclusion criteria introduce certain limitations. By restricting the dataset to peer-reviewed journal articles in English and excluding conference papers, gray literature, and preprints, some rel-

evant findings may not have been captured. Similarly, the focus on specific model keywords and financial instruments may have omitted studies using alternative DL architectures or applied to less common asset classes. Readers should interpret the findings with awareness of these selection boundaries.

2.2 Existing Surveys

Computational intelligence has long been applied in finance, with DL models emerging as particularly powerful tools, often surpassing traditional ML approaches. To date, a large body of surveys has attempted to consolidate progress in this area. Below, we discuss the most influential contributions, highlighting both their insights and their limitations.

One of the earliest comprehensive reviews bridging ML and DL, is by Rundo et al. [316], who span statistical models, ML techniques, DL architectures, and hybrids, while touching on emerging areas like reinforcement learning, sentiment analysis, and quantum finance. Their contribution lies in cataloging the most commonly used datasets in financial forecasting, namely BSE, S&P 500, DAX, NASDAQ, SSE, and in flagging the lack of standardized benchmarks and consistent evaluation metrics, as a central obstacle to reproducibility and fair comparison. However, their analysis is largely descriptive, lacking algorithmic formulations with technical depth and little discussion of key challenges such as overfitting or interpretability. A more focused historical mapping is offered by Durairaj and Mohan [96], who review 34 DL-based hybrid models between 1999 and 2019. They note the dominance of MLP-based hybrids compared to the fewer LSTM- or CNN-based architectures, highlight the use of evolutionary algorithms for network optimization, and note that most hybrids outperform standalone models. Yet, while their taxonomy is useful as a historical and methodological guide with chronological mapping of hybrid approaches, the review falls short in critically evaluating practical performance.

Among the most important contributions to the field are those of Ozbayoglu et al. [286] and Sezer et al. [335]. Ozbayoglu et al. [286] map DL architectures to applications ranging across algorithmic trading, risk assessment, portfolio management, fraud detection, derivatives markets. They emphasize the growing adoption of DL, especially in algorithmic trading and risk assessment, while highlighting trends like CNN–LSTM hybrids, and NLP ensembles for financial sentiment analysis, deep reinforcement learning, and cryptocurrencies. While they point to future directions such as cryptocurrencies, robo-advisory, sentiment analysis, high-frequency trading, and the potential of specific DL paradigms (e.g., 2D-CNNs, graph-CNNs, GANs, and transfer learning), their review overlooks methodological limitations and regulatory implications. Sezer et al. [335], deliver a systematic literature review of 140 studies on DL methods for financial time series forecasting (2005–2019). They detail architectures, datasets, features, forecasting horizons, performance metrics, and development frameworks across asset types. The

review finds that stock, trend, and index forecasting dominate DL research, with LSTMs leading due to their temporal modeling strengths, CNNs and DMLPs used mainly for classification, often requiring careful preprocessing or feature engineering, and a rising interest in deep reinforcement learning and cryptocurrency forecasting. However, they also caution against publication bias favoring DL, as newer DL studies may receive more attention. Future directions in hybrid DL–NLP, sentiment analysis, and adaptive reinforcement learning approaches are highlighted. These two works set the foundation for the current review, but both remain limited to pre-2020 studies.

Addressing the reproducibility concerns raised by earlier works, Buczyński et al. [61] undertake a rare large-scale replication of 15 DL architectures across diverse assets and market periods comparing them against classical benchmarks such as naive forecasts, ARIMA, and exponential smoothing. Their findings are sobering: naive forecasts often rival DL methods in the long run, while only multi-scale recurrent convolutional neural network and ModAugNet architectures show short-term advantages. Accuracy varies by asset class, with currency pairs and indices easier to predict than volatile stocks and cryptocurrencies. The study highlights the fragility of many DL gains, the need for frequent retraining and proper tuning in non-stationary markets, and full methodological transparency, while underscoring ongoing reproducibility challenges. This study represents a landmark contribution, shifting the focus from descriptive cataloguing toward rigorous evaluation.

Several subsequent surveys zoom in on specific model families or methodological issues. Thakkar and Chaudhari [368] conduct a controlled experimental comparison of nine DNN on a unified dataset (S&P 500, 2000–2017) using Independent Component Analysis for feature extraction to reduce reliance on large sets of technical indicators. They find DQNs particularly effective for non-linear dynamics, while LSTM/GRU remain strong baselines. They stress the potential of hybrids and metaheuristic optimization, though their tuning remains exploratory. The need for standardized benchmarking to improve reproducibility and comparability is emphasized. Shah et al. [337] concentrate on ARIMA, CNN, LSTM, and CNN–LSTM hybrids, showing that CNN–LSTM models are particularly effective, and sentiment integration adds a valuable behavioral dimension, enhancing short-term trends. They note key limitations: ARIMA’s linearity assumptions, and outlier sensitivity, LSTM’s vanishing-gradient issues and long runtimes, CNN’s heavy data and computational demands, and hybrid models inheriting these constraints. Hyperparameter tuning and large historical datasets are also critical. Ma’arif et al. [250] focus on LSTM hybrids for stock price prediction (2018–2024), covering optimization algorithms (e.g., GA-LSTM, LSTM-ARO), statistical/econometric techniques, preprocessing (e.g., EMD/EEMD, Wavelet), and feature extraction. GA-LSTM and CNN-LSTM are the most common, with cross-study RMSE comparisons, across diverse datasets and research contexts, identifying GA-LSTM, AGA-LSTM, and ARIMA-LSTM as top performers.

Broader reviews continue to appear, offering updated panoramas. Chen et al.

[71] reviews 85 works from 2015–2023 (identified via Web of Science), covering standalone models (CNN, LSTM, ensemble CNN) and hybrids (CNN-LSTM, LSTM-AM, CNN-LSTM-AM). LSTMs are acknowledged as the de-facto “should-have” element in DL frameworks, while hybrids—particularly CNN-LSTM and CNN-LSTM-AM—generally outperform standalone models. Practical aspects such as preprocessing, feature engineering, transfer learning, ensemble learning, validation, and explainable AI (e.g., LRP, SHAP) are noted, but the discussion is mostly generic. Key challenges include EMH-related delayed predictions, limited accessibility for finance domain experts, insufficient integration of domain knowledge, lack of standardized datasets, and difficulties with real-time or high-frequency prediction. Vuong et al. [391] review 73 Scopus-indexed stock price forecasting studies, cataloging ARIMA, ML, DL, and hybrid methods, and report a steady rise in publications from 2014–2023, with a notable surge in DL and hybrid studies from 2020 to 2023. DL and hybrid models, particularly LSTMs, CNN-LSTMs, transformers, and GANs, excel at capturing non-linear, heterogeneous data, but face data requirements, architectural complexity, and overfitting challenges. The review overlooks alternative data sources, advanced techniques, and excludes non-journal literature, thereby limiting comprehensiveness. Nevertheless, it identifies future directions, such as alternative data integration (e.g., sentiment, news, real-time indicators), AI-driven architectures (such as graph CNN, reinforcement learning, and meta-learning), real-time analysis, and quantum computing. While informative as a descriptive survey, the study’s scope and methodological constraints restrict its ability to provide a practice-oriented evaluation. Mienye et al. [265] provide perhaps the broadest overview of DL models (e.g., RNNs, LSTMs, CNNs, Transformers, GANs, deep RL) and their applications across finance, including trading, risk management, fraud detection, forecasting, and portfolio optimization. The study highlights emerging trends such as explainable AI, transfer learning, federated learning, quantum computing, and ethical AI, while noting challenges like data quality, overfitting, computational complexity, bias, and regulatory concerns. Though lacking quantitative evaluations and empirical validation, the work’s strengths lie in its broad coverage, connecting DL models to specific finance tasks, clear explanations of sophisticated concepts, and awareness of emerging trends and ethical considerations.

Certain reviews target niche but increasingly important aspects. Sonkavde et al. [351] highlight ensemble methods, particularly combinations such as Random Forest, XGBoost, and LSTM, as the most promising, offering superior predictive accuracy. The study underscores the critical role of hyperparameter tuning, proposes integration of fundamental, technical, and sentiment analysis for improved forecasting, and suggests future directions such as trend analysis, and computer vision for pattern identification and candlestick chart pattern analysis. However, the evaluation is limited to two Indian stocks and primarily price-based features, restricting generalizability and overlooking richer auxiliary data sources. Li and Bastos [221] systematically examine DL for technical analysis, showing LSTM dominance (73.5%), often in hybrid use with CNNs or sentiment analysis, but also critiquing the limited attention to profitability assessment (35.3%), risk management (rarely addressed), and trading design adopting simple buy-sell-hold logic, without

adaptive/AI-based approaches, or real-market implementations. While thorough in cataloging DL implementations, the study offers limited critical analysis of performance differences, preprocessing, or hybrid impacts, but stresses the need for algo trading validation, cost-aware profitability, qualitative data, cross-market testing, and clearer guidance on technical indicators. Olorunnimbe and Viktor [282] focus exclusively on studies with proper backtesting, finding that most research centers on trading strategy, price prediction (both often via DRL and LSTM), and portfolio management (with DRL particularly popular), while areas such as market simulation, stock selection, hedging, and risk management receive far less attention. Although several tasks have been framed as online learning, the connection remains largely unestablished, representing an opportunity to advance practice. Key challenges like limited data availability and quality, short-term prediction focus, trading costs neglect, and lack of specialized frameworks for online learning and non-stationary markets are acknowledged, which the authors suggest addressing through explainable models, public data availability, long-horizon perspectives, and dedicated financial DL frameworks to unify and mature the field. Blasco et al. [56] through novel lens focus on uncertainty quantification (UQ). The authors contrast traditional point-prediction DL methods with UQ approaches that generate probabilistic forecasts, distinguishing aleatoric uncertainty, arising from inherent data noise, from epistemic uncertainty, stemming from the model and its specification. They note growing interest in UQ since 2010, peaking in 2019 with 389 studies, with most studies focusing on price accuracy and epistemic uncertainty. Gaps remain in addressing aleatoric uncertainty (or its combination with epistemic), integrating technical, fundamental, and sentiment data, lack of streaming data, multi-step, or trend-based predictions, and underutilization of markets such as Forex. Together, these specialized surveys expand the conversation from raw predictive accuracy to real-world considerations of backtesting, risk, and uncertainty.

Parallel to methodological reviews, a number of surveys emphasize applications. Hu et al. [167] review 86 studies on stock and Forex forecasting, finding LSTM-based models, especially attention-LSTM and CNN-LSTM variants, dominate, while most rely on technical indicators and few incorporate sentiment, news, or expert recommendations. Wang and Yan [397] analyze DL in algorithmic trading, pointing to hybrid CNN-LSTM and DMLP-RL models for trading strategies, and DMLPs for regression and classification tasks with preprocessing. They note growing exploration of reinforcement learning and genetic algorithms for trading strategy development, while emphasizing the importance of parameter optimization, feature selection, and hybrid architectures, and warning on the security risks of widespread algorithmic trading given market fragility. Peng and Yan [295] adopt a data-centric view of financial risk prediction, highlighting the challenges of heterogeneity (structured versus unstructured forms), imbalance (or information asymmetry between companies and stakeholders), and multi-source origins. They identify model interpretability, systemic risk propagation, and limitations of incomplete datasets as key areas for future research. Noting that some studies neglected data imbalance and omitted preprocessing techniques, like undersampling or oversampling, the authors stress the need for careful data handling in financial risk mod-

eling. Finally, Qu et al. [303] review ML/DL for bankruptcy prediction, framing it as a binary classification problem. Despite offering limited methodological insight and practical guidance, the authors emphasize the potential of heterogeneous data sources (e.g., news, reports) and identify model interpretability as an emerging concern.

Building directly on Sezer et al. [335] and Ozbayoglu et al. [286], the present review extends this line of inquiry to the more recent period of 2020–2024. Unlike prior surveys, it not only catalogues model architectures but also systematically maps research across asset classes, with tabulated summaries of datasets, feature sets, methodological approaches, and performance metrics. By integrating insights from various model architectures (e.g., standalone, hybrids, novel), preprocessing techniques, optimization strategies, and application domains, this review moves beyond descriptive reporting toward an analytical synthesis of the field. This approach reframes the literature through a design-principles-based taxonomy, highlighting prevailing trends, methodological challenges, and gaps, and actionable directions for future research in financial forecasting.

Survey	Scope / Period	Models Covered	Main Contributions (as reported)
Rundo et al. [316]	ML + DL in finance (pre-2019)	ARIMA, GARCH, SVM, RF, CNN, RNN, LSTM, hybrids	Broad review of statistical, ML, and DL models; mapping of common datasets; highlights need for benchmark datasets and evaluation metrics.
Durairaj and Mohan [96]	34 hybrid DL models (1999–2019)	MLP-, LSTM-, CNN-based hybrids	Historical overview of MLP-, LSTM-, and CNN-based hybrid DL models; suggests combining DL with ML techniques, fuzzy logic, chaos theory, and evolutionary optimization.
Ozbayoglu et al. [286]	First “deep learning for financial applications” survey	CNN, RNN/LSTM, DMLP, RBM, DBN, autoencoders	Overview of DL architectures; maps DL methods to financial tasks (trading, risk, portfolio, fraud, cryptocurrency); identifies trends in financial sentiment analysis, cryptocurrencies, DRL, and hybrid architectures.
Sezer et al. [335]	First “financial time series forecasting” DL-only survey; 140 DL studies (2005–2019)	CNN, RNN, LSTM, RBM, DBN, autoencoders, DRL	Systematic review with structured taxonomy by asset type; stock, trend, and index forecasting dominate; LSTMs lead, CNNs/DMLPs for classification; rising DRL and cryptocurrency interest.

TABLE 1: Comparison of major surveys on deep learning and financial forecasting: scope, models, and reported contributions (Part I).

2.2. Existing Surveys

Survey	Scope / Period	Models Covered	Main Contributions (as reported)
Buczyński et al. [61]	Replication of 15 DL models	multi-scale recurrent CNN, ModAugNet, ARIMA, naïve, exponential smoothing	Large-scale reproduction across 29 assets and 5 market periods (2007–2022); unified evaluation framework; shows asset-dependent performance, and the need for frequent retraining and proper tuning of DL models.
Thakkar and Chaudhari [368]	Stock price and trend forecasting (2017–2020)	CNN, RNN, LSTM, GRU, ESN, DNN, DQN, RBM, DBN	Panoramic review of DNNs; controlled experimental benchmark of nine models on S&P 500 (2000–2017); finds DQN effective and supports hybrids, and metaheuristic optimization.
Shah et al. [337]	Stock prediction	ARIMA, CNN, LSTM, CNN–LSTM	Comparative review with flowcharts and tabulated summaries; highlights hybrid CNN–LSTM for combining feature extraction and temporal modeling; notes role of sentiment integration.
Ma’arif et al. [250]	LSTM hybrids (2018–2024)	GA-LSTM, PSO-LSTM, APSO-LSTM, CNN-LSTM, ARIMA-LSTM, ARIMA-EGARCH-LSTM, LSTM-EMD, etc.	Yearly classification of LSTM hybrids; tabulated comparison of models, datasets, and metrics; identifies GA-LSTM, AGA-LSTM, and ARIMA-LSTM as top performers.
Chen et al. [71]	85 DL studies (2015–2023)	CNN, LSTM, ensemble CNN, CNN–LSTM, LSTM-AM, CNN–LSTM–AM	Generalized framework for FTS prediction: architecture, training/prediction, hyperparameter tuning, and evaluation metrics; shows hybrids CNN–LSTM, CNN–LSTM–AM outperforming standalone.
Vuong et al. [391]	73 Scopus-indexed stock price forecasting studies (2014–2023)	ARIMA, ML, DL, hybrids	Bibliometric and content analysis with tabulated summaries of methods, datasets, targets, features, metrics, and results; highlights LSTMs, CNN–LSTMs, transformers, and GANs as particularly effective.
Mienye et al. [265]	DL in finance (broad)	feedforward network, RNN, LSTM, GRU, CNN, Transformer, GAN, DRL, DBN	Broad overview linking DL models to applications (trading, risk, fraud, portfolio); highlights trends in XAI, transfer learning, federated learning, quantum computing, and ethical AI.
Sonkavde et al. [351]	Ensemble forecasting	linear regression, KNN, SVM, logistic regression, ARIMA, FB Prophet, LSTM, GRU, RE, XG-Boost	Comprehensive review of ML/DL methods; proposes ensemble (RF + XGBoost + LSTM) with superior performance on Indian stocks; emphasizes hyperparameter tuning.
Li and Bastos [221]	DL + technical analysis (2017–2020)	34 DL models	Systematic review of DL for TA; analysis of predictor techniques, trading strategies, profitability metrics, and risk management; shows LSTM dominance and notes limited trading-strategy sophistication.
Olorunnimbe and Viktor [282]	DL with backtesting (35 papers)	feedforward NN, RNN, CNN, autoencoder, DRL	Review restricted to studies with backtesting; classifies by task (trade strategy, price prediction, portfolio, risk); discusses use of market, fundamental, and alternative data; emphasizes domain-specific evaluation.
Blasco et al. [56]	UQ ¹ in DL for finance (69 studies, 2001–2022; Scopus, Google Scholar)	Probabilistic DL, UQ methods	PRISMA-based review; distinguishes epistemic vs aleatoric UQ; recommend richer uncertainty representations, enhanced feature spaces, and improved posterior approximation methods.
Hu et al. [167]	DL for stock & Forex prediction (86 studies, 2015–2021)	CNN, RNN, LSTM, DNN, RL	Systematic review of stock + Forex prediction; highlights LSTM-based hybrids; notes limited sentiment integration.
Wang and Yan [397]	DL in algorithmic trading (35 studies, 2016–2021)	CNN, LSTM, DMLP	Reviews DL models for prediction, and arbitrage; highlights importance of model selection and quality financial data.
Peng and Yan [295]	DL for financial risk prediction	MLP, CNN, RNN, LSTM, BERT	Data-centric survey; emphasizes heterogeneity, imbalance, and multi-source data.
Qu et al. [303]	Bankruptcy prediction (ML + DL)	Classical ML + DL	Short review tracing shift from statistical to ML/DL; notes role of heterogeneous data and model interpretability.

TABLE 2: Comparison of major surveys on deep learning and financial forecasting: scope, models, and reported contributions (Part II).

¹Uncertainty Quantification

2.3 Deep Learning (DL)

In Fig. (2.3.1), we illustrate the position of DL, ML, and AI.

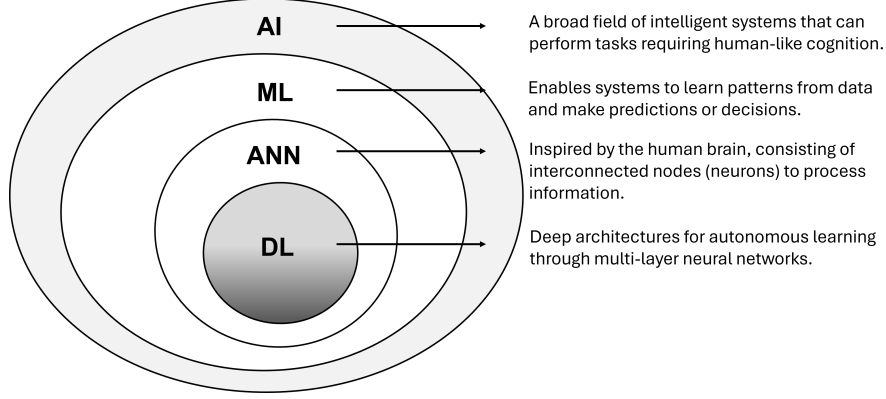


FIGURE 2.3.1: A depiction of the relationship between deep learning (DL), machine learning (ML), and artificial intelligence (AI)

DL technology originates from Artificial Neural Networks (ANNs), where the term "deep" refers to the number of hidden layers in the network. ANNs consist of interconnected neurons organized in layers. A simple ANN contains three layers: input, hidden, and output, as illustrated in Fig. 2.3.2. Information flows through the network via weighted connections. A bias term is added to the weighted inputs, and the sum is passed through an activation function, which determines the neurons responsible for feature extraction. The network output is compared to the target to compute the error, which is backpropagated to update the weights using an optimization algorithm. This process is repeated iteratively until convergence.

The mathematical formulation for a single-layer ANN is:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}), \quad (2.1)$$

where:

- $\mathbf{y} \in \mathbb{R}^m$ is the output vector of predicted financial variables,
- $\mathbf{x} \in \mathbb{R}^n$ is the input feature vector (e.g., historical prices, technical indicators),
- $\mathbf{W} \in \mathbb{R}^{m \times n}$ is the weight matrix connecting inputs to outputs,
- $\mathbf{b} \in \mathbb{R}^m$ is the bias vector, and
- $f(\cdot)$ is an element-wise activation function (e.g., ReLU, sigmoid, tanh).

Various kinds of DL models exist in the literature, including Deep Multilayer Perceptron (DMLP), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), Gated Recurrent Unit (GRU), Reservoir Computing (RC), Convolutional Neural Network (CNN), Temporal Convolutional Network (TCN), AutoEncoders, other deep structures, and hybrid models. In what follows these are briefly discussed.

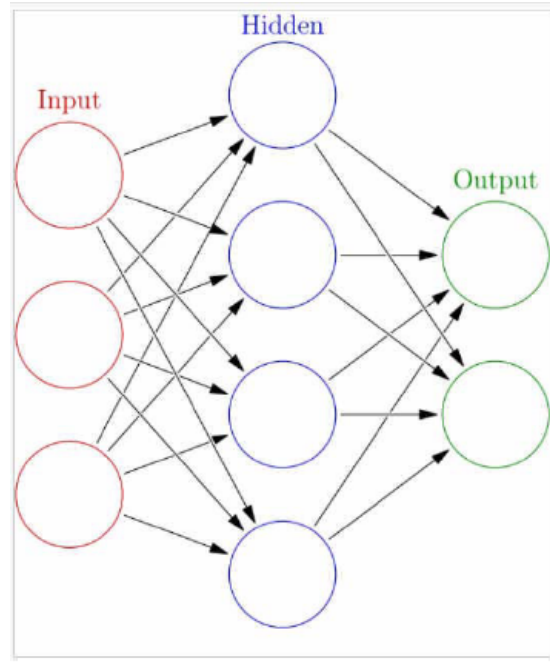


FIGURE 2.3.2: Illustration of a simple ANN architecture, comprising the input, hidden, and output layers

2.3.1 Deep Multi-Layer Perceptron (DMLP)

The Multi-Layer Perceptron (MLP) is a fully connected feedforward ANN consisting of an input layer, an output layer, and one or more hidden layers, which serve as the computational engines. MLP is regarded as the foundational architecture for all subsequent training algorithms within the ANN framework. As one of the earliest and simplest neural network models, MLPs stand as the backbone of DL [306]. Frank Rosenblatt established the perceptron in 1957, a neural network able to classify images [313]. Since then, advancements in computing power, increased data availability, and reduced capacity costs have facilitated the development of new techniques.

In DMLP, the deeper the network, the higher the complexity of the model and the higher the number of the hidden layers. In the hidden layers, each neuron has input (x), weight (w), bias (b) terms, and an activation function that produces the output of the network. The output of a single neuron in the network is depicted in Eq. (2.2).

The most widely used nonlinear activation functions are sigmoid (Eq. 2.3) [79], hyperbolic tangent (Eq. 2.4) [197], Rectified Linear Unit (ReLU) (Eq. 2.5) [274], leaky-ReLU (Eq. 2.6) [251], and softmax (Eq. 2.7) [171]. These activation functions introduce non-linearity into the network to learn complex mappings from data.

$$y_i = f\left(\sum_i w_i x_i + b_i\right) \quad (2.2)$$

where f is the activation function, chosen from:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2.3)$$

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2.4)$$

$$R(z) = \max(0, z) \quad (2.5)$$

$$R_{\text{leaky}}(z) = \alpha z \cdot 1(z < 0) + z \cdot 1(z \geq 0) \quad (2.6)$$

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad (2.7)$$

2.3.2 Recurrent neural network (RNN)

The RNN is a popular branch of ANN designed for processing sequential or time-series data. RNNs use outputs from previous time steps as inputs for the current step, enabling the model to rely on prior information in the sequence. The hidden layers of an RNN serve as memory, retaining information from earlier stages to forecast future steps. Fig. (2.3.3) illustrates the RNN architecture (left) and its unfolded representation (right).

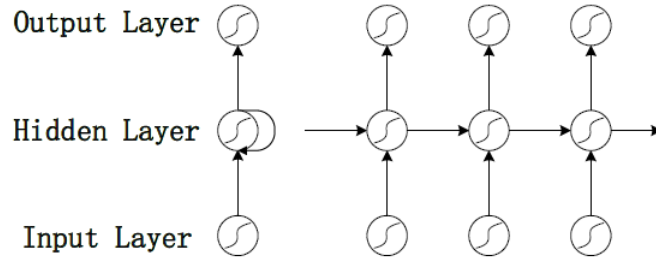


FIGURE 2.3.3: Illustration of a standard RNN (left) and its unfolded version (right) [441]

At each time step t , the previous hidden output h_{t-1} and the current input x_t are used to compute the next hidden output h_t and the prediction $\hat{y}_t$, as given by the following equations:

$$h_t = \sigma(W_{hh}h_{t-1} + W_{hx}x_t) \quad (2.8)$$

$$\hat{y}_t = \text{softmax}(W_S h_t) \quad (2.9)$$

Here, W_{hx} represents the weight matrix for the input x_t , W_{hh} is the weight matrix for the previous hidden output h_{t-1} , and σ denotes the sigmoid activation function. Additionally, h_0 corresponds to the initial hidden output at time step $t = 0$. Eq. (2.9) represents the output probability distribution over the dataset at each time step t . Essentially, $\hat{y}_t$ is the next predicted element, given the sequential information up to that point and the most recent input element, x_t .

RNNs are well-suited for modeling sequential data and time series. However, they suffer from vanishing and exploding gradients, which hinder learning from long data sequences. In the case of vanishing gradients, the gradients shrink over time during backpropagation, leading to minimal weight updates. In contrast, exploding gradients occur when the gradients increase exponentially during backpropagation, causing the model's weights to grow uncontrollably. This results in an unstable model that is unable to learn during training from long input sequences [290]. In the following, we discuss several popular variants of RNNs that address these challenges and perform well in many real-world applications.

Long Short-Term Memory (LSTM)

The LSTM network, introduced by Hochreiter [160], is a widely used variant of RNNs designed to deal with the vanishing gradient problem. It achieves this by employing gating mechanisms—the input, forget, and output gates—that regulate the flow of information within the network. The forget gate controls the retention or discarding of past cell state information, the input gate determines which new information should enter the cell state, and the output gate determines and controls the outputs. This design allows LSTMs to capture and retain long-term dependencies more effectively than standard RNNs. Fig. (??) illustrates the structure of an LSTM cell, where the upper horizontal line represents the cell state, facilitating information flow across time steps.

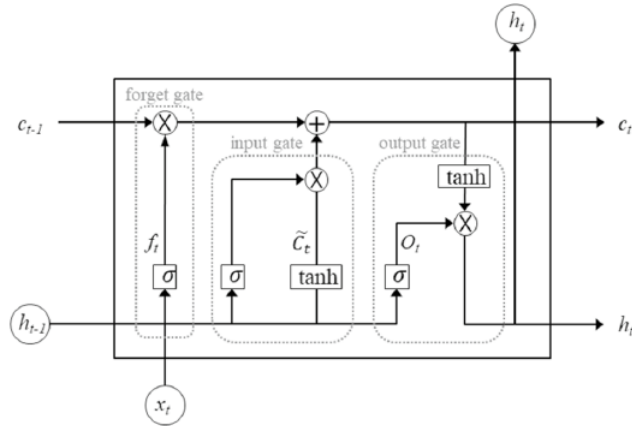


FIGURE 2.3.4: Structure of an LSTM cell [30]

The gating mechanisms are analog and based on the sigmoid function, which operates within the range (0, 1), selectively filtering, adding, or removing information.

At each time step t , the forget gate regulates past information retention by applying a sigmoid function to the previous hidden state h_{t-1} and input x_t :

$$f_t = \sigma(W_{f,x}x_t + W_{f,h}h_{t-1} + b_f) \quad (2.10)$$

Next, the input gate updates the cell state. A candidate value $\tilde{c}_t$ is generated via a tanh activation, while i_t determines its contribution:

$$\tilde{c}_t = \tanh(W_{\tilde{c},x}x_t + W_{\tilde{c},h}h_{t-1} + \tilde{b}_c) \quad (2.11)$$

$$i_t = \sigma(W_{i,x}x_t + W_{i,h}h_{t-1} + b_i) \quad (2.12)$$

The updated cell state c_t is computed as:

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \quad (2.13)$$

Finally, the output gate regulates the hidden state update:

$$o_t = \sigma(W_{o,x}x_t + W_{o,h}h_{t-1} + b_o) \quad (2.14)$$

$$h_t = o_t \circ \tanh(c_t) \quad (2.15)$$

The weight matrices governing the forget, input, candidate, and output computations ($W_{f,x}$, $W_{f,h}$, $\tilde{W}_{c,x}$, $\tilde{W}_{c,h}$, $W_{i,x}$, $W_{i,h}$, $W_{o,x}$, $W_{o,h}$), along with their corresponding bias vectors (b_f , $\tilde{b}_c$, b_i , b_o), are optimized during training to minimize the loss function.

Bidirectional Long Short-Term Memory (BiLSTM)

BiLSTM, introduced by Graves and Schmidhuber [144], processes sequential data in both forward and backward directions through two distinct hidden LSTM layers, as illustrated in Fig. (2.3.5). The forward layer processes the sequence in the same direction as the input, while the backward layer processes it in the reverse direction. BiLSTM connects both layers to the same output layer. The final output of the BiLSTM layer, y_t , is obtained by combining the forward and backward hidden states, as shown in Eq. (2.16), where the combination function σ can be concatenation, summation, averaging, or multiplication.

$$y_t = \sigma(\vec{h}_f, \overleftarrow{h}_b) \quad (2.16)$$

Gated Recurrent Units (GRUs)

GRU, introduced by Cho et al. [73], is another variant of RNN designed to mitigate the vanishing gradient problem. It utilizes gating mechanisms to control the flow of information between cells in the network, as illustrated in Fig. (2.3.6). GRU is a simplified variant of LSTM with fewer parameters. It uses two gates: the update gate and the reset gate, compared to LSTM's three gates (input, output, and forget gates) [205]. The update gate controls how much of the previous hidden state needs to be carried forward, enabling the GRU to capture long-range dependencies adaptively without discarding information from earlier parts of the sequence. The reset gate determines how much of the previous memory is combined with the new input in the current state calculation. The GRU transition equations are as follows:

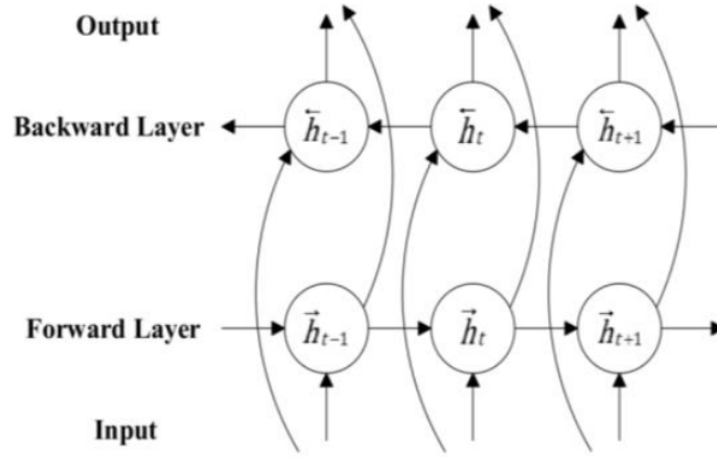


FIGURE 2.3.5: Workflow of BiLSTM [111]

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \quad (2.17)$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \quad (2.18)$$

$$\tilde{h}_t = \tanh(W \cdot [r_t \circ h_{t-1}, x_t]) \quad (2.19)$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t \quad (2.20)$$

The simpler structure of GRU leads to fewer matrix multiplications, enhancing its computational efficiency [434]. As a result, GRU enables faster training and tends to outperform LSTM when trained on limited datasets. In contrast, LSTM exhibits superior performance when dealing with larger datasets [68, 195, 428].

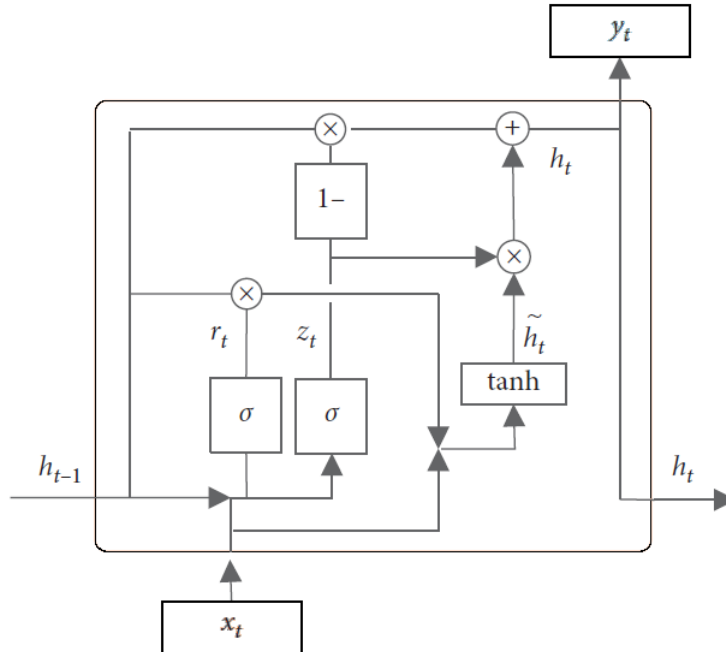


FIGURE 2.3.6: Structure of the hidden layer node in GRU [421]

Reservoir Computing (RC)

ESN [174, 177] is the virtual paradigm of RC derived from the theory of RNNs. The architecture of an ESN consists of an input layer, a reservoir, and a readout layer, as depicted in Fig. (2.3.7). The reservoir is a fixed, randomly initialized, and sparsely connected hidden layer of neurons, which maps input signals into a high-dimensional space. Generally treated as a "black box" [78], the reservoir serves as a high-dimensional dynamic system that transforms input signals into nonlinear internal representations, preserving temporal dependencies, known as reservoir states. Using the notation in [37], the evolution of the reservoir state (x_t) for a given input sequence (z_t) is given by:

$$\begin{aligned} x_{t+1} &= lr \cdot x_t + \\ &\quad (1 - lr) \cdot f \left(\rho \frac{W_{res}}{\|\lambda_{\max}(W_{res})\|} x_t + \gamma \frac{W_{in}}{\|W_{in}\|} z_{t+1} + \omega \frac{\zeta}{\|\zeta\|} \right), \\ y_{t+1} &= W_{out}^\top x_{t+1} \end{aligned} \quad (2.21)$$

Importantly, only the readout layer is trained, typically using linear regression or similar methods, while the reservoir and input weights remain fixed. This significantly simplifies the training process compared to traditional RNNs.

The key hyperparameters of the system include the leaking rate (lr), spectral radius (ρ), input scaling (γ), bias (ω), as well as the size of the reservoir (number of neurons) and the sparsity level. The weight matrices W_{in} and W_{res} remain fixed after initialization, while W_{out} is typically trained via linear regression [98, 244]:

$$W_{out} = (X^T X + \alpha I)^{-1} X^T Y \quad (2.22)$$

where X is the matrix of reservoir states collected over time, Y is the corresponding matrix of target outputs, and α is a regularization parameter that balances model complexity and generalization. ESNs offer a computationally efficient alternative for sequence modeling tasks, making them particularly relevant for financial forecasting applications.

A fundamental requirement for ESNs is that the reservoir satisfies the Echo State Property (ESP), ensuring that the influence of initial conditions vanishes over time. Although the condition $\rho(W_0) < 1$ is commonly cited to guarantee ESP, optimal system performance in practice depends on the design of the reservoir, hyperparameters, and overall system configuration. The ESP asserts that the reservoir dynamics maintain a stable "echo" state, preserving sensitivity to inputs without allowing their influence to grow exponentially over time. Achieving optimal performance requires careful hyperparameter tuning tailored to the task-at-hand. Since there are no universal design rules for ESNs, guidelines from [176, 243] offer practical insights, though empirical testing and task-specific adjustments remain essential.

In RC, reservoirs can be either virtual or physical. Independently and simultaneously with ESNs, Liquid State Machines (LSMs) [252] were developed as the physical paradigm of RC. LSMs employ spiking neurons, which more closely mimic the

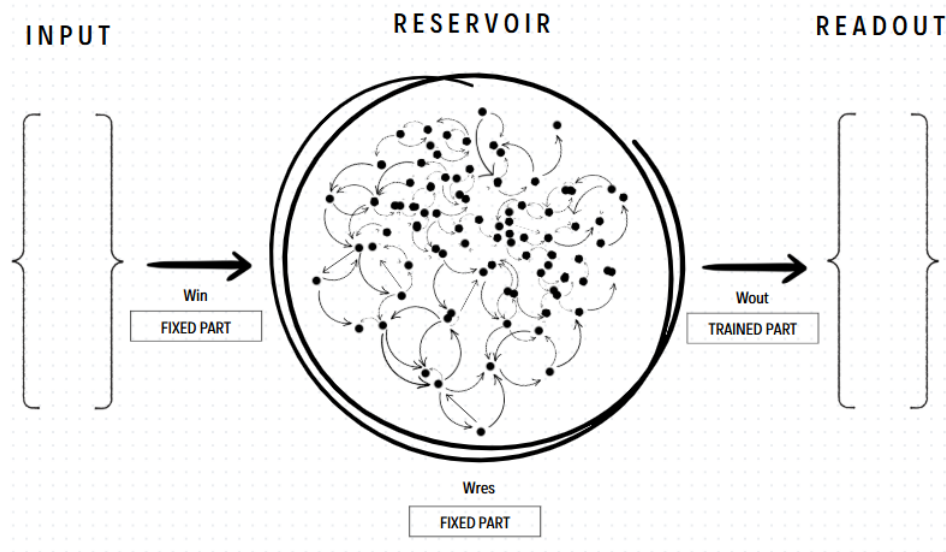


FIGURE 2.3.7: ESN Architecture: The input layer processes external signals, which are then mapped into high-dimensional states by the reservoir. The output layer maps these states to the desired output, enabling predictions.

behavior of biological neurons compared to traditional artificial neurons found in feedforward or recurrent neural networks. These spiking neurons process information through discrete voltage spikes, triggered when the membrane potential surpasses a predefined threshold. Physical reservoirs, such as a bucket of water serving as a reservoir that makes computations on inputs given as waves on its surface [112], are networks in the sense of continuous systems that operate in space and/or time.

Building upon LSMs, Liquid Neural Networks (LNNs) [156] provide greater flexibility and adaptability through trainable neuron dynamics in both the reservoir and output layers, unlike ESNs and LSMs, where only the output layer is trained. LNNs are considered as Liquid Time Constant (LTC)-based architectures, where LTC is a critical parameter controlling the rate of information decay within the reservoir dynamics. LTC networks enhance the plasticity of DL models by enabling dynamic adaptation to new information, effectively balancing the retention of prior knowledge with the integration of new data. This adaptability prevents overfitting to old data and avoids forgetting previously learned tasks, akin to a liquid surface that adjusts to disturbances without losing its fluidity. Further analysis of physical RC is beyond the scope of this chapter, but readers are suggested to follow [364].

2.3.3 Convolutional Neural Networks (CNNs)

CNNs [217] are discriminative DL architectures designed primarily for visual tasks such as image recognition, classification, and segmentation. The core components include the convolutional layer for feature extraction using filters or kernels, the pooling layer for dimensionality reduction through techniques like max or average pooling, summarizing clusters of neurons in one layer into a single neuron in the

next layer [51], followed by a flattened layer, and the fully connected layer enabling the model to combine the extracted features for classification or prediction. Non-linearity is introduced via activation functions such as ReLU, sigmoid, or tanh [171]. The output layer typically uses a logistic activation function, like softmax, to assign probability scores for classification [51, 196]. Convolutional and pooling layers often form pairs, with multiple such pairs integrated into a CNN architecture, as shown in Fig. (2.3.8). CNNs are designed for processing 2D data, making them widely used in applications such as visual recognition, medical image analysis, and natural language processing.

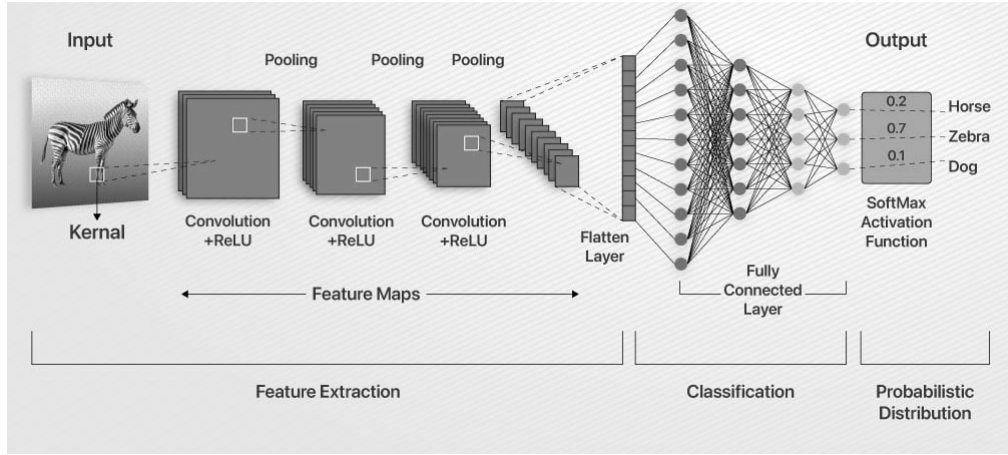


FIGURE 2.3.8: CNN example (Source: <https://www.analytixlabs.co.in/blog/convolutional-neural-network/>)

The convolution operation extracts features from the input. For 1D data, the operation is defined as in Eq. (2.23), where t denotes time, $s(t)$ is the feature map, k is the kernel, x is the input, and a denote variable. For 2D image data, Eq. (2.24) describes the convolution, where I is the input image, K is the kernel, (m, n) represents image dimensions, and (i, j) denote variables. The neural network architecture follows in Eq. (2.25), where W_{ij} are the weights, x_j is the input, b_i is the bias, and z_i is the output of neuron i . The softmax function, as given in Eq. (2.26), is then applied to generate the final output, which represents the class probabilities for each y_i .

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)k(t-a) \quad (2.23)$$

$$S(i, j) = \sum_m \sum_n I(m, n)K(i-m, j-n) \quad (2.24)$$

$$z_i = \sum_j W_{ij}x_j + b_i \quad (2.25)$$

$$y_i = \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (2.26)$$

Temporal Convolutional Network (TCN)

Introduced by [216], TCNs were further popularized and formalized by [31] for broader applications. Belonging to the family of CNNs, TCN is an extension designed to efficiently capture hidden temporal dependencies in sequential data. A TCN maps an input sequence x to an output of the same length using a 1D fully convolutional network architecture with zero-padding of size $(\text{kernel_size} - 1)$ to maintain sequence length [31]. Causal convolutions ensure that each output at time t depends only on inputs from time t and earlier, preventing information leakage from future to the past [31]. To capture long-term dependencies efficiently, TCNs employ dilated convolutions [381], where dilation factor d controls the spacing between filter elements, increasing the receptive field exponentially. A dilated convolution F on an element s of the 1D sequence x is given by

$$F(s) = \sum_{i=0}^{k-1} f(i) \cdot x_{s-d \cdot i}, \quad (2.27)$$

where k is the filter size. Additionally, TCN utilizes residual blocks [158] to enhance stability in deeper architectures. Each residual block comprises two layers of dilated causal convolutions, weight normalization, ReLU activation, and spatial dropout. A 1×1 convolution operation is also incorporated in each residual block to ensure input-output size consistency. A dilated causal convolution with a kernel size of 2 and dilation factors of $[1, 2, 4]$, is illustrated in Fig. (2.3.9).

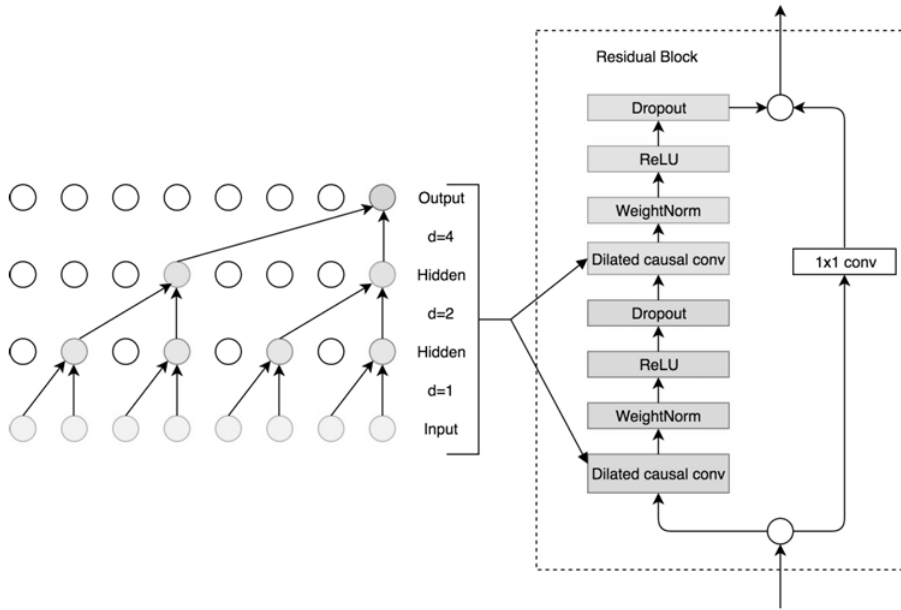


FIGURE 2.3.9: TCN Architecture. The left part is the dilated causal convolution with a kernel size of 2 and dilation factors $d = [1, 2, 4]$. The right part is the residual block [302]

2.3.4 Autoencoder (AE)

An AE [139] is a type of neural network designed for unsupervised learning, primarily employed in dimensionality reduction and feature extraction. It comprises

two main components: an encoder, which maps the input into a lower-dimensional latent representation, and a decoder, which reconstructs the input from this compressed representation, as illustrated in Fig. (2.3.10). Given an input $x \in [0, 1]^d$, the encoder compresses it into a latent representation h , preserving essential features while discarding redundant information. This transformation is governed by a function $f(x)$, parameterized by a weight matrix W_1 , a bias vector b_1 , and a nonlinear activation function σ_1 , typically a sigmoid or ReLU function. The decoder then reconstructs the input from the latent representation using an inverse function $g(h)$, which maps h back to the input space. This process involves another set of parameters: weight matrix W_2 , bias vector b_2 , and an activation function σ_2 . The goal is to generate an output r that closely approximates the original input x . The model is trained by minimizing the reconstruction loss, typically measured using the MSE. Mathematically, the encoding and decoding processes are expressed as:

$$h = f(x) = \sigma_1(W_1 x + b_1) \quad (2.28)$$

$$r = g(h) = \sigma_2(W_2 h + b_2) \quad (2.29)$$

$$L(x, r) = \|x - r\|^2 \quad (2.30)$$

AEs are widely applied in feature learning, anomaly detection, and data compression, leveraging their ability to learn compact and meaningful representations of input data.

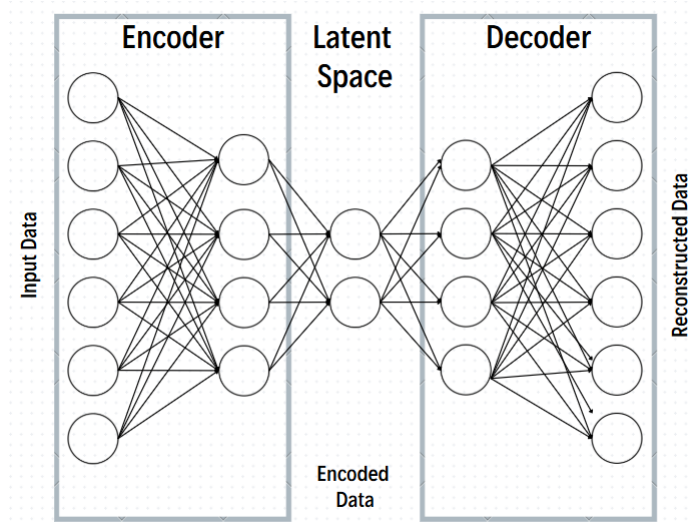


FIGURE 2.3.10: Standard architecture of AE

While encoder-decoder models [73] share a similar structure with autoencoders, they differ in that they are designed for supervised tasks involving input-output transformations, mapping an input to a distinct output, such as in sequence generation or machine translation. In contrast, autoencoders are unsupervised models focused on reconstructing the input from a compressed latent representation, with

the input and output being the same.

2.3.5 Other deep structures

Generative Adversarial Networks (GANs) [140] are a class of neural networks designed for generative modeling, capable of producing realistic data samples by learning the underlying distribution of a given dataset. A GAN consists of two competing networks: a generator G , which generates synthetic data resembling real samples, and a discriminator D , which evaluates whether a given sample is real or generated [236]. These networks are trained in an adversarial process, where the generator aims to produce increasingly realistic data to deceive the discriminator, while the discriminator improves its ability to distinguish real from fake samples. The training process converges at an ideal point known as the Nash equilibrium, where the discriminator can no longer reliably differentiate between real and generated data. At this point, the generator has successfully learned the underlying distribution of the original data [235]. GANs have been widely applied in image synthesis, data generation, and style transfer.

Transformers [382] are a class of DL models designed to handle sequential data efficiently by leveraging self-attention mechanisms. Unlike traditional recurrent architectures that process data sequentially, transformers establish direct connections between all elements in a sequence, enabling parallelization and improved long-term memory retention [116]. This architecture allows information to propagate across the entire sequence, improving performance in natural language processing and time series forecasting tasks. However, since all input elements are interconnected, the model relies on a self-attention mechanism to selectively focus on the most relevant information while filtering out noise, enabling context-aware learning. Several specialized variants have emerged for time series forecasting, including the Temporal Fusion Transformer (TFT) for multi-horizon and multivariate forecasting [409], and PatchTST [279] employing patching, where time series data is segmented into sub-sequences that serve as input tokens, and channel independence, wherein each channel corresponds to a single univariate time series sharing the same embedding and Transformer weights [106].

Neural Basis Expansion Analysis for Time Series (N-BEATS) [284] is a pure DL architecture designed for univariate time series forecasting, without relying on domain-specific components such as recurrent or convolutional structures. It outperformed the M4 competition-winning ES-RNN model [254] by 3%. N-BEATS is structured as a collection of stacks, each containing multiple blocks. Each block, serving as the basic processing unit, is a multi-layer fully connected network with ReLU activations, producing two outputs: a forecast for the target horizon and a backcast, which is the model's best estimate of its input given its functional limitations. The layer of blocks are combined together using a hierarchical doubly residual stacking topology. N-BEATS offers several advantages, including interpretability, ease of implementation across different domains without requiring modifica-

tions, and efficient training. Unlike many traditional time series models, it does not require specialized feature engineering or input scaling, making it a flexible and powerful approach to time series forecasting.

Trellis networks [32] are temporal convolutional networks designed for sequence modeling, characterized by two key features: first, weights are shared not only by all time steps but also by all network layers, tying them into a regular trellis pattern; second, the input at any given time step is fed into all layers of the network, not just the first layer. Bridging recurrent and convolutional architectures, they leverage techniques from both domains. Recent implementations include SA-TrellisNet for news-driven stock index prediction with Sentiment Attention [237] and GAN-TrellisNet, a multi-factor model integrating GANs, TrellisNet, and CNNs to capture temporal dependencies and refine predictions [236].

A **Deep Q-Network (DQN)** [267] is a reinforcement learning algorithm that combines Q-learning with deep neural networks to estimate and optimize action-value functions (Q-functions), enabling decision-making in high-dimensional state spaces. The core concepts of DQN include Q-learning, experience replay, and target networks. DQN uses Q-learning to approximate the optimal action-value function, mapping states to expected future rewards. Experience replay involves storing past experiences in a memory buffer and randomly sampling them during training to break temporal dependencies and stabilize learning. The target network is a separate network with the same architecture as the main network, periodically updated by copying the weights from the main network, which helps to stabilize learning. This approach has shown significant success in complex environments, such as Atari games.

2.3.6 Hybrids

Hybrid models, integrating DL, ML, and statistical methods, have become integral to financial forecasting. By combining the strengths of different modeling approaches, hybrid models can mitigate the limitations of individual methods, improving predictive accuracy, robustness, and adaptability. This improved performance, however, often comes at the expense of increased computational requirements and model complexity, necessitating careful model design, tuning, and evaluation. Examples include CNN-LSTM models [4, 16, 154, 269] which combine convolutional layers that extract short-term temporal features with recurrent layers that model long-term dependencies, LSTM-GRU hybrids for short-term cryptocurrency prediction [291], and optimization-enhanced DL frameworks like PSO-optimized LSTM/GRU [67] and GA-LSTM [70]. The applications, contributions, and limitations of the various hybrid approaches are discussed in the following sections across multiple financial forecasting domains.

2.4 Hyperparameters Tuning

Hyperparameters are critical in defining the architecture of DL models and have a significant impact on their predictive performance. Optimizing these parameters is essential for obtaining optimal results in financial forecasting tasks [3]. Practitioners should choose the optimization method that best aligns with the complexity of the search space, the available computational resources, and the specific characteristics of the model. Commonly employed hyperparameter tuning techniques are outlined below.

Manual tuning (which relies heavily on the practitioner's expertise), Random Search (which explores the hyperparameter space randomly), Grid Search (which exhaustively evaluates all combinations of hyperparameters), and Bayesian Optimization (a probabilistic, model-based approach) remain widely used for hyperparameter selection.

More advanced optimization techniques inspired by nature have gained popularity due to their ability to efficiently explore complex search spaces. Examples include evolutionary and swarm intelligence methods. The Artificial Rabbit Optimization (ARO) algorithm, for instance, draws inspiration from rabbit survival strategies, while Particle Swarm Optimization (PSO) mimics the social behavior of birds flocking and fish schooling. Adaptive PSO further refines this approach by dynamically adjusting parameters for faster convergence. Similarly, Artificial Bee Colony (ABC) optimization simulates the foraging behavior of honeybees, the Flower Pollination Algorithm (FPA) is inspired by the pollination behavior of flowers, Genetic Algorithms (GA) replicate natural selection processes, and Cuckoo Search leverages the brood parasitism strategy of cuckoo birds.

2.5 Financial Time series Forecasting

The application of DL models in finance is vast and diverse. In the previous section, we introduced the most prominent DL models. This section provides a review of DL models used in financial forecasting applications. We categorize the implementation areas and present them in separate subsections to offer a comprehensive overview. The aim is to summarize the most recent papers, specifically those published between 2020 and 2024, within the field of DL for financial forecasting.

In each subsection corresponding to a specific asset type (e.g., stock price forecasting, index prediction, forex price prediction, commodity price prediction, bond price forecasting, cryptocurrency price forecasting, and volatility forecasting), we further classify the studies based on their focus. We distinguish among standalone and hybrid models, novel approaches, technical and sentiment analysis-based models, and those employing preprocessing techniques like dimensionality reduction, decomposition, and optimization.

Moreover, in each subsection, we have tabulated the representative models, datasets, and features of the relevant studies to provide as much detailed information as possible. These tables can be found in Appendix .1.

Each table includes the following fields to provide the implementation details of the papers within the group: Article (Art.) and Dataset refer to the study and the data used, respectively. Period indicates the time frame for training, validation, and testing. Feature Set lists the input features used in the study. Method specifies the DL models used. Performance Criteria provides the evaluation metrics, and Environment (Env.) includes the development framework, software, or tools employed. In some cases, certain columns may be left empty if the corresponding information was not available in the paper. The goal is to highlight the dominant models, features, and data types that stand out within each application area and to briefly discuss how each research article contributes to the existing body of literature.

Note: Some studies evaluate their models across multiple asset classes. To maintain the autonomy of each section and accurately reflect the trends within each domain, these studies are included in all relevant sections (e.g., stock forecasting, index forecasting, etc.). Consequently, overlap between sections is expected, and certain studies may be cited in multiple asset-specific subsections.

Note: Each subsection concludes with a summary highlighting key observations, cross-study statistics, and thematic insights from the reviewed literature. These summaries cover trends in model usage, feature selection, forecasting targets, geographic focus, data timeframes, and performance metrics. Summaries are omitted in cases where the number of reviewed studies is insufficient (typically ≤ 10) to support meaningful aggregation or generalization.

2.5.1 Stock Price Forecasting

Stock price forecasting has become a central application area for DL, driven by the growing demand for accurate, data-driven financial decision-making. The literature reflects a transition from monolithic models, such as ARIMA and other statistical approaches with fixed assumptions, to more flexible and task-specific designs, each tailored to asset characteristics and market regimes. This evolution includes both standalone and hybrid models, using RNNs, CNNs, attention mechanisms, and generative frameworks. There is also increased integration of technical indicators, sentiment analysis, dimensionality reduction, and optimization strategies to improve accuracy. This section reviews these advances, covering architectural innovations, feature integration, optimization, and applications in varied financial contexts, including pandemic resilience and specialized prediction tasks, organized according to the outlined subsections.

Standalone Architectures

In what follows, recent literature is systematically reviewed and analyzed to highlight the main trends and developments in standalone DL models for stock forecasting.

Comparative Performance of RNN Variants LSTM remains a dominant baseline model. Moradi et al. [271] show that LSTM outperforms traditional ML models such as RF and SVR on Tehran Stock Exchange data, while Furizal et al. [121] demonstrate improved accuracy with multivariate LSTM models. Despite its widespread use, recent studies report that GRU and ESN frequently outperform LSTM in accuracy and convergence speed. For instance, Sun et al. [361] highlight ESN's efficiency over LSTM, and Lawi et al. [215], Ismailova et al. [173] find GRU consistently more effective. JING and PATHMANATHAN [190] further reveal GRU's strength in handling volatile assets, with LSTM performing better in less volatile ETFs. In contrast, models without gating mechanisms, such as VRNNs, and structurally different models like CNNs, often underperform due to limited ability to capture temporal dependencies.

Importantly, model suitability varies by forecasting horizon; LSTM excels in short-term forecasts compared to traditional technical stock market analysis models (e.g., SMA, EMA) but suffers from higher errors in long-term forecasts [55]. Probabilistic models like HMMs can rival or even outperform LSTM and ensemble-based classifiers under certain conditions [339].

Novel Model Architectures Several studies explore novel architectures and enhancements. Yang [423] use LSTM and GRU to forecast financial indicators (ROTA, PTSR, PER) achieving low error rates with comparable accuracy. Md et al. [264] propose a multi-layer sequential LSTM to mitigate the vanishing gradient problem, outperforming existing DL models. Gajjar et al. [123] compare LTC-based architectures with RNN variants, including their bidirectional versions, finding that LNN and particularly BiLTC deliver superior forecasting performance, highlighting LTC as promising alternatives. Similarly, Zhan and Kim [434] improve LSTM-based prediction by integrating time-window and convolutional filters to better capture both short- and long-term dependencies in financial data.

Integrating Technical and Sentiment Analysis Recent works also emphasize feature augmentation, combining technical indicators (e.g., RSI, EMA, ATR) [89, 293, 343] and sentiment analysis (tweets, reports) [348, 355] with LSTM. These studies report consistent improvements. Notably, advanced variants such as the Improved Recurrent Rider LSTM (IRRLSTM) [293], integrating feature selection and data augmentation techniques, has shown superior results.

Other Use Cases Beyond price prediction, studies have expanded DL use cases. Abuein et al. [6] predict optimal trading times using SVR and LSTM. Velmurugan and Indhumathy [383] treating future support and resistance values as a time-series problem, apply the Fibonacci sequence with LSTM, achieving high accuracy and valuable trading insights. GAN-based models (e.g., TimeGAN) address the differences of time-constrained stock markets versus continuous cryptocurrency markets [324], revealing higher predictive power in cryptocurrencies. Meanwhile, models like CNN-BI [336] and 2D-CNN [366] leverage image-based representations of stock data (i.e., time series, bar charts, technical indicators) to enhance trading signal classification and improve trading decisions. These diverse approaches illustrate the expanding versatility of DL techniques in capturing complex market behaviors beyond traditional price forecasting.

Hybrid Architectures

This part provides a systematic review and analysis of recent literature, focusing on the key trends and advancements in hybrid DL models for stock forecasting.

Fusion of CNNs, RNNs, and Autoencoders Several studies respond to the limitations of standalone models by proposing hybrid DL architectures that enhance forecasting accuracy. Models like CNN-LSTM [2, 4, 16, 269] and CNN-LSTM-GRU [277] effectively capture multi-scale dependencies. Others, such as KM-CAE-DLSTM [147] and NR-LSTM [349], incorporate clustering or trend-residual separation to improve learning. Additionally, signal denoising techniques, such as wavelet transforms, have proven to be effective preprocessing steps for enhancing the performance of LSTM-based forecasting models [182, 247].

Novel Model Architectures Novel studies on automated trading increasingly integrate DL with domain-specific enhancements to balance predictive accuracy and interpretability. Hybrid models like the VAR-LSTM by Sakib and Mustajab [318] capture both linear and nonlinear dependencies that identify buy and sell signals by analyzing market regimes defined by moving averages correlation. High-frequency forecasting via ODE-LSTM [223] improves dynamic feature learning and model interpretability, achieving strong returns in the Chinese A-share market for small-cap stocks. Ansari et al. [23] introduce a decision support system using DRL with a GRU-based forecasting network, integrated with past stock price trends to form an enriched observation state for the DRL agent, enabling informed trading decisions, offers fast and strong predictive performance, with stable profitability.

Li et al. [225] present a clustering-enhanced DL framework that improves stock return forecasting by grouping time-series with similar patterns using LWDTW, demonstrating that such clustering significantly boosts model performance, with

LSTM outperforming RNN, and GRU, and other similarity measures. For high-frequency forecasting, Roy Choudhury et al. [315] propose LSTM-VAE that denoise input data and predict short-term price deviations from recent averages, consistently outperforming traditional time-series models, while Zaznov et al. [432] fuse convolution layers and GRU in TFF-CL-GRU for price direction prediction, using market microstructure data, boost predictive and trading performance at a high computational cost. Rooted in chaos theory, the novel PSR-Transformer [298] combines Phase Space Reconstruction (PSR) with Transformer encoders, uncovering hidden patterns through time-delay embedding and effectively capturing long-range dependencies. Addressing signal noise, Akşehir and Kılıç [10] present ICE2DE-MDL, which combines two-level decomposition, entropy-based IMF classification as either high-frequency or noise-free, and model-specific forecasting (LSTM, GRU, SVR), demonstrating robust performance but increasing model complexity. These works reflect a broader trend: combining DL with signal processing, clustering, or regime-awareness can enhance performance, but often entails trade-offs between interpretability, scalability, and computation.

Several works introduce novel multi-modal hybrid architectures that combine temporal, spatial, and attention-based mechanisms. For instance, BiCuDNNLSTM-1dCNN [200] merges bidirectional LSTM with 1D-CNN to extract temporal information and abrupt changes, while AE-ACG [226] integrates CNN-GRU blocks within an autoencoder for hierarchical attention-guided decoding, enhancing information utilization. Similarly, Rath et al. [307] combine stacked Bi-LSTM, attention modules, and wavelet-denoised CNNs, further optimized using an equilibrium strategy. These models highlight an increasing trend toward modular, specialized architectures tailored for complex financial signals. Beyond classical DL, quantum-inspired methods like QGAF [416] demonstrate promising improvements in stock trend prediction by transforming time series into high-quality grayscale images, showcasing the growing potential of quantum computing in financial forecasting. Collectively, these studies underscore both the innovation and fragmentation in model design, calling for comparative benchmarks and generalization-focused evaluations.

Enhancing Prediction via Dimensionality Reduction To address feature redundancy and improve forecasting accuracy, several studies incorporate dimensionality reduction techniques. Ding et al. [91] apply GPCA with Grey Relation Analysis to extract key macroeconomic indicators and reduce input dimensionality, revealing profitability disparities across Chinese petroleum firms. Similarly, Mansoor et al. [258] combine Mutual Information for feature ranking with PCA to streamline inputs for LSTM, improving prediction efficiency.

Optimization-Integrated Models Several studies emphasize hyperparameter tuning as a performance bottleneck and apply bio-inspired optimization methods. Gülmez [149] use the ARO algorithm to optimize LSTM for DJIA stocks. Other studies deploy PSO, FPA, GA, and Tree-structured Parzen Estimator methods [211, 344],

with optimized hybrids improving intraday stock prediction, and generalize better across both short- and long-term forecasting horizons.

Novel hybrid optimization models combine metaheuristics and DL for improved stock forecasting. For instance, Pai et al. [287] introduce GSO-LSTM, which outperforms traditional tuning methods like grid and random search. Salama [320] presents the SHOAGAI-TSF framework that combines SHOA with Generative AI and CGAN for probabilistic prediction. Jayanth and Manimaran [184] develop DES-DA-ED-Bi-GRU-BO, incorporating trend decomposition and Bayesian Optimization, while Liu and Long [234] propose a hybrid system uniting EWT, dropout-LSTM, ORELM, and PSO, outperforming nine benchmark models.

Integrating Technical and Sentiment Analysis A growing body of work incorporates multi-modal inputs, combining price data with technical indicators and sentiment to capture richer market signals and improve forecasting robustness. Uçkan [379] propose a PCA-LSTM-CNN hybrid that enhances short-term prediction by identifying key technical indicators via PCA. Fan and Peng [107] develop a DRL-based model combining DQN and LSTM to make optimal trading decisions using raw and technical features. To better capture market dynamics, Zhang et al. [438] design a CNN-Bi-LSTM with attention and a modified loss function to enhance trading signals by adapting to the heavy-tailed distribution of stock price fluctuations. Chen and Zhou [70] introduce GA-LSTM, using market, technical, and financial factors. Addressing the limited data problem for newly listed stocks, AUDA-Net by Wei and Dai [399] leverages GAN-based Adversarial Unsupervised Domain Adaptation for trend prediction. The model generates synthetic unlabeled data and uses adversarial domain adaptation to transfer knowledge from source to newly listed stocks, achieving accuracy comparable to supervised DL models despite limited labeled data. Liu et al. [236] propose GAN-TrellisNet, integrating GANs, CNNs, and TrellisNet architectures with an enriched alpha158+OCHLVC feature set, which enhances learning depth and outperforms benchmarks across global markets over a decade.

Beyond technical factors, sentiment analysis has become a crucial complementary input. Chandola et al. [66], Ko and Chang [207], Kumar et al. [212] integrate sentiment extracted via BERT, Word2Vec, and Twitter polarity, respectively, with LSTM-based models. Eslamieh et al. [104] go further by introducing User2Vec, a model that combines market data with Twitter-based sentiment and user credibility scores, improving accuracy, particularly for stocks with high social engagement. Choi et al. [74] design a feature-interaction module to blend price and text features effectively, while Jin et al. [189] integrate sentiment from bullish/bearish comments with EMD decomposition and attention-LSTM. Other works incorporate denoising techniques ([172]) and news sentiment via FinBERT ([69]) to further enhance multi-modal fusion. These studies demonstrate that combining technical, and sentiment analysis with financial data consistently improves predictive performance.

Financial Forecasting in Pandemic Contexts The COVID-19 pandemic offered a natural experiment to test the resilience and adaptability of financial forecasting models. Notably, Lachaab and Omri [213] find that KNN outperforms LSTM and ARIMA across pandemic phases, implying that non-parametric models may better handle sudden shifts in market behavior. Their conclusion subtly challenges the assumption that complex models like LSTM are always superior. Conversely, Nazareth and Reddy [278] affirm LSTM's reliability for forecasting the sector-specific S&P BSE GREENEX Index, while emphasizing the importance of adjusting training window lengths to evolving market conditions, with longer windows during the pandemic and shorter ones post-pandemic to better capture shifting data dynamics. In our view, the key takeaway is that model effectiveness is context-dependent, and robustness under crisis conditions may favor simpler or dynamically tuned approaches.

Addressing Forecasting Challenges with Specialized Techniques Several studies have brought forward significant observations and findings. For instance, the Three-Point Moving Gradient (TPMG) method by Shoong et al. [340], an enhanced LSTM-based approach that predicts price gradients rather than absolute values, offering improved directional insight into market trends. AGNN [400] addresses market-style factor interference by using LSTM with attention and a cooperative network architecture, where subnetworks share a loss function based on stock rankings. The model outperforms ARIMA and LSTM, though it requires prior knowledge to guide effective interactions. N-BEATS [433] improves both performance and interpretability by incorporating past and future trends through backward-forward residual links and a deep stack of fully connected layers.

Fan and Zhang [106] compare multiple DL architectures (LSTM, GRU, CNN-LSTM, PatchTST) for both forecasting and nowcasting, showing that LSTM performs best overall. They also find that EMD is more effective for forecasting than nowcasting, and models generally perform better in nowcasting. Wu et al. [410] propose a novel parameterized Continuous Trend Labeling (CTL) method that improves return-based performance by refining how trends are labeled across models. For stock correlation forecasting, Luo et al. [246] design the novel CLATT model, integrating CNN, BiLSTM, and attention, with Bayesian hyperparameter tuning. Finally, Wang et al. [395] introduce the Multi-Dimensional Spatio-Temporal Fusion Transformer (MDSTFT) architecture, combining CNN, LSTM-attention, and Transformer layers to model complex spatial and temporal relationships in multivariate financial time series.

While these models demonstrate superior forecasting capabilities in controlled experiments, their generalization across market regimes and their computational demands remain open challenges.

Summary

The reviewed literature reflects a maturing yet fragmented landscape of DL applications in stock price forecasting. While LSTM remains a dominant baseline, empirical results increasingly favor alternatives such as GRU, ESN, and novel architectures like BiLTC and IRLSTM, especially when tailored to specific market conditions or enriched with technical and sentiment features. This diversity underscores the field's methodological innovation but also complicates cross-study comparability. The model effectiveness is highly context-dependent and is shaped by forecast horizon, data modalities, and asset volatility, highlighting the need for standardized benchmarks and rigorous ablation studies to disentangle architectural gains from feature engineering.

Recent literature on hybrid DL models for stock price prediction reveals a maturing convergence of architectures that integrate temporal, spatial, and attention-based components with preprocessing strategies such as denoising, dimensionality reduction, and clustering. These models aim to exploit the complementary strengths of their constituent modules, for instance, CNNs for extracting local patterns from raw price series, LSTM/GRUs for capturing long-term temporal dependencies, and attention mechanisms to selectively focus on the key information.

While many hybrid models (e.g., CNN-LSTM, LSTM-GRU, CNN-Transformer) achieve superior performance over traditional baselines, these benefits often come with trade-offs. Most notably, enhanced performance frequently incurs higher computational costs and greater sensitivity to hyperparameters. Moreover, interpretability tends to decline as models become more complex, particularly in architectures that fuse deep layers with multiple data sources (e.g., sentiment, financial indicators, and price signals), as it becomes harder to trace how input features influence the final predictions.

Although empirical results are promising, many studies focus on narrow asset classes or regions, limiting generalizability. Additionally, benchmarking remains inconsistent across works, with considerable variation in datasets, feature selection, and performance metrics. Less common metrics, some drawn from trading contexts, are used sporadically, showcasing the models' performance in real-life financial scenarios.

The rise of modular hybrid models, combining tailored preprocessing, architecture, and optimization, reflects growing flexibility in design. However, this also fragments the field, as varied setups hinder comparability. Unified evaluation protocols and broader cross-market testing are needed to assess generalizability and guide future model development.

Cross-study Observations Across the reviewed studies ($n = 75$), the most frequently employed models are standard (non-optimized) LSTM and LSTM-based hybrids ($\approx 61\%$ combined), followed by optimization-integrated architectures ($\approx$

11%), attention-enhanced models ($\approx 9\%$), and clustering- or decomposition-based frameworks ($\approx 7\%$). Despite the growing sophistication of hybrid designs, persistent challenges include computational complexity and sensitivity to hyperparameter settings.

TABLE 3: Cross-Study Overview of Stock Price Forecasting: Models, Assets, Features, and Timeframes

Category	Top Entities / Statistics
Most Common Models	Standalone: LSTM (31), ML (13), GRU (9); Hybrids: LSTM-based (15), CNN-LSTM-based (8). Total applications: 117 ¹² .
Standalone vs. Hybrid Use	Standalone: ~98%; Hybrid/ensemble models: ~52%; Novel architectures: ~21%. ³
Frequent Feature Types	Nearly all studies (~99%) use price-based features (OHLCV, Close, Adjusted Close); ~23% incorporate technical indicators; ~13% include sentiment (e.g., news, tweets). Notably, ~13% (10 studies) rely solely on Close price data. Rare features include stock bar chart images, financial factors, and correlation data.
Most Common Assets Forecasted	US stocks (e.g., Apple, Tesla, Amazon), Chinese A-shares, Indian stocks (e.g., Axis Bank), Indonesian equities (e.g., PT Bank Central Asia).
Geographic Focus	USA (NASDAQ, NYSE, S&P500, etc.): ~47% (35 studies); China (SSE, SZSE, CSI300): ~15% (11 studies); India (NSE, BSE): ~8% (6 studies); Indonesia: ~3% (2 studies); Multi-region/global: ~13% (10 studies).
Time Periods Covered	Most studies span 2000–2024, with a focus on 2010–2022. Long-term periods (>10 years) are used in ~45% (34 studies), while ~21% (16 studies) examine short periods (<5 years). Some datasets go back to the 1960s.
Sentiment-Integrated Models	10 studies combine market data with sentiment (from Twitter, news, forums). LSTM variants are most used (e.g., ABC-LSTM, Attention-LSTM), along with hybrid and embedding-based models (e.g., User2Vec).
Performance Metrics Used	The most commonly reported evaluation metrics include RMSE (~56%, 42 studies), MAE (~47%, 35), MAPE (~39%, 29), and MSE (~37%, 28). R^2 appears in ~29% (22 studies). Classification-oriented metrics such as Accuracy (~15%, 11), F1-Score (~13%, 10), Recall (~12%, 9), and Precision (~11%, 8) are less frequent.

¹ Some studies use multiple models (e.g., baselines, ensembles), contributing to total model count > number of papers (75). This reflects broader comparative experimentation.

² Model groups are defined as follows: LSTM-based (e.g., LSTM-ARIMA, PSO-LSTM, VAR-LSTM); CNN-LSTM-based (e.g., CNN-LSTM, CNN-LSTM-Transformer, CNN-BiLSTM).

³ Percentages represent the frequency of model appearances across the 75 studies. Since multiple models can be used within a single study, either as primary methods or for benchmarking, overlaps in counts are expected.

2.5.2 Index Forecasting

Rather than forecasting the price of individual stocks, many researchers focus on forecasting stock market indices, which tend to exhibit lower volatility due to their diversified composition, offering a better reflection of overall market momentum and economic conditions.

Standalone Architectures

The following section presents a systematic review and analysis of recent literature, emphasizing key trends and advancements in standalone DL models for stock market index forecasting.

LSTM-Based Approaches Recent research on standalone DL models for financial forecasting highlights the continued prominence of recurrent architectures, particularly LSTM networks, with effective performance across tasks and market conditions. For instance, Ta et al. [363] demonstrate that an LSTM-based approach to portfolio construction yields higher returns and more reliable risk management compared to classical approaches such as Equal-Weighted allocation, Monte Carlo Simulation, and Mean-Variance Optimization. This suggests that LSTM can effectively balance predictive power and decision support in investment contexts. LSTM also exhibits superior predictive performance over alternative architectures, outperforming MLP in forecasting cross-sector fluctuations of the Tadawul index [14], and surpassing both RNN and CNN models in Nifty50 index prediction [110].

Nevertheless, architectural refinements on LSTM are increasingly reported. Zhan and Kim [434], for example, improve LSTM's forecasting accuracy by employing the Tumbling Window technique with convolutional filters (Short-term Convolutional Filter), albeit at the cost of increased training time. In another direction, Tong and Yin [371] take a step toward decision-aware forecasting by integrating LSTM with reinforcement learning, offering promising results for adaptive trading strategies that respond dynamically to market conditions. This aligns with broader literature emphasizing LSTM's robustness in index prediction tasks.

Beyond LSTM Architectures GRU, CNN, and TCN models have also demonstrated competitive performance. For instance, Yang and Guo [421] demonstrate that GRU can outperform traditional statistical models such as ARMA, BVAR, and BP in forecasting inflation trends in China, highlighting its strength in capturing non-linear temporal dynamics with fewer parameters than LSTM. CNN-based models have also evolved in response to common limitations; to mitigate information loss during image transformation, Kirisci and Cagcag Yolcu [206] propose a modified CNN architecture specifically designed for financial time series, improving forecast accuracy for indices such as TAIEX and FTSE. Notably, Yilmaz and Yildiztepe [427]

find that TCNs outperform LSTM, GRU, ARIMA, and Random Walk models in stock return forecasting. Their success is largely attributed to the TCNs' causal, dilated convolution structure, which allows for efficient modeling of long-term dependencies without the vanishing gradient issues typically faced by recurrent networks.

Integrating Technical Analysis Several studies employ technical indicators to enhance model performance. [256] evaluate DL models for Nifty50 price prediction using two technical indicator datasets: TA1 (SMA, EMA, MACD, RSI, SMI, RVI, EMV, RN) and TA2 (SMA, EMA, MACD, RSI). Comparing three GRU variants with RNN and LSTM models, they find that GRU with TA1 performs best, emphasizing the great relevance of SMI and RN for index prediction. In contrast, Yan et al. [419] investigate the impact of different input features, and conclude that basic trading indicators and the CSI300 Index contribute most significantly to prediction performance. They further observe that technical indicators may lead to information loss, while financial indicators provide limited value for short-term forecasting.

Hybrid Models

The following section reviews recent literature on hybrid DL models for stock market index forecasting.

Fusions of LSTM and Beyond A growing body of literature explores hybrid models combining statistical and DL techniques. Donghwan et al. [93] present P-LSTM, a pattern-driven model that combines BIRCH clustering with LSTM, enabling the training of separate LSTM models on similar data subsequences to enhance stability in heterogeneous financial data. Song et al. [349] propose NR-LSTM, separating trend and residual components for improved bearish market performance. Other works [231, 413] follow a similar hybrid paradigm, coupling ARIMA with ML/DL models to capture linear–nonlinear patterns. Jia et al. [186] propose ResNLS, integrating ResNet for feature extraction and LSTM for temporal modeling, while Anjum et al. [22] propose a CNN-LSTM-RNN architecture to tackle noise, non-linearity, and volatility in financial data. Beyond LSTM-based models, ESNs have shown superior predictive accuracy and efficiency for non-stationary chaotic (Vosvrda macroeconomic system) and financial datasets (S&P 500 index) [135].

Dimensionality Reduction for Forecast Enhancement Mansoor et al. [258] propose using Mutual Information for feature selection, PCA for dimensionality reduction while preserving significant data variations, and LSTM for improved stock market forecasting. Similarly, Wang et al. [393] integrate PCA with an Improved GRU (IGRU) model, incorporating an Antioversaturation Conversion Module to enhance learning sensitivity and prevent oversaturation.

Novel Model Architectures Recent innovations reveal a growing trend toward hybrid models that blend decomposition, optimization, and chaos theory. Notable innovations include the hybrid CNN-LSTM model by Hao and Gao [154], which captures multi-scale features for trend forecasting; the Polylinear Regression with Augmented LSTM Neural Network (PLR-ALSTM-NN) by Ahmed et al. [8], designed for multivariate time series forecasting with strong long-term predictive capabilities; and the LSTM-BN model by Fang et al. [108], which enhances high-frequency movement forecasting by integrating Batch Normalization, an adaptive cross-entropy loss function, and a multi-path parallel structure with a voting mechanism.

Models like BiCuDNNLSTM-1dCNN [200] and Multi-Filters Neural Network (MFNN) [242] effectively combine CNN and RNN elements to capture both spatial and temporal dependencies in financial data. Others, such as the ResNet18-attention model [280] and AE-ACG [226], introduce advanced feature extraction and attention mechanisms to address multivariate complexity and enhance temporal representation. Notably, the use of DRL with GRU [23] highlights a shift toward decision-oriented forecasting, prioritizing not just prediction but actionable outcomes for automated trading. The SF-Transformer [259] marks a promising step in integrating domain-specific knowledge, like Spot-Forward parity, into Transformer architectures, suggesting improved long-term forecasting accuracy and reduced uncertainty. Zhang et al. [435] develop a 1D-CapsNet-LSTM that enhances long-horizon forecasts outperforming LSTM, RNN, and CNN-LSTM models.

Extending beyond LSTMs, advanced architectures like the Deep Convolutional GAN (DCGAN) [356] combine a CNN-BiLSTM generator with a CNN discriminator to simulate trader learning. The generator produces initial price predictions, which are iteratively refined based on discriminator feedback comparing predictions to actual data. This model outperforms ARIMAX-SVR, RF, LSTM, and standard GANs in both single- and multi-step forecasting tasks. Wu et al. [410] introduce Continuous Trend Labeling, improving classification accuracy and yielding higher net returns than traditional labeling and buy-and-hold strategies. [260] present three hybrid models, CNN with Technical Indicator Clustering (CNN-TIC), Evolutionary Optimized CNN-TIC (EO-CNN-TIC), and Residual Network with Technical Analysis (ResNet-TA), which incorporate *take profit* and *stop loss* techniques for automatic trading, with EO-CNN-TIC further optimized using Cuckoo Search for better trading precision.

Novel Decomposition-Driven Architectures

Towards decomposition-based architectures, a variety of hybrid models, such as EMD/CEEMD-based CNN-LSTM [310], and Akima-EMD-LSTM [17], demonstrate the efficacy of decomposing input signals into intrinsic mode functions (IMFs) for denoising and dimensionality reduction prior to sequential modeling. More advanced architectures, including CEEMDAN-TCN-GRU-CBAM [227] and GRU-CEEMDAN-Wavelet [301], enhance forecasting by applying fuzzy entropy for IMF reorganization and wavelet denoising to suppress noise in high-frequency components, thereby improving feature quality and model performance. In a

broader study, Yañez et al. [420] demonstrate that Transformer-based models, particularly when combined with CEEMDAN decomposition, consistently outperform LSTM-based, CNN-LSTM, and non-decomposed variants across global stock indices, with CEEMDAN-CNN-Transformer achieving top performance in 70% of cases. This highlights the combined benefit of frequency decomposition and advanced attention-based architectures. Meanwhile, leveraging entropy-based analysis, the Hierarchical Decomposition-based Forecasting Model (HDFM) [230] integrates CEEMDAN for initial signal decomposition, K-means clustering of IMFs based on entropy levels, and VMD for further decomposition of high-frequency components, before employing GRU for final predictions. Similarly, Akşehir and Kılıç [10] propose ICE2DE-MDL, leveraging a two-level decomposition strategy through ICEEMDAN and entropy-driven IMF classification, ensuring the retention of valuable high-frequency components while eliminating noise.

Novel Decomposition-Optimization-based Architectures

Several novel methods integrating decomposition, and optimization techniques have been proposed to enhance stock price forecasting. Liu and Long [234] introduce the EWT-dpLSTM-PSO-ORELM model, combining EWT for pre-processing, dropout-enhanced LSTM (dpLSTM) for forecasting, PSO for hyperparameter tuning, and ORELM for error correction post-processing. Deng et al. [88] propose MEMD-LSTM, leveraging MEMD to decompose multivariate time series into high-frequency, low-frequency, and residual signals, addressing issues such as mode mixing and scale misalignment inherent in conventional EMD methods, and LSTM optimized via orthogonal array tuning method. Wang et al. [392] develop SSA-PSO-LSTM, integrating SSA for decomposing and reconstructing the stock price time series into independent components, PSO for LSTM optimization, and feature selection of inputs, including technical indicators, based on Pearson correlation coefficients. Yan et al. [418] propose CEEMD-PCA-LSTM for one-step-ahead stock price prediction, integrating CEEMD for noise reduction and decomposition into IMFs, PCA for dimensionality reduction and feature extraction, and LSTM for sequential forecasting.

Novel Chaos-Driven Approaches

Through novel insights, chaos-based hybrid models have been developed to better capture the complex, noisy, and nonlinear dynamics of financial time series. DURAJ and MOHAN [94] introduce the Chaos-LSTM-PR, integrating Chaos Theory, LSTM, and PR to enhance prediction accuracy across foreign exchange, commodity prices, and stock indices. The approach begins by testing the FTS for chaotic behavior using the Lyapunov exponent. Chaos Theory is then applied to model the chaotic dynamics of the time series by reconstructing the phase space. For the creation of the phase space, it is necessary to determine the ideal lag from the time series, and the embedded dimension of the associated space. The Akaike Information Criterion (AIC) is used to determine the optimal lag from time series and Cao's method is used to achieve the optimal embedding dimension. The modeled time series serves as input for an LSTM model to generate initial forecasts. The sequence of errors derived from LSTM forecasts is PR appropriate for error predictions. Error

forecasts and original model forecasts are applied to produce the final hybrid model forecasts. The proposed Chaos-LSTM-PR outperforms ARIMA, Prophet, LSTM, and Chaos-LSTM, in terms of MSE, D-statistic, and Theil's U. Building on this, Durairaj et al. [97] further validate the Chaos-LSTM-PR model on the same datasets over an extended period, comparing it against a broader range of models, including ARIMA, Prophet, LSTM, CNN, CART, RF, Chaos-CART, Chaos-RF, and Chaos-LSTM, using metrics such as MSE, MAPE, Dstat, and Theil's U. The findings reaffirm Chaos-LSTM-PR's superiority, consistently outperforming all counterparts. Similarly, Durairaj and Mohan [95] propose Chaos-CNN-PR, which leverages CNN for initial forecasts of chaos-checked financial time series, refined by PR. This model achieves superior performance over traditional models (ARIMA, Prophet, CART, RF, CNN) and chaos-augmented models (Chaos-CART, Chaos-RF, Chaos-CNN) in forecasting exchange rates, commodity prices, and stock market indices.

Signal Decomposition and Denoising Architectures Several studies leverage decomposition techniques to improve the modeling of complex dynamics in financial time series. Fan and Zhang [106] utilize EMD to isolate underlying temporal patterns, finding that while EMD enhances forecasting by capturing long-term structures, it offers limited benefits for nowcasting, where recent fluctuations dominate. Guo et al. [151] extend this idea with EEMD-CNN-GRU, decomposing the input series into trend, low-frequency, and high-frequency components, each modeled separately and then aggregated, highlighting the dominance of the trend component in driving predictive accuracy. In a similar vein, Song et al. [350] apply VMD to separate financial signals into trend, periodic, and nonlinear disturbance components, where the linear components are assigned to ARIMA, and the nonlinear disturbances to LSTM. Similarly, Wen et al. [401] adopt a multilevel wavelet decomposition (mWDN) approach to capture long-term and short-term cyclic patterns, addressing data leakage and boundary issues, which can skew forecasting performance. LSTM is then used for prediction.

Other approaches incorporate denoising techniques, such as DAE-LSTM, for detection of rising change points [429] and wavelet denoising-enhanced hybrid LSTM and DB-ARIMA for smoothing data and revealing underlying trends [247].

Optimization-based Architectures Optimization-based architectures enhance DL models for stock prediction by improving convergence, feature selection, and parameter tuning. Studies such as Chen and Zhou [70] use GA to rank and select the most influential financial and technical factors for LSTM input, while Kumar et al. [210] introduce an Adaptive PSO-LSTM hybrid model that optimizes LSTM weights and biases, effectively addressing challenges such as overfitting and being stuck to local minima. Novel techniques leveraging PSO are introduced by Chauhan et al. [67], who utilize PSO for hyperparameter tuning of LSTM and GRU models. The hybrid (PSO-LSTM) + (LSTM-GRU) ensemble outperforms in terms of accuracy, precision, recall, and F1-score. Similarly, Ji et al. [185] propose the Improved PSO (IPSO)-LSTM model, which enhances optimization by incorporating an adap-

tive mutation factor, to avoid premature convergence to local optima, and a non-linear approach to improve the PSO's inertia weight. Other works, such as such as Gülmez [149] and Salahshour et al. [319], leverage ARO's balance of exploration and exploitation to optimize LSTM performance.

Integrating Technical and Sentiment Analysis Focus on Technical Analysis

Recent studies increasingly leverage technical indicators within hybrid DL frameworks to better capture stock market behavior. The CNN3D-DR+LSTM [422] converts technical indicators into deterministic trend signals and organize them in a 3D tensor using correlation-based ranking, enhancing CNN performance for better price prediction. The HW-LSTM-RL [218] use wavelet-denoised high-frequency components as input to an LSTM for reinforcement learning, focusing on optimizing trading strategies rather than direct prediction. Similarly, the Attention-enhanced Compound Neural Network (AComNN) [425] effectively models complex cross-regional (U.K., U.S., and China) financial dependencies using self-attention and a diverse set of technical and macroeconomic features, resulting in improved prediction accuracy and robust trading performance.

Hybrid architectures combining LSTM with ensemble methods, such as the LSTM-Forest model [289], incorporates multi-task learning for stock return prediction and direction classification, with a multi-loss function combining MSE and cross-entropy. Cross-market models combining wavelet transforms with multi-input LSTM [365] utilize technical indicators and data from both Chinese and U.S. stock markets, leveraging inter-market correlations to improve forecasting accuracy. Additionally, the PCA-ICA-LSTM framework [325] employs PCA to reduce dimensionality and noise, followed by ICA for extracting independent features, thereby improving data quality and boosting return rates without sacrificing accuracy. Notably, Kalra et al. [198] propose a novel hybrid bidirectional LSTM (H.BLSTM) model that combines incremental learning and DL techniques for real-time index price prediction. By leveraging univariate and multivariate time series data, including technical indicators, the model outperforms benchmark models across nine global stock indices with an average forecasting delay of just 2 seconds, making it ideal for real-time trading.

Focus on Sentiment Analysis

Li [222] demonstrate that extracting deep emotional information via CNNs, rather than relying on basic emotional features, enhances predictive performance, particularly when integrating multiple information streams. Building on this, Liu et al. [237] propose SA-TrellisNet, a news-driven stock market index prediction model combining an LSTM-CNN-based sentiment analysis with a sentiment attention mechanism to dynamically weight stock price features based on their sentiment index, reflecting their significance. Meanwhile, Abdullah et al. [3] construct sentiment indices (WHTCSI, HTCSI) using social media (Twitter) and Google Trends data (2006-2021) to forecast Halal tourism stock prices, showing strong predictive power via CNN over ARIMA, SVR, MLP, and LSTM models.

Focus on Technical - Sentiment Analysis

Integrating sentiment with technical indicators improves stock prediction, as shown by Prachyachuwong and Vateekul [300], who combine LSTM and BERT on sector-segmented economic news headlines and price data. Similarly, Gangopadhyay and Majumder [130] find that combining text and price data outperforms models based on price or text alone, with CNNs capturing price sequences well and Transformers, like BERT and FinBERT, excelling at text understanding.

Financial Forecasting in Crisis and Pandemic Contexts Extreme events like the COVID-19 pandemic act as real-world stress tests, exposing the strengths and limitations of financial forecasting models under pressure. Exploring LSTM-based forecasting under COVID-19 conditions, Lachaab and Omri [213] find KNN to outperform LSTM and ARIMA for CAC40 forecasts, highlighting the positive effect of containment and vaccination measures. In contrast, other works, such as Bathla et al. [43] and Nazareth and Reddy [278], confirm LSTM's strong performance on global and sector-specific indices, though challenges remain with extreme price fluctuations. Comparative studies like Ampountolas [20] and Widiputra and Juwono [404] highlight the advantages of hybrid and adaptive models in pandemic-era forecasting. Ampountolas [20] show that the ETS-ANN model, blending linear (ETS) and nonlinear (ANN) components, consistently yields the lowest error rates across European indices and cryptocurrencies during pre-, mid-, and post-pandemic periods, despite some occasional drawbacks. Meanwhile, Widiputra and Juwono [404] demonstrate that forecasting performance depends heavily on data characteristics: LSTM performs best with stable inputs, while CNN-LSTM better handles volatility. Vuković et al. [389] examine market efficiency using LSTM, ARIMA, and LightGBM, revealing structural inefficiencies during crises, such as the 2007–2009 financial crisis and COVID-19 pandemic, challenging EMH assumptions and supporting the AMH framework. These findings underscore the importance of context-sensitive model selection and hybrid architectures that balance complexity with adaptability.

Advanced Modeling Techniques and Approaches Recent efforts focus on practical deployment and robustness. For example, Barra et al. [40] employ CNN ensembles on Gramian Angular Fields images to extract spatial patterns for S&P500 trend prediction. Touzani and Douzi [372] illustrate cross-market learning by training LSTM and GRU models on U.S. and French market data to predict (short- and medium-term) Moroccan stock prices, highlighting the potential of international data integration. Detecting and compensating for outliers, Naidoo and Du [272] propose a two-stage DL framework, with an LSTM-based auto-encoder for anomaly detection, and an HONN to forecast stock prices. Wu et al. [411] enhance standard DL models (MLP, RNN, LSTM, GRU) with transfer learning using industrial chain data, improving stock market index trend prediction accuracy, especially for MLP.

Both Wang and Lee [394] and Aryal et al. [26] underscore the increasing effectiveness of adopting multifractal analysis to better capture the complex and nonlinear dynamics inherent in financial markets. Wang and Lee [394]’s neuro-inspired EINS model, combined with MultiFractal Asymmetric Detrended Cross-Correlation Analysis (MF-ADCCA), demonstrates notable improvements in volatility forecasting, suggesting that biologically motivated designs can enhance model adaptability to financial data intricacies. Complementing this, Aryal et al. [26]’s comparative study highlights that while architectures like N-BEATS, ES-LSTM, and TCN excel in stock market forecasting, LSTM variants and Encoder-Decoder models are more suited to forex markets, reflecting the necessity of model selection tailored to asset class characteristics.

Summary

Recent advances in stock market index forecasting utilize DL models, with LSTM, CNN, and GRU being the most commonly used standalone architectures. Notably, hybrid models, particularly those combining CNN and LSTM, shows exemplary forecasting performance. However, due to the inherently noisy nature of financial data, hybrid approaches often incorporate signal decomposition techniques, tailored optimization, and integration of technical and sentiment features to further enhance prediction accuracy.

Different market components contribute differently to prediction accuracy, highlighting the importance of decomposition in forecasting. Techniques such as EMD and wavelet transforms isolate intrinsic signal components, enabling models to focus on more stable patterns and better handle market volatility and regime shifts. However, while denoising improves signal clarity, it risks smoothing out high-frequency patterns vital to short-term forecasting, requiring careful tuning. Additionally, given the often marginal gains over simpler models, the added complexity of decomposition methods should be carefully justified in practical applications.

Incorporating diverse data sources, such as technical indicators and sentiment analysis from social media and news, has demonstrated clear benefits. This multi-dimensional input enriches the feature space, offering a broader context for price movements and enhancing forecast accuracy. Nevertheless, challenges including overfitting, and real-time computational demands remain critical concerns.

Innovative approaches highlight the growing sophistication of DL architectures and the relevance of hybrid model design in modeling market anomalies and structural breaks. These studies underscore the practical adaptability of DL models and the value of domain-specific augmentation (e.g., outlier handling, transfer learning) for real-world forecasting. Integration of multifractal dynamics, neurodynamics, and sentiment analytics represents a promising direction for modeling complex, behavior-driven financial systems.

Cross-study Observations Across the reviewed studies ($n = 83$), standard LSTM and LSTM-based hybrid models are most frequently employed ($\approx 69\%$ combined), followed by integrations of technical and sentiment analysis ($\approx 14\%$, 12 studies), as well as optimization techniques, signal decomposition-denoising methods, and crisis or pandemic-focused approaches, each representing $\approx 7\%$ (6 studies).

TABLE 4: Cross-Study Overview of Index Forecasting: Models, Assets, Features, and Timeframes

Category	Top Entities / Statistics
Most Common Models	Standalone: LSTM (25), CNN (9), GRU (9); Hybrids: LSTM-based (32), CNN-LSTM-based (8). Total applications: 131 ^{1,2} .
Standalone vs. Hybrid Use	Standalone: ~78%; Hybrid/ensemble models: ~77%; Novel architectures: ~31%. ³
Frequent Feature Types	Price-based features (e.g., OHLCV, Close, Adjusted Close) appear in most studies ($\approx 92\%$); technical indicators appear in $\approx 22\%$ of cases; sentiment features (e.g., news, tweets) in only $\approx 6\%$. Notably, $\approx 28\%$ (23 studies) rely solely on Close price data. Rarely used features: chaotic-modeled time series, dividend yields, GAF images.
Most Forecasted Stock Indices	U.S. indices dominate, led by S&P500 (~37%), DJI (~18%), and NASDAQ (~10%). Chinese indices follow with SSE (~19%) and CSI300 (~11%), along with HSI from Hong Kong (~8%) and Nifty50 from India (~7%).
Geographic Focus	Global or mixed-region studies dominate (~40%, 33 studies), followed by the single-study cases U.S. (~22%, 18 studies) and China (~18%, 15 studies). Other regional focuses include India (~6%, 5 studies), Europe (~4%, 3 studies), and Pakistan, Saudi Arabia, South Korea, and Thailand (each ~1%).
Time Periods Covered	Most studies use long-term datasets over 10 years (~70%, 58 studies), while fewer focus on periods <5 years (~28%, 23 studies). The earliest dataset begins in 1983, with many extending into the early 2020s. The 2010–2021 decade is the most studied (~20%, 17 studies). Some analyze multiple periods, but overall, longer historical datasets are preferred.
Performance Metrics Used	Most commonly reported evaluation metrics: RMSE (~58%, 48 studies), MAE (~54%, 45), MAPE (~49%, 41), and MSE (~37%, 31). R^2 appears in ~23% (19 studies). Classification-oriented metrics such as Accuracy (~19%, 16), F1-Score (~10%, 8), Precision (~10%, 8), and Recall (~8%, 7) are less frequent. A few studies also report Theil's U (5), and SMAPE (4).

¹ Some studies use multiple models (e.g., baselines, ensembles), contributing to total model count > number of papers (83). This reflects broader comparative experimentation.

² Model groups are defined as follows: LSTM-based (e.g., LSTM-ARIMA, PSO-LSTM, VAR-LSTM); CNN-LSTM-based (e.g., CNN-LSTM, CNN-LSTM-Transformer, CNN-BiLSTM).

58 Percentages represent the ratio of the total number of model appearances to the 83 studies reviewed. Note that individual studies may include multiple models, including both standalone and hybrid types.

2.5.3 Forex Price Forecasting

Foreign exchange (forex) markets are the largest in terms of trading volume, operating 24/7. With an average daily trading volume exceeding 7 trillion USD [49], forex is the most liquid and heavily traded financial market worldwide. The rise of online trading platforms providing real-time market access to diverse options has driven the demand for profitable trading strategies. As a result, a significant body of research on forex forecasting using DL models has emerged. Given the dominance of the US Dollar in global transactions, particular emphasis is placed on the EUR/USD currency pair, which accounts for over 80% of the total forex trading volume [426].

CNN-Based Architectures Recent studies on forex price forecasting demonstrate the growing sophistication of hybrid DL architectures.

Hybrid approaches combining CNN-based candlestick image processing with CNN-LSTM time series analysis improve profitability by classifying moving average crossover signals into profitable long, short, or non-profitable trades, outperforming traditional benchmarks [268]. However, CNN-based image transformation methods are limited by their dependence on input quality and data volume, reducing robustness in market trend classification for trading [377]. More broadly, evidence from stock and forex markets suggests that CNNs generally underperform due to their limited capacity to effectively incorporate past data [190].

Fusions of LSTM and Beyond Alternatively, LSTM and transformer-based methods show strong forecasting capabilities. Integrating XGBoost for feature selection with LSTM temporal modeling enhances predictive accuracy over classical baselines such as ARIMA [390]. Novel dual-LSTM architectures that separately model macroeconomic and technical indicators improve directional prediction, though at the cost of increased complexity [426]. Evaluations across different asset classes further show the strong performance of LSTM variants and encoder-decoder architectures in forex forecasting [26], while hybrid CNN-LSTM models have demonstrated effectiveness across stocks, forex, and cryptocurrencies [4]. Transformer architectures with time embeddings demonstrate promising results in short-term intraday forecasting, outperforming LSTM and ARIMA models, especially when incorporating multivariate inputs [116].

Chaos-based Advanced Modeling Techniques It is noteworthy that novel chaos-based hybrid models, extensively discussed in index forecasting literature, address nonlinear, noisy, and chaotic market behaviors by integrating chaos theory with DL models and post-processing refinements such as polynomial regression [94, 95, 97]. These approaches consistently outperform traditional statistical and DL models, emphasizing the value of domain-specific augmentations to better capture financial time series complexities. To avoid redundancy, detailed discussion of these chaos-based models is presented in the index forecasting section,

where their methodological innovations and empirical superiority are thoroughly described.

Summary

The literature on DL-based forex forecasting reveals a strong trend toward hybrid and multivariate modeling frameworks. CNN-based models, particularly those leveraging candlestick chart transformations, offer a promising approach to signal classification. However, their performance is often hindered by sensitivity to input quality and an inherent limitation in capturing temporal dependencies [190, 268, 377]. These findings align with a broader skepticism regarding CNNs for financial time series prediction, especially when past temporal context is crucial.

In contrast, LSTM architectures, especially those enhanced through feature selection (e.g., XGBoost) or hybridized with macroeconomic and technical indicators, consistently demonstrate strong predictive performance [390, 426]. Transformer-based models with time embeddings have emerged as a compelling alternative, particularly for high-frequency, short-horizon forecasting tasks [116]. My impression is that transformers are underutilized in forex relative to their potential, especially in scenarios requiring attention to multiple time scales and cross-asset dependencies.

The introduction of chaos-theory-enhanced models presents an intriguing deviation from conventional DL approaches. Evaluated under index forecasting contexts, their success in capturing market nonlinearity and noise suggests significant untapped potential in forex prediction as well [94, 95, 97].

Cross-study Observations Across the 11 reviewed studies, LSTM models, both standalone and hybrid, are predominant, appearing on average 1.18 times per study, indicating that each study uses at least one LSTM variant. Technical analysis features ($\approx 36\%$, 4 studies) and chaos-based models (3) are also commonly explored in forex forecasting. Notably, the US dollar consistently serves as the primary base currency, highlighting its dominant role in this research area.

TABLE 5: Cross-Study Overview of Forex Forecasting: Models, Assets, Features, and Timeframes

Category	Top Entities / Statistics
Most Common Models	Standalone: LSTM (6), CNN (4); Hybrids: LSTM-based (7). Total applications: 27 ¹² .
Standalone vs. Hybrid Use	Standalone: ~154%; Hybrid/ensemble models: ~90%; Novel architectures: ~36%. ³
Frequent Feature Types	Price-based features (e.g., OHLC, Open, Close) appear in most studies ($\approx 90\%$, 10 studies); close price data and technical indicators each appear in $\approx 36\%$ (4 studies). Rarely used features include chaotic-modeled time series and macroeconomic indicators.
Most Forecasted Forex Currency Pairs	EUR/USD leads ($\approx 55\%$, 6 studies); INR/USD, JPY/USD, and SGD/USD, follow, each appearing $\approx 27\%$ (3 studies). Notably, the US dollar serves as the primary base currency in forex forecasting studies, reflecting its global market dominance.
Time Periods Covered	Several studies use long-term datasets over 10 years ($\sim 45\%$, 5 studies), while fewer cover periods under 5 years ($\sim 27\%$, 3 studies). The longest dataset spans 32 years (1990–2021) and appears in two studies.
Performance Metrics Used	The most frequently reported evaluation metrics are MSE ($\sim 45\%$, 5 studies), RMSE and MAE (each $\sim 36\%$, 4), followed by MAPE, D-statistic, and Theil's U (each $\sim 27\%$, 3). Less commonly employed metrics include classification-based measures as well as trading-specific indicators such as the number of transactions and profit accuracy.

¹ Some studies use multiple models (e.g., baselines, ensembles), contributing to total model count > number of papers (11). This reflects broader comparative experimentation.

² LSTM-based models include Chaos-LSTM-PR, ES-LSTM, Stacked-LSTM, and XGBoost-LSTM

³ Percentages represent the ratio of total model appearances (standalone, hybrid, novel) to the number of studies (11), reflecting how frequently models appear per study on average. For example, 154% means standalone models appeared on average 1.54 times per study, implying each study uses at least one standalone model. Note that individual studies may include multiple standalone and/or hybrid models.

2.5.4 Commodity Price Forecasting

Commodity forecasting predicts the future prices of physical goods like oil, gold, agricultural products, and metals. Given their key role in global trade and sensitivity to factors such as supply and demand, geopolitical events, and economic conditions, accurate forecasting is vital for investors, businesses, and policymakers.

Leveraging Recurrent Architectures Recent studies leverage recurrent neural networks to enhance predictive performance. Halim et al. [152] demonstrate that an attention-enhanced multivariate LSTM outperforms baseline models (RNN, CNN, Transformer) in forecasting prices of 11 key Indonesian food commodities. In a related effort, Huang et al. [168] propose a CNN-GRU hybrid augmented with PCA for dimensionality reduction. Their findings on copper and soybean meal futures reveal that hybridization not only enhances prediction accuracy but also reduces computational cost, compared to individual CNN or GRU models, a valuable trait for real-time forecasting in commodity markets.

Integrating Reinforcement Learning and Interpretability Incorporating learning agents, Tong and Yin [371] and Ansari et al. [23] explore reinforcement learning-based systems for commodity trading. The former employs LSTM in a high-frequency trading setup for corn futures, outperforming a buy-and-hold strategy in risk-adjusted returns. The latter proposes a novel DRL-based decision support system integrating GRU forecasting with reinforcement learning. By enhancing the agent's observation state with both historical and predicted trends, this system enables fast training, with profitable trading decisions across multiple markets. These approaches underscore the potential of reinforcement learning to move beyond passive forecasting toward active decision-making, a shift that aligns well with the growing emphasis on autonomous financial systems.

More recently, Wu et al. [409] contribute a novel interpretable forecasting framework for corn futures price prediction, that marries signal decomposition (CEEM-DAN), CNN-based text mining, and an optimized Temporal Fusion Transformer (TFT) via adaptive differential evolution with optional external archiving (JADE). The model's use of multisource heterogeneous inputs, including supply-demand dynamics, policy changes, geopolitical risks, and market sentiment, resulting in highly accurate and interpretable weekly price forecasts, marks a step forward in modeling the complex drivers of commodity prices. Importantly, their emphasis on interpretability adds transparency to the modeling process, an often-overlooked but increasingly vital requirement in financial applications.

Chaos-based Advanced Modeling Techniques Finally, chaos-based methods, extensively discussed in the index forecasting section, also show promise in commodity markets, in addition to forex. The Chaos-LSTM-PR model by DURAIRAJ and MOHAN [94] and its subsequent extensions [95, 97] consistently outperform traditional approaches and chaos-augmented models across crude oil, gold, and soybeans. These results suggest that capturing chaotic structures may be particularly useful in markets characterized by nonlinearity and structural breaks, common features in commodity time series, and other asset classes.

In our view, while LSTM and hybrid DL models remain dominant, emerging approaches that emphasize decision support, interpretability, and heterogeneous data integration represent the most promising directions. The integration of reinforcement learning and decomposition techniques, in particular, signals a move toward more robust, adaptive, and explainable forecasting frameworks that better reflect the real-world complexity of commodity markets.

2.5.5 Bond Price Forecasting

Bond forecasting involves predicting the future prices or yields of debt securities issued by governments or corporations, with some experts arguing that bond prices better reflect economic health than the stock market. Only two studies were identified in our review that focus explicitly in this field. Both Xu et al. [414] and Ping et al. [299] emphasize the critical role of feature dimensionality reduction techniques, such as LASSO and PCA, in enhancing predictive performance when combined with LSTM architecture. Xu et al. [414] propose a hybrid LASSO-PCA-LSTM model to predict the monthly return rate of the Securities Bank of China, demonstrating that this combination yields superior accuracy than similar models that use GRU or RNN in place of LSTM. Similarly, Ping et al. [299] present a more elaborate framework integrating LASSO, SMLR, PCA, and Bayesian optimization with LSTM to forecast treasury bond yields. Their systematic evaluation shows that stepwise integration of these techniques incrementally improves model performance, with the LASSO-SMLR-PCA-LSTM configuration outperforming a wide range of baseline models, including BPNN, CNN, RNN, Transformer, and several other LSTM-based configurations such as LASSO-LSTM, PCA-LSTM, LASSO-SMLR-LSTM, and LASSO-PCA-LSTM.

These findings highlight the utility of integrating feature selection, dimensionality reduction, and recurrent networks, especially LSTM, in bond price prediction, though the limited research to date limits broad conclusions. In our view, these studies primarily reinforce LSTM's strong predictive capabilities, even in more stable asset classes like bonds, with specific enhancements further strengthening performance. Looking ahead, future research should aim to explore emerging architectures like ESNs, and enrich input data with alternative sources such as inflation expectations, employment statistics, and ESG-related indicators, to better capture bonds' sensitivity to macroeconomic and policy shifts, especially in the context of green finance.

2.5.6 Cryptocurrency Price Forecasting

Cryptocurrencies, as peer-to-peer digital currencies, rely on blockchain technology to securely record transactions within individual blocks [199]. The decentralized nature of blockchain provides advantages such as low transaction costs, seamless circulation, and enhanced security [229]. Bitcoin, introduced by [275] as the first

decentralized cryptocurrency, has since evolved into the most valuable digital currency worldwide.

GRU-Based Forecasting Several studies underscore the effectiveness of GRU models in cryptocurrency price prediction. GRU has been shown to outperform both LSTM and BiLSTM in forecasting the prices of cryptocurrencies such as Bitcoin, Ethereum, and Cardano [18, 99], and it also demonstrates superior accuracy compared to LSTM, CNN, and CNN-LSTM models in predicting price fluctuations of the NEAR cryptocurrency [138]. This suggests GRU's robustness across diverse crypto assets. However, Al-Nefaie and Aldhyani [13] challenge the dominance of GRU by showing that an MLP yields better predictive accuracy for Bitcoin, indicating that non-recurrent models may be competitive when features are carefully engineered or when temporal dependencies are minimal.

LSTM and Temporal Modeling Despite GRU's strong performance, LSTM networks remain prominent in forecasting cryptocurrencies such as Bitcoin, Ethereum, and Dogecoin, consistently outperforming other RNN variants, CNN models [273], and traditional methods like ARIMA [117], largely due to their superior ability to capture long-range temporal dependencies. High-frequency data further validate LSTM's strengths. Bouslimi et al. [58] analyze minute-level Bitcoin and Ethereum prices during COVID-19, revealing that LSTM, particularly for Ethereum, maintains high accuracy in volatile environments.

Alternative Models: FB-Prophet and ARIMA In contrast to DL models, Tripathy et al. [374] evaluate FB-Prophet, ARIMA, and LSTM, reporting that FB-Prophet achieves the highest predictive accuracy for Bitcoin. This result is noteworthy given FB-Prophet's simplicity and interpretability, highlighting its utility in scenarios with strong seasonality or trend components. El Mahjouby et al. [103] conduct a broader comparison involving linear regression, SGDRegressor, random forest, AdaBoost, LSTM, and ARIMA on CoinMarketCap and CryptoCurrency datasets. The study finds that ensemble and linear models (e.g., random forest regressor and linear regression) often outperform DL models. Although ARIMA performs competitively, the study ultimately advocates for a hybrid approach that combines multiple models to leverage their complementary strengths.

Hybrid Architectures and Advanced Modeling Techniques Hybrid DL models have shown notable improvements in cryptocurrency forecasting by combining the strengths of different architectures. For instance, the 1dCNN-GRU model [199] effectively integrates CNN's pattern recognition with GRU's temporal modeling, outperforming various benchmarks across Bitcoin, Ethereum, and Ripple cryptocurrencies. Similarly, the LSTM-GRU hybrid [291] improves short-term prediction accuracy over LSTM-based models for forecasting Litecoin and Monero prices. Other

hybrids like GRU-BiLSTM [111] and CNN-LSTM [4] extend these gains across diverse assets and markets. Additionally, the PSO-optimized Improved Deep Belief Network (IDBN) [229] exemplifies how combining neural architectures with optimization techniques can further boost accuracy.

In the context of the natural stress-testing scenario brought about by the COVID-19 pandemic, a comparative analysis by Ampountolas [20] reveals that hybrid models like ETS-ANN outperform traditional parametric (ARIMA) and non-parametric (kNN) approaches in forecasting both stock market indices and cryptocurrency (BTC/USD) prices. Despite this improvement, the study highlights the persistent challenge of achieving high predictive accuracy in volatile financial markets, where forecasting performance often remains limited. To address these limitations, ensemble learning techniques have shown strong potential in enhancing predictive performance. Livieris et al. [239] demonstrate that combining DL architectures, such as LSTM, BiLSTM, and convolutional layers, within ensemble frameworks (e.g., averaging, bagging, stacking) consistently outperforms individual models for cryptocurrency prices and directional movement prediction. Notably, stacking with a kNN meta-learner yields superior results, suggesting that combining diverse model perspectives can better capture market complexity. Further extending the frontier, generative models like TimeGAN [324] provide valuable insights into how market temporal characteristics affect predictive performance. Their analysis suggests that cryptocurrency markets, due to their continuous 24/7 nature, possess stronger discriminative and predictive power compared to traditional time-constrained stock markets. Incorporating additional features such as yield can further enhance the predictive power of stocks, indicating the potential for hybridizing generative and other DL approaches with feature engineering.

Novel Hybrid Approaches Several recent studies have introduced innovative approaches to cryptocurrency forecasting. For example, Ahmad et al. [7] propose a CNN-LSTM-Transformer model that leverages self-attention alongside convolutional and recurrent layers, achieving superior performance on extensive minute-level historical data of various cryptoassets dating back to the 1970s, as well as on international stock indices, albeit with increased computational demands. Similarly, Jin and Li [187] present the VMD-AGRU-RESVMD-LSTM hybrid model, which uses VMD and EMD for multi-stage decomposition, attention-enhanced GRUs for subsequences prediction, and LSTM for integration, outperforming both standalone and other hybrid models in Bitcoin and Ethereum forecasting.

Efforts to model more specialized phenomena include Kallurkar and Chandavarkar [196], who focus on Ethereum transaction fee prediction post-EIP-1559², using a CNN-LSTM model to capture local and temporal features, thereby addressing a notable gap in cryptocurrency transaction forecasting. The Multiple-Input

²Ethereum Improvement Proposal 1559 (EIP-1559), part of the Ethereum London upgrade, restructured the fee mechanism by introducing a base fee and an optional tip, aiming to reduce congestion and improve transaction confirmation times.

Convolution DL (MICDL) model by Livieris et al. [238] further extends hybridization by processing multiple cryptocurrencies independently through convolutional and recurrent layers, improving both regression and classification tasks.

Other studies focus on the integration of alternative data sources to enrich forecasting models. In particular, the use of sentiment data in cryptocurrency prediction, as shown by Patel et al. [292] and Jung et al. [194], highlights the increasing importance of information beyond traditional price series. Both studies leverage BiLSTM-based architectures to effectively integrate heterogeneous information, such as social media sentiment from platforms like Twitter, technical indicators, on-chain transaction metrics, and broader macroeconomic variables. Patel et al. [292] focus on combining hourly Bitcoin price data with sentiment extracted from social media, showing that models enriched with sentiment signals significantly outperform various LSTM variants and price-only models. Similarly, Jung et al. [194] extend this approach by incorporating a wider array of data categories, ranging from news and social media sentiment scores generated by VADER and FinBERT, Google Trends insights, to on-chain and financial asset data, evaluating multiple algorithms and demonstrating that BiLSTM models particularly excel in capturing short-term price movements with a three-day window. Their findings demonstrate that the fusion of sentiment, technical, and on-chain data outperforms price-only models. A key strength of their study is the extensive dataset, which spans over 2,000 days and encompasses more than five distinct categories of information, offering a wider and more detailed forecasting framework. These advances reflect a broader trend in the field toward hybrid modeling strategies that incorporate diverse and complementary data sources.

Summary

Overall, the literature reveals no universal best model for cryptocurrency forecasting. Model performance varies significantly based on the target cryptocurrency, data granularity, and feature selection. GRU and LSTM are consistently strong contenders. However, simpler models like MLP and FB-Prophet can outperform deep architectures in certain contexts, suggesting that model selection should be guided by empirical validation rather than architectural complexity. Furthermore, hybrid and ensemble methods represent a promising direction, with novel studies incorporating attention mechanisms, time series decomposition techniques, and sentiment analysis to enhance predictive accuracy. Notably, the majority of forecasting analyses concentrate on the post-2015 period, reflecting the market's transition into a more mature and data-rich phase, and Bitcoin is the most extensively studied asset.

Cross-study Observations Across the reviewed studies ($n = 23$), standard LSTM and ML models are most frequently employed ($\approx 96\%$ combined) in terms of model

appearances per study; technical and sentiment analysis are incorporated in 3 studies ($\approx 13\%$). Price-based features like OHLCV are universally utilized, though more specialized inputs such as on-chain data and hash rate are less common.

TABLE 6: Cross-Study Overview of Cryptocurrency Forecasting: Models, Assets, Features, and Timeframes

Category	Top Entities / Statistics
Most Common Models	Standalone: LSTM (12), ML (10), GRU (6); Hybrids: CNN-LSTM-based (4). Total applications: 53 ¹² .
Standalone vs. Hybrid Use	Standalone: $\sim 178\%$; Hybrid/ensemble models: $\sim 48\%$; Novel architectures: $\sim 26\%$. ³
Frequent Feature Types	Price-based features (e.g., OHLCV, Close, Adjusted Close) are used in all studies ($\approx 113\%$), with some relying exclusively on OHLCV and others combining it with Adjusted Close. Less common features include on-chain data metrics, hash rate ³ , and volatility.
Most Forecasted Crypto Assets	Bitcoin dominates ($\sim 78\%$, 18 studies), followed by Ethereum ($\sim 39\%$, 9) and Ripple ($\sim 17\%$, 4).
Time Periods Covered	Nearly half of the studies ($\sim 48\%$, 11 studies) use short-term datasets under five years, while only a few ($\sim 13\%$, 3) span ten years or more, with the longest covering 11 years. Most focus on the post-2015 period, reflecting the maturing phase of cryptocurrency markets.
Performance Metrics Used	The most commonly reported evaluation metrics include RMSE ($\sim 83\%$, 19 studies), MAE ($\sim 57\%$, 13), MSE ($\sim 39\%$, 9), and MAPE ($\sim 30\%$, 7). R^2 appears in $\sim 35\%$ (8 studies). Less common measures include classification-oriented metrics, and statistical or information-theoretic metrics such as Kullback–Leibler divergence and t-SNE.

¹ Some studies use multiple models (e.g., baselines, ensembles), contributing to total model count $>$ number of papers (23). This reflects broader comparative experimentation.

² CNN-LSTM-based models include CNN-LSTM, and CNN-LSTM-Transformer

³ Percentages represent the ratio of total model appearances (standalone, hybrid, novel) to the number of studies (23), reflecting how frequently models appear per study on average. For example, 178% means standalone models appeared on average 1.78 times per study, implying that most studies employ nearly two standalone models rather than just one. Note that individual studies may include multiple standalone and/or hybrid models.

2.5.7 Volatility Forecasting

Volatility forecasting plays a central role in risk management and asset pricing. It enables investors to better anticipate market fluctuations, optimize portfolios, and make informed decisions under uncertainty.

Standalone LSTM for Market Volatility Recent studies in volatility forecasting show the advantages of DL models, especially LSTM, over traditional methods like GARCH. Wang et al. [396] examine the predictive power of internal factors (e.g., lagged volatility, moving averages) and external factors (e.g., technology, financial, and policy uncertainty data) using LSTM and RF models, finding them superior to traditional volatility models like GARCH. The study highlights that GA- and ABC-optimized LSTMs, particularly ABC-LSTM, offer the highest accuracy for weekly volatility predictions. Additionally, a universal model trained on multiple cryptocurrencies outperforms cryptocurrency-specific models, reflecting the interdependent nature of the market. Similarly, Al-Khasawneh et al. [12] find LSTM models to be more effective than traditional approaches such as GARCH and TARCH in capturing the complex, nonlinear patterns characteristic of stock index volatility on the Pakistan Stock Exchange (2009–2021). Their use of multivariate BiLSTM architectures yields the highest forecasting accuracy, underlining the benefit of exploiting temporal and cross-variable dependencies. Extending the application to broader equity markets, Petrozziello et al. [297] explore both univariate and multivariate LSTM models for volatility forecasting using datasets from the DJI (2002–2008) and NASDAQ 100 (2012–2017). Their contribution lies in constructing a nonparametric, market-wide volatility model that leverages LSTM's strength in capturing cross-asset temporal dependencies. Their results confirm LSTM's superiority over traditional models such as R-GARCH and GJR-MEM, particularly under high-volatility regimes, as well as over RNN counterparts.

Hybrid Architectures for Stock Volatility Hybrid approaches to forecasting stock volatility show promising advancements by effectively combining the strengths of diverse models. For instance, Liu et al. [235] introduce the Ordinal Regression GAN for Financial Volatility Trends (ORGAN-FVT), which uses ConvLSTM as a generator to learn the data distribution and an MLP as a discriminator to classify volatility trends in stock data. Enhanced with ordinal regression, ORGAN-FVT outperforms LSTM, GRU, CNN, and ConvLSTM across multiple stock datasets using both stock prices and technical indicators. This highlights the added value of combining generative modeling with trend-aware classification for volatility forecasting.

Several studies propose novel approaches. For example, Akter et al. [11] focus on frontier markets, which often lack clear volatility parameters for risk assessment. They develop a stacking ensemble that combines CNN, LSTM, and BiLSTM base learners with a linear regression meta-learner. This model outperforms traditional

methods like Random Forest, AdaBoost, and Gradient Boosting in forecasting market risk in Bangladesh. Their results suggest that ensemble hybridization is an effective strategy to overcome data limitations and handle the complexities of volatility in emerging economies. Expanding on these efforts, Araya et al. [25] propose an ambitious multi-component hybrid model (SARIMA-GARCH-CNN-BiLSTM) that elegantly unites classical time series decomposition, volatility modeling, and deep feature extraction. While conceptually compelling, the resulting complexity poses valid concerns regarding scalability and applicability in real-time trading environments, where computational efficiency is critical. Similarly, Jin and Zhang [188]’s integration of Wavelet Transform, CNN, and Transformer achieves impressive results in financial risk prediction, particularly benefiting from multi-scale feature extraction and attention mechanisms. However, the sensitivity to extreme outliers and increased computational complexity underscore a key challenge: balancing predictive power with robustness and operational feasibility.

Hybrid Architectures for Index Volatility In the domain of index volatility, Di-Giorgi et al. [90] integrate GARCH with RNN variants (LSTM, BiLSTM, GRU) to capture heteroscedasticity and temporal dependencies. Empirical evaluations across global indices (IPSA, S&P 500, ASX200) confirm that BiLSTM-enhanced hybrids yield the most accurate short-term returns volatility forecasts. These results support the view that GARCH-DL integration provides a flexible and effective approach to modeling volatility structures. Complementing this, Jung and Choi [193] present a novel autoencoder-LSTM hybrid for forecasting FXVIX indices, specifically EUVIX, BPVIX, and JYVIX (2010-2019). Subperiod analysis around global economic events such as the subprime mortgage crisis and Brexit, shows that incorporating wider data distributions and outliers improves forecasting accuracy. The model’s ability to capture data variability and singularities makes it particularly well-suited for the highly reactive nature of currency volatility.

Hybrid Architectures to Commodity Volatility Lastly, Vidal and Kristjanpoller [388] explore volatility forecasting in commodity markets, proposing a CNN-LSTM model that transforms price time series into image-like representations for deep feature extraction. Applied to five decades of gold price data, the model outperforms traditional and DL baselines (GARCH, SVR, LSTM, CNN), with Model Confidence Set (MCS) analysis confirming its superior predictive performance. This highlights the potential of hybrid architectures in capturing the multifaceted nature of commodity volatility directly shaped by economic, geopolitical, and market-driven factors.

Recent advances in volatility forecasting highlight the effectiveness of LSTM both as a standalone model and within hybrid frameworks. Approaches that integrate GARCH with RNNs or combine wavelet decomposition with Transformers push the boundaries by leveraging complementary model strengths. From generative models like ORGAN-FVT to modular ensembles for frontier markets, the evo-

lution reflects not just a race for accuracy, but a deeper attempt to model financial complexity in a more interpretable and robust way. That said, a recurring challenge emerges: the trade-off between model sophistication and practical deployability.

2.5.8 Other Works

Several recent studies extend the application of DL techniques in financial forecasting from different perspectives.

Addressing key limitations of DL models like their black-box nature, tendency to overfit, sensitivity to parameter choices and noise, slow convergence, and high computational demands, Zhang et al. [437] propose obtaining prior knowledge of the correlation between historical and future stock prices. They introduce two feature extraction modes: the curve-shape-feature (CSF), in which future prices can be inferred from the curve shapes of historical price patterns, and the non-curve-shape-feature (NCSF), where such information is not encoded in price curve patterns. Their findings reveal that while a wide range of models, including SVM, RF, MLP, Bayesian classifiers, decision trees, CNN, embedded CNN, BiLSTM, and embedded BiLSTM, perform well under the CSF mode, they struggle under the NCSF mode. Importantly, real-world stock data suggests that CSF-mode contains limited predictive information, whereas NCSF-mode may offer stronger signals. This distinction suggests that much of the apparent success of existing models may be tied to their performance on datasets that do not reflect real market complexity, and highlights the need for novel modeling approaches capable of effectively capturing NCSF-mode for improved financial forecasting.

In the context of portfolio management, Ma et al. [249] integrate a variety of ML and DL forecasting models, such as RF, SVR, LSTM, and CNN, and the statistical ARIMA with two portfolio optimization techniques: Mean-Variance Framework (MVF) and Omega Function (OF). Their results, based on CSI 100 data from 2007–2015, indicate that RF delivers the best predictive performance among all methods. When integrated with MVE, RF yields the most profitable portfolios even after accounting for transaction costs, although high turnover reduces profitability significantly. Interestingly, SVR+OF surpasses RF+OF when costs are included. Notably, this is the first study to combine ML and DL methods with MV and Omega portfolio optimization. The study highlights the critical importance of incorporating practical constraints, such as turnover and costs, which are often overlooked in purely predictive exercises. It also reinforces the robustness of RF in financial applications.

Addressing portfolio optimization complexities and return forecasting challenges, Wang et al. [398] propose a hybrid model that combines LSTM networks for stock returns' temporal feature extraction, with a stochastic dominance filter for input dimensionality reduction. The model is designed to construct an index-tracking portfolio that replicates the performance of a specific market index by mirroring its

composition and returns, instead of actively selecting individual stocks. By extracting dominant returns from 100 stocks to track the FTSE 100 index return, experiments demonstrate that the model improves prediction accuracy and reduces estimation errors. The integrative use of stochastic dominance for feature selection is particularly noteworthy, as it aligns economic theory with ML practice. While the results are promising, key concerns remain regarding the model's scalability, robustness across different indices, and practical implementation, particularly in terms of transaction costs. A comparison with conventional index-tracking methods would also help contextualize its relative performance and interpretability.

Due to its use of specialized, non-standard datasets, the final study included is the one by Zhao et al. [439], who propose an advanced hybrid DL model for stock market prediction and investment decision-making, integrating CNN, BiLSTM, and an attention mechanism. The model utilizes CNN for feature extraction from historical stock prices and trading volumes, BiLSTM to capture temporal dependencies, and the attention layer to prioritize relevant features. Evaluation on datasets enriched with sentiment and market indicators (e.g., CAMBRIA, KRIRAN, SHARMA, and JAMES) shows that the CNN-BiLSTM-attention model outperforms state-of-the-art methods across multiple metrics. This work highlights the value of hybrid and attention-based architectures in complex, noisy environments like financial markets.

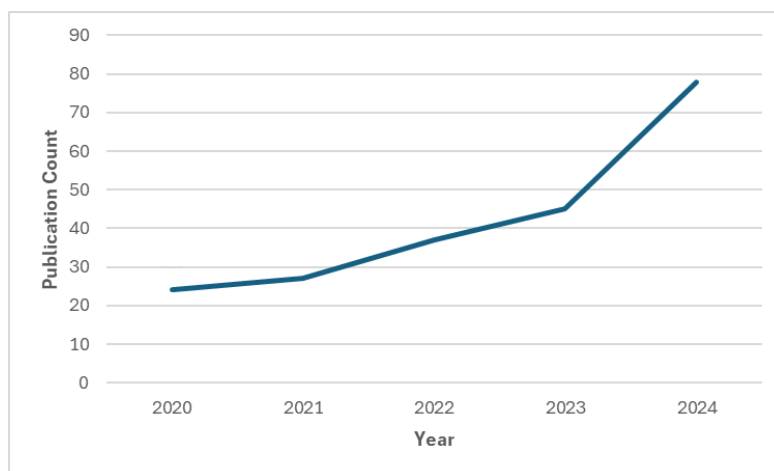


FIGURE 2.5.1: Annual distribution of publication count (2020–2024)

2.6 Current snapshot of DL research for financial applications

After reviewing research articles specifically focused on financial forecasting with DL models, we present an overview of the current state of the field. A total of 187 papers were analyzed, and categorized based on the forecasted asset type. Addi-

tionally, we analyzed these studies in terms of their chosen DL models, development environment, datasets, and other distinguishing factors such as feature sets and performance evaluation metrics. Given that some articles fall under multiple forecasted asset types and employ various model types, the findings provide a general overview of the prevailing trends rather than a one-to-one mapping. For instance, the work of DURAIRAJ and MOHAN [94] appears in three categories: index, forex, and commodity forecasting. A detailed breakdown of the reviewed studies is presented in Appendix .1, where the literature is systematically organized by asset class, including stocks, indices, foreign exchange, commodities, bonds, cryptocurrencies, and volatility, highlighting key characteristics such as datasets, time periods, feature sets, model architectures, evaluation metrics, and implementation environments.

The literature suggests that no single model is universally best, with optimal performance depending on task specificity, asset volatility, data characteristics, and the incorporation of domain knowledge. The following table provides key insights into DL model performance across multiple financial domains, highlighting research volume, common model types, and key observations within each domain.

Domain	# Studies	Common Hybrids and Notes
Stock	75	Standalone LSTM models dominate. Price-based features are most common; sentiment and technical indicators moderately used; datasets mostly long-term.
Index	83	LSTM-based hybrids dominate. Mostly price features; cross-market studies focus on US and Chinese indices; mixed standalone and hybrid use; includes chaos-based modeling; mostly long-term data.
Forex	11	Standalone LSTM or LSTM-based hybrids. Major currency pairs centered on USD; mix of short- and long-term periods; includes chaos-based modeling.
Commodities	8	LSTM hybrids. Underexplored domain; hybrids improve accuracy and efficiency; emerging reinforcement learning and interpretability methods; includes chaos-based modeling.
Bond	2	LASSO-PCA-LSTM hybrids. Very limited research; feature selection key; considered a stable asset class.
Cryptocurrency	23	LSTM models. Shorter timeframes common, especially post-2015; Bitcoin and Ethereum dominate; unique features like on-chain data.
Volatility	10	Standalone and hybrid LSTM models, outperforming traditional models (e.g., GARCH).

TABLE 7: Summary of Deep Learning Model Performance Across Financial Forecasting Domains

We now present a summary of our key findings, emphasizing important insights for researchers in the field. Figure (2.5.1) shows the annual distribution of published

2.6. Current snapshot of DL research for financial applications

articles from 2020 to 2024, while Figure (2.6.1) illustrates the allocation of research across different financial forecasting domains. The analysis reveals that index forecasting is the most extensively studied area, followed closely by stock price forecasting, with 83 and 75 studies, respectively. Together, these two domains represent over 70% of all reviewed research, underscoring their prominence in the literature. Additionally, cryptocurrency price prediction attracts considerable attention with 23 studies, while forex forecasting accounts for 11 studies, and market volatility forecasting for 10. Commodity forecasting is less explored, with 8 studies, and bond market forecasting remains the least studied, with only two publications to date.

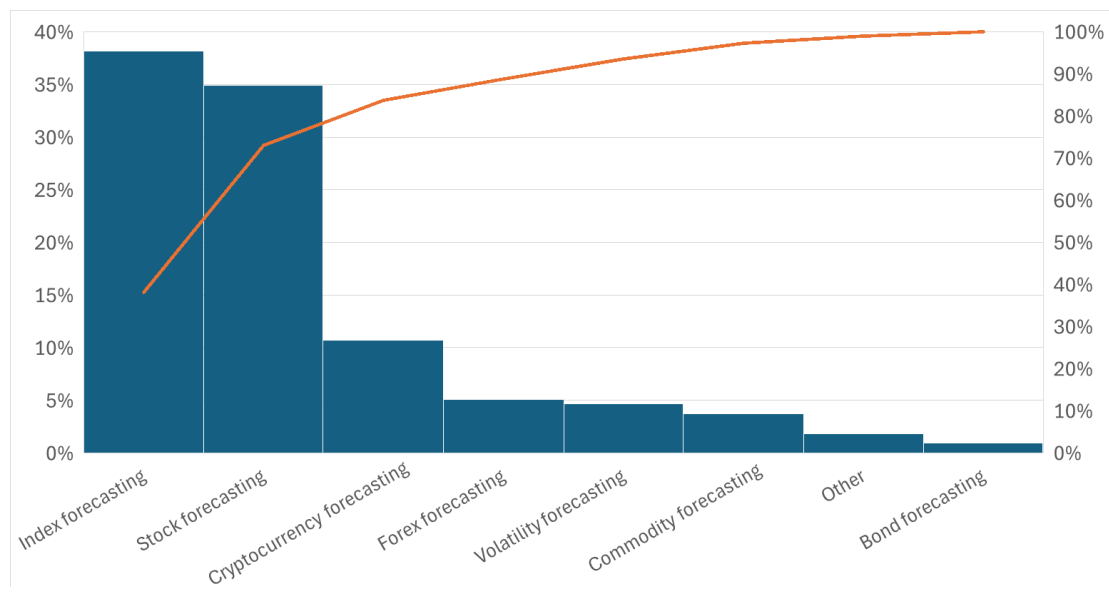


FIGURE 2.6.1: Distribution of studies by financial forecasting task (2020–2024)

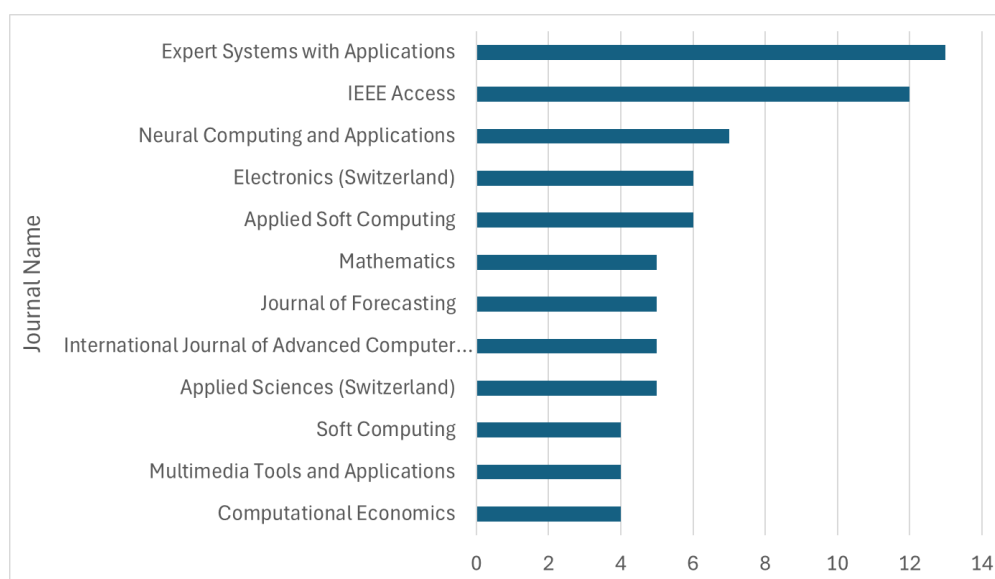


FIGURE 2.6.2: Top journals by article publication count for the period 2020–2024

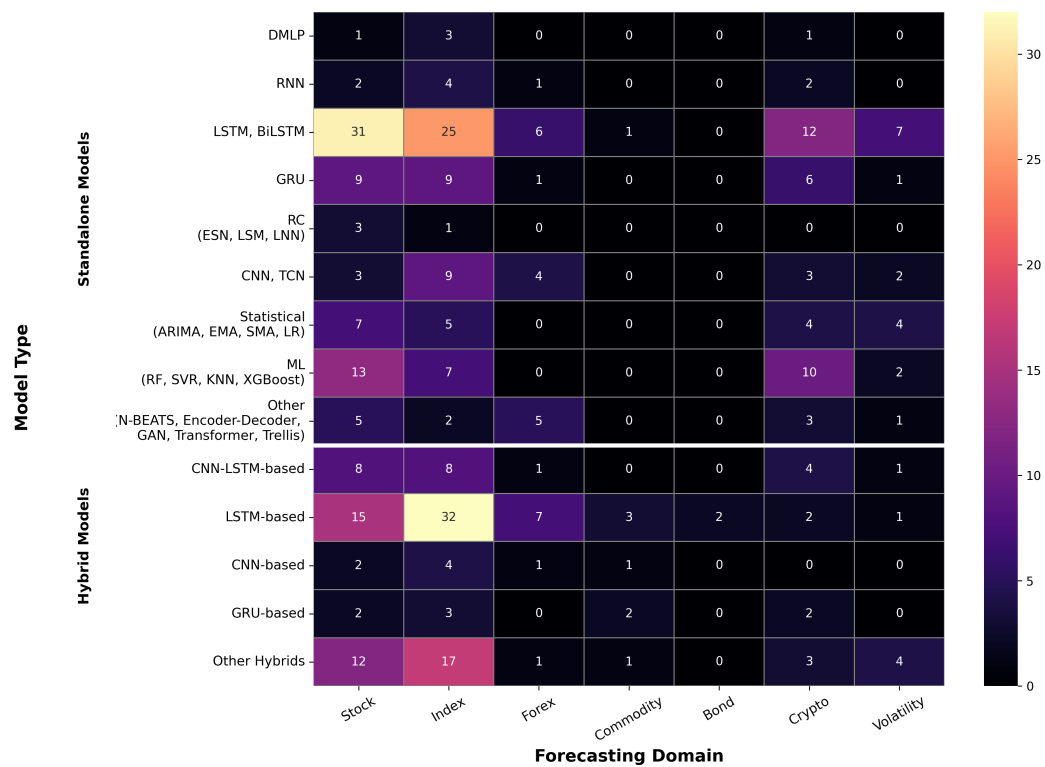


FIGURE 2.6.3: Topic-Model heatmap

A key consideration for researchers is where they can publish their findings. As part of this review, we analyzed the distribution of publications across academic journals. The results, presented in Fig. (2.6.2), highlight the leading journals for financial time series forecasting. The journals with the most published articles include Expert Systems with Applications, IEEE Access, Neural Computing and Applications, Electronics (Switzerland), Applied Soft Computing, Mathematics, Journal of Forecasting, International Journal of Advanced Computer Science and Applications, Applied Sciences (Switzerland), Soft Computing, Multimedia Tools and Applications, and Computational Economics.

Many researchers employ data-processing techniques in conjunction with DL models for enhanced forecasting performance. Time series decomposition is widely used to extract underlying patterns by separating components of varying frequencies [10, 10, 17, 106, 151, 227, 230, 301, 310, 350, 401, 418, 420]. Similarly, dimensionality reduction techniques, such as PCA, help preserve data variation while reducing complexity [91, 258, 325, 393]. Given the sensitivity of DL models to parameter configurations, various optimization methods are employed, including Grid Search [3, 188, 199, 318, 324], PSO [211, 392], Adaptive PSO [210], ABC [212], FPA [211], ARO [149, 319], Bayesian Optimization [184, 299, 427], GA [438], and Random Search [372].

Over the past five years, while standalone models remain widely used, often serving as benchmarks, the landscape of DL-based financial forecasting has in-

2.6. Current snapshot of DL research for financial applications

creasingly shifted towards adaptive hybrid architectures that integrate temporal, spatial, and attention mechanisms into unified frameworks. For a more detailed and interactive exploration of the DL models employed in financial forecasting, readers are encouraged to visit the online resource available [here](#). This interactive chart provides dynamic visualizations of model adoption trends within the field.

As illustrated, standalone models outnumber hybrid approaches, accounting for 61% (215 implementations) compared to 39% (139 implementations) for hybrids. The outer circle highlights the dominance of LSTM-based models in both categories, representing 38% of standalone and 45% of hybrid implementations. Notably, the more specialized CNN-LSTM hybrids comprise 16% of hybrid models. GRU-based architectures are also prevalent among standalone models (12%) but account for approximately 6% within hybrid configurations.

Our findings align with prior review papers [335], which documented the dominance of RNN-based models in financial prediction from 2005 to 2019. Specifically, in our analysis, RNN variants (including Vanilla RNN, LSTM, BiLSTM, GRU, and RC) constitute 56.28% (121 implementations) of standalone models and 51.08% (71 implementations) of hybrid models (primarily LSTM- and GRU-based). They constitute a significant portion of model usage, reinforcing the field's continued reliance on temporal modeling techniques. It is this nuanced interplay among model types, rather than raw counts, that drives ongoing innovation, a theme explored further in Section 2.7.

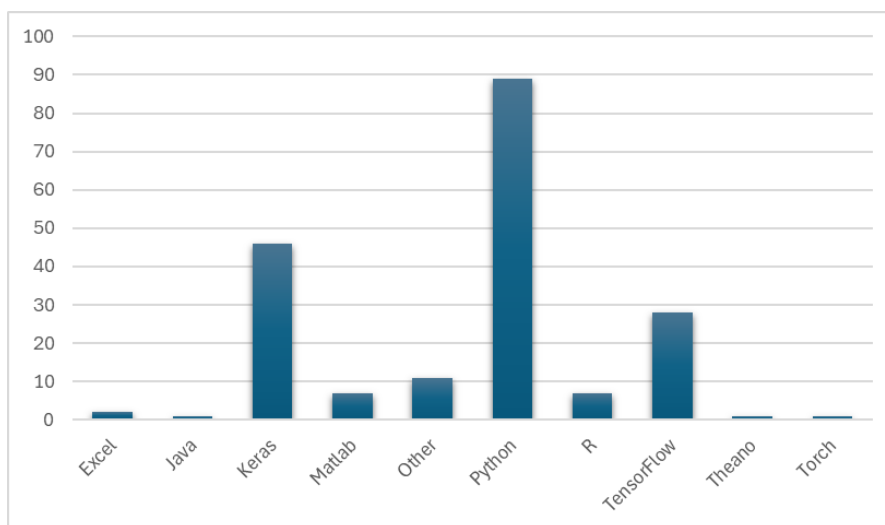


FIGURE 2.6.4: The preferred development environments. The Python category includes PyCharm Edu, Statsmodels, Tushare, Swarm-PackagePy, yfinance, investpy, seaborn, Matplotlib, Numpy, Pandas, sklearn, and PyTorch libraries. The Other category includes PostgreSQL15, Google Colab, Apache Hadoop, NVIDIA DIGITS, PyQt5, fbprophet, Math, and Gretl.

Fig. (2.6.3) illustrates model adoption trends across forecasting domains. LSTM and BiLSTM dominate across stock (31), index (25), and cryptocurrency (12) forecasting. GRU models are also preferred, while traditional statistical methods (e.g., ARIMA, SMA) and classical ML models (e.g., RF, SVR) remain common bench-

marks. Among hybrids, LSTM-based architectures lead, particularly in index (32), and stock (15) forecasting, followed by CNN-LSTM-based hybrids (8 in the same domains). This pattern highlights the continued reliance on LSTM-family models for sequential data modeling. GRU-based models provide competitive performance with fewer parameters. The rise of CNN components in hybrid models reflects a growing interest in capturing local patterns and multi-scale features, especially where temporal signals alone may not suffice. Diverse and custom hybrid designs, e.g., GAN-TrellisNet, CEEMDAN-TCN-GRU-CBAM, highlight the field's preference toward customized, task-specific solutions rather than a single, universally superior model.

Notably, RC methodologies remain relatively underexplored, with only four reported implementations mainly in stock and index forecasting. Despite their limited adoption, RC models exhibit strong predictive performance through simple design, computational efficiency, and fast training, underscoring their potential as an effective tool for financial time series forecasting.

Lastly, our analysis of preferred frameworks and development environments, as shown in Fig. (2.6.4), underscores an implicit methodological homogeneity: the vast majority of studies employ Python-based stacks. The *Python* category includes tools such as PyCharm Edu, Statsmodels, Tushare, Swarm-PackagePy, yfinance, investpy, seaborn, Matplotlib, Numpy, Pandas, scikit-learn, and PyTorch. The *Other* category includes PostgreSQL15, Google Colab, Apache Hadoop, NVIDIA DIGITS, PyQt5, fbprophet, Math, and Gretl. Across the reviewed studies, Python-based tools dominate the implementations, followed by Keras, and TensorFlow, highlighting their central role in the articles surveyed. However, it's important to acknowledge that some papers do not specify their development environment, which limits the depth of our analysis in this area.

2.7 Discussion and Open Issues

Despite remarkable progress, DL models in financial forecasting still face several limitations in robustness, interpretability, and adaptability to market complexity. As the field matures, a forward-looking research agenda is essential to guide future innovation. In this section, we synthesize key open issues and propose concrete directions for advancing DL research in finance. While certain application-specific opportunities are acknowledged, this review primarily adopts a methodological perspective, emphasizing the design, evaluation, and deployment of DL models in financial forecasting.

2.7.1 Dominance and Underutilization of RNN Variants

RNN-based models, especially LSTM and its variants, dominate financial forecasting due to their strength in modeling temporal dependencies. Almost half of the studies adopt LSTM, at least for benchmarking. RC models, being part of the RNN

family, remain vastly underexplored despite their computational efficiency and promising predictive performance. This gap suggests a missed opportunity for simpler, faster models in financial applications.

2.7.2 Feature Selection and Multimodal Data Integration

While raw time series data forms the backbone of feature sets, complementary data, such as fundamental indicators [116, 299], technical factors [69, 198, 260, 269, 422, 438], and increasingly, sentiment analysis [74, 89, 110, 130, 194, 212, 237, 292, 300, 348, 355], play vital roles. The rise of financial text mining, leveraging advanced NLP techniques, reflects a growing appreciation for incorporating investor and public market sentiment extracted from news, social media, tweets, forums, and blogs. However, the integration of these heterogeneous data types remains inconsistent and often lacks systematic evaluation of their incremental predictive value.

2.7.3 Asset Focus and Emerging Domains

Indices and stock prices dominate research focus due to their liquidity and rich historical data, providing stable settings for model evaluation. Meanwhile, cryptocurrency forecasting is emerging rapidly due to its high volatility and lucrative short-term returns. For instance, Bitcoin's price surged from about \$12 in late 2012 to nearly \$1,100 by November 2013, reaching an all-time high above \$67,000 in November 2021 [209]. This potential for significant gains drives increased academic and practical interest in cryptocurrencies. Additionally, the continuous, 24/7 nature of cryptocurrency markets, unlike the time-constrained trading hours of traditional stock exchanges, provides richer, more granular data streams, offering greater discriminative and predictive power. This characteristic further reinforces the importance of focused research on these emerging markets [324]. Nevertheless, the field must balance enthusiasm for crypto with rigorous validation to avoid hype-driven pitfalls.

2.7.4 Applications Beyond Forecasting

DL's application extends into algorithmic trading, portfolio management, and robo-advisory systems. In algorithmic trading, deep reinforcement learning enables more effective buy, sell, and hold strategies by adapting to dynamic market conditions [23, 218, 371]. The integration of financial forecasting with portfolio management, asset allocation, and risk assessment is further evident in robo-advisors, automated financial advisors that design and manage investment portfolios based on a client's financial situation, investment goals, and risk profile [60]. Further the increasing availability of high-frequency data, along with advancements in financial sentiment analysis through text mining, enhances decision-making by incorporating investor psychology and market sentiment. Examples of recent advancements in NLP include DistilBERT [322], a faster and more efficient variant of BERT, and

BigBird [431], a sparse attention mechanism for handling long sequences. These developments drive the ongoing automation of trading and investment strategies, positioning this branch of quantitative finance as a key area of future growth.

2.7.5 Handling Market Shocks and Global Crises

Several studies have examined the impact of extreme global events, such as the COVID-19 pandemic, on financial markets [20, 213, 278, 404]. The heightened market volatility during this period operated as a real-world stress test scenario. This global crisis can be seen as a catalyst for the development of more sophisticated models. The COVID-19 pandemic revealed the limitations of existing models, especially those that relied on historical data and stable market conditions, highlighting the need for more adaptive and resilient approaches that integrate real-time data, financial sentiment, and cross-market signals to better capture and respond to abrupt market shifts.

In response to these challenges, chaos theory-based hybrid models, such as Chaos-LSTM-PR [94, 97] and Chaos-CNN-PR [95], show promise by effectively capturing the nonlinear and chaotic dynamics of financial time series. While their structural complexity requires further scrutiny, these models have shown notable improvements in predictive accuracy across diverse asset classes, including index, forex, and commodities forecasting, highlighting the promise of chaos-augmented approaches as a fruitful research direction.

2.7.6 Quantum Computing: A Futuristic Promise

Quantum computing holds significant theoretical promise to revolutionize DL for financial forecasting by improving the representational power and computational efficiency. It leverages qubits, units of information that can exist in states ranging from zero to one. Unlike classical computing, where units are binary, qubits enable simultaneous exploration of multiple parallel paths in a single calculation unit [305], through superposition and entanglement. This could substantially improve the speed and accuracy of financial predictions. Quantum algorithms also provide a more comprehensive framework for DL than classical computing, aiding in the optimization of the objective function [405], which is critical for complex financial modeling and decision-making. However, current quantum hardware falls short due to the limited computational capacity and high error rates [416]. Despite these challenges, the potential for collaborative research at the intersection of DL and quantum computing is immense [132], and active efforts are needed to translate this potential into finance-ready solutions.

2.7.7 Interpretability and Explainability Challenges

The black-box nature of DL models remains a significant challenge in terms of interpretability and explainability, hindering users' ability to comprehend and trust the model's predictions. Interpretable and explainable ML techniques arise from the need to design *intelligible* ML systems, i.e. ones that a human mind can comprehend, and to understand and explain the predictions made by opaque models, such as deep neural networks [261]. While there is no universal agreement within the ML community or mathematical rigor on the exact definitions of interpretability and explainability, these are often closely tied and researchers use them interchangeably [233, 261]. However, these are distinct terms. We loosely define interpretability as the capacity to understand a model's internal mechanisms and diagnose its behavior, while explainability requires that models provide explicit reasoning for their decisions and outputs. A more formal definition is given by [137], wherein interpretability refers to the ability to understand what a model did, but on its own it is insufficient for trust in black-box systems. Explainability requires that models offer explicit justifications for their behavior, thereby fostering user trust and clarifying the underlying reasons for their decisions. While explainable models are inherently interpretable, the reverse is not always true. An explanation is assessed by its interpretability, the ability to describe the system's inner workings in a user-friendly language, and its completeness, the ability to describe the operation of a system in an accurate way [137]. Ultimately, the success of interpretability hinges on delivering descriptions that are both simple and meaningful to the user, taking into account the user's cognitive framework, knowledge, and biases [137]. To this end, explainable artificial intelligence (XAI) seeks to enhance the transparency, comprehension, and trustworthiness of complex DL models. Recent efforts have focused in explanation of deep network processing, representation, and system-level explanation generation, yielding promising results [137]. Interpretability and explainability remain active research areas, particularly given the integration of diverse DL models and their increasing deployment in real-world applications such as financial advisory, healthcare, and autonomous vehicles. We remain optimistic that continued interdisciplinary collaboration will facilitate the "opening" of black-box models, enabling deeper understanding, enhanced transparency, and ultimately, greater user trust.

2.7.8 Lightweight Models for Real-World Deployment

Deploying DL models on smart devices with limited computational resources necessitates lightweight architectures, as traditional models demand substantial processing power, hindering real-time financial predictions on mobile platforms or edge devices. To address this, techniques such as network pruning, model compression, quantization, and knowledge distillation are being utilized to develop efficient models. Network pruning reduces model complexity by eliminating neurons with minimal contributions, while quantization reduces the precision of model weights to save memory. Knowledge distillation involves training a smaller model (student)

to replicate the behavior of a larger, more complex model (teacher), thereby maintaining accuracy with fewer resources. Additionally, feature selection helps identify the most relevant input features, further streamlining the model. Collectively, these strategies enable simplified, scalable architectures that maintain high predictive accuracy even when deployed on mobile platforms and edge devices.

2.7.9 Need for Standards in Evaluation

A persistent challenge in financial DL research is the lack of standardized evaluation protocols, as emphasized by Rundo et al. [316], Thakkar and Chaudhari [368], and Chen et al. [71]. Differences in dataset quality, preprocessing, and validation strategies, such as using clean versus noisy data or applying random splits that inadvertently introduce look-ahead bias, undermine the comparability of results and increase the risk of overfitting. Additionally, publication bias exists in the field, whereby studies reporting positive results are more likely to be published, while negative or null findings are often omitted.

Although benchmark datasets and testing protocols are not fully realistic, often carrying biases such as survivorship or reliance on revised data, they nonetheless are useful for enabling relative comparisons across studies. Standardized benchmarks establish a common testing ground, ensure consistent market histories and train/test procedures, and help distinguish genuine predictive ability from flaws/shortcuts in the experimental design. For future research, we recommend a balanced approach that combines exploratory research on varied datasets and settings, with the use of standardized benchmarks, thereby ensuring both innovation and comparability.

2.7.10 A Design-Principles Taxonomy

Moving beyond mere model counts and acronyms, we present a taxonomy of financial forecasting architectures based on six design principles:

1. **Sequential-memory frameworks:** Pure recurrent architectures (LSTM, GRU) remain the baseline for smooth, univariate data modeling.
2. **Spatial-temporal hybrids:** CNN–LSTM/GRU and graph-augmented neural networks effectively capture spatial (cross-asset) and temporal (over-time) dependencies, useful for portfolio and market-wide analyses.
3. **Attention-augmented ensembles:** Transformers and attention-enhanced RNNs dynamically adapt to shifting market regimes, albeit at the expense of heavier compute.
4. **Chaos and quantum-inspired hybrid models:** These capture deep nonlinearities and computational parallelism but add complexity.

5. **Preprocessing-augmented pipelines:** Decomposition techniques like EMD, VMD, and wavelets reduce noise and isolate intrinsic modes.
6. **Diverse asset class forecasting:** While stocks and indices remain dominant, crypto forecasting is rising rapidly, offering unique modeling challenges and opportunities.
7. **Multimodal feature integration:** Combining technical, fundamental, and sentiment signals, especially from financial text mining, improves robustness but remains inconsistently applied across studies.
8. **Application-oriented modeling:** DL models increasingly support real-world use cases, including algorithmic trading engines, robo-advisory services, and portfolio optimization tools.

This taxonomy shifts the focus from brand-name architectures toward a design-principles perspective, enabling clearer identification of fertile intersections (e.g., attention-driven, chaos-filtered pipelines) for the next generation of research.

2.7.11 Critical Methodological Gaps

Two key shortcomings persist across the literature:

- **Lack of robustness evaluation:** Few models are tested under adverse market conditions or subjected to stress scenarios, raising questions about model resilience in real-world conditions where regime shifts, and extreme events are common. This issue is further compounded by the lack of standardized evaluation protocols, which continues to impede robust model assessment and comparability across studies.
- **Superficial explainability without domain validation:** While XAI methods such as SHapley Additive exPlanations (SHAP) [245], Local Interpretable Model-agnostic Explanations (LIME) [311], and integrated gradients are gaining traction, they are seldom coupled with domain-specific validation practices. Few studies apply XAI primarily to visualize feature importance or to provide superficial interpretability without linking explanations to financial reasoning, expert judgment, or regulatory frameworks. This lack of contextual validation reduces the practical interpretability of the models, especially in high-stakes environments where transparency, accountability, and compliance are essential. Coupling XAI methods with scenario-based stress tests, or regulatory compliance checks could significantly enhance the credibility of AI-driven insights in financial decision-making.

2.7.12 Initial Questions Answered

- Which DL models are preferred (and more successful) in different applications?

Response: While standalone models are more frequently used in applications, hybrid models, when tailored to the characteristics of the dataset, the forecasting task, and the underlying problem, often provide superior performance. Recurrent architectures, particularly LSTM and its variants, remain a dominant and effective choice in financial forecasting, both as standalone models and as core components in hybrid frameworks. CNNs, especially when combined with LSTMs (e.g., CNN-LSTM), are also widely preferred, reflecting their complementary strengths in capturing spatial and temporal dependencies.

- What challenges do DL models face when applied to financial time series data?

Response:

- **Data & market challenges:** DL models face significant challenges when applied to financial time series, primarily due to non-stationarity, volatility, chaotic behavior, and noise. These issues are further exacerbated by regime shifts and structural breaks, which hinder model generalization. Data scarcity and quality, especially in emerging or frontier markets, can impair model training and reliability. DL models also suffer from overfitting, especially when applied to high-dimensional, volatile financial datasets without robust validation strategies. These data quality concerns necessitate advanced preprocessing techniques such as decomposition, denoising, and dimensionality reduction.
- **Model optimization & feature engineering:** Proper hyperparameter tuning and task-specific architectural adjustments are critical. Feature selection (attention mechanisms) is essential for extracting meaningful patterns, while reliance on raw stock data alone often fails to capture the full market picture. Thus, integrating technical, fundamental, and sentiment data is important for richer representation.
- **Computational considerations:** While complex hybrid models can enhance forecasting accuracy, they also introduce significant computational costs. This can limit real-time applicability and deployment on resource-constrained platforms, highlighting the importance of balancing model complexity with efficiency.
- **Robustness & extreme conditions:** DL models often lack robustness under extreme market conditions. Their performance can degrade sharply during sudden regime shifts, market shocks, or crisis scenarios, representing a key challenge for reliable out-of-sample forecasting and operational resilience.
- **Interpretability & explainability:** Despite the increased adoption of explainable AI, many DL models remain largely black boxes, and explainability remains shallow, with limited grounding in financial logic or regulatory context. Addressing this limitation is essential to enhance model transparency and foster user trust.

- What is the future direction for DL research in financial forecasting?

Response: Addressing the methodological gaps and challenges identified earlier, future DL research in financial forecasting will emphasize developing more adaptive, interpretable, and robust models aligned with real-world financial constraints. Key directions include:

- **Real-time and continual learning:** Architectures capable of dynamically updating to handle non-stationarity and regime shifts, improving viability in high-frequency, volatile environments.
- **Robustness and stress testing:** Elevating robustness from an afterthought to a core design principle by incorporating stress-testing protocols simulating extreme market events (crises, flash crashes, geopolitical shocks) to ensure out-of-sample resilience.
- **Finance-specific explainability:** Enhancing explainability by integrating post-hoc methods like SHAP and LIME within finance-specific validation contexts, such as scenario testing, expert feedback, and regulatory compliance, and developing formal metrics to measure transparency, fidelity under stress, and consistency with financial reasoning for end users such as analysts, auditors, and regulators.
- **Modular hybrid models:** Designing task-specific hybrids balancing predictive accuracy and computational efficiency, particularly for real-time applications like algorithmic trading and crypto markets.
- **Behavioral finance integration:** Incorporating behavioral principles (herding, loss aversion, anchoring) and sentiment-driven signals during volatile or irrational phases can enhance models' predictive power by acknowledging the human element inherent in real-world financial decision-making.
- **Multi-modal data fusion:** Expanding DL applications to underexplored asset classes (e.g., crypto, commodities, emerging markets) while integrating diverse data types (technical, textual, macroeconomic, and alternative data) will improve generalizability and robustness across financial contexts.
- **Adaptive architectures:** Employing meta-learning systems that dynamically switch between submodels based on changing market regimes, and reinforcement learning agents that optimize the balance between prediction accuracy and computational efficiency, will further strengthen forecasting capabilities.
- **Causal inference and transparency:** Embedding causal inference and counterfactual reasoning improves generalization, supports “what-if” scenario planning, and enhances transparency, thereby fostering user trust. Integrating symbolic reasoning with DL, known as neuro-symbolic AI, incorporates domain knowledge and financial constraints, enhancing explainability, supporting decisions aligned with financial principles and regulatory requirements, thereby advancing transparent, reliable, and adaptive financial forecasting.

- **Emerging paradigms:** Exploring Liquid Neural Networks, chaos-inspired, and quantum-enhanced models to enhance performance while managing computational complexity.
- **Lightweight models for real-time deployment:** Lightweight DL models enable fast and efficient real-time applications in algorithmic trading, robo-advisors, and the continuously operating cryptocurrency markets, supporting sophisticated decision-making, automation, and personalization on mobile, edge devices, and low-latency platforms.

In short, the future of DL in finance lies not in ever-deeper architectures, but in designing context-aware, interpretable, and adaptive systems that align with the practical constraints and uncertainties of real-world markets.

2.8 Concluding Remarks

The financial community recognizes the growing potential of DL applications for forecasting, as the volume of research in this area continues to rise at an accelerated pace. In this survey, we review studies published between 2020 and 2024, to offer an overview of the current research landscape. We first describe the most prominent DL models in the field and then categorize the studies based on the implementation areas of financial forecasting. Within each category, we organize the studies based on their focus, distinguishing between standalone and hybrid models, novel approaches, sentiment, and technical analysis, and various preprocessing techniques employed, such as dimensionality reduction, decomposition, and optimization.

Despite the long history of financial forecasting, the field continues to evolve with the advent of new DL models, their integration, and the infusion of interdisciplinary methods. This presents ample opportunities for future research. By considering the aforementioned observations and open issues, the next generation of DL models can be developed, paving the way for more efficient, intelligent, and automated financial systems.

Notably, the limited exploration of RC, particularly ESNs, presents an opportunity to revisit recurrent architectures through the lens of optimization and adaptability. This thesis contributes to this emerging direction by proposing a novel self-adaptive hybrid optimization framework that is inspired by Cuckoo Search and Genetic Algorithms to effectively fine-tune ESN hyperparameters. Beyond enhancing predictive performance, this approach aims to enhance model resilience, ultimately demonstrating the untapped potential of ESNs for robust and reliable financial forecasting.

Chapter 3

Methodology

3.1 Echo State Networks: Architecture and Dynamic Behavior

This chapter establishes Echo State Networks (ESNs) as the forecasting model, given their suitability for time series forecasting tasks. It begins with an overview, the mathematical formulation of ESNs and the algorithmic structure, followed by a review of open-source frameworks that facilitate the building, training, and optimization of ESN models. A dedicated section on the hyperparameters of ESNs explores their impact on ESN performance, followed by a section on the fundamental properties of ESNs. The next section delves into the inner chaotic behavior of reservoir dynamics, discussing both its benefits and risks. Topics include attractors, the edge of chaos, estimation of the Largest Lyapunov Exponents (LLE), the separation ratio, controlled chaos, and visual interpretation using Kernel PCA, all of which are fundamental to a comprehensive understanding of the dynamic landscape of ESNs. The following section identifies key limitations in ESN design and optimization, highlighting the need for effective optimization techniques. The chapter concludes with a discussion of existing optimization approaches.

3.1.1 Overview of ESNs

Echo State Network (ESN) [174, 177] and Liquid State Machine (LSM) [252] were independently and simultaneously proposed, and along with further advancements build up the general term Reservoir Computing (RC). RC derives from the realm of RNNs [244], hence the computational framework of RC is designated to deal with sequential data processing, via recurrent connections, allowing information to cycle in the network's architecture, while remembering both recent and more distant input signals.

An ESN consists primarily of a reservoir and a readout layer (see Section 2.3.2). The reservoir is a fixed, randomly initialized network of neurons, which is generally treated as a "black box" [78], and reads and maps input signals into a higher-

dimensional space. This transformation facilitates a complex nonlinear representation of input patterns, improving linear separability of the input data. Then, the readout layer maps the reservoir states to the desired outputs. Only this layer is trained, typically using a simple learning algorithm like linear regression, applied to the reservoir states and target output signals [98, 244]. This represents the main advantage of the ESN system. A visual representation of an ESN architecture is provided in Figure (2.3.7).

This simple, yet effective morphology of the ESN system, reduces complexity and computational cost in the task of training the model, making ESN an attractive alternative to RNNs [36, 332].

The ESN system is also able to overcome the pitfall of gradient-descent inherited by the RNN training. This is because, in contrast to the conventional RNN techniques where all the weights need to be tuned to perform a particular task, in ESN, the training is reduced to a simple linear regression problem [244]. This minimal ESN design enables computational efficiency.

The memory feature is another crucial attribute of the ESN system affecting the learning process and generalization ability. This characteristic emerges from the dynamic interactions of the reservoir's neurons. It allows the ESN system to encode and remember past input signals, thus, enabling the extraction of time dependencies and adaptation to varying input patterns. At the following stage, this memory is utilized by the readout layer to conduct the task at hand [127, 244, 386].

These attributes justify the numerous applications in real-world problems, such as speech recognition [345, 387], audio processing or Music Information Retrieval [163, 359], image processing [131, 180, 181], robotics [24, 191, 321], time-series prediction [63, 174, 178, 412], and natural language processing tasks [59, 327, 329, 341]. In these studies, RC models were found to equal, if not outperform, the state-of-the-art techniques. The design of the reservoir topology has been studied widely and guidelines are offered herein [109, 125, 143, 159, 174, 176, 243, 312, 360, 386]. Another discussion on the appropriate configuration of the parameters and the design of the reservoir using metaheuristics is presented in [34].

The ESN represents the virtual paradigm of the RC model. In RC the reservoir can be either virtual or physical. In the virtual case, the reservoir is modeled as a neural network [332], while in the physical case, the reservoir is realized through natural systems exhibiting inherent non-linearity, such as the ripples on the surface of water. A detailed review of physical RC systems is provided by Tanaka et al. [364]. Our study's focus remains on virtual reservoirs, and while physical reservoirs are acknowledged, they fall beyond the scope of our investigation.

3.1.2 Mathematical Formulation

ESNs are nonlinear, dynamic, deterministic models where the reservoir updates its internal state based on the current input and its prior state. The dynamics can be represented by the state-space system:

$$\begin{cases} x_{t+1} = F(x_t, z_{t+1}), \\ y_{t+1} = g(x_{t+1}) \end{cases} \quad (3.1)$$

for $t \in \mathbb{Z}$, where $F: \mathbb{R}^N \times \mathbb{R}^K \rightarrow \mathbb{R}^N$, $N, K \in \mathbb{N}^+$ represents the reservoir map and $g: \mathbb{R}^N \rightarrow \mathbb{R}^L$, $L \in \mathbb{N}^+$, denotes the readout map. The sequences (z_t) , (y_t) , and (x_t) represent the input, output, and reservoir states, respectively.

In the case of an ESN, the reservoir resembles the neuron dynamics in a recurrent neural network and its dynamic system is updated by the following rule:

$$\begin{cases} x_{t+1} = lr \cdot x_t + (1 - lr) \cdot f(W_{res} \cdot x_t + W_{in} \cdot z_{t+1} + \zeta) \\ y_{t+1} = W_{out}^\top \cdot x_{t+1} \end{cases} \quad (3.2)$$

where

- $x_{t+1} \in \mathbb{R}^N$ represents the reservoir states at time $t + 1$ and belongs to an N -dimensional real vector space.
- $z_{t+1} \in \mathbb{R}^K$ represents the input signal at time $t + 1$, and belongs to a K -dimensional real vector space.
- $y_{t+1} \in \mathbb{R}^L$ represents the output signals at time $t + 1$, and belongs to an L -dimensional real vector space.
- $f: \mathbb{R}^N \rightarrow \mathbb{R}^N$ is known as the activation function, and operates on N -dimensional real vector space producing N -dimensional real vectors as output. A widely adopted choice being $f(\mathbf{x}) = \tanh(\mathbf{x})$ [133, 244].
- W_{res} denotes an $N \times N$ weight matrix that corresponds to the reservoir, the internal weight matrix.
- W_{in} denotes an $N \times K$ weight matrix that corresponds to the input, the input weight matrix.
- W_{out} denotes an $N \times L$ weight matrix. It is obtained after training the readout layer, the output weight matrix.
- $\zeta \in \mathbb{R}^N$ denotes an N -dimensional vector that represents the input shift.
- $lr \in [0, 1)$ represents the leaking rate of the reservoir.

At each time step, the input signal z_{t+1} is mapped to an internal reservoir state x_{t+1} according to equation (3.2). This process is repeated over d time steps, where d represents the input length. The resulting reservoir states are stored in a matrix X of dimensions $d \times N$, where N denotes the reservoir size and also represents the

dimensionality of the space onto which the input signals are projected. To provide a clearer understanding and visual interpretation, each row of the matrix X corresponds to a reservoir state in the N -dimensional space. The reservoir state at any given time is depicted as a point in the N -dimensional space. The internal reservoir state x_{t+1} is then mapped to the desired output y_{t+1} , which is stored in matrix Y with dimensions $d \times L$.

As mentioned, only the readout layer requires training (see Section 2.3.2 and Section 3.1.1). The readout connections, which are represented by the W_{out} matrix, are typically trained using linear regression [244] and are given by:

$$W_{out} = (X^T X + \alpha I_N)^{-1} X^T Y \quad (3.3)$$

Where X is a $d \times N_{res}$ matrix representing the internal reservoir states, and Y is a $d \times L$ matrix representing the output signal values. The model is trained on d samples, and α denotes the regularization parameter used in ridge regression. A larger value of α enforces stronger regularization, shrinking the regression coefficients and simplifying the model, while a smaller α allows the model to fit the training data more closely, but risks overfitting. That is, it may perform exceptionally well on the training data, but it might struggle to generalize to new, unseen data. Therefore, proper tuning of α is crucial to achieve good generalization. A complete mathematical derivation of the ridge regression solution is provided in Appendix 2 for reference.

Following the definition of Ballarin et al. [37], we can rewrite the reservoir state update in equation (3.2) as:

$$\begin{cases} x_{t+1} = lr \cdot x_t + (1 - lr) \cdot f(\rho \cdot \overline{W_{res}} \cdot x_t + \gamma \cdot \overline{W_{in}} \cdot z_{t+1} + \omega \cdot \bar{\zeta}) \\ y_{t+1} = W_{out}^\top \cdot x_{t+1} \end{cases} \quad (3.4)$$

where

- $\overline{W_{res}} = \frac{W_{res}}{\max_i |\lambda_i(W_{res})|}$ is the reservoir weight matrix normalized by its largest absolute eigenvalue.
- $\overline{W_{in}} = \frac{W_{in}}{\|W_{in}\|}$ denotes the normalized weight matrix corresponding to the input signals.
- $\bar{\zeta} = \frac{\zeta}{\|\zeta\|}$ denotes the normalized input shift.
- $\rho \in \mathbb{R}_+$ represents the spectral radius of the reservoir.
- $\gamma \in \mathbb{R}_+$ represents the input scaling of the reservoir.
- $\omega \in \mathbb{R}_+$ represents the shift scaling of the reservoir.

In summary, the richness of the reservoir’s internal representations facilitates the deep understanding and encapsulation of meaningful patterns within the underlying data distribution. This, in turn, empowers the reservoir to adapt effectively to variations in the data. One of the key advantages of the ESN is its simplified topology, as it requires training only for the readout layer. In practice, the ESN implementation primarily involves performing a linear regression between the reservoir’s internal states and the desired output. This emphasizes the straightforward nature and effectiveness of the ESN.

3.1.3 Algorithm

The corresponding procedure is shown in Algorithm 1.

3.1.4 Open-Source ML Frameworks for ESN Implementation

Recent innovations in the vicinity of RC, include the development of open-source ML frameworks. *PyRCN* [358] supports the building, training, and optimizing of custom architectures for RC. *RcTorch*, based on the PyTorch ML framework, is a toolbox for RC with automated hyperparameters’ tuning [192]. *EchoTorch*, also based on PyTorch, is a module for designing ESN models, developed solely for research purposes [326]. *ReservoirPy* [375, 376], provides a user-friendly framework with advanced features, concentrating on ESN models, enabling the composition of complex ‘deep’ models, with not only offline but also online learning rules. *ReservoirComputing.jl* [262] library is proposed for modular programming of ESN systems in Julia language, comprising a considerably fast and flexible computing option compared to other alternatives. Deep Echo State Network (*DeepESN*) toolbox, developed in Matlab, extends the RC paradigm towards the deep learning framework. Particularly, *DeepESN* adopts a hierarchical structure of stacked recurrent layers, wherein the output from each layer serves as the input for the subsequent layer [126–128]. Finally, *Oger* [385], a historical toolbox for RC is now deprecated. The evolution of open-source ML frameworks for RC models implementation is depicted in Figure (3.1.1).

3.1.5 Hyperparameters in ESN Models

The hyperparameters introduced in the reservoir state update equation (3.4), namely, the reservoir size N , leaking rate (lr), spectral radius ρ , input scaling γ , shift scaling ω , and the connectivity parameters of W_{res} and W_{in} , play crucial roles in shaping the internal dynamics of the reservoir. In what follows, we discuss the influence of each hyperparameter on the reservoir’s performance and dynamics.

It is quite self-evident from the theoretical explanation and the definition of an ESN that the size of the reservoir N , denoting the number of neurons composing the reservoir, and the dimension of the higher space where the input signal is projected, is pivotal for shaping the complexity of the reservoir dynamics. A generic

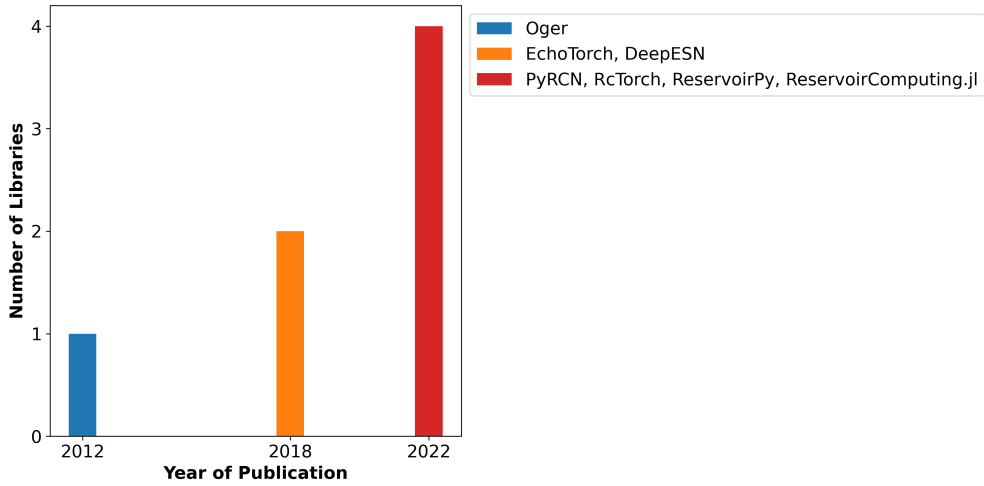


FIGURE 3.1.1: Open-Source ML Frameworks for RC System Implementation. Each bar represents the number of new libraries introduced per year.

recipe to produce "rich" reservoir dynamics suggests generating a large, and sparse reservoir, with N ranging from tens to thousands [244]. Recently, Budhiraja et al. [62] observed that contrary to expectations, a larger reservoir size (exceeding 1000 nodes) does not necessarily lead to improved performance in RC systems. Instead, it often results in overfitting, requiring appropriate regularization, and further, it demands huge computation resources. Through multiple simulations of reservoir sizes ranging from 50 to 700 nodes, with an incremental increase of 50 nodes at each step, it was determined that 400 neurons represent a suitable reservoir size for optimal performance. In a more recent study [358] into the application of the RC systems, significant emphasis was placed on the influence of the reservoir size on model performance. With reservoir sizes ranging between 50 and 500 neurons, the study concluded that an optimal reservoir size exists, specifically up to 200 neurons, wherein increasing the reservoir size leads to substantial benefits. However, it was found that beyond this threshold, the improvements become marginal [358].

Another hyperparameter, which is crucial for achieving loosely coupled reservoir activation signals, is the connectivity, representing the proportion of non-zero elements in the reservoir weight matrix (W_{res}) and the input weight matrix (W_{in}). This allows for intricate dynamics to take place but demands a balance to avoid computational challenges. The connectivity is defined through the configuration of the sparsity parameter. Different configurations for the sparsity levels of the W_{res} and W_{in} weight matrices are applied in the literature, however, a general rule of thumb suggests sparsity of 20% of non-zero connections [159, 244]. The sparse connectivity ensures that only a subset of possible connections is active. The weights in the W_{res} and W_{in} are drawn from a uniform distribution $\mathcal{U}(-1, 1)$, and the exact positions where these are placed are determined randomly according to the sparsity parameters constraints. The random signs of the weights serve functional roles: positive values impose excitatory effects, while negative values induce inhibitory influence. This balance is vital for generating diverse and stable reservoir dynam-

ics. Maintaining sparse connections between neurons facilitates not only efficient computations with sparse matrices but also tends to improve practical performance outcomes [125]. On the contrary, in densely connected reservoirs, the computational demands to update the reservoir states scales as $O(N^2)$, that is the cost increases quadratically with the size of the reservoir. An experimental analysis of the role of sparsity in the RC system is conducted in the work of [125].

The leaking rate (lr) or decay rate parameter, controls the influence of the previous reservoir state on the current state, playing a crucial role in memory retention and system stability. It regulates the speed of reservoir update dynamics. A natural constraint on the lr is that it lies between 0 and 1, meaning a neuron neither excessively retains nor leaks more activation than it holds. A lower leaking rate increases reliance on the input signal favoring faster adaptation to new inputs, while a higher leaking rate enhances memory retention by relying more on past reservoir states [244].

The input scaling parameter (γ) adjusts the system's sensitivity to external stimuli, with a higher value of γ amplifying the impact of the input. Practically, γ enables the network to capture meaningful features from the input signal without overwhelming the internal dynamics, thus facilitating a better balance between non-linear transformation and temporal integration of information.

The shift scaling parameter (ω) fine-tunes the input bias, controlling the reservoir's nonlinear behavior. It controls the amplitude of the injected noise term ζ , introducing stochasticity into the reservoir dynamics. This controlled randomness serves as a regularization mechanism that can prevent overfitting. When well-tuned, ω helps the reservoir maintain heterogeneity in its internal representations, contributing to robustness and generalization, especially in scenarios involving complex or noisy temporal data.

The spectral radius parameter (ρ) serves as a scaling factor of the reservoir's internal workings, influencing the amplification or attenuation of signals circulating within the reservoir. Thus, the spectral radius governs the effective propagation and flow of information throughout the system. Specifically, as ρ increases, it amplifies the internal weights, strengthening the influence of previous states on the current state and enhancing the reservoir's memory retention. Conversely, a smaller ρ reduces the impact of past states, leading to a more rapid decay of information over time. Venayagamoorthy and Shishir [384] study the effect of the spectral radius and settling time, the number of iterations needed for the reservoir's state to stabilize and reflect the current input, on ESN performance. They find that a spectral radius of 0.8 produces the best results, and increasing the settling time degrades performance. The increased settling time leads to decay and attenuation of echoing in the reservoir, ultimately impairing ESN effectiveness. While the condition $\rho < 1$ is traditionally associated with ensuring the Echo State Property, this condition is not strictly necessary. This is further discussed in the following section (see Section 3.1.6).

Together, these hyperparameters, including the ridge regression regularization (α) parameter, shape the performance of the ESN system.

3.1.6 Fundamental ESN Properties

Existing literature on ESN properties guides practitioners to ensure proper ESN performance. Within an ESN, the pairwise Separation Property (SP), the Fading Memory Property (FMP), and the Echo State Property (ESP) are fundamental properties that enable ESN to effectively retain relevant memory traces of the input signal and be computationally strong. The SP ensures distinct states in future time points, for input time series with differing pasts [146]. The FMP entails the gradual fading out of input influence over time [146]. This means that recent inputs have a stronger influence on the reservoir's current state, while older inputs gradually lose their impact. The ESP is an existence and uniqueness property, guaranteeing that at any given time the reservoir states are uniquely determined by the input history [124, 146, 174, 257]. A common misconception about the ESP is that the spectral radius $\rho < 1$, is a necessary and sufficient condition for its satisfaction. However, this is not true, especially for non-linear learning tasks. What is true is that ESP is violated when $\rho > 1$, under a tanh activation function, with zero input [244]. ESP does not guarantee optimal reservoir performance, or that the reservoir can model any type of nonlinearity. It merely asserts that the reservoir remains within a certain "echo state", meaning it exhibits sufficient input sensitivity, but it does not grow without any bound. Ultimately, reservoir performance depends on hyperparameter tuning, reservoir design, and the overall system configuration.

3.1.7 Probing Inner Chaos in Reservoir Dynamics

From a theoretical standpoint, a proper understanding of reservoir dynamics and measures for these dynamics is very important. This is an attempt to provide some insights into the internal opaque structure of the reservoir dynamics leveraging chaos theory principles. For an introduction to chaos, a good foundation is the work of [407].

Chaos is formally defined and widely comprehended as "a state of utter confusion", as stated in Merriam-Webster dictionary [1].

Another definition of chaos that pertains to the field of dynamical systems theory and nonlinear dynamics, which are fundamental notions to disciplines such as physics, mathematics, and various branches of engineering, is the following. "Sustained and random-like long-term evolution that satisfies certain special mathematical criteria and that happens in deterministic, nonlinear, dynamical systems" or "Largely unpredictable long-term evolution occurring in a deterministic, nonlinear dynamical system because of sensitivity to initial conditions" [407].

In Devaney's definition it is given a mathematical characterization of chaos, which

states that a system is said to be chaotic if it is topologically transitive, sensitive to initial conditions, and has dense periodic orbits [39]. To further explain, topological transitivity ensures that the system has trajectories exploring the entire phase space, while dense periodic orbits indicate the presence of recurrent, yet densely distributed, periodic behavior within the chaotic dynamics.

Chaos analysis is paramount in the study of dynamical systems. The significance of chaos analysis lies in its capacity to reveal limitations of reliable predictions and recognise conditions where long-term predictions are futile. "It is worthwhile knowing ahead of time when you can't predict something", as James Yorke pointed out [407]. The detection of chaos, provides insight into whether a seemingly erratic behavior is in fact deterministic, thereby bringing a clearer understanding of predictability capabilities.

Chaos manifests itself in deterministic, nonlinear, dynamical systems [407]. The predicting reservoir serves as a prime example of such systems. Its internal states evolve over time, influenced by prior internal dynamics and input signals, inherently forming a nonlinear system. By definition, the reservoir operates under fixed rules, making it deterministic, yet its internal mechanisms remain largely opaque, generally regarded as a "black box" [244]. Given these characteristics, it comes naturally to study the reservoir's dynamics for signs of chaos in an attempt to shed more light and provide a clearer understanding of the underlying laws governing its internal state.

Attractors: Order Amidst Chaos

The detection of chaos is a challenging task, especially when considering that chaos gives the appearance of randomness. However, it's crucial to note that chaos involves deterministic evolution. This means, there is order within chaos [407]. In the realm of chaos theory, a pivotal concept emerges—the attractor. Its role in the intricate dynamics of chaotic systems is described in what follows.

An attractor is a set of states toward which a dynamical system naturally evolves over time, defining its long-term behavior. The attractor can be considered the system's Identification Card (ID) [407]. In the context of a reservoir, the trajectories of its internal states in phase space gradually converge toward the attractor, regardless of initial conditions. This is because the attractor represents the system's asymptotic behavior or equilibrium state and remains stable over time. Strikingly, if an initial state lies on the attractor, the trajectory remains there indefinitely; while any other trajectory approaches it asymptotically without ever fully reaching it.

There is a taxonomy of attractors within dynamical systems, which classifies attractors into non-chaotic and chaotic. The non-chaotic attractors allow an accurate long-term evolution of prediction. In the non-chaotic class of attractors, belong the point, periodic or limit cycle, and torus. A popular example of the point attractor is

the movement of pendulum in an environment with friction, where it comes to rest in one state, with time. This is represented as a single fixed point in the phase space [77].

The limit cycle attractor is represented as a loop and describes a system of regular periodicity. A predator-prey data relation is generally depicted this way. The cyclical pattern begins with numerous prey, leading to a growing predator population. However, as the predator consume the prey, the prey's numbers decline, subsequently causing a reduction in the predator population due to limited food. Eventually, with reduced predation, the prey population recovers, restarting the cycle [5].

The last type of non-chaotic attractor is the torus. A torus is formed by the interaction of two or more limit cycles, where each cycle denotes an independent dynamical system. In this composite system, the torus, the overall structure typically resembles that of a three-dimensional doughnut within the phase space [407].

In the aforementioned instances of attractors, any minor initial errors do not pose a considerable long-term effect, and neighboring trajectories remain close. Hence, predictions of a trajectory's evolution are considerably valuable.

On the contrary, chaotic attractors' long-term evolution is inherently unpredictable. These attractors are often poorly understood, with their dynamics and geometry remaining largely unknown. They exhibit complex and irregular patterns, primarily resulting from two processes: stretching and folding.

Stretching describes the case when neighboring trajectories of the phase space diverge, and folding, occurs when trajectories reach the phase space boundaries and rebound. The repetitive folding maneuver results in a complex, multi-layered internal structure, which is usually a fractal, that is a repetitive pattern over different scales [407]. Chaotic attractors can still be considered as the limiting behavior of a dynamical system, representing a set of infinitely many points to which the system converges in phase space.

Otherwise defined, a chaotic attractor exhibits extreme sensitivity to initial conditions. Sensitivity to initial conditions refers to the phenomenon where even a tiny difference in the starting values of a system's variables can lead to vastly different outcomes over time. In non-chaotic systems, trajectories typically converge or remain equidistant, as seen on tori, while in chaotic systems, trajectories diverge.

It is a common misconception that the term strange attractor is synonymous with chaotic attractor. Formally, the term strange refers to the fractal geometry of the attractor, characterized by a complex, self-similar structure that repeats at different scales, regardless of whether the system's dynamics are chaotic. While all chaotic attractors are indeed strange due to their complex structure, not all strange attractors exhibit chaos. In fact, there are strange attractors that are non-chaotic, where the system displays fractal structure without sensitive dependence on initial conditions [367]. In Figure (3.1.2), the various types of attractors are illustrated.

The most common interpretation of sensitive dependence on initial conditions is unpredictability. Another interpretation is that an infinitely small change in any

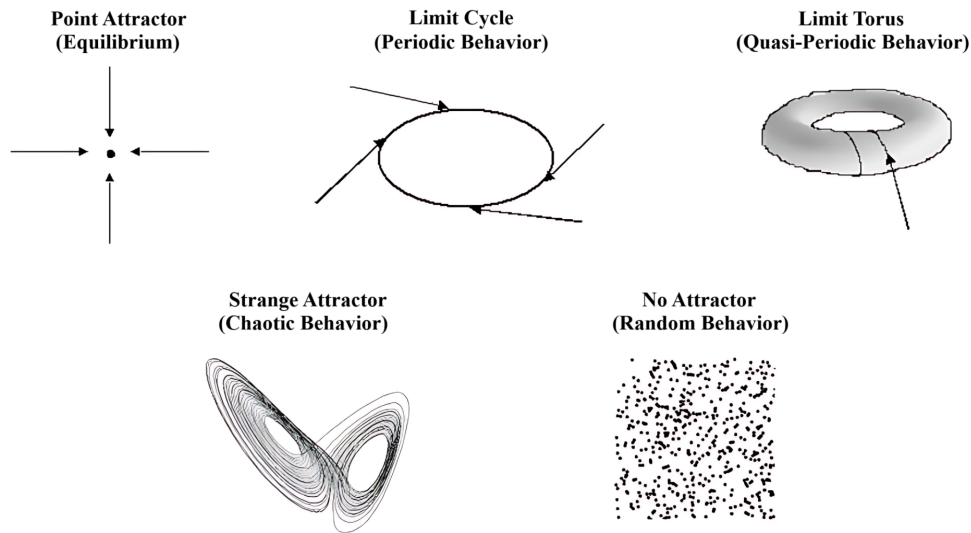


FIGURE 3.1.2: Types of Attractors: Point Attractor (Equilibrium) — a stable state where trajectories converge to a single point; Limit Cycle (Periodic Behavior) — a closed trajectory representing regular, repeating oscillations; Limit Torus (Quasiperiodic Behavior) — a phase space structure formed by the interaction of multiple limit cycles, each representing a separate dynamical system; Strange Chaotic Attractor — a fractal structure with sensitive dependence on initial conditions, exemplified by the Lorenz attractor; No Attractor (Random Behavior) — a random scatter of points in phase space, with no specific patterns [338]

external condition can cause significantly different and unexpected long-term behavior, as exemplified by the "butterfly effect." This is illustrated by the metaphor that the flap of a butterfly's wings in one part of the world, such as Brazil, could create different "initial conditions" and trigger a tornado in another part, like Texas, at least theoretically.

Sensitivity to initial conditions is a hallmark of chaos, and one practical implication is that, even though a system may be deterministic with known governing laws, its long-term predictions can be meaningless [407]. As Williams [407] states, "sensitivity to initial conditions points out the folly of placing too much emphasis on the idea of determinism." Extreme sensitivity, when combined with practical issues such as measurement inaccuracies, roundoff errors, and computational limitations, severely constrains the reliability of long-term predictions in chaotic systems.

Edge of Chaos

Research has focused on system dynamics at the boundary region between ordered and chaotic behavior, known as the "edge of chaos" [407]. This area is of particular interest, as it is where the behavior of dynamical systems is most complex. Furthermore, a link between the computational power of these systems and the edge of chaos has been proposed.

It is widely acknowledged that the maximal computational capacity of a dynamical system is not found in purely ordered or purely chaotic regimes, but somewhere in between, at the critical transition between the two. This intermediate zone, the edge of chaos, strikes a balance between stability and chaos, where a system is complex enough to capture intricate patterns, yet stable enough to be controlled and trained effectively.

The hypothesis that the computational power of dynamical systems is maximized at the boundary between ordered and chaotic system behavior can be traced back at least to Langton [214] who studied cellular automata. Further empirical support is provided by Bertschinger et al. [48], who emphasize that dynamics near the critical line are a general property of input-driven dynamical systems, enabling them to support complex real-time computations.

Natschläger et al. [276] examine input-driven recurrent networks with threshold units and find that computational performance is maximized at the phase transition. The authors introduce a complexity measure called the Network Mediated Separation (NM-separation), which serves as a predictor for the computational capability of a network, and it is found to peak at the critical phase, where order and chaos converge. As such, their work provides a direct link between the edge of chaos and computational efficiency, highlighting the importance of dynamics near the critical phase in input-driven dynamical systems.

Similarly, Legenstein and Maass [220] demonstrate that the edge of chaos coincides with regions of highest computational performance, particularly in tasks such as the classification of noisy spike templates using a trained linear readout. The authors suggest evaluating the computational power of a reservoir through measures such as kernel quality and generalizability. They argue that the reservoir's computational capabilities are characterized by a trade-off between these two properties, and show that it is most effective at the edge of chaos.

Later, Boedecker et al. [57] show that information storage and transfer are maximized near the phase transition, proving the computational utility of systems at the edge of chaos. Their analysis focuses on a local scale (node-level), studying information storage at each neuron and information transfer between each neuron pair within the reservoir. Similarly, Haruna and Nakajima [155] analytically investigate the short-term memory capacity of ESNs using three distinct measures, namely memory capacity, mutual information, and Fisher information. Their results demonstrate that all three measures peak before the edge of chaos, where the dynamics is stable for input-driven networks, but unstable when the input is absent.

More recently, Zhang et al. [436] demonstrate that the edge of chaos can be actively leveraged to achieve computational optimality by controlling the model's dynamical evolution within the order–chaos phase diagram during training. Using a simple feedforward architecture, it is shown that training with SGD with momentum drives the system from order to chaos, and generalization performance peaks precisely at the transition between these two regimes.

On the other hand, Carroll [65] questions the general applicability of this principle to dynamical systems, including reservoir computers. While many earlier studies report optimal performance near the edge of chaos, the author demonstrates that it is not universally true. It is shown that although maximum entropy may occur near the edge of stability (a term often used interchangeably with, the edge of chaos), computational performance (e.g., prediction accuracy) can degrade due to poor generalized synchronization and mismatched frequency responses. Their simulations suggest that optimal performance does not necessarily align with the edge of chaos, suggesting that synchronization between the input system and reservoir computers may be more critical for tasks like signal fitting and prediction.

Despite these criticisms, this research adopts the view that operating near the edge of chaos remains advantageous. This regime allows reservoirs to exhibit rich, expressive behavior while preserving a well-structured state space for consistent, and reliable learning. Systems in this zone exhibit neither overly rigid nor excessively erratic dynamics, facilitating the modeling and prediction of complex temporal patterns.

It is important to note that chaos is not a sharply defined property, but rather a broad and multifaceted phenomenon. No single metric can fully describe its complexity. As a result, researchers employ various quantitative measures to characterize systems near or within chaotic regimes. Common tools include entropy, which quantifies unpredictability, Lyapunov exponents, which measure sensitivity to initial conditions, and fractal structures, which capture geometric complexity. The use of multiple complementary metrics reflects the inherently ambiguous, “gray” nature of chaos, highlighting that the edge of chaos is not a fixed boundary, but a dynamic region where computational richness and reliability coexist.

Estimation of the largest Lyapunov exponent

The Lyapunov Exponent is a fundamental concept in describing chaos providing a quantitative measure of sensitivity to initial conditions [285, 314]. The rationale behind this is that small perturbations in the initial conditions of two otherwise identical systems will gradually die out if the system is in the ordered regime. In contrast, in a chaotic regime, such perturbations persist and are exponentially amplified. The Lyapunov exponent serves as a quantitative measure of this behavior.

In a more formal definition, Lyapunov Exponents quantify the average rate of divergence between nearby trajectories in a dynamical system [46, 357, 407], offering a numerical indicator of chaos. It can take positive, negative, or zero values. A positive Lyapunov Exponent denotes the presence of chaos within the reservoir [19], implying sensitivity to initial conditions and reduced predictability, which can hinder the reservoir’s ability to capture and learn patterns from the input signal. In contrast, a negative value indicates stable dynamics, where the reservoir may become overly rigid and insufficiently responsive to input variations, lacking the necessary

complexity.

The critical transition phase, where the Lyapunov Exponent crosses zero, marks the boundary between chaos and stability. This is particularly important, as reservoirs operating near zero Lyapunov Exponent are said to lie at the edge of chaos, a regime believed to balance complexity and stability optimally.

There are two distinct classes of methods, employed for the estimation of Lyapunov Exponents from observational data [263].

Direct methods, as proposed by Guckenheimer [148] and implemented by Wolf et al. [408], are grounded in the assumption that the norm of the difference between trajectories grows exponentially at a rate given by the largest (dominant) Lyapunov Exponent. These methods estimate the growth rate of an infinitesimal perturbation by tracking the divergence between initially close observed data points in a reconstructed phase space. Pairs of time points where the trajectories are sufficiently close are identified, and their divergence is monitored over a finite time until a predefined threshold is exceeded. When this occurs, a new nearby point is selected, and the process continues. The average divergence rate over the entire dataset provides an estimate of the largest Lyapunov exponent. A refinement of this approach was proposed by [314], who developed a more robust and computationally efficient algorithm that is also reliable for small datasets, by leveraging the logarithmic divergence of nearest neighbors in a reconstructed phase space and applying a linear least squares fit to extract the largest Lyapunov Exponent.

On the other hand, Jacobian methods, proposed by [101] and implemented by [100], involve estimating the Jacobian matrices of the system's dynamics from the time series data. A linear approximation is then employed to model the relationship between these data points, allowing for the estimation of the Jacobian matrix at a specific point. This linear approximation is used to determine the time evolution of the Jacobian matrix along the trajectory of the system. The Lyapunov Exponents are subsequently determined from the eigenvalues of the product of the estimated Jacobian matrices and their transposes. Working autonomously, [323] introduced a methodology for calculating Lyapunov Exponent values that shares a philosophical resemblance to the approach presented by [100], with distinguishing features lying in the orthonormalization technique employed.

Both classes of methods provide avenues for estimating Lyapunov Exponents, with direct methods relying on trajectory divergence and Jacobian methods incorporating linear approximations of the system's dynamics.

Our approach for computing the largest Lyapunov exponent is based on the method proposed by Rosenstein et al. [314], which estimates the exponent by examining the average exponential divergence of initially close trajectories. In this context, we utilize the internal reservoir states generated by an ESN and, for each selected reference state, we identify a set of nearest neighbors based on Euclidean distance within the high-dimensional reservoir state space. We then track the temporal evolution of the separation between each reference trajectory and its corresponding neighbors over a sequence of time steps. At each time delay τ , we compute

the mean of the stretching factors, defined as the logarithm of the ratio between the distances at time τ and at the initial time. By fitting a linear model to the average log-divergence values versus τ , we estimate the largest Lyapunov exponent as the slope of this linear fit. This procedure mirrors the core idea of the Rosenstein et al. [314] method, namely, quantifying the exponential rate at which nearby trajectories diverge to characterize the system's sensitivity to initial conditions.

While the original Rosenstein et al. [314] method typically involves explicit phase space reconstruction via time-delay embedding (e.g., using Takens' theorem), our implementation bypasses this step by leveraging the ESN reservoir. The reservoir maps the input into a high-dimensional state space, effectively capturing the system's temporal dependencies and serving as a surrogate for phase space embedding. Thus, the divergence of reservoir trajectories can be tracked directly in this latent space without manual embedding, offering a computational advantage.

Our proposed practical implementation for estimating the largest Lyapunov exponent, while it loosely resembles the method of Wolf et al. [408] in its use of nearest neighbors to assess divergence, differs significantly by not incorporating Gram-Schmidt re-orthonormalization, or neighbor replacement strategies, which are key components to Wolf et al. [408]'s technique. Therefore, despite a superficial similarity in the conceptual use of neighboring trajectories, our approach fundamentally aligns with the method of Rosenstein et al. [314].

The primary objective is to evaluate the system's sensitivity to initial conditions and to distinguish deterministic chaos from stochastic noise. By computing the largest Lyapunov exponent across multiple reference trajectories, we obtain a distribution of local divergence rates, which enables us to characterize the reservoir dynamics as stable, or chaotic. While both deterministic chaos and stochastic noise can produce seemingly irregular time series, chaotic systems are governed by deterministic rules, exhibit structured behavior, and are characterized by sensitivity to initial conditions and positive Lyapunov exponents. Detecting such structure is essential, as it indicates short-term predictability and the presence of potentially exploitable information within the reservoir dynamics.

In summary, by harnessing the ESN's capacity to represent temporal information in a nonlinear, high-dimensional state space, our approach provides a computationally efficient and conceptually robust framework for estimating the largest Lyapunov exponent in complex, data-driven systems. The theoretical foundation, practical implementation, and corresponding algorithm are presented below.

Theoretical Basis The Lyapunov exponent quantifies the average exponential rate at which nearby trajectories in a dynamical system diverge, offering a quantitative measure of sensitivity to initial conditions. For a system defined by the evolution rule $\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t)$, let $\delta\mathbf{x}_0$ denote an infinitesimally small perturbation to the initial condition. Then the separation at time t is approximated by:

$$\|\delta \mathbf{x}_t\| \approx \|\delta \mathbf{x}_0\| e^{\lambda t} \quad (3.5)$$

where λ is the Lyapunov exponent. Taking logarithms, one obtains:

$$\log \|\delta \mathbf{x}_t\| \approx \log \|\delta \mathbf{x}_0\| + \lambda t \quad (3.6)$$

This linear relationship suggests that the Lyapunov exponent λ can be estimated as the slope of $\log(\text{separation})$ versus time, providing insight into the system's local stability or chaotic behavior. Rearranging the equation, we solve for λ :

$$\lambda \approx \frac{1}{t} \log \left(\frac{\|\delta \mathbf{x}_t\|}{\|\delta \mathbf{x}_0\|} \right) \quad (3.7)$$

In multi-dimensional systems, a spectrum of Lyapunov exponents typically exists, denoted as $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \dots$, where each exponent characterizes the average rate of divergence or contraction along a particular direction in phase space. Among these, the largest Lyapunov exponent is of particular importance, as it dominates the system's long-term behavior. In this analysis, we do not aim to compute the full spectrum of Lyapunov exponents, instead, our focus is restricted to estimating only the largest Lyapunov exponent, to identify any chaotic behavior.

Practical Estimation We estimate the Largest Lyapunov Exponent (LLE) from the reservoir dynamics of the ESN by quantifying the average exponential divergence of nearby trajectories. For each reference state, we identify close neighbors in the state space and monitor how the Euclidean distance between them evolves over time (across τ time steps). By computing the average logarithmic divergence across multiple pairs over a range of time delays τ , we perform a linear fit to the divergence curve. The slope of this fit provides an estimate of the LLE.

Let $\mathbf{X} \in \mathbb{R}^{T \times D}$ be the matrix of reservoir states after the washout period, where T is the number of time steps and D is the dimensionality (i.e., number of neurons in the reservoir).

1. **Stratified Sampling:** The reservoir states are divided into $K = 5$ strata by segmenting the data into equally sized intervals. From each stratum, approximately 4% of the states are randomly selected as reference trajectories, resulting in a total of 20% of the total reservoir states across all strata being used as reference trajectories. This sampling strategy reduces computational overhead, offering a representative snapshot of the overall state space.
2. **Dynamic Neighborhood Construction:** For each reference trajectory $\mathbf{x}_i$, its dynamic neighborhood $\mathcal{N}_i$ is formed by selecting the top 1% of closest states $\mathbf{x}_j$ ($j \neq i$), based on the Euclidean distance:

$$\text{Distance}(i, j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \sqrt{\sum_{k=1}^D (x_{i,k} - x_{j,k})^2},$$

where D is the number of reservoir units. These neighborhoods are then used to track the divergence over time using the evolved states $\mathbf{x}_{i+\tau}$ and $\mathbf{x}_{j+\tau}$.

3. **Stretching Factor Computation:** For each time delay $\tau \in [\tau_{\min}, \tau_{\max}]$, compute the distance between the evolved states of the reference and neighboring trajectories at time τ .

$$d_0 = \|\mathbf{x}_i - \mathbf{x}_j\|, \quad d_\tau = \|\mathbf{x}_{i+\tau} - \mathbf{x}_{j+\tau}\|$$

and the corresponding stretching factor:

$$\sigma(\tau) = \log\left(\frac{d_\tau}{d_0}\right)$$

To reduce noise and obtain a robust estimate, this is averaged over all neighbors in $\mathcal{N}_i$, hence:

$$\bar{\sigma}_i(\tau) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \log\left(\frac{\|\mathbf{x}_{i+\tau} - \mathbf{x}_{j+\tau}\|}{\|\mathbf{x}_i - \mathbf{x}_j\|}\right)$$

4. **Linear Regression:** A linear regression is performed on the plot of $(\tau, \bar{\sigma}(\tau))$, where $\bar{\sigma}(\tau)$ denotes the average of $\log(d_\tau / d_0)$ across all states in the neighborhood. The slope of the resulting line gives an estimate of the largest Lyapunov exponent:

$$\lambda \approx \frac{d}{d\tau} \log\left(\frac{d_\tau}{d_0}\right)$$

Note: Delay values τ are chosen as consecutive integers from 1 to 100. The upper limit of 100 is selected as a practical baseline, large enough to ensure a stable linear fit of the stretching factors, yet not so large as to reduce the amount of valid data (since $\mathbf{x}_{i+\tau}$ must remain within the dataset bounds) or to entangle results with macro-scale effects, thereby no longer measuring local sensitivity to initial conditions.

Note: Since the system already operates in a high-dimensional state space, defined by the number of neurons in the reservoir, traditional state-space reconstruction is not required for estimating Lyapunov exponents. Each reservoir state vector inherently captures the system's dynamic evolution in this expanded space.

Algorithm The resulting LLE values quantify the system's sensitivity to initial conditions and are used to assess the level of chaos within the ESN's reservoir dynamics. The corresponding procedure is shown in Algorithm 2.

Separation Ratio

The Separation Ratio is a less commonly used metric for assessing the quality of a reservoir in an RC system, offering insight into the reservoir's ability to discriminate between different classes of input data. It was originally introduced by Goodman and Ventura [141] and further explored in subsequent works [75, 136, 281]. The

metric quantifies how well the reservoir states amplify differences in input signals, which is essential for ensuring meaningful downstream classification or regression tasks.

Traditionally, the separation ratio is less applicable in cases of a continuous field of values, where the input signals are not inherently categorized into discrete classes [136]. Extending the concept of separation, to cases without predefined classes, the separation concept serves as a general measure of reservoir quality based on the principle that significantly different input signals should induce correspondingly distinct reservoir states, thereby enabling the system to preserve input differences in its dynamic representation [80, 136, 219].

Let $\mathbf{z}_i(t)$ and $\mathbf{z}_j(t)$ denote two input signals that are significantly different such that:

$$\|\mathbf{z}_i(t) - \mathbf{z}_j(t)\| = D, \quad D \gg 0 \quad (3.8)$$

A well-functioning reservoir should produce corresponding reservoir states $\mathbf{x}_i(t)$ and $\mathbf{x}_j(t)$ that satisfy:

$$\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\| \geq D \quad (3.9)$$

Thus, the Separation Ratio S is defined as:

$$S = \frac{\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|}{\|\mathbf{z}_i(t) - \mathbf{z}_j(t)\|} \geq 1 \quad (3.10)$$

A separation ratio $S \geq 1$ indicates that the reservoir expands or at least preserves the distance between distinct inputs, thereby supporting effective input discrimination. Higher values of S are desirable, as they reflect improved separability of input patterns in the reservoir space.

The concept of separation has also inspired practical extensions in more complex scenarios. Gibbons [136] proposed the Separation Ratio Graph (SRG), which combines the separation ratio with Lyapunov exponent analysis, linking input separability to the underlying chaotic dynamics of the reservoir. Valencia et al. [380] applied SRG as a post-processing criterion following the application of a Genetic Algorithm (GA), to identify the optimal reservoir among several ESN initializations with optimized hyperparameters from the GA. This framework is referred to as ESN-GA-SRG.

Controlled Chaos

In classical dynamical systems, the largest Lyapunov Exponent characterizes the average exponential rate at which nearby trajectories in a dynamical system diverge over time, as expressed in equation (3.7). This formulation assumes an autonomous system, where internal dynamics solely drive the evolution of state trajectories. In such systems, the initial perturbation $|\delta \mathbf{x}_0|$ represents the distance between two infinitesimally close initial states.

However, in input-driven systems such as RC, the system's trajectory is largely shaped by the external input sequence rather than its internal dynamics' interplay. In our ESN setting, the notion of perturbation is better captured by differences in input sequences. We thus reinterpret $|\delta \mathbf{x}_0|$ as the Euclidean distance between two input signals $\mathbf{z}_i$ and $\mathbf{z}_j$ that generate reservoir states $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, i.e., $|\delta \mathbf{x}_0| \approx \|\mathbf{z}_i - \mathbf{z}_j\|_2$.

This reframing allows the Lyapunov Exponent to capture the system's input-responsiveness, that is, how small variations in the input propagate through the reservoir. As such, it does not purely reflect autonomous chaos, but rather the reservoir's input-driven sensitivity.

This leads us to the input-driven Lyapunov Exponent, given by:

$$\lambda \approx \frac{1}{t} \log \left(\frac{\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|}{\|\mathbf{z}_i(t) - \mathbf{z}_j(t)\|} \right) \quad (3.11)$$

where $\mathbf{z}_i(t)$ and $\mathbf{z}_j(t)$ are input signals that generate reservoir states $\mathbf{x}_i(t)$ and $\mathbf{x}_j(t)$ at time t , respectively. This formulation adapts the classical concept to quantify the reservoir's input-driven sensitivity, capturing how small input variations propagate through the system rather than purely measuring autonomous divergence.

As discussed in the previous section, researchers, including Bertschinger et al. [48], Boedecker et al. [57], Haruna and Nakajima [155], Langton [214], Legenstein and Maass [220], Natschlager et al. [276], Zhang et al. [436], support that optimal reservoir performance is achieved at the edge of chaos, a transitional phase where the largest Lyapunov exponent λ approaches zero.

According to equation (3.11), since λ is proportional to the logarithm of the ratio of the distance between reservoir states to the distance between their corresponding input signals, optimal performance occurs when:

$$\frac{\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|}{\|\mathbf{z}_i(t) - \mathbf{z}_j(t)\|} \approx 1 \quad (3.12)$$

This condition ensures that the relative distances between the reservoir states are similar to those between the input signals.

Building on the hypothesis that optimal reservoir performance lies at the edge of chaos, and based on equations (3.10) and (3.12), we argue that a separation ratio S defined as

$$S = \frac{\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|}{\|\mathbf{z}_i(t) - \mathbf{z}_j(t)\|} \in [1, 10]$$

supports controlled chaos within the reservoir dynamics. This range of S ensures maximum processing capabilities while maintaining system stability. It enables a moderate amplification of input signal differences within the reservoir, thereby enhancing input separability and information retention, all while preventing instability or unbounded divergence (erratic behavior) in highly nonlinear and dynamical

systems.

Preliminary empirical evaluations are conducted on our settings, and extensive tests are performed to identify the ideal separation ratio range. Based on these experiments, we determined that the range $[1, 10]$ is an effective indicator of "good" reservoir performance in the context of our frameworks (see Chapter 4). This range serves as a meaningful starting point for evaluating and tuning reservoir systems at the edge of chaos. Nevertheless, we recommend that practitioners experiment with and adjust this range based on the specific requirements of their task.

Interpreting Reservoir Dynamics through Kernel PCA Visualization

The aforementioned represent quantitative metrics from the theory of deterministic, nonlinear, dynamical systems. Complementary to this, we employ a qualitative tool to investigate the inner workings of an ESN reservoir. In particular, we employ *Kernel Principal Component Analysis* (kPCA) [331] as a probing mechanism to reveal latent structure in the reservoir activations, that is, whether an ordered or chaotic pattern emerges therein.

In more detail, we employ kPCA as a post-hoc diagnostic tool, applied after completing the optimized-ESN prediction task, to retrospectively analyze and interpret the reservoir's internal dynamics. Specifically, we apply kPCA to the reservoir states generated from the test data. This allows us to project the high-dimensional reservoir activations onto a lower-dimensional (specifically, three-dimensional) subspace defined by the leading kernel principal components. These components represent directions in the induced feature space that capture the highest variance in the data. By assessing the internal organization of the reservoir dynamics, we are looking for a sign of order therein.

We argue that the appearance of a coherent and well-structured pattern in the projected reservoir states is a sign of order within the reservoir dynamics. In particular, a clearly discernible structure within the test set's reservoir activations suggests that the reservoir is operating in a stable, ordered, and reliable regime.

The motivation for using kPCA stems from its ability to uncover latent structure within complex data, making it particularly suitable for analyzing the internal dynamics of ESNs. ESN reservoirs exhibit high-dimensional, nonlinear behavior that often cannot be effectively captured through linear methods like standard PCA. In contrast, kPCA allows us to detect patterns, regularities, and geometric organization in the reservoir dynamics that would otherwise remain hidden. This allows to diagnose the reservoir's operational behavior visually.

Related Work One of the earliest contributions coupling dimensionality reduction with ESNs is the work by Løkse et al. [241], where a modular framework is pro-

posed that integrates kPCA between the reservoir and readout layers. In this setting, the reservoir maps the input signal into a higher-dimensional state space. Subsequently, each reservoir state is projected onto a lower-dimensional subspace before being passed to the linear readout for training. The authors note the necessity of out-of-sample extension feature of kPCA, which is particularly important in this setting, as it enables the transformation of new test data.

Tian [370] employs kPCA as a preprocessing step for feature engineering, reducing the dimensionality of the input signals, for short-term wind power forecasting with ESN. The original multidimensional input signals are first transformed using kPCA to extract the principal components most relevant to wind power. This dimensionality reduction not only simplifies the model by reducing input complexity but also removes correlations among input features, thereby improving predictive performance. The transformed inputs are then fed into an ESN whose parameters are optimized via an improved PSO algorithm.

In our work, we diverge from these functional uses of kPCA during training. Instead, we adopt kPCA purely as a post-hoc diagnostic tool for interpretability. That is, rather than transforming inputs or aiding the readout layer, kPCA is applied after prediction to project the reservoir states of the test set onto a low-dimensional manifold. This allows us to qualitatively assess whether the internal dynamics of the ESN exhibit signs of order or chaos, thereby providing insights into the reservoir's operating regime.

In related work, Recurrence Plots (RPs) have also been used as post-hoc tools to investigate the dynamics of ESNs. For the interested reader, Bianchi et al. [53] address the interpretability challenge of ESNs by employing RPs and Recurrence Quantification Analysis (RQA) to visualize and characterize the temporal dynamics of reservoir neuron activations. Their method involves constructing two-dimensional representations via RPs to visualize the high-dimensional dynamics of a reservoir. They observe that reservoir and input denote similar line patterns in RPs in the case of a stable reservoir. Otherwise, the more unstable the reservoir, the more the RP of the reservoir shows instability patterns. From these visualizations, they extract complexity measures such as L_{max} , defined as the value of the longest diagonal line in the RP, which they show to be strongly correlated with the maximal local Lyapunov exponent. This finding suggests that RP-based measures offer an important alternative to traditional network stability metrics. Notably, they demonstrate that fluctuations in RQA metrics are more reliable indicators of the edge of stability, compared to traditional Jacobian-based criteria.

We approach this matter from a different angle: rather than asking how often reservoir states recur, we examine how they are spatially organized. By focusing on the geometric structure of the reservoir dynamics, as revealed through low-dimensional projections via kPCA, we aim to provide a spatial interpretation of the reservoir's inner workings during the test phase. While RPs and RQA offer valuable insights into temporal stability and dynamic complexity, our work adopts

a complementary perspective that emphasizes the spatial structure and topology of the reservoir state space. By projecting high-dimensional reservoir states into a lower-dimensional space, we aim to reveal structural regularities, separability, and smoothness in the reservoir dynamics, particularly during generalization. This form of visualization offers an alternative lens for interpretability, shifting the focus from recurrence patterns to the overall shape and organization of state trajectories. In doing so, our contribution extends the ESN interpretability toolbox by introducing a geometry-driven, variance-based diagnostic approach that enhances the understanding of reservoir behavior.

Theoretical Basis Principal Component Analysis (PCA) is a classical technique that linearly projects high-dimensional data onto a lower-dimensional orthonormal basis in the direction of the maximum variance.

Following the eigen-decomposition method, for a random vector $X \in \mathbb{R}^p$ the covariance matrix is $\Sigma_X = E\Lambda E^\top$, where $E = (e_1 e_2 \dots e_p)$ is the orthogonal eigenvector matrix and $\Lambda = \text{diag}(\lambda_i)$ the diagonal eigenvalue matrix. Then, the linear transformation $Y = E^\top X$ yields uncorrelated components in the transformed space, as the covariance matrix of Y is given by $\Sigma_Y = \Lambda$. Importantly, the total variance is preserved under this orthogonal transformation:

$$\sum_{i=1}^p \text{Var } X_i = \sum_{i=1}^p \text{Var } Y_i = \sum_{i=1}^p \lambda_i$$

Projecting Y onto the subspace spanned by the d leading eigenvectors, those associated with the largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$, enables dimensionality reduction while preserving as much of the variance in X as possible. Formally, this is expressed as $\hat{Y} = E_d^\top X$, where $E_d = [e_1 \ e_2 \ \dots \ e_d]$ is the truncated eigenvector matrix.

However, PCA is inherently linear. To capture nonlinear structure, we utilize kPCA, which implicitly maps the input data $\mathbf{X} \in \mathbb{R}^n$ into a high-dimensional Hilbert space $\mathcal{H}$ via a nonlinear function $\phi : \mathbb{R}^n \rightarrow \mathcal{H}$. In this feature space $\mathcal{H}$, classical PCA is then performed. The mapping is never computed explicitly; instead, kPCA operates through a kernel-trick, which allows computation of inner products in $\mathcal{H}$ via a symmetric, positive semi-definite kernel function:

$$K(h[i], h[j]) = \langle \phi(h[i]), \phi(h[j]) \rangle_{\mathcal{H}}.$$

A common choice is the Radial Basis Function (RBF) kernel:

$$K(h[i], h[j]) = \exp(-\gamma \|h[i] - h[j]\|^2),$$

where γ controls the kernel width.

Given a dataset $\{h[1], h[2], \dots, h[N]\}$, we construct the kernel matrix $K \in \mathbb{R}^{N \times N}$. After centering the kernel matrix (to ensure zero mean in $\mathcal{H}$), we solve the eigenvalue problem:

$$KE = E\Lambda,$$

where $E = [e_1, e_2, \dots, e_N]$ is the matrix of eigenvectors, and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_N)$ is the diagonal matrix of eigenvalues, sorted in descending order $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$.

The projection of in-sample data into the d leading principal components in $\mathcal{H}$ is then given by:

$$\tilde{H} = E_d \Lambda_d^{1/2},$$

where each row of $\tilde{H}$ represents the coordinates of a data point in the reduced kPCA space.

For a new point $h[k]$, the projection onto the l -th principal component is approximated by:

$$\tilde{h}_l[k] = \frac{1}{\sqrt{\lambda_l}} \sum_{i=1}^N e_l(i) K(h[i], h[k]).$$

where $e_l(i)$ is the i -th entry of the l -th eigenvector. This expression allows embedding of new data points into the learned nonlinear subspace. In essence, the new point's projection is the weighted sum of kernel similarities to the training points, scaled by the eigenvalues and eigenvectors approximated during training. This allows the kPCA to generalize beyond the training data without explicitly knowing the mapping to the high-dimensional space.

Implementation Formally, we consider the reservoir state matrix $\mathbf{X} \in \mathbb{R}^{T \times N}$, where T denotes the number of time steps and N the number of reservoir neurons. Prior to dimensionality reduction, the data is standardized using z-score normalization:

$$\tilde{\mathbf{X}} = \frac{\mathbf{X} - \mu(\mathbf{X})}{\sigma(\mathbf{X})}$$

where $\mu(\mathbf{X})$ and $\sigma(\mathbf{X})$ denote the column-wise mean and standard deviation, respectively. This ensures that each feature contributes equally to the kPCA embedding.

The kPCA is then applied to the standardized reservoir state matrix $\tilde{\mathbf{X}}$ using the RBF kernel. Given the non-linear nature of the ESN's internal dynamics, the RBF kernel facilitates capturing non-linear relationships in the data by implicitly mapping it to a high-dimensional feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

where γ is the kernel coefficient, and $\mathbf{x}_i, \mathbf{x}_j$ are reservoir state vectors at different time steps.

We retain the first three principal components ($n_{\text{components}} = 3$) from the kPCA transformation and visualize these in a 3-D scatter plot. Each point in this reduced space represents the reservoir state at a specific time step, projected onto the 3 leading nonlinear components. This visualization enables qualitative inspection of the reservoir's inner workings on a low-dimensional space.

The following Python snippet illustrates the standardization, kPCA transformation, and visualization procedure employed in our analysis:

```

1 from sklearn.decomposition import KernelPCA
2 from sklearn.preprocessing import StandardScaler
3 import matplotlib.pyplot as plt
4 import pandas as pd
5
6 scaler = StandardScaler()
7 kernel = 'rbf'
8 n_components = 3
9
10 # Standardize reservoir states from the test set
11 res_dynamics_standardized = scaler.fit_transform(pd.DataFrame(
12     reservoir_states_tests))
13
14 # Apply Kernel PCA
15 kpca = KernelPCA(n_components=n_components, kernel=kernel)
16 kpca_res_dynamics = kpca.fit_transform(res_dynamics_standardized)
17
18 # Convert to DataFrame for convenience
19 kpca_res_dynamics = pd.DataFrame(kpca_res_dynamics, columns=[f'
20     Component {i+1}' for i in range(n_components)])
21
22 # Extract components for plotting
23 x_vals, y_vals, z_vals = kpca_res_dynamics['Component 1'],
24     kpca_res_dynamics['Component 2'], kpca_res_dynamics['Component 3']
25
26 # Plot in 3D space
27 fig = plt.figure(figsize=(8, 6))
28 ax = fig.add_subplot(111, projection='3d')
29 ax.scatter(x_vals, y_vals, z_vals, marker='o', s=5, color='red')
30 ax.set_xlabel("Kernel PCA Component 1")
31 ax.set_ylabel("Kernel PCA Component 2")
32 ax.set_zlabel("Kernel PCA Component 3")
33 plt.tight_layout()
34 plt.show()

```

LISTING 3.1: Python code for applying kPCA to standardized reservoir states and visualizing the results in 3D

Implications We employ kPCA as a post-hoc exploratory tool to examine how reservoir structure maps to prediction reliability in ESNs. After training and prediction, kPCA is applied to the reservoir states derived from test data, projecting the high-dimensional activations onto a three-dimensional subspace defined by the leading kernel principal components, which capture the directions of maximum variance. The emergence of a coherent and structured pattern in this reduced space

is interpreted as evidence of ordered reservoir dynamics, indicative of a stable and reliable regime. Observing a similar geometric structure in reservoir states from the training data would further reinforce this interpretation, suggesting robustness and consistency in the reservoir’s internal response to inputs, and thereby suggesting good potential for generalization.

This insight leads to broader implications for ESN design and model selection. Given that ESN training is typically limited to solving a linear regression problem between the reservoir states and target outputs (e.g., via ridge regression), being able to assess the quality of the reservoir dynamics before engaging in full training can yield significant computational benefits. Specifically, by applying kPCA to reservoir states generated from input signals prior to output weight training, one can early identify reservoirs that lack structured dynamics, discard them, and proceed with re-initialization to generate a new candidate reservoir with potentially richer and more suitable dynamics.

A key contribution of this research is that the presence of discernible structure in the kPCA-projected reservoir dynamics serves as a reliable proxy for reservoir quality. While our use of kPCA is post-hoc, the same principles could guide future efforts to build pre-training diagnostics that enable more efficient model selection by filtering out poorly initialized reservoirs early in the modeling pipeline.

The experimental results of our analysis on probing inner chaos in reservoir dynamics are presented in Section 4.2, specifically in Subsection 4.2.4 titled *Reservoir Stability: Probing Inner Chaos*.

3.1.8 Limitations in ESN Design

ESNs share several limitations with other DL architectures, including the curse of dimensionality and overfitting. The curse of dimensionality applies in ESN, with computational demands to update the reservoir states scaling as $O(N^2)$. That is the cost increases quadratically with the size of the reservoir. Moreover, high connectivity levels (i.e., low sparsity) make the reservoir more densely connected which can increase computational cost and slow convergence.

Overfitting is another prominent concern. ESNs are particularly prone to overfitting when the training data is limited or noisy, and regularization is improperly applied. Larger reservoirs, though potentially more expressive, can exacerbate this issue by enabling the model to memorize rather than generalize patterns.

Beyond these general issues, ESNs face specific challenges arising from their unique architectural design. At the heart of an ESN lies the reservoir, a nonlinear dynamical system with sparse recurrent connectivity, which transforms input signals into rich, high-dimensional representations. However, the inherent dynamics of the reservoir may induce chaos within the system. Chaos, in this context, is not necessarily detrimental (as discussed in subsection 3.1.7). In fact, when appropriately regulated, it can enhance the network’s expressive power and its ability to capture

complex, nonlinear temporal patterns. Yet, this chaotic behavior is a double-edged sword. Excessive chaos may destabilize the network and significantly reduce the effective prediction horizon. Most critically, given that chaos entails extreme sensitivity to initial conditions, it can violate the ESP, which ensures that any reservoir's state is uniquely determined by its input history, thereby compromising reproducibility and reliability. As a result, a key challenge in the design of ESNs is to manage this trade-off: harnessing the richness of chaotic dynamics while preserving sufficient stability to maintain the ESP.

Another limitation lies in the black-box nature of the reservoir. Unlike traditional neural networks, where weight updates offer insight into feature importance and decision logic, ESNs rely on a randomly initialized and fixed reservoir. This reservoir maps inputs into a high-dimensional state space through complex nonlinear dynamics, making the internal computational processes conceptually opaque and difficult to interpret. Such lack of transparency can undermine user trust and hinder ESNs' adoption in high-stakes domains such as finance and healthcare, where interpretability is essential. Interdisciplinary approaches and novel techniques, such as analyzing internal state trajectories, visualizing reservoir dynamics, or employing interpretable surrogate models, could help reveal how information is processed and propagated within the reservoir.

Additionally, collinearity among reservoir states can degrade model performance. When internal states are highly correlated, the network's capacity to distinguish between different patterns is reduced, leading to diminished predictive accuracy.

ESNs are also particularly sensitive to hyperparameter initialization. Given that the reservoirs are untrained and randomly set, their initial configuration plays a pivotal role in model behavior. Different initializations can lead to significantly different behaviors, and as noted by Soltani et al. [347], ESNs exhibit clear disadvantages when parameters are not properly initialized. However, in the absence of universal design rules for ESNs, finding optimal configurations often requires computationally intensive search processes.

This brings us to a final paradox: although ESNs are efficient in training, as only the readout weights are learned, achieving high performance often depends on complex hyperparameter optimization. Advanced search techniques are typically employed to fine-tune the hyperparameters and architectural design of ESN models. Yet, these methods can incur substantial computational cost, especially for large-scale problems. On the other hand, simplifying the optimization process may speed up computation but typically comes at the cost of reduced accuracy. As a result, designing ESNs presents a key trade-off between optimization efficiency and predictive performance, a balance that remains an open challenge in the field.

To this end, the following subsection provides a discussion of existing optimization approaches for ESNs.

3.1.9 Existing Optimization Approaches for ESNs

Manual configuration was initially employed for ESN applications, with general guidelines given for optimal selection of individual hyperparameters [62, 125, 159, 244, 358, 384]. These studies have been previously reviewed in Subsection (3.1.5).

However, manual configuration involves a trial-and-error approach and its efficacy is heavily contingent on the experience or intuition of the practitioner. In general, it could lead to suboptimal performance, hence it is now advised to be avoided. After all, the performance of an ESN depends on the complex interplay of all its hyperparameters, rather than from any single parameter in isolation. Hence optimization should consider the entire set of parameters rather than adjusting them in isolation. To this end, algorithmic optimization methods, rely on computational processes to systematically explore the parameter space, often leading to more efficient and effective results, especially in complex, high-dimensional settings.

Straightforward optimization approaches include Grid Search and Random Search. Grid Search performs an exhaustive search over a predefined set of hyperparameters by systematically evaluating all possible combinations. Although it guarantees finding the optimal solution within the search space, it can be computationally expensive. In contrast, Random Search randomly selects hyperparameter combinations. While it does not guarantee an optimal solution, it is typically more efficient than Grid Search, as it explores a wider range of configurations in less time. However, its effectiveness depends on the size and distribution of the search space.

The metaheuristics family is a popular class of optimization methods for ESNs. These methods, inspired by natural evolution, physical phenomena, or animal behaviors, provide simple frameworks that are easy to simulate, hybridize, and evolve [266]. As such, metaheuristics offer an effective alternative to traditional optimization techniques, particularly in complex, high-dimensional search spaces.

They iteratively explore the solution space through a heuristic approach. Starting with a randomly chosen initial solution, they evaluate neighboring solutions based on an optimization criterion, updating the best-found solution accordingly. This process continues until a termination condition, such as convergence or a predefined iteration limit, is met, with the best solution identified as the optimal solution for the problem at hand.

Metaheuristics avoid being stuck to local optima due to an effective exploration-exploitation trade-off. Exploration searches across the search space to identify promising regions, while exploitation focuses on refining solutions within these regions to converge on the optimal result. The search space of real problems is usually unknown and very complex with many local optima, so metaheuristics are good options for optimizing these challenging real problems, without requiring to search the entire space extensively. Meta-heuristics can be classified into three main classes: the evolutionary algorithms inspired by nature, the physics-based

ones inspired by physical principles and laws, and swarm intelligence algorithms that are based on group behaviors [266, 334].

The most popular algorithm among the evolutionary algorithms is the Genetic Algorithm (GA). Inspired by Dawrin's theory of evolution, GA was proposed by Holland [162] and is based on the concept of 'survival of the fittest'. GA is a population-based method, starting with the generation of a random population (set of solutions). A new population is then created through the natural processes: selection, crossover, and mutation. Since the best individuals have a higher probability of participating in the generation of the new population, the new population is likely to be better than the previous generation(s). This process ensures that the initial random population is optimized over the course of generations.

GAs effectively optimize ESN design and topology, enhancing both prediction accuracy and computational efficiency. Preliminary research by Xu et al. [415] explores the use of GAs to optimize the readout layer of ESNs, offering a direct adaptive approach that removes the need for system identification in modeling nonlinear dynamical systems. Shifting the focus on the internal structure of the reservoir in ESNs, the work of Ferreira and Ludermit [113] is reported as the first to explore ESN optimization using GA. Specifically, GA was employed to optimize five global parameters: (i) the number of nodes in the reservoir, (ii) the type of activation function used in the reservoir nodes (tanh or sigmoid), (iii) the spectral radius, (iv) the presence or absence of interconnections between the input and output layers, and (v) the presence or absence of feedback connections in the output layer. Tested on a wind speed forecasting task, GA demonstrated superior performance compared to an exhaustive search.

Later, Ferreira and Ludermit [114] introduced RCDESIGN, leveraging GA for ESN optimization. In addition to the parameters optimized in their earlier work [113], this study further optimized (i) the type of readout training method (pseudo-inverse or ridge regression), (ii) the regularization parameter for ridge regression, (iii) the leaking rate, and (iv) the bias unit connections. Evaluated on wind speed time series prediction, RCDESIGN exhibited lower computational cost compared to other benchmark optimization methods.

Building on this work, Ferreira et al. [115] further assessed RCDESIGN on benchmark datasets, including the NARMA system and the Mackey–Glass chaotic attractor time series. Once again, RCDESIGN outperformed the baseline methods. Notably, both Ferreira and Ludermit [114] and Ferreira et al. [115] challenged the widely held but incorrect belief that the spectral radius of the reservoir weight matrix must be constrained to values less than or equal to 1 to create good reservoir computers. Other GA applications on ESN optimization followed [52, 54, 87].

Notably, Løkse et al. [241] introduce a novel ESN framework that enhances generalization by incorporating regularization constraints through dimensionality reduction. The method employs PCA and kernelPCA to project the internal reservoir state at each time step into a lower-dimensional space, which is then processed by the readout for prediction. Tested on benchmark datasets (including the Mackey–Glass time series, NARMA system, and Multiple Superimposed Oscillator), this approach

demonstrates enhanced predictive performance. The lower-dimensional embedding better preserves the system's dynamical properties compared to using the full-dimensional internal states. Additionally, a GA was employed to optimize the ESN hyperparameters.

Ma et al. [248] propose a novel hierarchical framework called Deep-ESNs, for predicting time series with multiple scale structure. An input time series is projected into the high-dimensional echo-state space, followed by an encoding layer (e.g., PCA, autoencoder, or random projection) that maps it to a lower-dimensional space. These representations are then processed by another ESN. Alternating projection and encoding layers in a Deep-ESN address collinearity in the reservoir's internal states (which hinders pattern differentiation) and capture multiscale dynamics. Connections from each encoding layer are added to the final output layer to integrate the multiscale representations. GA is employed to optimize the leaking rate, spectral radius, and input scaling ESN hyperparameters. Similarly, Zhong et al. [440] develop the Double-Reservoir ESN (DRESN), with two reservoirs trying to model two input sequences. GA is employed to optimize the size and the spectral radius of both reservoirs. Lastly, Moon et al. [270] explore hierarchical reservoir structures, particularly Wide-ESNs that build independent sub-reservoirs in parallel, and Deep-ESNs that stack the sub-reservoirs in series. GA is used to optimize the leaking rate, spectral radius, and input scaling hyperparameters for each sub-reservoir.

The Swarm Intelligence algorithms mimic the social behavior of swarms, herds, flocks, or schools of creatures in nature, where the social intelligence of creatures is adopted. The most popular algorithm in this category is the Particle Swarm Optimization (PSO) [202], inspired by the social behavior of birds flocking. Similarly, Cuckoo Search (CS) [424] mimics the parasitic behavior of cuckoo birds in their nesting process. Cuckoos lay their eggs in the nests of other bird species, and when the eggs hatch, the cuckoo chicks often push out the host eggs to take over the nest. This process is used as a metaphor for the algorithm's search mechanism, where solutions are represented as eggs placed in nests (candidate solutions). The search space is explored through Lévy flight-based movements, with better solutions replacing worse ones, mimicking the cuckoo's behavior. This process continues iteratively, balancing exploration and exploitation to find an optimal solution. Less frequently utilized, the CS algorithm is employed by Bala et al. [33] to optimize ESN hyperparameters, spectral radius, reservoir size, and reservoir connectivity, for Mackey-Glass time series prediction. In an extension of this work, Bala et al. [35] develop a CS-based optimization algorithm for ESN to predict failures in turbofan engines. This approach outperforms three PSO methods, an earlier CS method, and the classical ESN model.

For a detailed review of ESN optimization methods, the reader is advised to consult the work of [347], which examines papers published from 2004 to 2021. Classifying optimization methods into bio-inspired and non-bio-inspired ones, the authors report that the bio-inspired ones produce greater results [347]. A comprehensive re-

view of metaheuristic applications in ESNs can be found in [34].

Another aspect that needs to be considered in ESN optimization is the reliance on one-sided cost functions, which focus solely on minimizing prediction errors. The most commonly used objective functions include Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Normalized RMSE (NRMSE) [347]. However, this rationale often neglects the reservoir's intrinsic properties [34]. Critical metrics such as Lyapunov exponents, the separation ratio, and eigenvalue spread are rarely considered. Incorporating these reservoir quality metrics into the optimization process, in conjunction with traditional predictive error minimization metrics, could lead to more comprehensive cost functions tailored to the reservoir's operational characteristics and capabilities. While minimizing prediction error remains the primary goal, considering reservoir quality as a secondary criterion can help ensure a well-structured state space, ultimately enhancing the robustness, stability, and generalizability of the model. This gap inspired the development of the SACGA ESN optimization method, which is presented and discussed in detail in the following section.

3.2 Self-Adaptive-Cuckoo-Genetic Algorithm – A Novel Optimization Algorithm

This section introduces the novel Self-Adaptive Cuckoo Genetic Algorithm (SACGA), an enhanced Genetic Algorithm developed for optimizing ESNs. We begin by discussing the motivation behind this work, followed by a theoretical foundation of Genetic Algorithms, self-adaptation, and Cuckoo Search. Next, we present the SACGA methodology in a step-by-step manner along with the corresponding algorithm. Finally, we discuss the expected advantages of this approach over existing methods.

3.2.1 Motivation for SACGA

Genetic Algorithms (GAs), at the heart of evolutionary computation, are significant optimization methods, particularly in machine learning and deep learning. This part introduces the Self-Adaptive Cuckoo Genetic Algorithm (SACGA), a novel GA variant developed to optimize the hyperparameters of Echo State Networks (ESNs) [174], a type of recurrent neural network within the Reservoir Computing (RC) framework.

Although ESNs have a straightforward design, their performance heavily relies on hyperparameter configuration, which determines both structure and functionality. While guidelines for hyperparameter selection exist [159, 244], optimal values are task-specific and dataset-dependent, necessitating careful tuning. This chal-

lenge is compounded by the black-box nature of ESNs.

GAs evolve a population of candidate solutions guided by an objective fitness function. Selection chooses the fittest individuals for reproduction, crossover combines advantageous traits, and mutation introduces randomness to enhance exploration and diversity. This iterative process continues until a stopping criterion is met. GAs embody Darwin's principle of 'survival of the fittest'. Early GA applications in RC include [113, 415], with later studies by [52, 54, 87, 241, 248, 380, 440]. However, most research, except [380], simplifies the search space to speed up convergence, by optimizing fewer hyperparameters or model topologies, as discussed in [34]. In this context, a more recent survey on high-order neural networks and evolutionary optimization in financial forecasting highlights that little research has focused on the optimization of DL neural network (DLNN) hyperparameters, in contrast to the more common practice of optimizing DLNN parameters [44]. Moreover, traditional approaches to GAs for RC optimization tend to focus solely on error minimization in their cost functions, often neglecting the inherent properties of the reservoir [34]. These observations highlight two key research gaps: the constrained search space and the limited scope of cost functions.

SACGA addresses these limitations by optimizing all relevant hyperparameters across the entire search space and incorporating the concept of separation [80] as a reservoir quality metric, in the cost function during fitness evaluation alongside the standard predictive error minimization criterion. The SACGA has a novel design that combines a self-adaptive mutation mechanism with Cuckoo-inspired offspring and traditional GA processes (selection, crossover, mutation). Self-adaptation enables dynamic adjustment of parameters such as mutation rates, while Cuckoo-inspired offspring, leveraging the *Lévy* flight mechanism, explore boundary regions of the search space. This design ensures a balance between exploration and exploitation.

3.2.2 Theoretical foundation

This part outlines the theoretical foundation of the SACGA. We begin by introducing the Genetic Algorithm, the core component of the method, followed by the Self-Adaptation feature that enhances its flexibility. Next, we describe the Cuckoo Search algorithm, which inspires a unique element to the approach.

Genetic Algorithm

The GA, originally conceived by Holland [162], is the most widely used technique within the broader category of evolutionary algorithms and draws inspiration from the principles of natural selection [29]. The key concept is that it operates by selecting the fittest candidate individuals to produce descendants for the next generation.

This process continues until a specific selection criterion is met, leading to the discovery of an optimal set of hyperparameters.

The GAs operate by applying a fitness function to evaluate a population of candidate individuals (solutions), followed by selection, crossover, and mutation processes. Initially, a population is generated randomly, and the GA operates as an iterative optimization loop. In each iteration, a selection process identifies certain individuals, termed parents, from the current population. These parents are then used to produce offspring, known as children, through a combination of crossover and mutation processes, typically occurring at a fixed, low probability rate. The fitness of the children is subsequently evaluated. This iterative procedure continues until a predefined stopping criterion, such as convergence or a set number of iterations, is satisfied. The key phases of this process are illustrated in Figure (3.2.1).

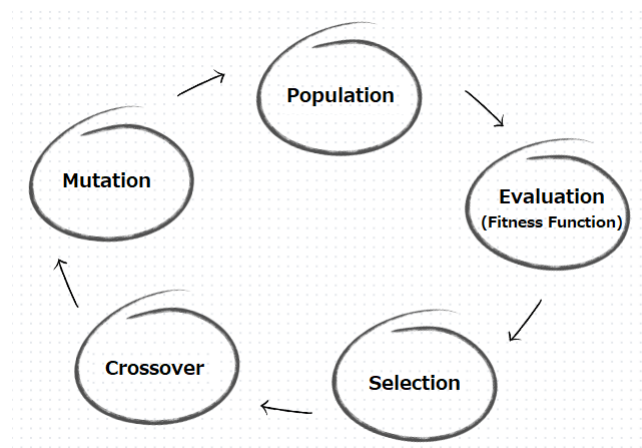


FIGURE 3.2.1: Flowchart of a GA Process, illustrating key steps: Population Initialization, Fitness Evaluation, Parent Selection, Crossover, and Mutation. This iterative process continues until a predefined stopping criterion is met.

The GA generates three types of children for the next generation. Elite children are the individuals with the best fitness scores in the current generation, automatically carrying over to the next. Crossover children are produced by exchanging traits or genes (hyperparameters) between a pair of parent individuals. For each trait, the algorithm selects which parent's genetic information (value) will be passed on to the child. Finally, mutation children are created by introducing random changes to one or more traits in an individual, exploring the search space further.

Elitism is employed to retain top-performing individuals, guaranteeing the survival of optimal solutions throughout successive generations. The algorithm introduces variability through mutation and crossover operations, where some individuals undergo random changes in their hyperparameters, while others merge the traits of two parents. Particularly, the mutation children contribute to the evolutionary part of the GA, introducing diversity in the population, while crossover contributes to the intensification feature. Figure (3.2.2) illustrates the three types of children in

GA.

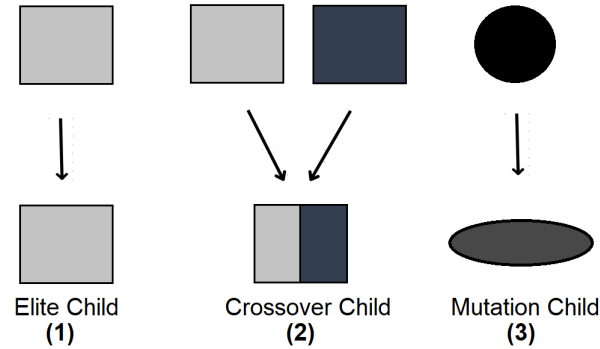


FIGURE 3.2.2: Illustration of the Three Types of Children in a GA Process: (1) Elite Children, who are the highest-performing individuals to survive in the next generation; (2) Crossover Children, produced by recombining traits (genes) from parent pairs; and (3) Mutation Children, generated by introducing random changes to traits, to enhance diversity and drive evolution.

GA provides a more efficient approach for hyperparameter optimization compared to traditional methods. Conventional techniques like grid search and random search are not suitable for tuning the hyperparameters of an RC system due to the increasing computational demands, associated with exploring the vast hyperparameters' search space. This challenge is exacerbated as the volume of the hyperparameters' search space grows exponentially with the number of hyperparameters. The curse of dimensionality renders these methods impractical in high-dimensional spaces [373, 430]. A more sophisticated approach, the Bayesian optimization method [47, 346] requires the tuning of covariance function parameters, which can be challenging and complex [203].

Another famous optimization algorithm is the PSO which is implemented to search for the maximum or minimum of a function defined on a multidimensional vector space. These represent the hyperparameter's bound values. Although PSO can be highly efficient, its suitability depends on the problem's complexity and the available computational resources. It is found that although PSO achieves high accuracy, it suffers from low computing speed [45]. Another research applies PSO for the adjustment of reservoir weights, however, due to high computational costs, only a subset of these weights is tuned [42].

Various Python libraries cater to diverse needs, providing options suitable for varying preferences and expertise levels regarding the implementation of GAs. Notable software packages in Python are described in Table (8).

Understanding the fundamental parameters of GAs is crucial for effectively harnessing their optimization capabilities. These parameters that ultimately define the GAs are the population size, crossover type, crossover rate, mutation rate, and elitist selection. The population size represents the number of individuals in each generation. The crossover type determines the recombination method of the parent traits.

Library	Description	Reference/URL
DEAP	Versatile Python library supporting various evolutionary algorithms, including GAs.	[85, 86, 120] (https://github.com/deap/deap)
PyEvolve	Python library offering a simple and flexible approach to working with GAs.	[296] (https://github.com/perone/Pyevolve)
Optuna	Primarily focused on Bayesian optimization, but also offers support for GA optimization.	[9] (https://github.com/optuna/optuna)
TPOT	Popular package for automated machine learning, including GA-based optimization of machine learning pipelines.	[283] (https://github.com/EpistasisLab/tpot)
PyGAD	Specifically designed for optimization using GA in Python, offering a simple interface appropriate for users with less experience in Python.	[122] (https://github.com/ahmedfgad/PyGAD)
EasyGA	Supports easy implementation of GA for optimization in Python.	(https://github.com/danielwilczak101/EasyGA)
LEAP	Recently published Python library for evolutionary algorithms, including GA support.	[76] (https://github.com/colettace/LEAP)
geneticalgorithm	Python package for easy implementation of GA optimization.	(https://github.com/rmsolgi/geneticalgorithm)

TABLE 8: An overview of Python libraries for GAs, featuring descriptions and associated URLs and references.

The crossover rate is the percentage of information recombination that occurs when two individuals are selected for recombination. If no crossover occurs, the individuals are directly copied to the next generation's population. The mutation rate is the probability of randomly flipping the value in each trait of each individual in every generation. Elitist selection, when activated, automatically carries the best individual(s) to the next generation. With elitist selection, the quality of the best solution consistently improves over time. Without elitist selection, there is a possibility of losing the best individuals due to stochastic errors [38].

The efficacy of the GA in optimization problems depends on the ability to converge to an optimum (local or global), once the region containing the optimum is identified, and the capability to explore new regions within the solution space to discover the global optimum [354]. The trade-off between these GA characteristics is determined by the values of the crossover rate, and mutation rate, along with the choice of the crossover technique [353].

Three types of crossover techniques exist. Firstly, the one-point crossover involves selecting a crossover point randomly and swapping the traits among the par-

ent individuals beyond that chosen point. Similarly, in the two-point crossover, a pair of crossover points is randomly chosen from the parent individuals. The traits of the parent individuals located between these two points are swapped. Lastly, the uniform crossover operates by randomly selecting each trait from either parent with an equal probability. This means that for each trait, there is an equal chance of the crossover child inheriting the trait from either parent [38].

Existing literature provides general guidelines for choosing values for crossover and mutation rates [83, 84, 145]. For a detailed review of the optimal crossover and mutation rates the reader is directed to [294].

A theoretical justification of the crossover and mutation operations' contribution in GA, on the grounds of disruption and construction, is demonstrated in [352]. The concepts of construction and disruption are tied to the exploration and exploitation characteristics of GA. Construction is associated with exploitation, creating and refining solutions, while disruption is linked to exploration, introducing random variations. The results indicate that in terms of disruption, mutation exhibits greater efficacy than crossover, albeit lacking the latter's ability to preserve common traits among individuals. Conversely, in terms of construction, crossover is more effective than mutation [352].

Nevertheless, the uncritical endorsement or dismissal of either crossover or mutation is undesired, as each operation plays a different role in the search process. Practitioners ought to consider that the optimal configuration of the crossover and mutation rates is contingent upon the unique features of the particular problem being addressed. On these grounds, another study suggests a second-level GA to determine the values of these parameters, as a distinct optimization problem, bearing the potential drawback of computational expenses [145].

In this study, the twin goal of exploration and exploitation can be efficiently achieved through the incorporation of self-adaptive mechanisms into the GA. This entails making dynamic adjustments to crossover and/or mutation rates, guided by the performance of each individual solution within the search space.

Self-Adaptation

The GA emanates from Darwin's theory of evolution. However, the static nature of its parameters hinders any adaptability to dynamic environments. This lack of versatility hampers the algorithm's ability to respond to changing conditions and thus affects its function in accurately converging to the global optimum in complex problems [153].

One promising avenue to address this limitation is self-adaptation. Self-adaptation is the property of a GA to autonomously adjust its parameters based on the evolving conditions of the optimization problem being solved. This adaptability can extend to several features of the GA including the crossover and mutation operations, and the population size. Several studies, such as [161, 166, 328, 354, 378, 402], explore the adaptation of crossover and mutation mechanisms in response to the perfor-

mance of individual solutions from the search space. The following provides a review of the most prominent research contributions, offering a panorama of the current literature in the field of self-adaptive GAs.

Fogarty [118], as cited in [294] is one of the seminal contributions on varying GA parameters. He studied the effects of different mutation rates across time and loci. The findings indicate that an exponentially decreasing mutation rate across generations improves the performance of GA for a single application [288, 354].

In the same year, Davis [82], as cited in [15, 165, 354], proposed adapting crossover and mutation probabilities based on the observed quality of the solution, favoring operators that generate superior performance.

The number and positions of crossover points are self-adapted by introducing special marks into the string representation, keeping track of the sites where crossover occurred, in [328]. Experiments show that adaptive crossover performs as well or better than a classical GA on a set of test problems. The adaptive crossover mechanism involves adjusting the distribution of crossover points based on the principles of 'survival-of-the-fittest' and 'recombination'. The proposed design incorporates special punctuation marks into the string representation to mark the 'site' of each crossover event. The rationale is that if particular crossover sites consistently lead to an inferior child, then they would fade away, taking their markings with them. Conversely, specific crossover sites consistently leading to superior outcomes would thrive and preserve their associated markings across generations. If there is no consistent relation to exploit, random allocation of markings is anticipated. This strategy aims to promote the persistence of successful crossover points in a string representation while allowing for flexibility in marking allocation where no consistent advantage is identified [328].

An adaptive mutation approach is adopted in [402]. This is integrated in GENITOR II, a distributed GA that builds upon the success of GENITOR [403]. One-at-a-time reproduction and rank-based allocation of reproductive opportunities are employed in this GA technique. At the core of this study is the implementation of adaptive mutation rates within the distributed algorithm. Results indicate that the adaptive mutation strategy, which dictates mutation rates vary based on the Hamming distance between the parent solutions, significantly improves search outcomes over static mutation settings.

Under a similar context, sustaining the diversity in the population without affecting the convergence properties, a self-organizing GA, known as the Adaptive Genetic Algorithm (AGA) is introduced in [354]. The AGA dynamically adjusts crossover and mutation rates based on the evaluation of individual solutions. High-fitness solutions are 'protected', by undergoing minimal mutation and being exempt from crossover operations. Conversely, solutions with sub-average performance are disrupted, with an assigned mutation rate of 0.5, fostering the creation of entirely new solutions [354].

The COst Based Operator Rate Adaptation (COBRA), a 'learning-rule' adaptation method is introduced and explored in [378]. COBRA periodically adjusts the operator probabilities according to a measure of the performance of each of the operators. The findings suggest that the choice for crossover probability depends on the nature of the problem, the population model, and the performance measure used. Further, the initial operator probabilities play a crucial role in influencing the overall GA performance. Lastly, while a direct impact of COBRA on performance improvement was not observed, the algorithm demonstrated a valuable characteristic of reducing the GA's sensitivity to operator probabilities, that is pruning the effect of bad choices. This potential makes COBRA a tool that can ease parameter tuning in certain applications. Yet cautious consideration is required, as it may lead to sub-optimal choices of operator probabilities in certain scenarios [378].

Cyclical varying mutation rates are explored at both parental and child levels within GAs in [161]. The findings suggest that cyclic-rate mutation, when compared to a constant mutation rate, tends to exhibit superior performance. Further, the results indicate that it doesn't significantly matter at which algorithmic locus (parental vs. child) the cyclic-rate mutation occurs. A plausible explanation for the observed performance advantage of the varying mutation rate over a constant one, is its periodic introduction of new genetic material into the population traits pool, thereby effectively maintaining diversity [161].

Another approach incorporating self-adaptation to the GAs, is the Dynamic Genetic Algorithm (DGA) [166]. DGA is introduced to tackle the challenge of automatically applying suitable crossover and mutation operators to generate a child to solve arbitrary problems [166]. The DGA utilizes multiple crossover and mutation operators to generate the next generation, and adjusts the ratios of crossover and mutation in response to the evaluation outcomes of the respective child. This adaptive approach aims to allow the really good operators to contribute more to the genetic procedure. Experimental results demonstrate that the proposed algorithm outperforms algorithms with a single crossover and a single mutation operator [166].

Further, it is important to bring to attention that, self-adaptation is situated within the broader category of Evolution Strategies (ES), aligning with GAs in drawing inspiration from natural evolution. Additionally, Evolutionary Programming (EP) is dedicated to the evolution of algorithms for effective problem-solving, aiming to optimize computational processes.

GAs, ES, and EP collectively fall under the general term of Evolutionary Computation (EC), as they emulate the underlying mechanisms of biological evolution, despite relatively minor differences. A particularly rigorous and invaluable resource in this domain is the book by Back [27], as cited in [119], standing out as one of the most comprehensive summaries of the EC field. It excels in providing clear insights and valuable information, covering both technical support and historical analysis fronts.

In closing, self-adaptation within GAs offers a promising avenue for enhancing adaptability to dynamic environments. The studies discussed, including various strategies of adaptive crossover and mutation mechanisms, accentuate the significance of dynamic parameter adjustments in GAs. These adaptations strive to enhance the probability of capturing the global optimum, elevate GA performance, and alleviate users from the complexities associated with defining the parameters' values within the GA.

Cuckoo Search Algorithm

The Cuckoo Search (CS) algorithm, introduced by Yang and Deb [424], is a recent addition to metaheuristic algorithms. It is inspired by the brood parasitic behavior of certain cuckoo species and the Lévy flight behavior observed in some birds and fruit flies.

Cuckoo birds or Cuckoos are known for their aggressive breeding strategy. This type of breeding behavior includes two approaches. Some species opt for communal nests, where they might remove others' eggs to enhance the likelihood of their own eggs hatching. Others engage in obligate brood parasitism, laying eggs in the nests of different host birds, employing tactics such as intraspecific brood parasitism, cooperative breeding, or nest takeover.

Host birds may respond with direct conflict, either discarding alien eggs or abandoning the nest to construct a new one elsewhere. However, some species of cuckoos have evolved specialized mimicry properties to reduce the risk of egg abandonment [129]. Furthermore, the timing of egg-laying is crucial. Parasitic cuckoos often select nests where the host bird has recently laid its eggs. Typically, the cuckoo eggs hatch slightly earlier than the host eggs, expelling the host eggs from the nest. In situations where the host bird's eggs hatch earlier than the cuckoo egg, the cuckoo chick, with its larger body size, pushes other chicks and eats more. After a few days, the host bird's own chicks perish from hunger, leaving only the cuckoo chick in the nest [304]. Remarkably, a cuckoo chick can also mimic the calls of host chicks to gain access to more feeding opportunities [129].

Relating this strategy to the CS, the algorithm mirrors the cuckoo's behavior by depositing solutions in different nests, representing potential solutions to an optimization problem. The algorithm uses a selection mechanism to identify high-quality solutions, akin to the successful hatching of cuckoo eggs. If a solution is deemed unfit, it is discarded or replaced, mimicking the host bird's response. The algorithm's success lies in its ability to optimize and evolve solutions over iterations, drawing inspiration from the adaptive and reproductive strategies observed in cuckoos.

This emulation of cuckoo behavior in the CS algorithm is grounded in three fundamental rules [92, 129, 333]. Firstly, only one random egg (solution) is deposited at a time to the population, mimicking a cuckoo choosing a nest at random. Secondly, nests containing high-quality eggs (solutions) have a greater likelihood of persis-

tence, aligning with the algorithm's selection mechanism. In the context of the population, solutions deemed high-quality are more likely to persist and undergo further optimization, analogous to nests with quality eggs in the cuckoo's world. Lastly, the number of host nests remains constant, and the host bird (algorithm) can detect the cuckoo egg (solution) based on a probability denoted by p in the range $[0, 1]$. This corresponds to the algorithm's evaluation process, where the fitness or quality of a solution is assessed. Upon detection of a suboptimal solution, the host bird (algorithm) may respond by discarding the egg (solution), abandoning the nest (replacing the solution), or constructing a new nest (exploring new solution spaces).

An integral part of the CS algorithm involves the Lévy flight concept. In nature, animals seek for food in a random manner. The foraging path of an animal is effectively a Random Walk, where the next move is determined based on the current location and the transition probability to the next location. Lévy-flight-style behavior is present in search patterns of fruit flies exploring their landscape [309], and hunter-gatherer foraging patterns in human behavior [41].

The CS algorithm integrates Lévy flight as a Random Walk mechanism to generate new solutions, exploring an extensive solution space. It starts with the generation of an initial population of host nests, each representing a potential solution to the optimization problem. The algorithm iteratively explores the search space by selecting cuckoos randomly and updating their positions through Lévy flights, as given by

$$x_i^{t+1} = x_i^t + \xi \oplus \text{Lévy}(\alpha) \quad (3.13)$$

where x_i^{t+1} represents the updated position of cuckoo i , and ξ is the step length drawn from a Lévy distribution parameterized by α , which acts as a scale factor controlling the trade-off between exploration (long jumps) and exploitation (short steps). The operator $\oplus$ denotes entry-wise multiplication. The Lévy distribution is characterized by heavy tails, meaning it has a higher probability of generating large values compared to a Gaussian distribution. This heavy-tailed nature allows for long jumps during the position update, promoting global exploration in the search space. A Lévy flight is essentially a Random Walk with step lengths sampled from a heavy-tailed distribution.

Following each positional update, the fitness of the cuckoo is assessed based on the specific optimization problem at hand. Through a process of random selection and fitness-based replacement of the solutions, nests are continually evolved, with superior solutions being retained. Additionally, a fraction of less fit nests (solutions) is abandoned, and new ones are introduced via Lévy flights. The algorithm proceeds through iterations until a termination criterion, such as a maximum number of generations, is met [129].

3.2.3 Methodology of SACGA

Overview of SACGA

In this section, we describe the architecture of the proposed optimization algorithm. The SACGA is built upon a conventional GA, [162], that incorporates deterministic elitism with replacement, uniform crossover, and an adaptive mutation mechanism as its core components. Additionally, a Cuckoo offspring, inspired by the Cuckoo Search (CS) Algorithm, is periodically introduced into the population, generated through the *Lévy* flight mechanism. This offspring is a candidate solution influenced by the exploratory behavior of cuckoos, as described in the next section. To avoid confusion, the term "Cuckoo" in the name of the proposed SACGA refers only to the Cuckoo-inspired offspring, and not to the Cuckoo Search algorithm itself. The proposed algorithm is fundamentally a GA.

The self-adaptive (SA) feature is incorporated in the mutation rate assignment scheme. Adopting measures for both error minimization and reservoir quality in the fitness evaluation is an important innovation and significantly improves the optimization process. The proposed SACGA is illustrated in Figure (3.2.3).

Our proposed self-adaptive GA, enhanced with Cuckoo-inspired (*Lévy* flight) offspring generation, bears conceptual similarity to the AGCS method introduced by Katha and Pabboju [201], which combines an Adaptive GA and Cuckoo Search to train ANNs. However, unlike AGCS, where the GA and CS components operate separately and their outputs are later merged, our approach integrates the *Lévy* flight mechanism directly into a unified evolutionary process. Furthermore, while AGCS is applied to training the weights of conventional ANNs, our method targets the optimization of ESN hyperparameters, which define the network's overall structure. Special emphasis is placed on the reservoir's fixed, randomly initialized architecture, whose quality is critical for achieving stable and reliable predictive performance.

The proposed SACGA operates as follows:

1. *Initialization*: Create an initial population of individuals, leaving room for the addition of the Cuckoo offspring, each representing a potential solution to the optimization problem. These individuals have randomly assigned values from the search space.
2. *Cuckoo Offspring Generation*: Generate a Cuckoo offspring using the *Lévy* flight mechanism to explore a broad spectrum of the solution space, incorporating values from the edges of the defined search boundaries.
3. *Fitness Evaluation*: Evaluate each individual's fitness in the ESN system, as each individual uniquely defines the system and thereby influences its predictive performance, by minimizing prediction error (RMSE) and considering reservoir quality in terms of separation.
4. *Genetic Operations*: Selection, crossover, and mutation are performed as in a conventional GA procedure, and a new population of individuals (new generation) is created. This part embodies the principles of natural selection

3.2. Self-Adaptive-Cuckoo-Genetic Algorithm – A Novel Optimization Algorithm

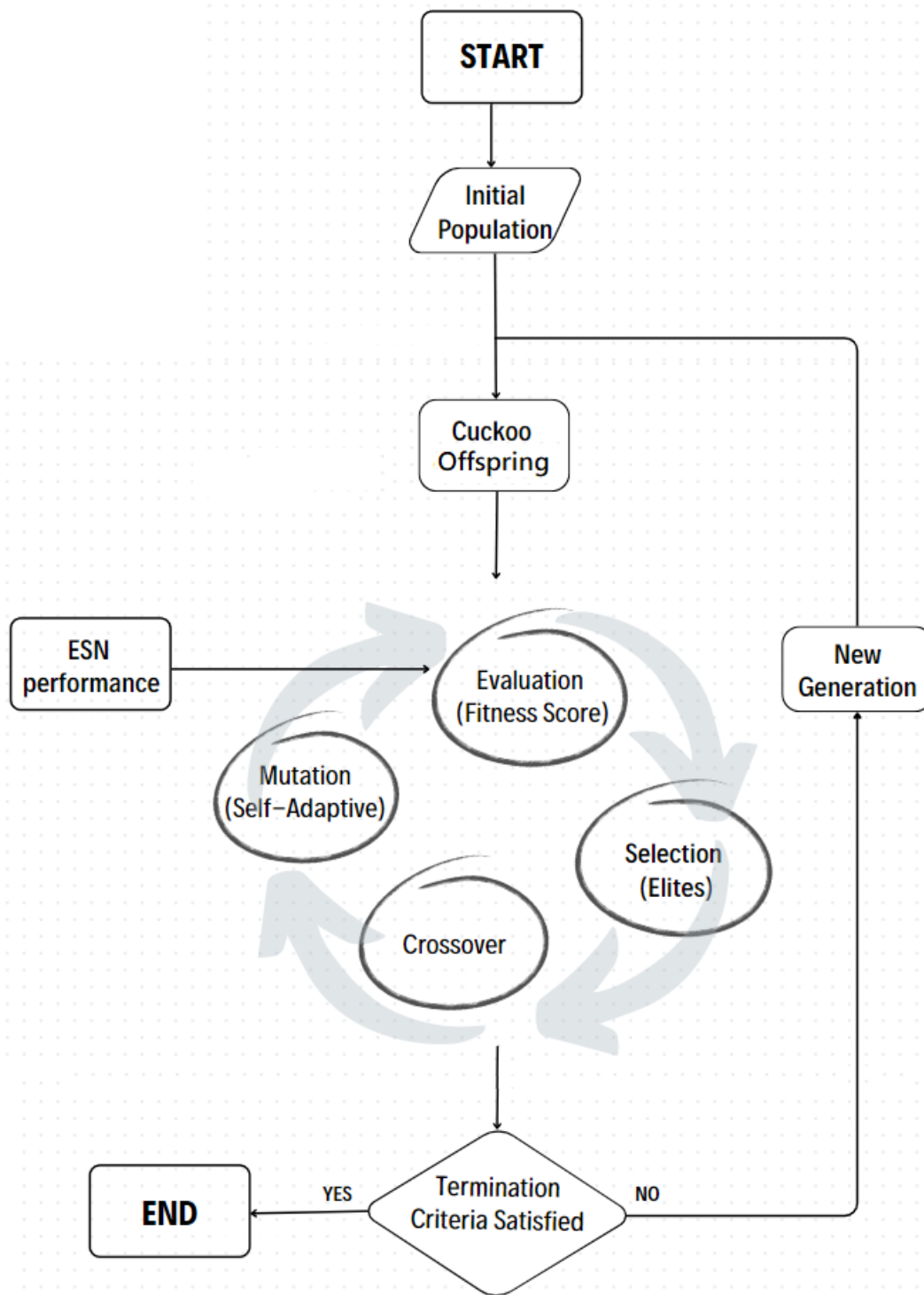


FIGURE 3.2.3: Flowchart of the proposed Self-Adaptive Cuckoo Genetic Algorithm (SACGA). This framework builds on a conventional Genetic Algorithm with elitism, crossover, and a Self-Adaptive mutation mechanism that dynamically adjusts mutation rates based on fitness scores and reservoir quality, ensuring a balanced exploration–exploitation trade-off. To further enhance exploration, a Cuckoo-inspired offspring generated via *Lévy* flights is periodically introduced into the population.

and genetic recombination. The self-adaptation feature is integrated into the mutation process by assigning varying mutation rates according to the performance ('fit') of each individual.

5. *Population Update*: The new generation, including elite individuals, offsprings from genetic operations (crossover and mutation), as well as a new Cuckoo offspring, replaces the current generation.
6. *Iteration*: Repeat steps 2-5 until the termination criterion (user-defined number of generations) is met.
7. *Optimal Individual Identification*: Select the individual with the lowest global predictive error (RMSE).

Cuckoo Offspring

The *Cuckoo Offspring* serves as an essential candidate solution in the SACGA model, introducing new genetic material through the *Lévy* flight mechanism. It is generated periodically using CS-inspired mechanisms, with its values positioned near the boundaries of the search space to enhance exploration.

The CS algorithm (Yang and Deb [424]), is inspired by the brood parasitic behavior of certain cuckoo species and the *Lévy* flight behavior of birds and fruit flies. Cuckoos are known for their aggressive breeding strategies, including communal nesting and obligate brood parasitism, where they lay their eggs in the nests of host birds, often ejecting the host's eggs or outcompeting the host chicks for food. This adaptive behavior is mirrored in the CS algorithm, which deposits candidates in various "nests", representing potential solutions, and employs a selection mechanism to identify high-quality candidates. Unfit candidates are discarded or replaced, akin to a host bird's response, allowing the algorithm to optimize and evolve candidates through iterations, effectively emulating the reproductive strategies of cuckoos.

To simulate the exploratory behavior of cuckoos, *Lévy* flight steps are generated for each hyperparameter of the cuckoo offspring. These steps are calculated using the *Lévy* flight distribution, which is known for its heavy-tailed nature. Specifically, for each hyperparameter c , a *Lévy* step Δc is generated as:

$$\Delta c = \text{levyFlight}(\alpha) \times \xi \quad (3.14)$$

where $\text{levyFlight}(\alpha)$ produces a step size according to the *Lévy* distribution parameterized by $\alpha=0.1$, and $\xi \sim \text{Uniform}(-1, 1)$ randomly determines the direction of the step, either positive or negative. The hyperparameters are then updated by adding these *Lévy* steps to their current values while ensuring that they remain within the predefined ranges of each hyperparameter's search space. The integration of a Cuckoo offspring aims to diversify the population and potentially uncover better solutions that might not be reachable through standard mutation and crossover operations alone.

Separation Metric

Separation measures how effectively a reservoir distinguishes between different classes of input data. It quantifies how well the system's reservoir states discriminate between distinct classes of inputs. Initially introduced by [141], this concept has been further explored in [75, 136, 281].

Strictly speaking, the concept of separation as a measure of reservoir quality is typically less applicable in cases of a continuous field of values, where the input signals are not inherently categorized into discrete classes [136]. To overcome this, the Separation Ratio Graph (SRG) unifies the separation with the Lyapunov Exponents, a measure of the chaotic nature of the reservoir [136]. In Valencia et al. [380], SRG is applied post-GA to select the most ideal reservoir from multiple ESN initializations with optimized hyperparameters, a method termed ESN-GA-SRG. However, in our work, since we integrate a reservoir quality measure directly within the SACGA framework, relying on visual interpretations of reservoir characteristics during the evaluation process would be unrealistic.

Extending the concept of separation, to cases without predefined classes allows the use of the separation concept as a reservoir quality metric. The basic assumption being that significantly different input signals should generate significantly different reservoir states, and hence result in different outputs [80, 136, 219]. It reflects how effectively a reservoir distinguishes between different input signals.

As discussed in subsection 3.1.7, and based on the idea that reservoirs perform best when operating near the edge of chaos, it is essential that Lyapunov Exponent values approximate 1 and Separation Ratio values be at least 1 for an ideal reservoir. The latter reflects the proportional distance between reservoir states and input distances [136, 380]. However, chaotic or complex datasets may require a broader range of separation ratio values. Preliminary experiments showed that a separation ratio between 1 and 10 is an effective indicator of reservoir performance in the context of complex, nonlinear, and dynamic time series. This supports controlled chaotic behavior, enabling the reservoir to manage nonlinearity and input separation effectively without causing instability, a condition that aligns with the requirement of Lyapunov Exponents being near 1, further reinforcing the reservoir's operation within the edge of chaos regime. A plausible interpretation is that amplifying the distances between reservoir states relative to input distances enhances the system's ability to distinguish between different inputs in the higher-dimensional reservoir state space.

GA and Self-Adaptive Mutation Mechanism

The GA operates through selection, crossover, and mutation processes. During selection, elite individuals are selected based on their superior performance in predictive error minimization (RMSE) and reservoir quality (separation ratios). An individual is considered elite if its mean separation ratio is between 1 and 10 and its fitness score ranks in the top 10% of the population. If this yields fewer than two elites, the selection disregards the reservoir quality metric, and is based solely on

error minimization, where the top performers are selected. Elite individuals have a zero mutation rate in order to preserve their traits and ensure their genetic material survives to the next generations.

The GA's effectiveness in optimization relies on its ability to converge to an optimum (local or global) after identifying the optimum's region (exploitation), as well as its capability to explore new areas of the solution space for global optimum discovery (exploration) [354]. The balance between exploitation and exploration is influenced by the crossover rate, mutation rate, and the crossover technique [353].

The *Self-Adaptive* feature is incorporated to assign mutation rates, promoting a balance between exploitation and exploration. Self-adaptation allows GA to autonomously adjust its parameters in response to the changing conditions of the optimization problem. This adaptability primarily occurs in crossover and mutation operations, as seen [161, 166, 328, 354, 378, 402]. A key innovation in this work is the self-adaptive mutation rate assignment, which accounts for both predictive error minimization and reservoir quality. Mutation rates are dynamically assigned to each individual based on their performance, as detailed in Figure (3.2.4). Individuals with lower mutation rates (e.g., 0.1) assigned are more prone to crossover, promoting exploitation by combining their traits with those of elites. In contrast, individuals with higher mutation rates (e.g., 0.8 or 0.95) are more likely to mutate, promoting exploration by significantly altering their genetic material. After all, larger mutation rates encourage exploration by producing offsprings that significantly differ from their parent, leading to a broader, undirected search [134].

After assigning mutation rates, the algorithm continues with mutation and crossover. For each non-elite individual, a random value determines whether crossover or mutation occurs. If the value exceeds the assigned mutation rate, the individual undergoes uniform crossover with a randomly selected elite, where each gene has an equal chance of being inherited from either parent. This approach enhances the individual by exploiting the superior traits of elites. Conversely, if the random value is below the mutation rate, mutation occurs, with the number and specific genes to mutate selected randomly. This process introduces diversity while retaining high-quality traits.

The new population, initially formed by the elite individuals, is then augmented by the crossover and mutated offsprings, forming a new generation of individuals. Further, the worst-performing individual (based on error minimization) is discarded to accommodate a Cuckoo-inspired offspring introduced in the next generation.

Fitness Evaluation Process

In the *Fitness Evaluation* phase, each individual in the population is assessed based on two criteria: predictive error minimization and reservoir quality. Predictive error is measured using the Root Mean Square Error (RMSE) from the ESN model's one-step-ahead predictions, calculated as follows:

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

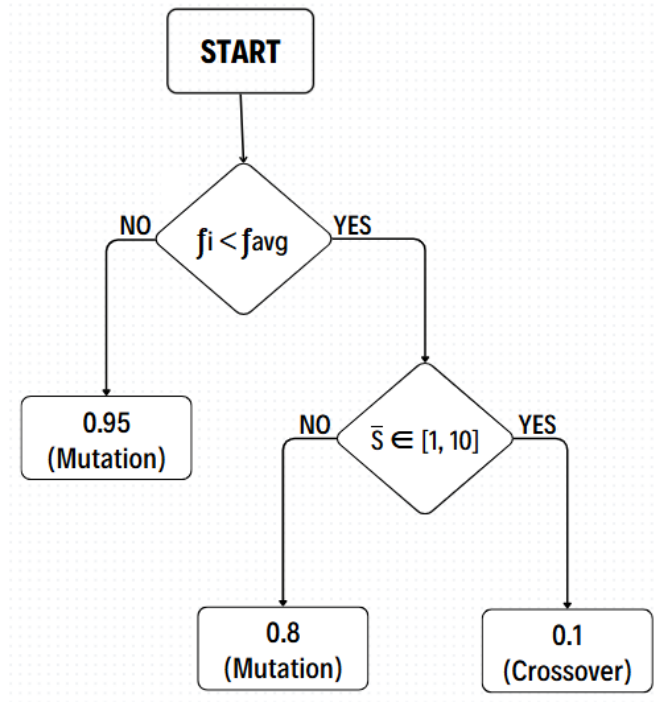


FIGURE 3.2.4: Flowchart for Self-Adaptive Mutation Rate Assignment: The SACGA model employs dynamic mutation rates to balance exploration and exploitation based on predictive error minimization (f_i) and reservoir quality ($\bar{S}$). If f_i exceeds the average predictive error (f_{avg}), a mutation rate of 0.95 is assigned. If f_i is below f_{avg} and $\bar{S}$ is between 1 and 10, the mutation rate is 0.1; otherwise, it is set to 0.8. Lower mutation rates encourage exploitation through crossover with elites, while higher mutation rates promote exploration through mutations.

where n is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value of the i_{th} observation. RMSE is one of the most widely used performance evaluation metrics [44].

Reservoir quality is measured by the separation ratio, reflecting how well the reservoir differentiates between input stimuli. The separation ratio compares the Euclidean distances between input vectors and their corresponding reservoir states, calculated for each input-reservoir pair as follows:

$$S_{ij} = \frac{D_{\text{reservoir}}(\mathbf{r}_i, \mathbf{r}_j)}{D_{\text{input}}(\mathbf{u}_i, \mathbf{u}_j)}$$

where $D_{\text{reservoir}}(\mathbf{r}_i, \mathbf{r}_j)$ is the distance between the reservoir states $\mathbf{r}_i$ and $\mathbf{r}_j$, and $D_{\text{input}}(\mathbf{u}_i, \mathbf{u}_j)$ is the distance between the input vectors $\mathbf{u}_i$ and $\mathbf{u}_j$. The separation ratio mean, $\bar{S}$, is given by:

$$\bar{S} = \frac{1}{N} \sum_{(i,j)} S_{ij}$$

where N is the number of pairs (i, j) .

Integration with Grid Search

In the ESN readout layer, the regularization parameter α plays a key role in controlling model training and fitting. A higher value of α simplifies the model, reducing

the risk of overfitting, but may also compromise predictive accuracy.

A *Grid Search* is used in the ESN system to optimize the Readout layer's regularization parameter α through Ridge Regression (Tikhonov regularization) [98]. This method balances model fit and coefficient size, reducing overfitting and improving noise tolerance [34, 133]. Grid Search systematically tests configurations to maximize the R-squared (R^2) score. The R^2 value ranging between zero and one measures the proportion of variance in the dependent variable (target signal) explained by the regression [81]. Integrating Grid Search with the SACGA model leverages their strengths, capturing their combined benefits for efficient and holistic optimization.

Computational Complexity of SACGA Compared to Genetic Algorithms and Cuckoo Search

Excluding the ESN-specific fitness evaluations, the computational complexity of the proposed SACGA method can be analyzed in terms of the number of generations G , the population size N (number of individuals), the problem dimensionality d (number of hyperparameters), and the number of Cuckoo-inspired offspring k introduced per generation (in our case $k = 1$). For a standard GA, the computational cost per generation is dominated by the evaluation and update of all individuals across all hyperparameters, along with the selection or elitism step typically involving a ranking operation of $O(N \log N)$. Hence, the overall complexity of the GA can be expressed as

$$\text{Cost}_{GA} = O(G \cdot (N \cdot d + N \log N)) \approx O(G \cdot N \cdot d).$$

This simplification holds because, when $d \geq \log N$, the ranking cost ($N \log N$) grows more slowly than the evaluation cost (Nd), and thus becomes asymptotically negligible.

The proposed SACGA extends the standard GA framework by introducing self-adaptive mutation rate assignment and the insertion of k Cuckoo-inspired offspring(s) per generation. These additional operations introduce an extra computational cost of $O(k \cdot d)$ per generation. Accordingly, the total computational complexity of SACGA can be written as

$$\text{Cost}_{SACGA} = O(G \cdot (N \cdot d + N \log N + k \cdot d)).$$

Given that $k = 1$ and $\log N \ll d$ for typical problem sizes, the leading term remains $O(G \cdot N \cdot d)$. This confirms that SACGA preserves the same asymptotic complexity class as the standard GA, although it introduces slightly larger constant factors due to the additional self-adaptation and offspring generation steps.

For comparison, the standard CS algorithm, which performs *Lévy*-flight updates for N nests and selection or replacement operations in each generation, has a complexity of

$$\text{Cost}_{CS} = O(G \cdot (N \cdot d + N \log N)) \approx O(G \cdot N \cdot d).$$

In summary, while SACGA introduces additional operations for self-adaptation and Cuckoo-inspired exploration, all three algorithms, namely GA, SACGA, and Cuckoo Search, share the same leading-order computational complexity of $O(G \cdot N \cdot d)$, with SACGA incurring slightly higher constant factors due to the extra operations.

3.2.4 Algorithm

For clarity and reproducibility, the complete pseudo-code of the proposed SACGA is provided in Algorithm 3.

3.2.5 Expected advantages over existing approaches

The performance of any DL model is inherently dependent on the proper configuration of its hyperparameters, which fundamentally determine the model's internal behavior and, consequently, its predictive capabilities. This issue is especially pronounced in models with intricate internal dynamics that defy intuitive interpretation, commonly referred to as "black-box" models. A prime example is the ESN [174]. While the ESN is praised for its computational efficiency and simplicity, its internal reservoir dynamics remain largely opaque, making hyperparameter tuning a necessary task.

In this context, the SACGA method offers several anticipated advantages on ESN optimization, over traditional and contemporary approaches.

To begin with, faster convergence to optimal solutions is expected. By integrating the exploratory capabilities of Cuckoo-inspired offsprings through *Lévy* flights with the adaptability of a self-adaptive mutation rate mechanism and the global search strengths of GAs, SACGA achieves a more balanced exploration–exploitation trade-off. This synergy is particularly beneficial in navigating the complex, high-dimensional search spaces of ESN hyperparameter optimization. Unlike conventional GAs, this hybrid approach enhances population diversity and facilitates escape from local optima, thereby accelerating convergence to high-quality solutions.

Furthermore, the method is designed to promote reliable and stable predictions. This is achieved through a dual-objective optimization strategy, where predictive error minimization serves as the primary criterion and the separation ratio, a measure of reservoir quality, acts as a secondary guiding factor. This dual focus ensures that the search process remains dynamic while being effectively steered toward high-potential regions of the search space. As a result, the method not only improves predictive accuracy but also cultivates a well-structured reservoir state space. Such state space is crucial for maintaining the separability and richness of internal representations over time. It allows the network to differentiate between distinct input patterns, retain meaningful temporal dependencies, and prevent redundancy

or collapse in state dynamics. These qualities directly enhance the ESN's memory capacity and generalization ability, increasing robustness to noise and making it better suited for capturing complex, nonlinear temporal patterns.

The use of the separation ratio as a lightweight yet effective reservoir quality metric enables the algorithm to leverage the reservoir's internal dynamics without compromising computational efficiency. Its adoption aims to promote controlled chaotic behavior, thereby ensure that the reservoir exhibits sufficient complexity to capture the underlying trends in the input signal, while preserving system stability to remain input-responsive rather than erratic. This balance is expected to elucidate on the otherwise opaque nature of the reservoir by providing a measurable and interpretable indicator of reservoir quality. In doing so, it facilitates a deeper understanding of the ESN's internal workings and enhances model transparency and trustworthiness.

Finally, the proposed method is not a generic optimization framework applied in isolation, but one specifically tailored to the structural nuances of the ESN itself. By embedding reservoir-specific quality assessments directly into the fitness evaluation, SACGA moves beyond purely error-driven optimization. It actively cultivates reservoirs that are both expressive and stable, ultimately improving the model's generalization ability, overall reliability, and trust.

Algorithm 1 Echo State Network

- 1: **Input:** Input data $\mathbf{Z}$, target output $\mathbf{Y}$, hyperparameter tuple individual $\mathbf{i} = (lr, \rho, \omega, \gamma, N, s_{\text{res}}, s_{\text{in}})$
- 2: **Output:** Predicted values and associated performance metrics

- 3: **Initialization:**
 Number of non-zero reservoir connections: $n_{\text{Wres}} = (1 - s_{\text{res}}) \times N^2$, where N is the reservoir size
 Number of non-zero input connections: $n_{\text{Win}} = (1 - s_{\text{in}}) \times N \times d_{\text{in}}$, where d_{in} is the input dimension.
- 4: Create sparse random matrices:
 $\mathbf{W}_{\text{res}} \in \mathbb{R}^{N \times N}$ with n_{Wres} non-zero values drawn from $\mathcal{U}(-1, 1)$
 $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N \times d_{\text{in}}}$ with n_{Win} non-zero values drawn from $\mathcal{U}(-1, 1)$
- 5: Normalize reservoir weights: $\overline{\mathbf{W}}_{\text{res}} \leftarrow \frac{\mathbf{W}_{\text{res}}}{|\lambda_{\text{max}}|}$, where $|\lambda_{\text{max}}|$ is the largest among the absolute values of the eigenvalues of $\mathbf{W}_{\text{res}}$.
- 6: Normalize input weights: $\overline{\mathbf{W}}_{\text{in}} \leftarrow \frac{\mathbf{W}_{\text{in}}}{\|\mathbf{W}_{\text{in}}\|}$
- 7: **Training phase:**
- 8: Initialize reservoir state $\mathbf{x}_0 = \mathbf{0}$; empty list $\mathbf{X}_{\text{train}} \leftarrow []$
- 9: **for** each timestep t in training data **do**
- 10: Sample random noise $\boldsymbol{\zeta}_t \sim \mathcal{U}(-1, 1)^N$ and normalize: $\bar{\boldsymbol{\zeta}}_t \leftarrow \frac{\boldsymbol{\zeta}_t}{\|\boldsymbol{\zeta}_t\|}$
- 11: Update state:

$$\mathbf{x}_t = lr \cdot \mathbf{x}_{t-1} + (1 - lr) \cdot \tanh\left(\rho \cdot \overline{\mathbf{W}}_{\text{res}} \cdot \mathbf{x}_{t-1} + \gamma \cdot \overline{\mathbf{W}}_{\text{in}} \cdot \mathbf{z}_t + \omega \cdot \bar{\boldsymbol{\zeta}}_t\right)$$

- 12: Append $\mathbf{x}_t$ to reservoir states matrix $\mathbf{X}_{\text{train}}$
- 13: **end for**
- 14: Discard first k samples to remove initial transients (washout period), then:

$$\mathbf{X}_{\text{train}} \in \mathbb{R}^{T \times N} \quad \text{and} \quad \mathbf{Y}_{\text{train}} \in \mathbb{R}^{T \times d_{\text{in}}}$$

- 15: Use grid search to find optimal α in Ridge regression:

$$\mathbf{w}_{\text{out}} = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{X}_{\text{train}} \mathbf{w} - \mathbf{Y}_{\text{train}}\|^2 + \alpha \|\mathbf{w}\|^2$$

- 16: Train Ridge regression model on training data using the optimal α :

$$\mathbf{w}_{\text{out}} \leftarrow \text{Ridge}(\alpha) \text{ fitted to } (\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$$

- 17: **Testing phase:**
 - 18: Initialize previous reservoir state $\mathbf{x}_{\text{prev}} \leftarrow \mathbf{x}_{\text{train}}^{\text{last}}$
 - 19: **for** each timestep t in test data **do**
 - 20: Compute reservoir state $\mathbf{x}_t$ using $\mathbf{x}_{\text{prev}}$, test input, $\mathbf{W}_{\text{res}}$, and $\mathbf{W}_{\text{in}}$
 - 21: Predict output $\hat{y}_t = \mathbf{w}_{\text{out}}^\top \mathbf{x}_t$
 - 22: Update previous state $\mathbf{x}_{\text{prev}} \leftarrow \mathbf{x}_t$
 - 23: **end for**
 - 24: **Evaluate model performance:** Root Mean Squared Error (RMSE)
 - 25: **return** Predicted values and associated performance metrics
-

Algorithm 2 Estimation of the Largest Lyapunov Exponent (LLE)

- 1: **Input:** Reservoir state matrix $\mathbf{X} \in \mathbb{R}^{T \times D}$, where T is the number of time steps and D is the reservoir dimensionality
 - 2: **Output:** List of estimated Largest Lyapunov Exponents (LLEs)
-

3: **Step 1: Select reference points**

- 4: Partition the dataset into k strata
- 5: Perform stratified sampling to select a set of reference indices $\{r\}$

6: **for** each reference index r in the sampled set **do**7: **Step 2: Identify nearest neighbors**

- 8: Find the top 1% nearest neighbors $\{j\}$ of $\mathbf{x}_r$ using Euclidean distance in $\mathbb{R}^D$

9: **Step 3: Compute local divergence over time (Stretching Factor Computation)**

- 10: **for** each time lag τ in a predefined range (e.g., 1 to 100) such that $r + \tau < T$ and $j + \tau < T$ **do**

- 11: For each neighbor index j , compute the local stretching factor:

$$\sigma_{r,j}(\tau) = \log \left(\frac{\|\mathbf{x}_{r+\tau} - \mathbf{x}_{j+\tau}\|}{\|\mathbf{x}_r - \mathbf{x}_j\|} \right)$$

- 12: Average over all neighbors j to obtain the mean stretching factor:

$$\bar{\sigma}_r(\tau) = \frac{1}{|\{j\}|} \sum_j \sigma_{r,j}(\tau)$$

13: **end for**14: **Step 4: Estimate LLE from slope**

- 15: Perform linear regression on the pairs $(\tau, \bar{\sigma}_r(\tau))$
- 16: The slope of the fitted line gives the LLE estimate for reference r
- 17: Append this LLE estimate to the list of LLEs

18: **end for**

- 19: **Return:** List of LLE estimates, one for each reference trajectory
-

Algorithm 3 SACGA for Echo State Network Optimization

```

1: Input: Initial population  $P$ , population size  $N$ , generations  $G$ , Levy flight
   scaling  $\alpha$ , parameter ranges for leaking rate, spectral radius, omega, gamma,
   reservoir size, reservoir sparsity, input sparsity, dataset  $D$ , training size  $T$ , input
   dimension  $d$ 
2: Initialize: Best fitness score  $\leftarrow \infty$ , best individual  $\leftarrow \emptyset$ 
3: for each generation  $g$  in  $G$  total generations do
4:   Initialize empty lists for RMSE scores and separation ratios
5:   Generate cuckoo population  $C$  of size 1
6:     For each individual parameter in cuckoo:
7:       Sample randomly within respective parameter ranges
8:       Apply Levy flight step to each parameter (scaled by  $\alpha$ ), with random sign
9:       Adjust each hyperparameter to ensure values stay within predefined
       bounds
10:    Convert reservoir size to integer (if necessary)
11:     $P \leftarrow P \cup C$ 
12:    for each individual  $ind$  in population  $P$  do
13:      Train ESN using dataset  $D$ , training size  $T$ , input dimension  $d$ , and
      hyperparameters from  $ind$ 
14:      Compute RMSE on validation set as error minimization score
15:      Compute separation ratio of reservoir states to inputs, on validation set, as
      reservoir quality score
16:      Update best individual if RMSE improved
17:      Store RMSE score and separation ratio
18:    end for
19:    Compute average RMSE and 10th percentile RMSE (quantile 10)
20:    Select elites:
21:      Individuals with separation ratio  $\in [1, 10]$  and  $RMSE \leq$  quantile 10
22:    if number of elites  $< 2$  then
23:      Select top 2 individuals by RMSE as elites
24:    end if
25:    Assign mutation rates: elites  $\rightarrow 0$  to preserve for the next generation,
    others  $\rightarrow 0.1, 0.8$ , or  $0.95$  based on RMSE and separation ratio
26:    Initialize new population with elites
27:    for each non-elite individual do
28:      if random()  $<$  mutation rate then
29:        Mutate random subset of genes within valid hyperparameter ranges
30:      else
31:        Perform crossover with random elite to create child
32:      end if
33:      Add child to new population
34:    end for
35:    Remove worst individual, based on RMSE, from new population to allow
    the addition of the Cuckoo-inspired offspring in the next generation, while
    maintaining the population size
36:     $P \leftarrow$  new_population
37:  end for
38: Output: Best individual and best RMSE score found

```

Chapter 4

Findings and Discussion

4.1 Experimental Validation on Benchmark Datasets

This chapter applies SACGA to ESNs and compares its performance with other optimization methods on benchmark datasets. It begins with an overview of the experimental setup, followed by sections on benchmark datasets and comparative methods used. The next section focuses on performance analysis, followed by a discussion of key findings, and the chapter concludes with final remarks.

4.1.1 Experimental Design

The proposed methodology is tested on three benchmark datasets commonly used in time-series forecasting, namely the prediction of the Multiple Superimposed Oscillator series [406, 417], the Mackey-Glass system [253], and the NARMA system [175]. Each dataset undergoes an 80-20 split. The 80% of the data is used for hyperparameter optimization. The remaining 20% of the data, not involved in the hyperparameter tuning, is used for final model evaluation. Once hyperparameter optimization is completed and the optimal set of hyperparameters is determined, the ESN's predictive performance is evaluated. For each GA method, we predict values for the test set using 32 independently initialized ESN models, based on the identified optimal set of hyperparameters. Each ESN model employs the same optimized hyperparameters specific to its respective GA method. The final results represent the average performance across 32 ESN instances, as in [241].

Table (9) summarizes the parameter settings of the GA methods employed in this study, including the population size and number of generations, as well as the specific ranges of the ESN hyperparameters, specifying their minimum and maximum values.

There are no general rules for determining the population size in GA, however, small populations may lack diversity, leading to suboptimal performance [145]. Typically, population sizes range from 50 to 100 [317], a convention followed in this study. Research suggests that small population sizes when allowed to evolve for many generations, yield the most effective results [157]. Notably, it is shown

<i>Parameter</i>	<i>Value / Range</i>
<i>Population Size</i>	50
<i>Generations</i>	15
<i>Leaking Rate (lr)</i>	(0.001, 0.99)
<i>Spectral Radius (ρ)</i>	(0.001, 9.999)
<i>Omega (ω)</i>	(0.001, 9.999)
<i>Gamma (γ)</i>	(0.001, 9.999)
<i>Reservoir Size (N)</i>	(40, 500)
<i>Sparsity Reservoir (s_{res})</i>	(0.01, 0.999)
<i>Sparsity Input (s_{in})</i>	(0.01, 0.999)
<i>Readout Regularization Parameter (α)</i>	$[1e^{-11}, 1e^{-6}, 1e^{-3}, 0.1, 1, 10, 100]$

TABLE 9: GA Optimization Parameters and ESN Hyperparameter Search Ranges

that smaller populations combined with relatively higher mutation rates outperform larger populations with lower mutation rates, a configuration commonly used in electromagnetics and broader GA communities [157]. Thus, we argue that a population size of 50, evolving over 15 generations, in conjunction with a self-adaptive mechanism assigning mutation rates of 0.1, 0.8, or 0.95, is sufficient to achieve satisfactory results. This setup is not presented as a universal solution, but as a well-performing configuration for this specific application, considering the computational demands of the ESN reservoir. Although the foundational research by Haupt [157], is two decades old on electrical engineering problems, primarily focused on the relationship between population size and mutation rates in GA, its insights remain relevant to ESN optimization. ESNs, as the virtual paradigm of Reservoir Computing, inherit engineering principles that make the findings transferable. The proposed SACGA setup can effectively balance exploration and exploitation while reducing computational overhead, which is crucial in ESN optimization, where fitness evaluations are computationally intensive due to the complex dynamics of the ESN reservoir. To exemplify, large reservoirs with low sparsity can lead to dense matrices, requiring costly matrix multiplications, which highlights the importance of optimizing computational efficiency.

The performance of all methods is evaluated by three common metrics: root mean squared error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE), formulated as follows:

$$\begin{aligned}\text{RMSE} &= \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \\ \text{MAE} &= \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \\ \text{MAPE} &= \frac{100\%}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|\end{aligned}$$

where n is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value of the i^{th} observation. For these three metrics, smaller values mean better performance.

This analysis is conducted using Python (version 3.7.16) on a Linux operating system (kernel version 6.8.0-48-generic), with an Intel Core i7-7700 CPU (3 cores, 3.6 GHz). Since computational performance is closely tied to hardware specifications, these details provide context for the reported runtime results of the SACGA framework, which are analyzed in Section 4.1.4.

4.1.2 Description of Benchmark Datasets

Multiple Superimposed Oscillator (MSO) series is a synthetic time series dataset created by combining multiple oscillators with different frequencies. The MSO problem [406, 417] serves as a benchmark for time series nonlinear prediction tasks, whilst it is particularly challenging for ESNs, due to the coupling of reservoir neurons [179]. The MSO time series is generated by summing up several different frequency sine wave functions, as shown in:

$$y(t) = \sum_{i=1}^m \sin(\beta_i t), \quad (4.1)$$

where m is the number of summed sine waves, β_i their frequencies, and t is the time index. The frequencies of the sine waves are selected from the set: $\beta_1 = 0.2$, $\beta_2 = 0.311$, $\beta_3 = 0.42$, $\beta_4 = 0.51$, $\beta_5 = 0.63$, $\beta_6 = 0.74$, $\beta_7 = 0.85$, and $\beta_8 = 0.97$ [208]. In this study, a time series of 700 data points generated by summing 5 sine waves, aligns with the length chosen in [164, 208, 224, 330]. The use of 5 sine waves is also explored in [208, 224].

Mackey-Glass [253] system is a synthetic dataset, known for its chaotic, nonlinear behavior and is often used as a benchmark system in time series prediction studies [115, 178, 241, 248, 270]. It is characterized by the delay, τ . The system is defined by the following delayed differential equation:

$$\frac{dx}{dt} = \frac{\alpha x(t - \tau)}{1 + x(t - \tau)^n} - \beta x(t), \quad t \geq 0 \quad (4.2)$$

In this work, the Mackey-Glass system is simulated using standard parameters, that is, $\tau = 17$, $\alpha = 0.2$, $\beta = 0.1$, $n = 10$, with an initial condition $x(0) = 1.2$. The time step for the numerical integration is $dt = 1$. The Mackey-Glass system is simulated for 1000 time steps, generating a time series of 1,000 data points.

The **NARMA** (Nonlinear Auto-Regressive Moving Average) dataset [175] models the current output based on both the current input and past outputs. The nonlinearity and potentially extensive memory make modeling this system particularly challenging. NARMA time series is one of the most important and widely used benchmark datasets used in the ESN literature [115, 175, 241, 248, 270, 386]. The NARMA task in this analysis models the output of the tenth-order NARMA system (NARMA10), as described by the following equation:

$$y_{t+1} = 0.3y_t + 0.05y_t \sum_{i=0}^9 y_{t-i} + 1.5x_t x_{t-9} + 0.1, \quad t \geq 0 \quad (4.3)$$

where x_t is drawn from a uniform distribution over the interval $[0, 0.5]$. The NARMA10 dataset is composed of 2,000 data points.

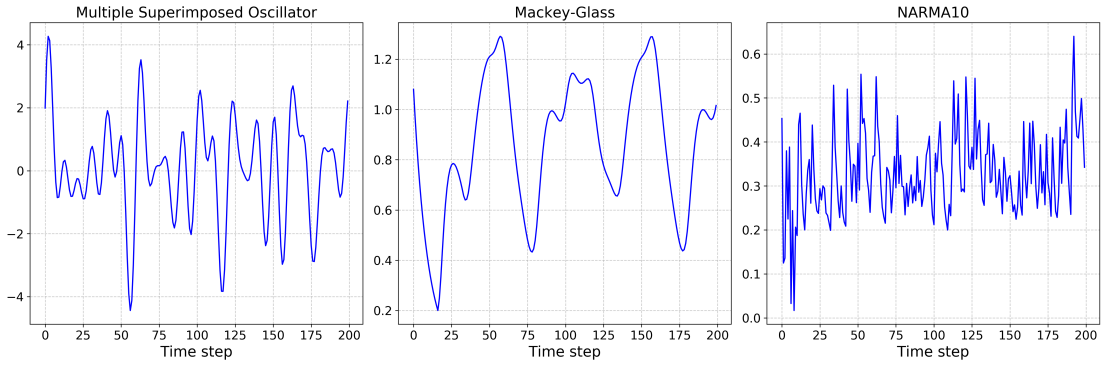


FIGURE 4.1.1: Snapshots of the MSO, Mackey-Glass, and NARMA10 time series datasets. The MSO dataset combines multiple sine waves of different frequencies, creating a complex oscillatory pattern. The Mackey-Glass series exhibits chaotic dynamics, and the NARMA10 dataset models a nonlinear autoregressive moving average process.

4.1.3 Comparative Methods

The performance of the proposed SACGA model is compared against fixed and deterministic mutation rate schemes.

The first model, referred to as a $GA(10\%)$, is a conventional GA framework with a fixed mutation rate at 10%. This mutation rate is used in similar texts, see for example [21, 150, 362]. The top 10% best-performing individuals are selected as elites and carried over unchanged to the next generation. For each non-elite individual, the algorithm applies either crossover or mutation. With a probability of 0.5, crossover is performed, where an offspring is generated by combining genes from a randomly selected elite and the individual itself using a uniform crossover method. Each gene is randomly inherited from either parent with equal probability. If crossover is not applied, mutation is performed with a 10% probability, whereby a randomly chosen subset of genes is modified. If neither crossover nor mutation occurs, the individual remains unchanged. The population size is fixed at 50 individuals, and the algorithm runs for 15 generations.

The second model, $GA(1\%)$, mirrors the first but operates with a mutation rate of 1%, while all other parameters remain the same. A mutation rate of 1% is employed in similar studies, see for example [142, 169, 232, 240, 308, 362].

The third approach utilizes a deterministic mutation rate scheme. In deterministic mutation control, the mutation rate changes according to some heuristic rule with no feedback from the search, typically in a time-varying manner [102]. In this study, we adopt the deterministically decreasing mutation scheme from Bäck and Schütz [28], where the mutation rate at generation t , denoted as p_t , is given by:

$$p_t = \left(2 + \frac{n-1}{T-1}t\right)^{-1} \quad (4.4)$$

Here, T represents the total number of generations, and n is the number of hyperparameters to be tuned in the GA, which in this case is $n=7$ ¹. For additional examples of deterministic mutation rate schemes, refer to Thierens [369].

The selection of benchmark methods is based on their relevance to the core structure of our proposed SACGA method. GAs are a well-established optimization technique widely used in ESN optimization [52, 54, 87, 113–115, 241, 248, 270, 415, 440], and since our approach is an enhanced GA, it is appropriate to compare it with other GA variants. Additionally, GAs with deterministic mutation schemes introduce controlled variability, in the mutation rates assignment, similar to the self-adaptation in our method, despite differences in feedback from the search.

The consistency of the experimental setup further supports the selection of these benchmarks, as they share the same algorithmic structure (e.g., population size, number of generations, and elite selection of the top 10% best-performing individuals). We chose these methods for their fundamental similarities to our optimization approach and their established effectiveness since their inception. This choice ensures a robust and meaningful comparison without the potential biases of newer, less-tested methods.

4.1.4 Performance Analysis

Figure (4.1.2) presents the convergence rates of the GA hyperparameter optimization methods across datasets, with RMSE Quantile 10 values for SACGA, GA with 10% and 1% mutation rates, and deterministic GA. As it is illustrated, the SACGA consistently outperforms the other methods throughout the generations, demonstrating faster convergence to lower error rates.

Upon completing the hyperparameter optimization task of the ESN model, the predictive accuracy is evaluated using RMSE, MAE, and MAPE metrics. Table (10)

¹Refers to the seven hyperparameters— *learning rate*, *spectral radius*, *omega*, *gamma*, *reservoir size*, *sparsity reservoir*, and *sparsity input*—optimized via GA methods. The regularization parameter is optimized through grid search within the GA.

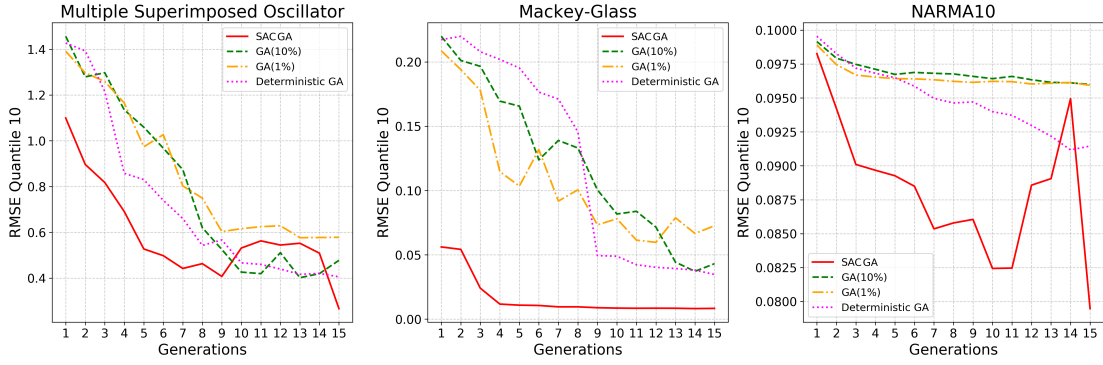


FIGURE 4.1.2: Convergence of RMSE Quantile 10 values across generations for the proposed SACGA, GA (mutation rates 10% and 1%), and deterministic GA on MSO (left), Mackey-Glass (center), and NARMA10 (right) time series. RMSE is plotted here as it guides the GA methods.

provides the average results and standard deviations. The proposed SACGA method yields the best results, achieving the lowest RMSE, MAE, and MAPE values across all datasets.

For the MSO dataset, SACGA achieves the lowest RMSE (0.272), MAE (0.214), and MAPE (102.811), significantly outperforming its counterparts, which show notably higher error metrics. For the Mackey-Glass dataset, the SACGA method once again proves its superiority, achieving exceptionally low RMSE (0.009), MAE (0.007), and MAPE (0.885). These indicate an exceptional ability to model complex nonlinear behavior. On the NARMA10 dataset, the SACGA continues to outperform other GA methods, with the lowest RMSE (0.078), MAE (0.063), and MAPE (17.691), demonstrating the best overall performance.

Compared to traditional GA variants, the SACGA shows significant improvement in predictive accuracy. Notably, the SACGA achieves consistently lower variation in error metrics (RMSE, MAE, and MAPE), indicating reliable and stable predictive performance. This is visually supported in Figure (4.1.3). The MAPE distributions for the ESN's predictive performance (averaged over 32 instances) of the GA methods employed, across the three benchmark datasets, show the superiority of the SACGA method. It consistently achieves the lowest MAPE values with reduced error variation, indicating reliable predictive accuracy and stability in model optimization. The substantial gap in performance between the SACGA and other GA approaches underscores its robustness. This is further illustrated in Figure (4.1.4), showing the ESN system outputs (averaged over 32 instances) against the target values, and the corresponding absolute errors.

The computational time of the SACGA depends on parameters such as population size, number of generations, fitness function complexity (including predictive error and separation ratio computations), and the integration of Cuckoo-inspired offspring. The latter, particularly when affecting reservoir size, can significantly increase computational complexity, as a larger number of neurons results in a larger weight matrix. This effect is amplified when low sparsity values are used. The average computational time of the SACGA across datasets is calculated using the following equation [114]:

4.1. Experimental Validation on Benchmark Datasets

MSO	RMSE	MAE	MAPE
SACGA	0.272 ± 0.0105	0.214 ± 0.0095	102.811 ± 15.0743
GA(10%)	0.563 ± 0.1295	0.415 ± 0.0805	156.739 ± 21.8954
GA(1%)	0.553 ± 0.0279	0.422 ± 0.0199	161.911 ± 19.9254
Deterministic GA	0.460 ± 0.0434	0.351 ± 0.0316	141.200 ± 24.7926
Mackey-Glass	RMSE	MAE	MAPE
SACGA	0.009 ± 0.0003	0.007 ± 0.0003	0.885 ± 0.0316
GA(10%)	0.046 ± 0.0018	0.038 ± 0.0016	4.671 ± 0.2160
GA(1%)	0.063 ± 0.0036	0.051 ± 0.0030	6.120 ± 0.4128
Deterministic GA	0.019 ± 0.0016	0.015 ± 0.0013	1.764 ± 0.1538
NARMA10	RMSE	MAE	MAPE
SACGA	0.078 ± 0.0027	0.063 ± 0.0025	17.691 ± 0.7797
GA(10%)	0.101 ± 0.0069	0.081 ± 0.0057	23.529 ± 2.1021
GA(1%)	0.095 ± 0.0019	0.077 ± 0.0014	22.061 ± 0.3923
Deterministic GA	0.098 ± 0.0026	0.078 ± 0.0023	22.599 ± 0.8206

TABLE 10: Prediction results of ESN with GA methods, on the test set for MSO, Mackey-Glass, and NARMA10 time series datasets.

$$\hat{t} = \frac{1}{N_{\text{gen}} \times N_{\text{pop}}} \sum \text{time}_i \quad (4.5)$$

where, $\hat{t}$ represents the average computational time, N_{gen} is the number of generations, N_{pop} is the population size, and time_i refers to the time spent from the algorithm's initiation until the last individual's evaluation. The average time consumption for the MSO, Mackey Glass, and NARMA10 datasets is 5.76, 10.56, and 8.64 minutes, respectively.

4.1.5 Discussion

In Figures (4.1.2), the evolution of the Quantile 10 value of the RMSE metric across generations is presented. The decision to use the Quantile 10 value as a representative for each generation stems from its role in guiding the selection of elites during the GA optimization methods employed. Although a common technique involves tracking the Average GA performance (fitness) across generations [145, 166, 183, 317], in this work, the Quantile 10 value is deemed more suitable. This is because the Cuckoo-inspired individuals introduced in each generation of the SACGA method can skew the error distribution, and since the worst individu-

Chapter 4. Findings and Discussion

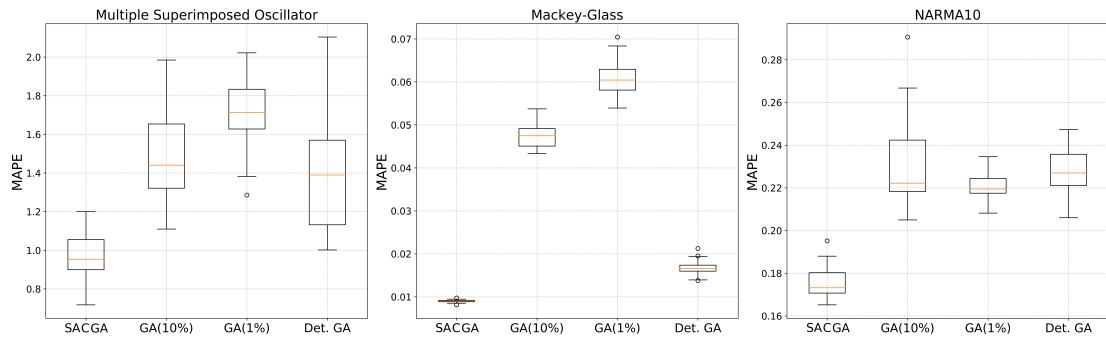


FIGURE 4.1.3: Boxplot of MAPE distributions for the proposed SACGA, GA (mutation rates 10% and 1%), and deterministic GA (Det. GA) on the MSO (left), Mackey-Glass (center), and NARMA10 (right) time series. The SACGA method consistently achieves lower MAPE values with reduced variation, indicating reliable predictive performance across all datasets.

als are then discarded, presenting the Average fitness (RMSE) might be misleading. From our perspective, the Quantile 10 value provides a more robust and reliable indicator of each GA method's performance across generations, reflecting the best-performing individuals.

An important contribution of this study is that it addresses a research gap identified by [34], where previous algorithms' cost functions optimized the error between predicted and actual outputs, neglecting the reservoir properties. A key innova-

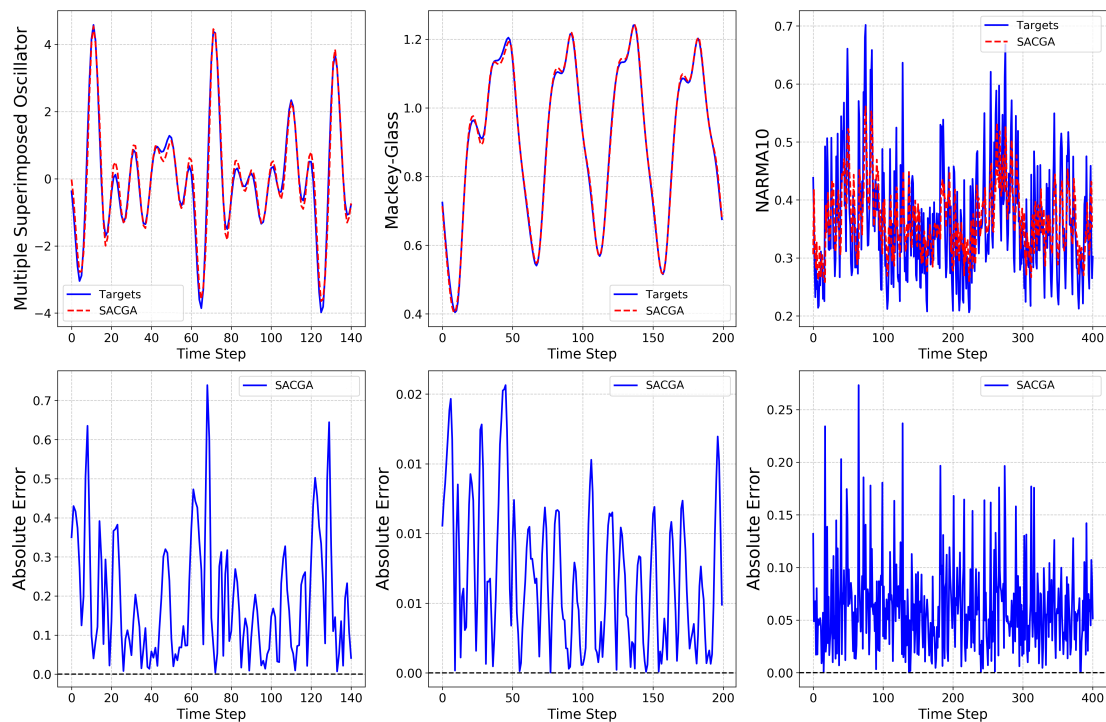


FIGURE 4.1.4: ESN network outputs (averaged over 32 instances) and targets visualization (top) and corresponding absolute errors (bottom), after applying the SACGA method to the MSO (left), Mackey-Glass (center), and NARMA10 (right) time series.

tion of the proposed SACGA model is its fitness evaluation function, which combines predictive error minimization and reservoir quality criteria. As shown in Figure (3.2.4) the primary criterion guides the optimization, while the secondary criterion plays a supportive role in the self-adaptive mutation rate assignment. Determining which criterion should lead this process is crucial. Preliminary findings suggest that prioritizing reservoir quality as the primary criterion yields suboptimal results. In contrast, using predictive error minimization as the primary guiding factor aligns with traditional GA principles, which prioritize fitness score optimization. Achieving a high level of predictive accuracy should be the initial goal; once this is attained, focusing on reservoir quality as a secondary criterion can further enhance the model. This approach ensures not only accuracy but also a well-structured reservoir state space, contributing to improved robustness and generalization by enabling the model to effectively differentiate between various inputs. This could be an important finding for those developing GA models with this or alternative reservoir quality metrics.

Another research gap highlighted by Bala et al. [34] is the oversimplification of search spaces in many ESN studies. Prior work frequently targets only a limited subset of hyperparameters [440], restricts the search range [54], or both [87, 248], often for the sake of computational feasibility. Moreover, much of the focus in the literature leans toward tuning network parameters rather than hyperparameters [44]. In contrast, our study adopts a more comprehensive strategy, optimizing a complete set of ESN hyperparameters across an extensively defined search space. This enables more flexible and thorough exploration, improving the model's capacity to adapt to varying tasks and data properties. In doing so, the SACGA offers a generalizable methodology for handling complex hyperparameter optimization in reservoir computing.

The reservoir quality itself is evaluated using the separation ratio, for which we adopt an empirical range of $[1, 10]$ as indicative of desirable behavior. This rationale suggests that controlled chaos within reservoir dynamics is essential, enhancing the ability to handle nonlinearity and input separation while preserving system stability. For ideal performance, the separation ratio of a reservoir should approximate 1, indicating that reservoir states proportionally reflect input distances [136, 380]. Yet, for complex, nonlinear, and dynamic series, a moderate amplification of input signals within the reservoir can improve its ability to differentiate between distinct signals effectively. Further experimentation is recommended for the specific task at hand. Nevertheless, while these numerical ranges are context-dependent, they still provide a useful starting point for experimentation and adjustment.

Our findings show strong predictive performance of the SACGA-optimized ESN on the MSO and Mackey-Glass datasets. However, while the SACGA outperforms other GA methods on NARMA10, alternative approaches in the literature show better results. For instance, GA applied to a limited set of hyperparameters and search space in Deep-ESN architecture shows improved performance [248], and GA optimization of spatial structures in a novel ESN design with variable signal delays

between neuron connections outperforms baseline ESNs [170]. This suggests that exploring alternative ESN architectures optimized with the SACGA could further improve predictive accuracy.

Finally, a noteworthy innovation of this work is the incorporation of reservoir quality metrics directly into the optimization process. Among the metrics considered, Lyapunov Exponents were initially explored due to their theoretical rigor in quantifying the sensitivity of dynamic systems to small perturbations. Their ability to characterize chaotic stability makes them appealing for evaluating internal reservoir dynamics. However, due to the high computational overhead required for their estimation, particularly over long time horizons and multiple trajectories, they were deemed impractical for efficient ESN training. In light of this, the separation ratio was chosen as a more computationally efficient proxy for reservoir quality. Importantly, the selected separation ratio range of $[1, 10]$ is consistent with Lyapunov Exponent values near zero, which indicate mild chaos at the so-called “edge of chaos.” Thus, while Lyapunov Exponents were not used directly, their theoretical insights are still embedded within the chosen optimization strategy. For a more detailed discussion on the dynamics within the reservoir and their role in model performance, refer to Subsection 3.1.7.

4.1.6 Concluding Remarks

This analysis introduces the Self-Adaptive-Cuckoo-GA (SACGA) for ESN model optimization. Experimental results on three benchmark datasets, namely Multiple Superimposed Oscillator, Mackey-Glass, and NARMA10, demonstrate that the SACGA outperforms conventional GA with fixed mutation rates at 10% and 1%, and deterministic GA methods, achieving superior RMSE, MAE, and MAPE values. Results show a great optimization improvement, achieving faster convergence to lower error rates.

The proposed SACGA builds on traditional GA by introducing self-adaptive features and adding a Cuckoo-inspired offspring in each generation, promoting exploration. Addressing two research gaps, this method includes a novel fitness evaluation that combines predictive error minimization with reservoir quality criteria, enhancing ESN model generalization and predictive accuracy. Moreover, the SACGA performs extensive optimization across a broad search space for all ESN hyperparameters, moving beyond the limitations of prior research that limit hyperparameters’ set or their search space.

This work places the SACGA model in the broader context of evolutionary algorithms, highlighting the benefits of new hybrid methods and adaptations. Future comparisons with more recent GA variants or other metaheuristic approaches could further strengthen the robustness of the SACGA framework. Additional directions for improvement include incorporating other reservoir quality metrics, such as eigenvalue spread, into the SACGA framework and extending its application to other ESN variants.

4.2 Real-World Application in Financial Forecasting

This section extends the application of SACGA optimized ESNs to the field of stock market forecasting. It begins with a detailed description of the proposed methodology. A detailed dataset and comparative methods description are then outlined. Next, the section is focused on performance analysis and concluding remarks.

4.2.1 Proposed Methodology

The following methodology addresses key challenges in evolutionary optimization and neural computation for one-step-ahead stock price forecasting. A novel hybrid framework is employed, combining an ESN with the novel SACGA optimization method to enhance predictive performance.

The architecture of the ESN is depicted in Figure (2.3.7), where the input layer processes external signals, which are then mapped into a high-dimensional state space by the reservoir. The output layer subsequently maps these states to the desired output, facilitating predictions. As detailed in Chapter (3), Section (3.1), the internal state of a reservoir with N neurons at time step t is represented by the vector $x(t) = (x_1, x_2, \dots, x_N)$, where each component $x_i(t)$ corresponds to the internal state of neuron i at time step t . To visualize it, each state $x(t)$ is a point in the N -dimensional state space, where inputs are projected through the reservoir dynamics. A key component of the ESN's performance is the reservoir, which is governed by the time-independent random weight matrix W_{res} of neuron connections. This weight matrix ensures the network's recurrent loops, which are essential for the model's memory retention.

The proposed methodology integrates the optimization capabilities of the SACGA with the predictive power of ESNs to enhance financial time series forecasting. This hybrid framework optimizes the ESN's hyperparameters through the SACGA, ensuring the model is configured for optimal predictive performance.

The methodology begins with an initial outer split of the dataset into training (80%) and testing (20%) subsets for the prediction task. The training subset, hereafter referred to as the prediction training set, is used for hyperparameter optimization via the SACGA algorithm. Within this optimization process, an inner split is applied to divide the prediction training dataset into an optimization training set (80%) and an optimization validation set (20%). This nested structure ensures that the ESN model configuration is performed using a validation split within the prediction training data, while the final model's performance is evaluated on a completely unseen prediction testing set.

At the core of this approach is an iterative optimization process, via the SACGA, starting with the generation of an initial population of potential solutions. Each solution represents a unique configuration of the ESN, defining key parameters such as learning rate, spectral radius, sparsity, reservoir size, and other ESN-specific attributes like input scaling (γ), and shift scaling (ω) related to the reservoir state update equation (3.4). These candidate solutions are evaluated by the SACGA based on their ESN predictive performance, using RMSE as the error minimization criterion and the separation ratio as a reservoir quality metric. This dual fitness evaluation ensures both accuracy and a well-structured reservoir state space, enhancing the model's robustness and generalization ability. The separation ratio plays a crucial role in regulating the diversity of the population throughout the selection, crossover, and mutation processes. With optimal values ranging between 1 and 10, the separation ratio promotes controlled chaos within the reservoir dynamics, allowing the system to handle nonlinearity and input separation while preserving stability. If the separation ratio deviates from this range, it promotes exploration by selecting more diverse solutions; otherwise, exploitation is promoted by focusing on solutions with low predictive error and good reservoir quality. The SACGA method operates through the selection, crossover, and mutation processes following the standard GA approach. Its innovative design features Cuckoo-inspired offsprings, leveraging the *Lévy* flight mechanism to explore the boundary regions of the search space, alongside a self-adaptive mutation rate mechanism applied during the mutation phase. The SACGA iterates through multiple generations of candidate solutions, ultimately determining the optimal set of hyperparameters once the termination criteria are met. The structural layout of the SACGA is shown in Figure (3.2.3), while a detailed description of the method is provided in Subsection 3.2.3.

Once the optimal hyperparameters are identified, they are applied to the ESN model, uniquely defining the ESN for prediction on the testing set. The ESN leverages its reservoir computing mechanism to capture the complex temporal dependencies of the time series, while the optimized hyperparameters ensure maximum predictive performance in forecasting future market trends.

Performance evaluation is conducted using various metrics, including regression metrics (MSE, RMSE, MAE, MAPE, R^2) and classification metrics (accuracy, precision, recall, and F1-score). Additionally, to analyze the reservoir's internal dynamics and chaos therein, quantitative metrics such as the Largest Lyapunov Exponent (LLE) and Separation Ratio, along with qualitative methods like Kernel PCA for visual interpretation, are employed.

This combined approach of SACGA optimization and ESN modeling, as illustrated in Figure (4.2.1), creates a robust, adaptive forecasting system capable of effectively predicting financial time series. By optimizing the ESN model design, the methodology leverages the strengths of evolutionary optimization and reservoir computing neural computation to deliver an efficient financial market prediction model.

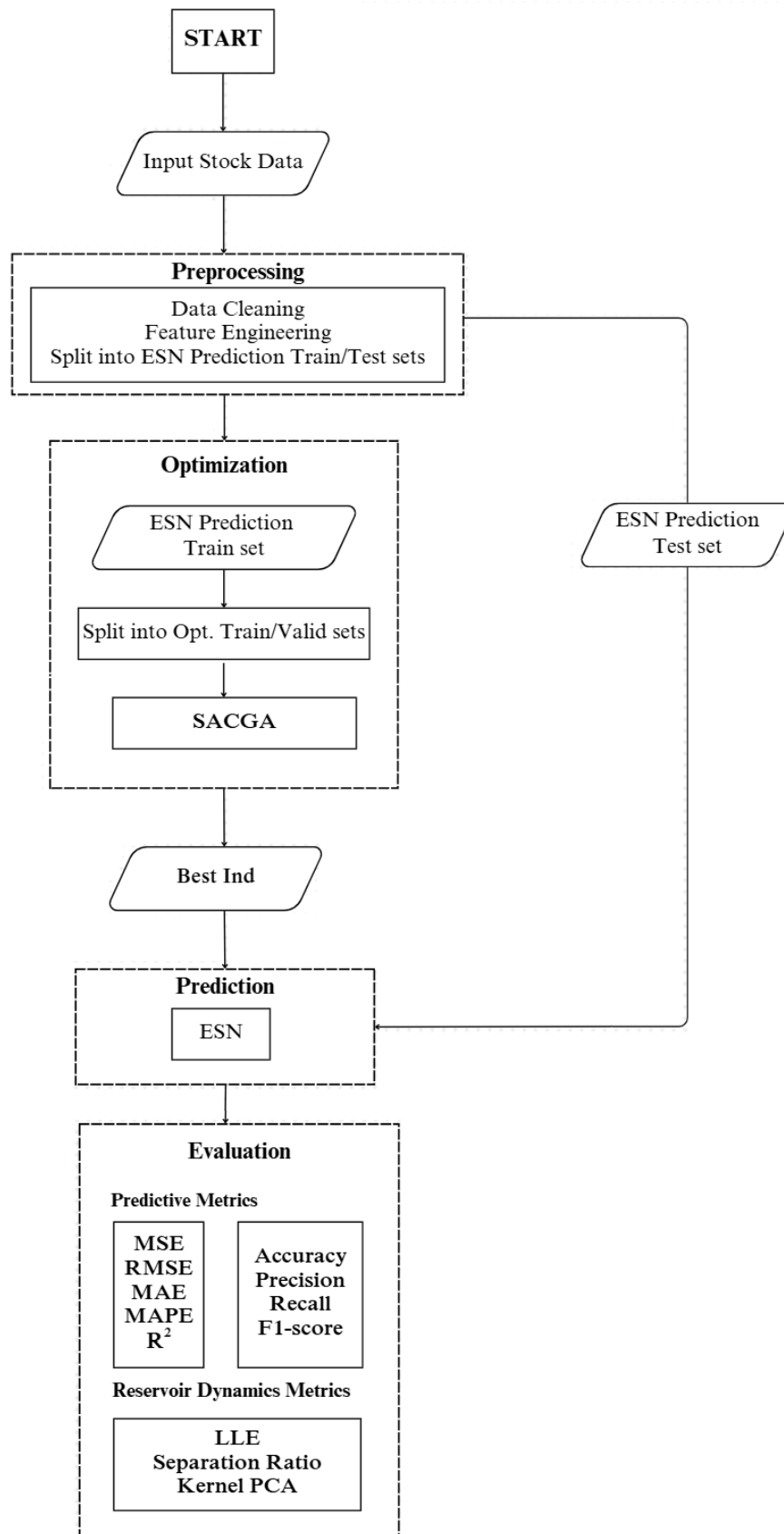


FIGURE 4.2.1: Overview of the proposed SACGA-ESN framework for one-step-ahead stock price forecasting, showing the workflow from data preprocessing through optimization, training, and performance evaluation using both predictive and reservoir dynamics metrics.

Experimental Design

Upon completing the SACGA hyperparameter optimization, the optimal set of hyperparameters is identified, uniquely defining the ESN for the prediction task. To ensure robust evaluation, predictions for the testing set are generated using 32 independently initialized ESN models. Each ESN instance is configured with the same optimized hyperparameters obtained from the SACGA method. The final predictive performance represents the averaged outcome across these 32 ESN instances, following the approach outlined in [241].

Table (11) summarizes the parameter settings for the SACGA framework, including the population size, and number of generations, as well as the corresponding search ranges for all ESN hyperparameters: leaking rate (lr), spectral radius (ρ), shift scaling (ω), input scaling (γ), reservoir size (N), reservoir sparsity (s_{res}), input sparsity (s_{in}), and readout regularization (α). See Section 3.1.5 for a detailed discussion of each hyperparameter. Importantly, all ESN hyperparameters are optimized in the proposed SACGA framework, highlighting its universality in treating the ESN as a complete, integrated system.

<i>Parameter</i>	<i>Value / Range</i>
<i>Population Size</i>	120
<i>Generations</i>	10
<i>Leaking Rate (lr)</i>	(0.001, 0.999)
<i>Spectral Radius (ρ)</i>	(0.001, 9.999)
<i>Omega (ω)</i>	(0.001, 9.999)
<i>Gamma (γ)</i>	(0.001, 9.999)
<i>Reservoir Size (N)</i>	(40, 500)
<i>Sparsity Reservoir (s_{res})</i>	(0.001, 0.999)
<i>Sparsity Input (s_{in})</i>	(0.001, 0.999)
<i>Readout Regularization Parameter (α)</i>	$[1e^{-11}, 1e^{-6}, 1e^{-3}, 0.1, 1, 10, 100]$

TABLE 11: SACGA Optimization Parameters and ESN Hyperparameter Search Ranges

As discussed in Section (4.1), there are no strict rules for GA population size, but small populations can limit diversity, potentially leading to suboptimal performance [145], while larger population sizes enhance the probability of capturing optimal solutions within the initial population [142]. Typical population sizes range from 50 to 100 [317]. In Section (4.1), with a SACGA setup of (pop, gen) = (50, 15) on benchmark datasets, it is observed faster convergence to lower error rates in the early generations. Based on these observations, for the real-world scenario considered here, an SACGA configuration of (pop, gen) = (120, 10) is selected, providing a sufficiently large population to capture optimal solutions and enough gener-

ations to evolve toward them. Preliminary experiments with alternative population and generation configurations were also conducted, but resulted in inferior performance or higher computational cost. The selected configuration ensures effective exploration and exploitation while maintaining computational efficiency, which is critical given the high cost of evaluating ESN reservoirs periodically. Although this setup is not intended as a universal solution, it represents a well-performing configuration for this specific SACGA application, and a practical reference for similar studies.

The ESN hyperparameter search ranges are chosen based on theoretical guidance, prior literature, and computational considerations. The leaking rate lr is naturally constrained to $(0, 1)$ to prevent excessive retention or loss of neuron activation. The spectral radius ρ is set in the range 0.001 – 9.999 following Trouvain et al. [376], while input scaling (γ) and shift scaling (ω) (~ 0.001 – 10) support dynamics from near-linear to strongly nonlinear. Regarding reservoir size N , although early guidelines suggested reservoirs from tens to thousands of neurons [244], more recent studies [62, 358] show that very large reservoirs increase computational cost and risk overfitting, with optimal sizes typically in a moderate range (around 200–400 units). Therefore, following Steiner et al. [358], we search reservoir sizes in $(40$ – $500)$, balancing representational capacity and efficiency. Sparsity parameters for the reservoir and input cover the full feasible range 0.001 – 0.999 . The readout regularization parameter α spans a wide logarithmic range, allowing the ESN readout layer to balance fitting accuracy and overfitting prevention. Apart from natural constraints, these ranges reflect the configurations evaluated in our framework, which are broad yet computationally feasible, and serve as general guidance, though task-specific experimentation is suggested.

In the implementation of the ESN, the first 200 data points are intentionally excluded from the training set, a common technique known as the initial washout or transient phase removal. This approach is necessary due to the way the ESN is structured, with its reservoir being randomly initialized. At the beginning of a run, the reservoir's internal state is typically influenced more by its random initial conditions than by the input signal itself. To avoid letting these initial conditions affect the model's performance, the first part of the data is used solely to drive the network, and not for training the readout layer. This allows the reservoir to stabilize and align with the actual input dynamics, ensuring that its state reflects the true temporal structure of the input sequence. By discarding the first 200 steps, the model effectively "washes out" any instability from the random initialization. Once this transient phase is over, the reservoir reaches an "echo state," where its dynamics are a reliable, nonlinear reflection of the recent input history, enabling more stable and accurate training of the output weights.

Model Performance Metrics

The performance of the ESN model is assessed using both regression and classification metrics, reflecting its ability to accurately predict not only the magni-

tude but also the direction of stock returns. Regression-based metrics include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and the Coefficient of Determination (R^2). These metrics evaluate the closeness of predicted return values to the actual returns. Lower values of RMSE, MAE, and MAPE indicate better predictive accuracy, while a higher R^2 value (closer to 1) reflects a stronger model fit.

Since the direction of return is a critical factor in financial decision-making, we also employ classification-based metrics, i.e. Accuracy, Precision, Recall, and F1-score, to assess the ESN's ability to correctly predict whether the return will be positive or negative. These metrics are derived by transforming the predicted and actual returns into binary directional classes (e.g., up/down). Here, TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. A higher value in these metrics indicates better classification performance, especially in capturing the directional movement of the market.

The formulas for both regression and classification metrics are summarized as follows:

Regression Metrics	Classification Metrics
$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$	$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$	$Precision = \frac{TP}{TP+FP}$
$MAE = \frac{1}{n} \sum_{i=1}^n y_i - \hat{y}_i $	$Recall = \frac{TP}{TP+FN}$
$MAPE = \frac{100\%}{n} \sum_{i=1}^n \left \frac{y_i - \hat{y}_i}{y_i} \right $	$F1-Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$
$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	

where n is the number of observations, y_i is the actual value, $\hat{y}_i$ is the predicted value of the i^{th} observation, and $\bar{y}$ is the mean of the actual values.

Hardware and Software Environment

All experiments were conducted on a PC running Linux (kernel version 6.8.0-48-generic) with an Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz (3 cores) and Python 3.7.16. Core libraries included NumPy and Pandas for numerical operations and data manipulation, PyTorch for deep learning model development and fast computations, and Scikit-learn for auxiliary machine learning tasks and evaluation.

4.2.2 Dataset Description

Our empirical study on real-world stock data utilizes ten-minute intervals of stock data from the S&P500 Index covering the period from April 29, 2019 to July 02, 2020.

The original dataset comprises 11,960 data points across 597 stocks. After addressing missing or undefined values and accounting for the need to use several initial data points for computing certain technical features, the final dataset includes 411 stocks, covering the period from May 6, 2019, to July 2, 2020, with a total of 11,721 data points. For the purposes of this research, this dataset serves as a proxy for a Composite Index representing the S&P 500. It is a valuable tool for identifying potential market anomalies in the capital market, considering it reflects the impact of extensive analyst coverage and high investor activity.

As explained in subsection (4.2.1), the dataset is divided into distinct phases for optimization and prediction tasks. The specific date ranges and index splits are presented in Table (12) and illustrated in Figures (4.2.2) and (4.2.3).

Task Phase	Date Range	Index Split
SACGA Training	May 6, 2019 16:00 - Jan. 27, 2020 09:30	[0:7520]
SACGA Validation	Jan. 27, 2020 09:40 - Apr. 02, 2020 09:30	[7520:9400]
ESN Testing	Apr. 02, 2020 09:40 - Jul. 02, 2020 16:00	[9400:11721]]

TABLE 12: Training, validation, and testing periods used for SACGA optimization and ESN prediction tasks, with corresponding date ranges and index splits

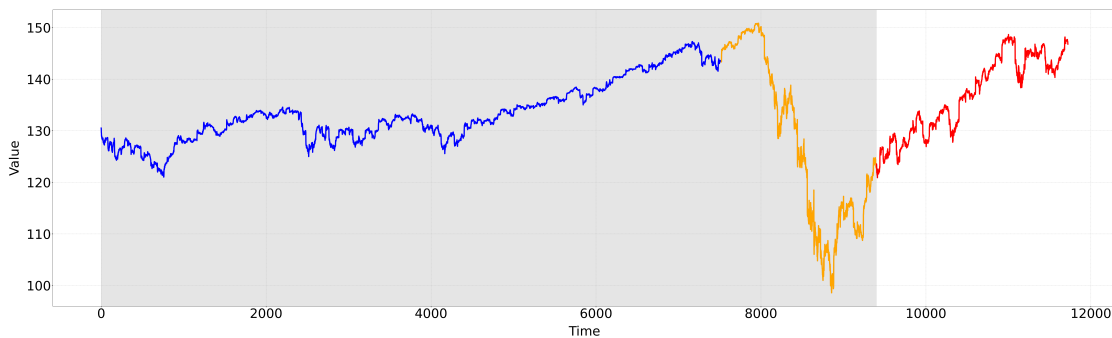


FIGURE 4.2.2: Visualization of the dataset split into training, validation, and testing phases, corresponding to the SACGA optimization and ESN prediction tasks for portfolio values. The gray-shaded region denotes the overall optimization interval. Within this region, the blue segment represents the SACGA optimization training phase, and the orange segment indicates the SACGA optimization validation phase. The red segment represents the ESN prediction test period.

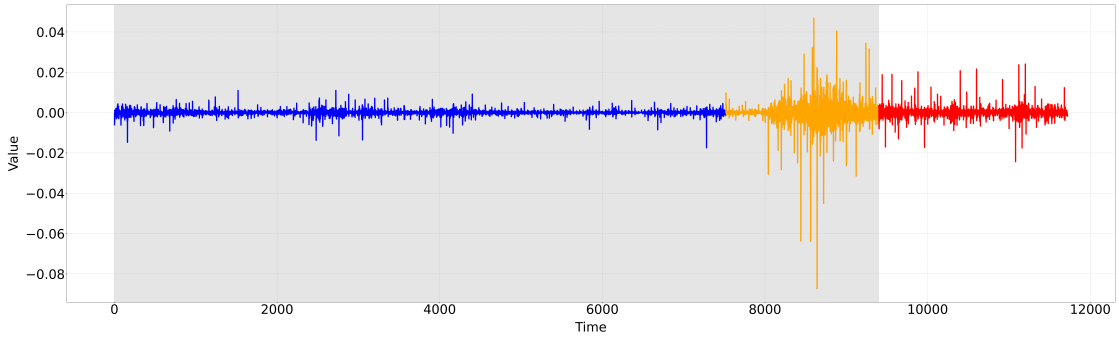


FIGURE 4.2.3: Visualization of the dataset split into training, validation, and testing phases, corresponding to the SACGA optimization and ESN prediction tasks for portfolio returns. The color scheme and phase descriptions are consistent with Figure 4.2.2.

In this study, we focus on the returns of the constructed Composite Index, as this approach aligns more closely with standard financial modeling practices, particularly in the context of portfolio optimization, risk assessment, and trading strategies. Consequently, the primary target variable is the return observed at each ten-minute interval. Formally, let P_t denote the portfolio value at time t . The return R_t at time t is defined as the percentage change in portfolio value from the previous time step $t - 1$ to the current time t . Mathematically, this is expressed as:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 \quad (4.6)$$

The summary statistics of the proxy Composite Index returns, as shown in Table 13, reveal key distributional features. The near-zero mean (0.000014) indicates that on average, returns are nearly neutral. The range between the maximum (0.046781) and minimum (-0.087303) values indicates the presence of extreme positive and negative returns. The standard deviation (0.002693) signifies moderate volatility of returns. Importantly, the strong negative skewness (-4.53) suggests that extreme negative returns are more frequent or severe than positive returns, which indicates downside risk. The exceptionally high kurtosis (187.47) signals heavy tails and a leptokurtic distribution, deviating significantly from a normal distribution. These distributional characteristics are also visually evident in Figure 4.2.4.

Sample	Mean	Max.	Min.	Std. Dev.	Skewness	Kurtosis
11721	0.000014	0.046781	-0.087303	0.002693	-4.534454	187.465265

TABLE 13: Summary statistical indicators of the proxy Composite Index returns representing the S&P 500 Index

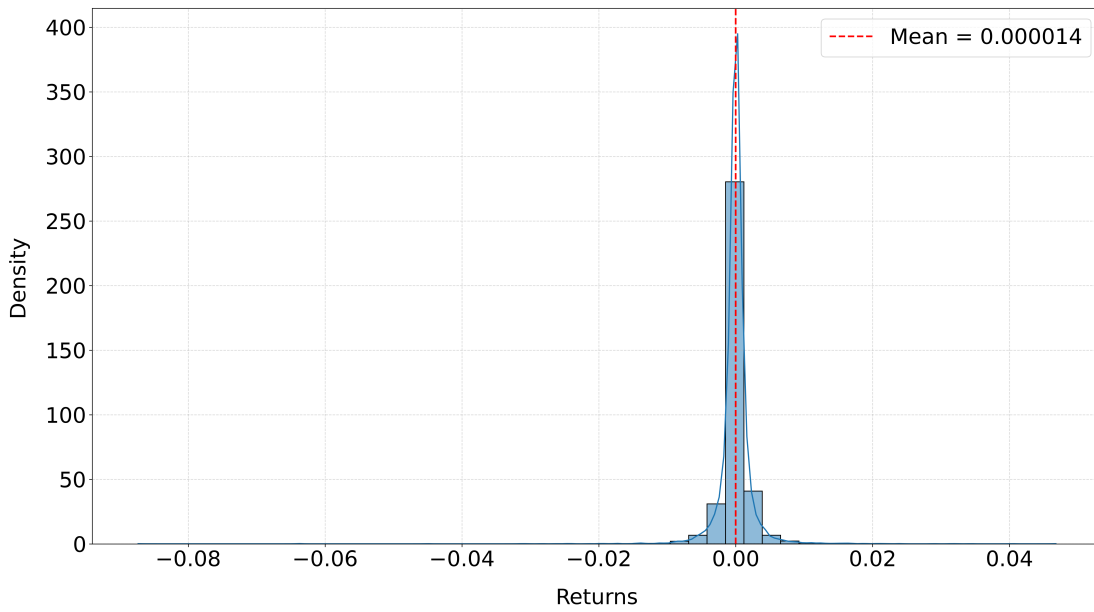


FIGURE 4.2.4: Histogram and Kernel Density Estimate (KDE) of the proxy Composite Index returns

Feature Description

A total of 30 technical indicators are constructed to capture diverse aspects of return dynamics, including trend-following averages, volatility measures, and temporal dependencies. The mathematical formulas of these features are detailed in the appendix (see Section .3).

Feature selection is performed using correlation analysis. A heatmap is generated to assess the linear relationships between candidate input features and the target return. Features with a positive correlation coefficient greater than 0.5 are retained as predictive variables. Negatively correlated features are excluded from consideration.

Unlike other approaches [50] that remove highly correlated features (e.g., correlation > 0.8) to prevent redundancy, our method intentionally retains features even if they carry somewhat overlapping information. This is because, within the ESN reservoir, similar signals can reinforce the predictive patterns and improve the model's robustness. At the same time, features that are highly negatively correlated with the target variable are excluded, even if strongly correlated, since they may introduce conflicting signals into the reservoir and degrade its ability to represent the target dynamics effectively. This feature selection strategy prioritizes enriching the reservoir with relevant information, rather than strictly removal of redundant features. Preliminary experiments confirmed that including negatively highly correlated features did not improve predictive performance, further justifying their exclusion.

The final selected features, together with the return series itself, serve as inputs to optimize the ESN architecture through the proposed SACGA for one-step-ahead return prediction.

Description of Selected Features As shown in Figure (4.2.5), the features exhibiting the strongest positive linear correlation with the target portfolio return (correlation coefficient > 0.5) are:

- **SMA3 (Simple Moving Average, window 3):** Captures very short-term trends in returns, smoothing out noise while remaining sensitive to recent price movements. Its strong correlation suggests that very recent momentum is highly predictive of the next return.
- **EMA3 and EMA6 (Exponential Moving Averages, windows 3 and 6):** These emphasize recent returns with exponentially decreasing weights, highlighting the importance of the latest returns more than older ones. Their inclusion reinforces the significance of short-term return dynamics.
- **log_returns (Logarithmic Returns):** Taking the logarithm of returns often stabilizes variance and normalizes the data, making the relationship with future returns more linear and predictable. The strong correlation here confirms that log returns retain important predictive information.
- **return*volatility40 (Return-Volatility Interaction with a 40-interval window, i.e., one trading day):** This feature captures the interaction effect between returns and their recent volatility. It may reflect risk-adjusted momentum or market states where volatility impacts return behavior, indicating a nuanced influence on future returns.
- **return_gap_40 (Difference between current return and return 40 intervals ago, i.e., one trading day):** This measures longer-term changes in return levels, possibly capturing shifts in market regimes or mean-reverting tendencies over the course of a day.
- **BB_Pct (Bollinger Bands Percent Position):** Position of the current return relative to upper and lower Bollinger Bands over a 200-interval window (approximately five trading days). This feature indicates whether returns are near historical highs or lows, which may signal overbought or oversold conditions impacting future return direction.

Together, these features combine short-term momentum, volatility interactions, longer-term regime shifts, and technical conditions related to price extremities, providing a comprehensive, multi-scale view of market dynamics predictive of next-step returns.

The final dataset used for model training and evaluation is based on this selected subset of features, not the full set depicted in the correlation heatmap. This refined

4.2. Real-World Application in Financial Forecasting

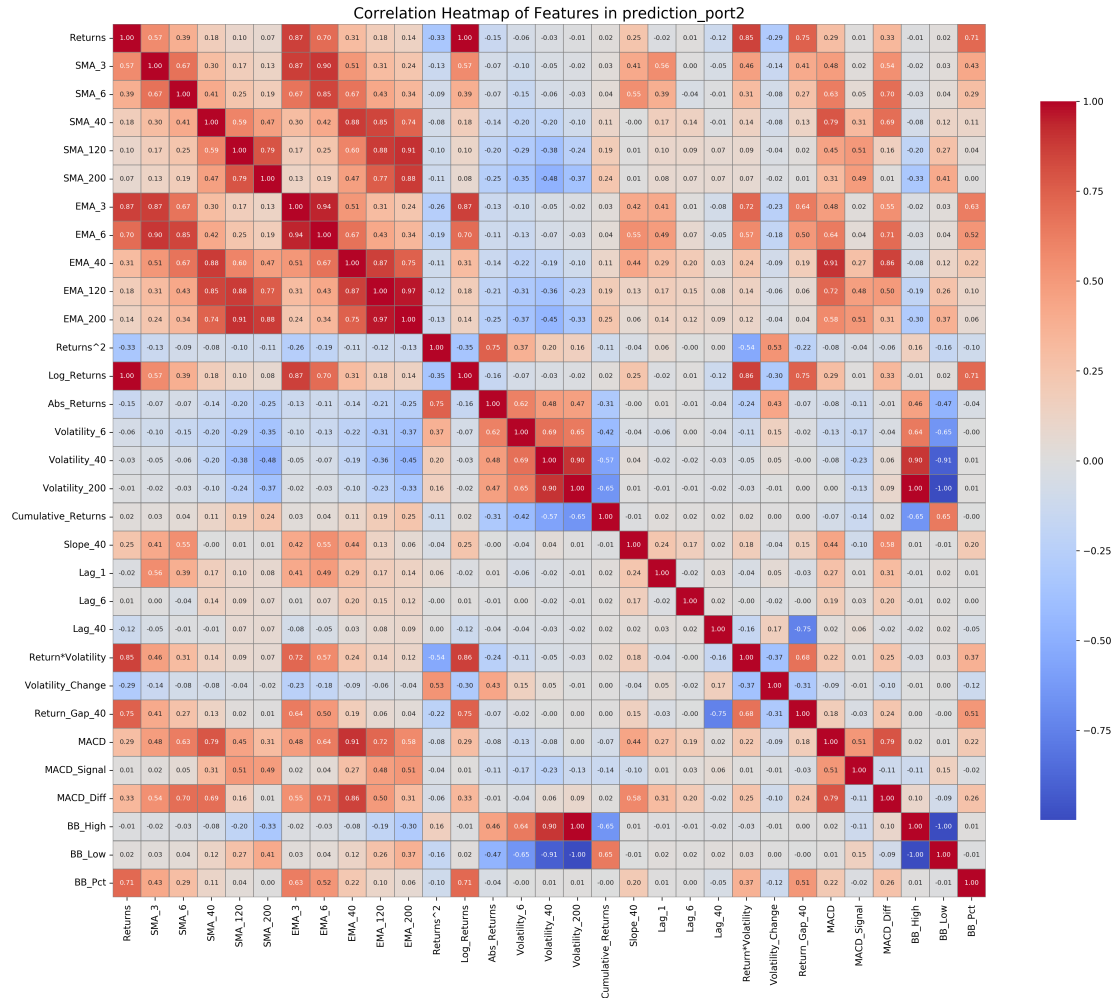


FIGURE 4.2.5: Heatmap illustrating the correlation coefficients between candidate technical indicators and the target return. Features with positive correlations above 0.5 are selected as inputs for the ESN model.

input set forms the basis for the training, validation, and testing splits described in Subsection (4.2.2).

4.2.3 Comparative Method

To benchmark the forecasting performance of the optimized ESN, a Random Walk model is considered. This choice provides a theoretically grounded, economically meaningful, and practically realistic baseline, with outperforming it constituting a key criterion for challenging the EMH. While comparisons with state-of-the-art DL models such as LSTM or GRU could offer additional empirical context, the primary goal of this study is to understand the dynamical mechanisms of the ESN reservoir, rather than to conduct purely empirical benchmarking.

Random Walk Model

The Random Walk model is a widely recognized naive forecasting strategy that assumes the best estimate for the future value of a time series is its most recent observed value. This model operates under the assumption that all changes in the time series are purely stochastic, implying that prices evolve according to a random process and that past values offer no predictive insight into future movements. Mathematically, a Random Walk model with drift can be expressed as:

$$p_{t+1} = p_t + \mu + \epsilon_{t+1} \quad (4.7)$$

where p_t denotes the observed value (e.g., price) of the time series at time step t , and p_{t+1} represents the predicted value at the next time step, $t+1$, μ is the drift term representing a constant trend over time, and ϵ_{t+1} is a stochastic error term, typically assumed to be drawn from a normal distribution with zero mean and constant variance, i.e., $\epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2)$.

In this analysis, we employ the simplified version of the Random Walk, where both the drift (μ) and the stochastic noise (ϵ_{t+1}) are set to zero. This reduces the model to:

$$p_{t+1} = p_t \quad (4.8)$$

This formulation embodies the core intuition of the naive forecasting approach, that the best estimate for tomorrow's price is simply today's price, without any adjustment for trend or randomness. Such a model provides a minimal performance baseline against which more sophisticated forecasting methods can be evaluated.

The theoretical foundation of the Random Walk is closely tied to the weak form of EMH, which posits that all publicly available information is immediately fully reflected in asset prices [105]. Under this framework, price movements are driven by the arrival of new information. That is, tomorrow's stock price change only reflects tomorrow's news. But, the news is unpredictable, and thus the resulting stock price must be unpredictable, and random [255].

Additionally, investor judgment in financial markets can occasionally be flawed. Some market participants are less rational. As a result, pricing irregularities and inefficiency spots can emerge and sometimes persist over short periods. However, these instances are typically short-lived. Rational and well-informed investors actively seek to exploit such opportunities, and their actions tend to correct mispricings, gradually pushing the market back toward efficiency. As Singal [342] explains, if a mispricing exists, smart investors and arbitrageurs will act to exploit the opportunity for excess returns, and as more of them participate, the mispricing should disappear. As a result, this collective investor behavior makes any arbitrage opportunities quickly identified and neutralized, reinforcing the idea that market prices approximate a Random Walk over time.

Despite its simplicity, the Random Walk serves as a crucial benchmark in time series forecasting. It provides a reference model performance under the scenario that the underlying time series contains no learnable structure. Comparing the performance of more sophisticated models against this naive approach helps determine whether they can extract meaningful predictive signals from data or their added complexity merely overfits to noise.

4.2.4 Performance Analysis

This subsection presents a comprehensive evaluation of the ESN models optimized via the proposed SACGA. It includes a detailed characterization of the best-performing individual, highlighting insights into the design of the SACGA optimization method, along with regression and classification performance comparisons against a Random Walk baseline, and an in-depth analysis of reservoir dynamics capacity. For the latter, both quantitative metrics, such as the Lyapunov Exponent and Separation Ratio, and qualitative visualizations via kernel PCA projections are employed to demonstrate the optimized ESN's superior stability, expressivity, and generalization capabilities compared to randomly initialized ESN models.

Best Individual Characterization

The best individual was identified in the final (10th) generation at index 91. While the index itself carries no inherent meaning due to the stochastic initialization of the population, it is noteworthy that the individual occupying this same position in the previous generation exhibited suboptimal characteristics, namely, an RMSE worse than the population average and a separation ratio that exceeded the predefined threshold ($17.72 > 10$). Such values are undesirable in our optimization setting, and consequently, a high mutation rate of 0.8 was assigned to this individual in order to encourage genetic diversity and further exploration, through mutation.

Despite the high mutation rate assigned, crossover clearly played a dominant role, as evidenced by the leaking rate ($lr = 0.999$) and input scaling factor ($\gamma = 0.922$)

of the best individual matching those of an elite solution previously discovered in generation 8, which was retained and carried forward into generations 9 and 10. Furthermore, the extreme value of the leaking rate ($lr = 0.999$), located near the boundary of the defined search space, proves that this hyperparameter value is inherited from a Cuckoo-inspired individual.

These findings underscore the inherently stochastic nature of the proposed SACGA evolutionary algorithm. More importantly, they demonstrate the effectiveness of coupling traditional error minimization with reservoir quality metrics, such as Lyapunov stability and separation ratio, within a multi-objective fitness function, as part of the self-adaptive mutation rate assignment mechanism. Furthermore, the addition of Cuckoo-inspired offsprings regularly plays a crucial role in efficiently exploring the boundaries of the search space. Overall, the SACGA framework guides the optimization process toward discovering a robust and dynamically well-structured ESN configuration.

The hyperparameter values of the best-performing individual are as follows:

Best Individual: $(lr, \rho, \omega, \gamma, N, s_{res}, s_{in}) = (\mathbf{0.999}, 0.324, 0.055, \mathbf{0.922}, 472, 0.801, 0.536)^*$

Regression & Classification Metrics

The tables below summarize the comparative performance of the SACGA optimized ESN and Random Walk models. The metrics reported include MSE, RMSE, MAE, MAPE, and R^2 for regression, and Accuracy, Precision, Recall, and F1-Score for classification. For the optimized ESN, results represent the mean $\pm$ standard deviation across 32 runs, whereas only single values are reported for Random Walk.

As shown in Table 14, the optimized ESN consistently outperforms the Random Walk baseline, achieving significantly lower error metrics and superior R^2 values. Similarly, Table 15 illustrates enhanced classification performance with the SACGA optimized ESN, demonstrating much higher accuracy, precision, recall, and F1-Score.

¹* Full-precision values: (**0.999**, 0.323864890018344, 0.055410055867875875, **0.9218945675714245**, 472, 0.8009725471797375, 0.5363572341987547)

Metric	SACGA ESN	Random Walk
MSE	6.060e-06 $\pm$ 7.547e-12	11.935e-06
RMSE	2.462e-03 $\pm$ 1.533e-09	3.454e-03
MAE	1.460e-03 $\pm$ 3.206e-09	2.156e-03
MAPE	1.003e+02 $\pm$ 1.480e-02	7.420e+02
R^2	-9.782e-04 $\pm$ 1.247e-06	-0.971

TABLE 14: Regression performance comparison of SACGA ESN (mean $\pm$ std over 32 runs) against Random Walk baseline.

Metric	SACGA ESN	Random Walk
Accuracy	74.860% $\pm$ 0.234%	33.147%
Precision	73.593% $\pm$ 0.290%	31.206%
Recall	75.277% $\pm$ 0.367%	31.233%
F1-Score	74.425% $\pm$ 0.242%	31.220%

TABLE 15: Classification performance comparison of SACGA ESN (mean $\pm$ std over 32 runs) against Random Walk baseline.

The predictive performance of the SACGA optimized ESN is further evaluated by focusing on its ability to forecast the direction of financial returns rather than their exact values. The continuous return series is transformed into binary directional labels by applying first-order differencing and a sign-based classification rule: an increase in return is labeled as 1 (up), while a decrease or no change is labeled as 0 (down). The resulting confusion matrix, presented in Table 16, highlights the model's ability to identify directional shifts, with 887 correct classifications of downward movements and 850 correct classifications of upward movements, totaling 1,737 accurate predictions. Misclassifications include 307 false positives (predicted up when the market moved down) and 277 false negatives. Overall, the SACGA optimized model exhibits a balanced predictive performance; however, a slight tendency to overestimate upward movements indicates a subtle optimism bias.

Actual Direction \ Predicted Direction	0 (Down)	1 (Up)
0 (Down)	887	307
1 (Up)	277	850

TABLE 16: Confusion Matrix of the SACGA Optimized ESN

Given that the Random Walk embodies the EMH's declaration of price unpredictability based solely on past values, any model that consistently outperforms it demonstrates an ability to extract meaningful, non-random temporal patterns from market data. That said, the superior directional accuracy of the SACGA optimized ESN demonstrates its ability to capture subtle, exploitable dependencies and structures that naive models cannot. The model's ability to correctly identify upward and downward trends at a rate significantly above chance provides a practical edge for informed decision-making and risk management.

Such improved predictive accuracy can translate into better timing of trades, enhanced portfolio optimization, and more effective algorithmic trading, all of which contribute to superior financial performance. Moreover, outperforming a Random Walk challenges the notion of pure market efficiency on short intraday horizons, indicating that transient inefficiencies or patterns may be leveraged before market correction mechanisms restore equilibrium.

Reservoir Stability: Probing Inner Chaos

This section investigates the internal stability and dynamic behavior of the ESN reservoir from both quantitative and qualitative perspectives. The goal is to determine whether the reservoir operates within a regime that supports stable and reliable predictions.

To contextualize the analysis, the dynamics of the optimized ESN are compared against a baseline ensemble of 100 randomly initialized ESNs. Given the inherent stochastic nature of the reservoir behavior, each random ESN is evaluated across 10 independent trials, and the reported metrics represent the mean and standard deviation across these runs. The full set of hyperparameter configurations used to generate the 100 random ESNs is detailed in Appendix .4.

Quantitative Analysis The goal here is to numerically characterize the underlying reservoir dynamics. Two complementary metrics are reported:

- **Largest Lyapunov Exponent (LLE):** This quantifies the average exponential rate of divergence of nearby trajectories in state space. Values near zero suggest the system operates near the edge of chaos, a desirable regime balancing stability with dynamic complexity. Strongly negative values imply overly stable, unresponsive dynamics, while large positive values indicate chaotic, erratic, and unpredictable behavior, both of which are typically undesirable. As a practical guideline within the scope of this research, an absolute LLE value below 0.001 (i.e., $|LLE| < 0.001$) can be considered to signify operation within the edge of chaos.
- **Separation Ratio (SR):** Measures how strongly the reservoir amplifies differences between input signals in its state space. A value greater than one indicates that distinct inputs remain distinguishable, reflecting better dynamic

separation and representational capacity. During the proposed SACGA optimization method, the fitness criterion for reservoir quality specifically favors SR values within the range of 1 to 10 to ensure effective and stable input separation.

Table 17 summarizes the computed values of LLE and SR for the optimized ESN, averaged over 10 runs.

Metric	Value	Interpretation
Largest Lyapunov Exponent	0.00045	Mild chaos; good sensitivity to input
Separation Ratio	7.587	Good input separability, no over-amplification

TABLE 17: Chaos-related metrics computed for the optimized-ESN.

These results indicate that the optimized reservoir operates in a highly favorable regime: the LLE value is positive but very close to zero, confirming mild chaos conducive to input sensitivity and rich computational dynamics without risking erratic reservoir behavior. This edge of chaos operation is often linked with superior performance in temporal processing tasks, as it allows the system to both retain information and adapt responsively to input variations.

Notably, the Separation Ratio for the optimized ESN is orders of magnitude smaller and more moderate than those of random ESNs (Table 18). While random ESNs exhibit extremely high SR values, suggesting over-separation or over-amplification of input differences, such behavior can lead to noisy, unstable, or overly sensitive reservoir representations that may hinder generalization. In contrast, the optimized ESN strikes an effective balance: the reservoir amplifies differences between input classes enough to ensure clear and meaningful separation, but without exaggerating minor variations that could cause fragmentation or noise. This controlled amplification within the high-dimensional reservoir state space enables reliable discrimination of temporal patterns, thereby promoting improved generalization performance.

The LLE and SR boxplots (Figures 4.2.6 and 4.2.7) further validate these conclusions by visually confirming that the optimized ESN reservoir exhibits greater system stability and more meaningful input class separation compared to the wide variability observed in random ESN initializations.

Metric	Random ESNs (Mean $\pm$ SD across 100 means)	Optimized ESN (Mean $\pm$ SD)
Largest Lyapunov Exponent	0.0011 $\pm$ 0.001	0.00045 $\pm$ 0.00026
Separation Ratio	88472.14 $\pm$ 42408.57	7.587 $\pm$ 0.11

TABLE 18: Reservoir dynamics comparison between 100 random ESNs and the optimized ESN (mean $\pm$ SD over 10 runs).

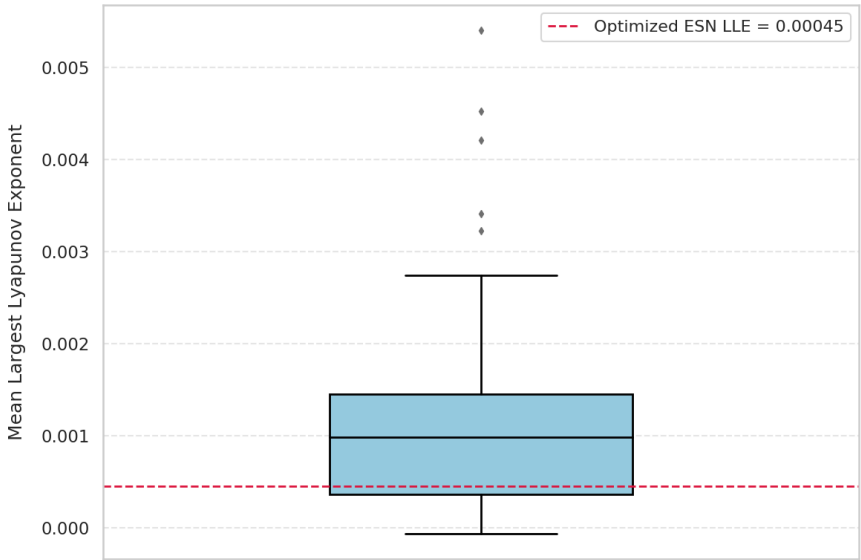


FIGURE 4.2.6: Boxplot of Largest Lyapunov Exponent (LLE) values across 100 random ESNs (each averaged over 10 runs). The red line indicates the optimized ESN, which lies near the lower end of the distribution, reflecting enhanced dynamic stability.

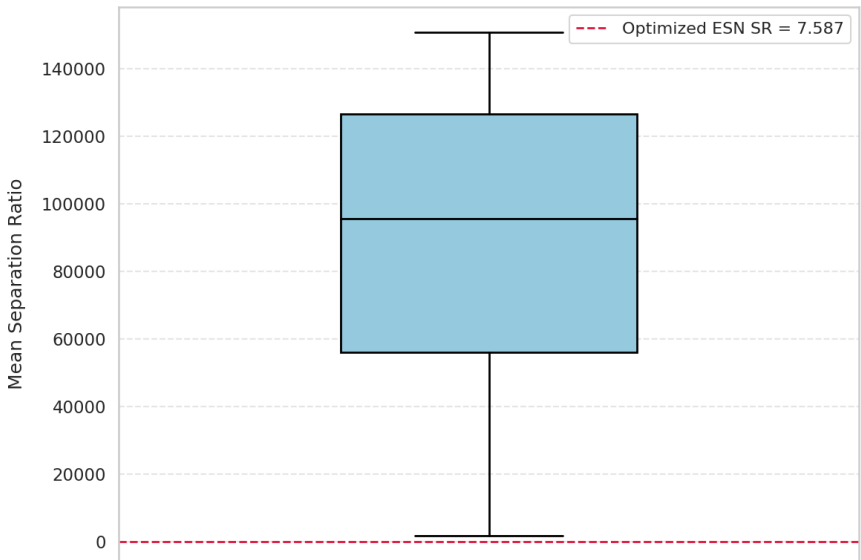


FIGURE 4.2.7: Distribution of Separation Ratio (SR) values for 100 random ESNs (averaged over 10 runs each). The optimized ESN (red horizontal line) lies near the lower whisker, achieving a more rational input class separation in the reservoir dynamics of the optimized ESN model.

Qualitative Analysis To complement the numerical metrics, we apply kPCA to visualize the structure of the reservoir state space at different stages of training. Specifically, the reservoir activations from the training set, before readout training, and the test set, after readout training, are projected into a three-dimensional subspace defined by the leading kernel principal components. This three-dimensional nonlinear projection provides a reduced representation of the reservoir states in a feature space that captures complex nonlinear relationships, effectively revealing the dominant patterns and intrinsic geometric structures in the high-dimensional (N -dimensional) reservoir state space. In what follows, we present the kPCA projection of the optimized ESN, which consistently yields the same structure across multiple runs. We also show representative projections from the 100 randomly initialized ESNs, specifically the most frequently occurring patterns.

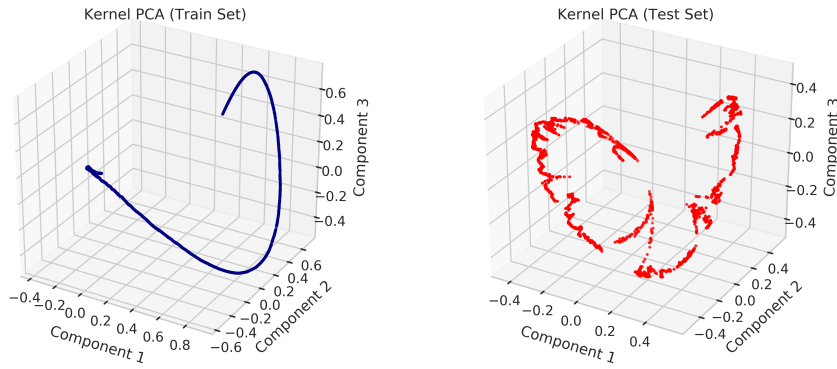


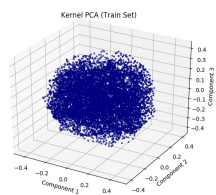
FIGURE 4.2.8: kPCA visualization of training set dynamics (left) and test set dynamics (right) for the optimized ESN reservoir.

Figure 4.2.8 shows the resulting kPCA projection of the reservoir states generated by the optimized ESN for both training and test sets. The visualization highlights the internal dynamics captured by the reservoir. In the training set, the reservoir states evolve along a smooth, continuous structure, reflecting well-organized reservoir dynamics with sufficient complexity to encode temporal dependencies in the input. This suggests that the network has effectively internalized the temporal structure of the input signals. The test set projection, while more fragmented, still exhibits traces of structure, demonstrating the reservoir’s ability to generalize its learned dynamics beyond the training distribution. The deviations and increased irregularity in the test set trajectory reflect a distributional shift relative to the training data. However, the emergence of recurrent substructures (motifs) in both the training and test sets suggests that the reservoir retains a degree of dynamical consistency and generalization. This implies it continues to encode meaningful structure in response to novel input. Such behavior is characteristic of a well-tuned reservoir, exhibiting robust, though imperfect, extrapolative capacity as revealed through the kernel PCA projection. This supports the notion that the optimized ESN not only learns the training dynamics effectively but also internalizes a representational structure

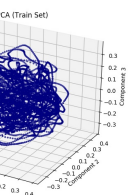
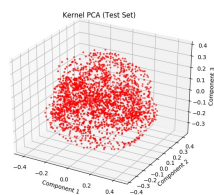
robust enough to extrapolate beyond the training regime.

This visual interpretation supports the earlier quantitative finding of ordered dynamics via LLE and separation ratio, reinforcing the conclusion that the reservoir has internalized structure useful for prediction tasks.

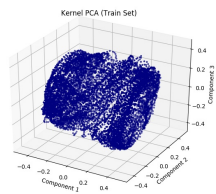
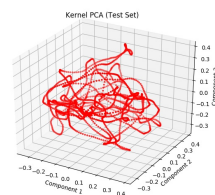
From the 100 randomly initialized ESNs, kPCA projections of their reservoir dynamics are generated, with the most representative examples presented in Figure 4.2.9. In stark contrast to the ordered, low-dimensional structure observed in the optimized ESN (Figure 4.2.8), the projections here are irregular and diffuse. The training sets (left-blue) form dense, amorphous clouds with no clear directional flow or geometric structure, indicating a lack of internal organization in the reservoir's response to input signals. This suggests that the reservoirs fail to compress temporal information into meaningful latent representations. Similarly, the test sets (right-red) display convoluted, spiraling patterns, potentially hinting at over-saturation or unstable internal behavior, without evident alignment to task-informative structure.



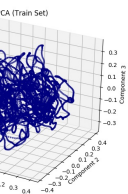
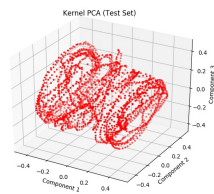
Random ESN – Example 1



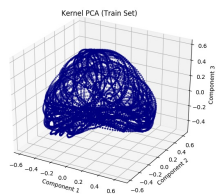
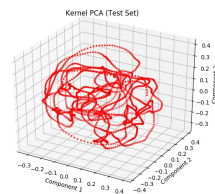
Random ESN – Example 2



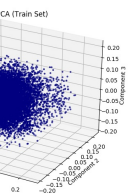
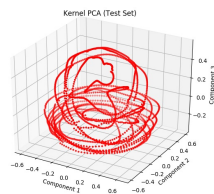
Random ESN – Example 3



Random ESN – Example 4



Random ESN – Example 5



Random ESN – Example 6

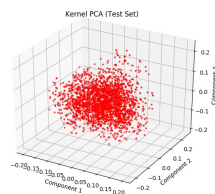


FIGURE 4.2.9: Random ESN initializations projected into 3D kernel PCA space. Each image shows training set dynamics (left) and test set dynamics (right) for a given random reservoir. Examples demonstrate chaotic, unstructured behavior.

In Table 19, we summarize how reservoir dynamics differ between optimized and random ESNs based on the kPCA projections shown above.

The optimized ESN displays a clear and smooth low-dimensional structure, indicative of stable and organized internal dynamics that are well-suited for capturing temporal patterns in the input data. This coherent structure suggests the reservoir has effectively internalized meaningful dependencies from the input signals, enabling better generalization despite some residual noise inherent in real-world financial data.

In contrast, random ESNs display scattered distributions with little visible structure, reflecting weak internal representations and poor task-specific organization. Such reservoirs often exhibit chaotic or over-saturated dynamics that struggle to encode input-output relationships effectively, resulting in limited generalization and reduced predictive accuracy. This disordered behavior indicates that ESN initializations lacking a structured state space are unable to generate the stable internal dynamics necessary for robust learning.

Importantly, the presence of some noise in the optimized ESN’s generalization performance (as illustrated in the right panel of Figure 4.2.8) is not necessarily negative. In real-world systems, data is often noisy, and the fact that the ESN can still generalize despite this noise suggests that it is robust. On the other hand, the random ESN’s lack of structure means it has no inherent noise resilience and performs poorly even under ideal conditions.

Feature	Optimized ESN	Random ESNs
Structure	Well-defined, smooth, continuous trajectories	Scattered, disordered, random blobs with no coherent path
Generalization	Good, with some (expected) noise in test set	No apparent generalization
Reservoir Behavior	Coherent, and dynamic	Erratic, possibly chaotic or over-saturated
Exploration of State Space	Explores regions of the state space smoothly	Random exploration with no meaningful progression
Stability	Stable state transitions	Unstable, with random jumps between states

TABLE 19: Interpretation of reservoir dynamics from kPCA projections for optimized and random ESNs.

To be clear, although the kPCA visualizations reveal structured behavior in the optimized ESN, we do not claim that these projections represent true attractors in the dynamical systems sense. Rather, they serve as post-hoc diagnostic tools that qualitatively illustrate how internal reservoir dynamics evolve in response to input

sequences. It is argued that such projections are indicative of rich, structured reservoir internal representations that exhibit just enough chaotic behavior to capture the complexity of the input, while maintaining sufficient regularity to ensure predictability and stability. This balance is critical; it suggests the reservoir operates in a functional regime where complex temporal dependencies can be modeled effectively without the system devolving into erratic or unstable behavior. The small positive LLE observed supports the interpretation that the reservoir operates within a controlled chaotic regime, at the edge of chaos, structured enough to efficiently encode temporal patterns, yet bounded enough to ensure reliability.

This detailed analysis of the reservoirs' internal dynamics aims to distinguish deterministic chaos from noise. While both phenomena can produce seemingly erratic output, their origins and implications differ fundamentally. Deterministic chaos arises from nonlinear interactions within the system, leading to sensitive dependence on initial conditions and long-term unpredictability, yet it remains governed by fixed rules. Noise, on the other hand, reflects external randomness without any underlying structure. We aim to identify patterns that suggest structured chaotic behavior as opposed to uncorrelated randomness. This distinction is essential not only for theoretical understanding but also for practical applications, as it informs the design of more robust and reliable reservoir computing architectures.

4.2.5 Concluding Remarks

This work introduces a hybrid framework that integrates the novel SACGA to optimize ESNs for one-step-ahead financial forecasting using high-frequency data. The approach is validated on a cleaned dataset of 411 stocks from the S&P 500, sampled at ten-minute intervals between May 6, 2019, and July 2, 2020, representing a comprehensive market proxy.

While ESNs excel in recurrent modeling, their internal reservoir dynamics often remain a black box. Addressing this gap, this contribution is twofold: not only does it evolve reservoirs via SACGA optimization with improved predictive power in a real-world scenario, but it also provides a novel lens through which reservoirs' behavior can be quantitatively and qualitatively evaluated post-hoc.

A key innovation is applying kPCA to reservoir states after readout training, offering a geometry-driven, variance-based approach to visualize and understand how temporal inputs are internally represented. This method complements established metrics like the Lyapunov Exponent and Separation Ratio, enriching the interpretability toolbox and providing deeper insight into reservoir stability, expressiveness, and structure. Table 20 below summarizes these key metrics alongside methodological notes on the design and behavior of the optimized ESN.

Beyond merely confirming better predictive performance, this study touches on a deeper objective: bringing transparency to black-box temporal models. The shift

Metric / Aspect	Details and Methodological Notes
Separation Ratio (SR), Test Set	$1 < \sim 7.587 < 10$, indicating strong input separability within the reservoir state space, with moderate signal amplification, allowing temporal dependencies to be effectively modeled
Largest Lyapunov Exponent (LLE), Test Set	$\sim +0.00045$, indicating mild chaotic dynamics, within the edge of chaos regime ($[-0.001, +0.001]$)
Kernel PCA (Train Set)	Reveals a well-defined, smooth, and coherent low-dimensional structure
Kernel PCA (Test Set)	Shows a fragmented yet structured trajectory with nontrivial topology
Design Methodology	ESN evolved via the novel Self-Adaptive Cuckoo-Genetic Algorithm, guided by reservoir stability and representational expressivity, in addition to the standard error minimization criterion

TABLE 20: Summary of Metrics and Design Methodology of the Optimized ESN

from noisy, unstructured activity to smooth, organized manifold structures offers a visual language for understanding and evaluating reservoir behavior. Tools like kPCA provide designers with insight into the system's internal state space, allowing qualitative assessment of how well a reservoir responds to temporal patterns, potentially even before readout training. In this way, kPCA projections can serve as valuable diagnostic tools during model development. This approach may inspire further research leveraging manifold learning and state-space geometry to enhance and explain recurrent neural network behavior, advancing explainable models that not only perform well, but whose behavior we can visualize, reason about, and trust.

In summary, this work not only proposes a robust optimization framework for ESNs but also shifts the perspective from treating them as black-box systems toward viewing them as interpretable dynamical machines. The combination of quantitative stability metrics and geometric insights enables deeper understanding and trust in reservoir computing. Future research could extend the SACGA optimized ESN framework by incorporating additional input features (e.g., sentiment, macroeconomic indicators, cross-market signals), benchmarking against alternative DL architectures, and initializing the SACGA method with larger populations. It is hoped that this framework inspires further work at the intersection of optimization, interpretability, and recurrent dynamics.

Chapter 5

Conclusion and Future Directions

Research Overview

This research contributes to the intersection of deep learning and evolutionary optimization, with a focus on ESNs, an emerging yet opaque architecture known for its simplicity and strong predictive potential in financial forecasting. A comprehensive review of recent deep learning applications (2020-2024) in financial forecasting highlights ESNs for their straightforward design and exceptional forecasting performance. However, achieving optimal, task-specific ESN design remains tightly coupled with effective model optimization.

A key limitation in existing ESN optimization methods is their simplification of the search space, and more importantly, their narrow focus on minimizing predictive error, often neglecting the inherent dynamic quality of the ESN's reservoir itself. This can be detrimental for predictive performance, as the richness and complexity of the reservoir's internal dynamics are essential for effective input representation and generalization.

To address these challenges, we propose a novel optimization algorithm, the Self-Adaptive Cuckoo Genetic Algorithm (SACGA), specifically designed for ESN optimization. Unlike traditional approaches, the proposed method incorporates both predictive error minimization and reservoir quality metrics into the fitness evaluation function. By embedding a self-adaptive mutation rate mechanism and Cuckoo-inspired offspring addition, periodically, the algorithm enhances its exploratory capabilities, leading to more robust and efficient convergence in complex search spaces.

Empirical evaluations were conducted on two distinct settings: benchmark chaotic datasets and real-world financial time series. Results consistently showed that ESNs optimized via SACGA outperformed baseline models, both in predictive accuracy and robustness. On standard chaotic systems such as the Multiple Superimposed Oscillator, Mackey-Glass, and NARMA10 time series, the SACGA optimized ESNs achieved consistently lower error rates, across RMSE, MAE, and MAPE metrics, compared to baseline GAs with fixed or deterministic mutation strategies.

These improvements reflect enhanced model convergence and reliability, particularly in handling temporal dependencies and nonlinear dynamics across varying datasets. In the financial forecasting scenario, the model showed strong directional accuracy, significantly outperforming a Random Walk benchmark. The SACGA optimized ESN demonstrated an ability to reliably detect market trends, with a balanced capacity to identify both upward and downward price movements, thus enabling practical improvements in trade timing and risk management.

Beyond predictive performance, this research also addresses the interpretability of reservoir dynamics. Chaos-related metrics (including Lyapunov Exponents, and separation ratio) along with kernel Principal Component Analysis (kPCA) are employed to analyze the internal dynamics of ESNs. One key insight is that the emergence of discernible structures in kPCA-projected state trajectories can serve as a reliable proxy for reservoir quality. Although used here as a post-hoc diagnostic tool, such visual and geometric insights could pave the way for pre-training diagnostics, allowing for the identification of poorly initialized reservoirs early in the modeling pipeline, thus guiding more efficient model development.

By advancing optimization techniques and interpretability frameworks, this work sets the foundation for more reliable and transparent ESN-based financial forecasting, bridging gaps between theoretical innovation and practical application.

Key Novel Contributions

This work has achieved several notable outcomes, advancing the state of research at the intersection of reservoir computing, evolutionary optimization, and financial forecasting:

- **Systematic Literature Review:** Delivered a comprehensive review of 187 Scopus-indexed deep learning studies (2020–2024) in financial forecasting, categorizing modeling approaches by domain and highlighting open challenges, model preferences, and future directions.
- **New View on Reservoir Dynamics:** Provided a fresh perspective on ESN architecture by investigating its intrinsic chaotic behavior, not as a flaw to be feared, but as a useful resource to enhance input representation and forecasting reliability.
- **Algorithmic Innovation:** Developed and successfully applied the Self-Adaptive Cuckoo Genetic Algorithm (SACGA) specifically for ESNs, a novel hybrid chaos-aware optimizer that jointly optimizes predictive error and enhances intrinsic reservoir dynamics, leading to more robust and reliable ESN performance.

- **Demonstrated Efficacy:** Showed improved forecasting accuracy and robustness on benchmark chaotic datasets and real-world financial time series, outperforming traditional GAs and random walk baselines across multiple metrics.
- **Theoretical Insights and Interpretability:** Employed chaos-theoretic and topological tools to better understand and interpret ESN dynamics, uncovering structural properties related to system stability, memory, and separability that underpin strong generalization.

Limitations and Future Directions

While this research presents promising results and contributions, it also opens several avenues for further investigation and improvement. These are summarized below:

1. Application-Oriented Refinements

a. Model Design Enhancements

- Exploring ESN architectures with output feedback could improve the model's capacity to learn complex temporal dependencies, by allowing past predictions to influence future reservoir dynamics.
- Leveraging reservoir plasticity represents a promising extension. In the current setting, a non-plastic reservoir is employed, where the internal weights and neuron parameters remain fixed after the random initialization. Introducing intrinsic plasticity, which adapts neuron-level properties such as gain and bias to improve information encoding, could enhance the quality of internal representations. In parallel, incorporating synaptic plasticity, where internal reservoir connections are updated according to activity-dependent learning rules (e.g., Hebbian), would allow the reservoir to improve its temporal memory and adaptability. Together, these mechanisms could make the ESN more responsive to structural changes and regime shifts commonly observed in financial markets.
- Incorporating a broader set of input features, such as sentiment indicators, technical features, macroeconomic variables, and behavioral data, could enhance predictive accuracy. In particular, the inclusion of changepoint features may further improve performance by enabling the model to account for structural breaks and regime shifts in financial time series.
- Future implementations of the SACGA algorithm could benefit from exploring alternative population and generation configurations beyond the settings adopted here (e.g., $(pop, gen) = (1000, 50)$), which may further improve diversity and the probability of identifying high-quality optima, albeit at increased computational cost. Additionally, a sensitivity analysis of the most influential ESN hyperparameters, which is not included in this work, is an important direction for future research, as it

would clarify robustness and guide more principled, task-specific selection of hyperparameters and their search ranges.

- Financial price series are typically non-stationary in levels and often integrated of order one ($I(1)$). Modeling such series directly can lead to spurious correlations, unstable forecasts, and poor generalization, because trends and drifts may be misinterpreted as predictive patterns. In this work, this issue is mitigated by using returns of the constructed Composite Index at ten-minute intervals, which reduces non-stationary behavior while retaining meaningful temporal dependencies. While returns may still exhibit time-varying volatility, heavy tails, and regime shifts, strict stationarity is not fully required for ESNs. Their high-dimensional, nonlinear reservoir effectively captures temporal patterns and latent dynamics in the input, enabling robust predictive performance even with non-stationary inputs. This is one of the practical strengths of ESNs in forecasting applications. Unlike other recurrent architectures that often require preprocessing to handle non-stationarity (e.g., MEMD-LSTM [88], GRU-CEEMDAN-Wavelet [301], VMD-LSTM [350], multiWDN-LSTM [401]), ESNs can model raw time series directly. Nevertheless, while decomposition is not strictly necessary, its potential to improve ESN performance should be acknowledged for future work. For instance, AD-DeepESN [204] applies Additive Decomposition of trend, seasonality, and residuals together with the original raw series as inputs to a Deep ESN, achieving superior performance on highly non-stationary datasets compared to traditional recurrent and reservoir computing models.

b. Financial Forecasting Extensions

- Outperforming the Random Walk benchmark is a critical step in challenging the EMH, particularly in financial time series where unpredictability is the norm. Accordingly, this study adopts the Random Walk as a rigorous and economically meaningful baseline. The present thesis prioritizes understanding of reservoir dynamics, emphasizing when and why predictive gains emerge rather than empirical dominance over alternative DL architectures. Nevertheless, comparisons with state-of-the-art DL models, such as LSTM or GRU, constitute an important direction for future research, to further validate the predictive advantages of the SACGA-optimized ESN in broader modeling contexts.
- Applying the SACGA-optimized ESN to cryptocurrency markets represents a promising yet challenging direction for future research. Cryptocurrency markets remain inefficient and exhibit persistence [64], suggesting opportunities for exploitable predictability, and offering a suitable setting to assess whether the proposed framework can outperform the Random Walk baseline and challenge the EMH. At the same time, these markets are dominated by pronounced non-stationarity, frequent

structural breaks, and high volatility, implying that static model calibrations may quickly become outdated. Thus, the ESN configuration must employ an adaptive learning architecture to remain effective over time. Accordingly, adaptive strategies, such as rolling-window retraining, online learning, reservoir plasticity, and periodic SACGA-based re-optimization of ESN hyperparameters, become central components of the framework. Beyond price histories, incorporating on-chain activity, order-flow, changepoint indicators, and market-wide sentiment into the reservoir state may further enhance the model's ability to track evolving information structures in decentralized markets.

- Rolling-window evaluation is a standard approach in time series forecasting, as it better reflects real-time learning under evolving market conditions. In this study, a fixed train-validation-test split is deliberately adopted to enable controlled and consistent comparisons across models and to support a detailed analysis of the internal dynamics of the SACGA-optimized ESN. Implementing a rolling-window framework would require repeated SACGA-based ESN hyperparameter optimization at each window update, substantially increasing computational cost and complicating the analysis of reservoir dynamics. Nevertheless, rolling-window evaluation remains a more realistic assessment for practical deployment and represents a valuable direction for future research, particularly in highly non-stationary financial environments.
- A limitation of the current study is that formal statistical significance tests (e.g., Wilcoxon signed-rank or t-tests) were not conducted in Section (4.1). To ensure robustness against variability, predictions for each GA method on the test set were obtained using 32 independently initialized ESN models, each employing the same optimized hyperparameters, following the approach of [241]. The reported results represent the average performance with standard deviations across these runs. Notably, the improvements of SACGA over the baseline methods are substantially larger than the observed standard deviations, indicating consistent and robust gains. Future work will focus on including formal statistical significance testing to further strengthen these conclusions.
- Structural breaks and regime shifts in financial time series can challenge predictive models. The proposed SACGA-optimized ESN demonstrates robust predictive performance, and well-defined, ordered internal reservoir dynamics supporting reliable temporal encoding even under evolving market conditions. From a dynamical systems perspective, the reservoir acts as a high-dimensional mapping of past inputs, capturing latent patterns that provide the capacity to accommodate changes in the underlying system. However, because the current framework relies on a fixed reservoir configuration optimized on historical data, sudden changes in market structure (e.g., volatility bursts, crises, policy interventions, or bull/bear regime transitions) may temporarily degrade predictive performance. To further enhance adaptability, the framework can be extended with change-point detection to identify regime transi-

tions, rolling-window training, periodic SACGA-based re-optimization of reservoir hyperparameters, and online learning. Additionally, leveraging intrinsic and synaptic plasticity could enable the reservoir to dynamically adjust its internal state, and better track evolving market dynamics.

- Multi-step-ahead forecasting is an important extension of the SACGA-optimized ESN framework. It can be implemented via recursive forecasting, where one-step predictions are fed back as inputs to the reservoir to generate subsequent horizons, or via a direct multi-output strategy, in which the readout layer predicts multiple future values simultaneously. While these approaches allow forecasting over longer horizons, they introduce practical challenges: recursive methods are prone to error propagation as prediction noise accumulates across steps, whereas multi-output strategies increase the dimensionality of the output layer and may require additional data and careful regularization to remain stable. Addressing these issues requires careful design of the readout layer and consideration of stability, particularly when forecasting further into the future. Incorporating multi-step forecasting would broaden the framework's applicability to real-world financial time series, and can be easily implemented within the existing SACGA-optimized ESN framework.

c. Trade Timing and Decision-Making

- The directional forecasting capacity of the SACGA optimized ESN indicates practical utility in trend-following systems, where the proposed approach could serve as a robust decision-making engine. However, further calibration is required to manage asymmetric prediction errors, particularly in controlling for overreactive upward signals that could lead to excessive trading or misallocated capital.

2. Theoretical Advances

a. Reservoir Interpretability

- Having analyzed the reservoir's internal representations both pre- and post-training, a valuable future direction is to compare the Lyapunov Exponents of these reservoir states (pre-training) with those of the target stock prices, to assess how well the reservoir captures and replicates the underlying market dynamics.
- Interpretability techniques such as SHAP values could be applied to understand the internal workings of the reservoir. This would improve transparency and help explain how different inputs influence the ESN's outputs.
- Using the Hurst exponent, which shows how much a time series tends to remember its past, to evaluate the ESN's internal states could provide valuable insights into how well the network captures long-term dependencies and market persistence. This could guide better tuning of reservoir memory.

b. Topological Analysis

- Chen et al. [72] examine reservoir computing from a geometric angle by analyzing metrics such as correlation dimension, multiscale entropy, and memory effects. Their findings reveal that trained reservoirs preserve these key geometric characteristics of the target chaotic system, suggesting a form of topological conjugacy between the reservoir and the observed dynamical system. In essence, despite deviations in predicted trajectories from the original system, the underlying structural organization of the system remains intact in the trained reservoir.

Building on these insights, a promising direction for future research is to investigate whether such geometric features can be recovered when training reservoirs on real-world financial time series. Demonstrating topological conjugacy in this context would provide compelling evidence that the reservoir not only models the surface-level behavior of financial markets but also captures deeper structural dynamics. Such a discovery could significantly enhance the theoretical understanding of financial modeling using reservoir computing.

c. Synchronization and Energy Efficiency

- A promising research direction is to explore the relationship between synchronization and energy efficiency in ESNs. Prior work indicates that stronger internal synchronization, achieved through higher coupling strengths, can lead to reduced energy consumption in neural systems- "the stronger the neural network synchronization was, the less energy the network consumed"[228]. This suggests that carefully tuning coupling strength may enhance energy efficiency.

Extending this idea, future studies could examine whether promoting synchronization not only within the reservoir (internal synchronization) but also between the reservoir and the target system (system-reservoir synchronization), for example, via joint recurrence plots, enhances both predictive performance and energy efficiency. The SACGA may be leveraged to optimize performance while promoting such synchrony, potentially leading to the design of energy-aware ESNs suitable for low-power or real-time forecasting applications.

Appendices

.1 Technical Summary of Reviewed Studies for Chapter 2

TABLE 21: Stock Price Forecasting: Standalone Models

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[361]	Kweichow Moutai stock	2009-2011	OHLCV, Adj. Close	ESN, LSTM	RMSE, MAE	Python
[215]	4 stocks in NASDAQ	2010-2022	Close price data	LSTM, GRU	MAPE, RM-SPE, RMDPE	Python
[173]	Apple stock	-	Price data, macroeconomic indicators, technical indicators	LSTM, GRU	MSE, MAE, Acc, Loss function	Keras, PyQt5, yfinance
[190]	AAPL, TSLA, ROKU, BAC, GBP/USD, US-D/SEK, SQQQ, SPXS	2017-2021	Close data	VRNN, LSTM, GRU, CNN	RMSE, MAPE	Python, Keras (Chollet 2015)
[271]	200 stocks in Tehran stock exchange	2015-2019	Returns data	LSTM, DQN, RF, SVR	MSE, RMSE, MAE, MAPE, R^2	Python, Excel
[121]	Bank Central Asia stock	2004-2024	OHLCV, Adj. Close	LSTM	MSE, MAE	-
[182]	4 stocks in Tadawul	2019-2023	Close price data	LSTM, ARIMA	MSE, RMSE, MAE	Python

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 22: Stock Price Forecasting: Standalone Models (Continued)

Art.	Dataset	Period	Feature Set		Method	Performance Criteria	Env.
[55]	Apple, Amazon, Google, Tesla, Netflix, BEX-IMCO Pharmaceutical stocks	-	OHLC		EMA, SMA, LSTM	RMSE, MAPE	-
[339]	5 technology stocks in NASDAQGS and NYSE	2005-2022	OHLCV, Close	Adj.	RF, XGBoost, Bagging, Ada Boost, Extra Trees, Voting Classifiers, LSTM, ARIMA, Hidden Markov Model	Accuracy, Precision, Recall, F1-Score, Specificity, AUC	Python, Sklearn
[383]	Axis Bank, HDFC Bank stocks	2000-2020	Close data	price	LSTM	RMSE	Keras, Tensor-flow
[324]	Apple, Google, Amazon, Tesla stocks, Bitcoin, Ethereum, Bitcoin Cash	2017-2019	OHLV, Close	Adj.	GAN	t-SNE, Kullback-Leibler divergence, discriminative score, predictive score	-
[6]	stocks and options in JPX	2017-2022	OHLCV		LSTM, SVR	MSE, RMSE, MAE	-

TABLE 23: Stock Price Forecasting: Novel Approaches

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[336]	30 Stocks in DJI	1997-2017	stock bar chart images	CNN-BI	Trade statistics	Keras, Tensor-flow
[423]	ROTA, PTSR, PER financial indicators of 2 US stocks	2014-2018	-	LSTM, GRU	MAE	Python, Keras, Tensor-flow
[264]	Samsung stock	2016-2021	Close data	MLS LSTM	Normalized RMSE, MAPE, R^2	Python
[123]	3 stocks in NSE	2000-2021	OHLCV	LNN, BiLTC	MSE, RMSE, R^2	keras-ncp
[434]	Apple, AMD, Microsoft, Google, S&P500	2014-2019	Price data	LSTM	MSE, MAPE, CBPM	Python

TABLE 24: Stock Price Forecasting: Technical & Sentiment Analysis-Based Models

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[89]	7 stocks in NSE	2012-2023	OHLCV, technical indicators	LSTM	RMSE, MAPE, R^2	-
[343]	Intel, Indian Oil, NTPC, Citigroup stocks	2011-2022	OHLC, technical indicators	GRU, LSTM	RMSE, MAE, MAPE, R^2	Python, Tensor-Flow, Keras, NumPy, Pandas, Sklearn, Math
[293]	3 stocks in NSE	2000-2020	OHLCV, spread, technical indicators	Improved Recurrent Rider LSTM	MSE, RMSE	-
[366]	Walmart stock	-	OHLCV, Adj. Close, 15 technical indicators	CNN, 2D-CNN, LSTM, MLP	precision, recall, confusion matrix, F1-Score	-
[348]	Stocks in China's A-share	2014-2017	price data, research reports, investor sentiment	LSTM	accuracy, F1-Score	-
[355]	5 stocks in Nifty50	2009-2020	OHLCV, Adj. Close, Twitter sentiment	LSTM	RMSE	Keras

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 25: Stock Price Forecasting: Hybrid Models

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[277]	Stocks of Tesla, Google, Twitter	2017-2022	OHLC, Adj. Close	CNN-LSTM-GRU	RMSE, MAE	Python
[269]	Apple stock	2004-2023	OHLCV, Adj. Close, 15 technical indicators	CNN-LSTM	Confusion matrix, precision, recall, F1-Score, weighted F1-Score, Kappa score	-
[147]	22 stocks of 11 industries in China's stock market	2012-2022	OHLC, image features	KM-CAE-DLSTM	RMSE, R^2 , DA	-
[349]	Ping An Bank (China) stock	2018-2021	Price data	NR-LSTM	MSE, RMSE,	Python
	FTSE100	2019-2020			MAE, MAPE,	
	S&P500	2019-2020			SMAPE, R^2 ,	
	CSI300	2022			RMSPE	
[247]	SSE, SZSC, CSI300, 9 stocks in CSI300	2016 (1 st dataset), 2022 (2 nd dataset)	Close price data, OHC	hybrid LSTM and DB-ARIMA model	RMSE, MAPE, MAD	Python, Tensorflow, Statsmodels, fbprophet
[16]	Tesla stock	2014-2017	-	CNN-LSTM	MSE, RMSE,	MATLAB
	Apple stock	2010-2020		LSTM	NRMSE, R^2	
[4]	AAPL, AAL, CBMB, forex, cryptocurrency	2016-2020	Price data	CNN-LSTM	MSE, MAE	Python (3.8.3), Keras, Tensor-Flow, sklearn, Statsmodels
[2]	Bharti Airtel stock	2009-2023	OHLCV, change	CNN-LSTM	RMSE, MAE, R^2	Python
[213]	40 stocks in CAC40, CAC40	2019-2021	OHLC	KNN, LSTM, ARIMA	mean, max, min deviations, RMSE	Python, Tensorflow, Keras

TABLE 26: Stock Price Forecasting: Hybrid Models (Continued)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[278]	S&P BSE GREENEX Index, 22 con- stituent stocks	2012-2022, 2021-2023	OHLC	LSTM	RMSE, MAE	Python, Keras
[340]	245 stocks in KLSE	2019	Price data, TPMG gradi- ents	LSTM	RMSE, MAE, Residual Analysis, R^2	-
[400]	stocks in RMB market	2015-2020	OHLCV, trad- ing amount	AGNN	regression errors, economic indicators	-
[433]	PT Bank Central Asia Tbk stock data	2013-2023	Close price data	N-BEATS	MAPE	-
[106]	Google, Apple stocks, IXIC	-	OHLCV, in-day- change, trend	LSTM, GRU, CNN-LSTM, PatchTST	MSE, MAE	-

TABLE 27: Stock Price Forecasting: Hybrid Models with Data Pre-processing (Dimensionality Reduction, Decomposition, Optimization)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[91]	16 stocks in China's A-share	2010-2020	financial indicators	GPCA-GLSTM	RMSE, MAPE	Keras
[258]	TSLA NASDAQ S&P500	2010-2020 1999-2020 1999-2022	OHLCV, Adj. Close	LSTM	MSE, RMSE, MAPE, Correlation coefficient	
[149]	30 stocks in DJIA, DJIA	2018-2023	Price data	LSTM-ARO	MSE, MAE, MAPE, R^2	Python
[211]	18 stocks in LON, NSE, NYSE, BOM, NASDAQ, TSE	2018	OHLCV, 14 technical indicators	LRC with FPA-LSTM, LRC with PSO-LSTM	precision, recall, F1-Score, error rate	Python, Keras, Tensor-flow
[344]	stocks of Lloyds Bank Group, Barclays Bank, Royal Bank of Scotland	2009-2019	OHLC, technical indicators	Time series models, DL models	MAE, MAPE	Python, Keras
[287]	5 stocks in NSE, 5 stocks in BSE	2000-2021	OHLC, technical indicators	LSTM	MSE, RMSE, MAE, R^2	Python
[320]	6 stocks in NSE	Price data	-	CGAN	MSE, MAE, correlation	-
[184]	Microsoft, Amazon, General Electric stocks	1986-2024 1997-2024 1962-2024	OHLCV, Adj. Close	DES-DA-ED -Bi-GRU-BO	MSE, RMSE, MAE, R^2 , Theil's U	Keras
[234]	S&P500 DJI CMSB stock	2010-2013 2014-2017 2013-2016	Close data	EWT- dpLSTM- PSO-ORELM	RMSE, MAE, MAPE, SDE	MATLAB

TABLE 28: Stock Price Forecasting: Hybrid Models with Technical and Sentiment Analysis

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[70]	China Construction Bank stock, CSI300	2010-2020	Market, technical, financial factors	GA-LSTM	MSE	-
[379]	5 stocks in BIST100	2014-2024	Close price, 10 technical indicators	PCA-LSTM-CNN	MSE, MAE, MAPE, R^2	-
[107]	5 Stocks in China's A-share	2012-2021	OHLCV, technical indicators	DQN with LSTM	ROR, SR, SD	-
[438]	30 Stocks in China's A-share	2000-2020	OHLCV, turnover, technical indicators	CNN-Bi-LSTM	Profit, Return, Volatility, Sharpe Ratio, Retracement, MSE, RMSE, MAE, MAPE, SMAPE, R^2	-
[399]	Amazon, ORCL, Apple, EBAY, INTC, Alibaba Group, QCOM stocks	2019	OHLCV, Adj. Close, technical indicators	AUDA-Net ¹	Acc	-
[236]	20 stocks in A-shares, US, UK, Hong Kong markets	2012-2022	OHLCV, change, technical indicators	GAN-TrellisNet	MSE, RMSE, MAE, MAPE	Python
[207]	6 stocks in TWSE	2015-2020	OHLCV, news sentiment, forum posts sentiment	LSTM	RMSE	Python
[66]	Apple, PepsiCo, NRG, AT&T, APEI stocks	2008-2016	Close data, news head-lines	Word2vec-LSTM	Acc	Python

¹Adversarial Unsupervised Domain Adaptation Network

TABLE 29: Stock Price Forecasting: Hybrid Models with Technical and Sentiment Analysis (Continued)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[74]	85 stocks in S&P500	2014-2016	Price data (OHLCV), Twitter sentiment	hybrid information mixing model	accuracy, precision, recall, specificity, F1-Score, MCC	-
[212]	Apple, Microsoft, Intel stocks	2010-2020	OHCV, Adj. Close, technical indicators, tweets, news sentiment	ABC-LSTM	RMSE, MAPE	Python, Apache Hadoop
[104]	Dow Jones 30 stocks	2018-2019	Social Information, Embedded Text, Sentiment, User Score, Market Data (OHLCV, Adj. Close)	User2Vec	Accuracy, precision, recall, F-score	-
[189]	Apple stock	2013-2018	OHLCV, sentiment	S-EMD-AttentionLSTM	RMSE, MAE, MAPE, R^2	Python
[172]	Apple stock	2021	Price data (OHLCV), Twitter sentiment	FMHP ² +ML/DLR	MSE, MAPE, Acc	-
[69]	12 stocks	2021, 2022	OHLCV, Adj. Close, technical indicators, news sentiment	LSTM	Precision, Recall, F1-Score, Profit	Python, Keras

²Fully Modified Hodrick-Prescott filter

TABLE 30: Stock Price Forecasting: Hybrid Models – Novel Approaches

Art.	Dataset	Period	Feature Set		Method	Performance Criteria	Env.
[318]	Amazon, Apple, Microsoft stock	2012-2023	OHLCV, Adj. Close		VAR-LSTM	MSE, RMSE, MAPE, Acc	Python
[223]	50 small market cap stocks	2022-2023	OHLCV		ODE-LSTM	RMSE, MAE, R^2	-
[23]	Tesla, IBM, Amazon, CSCO, Chinese stocks, SSE, Nifty50, US Commodity Index Fund	2017-2022	Close price data		GRU	MSE, RMSE, MAE, Profit	-
[225]	322 US stocks in NYSE, NASDAQ	2005-2017	OHLCV		LSTM, GRU, RNN	MSE, RMSE, MAE, MAPE, R^2	-
[315]	12 stocks in NASDAQ	2019	OHLCV		LSTM-VAE	RMSE, MAPE, R^2	Python
[432]	Order flow, limit order book data for 6 stocks in MICEX, NASDAQ	-	-		TFF-CL-GRU	Accuracy, precision, recall, F1-Score	-
[10]	3 stocks from China's A-share market, CAC40, FTSE100, KOSPI, NASDAQ100, N225, SET, S&P500, SSE	2010-2020	Close price data		ICE2DEMDL	RMSE, MAE, MAPE, R^2	-
[410] 188	3 stocks in SSCI, and SZCI, SSCI, SZCI	1990-2018	Close data		logistic regression, RF, KNN, SVM, LSTM, GRU	Accuracy, precision, recall, F1, AUC, NYR	-

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 31: Stock Price Forecasting: Hybrid Models – Novel Approaches (Continued))

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[298]	Intel Inc., IBM Inc. stocks	2003-2023	Price data	PSR- Transformer	RMSE, MAE, MAPE	Python
[246]	8 stocks in China's A-share	2017-2022	Return Data, correlation data	CNNBiLSTM- Attention	MSE, MAE, MAPE, R^2	-
[395]	PT Bank Central Asia Tbk stock data	2019-2024	OHLCV	MDSTFT	MRE, NMSE, NRMSE	PyTorch
[200]	Crude Oil Crude Oil DAX GDAXI HSI	1983-2018 2000-2021 2014-2020 2000-2021 2000-2021	OHLCV	BiCuDNNLSTM -1dCNN	RMSE, MAE	Python, Keras, Tensorflow
[226]	SZSE SSE50 CSI300, IXIC, DAX, 4 stocks in NASDAQ	1991-2021 2004-2021 2002-2021	OHLC, prior close price, change, change rate	AE-ACG	MSE, RMSE, MAE, MAPE	Python, PyTorch, PyTorch Lightning
[307]	12 stocks in Bank Nifty	-	OHLCV, Adj. Close	Stacked Bi- LSTM and E-Optimized CNN	MSE, RMSE, MAE, R^2	Python
[416]	15 stocks from China A-share, Hong Kong, U.S. stock markets	2000-	Return price data	QGASF and CNN	MSE, MAE	yfinance, Tushare

TABLE 32: Index Price Forecasting: Standalone Models

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[363]	S%P500	2008-2018	OHLCV	LSTM	MSE, MAE	Python
[421]	CPI	2005-2021	CSI300, 7-day interbank offered rate, RMB/USD, monetary policy indicators, industrial activity, consumer demand, real estate metrics, wheat	GRU	MSE, MAPE, SMAPE	-
[14]	Tadawul	2018-2020	OHLCV, Change, Value-traded	MLP, LSTM	MSE, RMSE, NRMSE, R^2	Matlab
[371]	HSI, Corn future prices	2014-2019 2019	OHLCV, Close price data	LSTM	MAPE, Total Profit, SR, TR	-
[206]	TAIEX, FTSE	2000-2004 2010-2014	Price data	CNN	RMSE	Keras, Tensorflow
[110]	Nifty50	2014-2018	OHLCV	RNN, LSTM, CNN	MSE, RMSE, MAE, R^2	Python, Keras, Tensorflow, R, Excel
[427]	S&P500, HSI, N225	2015-2020	Returns price data	RW, ARIMA, RNN, LSTM, GRU, TCN	RMSE, MAE, MAPE, DA	Python, R, TensorFlow, Keras
[434]	S&P500, Apple, AMD, Microsoft, Google	2014-2019	Price data	LSTM	MSE, MAPE, CBPM	Python

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 33: Index Price Forecasting: Technical & Sentiment Analysis-Based Models

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[256]	Nifty50	2011-2019	OHLCV, technical indicators	RNN, LSTM, GRU	MSE, RMSE, MAE	Python, Keras, Theano
[419]	290 stocks in CSI300	2008-2019	CSI300, trading, technical, financial indicators	LSTM	Acc	-

TABLE 34: Index Price Forecasting: Hybrid Models

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[93]	KOSPI, KOSDAQ	2001-2019	OHLCV	P-LSTM	RMSE, MAE, MAPE	-
[231]	CSI300	2017-2019	Close data	ARIMA-LSTM, ARIMA-SVM	MSE, RMSE, MAE, MAPE, R^2 , RMSPE, Annualized Return, Sharpe Ratio, Return Volatility, Information Ratio	Python
[413]	S&P500	2010-2019	Adj. Close price	ARIMA-LSTM	MSE, RMSE, MAE	Python
[429]	Russell 2000 Index	2000-2019	Price data	DAE-LSTM	MSE	-
[247]	SSE, SZSC, CSI300, 9 stocks in CSI300	2016 (1 st dataset), 2022 (2 nd dataset)	Close price data, OHC	hybrid LSTM and DB-ARIMA model	RMSE, MAPE, MAD	Python, Tensorflow, Statsmodels, fbprophet
[349]	CSI300 FTSE100 S&P500 Ping An Bank (China) stock	2022 2019-2020 2019-2020 2018-2021	Price data	NR-LSTM	MSE, RMSE, MAE, MAPE, SMAPE, R^2 , RMSPE	Python
[186]	SSE CSI300	2011-2021 2009-2019	Close price data	ResNLS	MSE, RMSE, MAE, ARR	-
[258]	S&P500 NASDAQ TSLA	1999-2022 1999-2020 2010-2020	OHLCV, Adj. Close	LSTM	MSE, RMSE, MAPE, Correlation coefficient	
[135]	S&P500	2010-2019	Price data	LSTM, CNN-LSTM, ESN	RMSE, MAE, trend prediction accuracy, training time	Matlab, Python, TensorFlow
[22]	SSE	1997-2022	OHLCV, Adj. Close	CNN-LSTM-RNN	RMSE, MAE, R^2	Python, Keras, Numpy, Pandas

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 35: Index Price Forecasting: Hybrid Models (Continued)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[393]	SCI, HSI, NASDAQ, DJIA, SZ	1992-2022	OHLCV, Turnover, Rise & Fall, Rate, Close price data	PCA-IGRU	MAE, DS, R^2	Python, Keras, Tensor-Flow
[389]	S&P500	2000-2021	Price data	LSTM	RMSE, MPE, MAPE, MDAPE, R^2	Python (PyTorch, Seaborn, NumPy, Pandas, Matplotlib) and R (crypto2)
[94]	S&P500, Nifty50, SCI, IN-R/USD, JPY/USD, SGD/USD, Crude Oil, Gold, Soya Beans	2014-2020	Price data	Chaos-LSTM-PR	MSE, Dstat, Theil's U	Python, Gretl, R
[97]	S&P500, Nifty50, SCI, IN-R/USD, JPY/USD, SGD/USD, Crude Oil Price, Gold Price, Soya Beans Price	1990-2021	Price data	Chaos-LSTM-PR	MSE, MAPE, Dstat, Theil's U	Python, Gretl, R
[95]	S&P500, Nifty50, SSECI, IN-R/USD, JPY/USD, SGD/USD, Gold, Crude Oil, Soya beans	1990-2021	chaotic modeled time series	Chaos-CNN-PR	MSE, MAPE, Dstat, Theil's U	Python, Gretl, R

TABLE 36: Index Price Forecasting: Hybrid Models (Continued)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[213]	CAC40, 40 stocks in CAC40	2019-2021	OHLC	KNN, LSTM, ARIMA	mean, max, min deviations, RMSE	Python, Keras, Tensorflow
[43]	NSE, BSE, NYSE, DJI, N225, S&P500, NASDAQ	2010-2020	Close data	LSTM	MAPE	Python, Keras, Numpy, Pandas
[20]	FCHI, GDAXI, FTSE, SSMI, N100, BTC/USD	2018-2021	Price data	ARIMA, ETS-ANN, kNN	RMSE, MAE, MAPE	-
[404]	JSX, HSX, N225, STI	2020	Price data	CNN, LSTM, Stacked-LSTM, CNN-LSTM, ConvLSTM	RMSE, MAPE	Python
[278]	S&P BSE GREENEX Index, 22 constituent stocks	2012-2022, 2021-2023	OHLC	LSTM	RMSE, MAE	Python, Keras
[40]	S&P500	2000-2015	GAF images	CNN	Accuracy, Coverage, Annualized Sharpe Ratio, Net Profit, Sortino Ratio, Return Over Maximum Drawdown (RoMaD)	-
[372]	S&P500, CAC40, 32 stocks in MASI	2010-2019, 2010-2019, 2010-2021	Price data	LSTM, GRU	MSE, RMSE, MAPE, return-oriented metrics	Keras, pandas, Investpy

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 37: Index Price Forecasting: Hybrid Models (Continued)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[272]	NYSE	1995-	Close price data, anomalous events data	LSTM	MSE, MAE, MAPE, Error spread	Keras, Tensor-Flow
[411]	379 stock indices from different industries in China's market	2012-2020	OHLC	MLP, RNN, LSTM, GRU	Loss, Accuracy, F1-Score	-
[26]	S&P500, DJIA, NASDAQ100,	1999-2019	Close price data	LSTM, Stacked-LSTM,	RMSE, MAE	Keras, Tensor-flow
	EUR/USD, EU-R/GBP, EUR/JPY	2003-2019		BiLSTM, Encoder-Decoder, ES-LSTM, TCN, N-BEATS		
[394]	SSECI	2014-2020	Price data	EINS, LSTM, GRU	MSE, R^2 , MAE	Python
[3]	Halal tourism stock prices	2006-2021	WHTCSI, HTCSI	ARIMA, SVR, MLP, CNN, LSTM	RMSE, MAE, MAPE	Python

TABLE 38: Index Price Forecasting: Hybrid Models with Data Pre-processing (Dimensionality Reduction, Decomposition, Optimization)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[106]	IXIC, Google, Apple stocks	-	OHLCV, inday-change, trend	LSTM, GRU, CNN-LSTM, PatchTST	MSE, MAE	-
[350]	ChiNext	2019-2020	Price data	VMD-LSTM	MSE, RMSE, MAE, MAPE, SMAPE	Python
[401]	SSE, HSI	1991-2020	OHLCV, turnover, ups and downs, change	mWDN-LSTM	RMSE, MAE, R^2	-
[151]	SSE	2011-2018	OHLCV, technical indicators, close price data of DJIA, S&P500, HSI, SX5R, NI225	EEMD-CNN-GRU	MAPE, Theil	-
[310]	S&P500, DJI, DAX, N225	2010-2019	Close price data	EMD-CNN-LSTM, CEEMD-CNN-LSTM	RMSE, MAE, MAPE	-
[17]	KSE100	2015-2022	Close price data	Akima-EMD-LSTM	RMSE, MAE, MAPE	-
[418]	SSE, SZSE, GEM, HSI, DJI, S&P500	2010-2018	Close price data	CEEMD-PCA-LSTM	RMSE, NMSE, MAE, DS	-
[227]	SSEC, BSESN, SPX500, N225	2013-2023	Close price data	CEEMDAN-TCN-GRU-CBAM	RMSE, MAE, MAPE, R^2	Python, Tensor-Flow
[301]	S&P500, CSI300	2018-2021	OHLC	GRU-CEEMDAN-wavelet	MSE, RMSE, MAE, R^2	-
[420]	S&P500, DJI, DAX, N225, FTSE100, CAC40, S&P TSX, S&P IPC, BEL20, Hang Seng	1992-2021	Returns price data	CEEMDAN-CNN-Transformer	MSE, RMSE, MAE, MCS, time	Python, Keras

TABLE 39: Index Price Forecasting: Hybrid Models with Data Pre-processing (Dimensionality Reduction, Decomposition, Optimization) (Continued)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[230]	SSEC, SZI, SPX	1991-2023	Close price data	HDFM ³	RMSE, MAE, MAPE, R^2	Python, Keras, Tensor-flow
[10]	CAC40, FTSE100, KOSPI, NASDAQ100, N225, SET, S&P500, SSE, 3 stocks from China's A-share market	2010-2020	Close price data	ICE2DEMDL	RMSE, MAE, MAPE, R^2	-
[70]	CSI300, China Construction Bank stock	2010-2020	Market, technical, financial factors	GA-LSTM	MSE	-
[210]	S&P500, Sensex, Nifty50	2015-2021	OHLC, technical indicators	Adaptive PSO-LSTM	MSE, RMSE, MAAPE, SMAPE, Theil's U	Python, Matlab
[67]	Nifty50	2000-2021	Closing price data, technical indicators	PSO-LSTM, PSO-GRU	Accuracy, precision, recall, F1-Score	TensorFlow
[185]	XJO, DJI, IXIC, HSI, N225	2010-2020	Close data	IPSO-LSTM	RMSE, MAE, MAPE, R^2	Python, PyCharm Edu, TensorFlow
[149]	DJIA	2018-2023	Price data	LSTM-ARO	MSE, MAE, MAPE, R^2	Python
[319]	DJIA	-	Price data	LSTM-ARO	MSE, MAE, MAPE, R^2	Python
[234]	S&P500, DJI, CMSB stock	2010-2013 2014-2017 2013-2016	Close data	EWT-dpLSTM-PSO-ORELM	RMSE, MAE, MAPE, SDE	MATLAB
[88]	SPX, SSE, HSI	2010-2020	OHLCV	MEMD-LSTM	RMSE, MAPE, DS, DM test	Python
[392]	SSEC	2012-2022	OHLCV, turnover, technical indicators	SSA-PSO-LSTM	MSE, RMSE, MAE, MAPE, R^2 , Acc., Prec., Recall, F1-Score, AUC, Cumulative Profits.	Python, Tensor-flow, Keras

TABLE 40: Index Price Forecasting: Hybrid Models with Technical and Sentiment Analysis

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[422]	S&P500, NASDAQ, DJIA, RUSSELL, NYSE, DAX, N225, FTSE, CAC40, HSI, SSE	2010-2017	10 technical indicators	CNN3D-DR+LSTM	Precision, recall, accuracy, F-measure	Python, Keras, NumPy
[218]	S&P500, DJI, KOSPI200	2004-2018	OCV, technical indicators	HW-LSTM-RL	Return, Profit	Python, Keras
[425]	HSI	2003-2019	OHLCV, technical indicators of HSI/H&P500/FTSE100, China, UK, US economic indicators	AComNN	Acc	Tensorflow
[289]	S&P500, KOSPI200, SSE	2002-2018 2002-2018 2003-2018	OHLCV, technical indicators	LSTM-forest	RMSE, MAE, MAPE, Accuracy, balanced accuracy	Keras
[365]	SSE, IXIC, S&P500, DJI	2013-2020	SSE OHLC, IXIC, S&P500, DJI Index data, 10 technical indicators	Wavelet + LSTM	Accuracy, sensitivity, specificity, precision	Keras
[198]	DJI, DXY, FTSE, NDX, NI225, NIFTY, SSE, SPX, DAXEUR	-	Price data, technical indicator	H.BLSTM	RMSE, MAE, MAPE	Google Colab, Python
[325]	S&P500	2000-2017	OHLCV, Adj. Close, technical indicators	PCA-ICA-LSTM	MSE, RMSE, MAE, MAPE, R^2 , Max Error	Python, Keras
[222]	SSE, SZSC	2015-2019	OHLCV, rise and fall	CNN	RMSE, MAE, MAPE, Acc	-

³Hierarchical Decomposition-based Forecasting Model

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 41: Index Price Forecasting: Hybrid Models with Technical and Sentiment Analysis (Continued)

Art.	Dataset	Period	Feature Set		Method	Performance Criteria	Env.
[237]	S&P500, NYSE, DJI, NASDAQ, FTSE100, N225, SSE	2010-2019	Price data, news senti-ment		SA-TrellisNet	MSE, RMSE, MAE, MAPE, R^2	-
[300]	SET50	2014-2020	OHLCV, technical indicators, news head-lines	15	LSTM	Accuracy, precision, recall, F1-Score, re-returns	-
[130]	Nifty50, NIFTY Next 50, NIFTY Bank	2010-2021	Index (OHLC), technical indicators, News	data	CNN	Acc, ROC-AUC, Nor-malized cross-correlation	-

TABLE 42: Index Price Forecasting: Hybrid Models – Novel Approaches

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[154]	S&P500	1999-2019	Close data	CNN-LSTM	Acc	-
[8]	S&P500, SZSE, DJI, sales dataset	1994-2006, 2010-2021, 2012-2017, 2010-2021	OHLC	PLR-ALSTM- NN	RMSE, MAE, MAPE	Python, Keras, Tensorflow
[108]	S&P500, CSI300, SSE180	2008-2009 2017-2018 2018	Price data	adaptive LSTM-BN	Acc, returns ratio	PyTorch
[200]	Crude Oil Crude Oil DAX GDAXI HSI	1983-2018 2000-2021 2014-2020 2000-2021 2000-2021	OHLCV	BiCuDNNLSTM -1dCNN	RMSE, MAE	Python, Keras, Tensorflow
[242]	CSI300	2013-2016	OHLCV, amount ⁴	MFNN	Accuracy, Profitability	-
[23]	SSE, Nifty50, Tesla, IBM, Amazon, CSCO, Chinese stocks, US Com- modity Index Fund	2017-2022	Close price data	GRU	MSE, RMSE, MAE, Profit	-
[280]	S&P500, KOSPI, DJIA	2021-2022	Price data	ResNet	Precision, Recall, F1- Score	Python, Torch, Torchvi- sion, sklearn, Numpy
[226]	SZSE SSE50 CSI300, IXIC, DAX, 4 stocks in NASDAQ	1991-2021 2004-2021 2002-2021	OHLC, prior close price, change, change rate	AE-ACG	MSE, RMSE, MAE, MAPE	Python, PyTorch, PyTorch Lightning

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 43: Index Price Forecasting: Hybrid Models – Novel Approaches (Continued)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[259]	Daily data of Chinese stock index futures:IF, IH, IC	2015-2022	Close price data of stock indices futures and stock indices, risk-free interest rates, stock indices dividend yields	SF-Transformer	MAPE	-
[356]	10 stocks in FTSE MIB	2005-2021	OHLCV, technical indicators	DCGAN	RMSE, MAE, MAPE	Python, Gretl
[435]	S&P500, DJIA, IXIC, NYSE	2010-2019	Close price data	1D-CapsNet-LSTM	RMSE, MAE, MAPE, TIC	Python, TensorFlow, Keras
[410]	SSCI, SZCI, 3 stocks in SSCI, and SZCI	1990-2018	Close data	logistic regression, RE, KNN, SVM, LSTM, GRU	Accuracy, precision, recall, F1, AUC, NYR	-
[260]	DOW30, ETFs	2002-2016	15 technical indicators	CNN-TIC, EO-CNN-TIC, ResNet-TA	Precision, Recall, F1-Score	TA4J ⁵ , Swarm-PackagePy

⁴the amount of money traded in a security during each time step

⁵Technical Analysis for Java

TABLE 44: Forex Price Forecasting

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[268]	EUR/USD	2020	OHLC	Hybrid (CNN, CNN-LSTM)	Accuracy, precision, recall, F1-Score	Python, Keras
[390]	EUR/USD	2008–2018	Price data, technical indicators	XGBoost-LSTM	MSE, RMSE, MAE	Keras
[426]	EUR/USD	2013-2018	macroeconomic data, technical indicators	LSTM	profit accuracy, number of transactions	-
[377]	Japanese Yen exchange rates	2010-2011	Open, Close, technical indicators	CNN	Confusion matrix	Python, Keras, NVIDIA DIGITS
[116]	EUR/USD, USD/JPY, GBP/USD	2020-2022	Technical & fundamental features	Transformer	Trading Performance (No. of Trades, $r(\%)^6$, Sharpe Ratio)	-
[26]	EUR/USD, EU-R/GBP, EUR/JPY,	2003-2019	Close price data	LSTM, Stacked-LSTM,	RMSE, MAE	Keras, Tensorflow
	S&P500, DJIA, NASDAQ100	1999-2019		BiLSTM, Encoder-Decoder, ES-LSTM, TCN, N-BEATS		
[190]	GBP/USD, US-D/SEK, AAPL, TSLA, ROKU, BAC, SQQQ, SPXS	2017-2021	Close price data	VRNN, LSTM, GRU, CNN	RMSE, MAPE	Python, Keras (Chollet 2015)
[4]	EUR/USD, US-D/TRY, EU-R/GBP, stocks, cryptocurrency	2016-2020	Price data	CNN-LSTM	MSE, MAE	Python, Keras, Tensor-Flow, sklearn, Statsmodels

TABLE 45: Forex Price Forecasting (Continued)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[94]	INR/USD, JPY/USD, SGD/USD, S&P500, Nifty50, SCI, Crude Oil, Gold, Soya Beans	2014-2020	Price data	Chaos- LSTM-PR	MSE, Dstat, Theil's U	Python, Gretl, R
[95]	INR/USD, JPY/USD, SGD/USD, Gold, Crude Oil, Soya beans, S&P500, Nifty50, SSECI	1990-2021	chaotic mod- eled time se- ries	Chaos-CNN- PR	MSE, MAPE, Dstat, Theil's U	Python, Gretl, R
[97]	INR/USD, JPY/USD, SGD/USD, S&P500, Nifty50, SCI, Crude Oil, Gold, Soya Beans	1990-2021	Price data	Chaos- LSTM-PR	MSE, MAPE, Dstat, Theil's U	Python, Gretl, R

⁶Absolute return⁷Standard deviation

TABLE 46: Commodity Forecasting

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[152]	11 commodities	2017-2021	Price data	LSTM	MAE	PyTorch
[371]	Corn future prices, HSI	2019, 2014-2019	Close price data, OHLCV	LSTM	Total Profit, SR, TR, MAPE	-
[168]	Shanghai Copper futures, soybean meal futures, PTA futures,	2015-2020	OHLCV, accumulated position	CNN-GRU	MSE, RMSE, MAE	-
[23]	Tesla, IBM, Amazon, CSCO, Chinese stocks, SSE, Nifty50, US Commodity Index Fund	2017-2022	Close price data	GRU	MSE, RMSE, MAE, Profit	-
[409]	Corn future prices	2019-2023	Corn futures prices, news information, oil futures prices, hog price index, geopolitical risk index, 'corn' and 'corn price' Baidu search trends	JADE-TFT	RMSE, MAE, MAPE	Python

TABLE 47: Commodity Forecasting (Continued)

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[94]	Crude Oil, Gold, Soya Beans, IN-R/USD, JPY/USD, SGD/USD, S&P500, Nifty50, SCI	2014-2020	Price data	Chaos-LSTM-PR	MSE, Dstat, Theil's U	Python, Gretl, R
[97]	Crude Oil, Gold, Soya Beans, IN-R/USD, JPY/USD, SGD/USD, S&P500, Nifty50, SCI	1990-2021	Price data	Chaos-LSTM-PR	MSE, MAPE, Dstat, Theil's U	Python, Gretl, R
[95]	Crude Oil, Gold, Soya beans, S&P500, Nifty50, SSECI, IN-R/USD, JPY/USD, SGD/USD	1990-2021	chaotic modeled time series	Chaos-CNN-PR	MSE, MAPE, Dstat, Theil's U	Python, Gretl, R

TABLE 48: Bond Forecasting

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[299]	ten-year treasury bond yields of the China Bond market	2019-2023	economic fundamentals, macro monetary policy, money market prices, large asset allocation, bond trading sentiment	LSTM	MSE, MAE, R^2 , EVS	Python, RStudio
[414]	Securities Bank of China	2005-2021	Close data	LSTM, GRU, RNN	RMSE, MAE, MAPE, R^2 , accuracy	-

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 49: Cryptocurrency Price Forecasting: Standalone Models

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[99]	Bitcoin	2010-2019	Lag returns, volatility, transaction data, hash rate, money supply, financial indicators (e.g., VIX, S&P dollar returns, gold price), Google Trends, MACD, XRP price data	GRU, LSTM	RMSE	Python, Keras
[13]	Bitcoin	2021-2022	Price data	GRU, MLP	MSE, RMSE, PCC, R^2	Matlab
[273]	Bitcoin	2014-2022	OHLCV, Adj. Close	RNN, LSTM, GRU, BiLSTM, 1dCNN	MSE, RMSE, MAE, R^2	-
[374]	Bitcoin	2012-2020	OHLCV, Volume currency, Weighted price	ARIMA, LSTM, Prophet	RMSE, MAE	Python
[117]	EOS	2017-2022	Close data	LSTM	RMSE	Python, Keras
[58]	Bitcoin, Ethereum	2019-2020	Price data	LSTM	RMSE, MAPE, R^2	-
[18]	Bitcoin,	2014-2022	OHLCV,	BiLSTM, GRU	MSE, RMSE,	Python
	Ethereum, Cardano	2017-2022 2017-2022	Adj.Close		MAE, MAPE, R^2	
[138]	NEAR, Bitcoin	2020-2022	OHLCV	LSTM, GRU, CNN, CNN-LSTM	MSE, RMSE	Python, sklearn, Keras, Tensor-Flow,
[103]	Crypto,	2014-2023	OHLC	SGDRegressor, R^2		-
	CoinMarketCap	2013-2023		LR, RF regressor, adaptive boosting regressor, LSTM, ARIMA		

TABLE 50: Cryptocurrency Price Forecasting: Hybrid Models

Art.	Dataset	Period	Feature Set		Method	Performance Criteria	Env.
[291]	Litecoin,	2016-2020	Price, OHLCV		LSTM-GRU	MSE, RMSE,	-
	Monero	2015-2020			Hybrid	MAE, MAPE	
[199]	Bitcoin, Ethereum, Ripple	2021	Close price data		1dCNN-GRU	RMSE	-
[111]	Bitcoin, Ethereum, Ripple, Binance	2017-2022	OHLCV, Adj.Close		GRU-Bi-LSTM	RMSE, MAPE	Python
[4]	Bitcoin, Theta, VeChain, stocks, forex	2016-2020	Price data		CNN-LSTM	MSE, MAE	Python, Keras, Tensor- Flow, sklearn, Statsmod- els
[229]	Bitcoin, Bitcoin Cash, Ethereum	2018-2020	Price data		PSO-IDBN	MSE, RMSE, MAE	Matlab
	2 stocks in SSE GBP/USD, IBEX35	2015-2019 - -					
[20]	Bitcoin, FCHI, GDAXI, FTSE, SSMI, N100	2018-2021	Price data		ARIMA, ETS- ANN, kNN	RMSE, MAE, MAPE	-
[239]	Bitcoin, Ethereum, Ripple	2018-2019	Price data		Averaging, Bagging, Stacking	RMSE, accu- racy, AUC, F1	Python
[324]	Bitcoin, Ethereum, Bitcoin Cash, Apple, Google, Amazon, Tesla stocks	2017-2019	OHLCV, Close	Adj.	GAN	t-SNE, Kullback- Leibler divergence, discrimina- tive score, predictive score	-

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 51: Cryptocurrency Price Forecasting: Hybrid Models – Novel Approaches

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[7]	G-Research Crypto dataset	-	OHLCV, count, VWAP, target	CNN-LSTM- Transformer	MSE, RMSE, MAE, MSLE, R^2	Python
[187]	Bitcoin, Ethereum	2017-2020	Close data	VMD-AGRU- RESVMD- LSTM	RMSE, MAE, MAPE	-
[196]	Ethereum	2022-2023	transaction details	CNN-LSTM	MAE	PostgreSQL15, Python, Pandas, Keras, Tensor-flow
[238]	Bitcoin, Ethereum, Ripple	2017-2020	Price data	MICDL	RMSE, MAE, R^2 , accuracy, geometric mean, sensitivity, specificity, AUC	-
[292]	Bitcoin	-	Price data, Twitter sentiment	BiLSTM	MSE, RMSE, MAE	-
[194]	Bitcoin	2017-2023	OHLCV, technical indicators, news, social media, tweets, Google trends sentiment, Bitcoin's on-chain data, traditional assets price, macroeconomic data	SVR, XG-Boost Re-gressor, CNN, RNN, LSTM, BiLSTM, GRU, iTransformer, TSMixer	RMSE, MAE, MAPE, R^2	Python

TABLE 52: Volatility Forecasting

Art.	Dataset	Period	Feature Set	Method	Performance Criteria	Env.
[396]	Bitcoin, Ethereum, Litecoin, Ripple	2017-2022	Internal (OHLCV, trading count), External (technology, financial, policy uncertainty factors)	Random Forest, LSTM, GARCH	RMSE, MAPE, NMSE, DA	Python
[12]	PSX-100	2009-2021	Return price data	LSTM	MSE, RMSE, RMSPE	-
[297]	29 assets (DJI500+SPY)	2002-2008	Volatility, open-close	LSTM	MSE, QLIKE, PCC,	-
	92 assets in NASDAQ100	2012-2017	return		DM test, VaR, ES	
[235]	MSFT	1999-2018	OHLC,	GAN	AUC,	Keras,
	PAICC	2004-2018	technical		precision,	Tensorflow
	TSLA	2010-2021	indicators		recall, F1Score	
[11]	20 stocks in Bangladesh's frontier market	2018-2020	OHLC	Stacking Ensemble of Neural Network	RMSE, MAE	-
[25]	Apple stock	2018-2023	Price data	SARIMA-GARCH-CNN-BiLSTM	RMSE, MAE	-
[188]	stocks in Yahoo Finance Historical Stock Market Dataset	2000-2020	OHLCV	CNN-Transformer-Wavelet Transform	Prediction Acc., Identification Prec., Risk Recall, Comprehensive F1-Score	-
[90]	IPSA	2000-2021	Return price	GARCH +	MAE, MAPE,	R, Stochvol
	S&P500	1988-2019	data	LSTM/	RMSE, R^2 ,	
	ASX200	1992-2019		BiLSTM/	Spearman	
	Stochastic vol	2980 samples		GRU		
[193]	EUVIX, BPVIX, JYVIX	2010-2019	Volatility	autoencoder-LSTM	MSE, RMSE, MAE, MAPE, MPE	Keras
[388] 210	Gold	1968-2017	Return price data	CNN-LSTM	MSE	-

.1. Technical Summary of Reviewed Studies for Chapter 2

TABLE 53: Other Forecasting

Art.	Dataset	Period	Feature Set		Method	Performance Criteria	Env.
[249]	49 stocks in CSI100	2007-2015	Returns price data		DMLP, LSTM, CNN, SVR, RF, ARIMA	MSE, MAE, Hit Rate, Positive Hit Rate, Negative Hit Rate, Information Ratio, Maximum Drawdown, Turnover Rate	Keras, sklearn
[398]	FTSE 100	2009-2020	Adj. price	Close	LSTM	MSE, entropy	Python, Keras, Tensor-Flow
[439]	CAMBRIA, KRIRAN, SHARMA, JAMES	-	Price, volume data	Vol-	CNN-BiLSTM-attention	Accuracy, Recall, F1-Score, AUC, Parameters, Flops ⁸ , Training-Time, InferenceTime, MSE, RMSE, MAE, MAPE	-

⁸Floating Point Operations

.2 Derivation of Ridge Regression Solution

Ridge regression aims to find the weight vector $\mathbf{w}_{\text{out}}$ that minimizes the objective function:

$$\mathbf{w}_{\text{out}} = \underset{\mathbf{w}}{\text{argmin}} \|\mathbf{X}_{\text{train}}\mathbf{w} - \mathbf{Y}_{\text{train}}\|^2 + \alpha \|\mathbf{w}\|^2,$$

where $\alpha > 0$ is the regularization parameter controlling the trade-off between fitting the training data and shrinking the weights to prevent overfitting.

To find the minimizer, we take the gradient of the objective function with respect to $\mathbf{w}$ and set it to zero:

$$\nabla_{\mathbf{w}} (\|\mathbf{X}_{\text{train}}\mathbf{w} - \mathbf{Y}_{\text{train}}\|^2 + \alpha \|\mathbf{w}\|^2) = 0$$

$$\Leftrightarrow$$

$$2\mathbf{X}_{\text{train}}^T (\mathbf{X}_{\text{train}}\mathbf{w} - \mathbf{Y}_{\text{train}}) + 2\alpha\mathbf{w} = \mathbf{0}.$$

$$\Leftrightarrow$$

$$\mathbf{X}_{\text{train}}^T \mathbf{X}_{\text{train}}\mathbf{w} + \alpha\mathbf{w} = \mathbf{X}_{\text{train}}^T \mathbf{Y}_{\text{train}}$$

$$\Leftrightarrow$$

$$(\mathbf{X}_{\text{train}}^T \mathbf{X}_{\text{train}} + \alpha \mathbf{I})\mathbf{w} = \mathbf{X}_{\text{train}}^T \mathbf{Y}_{\text{train}},$$

where $\mathbf{I}$ is the identity matrix of appropriate size.

Solving for $\mathbf{w}$ yields the closed-form solution:

$$\mathbf{w}_{\text{out}} = (\mathbf{X}_{\text{train}}^T \mathbf{X}_{\text{train}} + \alpha \mathbf{I})^{-1} \mathbf{X}_{\text{train}}^T \mathbf{Y}_{\text{train}}, \quad (1)$$

which provides the optimal weights minimizing the regularized least squares problem.

.3 Technical Features

Let r_t denote the portfolio returns at time t . The following features are constructed:

- **Simple Moving Averages (SMA):**

$$\text{SMA}_k(t) = \frac{1}{k} \sum_{i=t-k+1}^t r_i$$

computed for window sizes $k = \{3, 6, 40, 120, 200\}$.

- **Exponential Moving Averages (EMA):**

$$\text{EMA}_k(t) = \alpha r_t + (1 - \alpha) \text{EMA}_k(t-1)$$

with smoothing factor $\alpha = \frac{2}{k+1}$, for $k = \{3, 6, 40, 120, 200\}$.

- **Squared Returns:**

$$r_t^2$$

- **Logarithmic Returns:**

$$\log(1 + r_t)$$

- **Absolute Returns:**

$$|r_t|$$

- **Rolling Volatility:**

$$\sigma_k(t) = \sqrt{\frac{1}{k-1} \sum_{i=t-k+1}^t (r_i - \bar{r}_k)^2}$$

where $\bar{r}_k$ is the mean return over window k , with $k = \{6, 40, 200\}$.

- **Cumulative Returns:**

$$\prod_{i=1}^t (1 + r_i)$$

- **Slope of Returns (via Rolling Linear Regression):** The slope β from fitting a linear model to returns over a rolling window of size 40:

$$r_{t-39}, \dots, r_t \rightarrow \beta_t$$

- **Lagged Returns:** Returns shifted by 1, 6, and 40 time steps:

$$r_{t-1}, \quad r_{t-6}, \quad r_{t-40}$$

- **Return-Volatility Interaction:** Product of return and rolling volatility (window 40):

$$r_t \times \sigma_{40}(t)$$

- **Volatility Change (Shock Detection):** First difference of rolling volatility (window 40):

$$\Delta\sigma_{40}(t) = \sigma_{40}(t) - \sigma_{40}(t-1)$$

- **Return Gap:** Difference between current return and return 40 steps ago:

$$r_t - r_{t-40}$$

- **Moving Average Convergence Divergence (MACD):**

$$\text{MACD} = \text{EMA}_{\text{fast}}(r_t) - \text{EMA}_{\text{slow}}(r_t)$$

with signal line and histogram computed accordingly. Windows used: fast = 40, slow = 840 (21*40, 1 month), signal = 200 (5*40, 1 week).

- **Bollinger Bands:** Upper and lower bands based on rolling mean and standard deviation (window 200), with band width determined by 2 standard deviations:

$$\text{BB}_{\text{High}} = \mu_{200} + 2\sigma_{200}, \quad \text{BB}_{\text{Low}} = \mu_{200} - 2\sigma_{200}$$

and percent band position:

$$\text{BB}_{\%} = \frac{r_t - \text{BB}_{\text{Low}}}{\text{BB}_{\text{High}} - \text{BB}_{\text{Low}}}$$

.4 Summary of the Random Parameter Sets for ESN Initialization for Chapter 4

No.	lr	ρ	ω	γ	N	s_{res}	s_{in}
1.	0.5035	4.9676	5.0411	6.0644	201	0.7268	0.1004
2.	0.9964	8.0632	5.0362	5.8451	420	0.0397	0.1605
3.	0.9607	2.3900	6.4059	0.6511	118	0.9293	0.9116
4.	0.6067	4.6238	7.5282	9.2858	68	0.3796	0.9626
5.	0.4069	4.9444	0.8630	3.2877	440	0.5835	0.3693
6.	0.7242	8.4241	4.1346	3.9129	62	0.7829	0.1960
7.	0.8550	5.2484	2.0225	0.8927	143	0.9284	0.4886
8.	0.2105	7.6520	9.6705	6.3264	279	0.8139	0.1734
9.	0.2084	8.3966	4.7490	0.4296	58	0.8068	0.7636
10.	0.6462	2.4060	1.0169	5.5829	421	0.9417	0.7999
11.	0.3827	1.8596	4.9379	4.5248	308	0.9570	0.4016
12.	0.8372	8.7630	8.4865	8.3733	146	0.4208	0.7528
13.	0.5910	8.8588	2.3700	8.3149	210	0.3972	0.3655
14.	0.3619	5.9263	0.7211	3.4428	88	0.4173	0.4704
15.	0.8105	4.7372	4.7972	3.8766	453	0.6938	0.4686
16.	0.2162	3.1776	5.3512	9.7126	79	0.7614	0.5091
17.	0.0738	8.6868	1.7834	0.7749	136	0.5237	0.8241
18.	0.7130	9.9063	3.5188	7.0186	449	0.9376	0.9432
19.	0.1353	3.2415	4.6573	1.0252	125	0.8467	0.5449
20.	0.6770	2.7397	4.4118	7.9604	389	0.8247	0.8885
21.	0.6722	4.7033	2.8813	0.9291	302	0.9030	0.4791
22.	0.8511	8.7085	2.6122	5.0584	367	0.8575	0.1807
23.	0.3753	8.4751	6.7386	6.5566	453	0.4876	0.6297
24.	0.4856	9.2654	5.5762	8.8275	238	0.7575	0.7969
25.	0.5950	2.9708	2.6848	8.2868	232	0.3344	0.3757
26.	0.2161	3.2573	2.7331	0.2729	369	0.2241	0.4584
27.	0.2043	2.7821	5.8473	9.1814	484	0.9296	0.0195
28.	0.3345	7.2167	6.4982	0.6595	427	0.2822	0.6704
29.	0.7398	4.3471	6.3964	5.3420	421	0.3323	0.8792
30.	0.5685	5.7177	2.4706	9.7044	268	0.4386	0.3917
31.	0.0793	2.3302	3.0537	0.7038	465	0.3927	0.4062

TABLE 54: First part of the 100 randomly generated individuals used for ESN initialization. Each individual consists of values for leaking rate (lr), spectral radius (ρ), shift scaling (ω), input scaling (γ), reservoir size (N), reservoir sparsity (s_{res}), and input sparsity (s_{in}).

Chapter 5. Conclusion and Future Directions

No.	lr	ρ	ω	γ	N	s_{res}	s_{in}
32.	0.0727	0.9564	3.4415	1.4208	221	0.1259	0.0597
33.	0.4103	3.7465	6.4787	6.6044	296	0.8475	0.2348
34.	0.2851	7.7742	6.7303	3.0757	440	0.1523	0.5410
35.	0.8507	0.8770	0.4876	2.3153	485	0.6282	0.5170
36.	0.4160	1.7439	8.3474	2.4735	232	0.4697	0.4544
37.	0.2202	6.6985	8.1911	7.9788	150	0.5763	0.9934
38.	0.0042	2.6047	7.7839	0.7379	115	0.0864	0.2472
39.	0.3538	3.8739	7.8516	7.8277	240	0.9907	0.9638
40.	0.7849	1.0317	6.7070	7.0499	498	0.8356	0.6397
41.	0.1618	4.4328	1.6751	5.1913	430	0.2324	0.1311
42.	0.3689	9.0094	2.4119	9.0277	386	0.1256	0.2899
43.	0.8603	0.6540	5.5976	0.4012	154	0.7117	0.8391
44.	0.3004	5.8441	8.3248	9.9459	213	0.6476	0.0330
45.	0.4061	5.9169	5.0774	3.7575	363	0.8121	0.7731
46.	0.9877	0.5433	9.4638	0.9673	253	0.0156	0.7345
47.	0.6477	6.0383	5.8382	1.3068	208	0.8256	0.2213
48.	0.4374	7.3109	7.5536	9.7259	235	0.4049	0.7710
49.	0.8784	6.8689	4.4759	2.6335	410	0.7584	0.3132
50.	0.4903	4.9299	4.0200	9.9450	361	0.9526	0.6170
51.	0.6413	9.1066	5.4562	7.8861	237	0.4326	0.9598
52.	0.8937	2.8802	1.5361	9.2434	191	0.9720	0.8941
53.	0.4563	0.0189	5.3746	0.7356	424	0.5319	0.7915
54.	0.6666	6.2294	8.8436	2.0668	257	0.5429	0.1593
55.	0.1709	4.4583	4.7117	6.4374	423	0.2013	0.5860
56.	0.9919	1.9371	2.0354	6.1253	81	0.5663	0.2688
57.	0.0616	0.1427	3.4865	3.3625	116	0.3483	0.7344
58.	0.5983	0.8721	5.4395	6.3175	487	0.2358	0.8274
59.	0.4132	8.8989	0.4439	3.2153	89	0.3756	0.7277
60.	0.5704	7.3604	7.6543	4.9291	253	0.6515	0.1917
61.	0.5873	0.8156	2.0539	8.0788	307	0.1146	0.3279
62.	0.4835	1.9291	7.2060	4.2555	301	0.5412	0.8569
63.	0.7598	3.4904	4.2083	2.1689	142	0.5381	0.2464
64.	0.5253	6.4446	7.2980	9.2252	150	0.3242	0.7908
65.	0.1413	9.7649	3.3994	1.6799	428	0.5212	0.3260
66.	0.8907	9.4799	4.8731	8.0907	278	0.5562	0.9626

TABLE 55: Second part of the 100 randomly generated individuals used for ESN initialization. See Table 54 for description of the hyperparameters.

.4. Summary of the Random Parameter Sets for ESN Initialization for Chapter 4

No.	lr	ρ	ω	γ	N	s_{res}	s_{in}
67.	0.7340	3.9454	9.8452	3.9734	164	0.4272	0.5205
68.	0.8060	5.7223	2.2740	3.4125	437	0.9353	0.0922
69.	0.3726	7.4560	1.7534	8.1227	206	0.9507	0.2289
70.	0.8172	1.6719	0.0580	5.8780	379	0.3868	0.0628
71.	0.6862	5.1072	4.5303	0.2938	500	0.8354	0.2584
72.	0.5023	0.8835	4.7595	9.8276	474	0.7473	0.9145
73.	0.7638	2.3847	7.1514	5.4450	48	0.6550	0.0846
74.	0.6632	8.6967	2.0176	2.1836	466	0.0301	0.4449
75.	0.5871	4.6054	6.6061	0.0694	486	0.1801	0.9280
76.	0.0464	8.9465	4.9787	3.5434	90	0.8480	0.8208
77.	0.2600	4.9823	0.7150	0.0111	108	0.8521	0.9242
78.	0.1656	5.4436	5.8912	6.2297	136	0.7392	0.7559
79.	0.4019	2.0228	1.2406	8.4365	146	0.5020	0.7905
80.	0.0702	7.9807	9.1455	2.8226	178	0.4061	0.4446
81.	0.8344	1.3793	1.1812	0.3632	296	0.0873	0.5154
82.	0.1082	8.6309	0.1750	4.6455	370	0.1387	0.0041
83.	0.7162	8.2433	8.7498	5.5319	203	0.6194	0.0668
84.	0.1883	5.9254	1.6894	3.5253	235	0.2836	0.6099
85.	0.5704	5.8193	0.2906	6.0632	495	0.5691	0.0393
86.	0.1927	4.6264	6.0123	5.3491	58	0.8584	0.8888
87.	0.6297	5.1596	8.2679	8.6203	270	0.4364	0.0900
88.	0.2141	8.9510	1.9756	7.2608	466	0.3467	0.5637
89.	0.2676	4.6516	3.4238	7.9936	314	0.5984	0.2157
90.	0.8127	0.8944	6.5179	7.1993	417	0.2937	0.5953
91.	0.2454	8.8325	2.7040	3.8295	75	0.3335	0.8328
92.	0.3471	4.2506	5.0326	3.1497	257	0.3584	0.3101
93.	0.9525	0.3708	5.9658	2.2729	247	0.0605	0.1385
94.	0.8371	7.4114	0.1069	7.6569	391	0.1126	0.6175
95.	0.1127	6.3757	4.0479	3.2106	400	0.5677	0.1239
96.	0.3317	9.7082	2.7716	3.2936	77	0.9667	0.2091
97.	0.4878	7.2104	5.9412	3.5134	346	0.8266	0.3800
98.	0.2429	7.0292	6.1121	9.7616	189	0.8010	0.3893
99.	0.9537	3.5555	0.9772	5.5150	414	0.1526	0.8130
100.	0.6036	7.5977	6.8917	0.6705	82	0.9636	0.4050

TABLE 56: Third part of the 100 randomly generated individuals used for ESN initialization. See Table 54 for description of the hyperparameters.

.5 Glossary

ABC	Artificial Bee Colony	CART	Classification and Regression Tree
Acc	Accuracy	CBAM	Convolutional Block Attention Module
AE	Autoencoder	CBPM	Combined Bivariate Performance Measure
AEX	Amsterdam Exchange Index	CCI	Commodity Channel Index
AGNN	Adversarial Game Neural Network	CEEMD	Complete Ensemble Empirical Mode Decomposition
AM	Attention Mechanism	CEEMDAN	Complete Ensemble Empirical Mode Decomposition with Adaptive Noise
AMH	Adaptive Market Hypothesis	CGAN	Conditional GAN
AR	Autoregressive	CNN	Convolutional Neural Network
ARIMA	Autoregressive Integrated Moving Average	CNN-BI	CNN with Bar Images
ARO	Artificial Rabbit Optimization	ConvLSTM	Convolutional LSTM
ARR	Average Rate of Return	CPI	Consumer Price Index
ATR	Average True Range	CSI	China Securities Index
ATX	Austrian Traded Index	DA	Direction Accuracy
AUC	Area Under the Curve	DAE	Denoising Autoencoder
BACC	Balanced Accuracy	DBN	Deep Belief Network
BB	Bollinger Bands	DE-LSTM	Differential Evolution-LSTM
BERT	Bidirectional Encoder Representations from Transformers	DJ30	Dow Jones 30
BiLSTM	Bidirectional Long Short-Term Memory	DJI	Dow Jones Industrial Average
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies	DMLP	Deep Multilayer Perceptron
BIST100	Borsa Istanbul	DM	Diebold-Mariano test
BPNN	Back Propagation Neural Network	DQN	Deep Q-Network
BSE	Bombay Stock Exchange	DRL	Deep Reinforcement Learning
BTC	Bitcoin	DS	Directional Symmetry
CAC40	A benchmark French stock market index comprising 40 major companies listed on the Euronext Paris exchange	Dstat	Directional Change Statistic
		DT	Decision Tree
		DTR	Decision Tree Regression
		DWT	Discrete Wavelet Transform

.4. Summary of the Random Parameter Sets for ESN Initialization for Chapter 4

EEMD	Ensemble Empirical Mode Decomposition	GEM Index	399006.SZ
EINS	Excitatory and Inhibitory Neuronal Synapse Unit	GJR-GARCH	Glosten–Jagannathan–Runkle GARCH
EMA	Exponential Moving Average	GJR-MEM	Glosten–Jagannathan–Runkle Multiplicative Error Models
EMD	Empirical Mode Decomposition	GM	Geometric Mean
EMH	Efficient Markets Hypothesis	GRU	Gated Recurrent Unit
EMV	Ease of Movement	GSO	Glowworm Swarm Optimization
ES	Expected Shortfall	HMM	Hidden Markov Model
ESN	Echo State Network	HONN	Higher-Order Neural Network
ES-LSTM	Exponential Smoothing-LSTM	HSI	Hong Kong Hang Seng Index
ETFs	Exchange-Traded Funds		
ETH	Ethereum	HTCSI	Halal Tourism Composite Sentiment Index, Country-specific
ETS	Exponential Smoothing State Space Model		
EVS	Explained Variance Score	ICA	Independent Component Analysis
EWMA	Exponentially Weighted Moving Average	ICEEMDAN	Improved Complete Ensemble Empirical Mode Decomposition with Adaptive Noise
EWT	Empirical Wavelet Transform		
FCHI	Ticker symbol for CAC40 Index	IDBN	Improved Deep Belief Network
FI	Force Index	IMFs	Intrinsic Mode Functions
FMHP	Fully Modified Hodrick-Prescott	IPSA	The Índice de Precio Selectivo de Acciones, Chile's primary stock market index, includes the 30 stocks with the highest average annual trading volume on the Santiago Stock Exchange
FPA	Flower Pollination Algorithm		
FPR	False Positive Rate		
FTS	Financial Time Series		
FTSE	Financial Times Stock Exchange	IRRLSTM	Improved Recurrent Rider LSTM
GA	Genetic Algorithm		
GAF	Gramian Angular Field	IXIC	Nasdaq Composite
GAN	Generative Adversarial Network	JKSE	Jakarta Stock Exchange Composite Index
GARCH	Generalized Autoregressive Conditional Heteroskedasticity	KLSE	Kuala Lumpur Stock Exchange
		KNN	K-Nearest Neighbors
GDAXI	A leading German stock market index consisting of 40 major companies trading on the Frankfurt Stock Exchange	KOSDAQ	Korea Securities Dealers Automated Quotation
		KOSPI200	Korean Composite Stock Price Index

Chapter 5. Conclusion and Future Directions

kPCA	Kernel Principal Component Analysis	NR	Nonparametric Regression
KSE	Karachi Stock Exchange	NSE	National Stock Exchange of India
LASSO	Least Absolute Shrinkage and Selection Operator	NYR	Net Yield Rate
LNN	Liquid Neural Network	NYSE	New York Stock Exchange
LR	Linear regression	OHLCV	Open, High, Low, Close, Volume
LRC	Logistic Regression Classifier	OHLCVC	Open, High, Low, Close, Volume, Change
LSTM	Long Short-Term Memory	ORELM	Outlier-Robust Extreme Learning Machine
LSTM-BN	LSTM with Batch Normalization	PatchTST	Patch Time Series Transformer
LTC	Litecoin	PCA	Principal Component Analysis
LTCs	Liquid Time-Constant Networks	PCC	Pearson's Correlation Coefficient
MA	Moving Average	PER	Price-Earnings Ratio
MACD	Moving Average Convergence Divergence	PR	Polynomial Regression
MAE	Mean Absolute Error	Prophet	Prophet
MAPE	Mean Absolute Percentage Error	PSO	Particle Swarm Optimization
MCS	Model Confidence Set	PSR	Phase Space Reconstruction
MEMD	Multivariate Empirical Mode Decomposition	PSX	Pakistan Stock Exchange
MF-ADCCA	Multifractal Asymmetric Detrended Cross-Correlation Analysis	PTSR	Price To Sales Ratio
MF-DFA	Multifractal Detrended Fluctuation Analysis	R^2	Coefficient of Determination
MFNN	Multi-Filters Neural Network	R-GARCH	Realized Generalized Autoregressive Conditional Heteroskedasticity
MICEX	Moscow Stock Exchange	RBF	Radial Basis Function
MLR	Multiple Linear Regression	RC	Reservoir Computing
MSE	Mean Squared Error	RF	Random Forest
mWDN	Multilevel Wavelet Decomposition Network	RL	Reinforcement Learning
N100	NASDAQ-100	RMSE	Root Mean Square Error
N225	Nikkei 225 Index	RN	Reynolds Number
N-BEATS	Neural Basis Expansion Analysis	RNN	Recurrent Neural Network
NLP	Natural Language Process	ROC-AUC	Receiver Operating Curve-Area Under Curve
NMSE	Normalized Mean Squared Error	ROC	Price Rate of Change
		RoR	Rate Of Return

.4. Summary of the Random Parameter Sets for ESN Initialization for Chapter 4

RP s	Recurrence Plots	SVM	Support Vector Machine
RQA	Recurrence Quantification Analysis	SVR	Support Vector Regression
ROTA	Return On Tangible Assets	SZCI	Shenzhen Component Stock Exchange Index
RSI	Relative Strength Index	SZI	Shenzhen Component Index
RVI	Relative Volatility Index	SZSC	Shenzhen Component Index
S&P/ASX200	S&P/ASX 200 index (XJO)	SZSE	SZSE Component Index
SARIMA	Seasonal Autoregressive Integrated Moving Average	Tadawul	Saudi Arabian stock market
SCI	Shanghai Composite Index	TAIEX	Taiwan Stock Exchange Capitalization Weighted Stock Index
SD	Standard Deviation	TCN	Temporal Convolution Network
SDE	Standard Deviation of Error	Theil's U	Theil's Inequality Coefficient
Sen	Sensitivity	TimeGAN	Time-Series Generative Adversarial Network
SGD	Stochastic Gradient Descent	TPMG	Three-Point Moving Gradient
SGDRegressor	Stochastic Gradient Descent Regressor	TPR	True Positive Rate
SHAP	SHapley Additive exPlanations	TRIX	Triple Exponential Average
SHOA	Spotted Hyena Optimization Algorithm	USCI	United States Commodity Index Fund
SHOAGAI-TSF	Spotted Hyena Optimization Algorithm with Generative Artificial Intelligence for Time Series Forecasting	VAR	Vector Autoregressive
Sklearn	sklearn	VaR	Value At Risk
SMA	Simple Moving Average	VET	VeChain
SMI	Stochastic Momentum Index	VMD	Variational Mode Decomposition
Spe	Specificity	WHTCSI	World Halal Tourism Composite Sentiment Index
SPX	S&P 500	WT	Wavelet Transform
SR	Sharpe Ratio	XGBoost	Extreme Gradient Boosting
SSA	Singular Spectrum Analysis	XGBoost Regressor	XGBoost Regressor
SSCI	Shanghai Stock Exchange Composite Index	XJO	S&P/ASX200 Index - Australia's leading stock market index, comprising the top 200 companies listed on the Australian Securities Exchange (ASX)
SSE	Shanghai Stock Exchange	XRP	Ripple
SSEC	Shanghai Stock Exchange Composite		
SSECI	Shanghai Stock Exchange Composite Index		
SSMI	Swiss Market Index		

List of Figures

2.3.1	A depiction of the relationship between deep learning (DL), machine learning (ML), and artificial intelligence (AI)	26
2.3.2	Illustration of a simple ANN architecture, comprising the input, hidden, and output layers	27
2.3.3	Illustration of a standard RNN (left) and its unfolded version (right) [441]	28
2.3.4	Structure of an LSTM cell [30].	29
2.3.5	Workflow of BiLSTM [111]	31
2.3.6	Structure of the hidden layer node in GRU [421]	31
2.3.7	ESN Architecture: The input layer processes external signals, which are then mapped into high-dimensional states by the reservoir. The output layer maps these states to the desired output, enabling predictions.	33
2.3.8	CNN example (Source: https://www.analytixlabs.co.in/blog/convolutional-neural-network/)	34
2.3.9	TCN Architecture. The left part is the dilated causal convolution with a kernel size of 2 and dilation factors $d = [1, 2, 4]$. The right part is the residual block [302]	35
2.3.10	Standard architecture of AE.	36
2.5.1	Annual distribution of publication count (2020–2024)	71
2.6.1	Distribution of studies by financial forecasting task (2020–2024)	73
2.6.2	Top journals by article publication count for the period 2020–2024	73
2.6.3	Topic-Model heatmap	74
2.6.4	The preferred development environments. The Python category includes PyCharm Edu, Statsmodels, Tushare, Swarm-PackagePy, yfinance, investpy, seaborn, Matplotlib, Numpy, Pandas, sklearn, and PyTorch libraries. The Other category includes PostgreSQL15, Google Colab, Apache Hadoop, NVIDIA DIGITS, PyQt5, fbprophet, Math, and Gretl.	75
3.1.1	Open-Source ML Frameworks for RC System Implementation. Each bar represents the number of new libraries introduced per year.	90

3.1.2	Types of Attractors: Point Attractor (Equilibrium) — a stable state where trajectories converge to a single point; Limit Cycle (Periodic Behavior) — a closed trajectory representing regular, repeating oscillations; Limit Torus (Quasiperiodic Behavior) — a phase space structure formed by the interaction of multiple limit cycles, each representing a separate dynamical system; Strange Chaotic Attractor — a fractal structure with sensitive dependence on initial conditions, exemplified by the Lorenz attractor; No Attractor (Random Behavior) — a random scatter of points in phase space, with no specific patterns [338]	95
3.2.1	Flowchart of a GA Process, illustrating key steps: Population Initialization, Fitness Evaluation, Parent Selection, Crossover, and Mutation. This iterative process continues until a predefined stopping criterion is met.	116
3.2.2	Illustration of the Three Types of Children in a GA Process: (1) Elite Children, who are the highest-performing individuals to survive in the next generation; (2) Crossover Children, produced by recombining traits (genes) from parent pairs; and (3) Mutation Children, generated by introducing random changes to traits, to enhance diversity and drive evolution.	117
3.2.3	Flowchart of the proposed Self-Adaptive Cuckoo Genetic Algorithm (SACGA). This framework builds on a conventional Genetic Algorithm with elitism, crossover, and a Self-Adaptive mutation mechanism that dynamically adjusts mutation rates based on fitness scores and reservoir quality, ensuring a balanced exploration–exploitation trade-off. To further enhance exploration, a Cuckoo-inspired offspring generated via <i>Lévy</i> flights is periodically introduced into the population.	125
3.2.4	Flowchart for Self-Adaptive Mutation Rate Assignment: The SACGA model employs dynamic mutation rates to balance exploration and exploitation based on predictive error minimization (f_i) and reservoir quality ($\bar{S}$). If f_i exceeds the average predictive error (f_{avg}), a mutation rate of 0.95 is assigned. If f_i is below f_{avg} and $\bar{S}$ is between 1 and 10, the mutation rate is 0.1; otherwise, it is set to 0.8. Lower mutation rates encourage exploitation through crossover with elites, while higher mutation rates promote exploration through mutations.	129
4.1.1	Snapshots of the MSO, Mackey-Glass, and NARMA10 time series datasets. The MSO dataset combines multiple sine waves of different frequencies, creating a complex oscillatory pattern. The Mackey-Glass series exhibits chaotic dynamics, and the NARMA10 dataset models a nonlinear autoregressive moving average process.	140
4.1.2	Convergence of RMSE Quantile 10 values across generations for the proposed SACGA, GA (mutation rates 10% and 1%), and deterministic GA on MSO (left), Mackey-Glass (center), and NARMA10 (right) time series. RMSE is plotted here as it guides the GA methods.	142

4.1.3	Boxplot of MAPE distributions for the proposed SACGA, GA (mutation rates 10% and 1%), and deterministic GA (Det. GA) on the MSO (left), Mackey-Glass (center), and NARMA10 (right) time series. The SACGA method consistently achieves lower MAPE values with reduced variation, indicating reliable predictive performance across all datasets.	144
4.1.4	ESN network outputs (averaged over 32 instances) and targets visualization (top) and corresponding absolute errors (bottom), after applying the SACGA method to the MSO (left), Mackey-Glass (center), and NARMA10 (right) time series.	144
4.2.1	Overview of the proposed SACGA-ESN framework for one-step-ahead stock price forecasting, showing the workflow from data preprocessing through optimization, training, and performance evaluation using both predictive and reservoir dynamics metrics.. . . .	149
4.2.2	Visualization of the dataset split into training, validation, and testing phases, corresponding to the SACGA optimization and ESN prediction tasks for portfolio values. The gray-shaded region denotes the overall optimization interval. Within this region, the blue segment represents the SACGA optimization training phase, and the orange segment indicates the SACGA optimization validation phase. The red segment represents the ESN prediction test period.. . . .	153
4.2.3	Visualization of the dataset split into training, validation, and testing phases, corresponding to the SACGA optimization and ESN prediction tasks for portfolio returns. The color scheme and phase descriptions are consistent with Figure 4.2.2.	154
4.2.4	Histogram and Kernel Density Estimate (KDE) of the proxy Composite Index returns.	155
4.2.5	Heatmap illustrating the correlation coefficients between candidate technical indicators and the target return. Features with positive correlations above 0.5 are selected as inputs for the ESN model. . . .	157
4.2.6	Boxplot of Largest Lyapunov Exponent (LLE) values across 100 random ESNs (each averaged over 10 runs). The red line indicates the optimized ESN, which lies near the lower end of the distribution, reflecting enhanced dynamic stability.. . . .	164
4.2.7	Distribution of Separation Ratio (SR) values for 100 random ESNs (averaged over 10 runs each). The optimized ESN (red horizontal line) lies near the lower whisker, achieving a more rational input class separation in the reservoir dynamics of the optimized ESN model. . . .	164
4.2.8	kPCA visualization of training set dynamics (left) and test set dynamics (right) for the optimized ESN reservoir.	165
4.2.9	Random ESN initializations projected into 3D kernel PCA space. Each image shows training set dynamics (left) and test set dynamics (right) for a given random reservoir. Examples demonstrate chaotic, unstructured behavior.	166

List of Figures

List of Tables

1	Comparison of major surveys on deep learning and financial forecasting: scope, models, and reported contributions (Part I).	24
2	Comparison of major surveys on deep learning and financial forecasting: scope, models, and reported contributions (Part II).	25
3	Cross-Study Overview of Stock Price Forecasting: Models, Assets, Features, and Timeframes	48
4	Cross-Study Overview of Index Forecasting: Models, Assets, Features, and Timeframes	58
5	Cross-Study Overview of Forex Forecasting: Models, Assets, Features, and Timeframes	61
6	Cross-Study Overview of Cryptocurrency Forecasting: Models, Assets, Features, and Timeframes	67
7	Summary of Deep Learning Model Performance Across Financial Forecasting Domains	72
8	An overview of Python libraries for GAs, featuring descriptions and associated URLs and references.	118
9	GA Optimization Parameters and ESN Hyperparameter Search Ranges	138
10	Prediction results of ESN with GA methods, on the test set for MSO, Mackey-Glass, and NARMA10 time series datasets.	143
11	SACGA Optimization Parameters and ESN Hyperparameter Search Ranges	150
12	Training, validation, and testing periods used for SACGA optimization and ESN prediction tasks, with corresponding date ranges and index splits	153
13	Summary statistical indicators of the proxy Composite Index returns representing the S&P 500 Index	154
14	Regression performance comparison of SACGA ESN (mean $\pm$ std over 32 runs) against Random Walk baseline.	161
15	Classification performance comparison of SACGA ESN (mean $\pm$ std over 32 runs) against Random Walk baseline.	161
16	Confusion Matrix of the SACGA Optimized ESN	161
17	Chaos-related metrics computed for the optimized-ESN.	163

List of Tables

18	Reservoir dynamics comparison between 100 random ESNs and the optimized ESN (mean $\pm$ SD over 10 runs)..	164
19	Interpretation of reservoir dynamics from kPCA projections for optimized and random ESNs.	167
20	Summary of Metrics and Design Methodology of the Optimized ESN.	169
21	Stock Price Forecasting: Standalone Models.	180
22	Stock Price Forecasting: Standalone Models (Continued)	181
23	Stock Price Forecasting: Novel Approaches	182
24	Stock Price Forecasting: Technical & Sentiment Analysis-Based Models	182
25	Stock Price Forecasting: Hybrid Models.	183
26	Stock Price Forecasting: Hybrid Models (Continued)	184
27	Stock Price Forecasting: Hybrid Models with Data Pre-processing (Dimensionality Reduction, Decomposition, Optimization)	185
28	Stock Price Forecasting: Hybrid Models with Technical and Sentiment Analysis	186
29	Stock Price Forecasting: Hybrid Models with Technical and Sentiment Analysis (Continued)	187
30	Stock Price Forecasting: Hybrid Models – Novel Approaches	188
31	Stock Price Forecasting: Hybrid Models – Novel Approaches (Continued))	189
32	Index Price Forecasting: Standalone Models.	190
33	Index Price Forecasting: Technical & Sentiment Analysis-Based Models	191
34	Index Price Forecasting: Hybrid Models.	192
35	Index Price Forecasting: Hybrid Models (Continued)	193
36	Index Price Forecasting: Hybrid Models (Continued)	194
37	Index Price Forecasting: Hybrid Models (Continued)	195
38	Index Price Forecasting: Hybrid Models with Data Pre-processing (Dimensionality Reduction, Decomposition, Optimization)	196
39	Index Price Forecasting: Hybrid Models with Data Pre-processing (Dimensionality Reduction, Decomposition, Optimization) (Continued)	197
40	Index Price Forecasting: Hybrid Models with Technical and Sentiment Analysis	198
41	Index Price Forecasting: Hybrid Models with Technical and Sentiment Analysis (Continued)	199
42	Index Price Forecasting: Hybrid Models – Novel Approaches	200
43	Index Price Forecasting: Hybrid Models – Novel Approaches (Continued)	201
44	Forex Price Forecasting	202
45	Forex Price Forecasting (Continued)	203
46	Commodity Forecasting	204
47	Commodity Forecasting (Continued)	205
48	Bond Forecasting	206
49	Cryptocurrency Price Forecasting: Standalone Models	207

50	Cryptocurrency Price Forecasting: Hybrid Models	208
51	Cryptocurrency Price Forecasting: Hybrid Models – Novel Approaches	209
52	Volatility Forecasting.	210
53	Other Forecasting.	211
54	First part of the 100 randomly generated individuals used for ESN initialization. Each individual consists of values for leaking rate (lr), spectral radius (ρ), shift scaling (ω), input scaling (γ), reservoir size (N), reservoir sparsity (s_{res}), and input sparsity (s_{in}).	215
55	Second part of the 100 randomly generated individuals used for ESN initialization. See Table 54 for description of the hyperparameters. . . .	216
56	Third part of the 100 randomly generated individuals used for ESN initialization. See Table 54 for description of the hyperparameters. . . .	217

Declaration of competing interest

The author declares no conflict of interest.

Data availability

The database compiled during the data collection process is available upon request.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Bibliography

- [1] (2024). Chaos. Retrieved January 5, 2024.
- [2] A, J. and N, S. (2024). A novel deep convolutional neural network algorithm for equity price prediction. *International Research Journal of Multidisciplinary Technovation*, pages 275–291.
- [3] Abdullah, M., Sulong, Z., and Chowdhury, M. A. F. (2024). Explainable deep learning model for stock price forecasting using textual analysis. *Expert Systems with Applications*, 249:123740.
- [4] Abdullah, W. and Salah, A. (2023). A novel hybrid deep learning model for price prediction. *International Journal of Electrical and Computer Engineering (IJECE)*, 13(3):3420–3431.
- [5] Abraham, R. and Shaw, C. D. (1982). Dynamics—the geometry of behavior. *Dynamics—the geometry of behavior*.
- [6] Abuein, Q. Q., Shatnawi, M. Q., Aljawarneh, E. Y., and Manasrah, A. (2024). Time series forecasting model for the stock market using lstm and svr. *International Journal of Advances in Soft Computing & Its Applications*, 16(1).
- [7] Ahmad, Z., Bao, S., and Chen, M. (2024). Deeponet-inspired architecture for efficient financial time series prediction. *Mathematics*, 12(24):3950.
- [8] Ahmed, S., Chakraborty, R. K., Essam, D. L., and Ding, W. (2022). Poly-linear regression with augmented long short term memory neural network: Predicting time series data. *Information Sciences*, 606:573–600.
- [9] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.
- [10] Akşehir, Z. D. and Kılıç, E. (2024). Multi level perspectives in stock price forecasting: Ice2de-mdl. *PeerJ Computer Science*, 10:e2125.
- [11] Akter, M. S., Shahriar, H., Chowdhury, R., and Mahdy, M. (2022). Forecasting the risk factor of frontier markets: A novel stacking ensemble of neural network approach. *Future Internet*, 14(9):252.

- [12] Al-Khasawneh, M. A., Raza, A., Khan, S. U. R., and Khan, Z. (2024). Stock market trend prediction using deep learning approach. *Computational Economics*, pages 1–32.
- [13] Al-Nefaie, A. H. and Aldhyani, T. H. (2022a). Bitcoin price forecasting and trading: Data analytics approaches. *Electronics*, 11(24):4088.
- [14] Al-Nefaie, A. H. and Aldhyani, T. H. (2022b). Predicting close price in emerging saudi stock exchange: time series models. *Electronics*, 11(21):3443.
- [15] Alba, E. and Dorronsoro, B. (2005). The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE transactions on evolutionary computation*, 9(2):126–142.
- [16] Aldhyani, T. H. and Alzahrani, A. (2022). Framework for predicting and modeling stock market prices based on deep learning algorithms. *Electronics*, 11(19):3149.
- [17] Ali, M., Khan, D. M., Alshanbari, H. M., and El-Bagoury, A. A.-A. H. (2023). Prediction of complex stock market data using an improved hybrid emd-lstm model. *Applied Sciences*, 13(3):1429.
- [18] Aljadani, A. (2022). Dlcp2f: a dl-based cryptocurrency price prediction framework. *Discover Artificial Intelligence*, 2(1):20.
- [19] Alligood, K. T., Sauer, T. D., Yorke, J. A., and Chillingworth, D. (1998). Chaos: an introduction to dynamical systems. *SIAM Review*, 40(3):732–732.
- [20] Ampountolas, A. (2023). Comparative analysis of machine learning, hybrid, and deep learning forecasting models: evidence from european financial markets and bitcoins. *Forecasting*, 5(2):472–486.
- [21] Angelova, M. and Pencheva, T. (2011). Tuning genetic algorithm parameters to improve convergence time. *International Journal of Chemical Engineering*, 2011(1):646917.
- [22] Anjum, N., Hussain, S., Algarni, A., Iqbal, J., Bourouis, S., and Ullah, S. S. (2023). A multi parameter forecasting for stock time series data using lstm and deep learning model. *Mathematics*, 11:590.
- [23] Ansari, Y., Yasmin, S., Naz, S., Zaffar, H., Ali, Z., Moon, J., and Rho, S. (2022). A deep reinforcement learning-based decision support system for automated stock market trading. *IEEE Access*, 10:127469–127501.
- [24] Antonelo, E., Schrauwen, B., and Stroobandt, D. (2008). Modeling multiple autonomous robot behaviors and behavior switching with a single reservoir computing network. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 1843–1848. IEEE.

- [25] Araya, H. T., Aduda, J., and Berhane, T. (2024). A hybrid garch and deep learning method for volatility prediction. *Journal of Applied Mathematics*, 2024(1):6305525.
- [26] Aryal, S., Nadarajah, D., Rupasinghe, P., Jayawardena, C., and Kasthurirathna, D. (2020). Comparative analysis of deep learning models for multi-step prediction of financial time series.
- [27] Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.
- [28] Bäck, T. and Schütz, M. (1996). Intelligent mutation rate control in canonical genetic algorithms. In *International Symposium on Methodologies for Intelligent Systems*, pages 158–167. Springer.
- [29] Bäck, T. H., Kononova, A. V., van Stein, B., Wang, H., Antonov, K. A., Kalkreuth, R. T., de Nobel, J., Vermetten, D., de Winter, R., and Ye, F. (2023). Evolutionary algorithms for parameter optimization—thirty years later. *Evolutionary Computation*, 31(2):81–122.
- [30] Baek, H. (2024). A cnn-lstm stock prediction model based on genetic algorithm optimization. *Asia-Pacific Financial Markets*, 31(2):205–220.
- [31] Bai, S., Kolter, J. Z., and Koltun, V. (2018a). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.
- [32] Bai, S., Kolter, J. Z., and Koltun, V. (2018b). Trellis networks for sequence modeling. *arXiv preprint arXiv:1810.06682*.
- [33] Bala, A., Ismail, I., and Ibrahim, R. (2018a). Cuckoo search based optimization of echo state network for time series prediction. In *2018 International Conference on Intelligent and Advanced System (ICIAS)*, pages 1–6. IEEE.
- [34] Bala, A., Ismail, I., Ibrahim, R., and Sait, S. M. (2018b). Applications of metaheuristics in reservoir computing techniques: a review. *IEEE Access*, 6:58012–58029.
- [35] Bala, A., Ismail, I., Ibrahim, R., Sait, S. M., and Salami, H. O. (2019). Prediction using cuckoo search optimized echo state network. *Arabian Journal for Science and Engineering*, 44:9769–9778.
- [36] Baldini, P. (2022). Reservoir computing in robotics: a review. *arXiv preprint arXiv:2206.11222*.
- [37] Ballarin, G., Dellaportas, P., Grigoryeva, L., Hirt, M., van Huellen, S., and Ortega, J.-P. (2022). Reservoir computing for macroeconomic forecasting with mixed frequency data. *arXiv preprint arXiv:2211.00363*.

- [38] Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In *Machine Learning Proceedings 1995*, pages 38–46. Elsevier.
- [39] Banks, J., Brooks, J., Cairns, G., Davis, G., and Stacey, P. (1992). On devaney's definition of chaos. *The American mathematical monthly*, 99(4):332–334.
- [40] Barra, S., Carta, S. M., Corrigan, A., Podda, A. S., and Recupero, D. R. (2020). Deep learning and time series-to-image encoding for financial forecasting. *IEEE/CAA Journal of Automatica Sinica*, 7(3):683–692.
- [41] Barthelemy, P., Bertolotti, J., and Wiersma, D. S. (2008). A lévy flight for light. *Nature*, 453(7194):495–498.
- [42] Basterrech, S., Alba, E., and Snášel, V. (2014). An experimental analysis of the echo state network initialization using the particle swarm optimization. In *2014 sixth world congress on nature and biologically inspired computing (NaBIC 2014)*, pages 214–219. IEEE.
- [43] Bathla, G., Rani, R., and Aggarwal, H. (2023). Stocks of year 2020: prediction of high variations in stock prices using lstm. *Multimedia Tools and Applications*, 82(7):9727–9743.
- [44] Behera, D., Manzoor, M., Sharma, R., Salah, M. M., Stich, I., and Mukherjee, S. K. (2023). A comprehensive first-principles investigation of SnTe perovskite for optoelectronic and thermoelectric applications. *Crystals*, 13(3):408.
- [45] Bendali, W., Saber, I., Boussetta, M., Bourachdi, B., and Mourad, Y. (2022). Optimization of deep reservoir computing with binary genetic algorithm for multi-time horizon forecasting of power consumption. *Journal Européen des Systèmes Automatisés*, 55(6):701.
- [46] Benettin, G., Galgani, L., Giorgilli, A., and Strelcyn, J.-M. (1980). Lyapunov characteristic exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. part 1: Theory. *Meccanica*, 15:9–20.
- [47] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyperparameter optimization. *Advances in neural information processing systems*, 24.
- [48] Bertschinger, N., Natschläger, T., and Legenstein, R. (2004). At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks. *Advances in neural information processing systems*, 17.
- [49] BestBrokers (2025). Forex daily trading volume: How much forex is traded? Accessed: March 6, 2025.
- [50] Bhandari, H. N., Rimal, B., Pokhrel, N. R., Rimal, R., Dahal, K. R., and Khatri, R. K. (2022). Predicting stock market index using lstm. *Machine Learning with Applications*, 9:100320.

- [51] Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., and Ghayvat, H. (2021). Cnn variants for computer vision: History, architecture, application, challenges and future scope. *Electronics*, 10(20):2470.
- [52] Bianchi, F. M., De Santis, E., Rizzi, A., and Sadeghian, A. (2015a). Short-term electric load forecasting using echo state networks and pca decomposition. *Ieee Access*, 3:1931–1943.
- [53] Bianchi, F. M., Livi, L., and Alippi, C. (2016). Investigating echo-state networks dynamics by means of recurrence analysis. *IEEE transactions on neural networks and learning systems*, 29(2):427–439.
- [54] Bianchi, F. M., Scardapane, S., Uncini, A., Rizzi, A., and Sadeghian, A. (2015b). Prediction of telephone calls load using echo state network with exogenous variables. *Neural Networks*, 71:204–213.
- [55] Billah, M. M., Sultana, A., Bhuiyan, F., and Kaosar, M. G. (2024). Stock price prediction: comparison of different moving average techniques using deep learning model. *Neural Computing and Applications*, 36(11):5861–5871.
- [56] Blasco, T., Sánchez, J. S., and García, V. (2024). A survey on uncertainty quantification in deep learning for financial time series prediction. *Neurocomputing*, 576:127339.
- [57] Boedecker, J., Obst, O., Lizier, J. T., Mayer, N. M., and Asada, M. (2012). Information processing in echo state networks at the edge of chaos. *Theory in Biosciences*, 131:205–213.
- [58] Bouslimi, J., Boubaker, S., and Tissaoui, K. (2024). Forecasting of cryptocurrency price and financial stability: Fresh insights based on big data analytics and deep learning artificial intelligence techniques. *Engineering, Technology & Applied Science Research*, 14(3):14162–14169.
- [59] Bozhkov, L., Koprinkova-Hristova, P., and Georgieva, P. (2016). Learning to decode human emotions with echo state networks. *Neural Networks*, 78:112–119.
- [60] Brenner, L. and Meyll, T. (2020). Robo-advisors: a substitute for human financial advice? *Journal of behavioral and experimental finance*, 25:100275.
- [61] Buczyński, M., Chlebus, M., Kopczewska, K., and Zajenkowski, M. (2023). Financial time series models—comprehensive review of deep learning approaches and practical recommendations. *Engineering Proceedings*, 39(1):79.
- [62] Budhiraja, R., Kumar, M., Das, M. K., Bafila, A. S., and Singh, S. (2021). A reservoir computing approach for forecasting and regenerating both dynamical and time-delay controlled financial system behavior. *Plos one*, 16(2):e0246737.
- [63] Butcher, J. B., Verstraeten, D., Schrauwen, B., Day, C. R., and Haycock, P. W. (2013). Reservoir computing and extreme learning machines for non-linear time-series data analysis. *Neural networks*, 38:76–89.

- [64] Caporale, G. M., Gil-Alana, L., and Plastun, A. (2018). Persistence in the cryptocurrency market. *Research in International Business and Finance*, 46:141–148.
- [65] Carroll, T. L. (2020). Do reservoir computers work best at the edge of chaos? *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(12).
- [66] Chandola, D., Mehta, A., Singh, S., Tikkiwal, V. A., and Agrawal, H. (2023). Forecasting directional movement of stock prices using deep learning. *Annals of Data Science*, 10(5):1361–1378.
- [67] Chauhan, A., Shivaprakash, S., Sabireen, H., Md, A. Q., and Venkataraman, N. (2023). Stock price forecasting using pso hypertuned neural nets and ensembling. *Applied Soft Computing*, 147:110835.
- [68] Chen, C. (2023). Stock price prediction based on the fusion of cnn-gru combined neural network and attention mechanism. In *2023 6th International Conference on Electronics Technology (ICET)*, pages 1166–1170. IEEE.
- [69] Chen, P., Boukouvalas, Z., and Corizzo, R. (2024). A deep fusion model for stock market prediction with news headlines and time series data. *Neural Computing and Applications*, pages 1–43.
- [70] Chen, S. and Zhou, C. (2020). Stock prediction based on genetic algorithm feature selection and long short-term memory neural network. *IEEE Access*, 9:9066–9072.
- [71] Chen, W., Hussain, W., Cauteruccio, F., and Zhang, X. (2023). Deep learning for financial time series prediction: A state-of-the-art review of standalone and hybrid models. *CMES-Computer Modeling in Engineering and Sciences*.
- [72] Chen, X., Weng, T., Yang, H., Gu, C., Zhang, J., and Small, M. (2020). Mapping topological characteristics of dynamical systems into neural networks: A reservoir computing approach. *Physical Review E*, 102(3):033314.
- [73] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- [74] Choi, J., Yoo, S., Zhou, X., and Kim, Y. (2023). Hybrid information mixing module for stock movement prediction. *IEEE Access*, 11:28781–28790.
- [75] Chrol-Cannon, J. and Jin, Y. (2014). On the correlation between reservoir metrics and performance for time series classification under the influence of synaptic plasticity. *PloS one*, 9(7):e101792.
- [76] Coletti, M. A., Scott, E. O., and Bassett, J. K. (2020). Library for evolutionary algorithms in python (leap). In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 1571–1579.

- [77] Crutchfield, J., Farmer, J., Packard, N., and Shaw, R. (1986). Chaos scientific american 225 (6): 46-57. *Crutchfeld646225Scientific American*.
- [78] Cucchi, M., Abreu, S., Ciccone, G., Brunner, D., and Kleemann, H. (2022). Hands-on reservoir computing: a tutorial for practical implementation. *Neuromorphic Computing and Engineering*, 2(3):032002.
- [79] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.
- [80] Dale, M., Miller, J. F., and Stepney, S. (2017). Reservoir computing as a model for in-materio computing. *Advances in Unconventional Computing: Volume 1: Theory*, pages 533–571.
- [81] Dancer, D. and Tremayne, A. (2005). R-squared and prediction in regression with ordered quantitative response. *Journal of Applied Statistics*, 32(5):483–493.
- [82] Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 61–69.
- [83] De Jong, K. (1980). Adaptive system design: a genetic approach. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(9):566–574.
- [84] De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. University of Michigan.
- [85] De Rainville, F.-M., Fortin, F.-A., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). Deap: A python framework for evolutionary algorithms. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 85–92.
- [86] De Rainville, F.-M., Fortin, F.-A., Gardner, M.-A., Parizeau, M., and Gagné, C. (2014). Deap: enabling nimbler evolutions. *ACM SIGEVOlution*, 6(2):17–26.
- [87] Deihimi, A. and Showkati, H. (2012). Application of echo state networks in short-term electric load forecasting. *Energy*, 39(1):327–340.
- [88] Deng, C., Huang, Y., Hasan, N., and Bao, Y. (2022). Multi-step-ahead stock price index forecasting using long short-term memory model with multivariate empirical mode decomposition. *Information Sciences*, 607:297–321.
- [89] Dhokane, R. M. and Agarwal, S. (2024). A predictive model of the stock market using the lstm algorithm with a combination of exponential moving average (ema) and relative strength index (rsi) indicators. *Journal of The Institution of Engineers (India): Series B*, pages 1–13.
- [90] Di-Giorgi, G., Salas, R., Avaria, R., Ubal, C., Rosas, H., and Torres, R. (2023). Volatility forecasting using deep recurrent neural networks as garch models. *Computational Statistics*, pages 1–27.

- [91] Ding, X., Ye, L., Jin, J., and Wang, Y. (2023). Financial profitability evaluation and forecasting using the deep learning: A case of the chinese petroleum industry. *Montenegrin Journal of Economics*, 19(4):55–64.
- [92] Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., and Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137:106040.
- [93] Donghwan, S., Busogi, M., CHUNG BAEK, A. M., and Kim, N. (2020). Forecasting stock market index based on pattern driven long short-term memory. *Economic Computation & Economic Cybernetics Studies & Research*, 54(3).
- [94] DURAIRAJ, D. M. and MOHAN, B. (2021). Statistical evaluation and prediction of financial time series using hybrid regression prediction models. *International Journal of Intelligent Systems and Applications in Engineering*, 9(4):245–255.
- [95] Durairaj, D. M. and Mohan, B. K. (2022). A convolutional neural network based approach to financial time series prediction. *Neural Computing and Applications*, 34(16):13319–13337.
- [96] Durairaj, M. and Mohan, B. (2019). A review of two decades of deep learning hybrids for financial time series prediction. *International Journal on Emerging Technologies*, 10(3):324–331.
- [97] Durairaj, M., Suneetha, C., and Mohan, B. K. (2023). Original research article financial time series prediction using deep computing approaches. *Journal of Autonomous Intelligence*, 6(1).
- [98] Dutoit, X., Schrauwen, B., Van Campenhout, J., Stroobandt, D., Van Brussel, H., and Nuttin, M. (2009). Pruning and regularization in reservoir computing. *Neurocomputing*, 72(7-9):1534–1546.
- [99] Dutta, A., Kumar, S., and Basu, M. (2020). A gated recurrent unit approach to bitcoin price prediction. *Journal of risk and financial management*, 13(2):23.
- [100] Eckmann, J.-P., Kamphorst, S. O., Ruelle, D., and Ciliberto, S. (1986). Liapunov exponents from time series. *Physical Review A*, 34(6):4971.
- [101] Eckmann, J.-P. and Ruelle, D. (1985). Ergodic theory of chaos and strange attractors. *Reviews of modern physics*, 57(3):617.
- [102] Eiben, Á. E., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on evolutionary computation*, 3(2):124–141.
- [103] El Mahjouby, M., Bennani, M. T., Lamrini, M., El Far, M., Bossoufi, B., and Alghamdi, T. A. (2024). Predicting market performance using machine and deep learning techniques. *IEEE Access*.

- [104] Eslamieh, P., Shajari, M., and Nickabadi, A. (2023). User2vec: A novel representation for the information of the social networks for stock market prediction using convolutional and recurrent neural networks. *Mathematics*, 11(13):2950.
- [105] Fama, E. F. (1970). Efficient capital markets. *Journal of finance*, 25(2):383–417.
- [106] Fan, C. and Zhang, X. (2024). Stock price nowcasting and forecasting with deep learning. *Journal of Intelligent Information Systems*, pages 1–18.
- [107] Fan, J. and Peng, S. (2022). Financial stock investment management using deep learning algorithm in the internet of things. *Computational Intelligence and Neuroscience*, 2022(1):4514300.
- [108] Fang, Z., Ma, X., Pan, H., Yang, G., and Arce, G. R. (2023). Movement forecasting of financial time series based on adaptive lstm-bn network. *Expert Systems with Applications*, 213:119207.
- [109] Farkaš, I., Bosák, R., and Gergel', P. (2016). Computational analysis of memory capacity in echo state networks. *Neural Networks*, 83:109–120.
- [110] Fathali, Z., Kodia, Z., and Ben Said, L. (2022). Stock market prediction of nifty 50 index applying machine learning techniques. *Applied Artificial Intelligence*, 36(1):2111134.
- [111] Ferdiansyah, F., Othman, S. H., Radzi, R. Z. M., Stiawan, D., and Sutikno, T. (2023). Hybrid gated recurrent unit bidirectional-long short-term memory model to improve cryptocurrency prediction accuracy. *IAES International Journal of Artificial Intelligence*, 12(1):251.
- [112] Fernando, C. and Sojakka, S. (2003). Pattern recognition in a bucket. In *European conference on artificial life*, pages 588–597. Springer.
- [113] Ferreira, A. A. and Ludermir, T. B. (2009). Genetic algorithm for reservoir computing optimization. In *2009 International Joint Conference on Neural Networks*, pages 811–815. IEEE.
- [114] Ferreira, A. A. and Ludermir, T. B. (2011). Comparing evolutionary methods for reservoir computing pre-training. In *The 2011 International Joint Conference on Neural Networks*, pages 283–290. IEEE.
- [115] Ferreira, A. A., Ludermir, T. B., and De Aquino, R. R. (2013). An approach to reservoir computing design and training. *Expert systems with applications*, 40(10):4172–4182.
- [116] Fischer, T., Sterling, M., and Lessmann, S. (2024). Fx-spot predictions with state-of-the-art transformer and time embeddings. *Expert Systems with Applications*, 249:123538.

- [117] Fleischer, J. P., von Laszewski, G., Theran, C., and Parra Bautista, Y. J. (2022). Time series analysis of cryptocurrency prices using long short-term memory. *Algorithms*, 15(7):230.
- [118] Fogarty, T. C. (1989). Varying the probability of mutation in the genetic algorithm. In *Proceedings of the 3rd international conference on genetic algorithms*, pages 104–109.
- [119] Fogel, D. B. (1997). Evolutionary algorithms in theory and practice.
- [120] Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A. G., Parizeau, M., and Gagné, C. (2012). Deap: Evolutionary algorithms made easy. *The Journal of Machine Learning Research*, 13(1):2171–2175.
- [121] Furizal, F., Ritonga, A., Ma’arif, A., and Suwarno, I. (2024). Stock price forecasting with multivariate time series long short-term memory: A deep learning approach. *Journal of Robotics and Control (JRC)*, 5(5):1322–1335.
- [122] Gad, A. F. (2021). Pygad: An intuitive genetic algorithm python library. *arXiv preprint arXiv:2106.06158*.
- [123] Gajjar, P., Saxena, A., Acharya, K., Shah, P., Bhatt, C., and Nguyen, T. T. (2024). Liquidit: stock market analysis using liquid time-constant neural networks. *International Journal of Information Technology*, 16(2):909–920.
- [124] Gallicchio, C. (2018). Chasing the echo state property. *arXiv preprint arXiv:1811.10892*.
- [125] Gallicchio, C. (2020). Sparsity in reservoir computing neural networks. In *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–7. IEEE.
- [126] Gallicchio, C. and Micheli, A. (2017). Deep echo state network (deepesn): A brief survey. *arXiv preprint arXiv:1712.04323*.
- [127] Gallicchio, C., Micheli, A., and Pedrelli, L. (2017). Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, 268:87–99.
- [128] Gallicchio, C., Micheli, A., and Pedrelli, L. (2018). Design of deep echo state networks. *Neural Networks*, 108:33–47.
- [129] Gandomi, A. H., Yang, X.-S., and Alavi, A. H. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with computers*, 29:17–35.
- [130] Gangopadhyay, S. and Majumder, P. (2023). Text representation for direction prediction of share market. *Expert Systems with Applications*, 211:118472.
- [131] Ganguly, S., Gu, Y., Xie, Y., Stan, M. R., Ghosh, A. W., and Dhar, N. K. (2018). Reservoir computing based neural image filters. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pages 3206–3210. IEEE.

- [132] Garg, S. and Ramakrishnan, G. (2020). Advances in quantum deep learning: An overview. *arXiv preprint arXiv:2005.04316*.
- [133] Gauthier, D. J., Bollt, E., Griffith, A., and Barbosa, W. A. (2021). Next generation reservoir computing. *Nature communications*, 12(1):5564.
- [134] Gehlhaar, D. K., Verkhivker, G. M., Rejto, P. A., Sherman, C. J., Fogel, D. R., Fogel, L. J., and Freer, S. T. (1995). Molecular recognition of the inhibitor ag-1343 by hiv-1 protease: conformationally flexible docking by evolutionary programming. *Chemistry & biology*, 2(5):317–324.
- [135] Georgopoulos, S. P., Tziatzios, P., Stavrinides, S. G., Antoniadis, I. P., and Haniyas, M. P. (2023). Reservoir computing vs. neural networks in financial forecasting. *International Journal of Computational Economics and Econometrics*, 13(1):1–22.
- [136] Gibbons, T. E. (2010). Unifying quality metrics for reservoir networks. In *The 2010 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE.
- [137] Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE.
- [138] Gollapalli, M., Rahman, A., Hakami, O., Alhashim, M., Arab, B., Almashharwai, F., Youldash, M., Saadeldeen, A., Alturkey, A., and Alkhulaifi, D. (2023). Predictive modeling of near cryptocurrency pricing using deep learning: Influence of bitcoin market movements. *Mathematical Modelling of Engineering Problems*, 10(6):2255–2264.
- [139] Goodfellow, I. (2016). Deep learning.
- [140] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.
- [141] Goodman, E. and Ventura, D. (2006). Spatiotemporal pattern recognition via liquid state machines. In *The 2006 IEEE international joint conference on neural network proceedings*, pages 3848–3853. IEEE.
- [142] Gotshall, S. and Rylander, B. (2002). Optimal population size and the genetic algorithm. *Population*, 100(400):900.
- [143] Goudarzi, A., Marzen, S., Banda, P., Feldman, G., Teuscher, C., and Stefanovic, D. (2016). Memory and information processing in recurrent neural networks. *arXiv preprint arXiv:1604.06929*.
- [144] Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.

- [145] Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics*, 16(1):122–128.
- [146] Grigoryeva, L. and Ortega, J.-P. (2019). Differentiable reservoir computing. *J. Mach. Learn. Res.*, 20(179):1–62.
- [147] Guan, B., Zhao, C., Yuan, X., Long, J., and Li, X. (2024). Price prediction in china stock market: an integrated method based on time series clustering and image feature extraction. *The Journal of Supercomputing*, 80(7):8553–8591.
- [148] Guckenheimer, J. (1982). Noise in chaotic systems. *Nature*, 298(5872):358–361.
- [149] Gülmez, B. (2023). Stock price prediction with optimized deep lstm network with artificial rabbits optimization algorithm. *Expert Systems with Applications*, 227:120346.
- [150] Guo, F. and Qi, M. (2011). Application of elman neural network based on improved niche adaptive genetic algorithm. In *2011 2nd International Conference on Intelligent Control and Information Processing*, volume 2, pages 660–664. IEEE.
- [151] Guo, L., Sun, Y., et al. (2024). Economic forecasting analysis of high-dimensional multifractal action based on financial time series. *International Journal for Housing Science and Its Applications*, 45:11–19.
- [152] Halim, M. P., Yudistira, N., and Dewi, C. (2022). Multicommodity prices prediction using multivariate data-driven modeling: Indonesia case. *IEEE Transactions on Computational Social Systems*.
- [153] Han, S. and Xiao, L. (2022). An improved adaptive genetic algorithm. In *SHS Web of Conferences*, volume 140, page 01044. EDP Sciences.
- [154] Hao, Y. and Gao, Q. (2020). Predicting the trend of stock market index using the hybrid neural network based on multiple time scale feature learning. *Applied Sciences*, 10(11):3961.
- [155] Haruna, T. and Nakajima, K. (2019). Optimal short-term memory before the edge of chaos in driven random recurrent networks. *Physical Review E*, 100(6):062312.
- [156] Hasani, R., Lechner, M., Amini, A., Rus, D., and Grosu, R. (2021). Liquid time-constant networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7657–7666.
- [157] Haupt, R. L. (2000). Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors. In *IEEE Antennas and Propagation Society International Symposium. Transmitting Waves of Progress to the Next Millennium. 2000 Digest. Held in conjunction with: USNC/URSI National Radio Science Meeting (C)*, volume 2, pages 1034–1037. IEEE.

- [158] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [159] Hinaut, X. and Trouvain, N. (2021). Which hype for my new task? hints and random search for echo state networks hyperparameters. In *International Conference on Artificial Neural Networks*, pages 83–97. Springer.
- [160] Hochreiter, S. (1997). Long short-term memory. *Neural Computation MIT-Press*.
- [161] Hoehn, T. P. and Pettey, C. C. (1999). Parental and cyclic-rate mutation in genetic algorithms: an initial investigation. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 297–304.
- [162] Holland, J. H. (1992). Genetic algorithms. *Scientific american*, 267(1):66–73.
- [163] Holzmann, G. (2009). Reservoir computing: a powerful black-box framework for nonlinear audio processing. In *International Conference on Digital Audio Effects (DAFx)*.
- [164] Holzmann, G. and Hauser, H. (2010). Echo state networks with filter neurons and a delay&sum readout. *Neural Networks*, 23(2):244–256.
- [165] Hong, T.-P., Wang, H.-S., and Chen, W.-C. (2000). Simultaneously applying multiple mutation operators in genetic algorithms. *Journal of heuristics*, 6:439–455.
- [166] Hong, T.-P., Wang, H.-S., Lin, W.-Y., and Lee, W.-Y. (2002). Evolution of appropriate crossover and mutation operators in a genetic process. *Applied intelligence*, 16:7–17.
- [167] Hu, Z., Zhao, Y., and Khushi, M. (2021). A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, 4(1):9.
- [168] Huang, A., Chen, H., Hu, X., and Dai, L. (2023). The analysis of enterprise improvement in global commodity price prediction based on deep learning. *Journal of Global Information Management (JGIM)*, 31(3):1–20.
- [169] Huang, Y.-P. and Shi, K.-Q. (1996). Genetic algorithms in the identification of fuzzy compensation system. In *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No. 96CH35929)*, volume 2, pages 1090–1095. IEEE.
- [170] Iacob, S., Freiburger, M., and Dambre, J. (2022). Distance-based delays in echo state networks. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 211–222. Springer.
- [171] Ian, G., Yoshua, B., and Aaron, C. (2023). Deep learning, 2016. *Book in preparation for MIT Press*. URL: <http://www.deep.learning.book.org>.

- [172] Ilyas, Q. M., Iqbal, K., Ijaz, S., Mehmood, A., and Bhatia, S. (2022). A hybrid model to predict stock closing price using novel features and a fully modified hodrick–prescott filter. *Electronics*, 11(21):3588.
- [173] Ismailova, A., Beldeubayeva, Z., Kadirkulov, K., Doumcharieva, Z., Konyrkhanova, A., Ussipbekova, D., Aripbayeva, A., and Yesmukhanova, D. (2024). Forecasting stock market prices using deep learning methods. *International Journal of Electrical & Computer Engineering (2088-8708)*, 14(5).
- [174] Jaeger, H. (2001). The “echo state” approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13.
- [175] Jaeger, H. (2002a). Adaptive nonlinear system identification with echo state networks. *Advances in neural information processing systems*, 15.
- [176] Jaeger, H. (2002b). Tutorial on training recurrent neural networks, covering bppt, rtl, ekf and the echo state network approach.
- [177] Jaeger, H. (2007). Echo state network. *scholarpedia*, 2(9):2330.
- [178] Jaeger, H. and Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80.
- [179] Jaeger, W. (2004). The echo state approach to recurrent neural networks.
- [180] Jalalvand, A., De Neve, W., Van de Walle, R., and Martens, J.-P. (2016). Towards using reservoir computing networks for noise-robust image recognition. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1666–1672. IEEE.
- [181] Jalalvand, A., Van Wallendael, G., and Van de Walle, R. (2015). Real-time reservoir computing network-based systems for detection tasks on visual contents. In *2015 7th International Conference on Computational Intelligence, Communication Systems and Networks*, pages 146–151. IEEE.
- [182] Jarrah, M. (2024). Long short-term memory and discrete wavelet transform based univariate stock market prediction model. *Journal of Information and Organizational Sciences*, 48(2):263–277.
- [183] Javadi, A. A., Farmani, R., and Tan, T. P. (2005). A hybrid intelligent genetic algorithm. *Advanced Engineering Informatics*, 19(4):255–262.
- [184] Jayanth, T. and Manimaran, A. (2024). Developing a novel hybrid model double exponential smoothing and dual attention encoder-decoder based bi-directional gated recurrent unit enhanced with bayesian optimization to forecast stock price. *IEEE Access*.

- [185] Ji, Y., Liew, A. W.-C., and Yang, L. (2021). A novel improved particle swarm optimization with long-short term memory hybrid model for stock indices forecast. *Ieee Access*, 9:23660–23671.
- [186] Jia, Y., Anaissi, A., and Suleiman, B. (2024). Resnls: An improved model for stock price forecasting. *Computational Intelligence*, 40(1):e12608.
- [187] Jin, C. and Li, Y. (2023). Cryptocurrency price prediction using frequency decomposition and deep learning. *Fractal and Fractional*, 7(10):708.
- [188] Jin, J. and Zhang, Y. (2024). Innovation in financial enterprise risk prediction model: A hybrid deep learning technique based on cnn-transformer-wt. *Journal of Organizational and End User Computing (JOEUC)*, 36(1):1–26.
- [189] Jin, Z., Yang, Y., and Liu, Y. (2020). Stock closing price prediction based on sentiment analysis and lstm. *Neural Computing and Applications*, 32:9713–9729.
- [190] JING, Y. Z. and PATHMANATHAN, D. (2023). Financial market predictions with deep learning. *Journal of Quality Measurement and Analysis IQMA*, 19(2):81–97.
- [191] Joshi, P. and Maass, W. (2004). Movement generation and control with generic neural microcircuits. In *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pages 258–273. Springer.
- [192] Joy, H., Mattheakis, M., and Protopapas, P. (2022). Rctorch: a pytorch reservoir computing package with automated hyper-parameter optimization. *arXiv preprint arXiv:2207.05870*.
- [193] Jung, G. and Choi, S.-Y. (2021). Forecasting foreign exchange volatility using deep learning autoencoder-lstm techniques. *Complexity*, 2021(1):6647534.
- [194] Jung, H. S., Kim, J. H., and Lee, H. (2024). Decoding bitcoin: leveraging macro- and micro-factors in time series analysis for price prediction. *PeerJ Computer Science*, 10:e2314.
- [195] Kaiser, Ł. and Sutskever, I. (2015). Neural gpu learn algorithms. *arXiv preprint arXiv:1511.08228*.
- [196] Kallurkar, H. S. and Chandavarkar, B. (2024). A hybrid cnn-lstm model for transaction fee forecasting in post eip-1559 ethereum. *SN Computer Science*, 5(5):1–21.
- [197] Kalman, B. L. and Kwasny, S. C. (1992). Why tanh: choosing a sigmoidal function. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 578–581. IEEE.
- [198] Kalra, R., Singh, T., Mishra, S., Kumar, N., Kim, T., Kumar, M., et al. (2024). An efficient hybrid approach for forecasting real-time stock market indices. *Journal of King Saud University-Computer and Information Sciences*, 36(8):102180.

- [199] Kang, C. Y., Lee, C. P., and Lim, K. M. (2022). Cryptocurrency price prediction with convolutional neural network and stacked gated recurrent unit. *Data*, 7(11):149.
- [200] Kanwal, A., Lau, M. F., Ng, S. P., Sim, K. Y., and Chandrasekaran, S. (2022). Bicudnnlstm-1dcnn—a hybrid deep learning-based predictive model for stock price prediction. *Expert Systems with Applications*, 202:117123.
- [201] Katha, K. K. and Pabboju, S. (2019). Agcs technique to improve the performance of neural networks. *Journal of Intelligent Systems*, 29(1):1235–1245.
- [202] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. iee.
- [203] Khalid, R. and Javaid, N. (2020). A survey on hyperparameters optimization algorithms of forecasting models in smart grid. *Sustainable Cities and Society*, 61:102275.
- [204] Kim, T. and King, B. R. (2020). Time series prediction using deep echo state networks. *Neural Computing and Applications*, 32(23):17769–17787.
- [205] Kiran Kumar, M., Chandra Patni, J., Raparathi, M., Sherkuziyeva, N., Abdullah Bilal, M., and Aurangzeb, K. (2024). Approach advancing stock market forecasting with joint rmse loss lstm-cnn model. *Fluctuation and Noise Letters*, 23(2):2440018–28790.
- [206] Kirisci, M. and Cagcag Yolcu, O. (2022). A new cnn-based model for financial time series: Taix and ftse stocks forecasting. *Neural Processing Letters*, 54(4):3357–3374.
- [207] Ko, C.-R. and Chang, H.-T. (2021). Lstm-based sentiment analysis for stock price forecast. *PeerJ Computer Science*, 7:e408.
- [208] Koryakin, D., Lohmann, J., and Butz, M. V. (2012). Balanced echo state networks. *Neural Networks*, 36:35–45.
- [209] Kraken (2024). The history of bitcoin halvings: Timeline and 2024 insights. Accessed: 2025-03-05.
- [210] Kumar, G., Singh, U. P., and Jain, S. (2022a). An adaptive particle swarm optimization-based hybrid long short-term memory model for stock price time series forecasting. *Soft Computing*, 26(22):12115–12135.
- [211] Kumar, K. and Haider, M. T. U. (2021). Enhanced prediction of intra-day stock market using metaheuristic optimization on rnn-lstm network. *New Generation Computing*, 39(1):231–272.
- [212] Kumar, R., Kumar, P., and Kumar, Y. (2022b). Integrating big data driven sentiments polarity and abc-optimized lstm for time series forecasting. *Multimedia Tools and Applications*, 81(24):34595–34614.

- [213] Lachaab, M. and Omri, A. (2023). Machine and deep learning-based stock price prediction during the covid-19 pandemic: The case of cac 40 index. *EuroMed Journal of Business*, (ahead-of-print).
- [214] Langton, C. G. (1990). Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: nonlinear phenomena*, 42(1-3):12–37.
- [215] Lawi, A., Mesra, H., and Amir, S. (2022). Implementation of long short-term memory and gated recurrent units on grouped time-series data to predict stock prices accurately. *Journal of Big Data*, 9(1):89.
- [216] Lea, C., Vidal, R., Reiter, A., and Hager, G. D. (2016). Temporal convolutional networks: A unified approach to action segmentation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8–10 and 15–16, 2016, Proceedings, Part III 14*, pages 47–54. Springer.
- [217] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- [218] Lee, J., Koh, H., and Choe, H. J. (2021). Learning to trade in financial time series using high-frequency through wavelet transformation and deep reinforcement learning. *Applied Intelligence*, 51(8):6202–6223.
- [219] Legenstein, R. and Maass, W. (2007a). Edge of chaos and prediction of computational performance for neural circuit models. *Neural networks*, 20(3):323–334.
- [220] Legenstein, R. and Maass, W. (2007b). What makes a dynamical system computationally powerful. *New directions in statistical signal processing: From systems to brain*, pages 127–154.
- [221] Li, A. W. and Bastos, G. S. (2020). Stock market forecasting using deep learning and technical analysis: a systematic review. *IEEE access*, 8:185232–185242.
- [222] Li, J. (2021). Research on market stock index prediction based on network security and deep learning. *Security and Communication Networks*, 2021.
- [223] Li, J., Chen, W., Liu, Y., Yang, J., Zeng, D., and Zhou, Z. (2024a). Neural ordinary differential equation networks for fintech applications using internet of things. *IEEE Internet of Things Journal*.
- [224] Li, L., Fang, N., Wang, L., and Huang, Z. (2017). Modeling of the multiple superimposed oscillator problem using linear echo state network with leaky-integrator neurons. In *Proceedings of the 6th International Conference on Information Engineering*, pages 1–5.
- [225] Li, M., Zhu, Y., Shen, Y., and Angelova, M. (2023a). Clustering-enhanced stock price prediction using deep learning. *World Wide Web*, 26(1):207–232.

- [226] Li, S., Huang, X., Cheng, Z., Zou, W., and Yi, Y. (2023b). Ae-acg: A novel deep learning-based method for stock price movement prediction. *Finance Research Letters*, 58:104304.
- [227] Li, S., Tang, G., Chen, X., and Lin, T. (2024b). Stock index forecasting using a novel integrated model based on ceemdan and tcn-gru-cbam. *IEEE Access*.
- [228] Li, S., Yan, C., and Liu, Y. (2023c). Energy efficiency and coding of neural network. *Frontiers in Neuroscience*, 16:1089373.
- [229] Li, X., Liu, Q., and Wu, Y. (2022). Prediction on blockchain virtual currency transaction under long short-term memory model and deep belief network. *Applied Soft Computing*, 116:108349.
- [230] Li, Y., Chen, L., Sun, C., Liu, G., Chen, C., and Zhang, Y. (2024c). Accurate stock price forecasting based on deep learning and hierarchical frequency decomposition. *IEEE Access*.
- [231] Li, Z., Han, J., and Song, Y. (2020). On the forecasting of high-frequency financial time series based on arima model improved by deep learning. *Journal of Forecasting*, 39(7):1081–1097.
- [232] Lin, C. (2009). An adaptive genetic algorithm based on population diversity strategy. In *2009 Third International Conference on Genetic and Evolutionary Computing*, pages 93–96. IEEE.
- [233] Linardatos, P., Papastefanopoulos, V., and Kotsiantis, S. (2020). Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18.
- [234] Liu, H. and Long, Z. (2020). An improved deep learning model for predicting stock market price time series. *Digital Signal Processing*, 102:102741.
- [235] Liu, L., Pei, Z., Chen, P., Luo, H., Gao, Z., Feng, K., and Gan, Z. (2023). An efficient gan-based multi-classification approach for financial time series volatility trend prediction. *International Journal of Computational Intelligence Systems*, 16(1):40.
- [236] Liu, W., Ge, Y., and Gu, Y. (2024a). Multi-factor stock price prediction based on gan-trellisnet. *Knowledge and Information Systems*, pages 1–22.
- [237] Liu, W.-J., Ge, Y.-B., and Gu, Y.-C. (2024b). News-driven stock market index prediction based on trellis network and sentiment attention mechanism. *Expert Systems with Applications*, 250:123966.
- [238] Livieris, I. E., Kiriakidou, N., Stavroyiannis, S., and Pintelas, P. (2021). An advanced cnn-lstm model for cryptocurrency forecasting. *Electronics*, 10(3):287.
- [239] Livieris, I. E., Pintelas, E., Stavroyiannis, S., and Pintelas, P. (2020). Ensemble deep learning models for forecasting cryptocurrency time-series. *Algorithms*, 13(5):121.

- [240] Lizhe, Y., Bo, X., and Xianjie, W. (2010). Bp network model optimized by adaptive genetic algorithms and the application on quality evaluation for class teaching. In *2010 2nd International Conference on Future Computer and Communication*, volume 3, pages V3–273. IEEE.
- [241] Løkse, S., Bianchi, F. M., and Jenssen, R. (2017). Training echo state networks with regularization through dimensionality reduction. *Cognitive Computation*, 9:364–378.
- [242] Long, W., Lu, Z., and Cui, L. (2019). Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems*, 164:163–173.
- [243] Lukoševičius, M. (2012). A practical guide to applying echo state networks. In *Neural Networks: Tricks of the Trade: Second Edition*, pages 659–686. Springer.
- [244] Lukoševičius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer science review*, 3(3):127–149.
- [245] Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.
- [246] Luo, A., Zhong, L., Wang, J., Wang, Y., Li, S., and Tai, W. (2024). Short-term stock correlation forecasting based on cnn-bilstm enhanced by attention mechanism. *IEEE Access*.
- [247] Ma, C., Wu, J., Hu, H., Chen, Y. N., and Li, J. Y. (2024). Predicting stock prices using hybrid lstm and arima model. *IAENG International Journal of Applied Mathematics*, 54(3).
- [248] Ma, Q., Shen, L., and Cottrell, G. W. (2017). Deep-esn: A multiple projection-encoding hierarchical reservoir computing framework. *arXiv preprint arXiv:1711.05255*.
- [249] Ma, Y., Han, R., and Wang, W. (2021). Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications*, 165:113973.
- [250] Ma'arif, A., Firdaus, A. A., Suwarno, I., et al. (2024). Capability of hybrid long short-term memory in stock price prediction: A comprehensive literature review. *International Journal of Robotics & Control Systems*, 4(3).
- [251] Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA.
- [252] Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560.
- [253] Mackey, M. C. and Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289.

- [254] Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2020). The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74.
- [255] Malkiel, B. G. (2003). The efficient market hypothesis and its critics. *Journal of economic perspectives*, 17(1):59–82.
- [256] Manjunath, C., Marimuthu, B., and Ghosh, B. (2021). Deep learning for stock market index price movement forecasting using improved technical analysis. *International Journal of Intelligent Engineering & Systems*, 14(5).
- [257] Manjunath, G. and Jaeger, H. (2013). Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. *Neural computation*, 25(3):671–696.
- [258] Mansoor, Z. K., Al-Sultan, A. Y., et al. (2025). Enhancing stock price prediction using mutual information, pca, and lstm: A deep learning approach. *Fusion: Practice and Applications*, (1):196–96.
- [259] Mao, W., Liu, P., and Huang, J. (2024). Sf-transformer: A mutual information-enhanced transformer model with spot-forward parity for forecasting long-term chinese stock index futures prices. *Entropy*, 26(6):478.
- [260] Maratkhan, A., Ilyassov, I., Aitzhanov, M., Demirci, M. F., and Ozbayoglu, A. M. (2021). Deep learning-based investment strategy: technical indicator clustering and residual blocks. *Soft Computing*, 25(7):5151–5161.
- [261] Marcinkevičs, R. and Vogt, J. E. (2020). Interpretability and explainability: A machine learning zoo mini-tour. *arXiv preprint arXiv:2012.01805*.
- [262] Martinuzzi, F., Rackauckas, C., Abdelrehim, A., Mahecha, M. D., and Mora, K. (2022). Reservoircomputing. jl: an efficient and modular library for reservoir computing models. *The Journal of Machine Learning Research*, 23(1):13093–13100.
- [263] McCaffrey, D. F., Ellner, S., Gallant, A. R., and Nychka, D. W. (1992). Estimating the lyapunov exponent of a chaotic system with nonparametric regression. *Journal of the American Statistical Association*, pages 682–695.
- [264] Md, A. Q., Kapoor, S., AV, C. J., Sivaraman, A. K., Tee, K. F., Sabireen, H., and Janakiraman, N. (2023). Novel optimization approach for stock price forecasting using multi-layered sequential lstm. *Applied Soft Computing*, 134:109830.
- [265] Mienye, E., Jere, N., Obaido, G., Mienye, I. D., and Aruleba, K. (2024). Deep learning in finance: A survey of applications and techniques. *AI*, 5(4):2066.
- [266] Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, 69:46–61.

- [267] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.
- [268] Moghaddam, A. H. and Momtazi, S. (2021). Image processing meets time series analysis: Predicting forex profitable technical pattern positions. *Applied Soft Computing*, 108:107460.
- [269] Mohamed Rida, S., Hamza, E., and Taher, Z. (2024). From technical indicators to trading decisions: A deep learning model combining cnn and lstm. *International Journal of Advanced Computer Science & Applications*, 15(6).
- [270] Moon, J., Wu, Y., and Lu, W. D. (2021). Hierarchical architectures in reservoir computing systems. *Neuromorphic Computing and Engineering*, 1(1):014006.
- [271] Moradi, M., Neshat, N., and Ahmadzade Semeskande, A. M. (2023). Improving financial investment by deep learning method: predicting stock returns of tehran stock exchange companies. *Journal of Mathematics and Modeling in Finance*, 3(1):145–164.
- [272] Naidoo, V. and Du, S. (2022). A deep learning method for the detection and compensation of outlier events in stock data. *Electronics*, 11(21):3465.
- [273] Nair, M., Marie, M. I., and Abd-Elmegid, L. A. (2023). Prediction of cryptocurrency price using time series data and deep learning algorithms. *International Journal of Advanced Computer Science and Applications*, 14(8):338–347.
- [274] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.
- [275] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Satoshi Nakamoto*.
- [276] Natschläger, T., Bertschinger, N., and Legenstein, R. (2005). At the edge of chaos: Real-time computations and self-organized criticality in recurrent neural networks. *Advances in neural information processing systems*, 17:145–152.
- [277] Naufal, W. (2023). Time series forecasting based on deep learning cnn-lstm-gru model on stock prices. *International Journal of Engineering Trends and Technology*, 71(6):126–133.
- [278] Nazareth, N. and Reddy, Y. (2024). Predictive analysis of s&p bse greenex index: Unlocking insights for sustainable investments.
- [279] Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2022). A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*.

- [280] Noh, Y., Kim, J.-M., Hong, S., and Kim, S. (2023). Deep learning model for multivariate high-frequency time-series data: Financial market index prediction. *Mathematics*, 11(16):3603.
- [281] Norton, D. and Ventura, D. (2010). Improving liquid state machines through iterative refinement of the reservoir. *Neurocomputing*, 73(16-18):2893–2904.
- [282] Olorunnimbe, K. and Viktor, H. (2023). Deep learning in the stock market—a systematic survey of practice, backtesting, and applications. *Artificial Intelligence Review*, 56(3):2057–2109.
- [283] Olson, R. S., Bartley, N., Urbanowicz, R. J., and Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the genetic and evolutionary computation conference 2016*, pages 485–492.
- [284] Oreshkin, B. N., Carпов, D., Chapados, N., and Bengio, Y. (2019). N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*.
- [285] Ott, E. (2002). *Chaos in dynamical systems*. Cambridge university press.
- [286] Ozbayoglu, A. M., Gudelek, M. U., and Sezer, O. B. (2020). Deep learning for financial applications: A survey. *Applied soft computing*, 93:106384.
- [287] Pai, N., Velchamy, I., and Ramesh, N. (2023). Forecasting of multistep multivariate financial data through gso algorithm infused vanilla lstm model. *Journal of Computer Science*, 19:909–924.
- [288] Pandey, H. M., Chaudhary, A., and Mehrotra, D. (2014). A comparative review of approaches to prevent premature convergence in ga. *Applied Soft Computing*, 24:1047–1077.
- [289] Park, H. J., Kim, Y., and Kim, H. Y. (2022). Stock market forecasting using a multi-task approach integrating long short-term memory and the random forest framework. *Applied Soft Computing*, 114:108106.
- [290] Pascanu, R. (2013). On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.
- [291] Patel, M. M., Tanwar, S., Gupta, R., and Kumar, N. (2020). A deep learning-based cryptocurrency price prediction scheme for financial institutions. *Journal of information security and applications*, 55:102583.
- [292] Patel, R., Chauhan, J., Tiwari, N. K., Upadhyay, V., and Bajpai, A. (2024). A deep learning framework for hourly bitcoin price prediction using bi-lstm and sentiment analysis of twitter data. *SN Computer Science*, 5(6):767.
- [293] Patil, P. R., Parasar, D., and Charhate, S. (2023). An effective deep learning model with reduced error rate for accurate forecast of stock market direction. *Intelligent Decision Technologies*, 17(3):621–639.

- [294] Patil, V. and Pawar, D. (2015). The optimal crossover or mutation rates in genetic algorithm: A review. *International Journal of Applied Engineering and Technology*, 5(3):38–41.
- [295] Peng, K. and Yan, G. (2021). A survey on deep learning for financial risk prediction. *Quantitative Finance and Economics*, 5(4):716–737.
- [296] Perone, C. S. (2009). Pyevolve: a python open-source framework for genetic algorithms. *Acm Sigevolution*, 4(1):12–20.
- [297] Petrozziello, A., Troiano, L., Serra, A., Jordanov, I., Storti, G., Tagliaferri, R., and La Rocca, M. (2022). Deep learning for volatility forecasting in asset management. *Soft Computing*, 26(17):8553–8574.
- [298] Phien, N. N. and Platos, J. (2023). The psr-transformer nexus: A deep dive into stock time series forecasting. *International Journal of Advanced Computer Science & Applications*, 14(12).
- [299] Ping, W., Hu, Y., and Luo, L. (2024). Price forecast of treasury bond market yield: Optimize method based on deep learning model. *IEEE Access*.
- [300] Prachyachuwong, K. and Vateekul, P. (2021). Stock trend prediction using deep learning approach on technical indicator and industrial specific information. *Information*, 12(6):250.
- [301] Qi, C., Ren, J., and Su, J. (2023). Gru neural network based on ceemdan-wavelet for stock price prediction. *Applied Sciences*, 13(12):7104.
- [302] Qin, H. (2019). Comparison of deep learning models on time series forecasting: A case study of dissolved oxygen prediction. *arXiv preprint arXiv:1911.08414*.
- [303] Qu, Y., Quan, P., Lei, M., and Shi, Y. (2019). Review of bankruptcy prediction using machine learning and deep learning techniques. *Procedia Computer Science*, 162:895–899.
- [304] Rajabioun, R. (2011). Cuckoo optimization algorithm. *Applied soft computing*, 11(8):5508–5518.
- [305] Ramezani, S. B., Sommers, A., Manchukonda, H. K., Rahimi, S., and Amirlatifi, A. (2020). Machine learning algorithms in quantum computing: A survey. In *2020 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE.
- [306] Rasekhschaffe, K. C. and Jones, R. C. (2019). Machine learning for stock selection. *Financial Analysts Journal*, 75(3):70–88.
- [307] Rath, S., Das, N. R., and Pattanayak, B. K. (2024). Stacked bi-lstm and e-optimized cnn-a hybrid deep learning model for stock price prediction. *Optical Memory and Neural Networks*, 33(2):102–120.

- [308] Renders, J.-M. and Flasse, S. P. (1996). Hybrid methods using genetic algorithms for global optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(2):243–258.
- [309] Reynolds, A. M. and Frye, M. A. (2007). Free-flight odor tracking in drosophila is consistent with an optimal intermittent scale-free search. *PloS one*, 2(4):e354.
- [310] Rezaei, H., Faaljou, H., and Mansourfar, G. (2021). Stock price prediction using deep learning and frequency decomposition. *Expert Systems with Applications*, 169:114332.
- [311] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.
- [312] Rodan, A. and Tino, P. (2010). Minimum complexity echo state network. *IEEE transactions on neural networks*, 22(1):131–144.
- [313] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.
- [314] Rosenstein, M. T., Collins, J. J., and De Luca, C. J. (1993). A practical method for calculating largest lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, 65(1-2):117–134.
- [315] Roy Choudhury, A., Abrishami, S., Turek, M., and Kumar, P. (2020). Enhancing profit from stock transactions using neural networks. *Ai Communications*, 33(2):75–92.
- [316] Rundo, F., Trenta, F., Di Stallo, A. L., and Battiato, S. (2019). Machine learning for quantitative finance applications: A survey. *Applied Sciences*, 9(24):5574.
- [317] Saemi, M., Ahmadi, M., and Varjani, A. Y. (2007). Design of neural networks using genetic algorithm for the permeability estimation of the reservoir. *Journal of Petroleum Science and Engineering*, 59(1-2):97–105.
- [318] Sakib, M. and Mustajab, S. (2024). Enhanced multi-variate time series prediction through statistical-deep learning integration: The var-stacked lstm model. *SN Computer Science*, 5(5):573.
- [319] Salahshour, S., Salimi, M., Tehranian, K., Erfanibehrouz, N., Ferrara, M., and Ahmadian, A. (2024). Deep prediction on financial market sequence for enhancing economic policies. *Decisions in Economics and Finance*, pages 1–20.
- [320] Salama, R. (2025). Integrating spotted hyena optimization technique with generative artificial intelligence for time series forecasting. *Expert Systems*, 42(2):e13681.
- [321] Salmen, M. and Ploger, P. G. (2005). Echo state networks used for motor control. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 1953–1958. IEEE.

- [322] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- [323] Sano, M. and Sawada, Y. (1985). Measurement of the lyapunov spectrum from a chaotic time series. *Physical review letters*, 55(10):1082.
- [324] Santoro, D. and Grilli, L. (2022). Generative adversarial network to evaluate quantity of information in financial markets. *Neural Computing and Applications*, 34(20):17473–17490.
- [325] Sarıkoç, M. and Celik, M. (2024). Pca-ica-lstm: A hybrid deep learning model based on dimension reduction methods to predict s&p 500 index price. *Computational Economics*, pages 1–67.
- [326] Schaetti, N. (2018). Echotorch: Reservoir computing with pytorch. *GitHub Repos. GitHub <https://github.com/nschaetti/EchoTorch>*.
- [327] Schaetti, N. (2019). Behaviors of reservoir computing models for textual documents classification. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE.
- [328] Schaffer, J. D., Morishima, A., et al. (1987). An adaptive crossover distribution mechanism for genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 36–40. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.
- [329] Scherer, S., Oubbati, M., Schwenker, F., and Palm, G. (2008). Real-time emotion recognition from speech using echo state networks. In *Artificial Neural Networks in Pattern Recognition: Third IAPR Workshop, ANNPR 2008 Paris, France, July 2-4, 2008 Proceedings 3*, pages 205–216. Springer.
- [330] Schmidhuber, J., Wierstra, D., Gagliolo, M., and Gomez, F. (2007). Training recurrent networks by evolino. *Neural computation*, 19(3):757–779.
- [331] Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer.
- [332] Schrauwen, B., Verstraeten, D., and Van Campenhout, J. (2007). An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th european symposium on artificial neural networks. p. 471-482 2007*, pages 471–482.
- [333] Senthilnath, J., Das, V., Omkar, S., and Mani, V. (2013). Clustering using levy flight cuckoo search. In *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012) Volume 2*, pages 65–75. Springer.

- [334] Seyyedabbasi, A. and Kiani, F. (2021). I-gwo and ex-gwo: improved algorithms of the grey wolf optimizer to solve global optimization problems. *Engineering with Computers*, 37(1):509–532.
- [335] Sezer, O. B., Gudelek, M. U., and Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90:106181.
- [336] Sezer, O. B. and Ozbayoglu, A. M. (2019). Financial trading model with stock bar chart image time series with deep convolutional neural networks. *arXiv preprint arXiv:1903.04610*.
- [337] Shah, J., Vaidya, D., and Shah, M. (2022). A comprehensive review on multiple hybrid deep learning approaches for stock prediction. *Intelligent Systems with Applications*, 16:200111.
- [338] Sharma, V. (2009). Deterministic chaos and fractal complexity in the dynamics of cardiovascular behavior: perspectives on a new frontier. *The Open Cardiovascular Medicine Journal*, 3:110.
- [339] Sher, T., Rehman, A., Kim, D., and Ihsan, I. (2023). Exploiting data science for measuring the performance of technology stocks. *Computers, Materials and Continua*, 76(3):2979–2995.
- [340] Shoong, W. P., Asmai, S. A., Zulkarnain, N. Z., and Chuan, T. C. (2022). An improved lstm technique using three-point moving gradient for stock price forecasting. *International Journal of Computer Information Systems and Industrial Management Applications*, 14:9–9.
- [341] Simov, K., Koprinkova-Hristova, P., Popov, A., and Osenova, P. (2020). A reservoir computing approach to word sense disambiguation. *Cognitive Computation*, pages 1–10.
- [342] Singal, V. (2004). *Beyond the random walk: A guide to stock market anomalies and low-risk investing*. Oxford University Press, USA.
- [343] Sivadasan, E., Mohana Sundaram, N., and Santhosh, R. (2024). Stock market forecasting using deep learning with long short-term memory and gated recurrent unit. *Soft Computing*, 28(4):3267–3282.
- [344] Sivapurapu, S. A. (2020). Comparative study of time series and deep learning algorithms for stock price prediction. *International Journal of Advanced Computer Science and Applications*, 11(6).
- [345] Skowronski, M. D. and Harris, J. G. (2007). Automatic speech recognition using a predictive echo state network classifier. *Neural networks*, 20(3):414–423.
- [346] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

- [347] Soltani, R., Benmohamed, E., and Ltifi, H. (2023). Echo state network optimization: A systematic literature review. *Neural Processing Letters*, 55(8):10251–10285.
- [348] Song, H., Peng, D., and Huang, X. (2020). Incorporating research reports and market sentiment for stock excess return prediction: A case of mainland china. *Scientific Programming*, 2020(1):8894757.
- [349] Song, Y., Cai, C., Ma, D., and Li, C. (2024). Modelling and forecasting high-frequency data with jumps based on a hybrid nonparametric regression and lstm model. *Expert Systems with Applications*, 237:121527.
- [350] Song, Y., Li, Z., Ma, Z., and Sun, X. (2023). Dynamic forecasting for nonstationary high-frequency financial data with jumps based on series decomposition and reconstruction. *Journal of Forecasting*, 42(5):1055–1068.
- [351] Sonkavde, G., Dharrao, D. S., Bongale, A. M., Deokate, S. T., Doreswamy, D., and Bhat, S. K. (2023). Forecasting stock market prices using machine learning and deep learning models: A systematic review, performance analysis and discussion of implications. *International Journal of Financial Studies*, 11(3):94.
- [352] Spears, W. M. (1993). Crossover or mutation? In *Foundations of genetic algorithms*, volume 2, pages 221–237. Elsevier.
- [353] Spears, W. M. and De Jong, K. A. (1991). An analysis of multi-point crossover. In *Foundations of genetic algorithms*, volume 1, pages 301–315. Elsevier.
- [354] Srinivas, M. and Patnaik, L. M. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):656–667.
- [355] Srivastava, S., Pant, M., and Gupta, V. (2023). Analysis and prediction of indian stock market: a machine-learning approach. *International Journal of System Assurance Engineering and Management*, 14(4):1567–1585.
- [356] Staffini, A. (2022). Stock price forecasting by a deep convolutional generative adversarial network. 2022. URL: <https://www.frontiersin.org/articles/10.3389/frai>.
- [357] Stefanski, A. (2000). Estimation of the largest lyapunov exponent in systems with impacts. *Chaos, Solitons & Fractals*, 11(15):2443–2451.
- [358] Steiner, P., Jalalvand, A., Stone, S., and Birkholz, P. (2022). Pyrcn: A toolbox for exploration and application of reservoir computing networks. *Engineering Applications of Artificial Intelligence*, 113:104964.
- [359] Steiner, P., Stone, S., Birkholz, P., and Jalalvand, A. (2021). Multipitch tracking in music signals using echo state networks. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pages 126–130. IEEE.

- [360] Strauss, T., Wustlich, W., and Labahn, R. (2012). Design strategies for weight matrices of echo state networks. *Neural computation*, 24(12):3246–3276.
- [361] Sun, G., Lin, J., Yang, C., Yin, X., Li, Z., Guo, P., Sun, J., Fan, X., and Pan, B. (2021). Stock price forecasting: An echo state network approach. *Comput. Syst. Sci. Eng.*, 36(3):509–520.
- [362] Sun, N. and Lu, Y. (2019). A self-adaptive genetic algorithm with improved mutation mode based on measurement of population diversity. *Neural Computing and Applications*, 31:1435–1443.
- [363] Ta, V.-D., Liu, C.-M., and Tadesse, D. A. (2020). Portfolio optimization-based stock prediction using long-short term memory network in quantitative trading. *Applied Sciences*, 10(2):437.
- [364] Tanaka, G., Yamane, T., Héroux, J. B., Nakanehands, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., and Hirose, A. (2019). Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123.
- [365] Tang, P., Tang, C., and Wang, K. (2024). Stock movement prediction: A multi-input lstm approach. *Journal of Forecasting*.
- [366] TATANE, K., SAHIB, M. R., and ZAKI, T. (2024). From time series to images: Revolutionizing stock market predictions with convolutional deep neural networks. *International Journal of Advanced Computer Science & Applications*, 15(1).
- [367] Taylor, R. (2010). Attractors: Nonstrange to chaotic. *Society for industrial and applied mathematics, undergraduate research online*, pages 72–80.
- [368] Thakkar, A. and Chaudhari, K. (2021). A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions. *Expert Systems with Applications*, 177:114800.
- [369] Thierens, D. (2002). Adaptive mutation rate control schemes in genetic algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 1, pages 980–985. IEEE.
- [370] Tian, Z. (2021). Approach for short-term wind power prediction via kernel principal component analysis and echo state network optimized by improved particle swarm optimization algorithm. *Transactions of the Institute of Measurement and Control*, 43(16):3647–3662.
- [371] Tong, G. and Yin, Z. (2022). Adaptive trading system of assets for international cooperation in agricultural finance based on neural network. *Computational Economics*, 59(4):1557–1576.
- [372] Touzani, Y. and Douzi, K. (2021). An lstm and gru based trading strategy adapted to the moroccan market. *Journal of big Data*, 8(1):126.

- [373] Trierweiler Ribeiro, G., Guilherme Sauer, J., Fraccanabbia, N., Cocco Mariani, V., and dos Santos Coelho, L. (2020). Bayesian optimized echo state network applied to short-term load forecasting. *Energies*, 13(9):2390.
- [374] Tripathy, N., Hota, S., and Mishra, D. (2023). Performance analysis of bitcoin forecasting using deep learning techniques. *Indonesian Journal of Electrical Engineering and Computer Science*, 31(3):1515–1522.
- [375] Trouvain, N. and Hinaut, X. (2022). reservoirpy: A simple and flexible reservoir computing tool in python.
- [376] Trouvain, N., Pedrelli, L., Dinh, T. T., and Hinaut, X. (2020). Reservoirpy: an efficient and user-friendly library to design echo state networks. In *International Conference on Artificial Neural Networks*, pages 494–505. Springer.
- [377] Tsai, Y.-C., Chen, J.-H., and Wang, J.-J. (2019). Predict forex trend via convolutional neural networks. *Journal of Intelligent Systems*, 29(1):941–958.
- [378] Tuson, A. and Ross, P. (1996). Cost based operator rate adaptation: An investigation. In *International Conference on Parallel Problem Solving from Nature*, pages 461–469. Springer.
- [379] Uçkan, T. (2024). Integrating pca with deep learning models for stock market forecasting: An analysis of turkish stocks markets. *Journal of King Saud University-Computer and Information Sciences*, 36(8):102162.
- [380] Valencia, C. H., Vellasco, M. M., and Figueiredo, K. (2023). Echo state networks: novel reservoir selection and hyperparameter optimization model for time series forecasting. *Neurocomputing*, 545:126317.
- [381] Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K., et al. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12.
- [382] Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.
- [383] Velmurugan, T. and Indhumathy, T. (2020). Predicting support and resistance indicators for stock market with fibonacci sequence in long short-term memory. *Journal of Computer Science*, 16(10):1428–1438.
- [384] Venayagamoorthy, G. K. and Shishir, B. (2009). Effects of spectral radius and settling time in the performance of echo state networks. *Neural Networks*, 22(7):861–863.
- [385] Verstraeten, D., Schrauwen, B., Dieleman, S., Brakel, P., Buteneers, P., and Pecevski, D. (2012). Oger: modular learning architectures for large-scale sequential processing. *The Journal of Machine Learning Research*, 13(1):2995–2998.

- [386] Verstraeten, D., Schrauwen, B., d’Haene, M., and Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural networks*, 20(3):391–403.
- [387] Verstraeten, D., Schrauwen, B., and Stroobandt, D. (2006). Reservoir-based techniques for speech recognition. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1050–1053. IEEE.
- [388] Vidal, A. and Kristjanpoller, W. (2020). Gold volatility prediction using a cnn-lstm approach. *Expert Systems with Applications*, 157:113481.
- [389] Vuković, D. B., Radenković, S. D., Simeunović, I., Zinovev, V., and Radovanović, M. (2024). Predictive patterns and market efficiency: A deep learning approach to financial time series forecasting. *Mathematics*, 12(19):3066.
- [390] Vuong, P. H., Dat, T. T., Mai, T. K., Uyen, P. H., et al. (2022). Stock-price forecasting based on xgboost and lstm. *Computer Systems Science & Engineering*, 40(1).
- [391] Vuong, P. H., Phu, L. H., Van Nguyen, T. H., Duy, L. N., Bao, P. T., and Trinh, T. D. (2024). A bibliometric literature review of stock price forecasting: from statistical model to deep learning approach. *Science Progress*, 107(1):00368504241236557.
- [392] Wang, C.-H., Yuan, J., Zeng, Y., and Lin, S. (2024a). A deep learning integrated framework for predicting stock index price and fluctuation via singular spectrum analysis and particle swarm optimization. *Applied Intelligence*, 54(2):1770–1797.
- [393] Wang, J., Liu, D., Jin, L., Sun, Q., and Xue, Z. (2023a). A pca-igru model for stock price prediction. *Journal of Internet Technology*, 24(3):621–629.
- [394] Wang, L. and Lee, R. S. (2023). Stock index return volatility forecast via excitatory and inhibitory neuronal synapse unit with modified mf-adcca. *Fractal and Fractional*, 7(4):292.
- [395] Wang, S., Lin, Y., Jia, Y., Sun, J., and Yang, Z. (2024b). Unveiling the multi-dimensional spatio-temporal fusion transformer (mdstft): A revolutionary deep learning framework for enhanced multi-variate time series forecasting. *IEEE Access*.
- [396] Wang, Y., Andreeva, G., and Martin-Barragan, B. (2023b). Machine learning approaches to forecasting cryptocurrency volatility: Considering internal and external determinants. *International Review of Financial Analysis*, 90:102914.
- [397] Wang, Y. and Yan, G. (2021). Survey on the application of deep learning in algorithmic trading. *Data Science in Finance and Economics*, 1(4):345–361.
- [398] Wang, Y.-J., Wu, L.-H., and Wu, L.-C. (2024c). An integrative extraction approach for index-tracking portfolio construction and forecasting under a deep learning framework. *The Journal of Supercomputing*, 80(2):2047–2066.

- [399] Wei, Q. and Dai, Q. (2023). Adversarial unsupervised domain adaptation based on generative adversarial network for stock trend forecasting. *Intelligent Data Analysis*, (Preprint):1–26.
- [400] Wei, S., Wang, S., Sun, S., and Xu, Y. (2022). Stock ranking prediction based on an adversarial game neural network. *IEEE Access*, 10:65028–65036.
- [401] Wen, H., Yuan, M., Wang, S., Liang, L., and Fu, X. (2024). A multilevel wavelet decomposition network hybrid model utilizing cyclic patterns for stock price prediction. *Complexity*, 2024(1):1124822.
- [402] Whitley, D. and Starkweather, T. (1990). Genitor ii: A distributed genetic algorithm. *Journal of Experimental & Theoretical Artificial Intelligence*, 2(3):189–214.
- [403] Whitley, L. D. et al. (1989). *The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best*. Colorado State University, Department of Computer Science.
- [404] Widiputra, H. and Juwono, E. (2024). Parallel multivariate deep learning models for time-series prediction: A comparative analysis in asian stock markets. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 13:475.
- [405] Wiebe, N., Kapoor, A., and Svore, K. M. (2014). Quantum deep learning. *arXiv preprint arXiv:1412.3489*.
- [406] Wierstra, D., Gomez, F. J., and Schmidhuber, J. (2005). Modeling systems with internal state using evoluno. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1795–1802.
- [407] Williams, G. (1997). *Chaos theory tamed*. CRC Press.
- [408] Wolf, A., Swift, J. B., Swinney, H. L., and Vastano, J. A. (1985). Determining lyapunov exponents from a time series. *Physica D: nonlinear phenomena*, 16(3):285–317.
- [409] Wu, B., Wang, Z., and Wang, L. (2024). Interpretable corn future price forecasting with multivariate time series. *Journal of Forecasting*.
- [410] Wu, D., Wang, X., Su, J., Tang, B., and Wu, S. (2020). A labeling method for financial time series prediction based on trends. *Entropy*, 22(10):1162.
- [411] Wu, D., Wang, X., and Wu, S. (2022). Jointly modeling transfer learning of industrial chain information and deep learning for stock prediction. *Expert Systems with Applications*, 191:116257.
- [412] Wyffels, F. and Schrauwen, B. (2010). A comparative study of reservoir computing strategies for monthly time series prediction. *Neurocomputing*, 73(10-12):1958–1964.

- [413] Xiao, D. and Su, J. (2022). Research on stock price time series prediction based on deep learning and autoregressive integrated moving average. *Scientific Programming*, 2022(1):4758698.
- [414] Xu, B., Su, X., and He, Y. (2024a). Index prediction model based on lasso-pca and deep learning. *International Journal of Crowd Science*, 8(4):176–183.
- [415] Xu, D., Lan, J., and Principe, J. C. (2005). Direct adaptive control: an echo state network and genetic algorithm approach. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 3, pages 1483–1486. IEEE.
- [416] Xu, Z., Wang, Y., Feng, X., Wang, Y., Li, Y., and Lin, H. (2024b). Quantum-enhanced forecasting: Leveraging quantum gramian angular field and cnns for stock return predictions. *Finance Research Letters*, 67:105840.
- [417] Xue, Y., Yang, L., and Haykin, S. (2007). Decoupled echo state networks with lateral inhibition. *Neural Networks*, 20(3):365–376.
- [418] Yan, B., Aasma, M., et al. (2020). A novel deep learning framework: Prediction and analysis of financial time series using ceemd and lstm. *Expert systems with applications*, 159:113609.
- [419] Yan, X., Weihan, W., and Chang, M. (2021). Research on financial assets transaction prediction model based on lstm neural network. *Neural Computing and Applications*, 33(1):257–270.
- [420] Yañez, C., Kristjanpoller, W., and Minutolo, M. C. (2024). Stock market index prediction using transformer neural network models and frequency decomposition. *Neural Computing and Applications*, pages 1–21.
- [421] Yang, C. and Guo, S. (2021). Inflation prediction method based on deep learning. *Computational Intelligence and Neuroscience*, 2021(1):1071145.
- [422] Yang, C., Zhai, J., and Tao, G. (2020). Deep learning for price movement prediction using convolutional neural network and long short-term memory. *Mathematical Problems in Engineering*, 2020(1):2746845.
- [423] Yang, S. (2021). A novel study on deep learning framework to predict and analyze the financial time series information. *Future Generation Computer Systems*, 125:812–819.
- [424] Yang, X.-S. and Deb, S. (2009). Cuckoo search via lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)*, pages 210–214. Ieee.
- [425] Yang, Z., Keung, J., Kabir, M. A., Yu, X., Tang, Y., Zhang, M., and Feng, S. (2021). Acomnn: Attention enhanced compound neural network for financial time-series forecasting with cross-regional features. *Applied Soft Computing*, 111:107649.

- [426] Yildirim, D. C., Toroslu, I. H., and Fiore, U. (2021). Forecasting directional movement of forex data using lstm with technical and macroeconomic indicators. *Financial Innovation*, 7:1–36.
- [427] Yilmaz, F. M. and Yildiztepe, E. (2024). Statistical evaluation of deep learning models for stock return forecasting. *Computational Economics*, 63(1):221–244.
- [428] Yin, W., Kann, K., Yu, M., and Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.
- [429] Yoo, S., Jeon, S., Jeong, S., Lee, H., Ryou, H., Park, T., Choi, Y., and Oh, K. (2021). Prediction of the change points in stock markets using dae-lstm. *Sustainability*, 13(21):11822.
- [430] Yperman, J. and Becker, T. (2016). Bayesian optimization of hyperparameters in reservoir computing. *arXiv preprint arXiv:1611.05193*.
- [431] Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. (2020). Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.
- [432] Zaznov, I., Kunkel, J. M., Badii, A., and Dufour, A. (2024). The intraday dynamics predictor: a trio-flow fusion of convolutional layers and gated recurrent units for high-frequency price movement forecasting. *Applied Sciences*, 14(7):2984.
- [433] Zein, M. H., Yudistira, N., and Adikara, P. P. (2024). Indonesian stock price prediction using neural basis expansion analysis for interpretable time series method. *Journal of ICT*, 23(3):361–392.
- [434] Zhan, Z. and Kim, S.-K. (2024). Versatile time-window sliding machine learning techniques for stock market forecasting. *Artificial Intelligence Review*, 57(8):209.
- [435] Zhang, C., Sjarif, N. N. A., and Ibrahim, R. (2024). 1d-capsnet-lstm: A deep learning-based model for multi-step stock index forecasting. *Journal of King Saud University-Computer and Information Sciences*, 36(2):101959.
- [436] Zhang, L., Feng, L., Chen, K., and Lai, C. H. (2021a). Edge of chaos as a guiding principle for modern neural network training. *arXiv preprint arXiv:2107.09437*.
- [437] Zhang, P., Yang, J.-Y., Zhu, H., Hou, Y.-J., Liu, Y., and Zhou, C.-C. (2021b). Failure in stock price prediction: A comparison between the curve-shape-feature and non-curve-shape-feature modes of existing machine learning algorithms. *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL*, 16(6).
- [438] Zhang, W., Li, S., Guo, Z., and Yang, Y. (2023). A hybrid forecasting model based on deep learning feature extraction and statistical arbitrage methods for stock trading strategies. *Journal of Forecasting*, 42(7):1729–1749.

Bibliography

- [439] Zhao, R., Lei, Z., and Zhao, Z. (2024). Research on the application of deep learning techniques in stock market prediction and investment decision-making in financial management. *Frontiers in Energy Research*, 12:1376677.
- [440] Zhong, S., Xie, X., Lin, L., and Wang, F. (2017). Genetic algorithm optimized double-reservoir echo state network for multi-regime time series prediction. *Neurocomputing*, 238:191–204.
- [441] Zhuge, Q., Xu, L., and Zhang, G. (2017). Lstm neural network with emotional analysis for prediction of stock price. *Engineering letters*, 25(2).