



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E
DE COMPUTAÇÃO



ELSA - Expanded Latent Space Autoencoder Architecture for Feature Extraction: A Case Study Application to Covid-19 Time Series Forecasting

Emerson Vilar de Oliveira

Advisor: Prof. Dr. Luiz Marcos Garcia Gonçalves

Doctorate Thesis presented to the Graduate Program in Electrical and Computer Engineering of UFRN (concentration area: Computer Engineering) as part of the requirements for obtaining the title of Doctor in Sciences.

Ordem number PPgEEC: D364
Natal, RN, September, 20th, 2024

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Oliveira, Emerson Vilar de.

ELSA - Expanded Latent Space Autoencoder Architecture for Feature Extraction: A Case Study Application to Covid-19 Time Series Forecasting / Emerson Vilar de Oliveira. - 2024.
91 f.: il.

Tese (doutorado) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Natal, RN, 2024.

Orientação: Prof. Dr. Luiz Marcos Garcia Gonçalves.

1. Redes Neurais Artificiais - Tese. 2. Autoencoder - Tese. 3. Autoencoders Empilhados - Tese. 4. SARS-CoV-2 - Tese. 5. Séries Temporais - Tese. 6. Espaço Latente - Tese. I. Gonçalves, Luiz Marcos Garcia. II. Título.

RN/UF/BCZM

CDU 004.7(043.2)

Acknowledgment

To my parents, my sister, my wife, and everyone who shared time with me on this journey.

To all the teachers I've had throughout my academic life, especially Professors Luiz Marcos and Luiz Affonso, who guided me along this path.

To my fellow students who, amidst all the conversations, taught me Science.

To CAPES, for the financial support received through the project entitled "Methods for predicting the dynamics of viral epidemics and pandemics with analysis of grouped data from the perspective of artificial intelligence", process 88887.506889/2020-00, of the CAPES program "951 - Strategic Emergency Program for the Prevention and Combat of Outbreaks, Endemics, Epidemics and Pandemics", coordinated by Prof. Luiz Marcos Garcia Gonçalves, UFRN.

Abstract

The global SARS-CoV-2 pandemics compelled governments, institutions, and researchers to assess its impact and develop strategies based on general indicators to achieve the most accurate predictions possible, in order to help managers mitigating its effect. While known epidemiological models were naturally used, they often produced uncertain forecasts due to insufficient or missing data. In addition to data limitation, various machine-learning models such as random forests, support vector regression, LSTM, auto-encoders, and traditional time-series models like Prophet and ARIMA—were employed, yielding impressive yet somewhat limited results. Some of these methods struggle with precision when handling multi-variable inputs, which are crucial for problems like pandemics time series prediction that require both short- and long-term forecasting. In response to this challenge, we propose a novel approach for time-series prediction that utilizes a stacked auto-encoder structure. Our model uses n internal autoencoders to process the input and generate different latent spaces for this respective input. Then these different latent spaces are concatenated and the expanded latent space is obtained. We conducted an experiment using previously published data series on COVID-19 cases, deaths, temperature, humidity, and the air quality index (AQI) in São Paulo City, Brazil. This experiment assessed the suitability of our model for short-, medium-, and long-term forecasting. Furthermore, we directly compared our proposed model with two existing works in the literature that have already undergone expert scrutiny. The first comparison places our model among those that use one network for feature extraction and another for predicting the pandemic trends. The second comparison highlights our model's effectiveness in multi-series forecasting of pandemic indicators. The results suggest that our proposed model possesses strong capabilities in both feature extraction and multi-series forecasting, offering improvements over the two comparison works. Finally, the model demonstrates promising forecasting accuracy and versatility across datasets of varying lengths, making it a standout option for time-series forecasting tasks.

Keywords: autoencoder; stacked autoencoder; SARS-CoV-2; forecast; time-series, latent space; artificial neural networks.

Resumo

A pandemia global de SARS-CoV-2 levou governos, instituições e pesquisadores a avaliar seu impacto e a desenvolver estratégias baseadas em indicadores gerais para alcançar as previsões mais precisas possíveis. Embora os modelos epidemiológicos tenham sido amplamente utilizados, eles frequentemente geravam previsões incertas devido a dados insuficientes ou ausentes. Além das limitações de dados, diversos modelos de aprendizado de máquina, como florestas aleatórias, regressão por vetores de suporte, LSTM, autoencoders e modelos tradicionais de séries temporais, como Prophet e ARIMA, foram empregados, resultando em desempenhos impressionantes, mas com algumas limitações. Alguns desses métodos enfrentam dificuldades de precisão ao lidar com entradas multivariadas, o que é crucial para problemas como a previsão de séries temporais em pandemias, que exigem previsões de curto e longo prazo. Em resposta a esse desafio, propomos uma abordagem inovadora para a previsão de séries temporais que utiliza uma estrutura de autoencoder empilhado. Nosso modelo utiliza n autoencoders internos para processar as entradas e gerar diferentes espaços latentes para essas respectivas entradas. Em seguida, esses diferentes espaços latentes são concatenados e o espaço latente expandido é obtido. Conduzimos um experimento usando um conjunto de séries previamente publicados sobre casos de COVID-19, mortes, temperatura, umidade e índice de qualidade do ar (IQA) na cidade de São Paulo, Brasil. Esse experimento avaliou a adequação do nosso modelo para previsões de curto, médio e longo prazo. Além disso, comparamos diretamente nosso modelo proposto com dois trabalhos existentes na literatura que já passaram pelo escrutínio de especialistas. A primeira comparação posiciona nosso modelo entre aqueles que utilizam uma rede para extração de características e outra para prever tendências da pandemia. A segunda comparação destaca a eficácia do nosso modelo na previsão de múltiplas séries de indicadores pandêmicos. Os resultados sugerem que o modelo proposto possui fortes capacidades tanto na extração de características quanto na previsão de múltiplas séries, oferecendo melhorias em relação aos dois trabalhos comparados. Finalmente, o modelo demonstra uma precisão promissora em previsões e versatilidade em conjuntos de dados de diferentes tamanhos, tornando-se uma opção destacada para tarefas de previsão de séries temporais.

Palavras-chave: autoencoder; autoencoders empilhados; SARS-CoV-2; predição; séries temporais; espaço latente; redes neurais artificiais.

Contents

Summary	i
List of Figures	iii
List of Tables	v
1 Introduction	1
1.1 Theme, Problem, and Research Hypothesis	3
1.2 Contribution and Applications	5
1.3 Work Structure	5
List of Symbols and Abbreviations	1
2 Theory	7
2.1 Time Series	7
2.2 Artificial Neural Networks	10
2.2.1 Perceptrons and Multilayer Perceptrons	11
2.2.2 Backpropagation	12
2.2.3 Stochastic Gradient Descent	13
2.3 Autoencoders	14
3 Related Works	17
4 Problem and Proposal	27
4.1 Time Series Prediction	27
4.1.1 COVID-19 Time Series Prediction	28
4.2 Expanded Latent Space Auto-encoder	31
4.2.1 Modified Auto-Encoder	31
4.2.2 Multi-Variable ELSA	33
5 Implementation	35
5.1 Initial Validations	35
5.2 Proposal Validation	35
5.3 Reproducibility	39
5.4 Identical Internal Autoencoders	40
5.5 Different Datasets	41
5.6 Influence on Classification	41

5.7	Conventional Latent Space $\times$ Expanded Latent Space	44
5.8	Variation of N	48
6	Experiments	57
6.1	COVID-19 Dataset	57
6.2	Short-medium-long term forecasting	58
6.3	Works Comparison	61
7	Results and Analysis	67
7.1	Results from Short-medium-long-term Forecasting	67
7.2	Results from Works Comparison	73
7.3	Discussion	75
8	Conclusion	81
	References	84

List of Figures

2.1	Graphical representation of a linear trend component in a time series. . . .	8
2.2	Graphical representation of a sinusoidal seasonality component in a time series.	8
2.3	Graphical representation of a normal (Gaussian) residual component in a time series.	9
2.4	Graphical representation of a time series.	9
2.5	Representative diagram of an artificial neuron.	11
2.6	Example of a multilayer perceptron with three layers.	12
2.7	High-level representation of an autoencoder.	15
4.1	Plot of the target time series for prediction in this work.	29
4.2	High-level representation of how the flow between input, a trained model, and prediction should function.	30
4.3	First proposed architecture named Modified Auto-Encoder model. Image used with author's authorization.	31
4.4	Representative figure of the sequence flowchart of entries into ELSA. . .	32
4.5	Flowchart representation of how a multi-variable ELSA network works. .	33
5.1	Image depicting the plot of generated values in the latent space for each of the internal encoders separately.	37
5.2	Image depicting the plot of distributions generated from the calculation of the mean for each of the values obtained in the internal encoders, along with a moving average plot of the distribution of values obtained in the expanded latent space.	37
5.3	Plot of the distributions obtained from training a model equal to that presented in the previous validation, with a change in the value of the seed generator.	39
5.4	Image with the plot of distributions generated from the calculation of the mean for each of the values obtained in the internal encoders, along with a plot of moving average of the distribution of values obtained in the expanded latent space for a network with two identical internal autoencoders.	40
5.5	Visual representation of the distribution of means generated from <i>encoders</i> ₁ , <i>encoders</i> ₂ , and expanded latent space for each trained model.	42
5.6	Boxplot image of the distributions generated from extracting the accuracy of the test dataset for the described validation.	45
5.7	Boxplot image of the distributions generated from the extraction of test dataset accuracy for the described validation.	47

5.8	Visual representation in boxplot form of the distributions generated from the extraction of test dataset accuracy for the described validation.	49
5.9	Boxplot representation of the distributions generated from the extraction of accuracy from the test dataset for the described validation.	50
5.10	Boxplot representation of the distributions generated from the extraction of accuracy from the test dataset for the described validation.	52
5.11	Boxplot representation of the distributions generated from the extraction of accuracy from the test dataset for the described validation.	54
6.1	Plot of the number of humidity, temperature, aqi, and their moving average for the São Paulo city, Brazil, from 2020-04-01 to 2023-02-28.	58
7.1	Boxplot representation of the distributions generated from the extraction of RMSE from the test dataset for the short-medium-long-term forecasting experiment.	68
7.2	Plot of the prediction for the time series related to the number of deaths in São Paulo - Brazil from 2023-01-01 to 2023-02-25.	70
7.3	Plot of the prediction for the time series related to the number of deaths in São Paulo - Brazil from 2023-01-01 to 2023-02-25.	71
7.4	Plot of the prediction for the time series related to the number of deaths in São Paulo - Brazil from 2023-01-01 to 2023-02-25.	72
7.5	Prediction plot for the time series related to the number of global COVID-19 cases.	74
7.6	Prediction plot for each country time series related to the number of COVID-19 cases.	77

List of Tables

5.1	Table of hyperparameters used to demonstrate the creation of an expanded latent space.	36
5.2	Table of hyperparameters used in each ELSA network for classifier training.	42
5.3	Table of sample sizes (images) of the datasets for training each ELSA network for the MNIST dataset.	43
5.4	Table of hyperparameters used for the classifier networks.	44
5.5	Table of hyperparameters used in each ELSA network with expanded latent space and conventional latent space for classifier training.	46
5.6	Table of quantitative values obtained from the distributions generated in the comparison validation for the FashionMNIST dataset.	48
5.7	Table of quantitative values obtained from the distributions generated in the comparison validation for the FashionMNIST dataset.	51
5.8	Table of quantitative values obtained from the distributions generated in the comparison validation for the CIFAR10 dataset.	53
5.9	Table of quantitative values obtained from the distributions generated in the comparison validation for the CIFAR10 dataset.	54
6.1	Table of hyperparameters used in each ELSA network.	60
6.2	Table of hyperparameters used for the predictor networks.	60
6.3	Table of hyperparameters used in the comparison ELSA network.	62
6.4	Table of hyperparameters used for the predictor network.	62
6.5	Table of hyperparameters used in the comparison ELSA network.	64
6.6	Table of hyperparameters used for the predictor network.	64
7.1	Table of quantitative values obtained from the distributions generated in the short, medium, and long-term prediction evaluation experiment.	68
7.2	Table of quantitative values obtained from the average RMSE for each input series in the short, medium, and long-term prediction evaluation experiment.	69
7.3	Table of quantitative values obtained from the RMSE for each input country series in the prediction evaluation comparison experiment.	75

Chapter 1

Introduction

Respiratory viruses are responsible for a multitude of human diseases and significantly impact global morbidity and mortality rates (James et al. 2018, Fernandes et al. 2018). Among these viruses, the Coronaviridae family includes several critical agents of respiratory illnesses (Mousavizadeh & Ghasemi 2021, Cui et al. 2019). Despite their long-standing presence, coronaviruses were not considered highly pathogenic to humans until the emergence of SARS-CoV in 2003 and MERS-CoV in 2012 (Drosten et al. 2003, Ksiazek et al. 2003, Zaki et al. 2012).

The COVID-19 outbreak was first reported in December 2019, leading to substantial losses and requiring several non-pharmaceutical interventions to achieve initial control (ECDC n.d.). On January 30, 2020, the World Health Organization (WHO) declared COVID-19 an international public health emergency. By April 1, 2020, COVID-19 cases and deaths had been reported on every continent, with the United States, Spain, France, Italy, and the United Kingdom emerging as the epicenters. The United States reported over 700 deaths per day by early April, with this number escalating to 2000 daily deaths although the same month. The mutation rates and transmission capabilities of SARS-CoV-2 surpassed those of its predecessors, SARS-CoV and MERS-CoV (Mousavizadeh & Ghasemi 2021, Jang et al. 2020, He et al. 2020). Additionally, the disease's impact varied widely in mild to moderate cases, influenced significantly by factors such as age and gender (Lechien et al. 2020). The crisis also brought to light the disparities in healthcare access and outcomes across different regions and populations. Vulnerable groups, including the elderly, individuals with pre-existing conditions, and economically disadvantaged communities, were disproportionately affected by the virus. The rapid global spread and severe impact of COVID-19 underscored the necessity for a coordinated international response and robust public health infrastructure. Governments around the world had to implement unprecedented measures, including lockdowns, travel restrictions, and social distancing protocols, to slow the virus's spread and prevent healthcare systems from being overwhelmed. These interventions were crucial in managing the initial outbreak and buying time for the development of medical countermeasures (Anastassopoulou et al. 2020, Organization & others 2020).

Moreover, the pandemic highlighted the importance of continuous monitoring of viral mutations and the adaptability of public health strategies. As the virus evolved, it became clear that real-time genomic surveillance was essential in identifying new variants and understanding their potential impact on transmissibility, severity, and vaccine efficacy.

This adaptability in response strategies was a key factor in managing subsequent waves of infections and mitigating the pandemic's impact (Cyranoski 2021, Lambrou 2022).

To aid in disease control, researchers worldwide have proposed various estimation techniques to predict case numbers and model virus spread (Cooper et al. 2020, Yang et al. 2020, Djaparidze & Lois 2020). However, the global pandemic may be exacerbated by numerous factors not immediately evident, such as climatic conditions and social and political influences (Aragão et al. 2022). Given the complexity of real-world scenarios, many elements can directly or indirectly affect the pandemic's behavior. Current studies are still investigating the factors that significantly influence the spread of SARS-CoV-2. Factors such as air quality, social distancing, and the use of specific equipment are known to impact the spread of the virus. Although researchers are not yet certain which factors are most influential, there is emerging guidance on those that may promote the virus's spread (Elsaid & Ahmed 2021, Xu et al. 2020, Ismail et al. 2021, Freitas et al. 2020, Araújo et al. 2021).

Currently, over 770 million confirmed cases and more than 7 million deaths have been recorded worldwide, yielding a mortality rate of approximately 1% of infected people. The global scale of the COVID-19 pandemic has highlighted the critical need for robust surveillance systems and advanced forecasting models to predict the spread of such viruses. Data-driven forecasting models, in particular, became invaluable tools in anticipating the trajectory of COVID-19 outbreaks, helping to inform policy decisions and public health interventions. Among these techniques, as first applicable methodology, the SIR (Susceptible-Infected-Recovered), SEIR (Susceptible-Exposed-Infected-Recovered), and SEIRS (Susceptible-Exposed-Infected-Recovered-Susceptible) models are prominent (Cooper et al. 2020, Yang et al. 2020, Djaparidze & Lois 2020). These traditional compartmental models typically utilize population data divided into subgroups, estimating the rate and probability of individuals moving from one subgroup to another. These subgroups vary by model, typically considering the number of infected, recovered, and other indicators. Initially, these models are extremely valuable and straightforward to apply, being relatively easy to understand in theory.

However, the complexity of a global pandemic involves numerous aggravating factors, such as climatic, social, and political influences. For instance, variations in climate can affect the virus's survivability and transmission rates. Air quality, can influence respiratory health, potentially exacerbating the effects of COVID-19. Social behaviors and political decisions can significantly alter the effectiveness of interventions. Social distancing measures, though effective, rely heavily on public compliance and can be influenced by cultural and political factors. The use of personal protective equipment (PPE) is another critical factor, with its effectiveness depending on proper usage and availability. Despite the recognition of these factors, it remains uncertain which have the greatest influence, but there is significant direction regarding those that may stimulate virus spread (Elsaid & Ahmed 2021, Xu et al. 2020, Ismail et al. 2021, Freitas et al. 2020). Therefore, these traditional methodologies may not be entirely sufficient to fully comprehend the behavior of the current COVID-19 pandemic. As a result, there is a need for more sophisticated approaches that can incorporate a broader range of variables and adapt to the evolving nature of the pandemic (Kong et al. 2021).

In response to these complexities, researchers have turned to more complex data-driven methodologies to better understand the consequences and behavior of the COVID-19 pandemic. These methods aim to predict infection, confirmed cases, and death numbers, as well as virus transmission rates, by utilizing most populous datasets and advanced computational techniques. Complex data-driven methods often leverage artificial intelligence (AI) to process and analyze data, uncovering patterns and insights that might be missed by traditional models. For instance, machine learning algorithms can identify correlations between various factors and COVID-19 outcomes, providing a more nuanced understanding of the pandemic's dynamics. Moreover, these techniques can generate useful tools for specific actions, such as predicting healthcare resource needs or assessing the impact of policy changes, and make future estimates from time series data.

A time series approach is particularly valuable for modeling pandemic data and its spread globally, given the temporal dependencies in its values. Time-series analysis can reveal trends and cycles in the data, helping to forecast future developments and inform public health strategies. Ultimately, data-driven models can assimilate complex information from this time-series to aid in decision-making, providing a more robust framework for responding to the pandemic. Among the most common techniques in data-driven models are artificial neural networks, which are strong allies in using this type of data as information sources. Neural networks excel at approximating complex functions, making them well-suited for modeling real-world problems with many interacting variables (Coppin 2004).

Given the pandemic, developing methods to understand pandemic-related data is of considerable relevance. Not only does this provide meaning to the data, but it also enables us to better face potential future events. By understanding the factors that drive the spread of COVID-19, we can design more effective interventions and improve our preparedness for future pandemics. Motivated by this context, this work proposes a new data-driven model aimed at predicting COVID-19 related time series. Specifically, the proposed model is an autoencoder-based artificial neural network, with specific modifications in its operation. Autoencoders are a class of neural networks designed for both supervised and unsupervised learning, capable of discovering the underlying structure of the data and reducing its dimensionality. This model aims to predict the number of deaths or other related time-series for a given dataset, capturing the complex relationships between various factors.

In summary, the COVID-19 pandemic has not only reaffirmed the threat posed by respiratory viruses but has also highlighted the necessity for advanced predictive models and rapid response strategies to mitigate their impact on global health. The lessons learned from SARS-CoV-2 can inform future efforts to manage and prevent similar outbreaks, ensuring that societies are better prepared to face emerging infectious diseases.

1.1 Theme, Problem, and Research Hypothesis

The general theme that we deal with in this research is time series forecasting (or prediction). Going to a more detailed specificity, we work with time series prediction for the COVID-19 data provided by several types of sources and acquisition systems. Time series

prediction is a widely addressed topic in the literature, with various approaches treating prediction differently. However, these approaches and methods have a certain margin of error. These errors and uncertainties are inherent to the problem, as any prediction problem involves such errors.

Time series prediction can additionally be influenced by the nature of predicted values. When predicting rational numbers, the error and model evaluation differ from classifying or predicting integer values, for example. Predicting multiple rational values can accumulate small errors, which, when summed, may underestimate the true extent of the issue. Generally, this problem allows for numerous application variations, such that there is no definitive model and methodology that solve for all cases. Factors such as data nature, sampling, training methodology, network type (for neural networks), and others actively influence model responses.

One of the main problems in predicting time series is the potential for errors at various points in the task. Therefore, there is room for new models with different approaches to propose modifications and improvements in this area. Consequently, this work's problem can be formulated in the form of a single question, as:

“Is it possible to improve the prediction of COVID-19 related time series using data-driven models?”

On trying to answer this single question, several other questions appear as:

1. Is there a single AI based method that solves the problem, for all data series types?
2. How can the model be generalized to predict other pandemics or similar public health crises?
3. Can a stacked autoencoder effectively capture the temporal dependencies in COVID-19 time series data?
4. How does the model handle different types of time-series data, such as case numbers, death rates, and environmental markers?
5. Does the model make more accurate predictions for COVID-19-related time series in the short, medium, or long term?
6. What is the role of dimensionality embedding in the performance of a stacked autoencoder for time-series forecasting?
7. What are the potential trade-offs between model complexity and performance in the context of your proposed methodology?
8. How does the proposed model compared to other machine learning/AI based techniques?

Trying to answer the above questions, we can formulate our research hypothesis, which can be defined by using the following assertive:

“It is possible to propose modifications or a new artificial neural network data-driven model that, using COVID-19 related time series as inputs, improves the accuracy of predicting the number of deaths or other pandemic indicator.”

Given the topic's relevance and the past pandemic context, this work aims to validate the proposed model. Thus, evaluating if the proposed methodology can predict one (or multiple) COVID-19 related time series comparable to other state-of-the-art methodologies. This research was conducted within the context of the global pandemic triggered by

SARS-CoV-2, impacting a significant portion of the world, which serves as our primary motivation.

The COVID-19 crisis emerged as an unforeseen global shock, with no prior data to accurately predict its extensive impact. The experience with SARS-CoV-2, along with insights from previous outbreaks of SARS-CoV and MERS-CoV, highlighted the risks societies face when a highly transmissible virus spreads, causing widespread disruption. This includes economic downturns, severe strain on healthcare workers, such as nurses and sanitation personnel, and the health of infected individuals. Vaccination efforts have played a crucial role in combating such health threats; however, the urgency often outpaces the vaccine delivery timeline. Consequently, it becomes imperative to develop a comprehensive map of events and patterns, or to identify health determinants, drivers, and incidents that can be linked to the spread, pathogenesis, and mortality of viral pathogens. In the context of the COVID-19 outbreak, these elements should correlate with disease spread, reflecting the number of cases and deaths, whether positively or negatively.

Understanding when and how to implement government interventions relies on a series of data projections and potential events that can influence virus transmission and cause socioeconomic impacts. The aim is to support community leaders and healthcare providers in managing emergencies like SARS-CoV-2 and other potential future infections through informed decision-making based on time-series forecasting.

1.2 Contribution and Applications

This study aims to contribute to the problem of time series prediction related to COVID-19 pandemic. Despite a range of techniques and methodologies for time series prediction, new methods addressing the problem in varied ways can be proposed. Besides its applicability to COVID-19 related time series, the proposed model can be applied in various fields, such as failure prediction and diagnostics, stock market prediction, equipment lifespan prediction, among others. Generally, any problem with temporally related data samples can apply the proposed methodology. Also, the proposed model nature allow it to be applied out of time series forecasting task, such as image classification or word embedding.

1.3 Work Structure

Despite this Introduction Chapter, this thesis is structured into seven more chapters, each dedicated to a specific aspect of the research.

The Chapter 2 provides the essential mathematical background required to understand the proposed model. It covers fundamental concepts in time-series, artificial neural networks, autoencoders, and machine learning techniques that underpin the research.

In the Chapter 3, we review the existing literature on COVID-19 time-series forecasting. It includes an analysis of conventional machine learning models, artificial neural networks, and models utilizing autoencoders. The goal is to position the current research

within the broader context of existing studies and identify gaps that the proposed model aims to fill.

The Problem and Proposal chapter, Chapter 4, formally defines the problem addressed by the thesis and presented the proposed solution. It presents the mathematical formulation of the COVID-19 time-series forecasting problem and outlines the novel approach developed to improve prediction accuracy and reliability.

In Chapter 5 we detail the implementation of the proposed model. This includes the architecture used for model validation, the possible process of data collection and preprocessing, and the validation steps taken to ensure the model's robustness.

The Chapter 6 describes the experiments conducted to evaluate the proposed model's performance for COVID-19 time-series task. Here, we specifies the datasets used, the experimental setup, and the metrics employed to measure forecasting accuracy. Also, we show the methodologies of the comparison works, used to point our proposed model on literature.

The results of the experiments are presented and analyzed in the Chapter 7. It includes a comprehensive discussion on the model's performance, the comparisons with existing models, and an interpretation of the findings in the context of COVID-19 time-series forecasting.

The final chapter (Chapter 8) summarizes the key findings of the research, discusses the implications of the results, and suggests directions for future work. It reflects on the contributions of the thesis to the field of time-series forecasting and public health preparedness.

By structuring the thesis in this manner, we aim to provide a coherent and comprehensive narrative that guides the reader through the research journey, from theoretical foundations to practical implementation and critical analysis. The overarching goal is to contribute valuable insights and tools to the effort of mitigate the impacts of global pandemics through advanced predictive modeling.

Chapter 2

Theory

This chapter will cover the theoretical knowledge necessary to understand the remainder of the text. Initially, we will succinctly present concepts defining time series, as well as their components, formulation, and prediction. Next, basic concepts related to artificial neural networks, weight adjustment algorithms and optimization. Finally, mathematical concepts regarding autoencoders and some of their main variations in the literature will be introduced.

2.1 Time Series

A time series can be defined as a sequence of values ordered chronologically and observed over time. Although time is a continuous (or analog) measure, a time series is typically sampled at constant intervals denominated as sampling frequency (Torres et al. 2021). This definition is valid for most applications; however, in some circumstances, a time series may vary in its definition. The first and most common situation occurs when there are missing or lost data. For example, there are various situations where data is missing for certain periods. This issue generally arises due to a lack of reliability in data collection. There are several methods to handle missing data, the most common being their replacement with other information, such as an average, or the complete omission of these missing data. Another common problem in time series is the presence of outliers. Outliers are values that do not statistically correspond to the distribution of the series. In other words, they are significantly different from the values considered common for a given dataset. Statistical methods are most commonly used to remove or integrate these outlier values into the series. Finally, if a data sequence has irregular intervals between data collection, it is termed a time series with non-uniform spacing, or data streams (Torres et al. 2021).

In addition to the points mentioned, in general, a time series can be divided into three main components: trend, seasonality, and irregular components, also called residuals (or noise) (Torres et al. 2021, Shumway et al. 2000). Each of these components can be described as follows:

- **Trend:** As the name suggests, this is the general direction, sense, or movement of the signal (series) over the observed time. This does not consider seasonality or

residual components. Common types of trends in time series include linear, exponential, and parabolic. In another nomenclature, the trend is also called long-term variation. Figure 2.1 shows a graphical representation of a linear trend component in a time series.

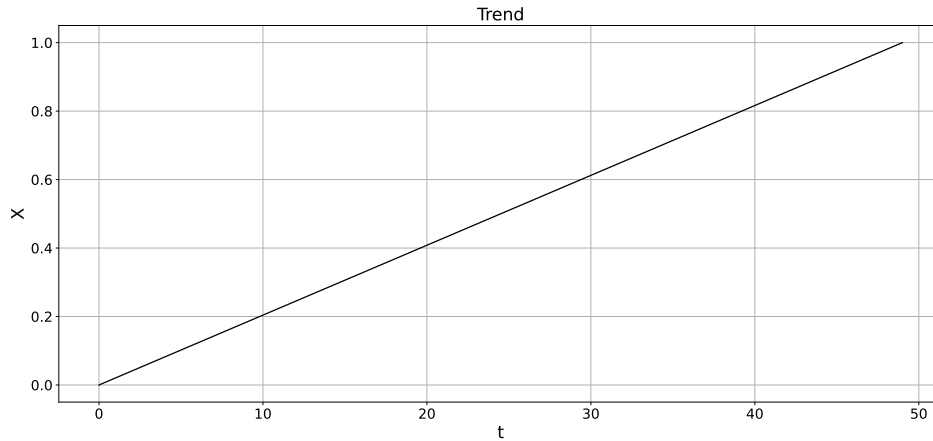


Figure 2.1: Graphical representation of a linear trend component in a time series.

- **Seasonality:** This component represents variations that occur at regular time intervals in the series. This characteristic can convey important information about the signal when the signal's behavior repeats similarly over time intervals. In real signals, this effect can be caused by various factors, such as climate, economic cycles, seasons, or even holidays. Figure 2.2 shows a graphical representation of a sinusoidal seasonality in a time series.

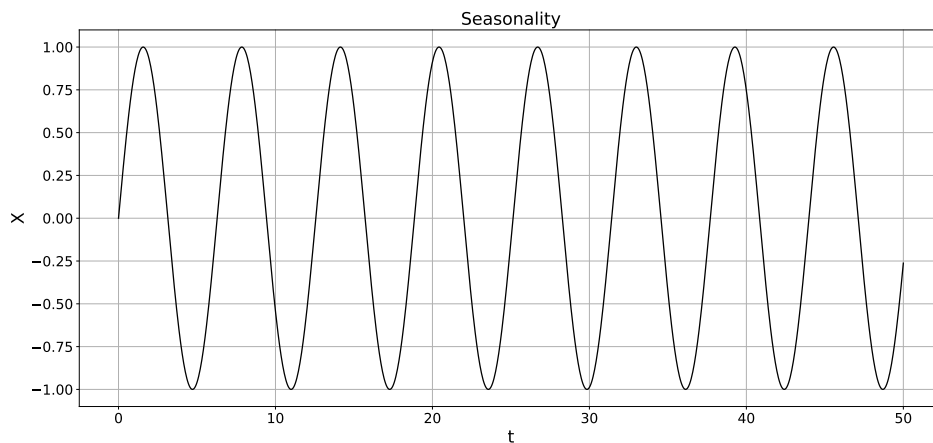


Figure 2.2: Graphical representation of a sinusoidal seasonality component in a time series.

- **Residuals:** With the other two components defined, calculated, and removed, the series may still contain some residual values. These values can be significant enough to mask the actual trend and seasonality of the series. In such cases, these residuals are termed outliers. There are various origins for this type of behavior, which can

complicate the prediction of the desired signal. However, once the origin or statistical composition is determined, the residual can be beneficial for possible changes in the overall trend of the series. Figure 2.3 shows a graphical representation of a normal (Gaussian) residual component in a time series.

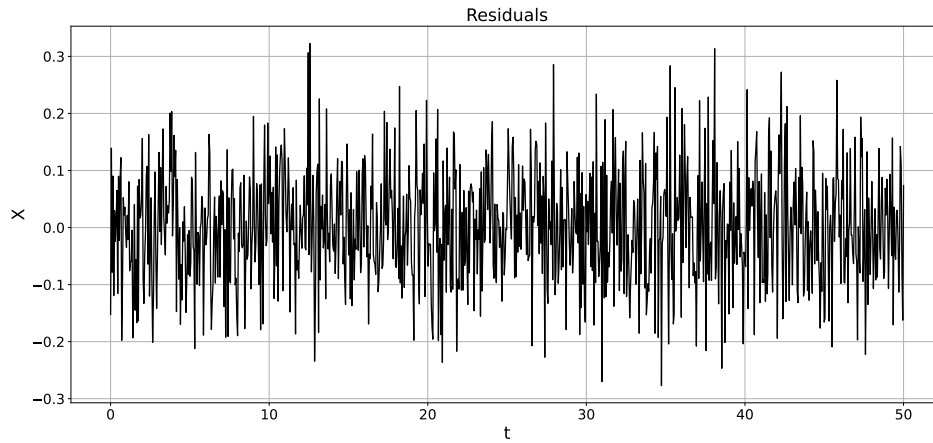


Figure 2.3: Graphical representation of a normal (Gaussian) residual component in a time series.

A time series can essentially be considered as the combination of these three components. Figure 2.4 shows a graphical representation of a time series formed by the combination of the components presented in Figures 2.1, 2.2, and 2.3.

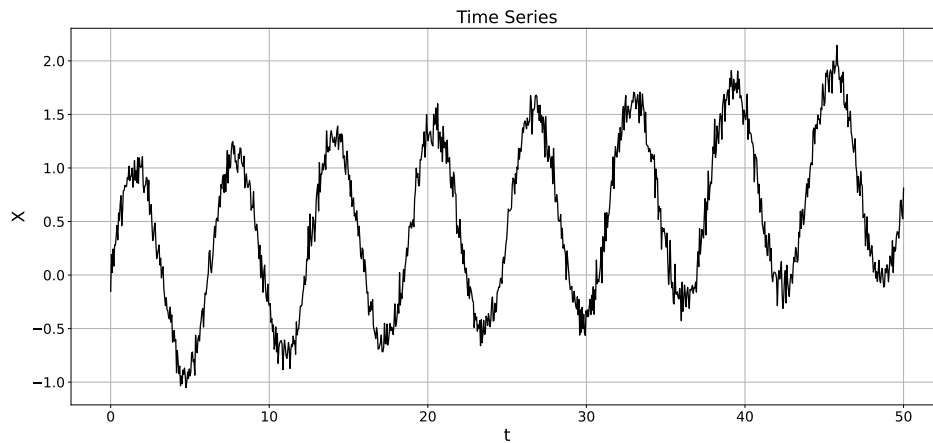


Figure 2.4: Graphical representation of a time series.

From Figure 2.4, it is possible to observe how the three components form the original signal. Visually, the general upward trend component, the sinusoidal seasonality component, and the Gaussian noise residual component are perceptible. Time series obtained from real-world data exhibit significant irregularity in these components. Additionally, real series are usually non-stationary, meaning their mean and variance are not constant over time. This behavior complicates the modeling and prediction of time series by less

complex models. For this reason, some more classical time series modeling and prediction models attempt to decompose the series into the three previously described components and then perform prediction separately (Torres et al. 2021).

2.2 Artificial Neural Networks

The operation of artificial neural networks is inspired by the behavior of the human brain. The human brain contains billions of simple processing structures called neurons. Biological neurons consist primarily of a cell body, axon, and dendrites, which are responsible for transmitting and receiving stimuli. One accepted theory is that after a chemical reaction, a neuron sends an action through the axon towards the synapses connected to the dendrites of other neurons. Individually, these neurons are relatively simple, but in a network composed of billions of these structures, they can process complex information extremely quickly (Coppin 2004, Haykin 2007).

Artificial Neural Networks (ANNs) attempt to function analogously to these structures, aiming to model a function that responds to a specific problem. ANNs have a significantly smaller number of neurons and far fewer connections than a human brain, making them less complex. To simulate biological neurons, artificial neurons receive multiple inputs, which, after being processed, produce outputs that feed into functions known as activation functions. These functions, in some cases, delimit how much the neuron's output can vary. The most common activation function is the linear threshold function, where, after the weighted sum of the inputs is calculated, the result is compared to a defined threshold. If this sum exceeds the threshold, the neuron transmits its information; otherwise, a null information is transmitted. The weights of artificial neurons are analogous to the synapses of biological neurons, responsible for weighting the input values, which are then summed with a bias, configuring a linear operation (Coppin 2004). Figure 2.5 exemplifies the structure of a neuron.

As shown in Figure 2.5, $w_{k1}, w_{k2}, \dots, w_{km}$ are the weights of the artificial neuron, which receives m inputs. Along with the weights, a bias denoted b_k is added at the input of the activation function ϕ . Equation (2.1) describes the summation of the inputs to the artificial neuron u_k .

$$u_k = \sum_{i=1}^m w_{ki}x_i \quad (2.1)$$

From Equation (2.1), u is the sum of the m inputs x to neuron k , weighted by the weights w . After this summation, the bias is added. Thus, Equation (2.2) describe this operation.

$$v_k = u_k + b_k \quad (2.2)$$

From equation (2.2), we obtain the result v_k , which is then subjected to the activation function ϕ . The threshold activation function is also commonly referred to as the Heaviside function or step function. Equation (2.3) describes this type of activation function.

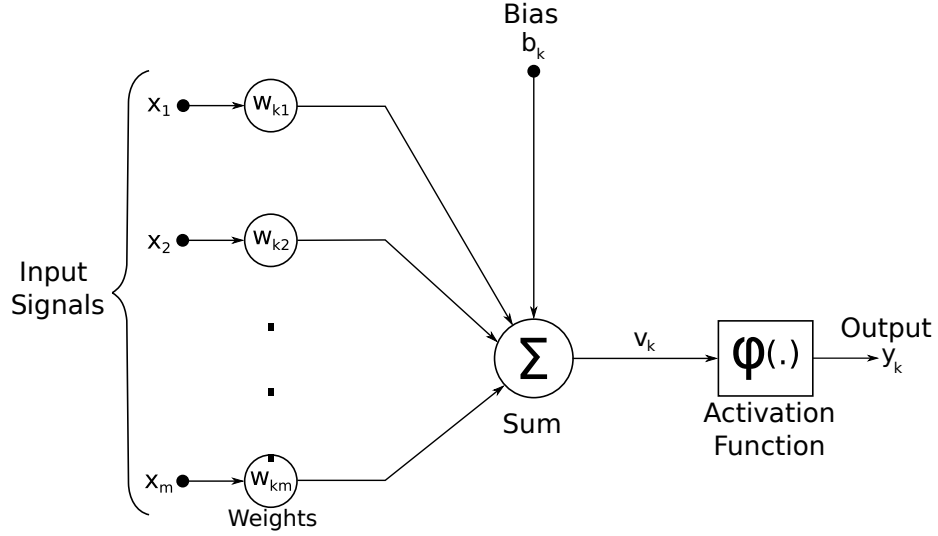


Figure 2.5: Representative diagram of an artificial neuron.

$$\phi(v_k) = \begin{cases} 1, & \text{for } v_k \geq \text{threshold} \\ 0, & \text{for } v_k < \text{threshold} \end{cases} \quad (2.3)$$

In this case, y , described in Figure 2.5, is the output of the activation function for neuron k (Haykin 2007).

2.2.1 Perceptrons and Multilayer Perceptrons

The perceptron is the simplest form of an artificial neural network, consisting of only one neuron, capable of modeling a function that separates its inputs into two classes (Coppin 2004). The perceptron essentially works by adjusting its weights and bias to model a function that describes a surface capable of delineating two linearly separable classes. To differentiate three or more classes, additional perceptrons are required, but these classes must remain linearly separable for the perceptron to function adequately (Haykin 2007). As previously mentioned, real-world time series are complex and rarely meet this criterion. To address problems like this, multilayer perceptrons (MLPs) were developed, capable of modeling more complex problems. An MLP consists of more than one layer of interconnected perceptrons. Figure 2.6 abstractly illustrates an MLP with three layers, where each circle represents a perceptron, also called neuron or node.

From Figure 2.6, the arrows on the left side of the network represent the inputs, and the arrows on the right side represent the outputs. The arrows connecting one node to another represent the transmission of information between layers. This type of network is feed-forward, where the flow of information moves from the input perceptrons to the output perceptrons.

The neurons in MLP networks function similarly to a standard perceptron, following all the equations expressed in Section 2.2. As seen in Equation (2.1), there are values

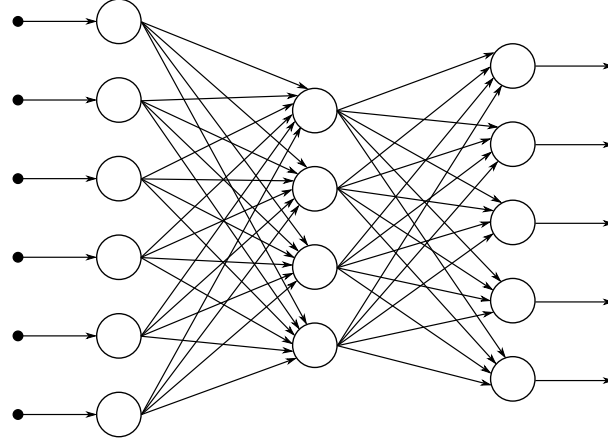


Figure 2.6: Example of a multilayer perceptron with three layers.

called weights, responsible for weighting the inputs of the neurons. These weights enable the neurons to model functions that approximate the inputs to the desired output values. Due to the larger number of perceptrons in MLP networks, appropriate methods were developed to adjust these weights.

2.2.2 Backpropagation

Backpropagation is a fundamental algorithm used to train artificial neural networks. It is a process which the network learns by adjusting its weights based on the error of its output. This algorithm is essential for the efficiency and effectiveness of deep learning models (Coppin 2004, Goodfellow et al. 2016). The algorithm lays on the basic concepts of forward propagation, loss calculation and backward propagation. Given a neural network with L layers, the forward propagation starts when the input $\mathbf{x}$ is fed into the network. For each layer l , the input to the layer is $\mathbf{a}^{(l-1)}$. Equation (2.4) shows how the output of the forward propagation step is calculated.

$$\begin{aligned} z^{(l)} &= \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \\ \mathbf{a}^{(l)} &= \sigma(z^{(l)}) \end{aligned} \quad (2.4)$$

From Equation (2.4), $\mathbf{W}^{(l)}$ are the weights, $\mathbf{b}^{(l)}$ are the biases, $\mathbf{z}^{(l)}$ is the linear combination, and σ is the activation function. The final output $\mathbf{a}^{(L)}$ is produced after passing through all layers. The loss $\mathcal{L}$ computation step is performed using a suitable loss function, such as mean squared error (MSE) for regression or cross-entropy loss for classification. For example, Equation (2.5) shows how the mean squared error is computed.

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (2.5)$$

From Equation (2.5), y_i is the true target value and $\hat{y}_i$ is the predicted value. The final

value $\mathcal{L}$ is the MSE concerning the true and predicted value. The last step is the backward propagation, which calculates the gradient of the loss function with respect to each weight by applying the chain rule of calculus. This gradient is then used to update the weights in the network. Firstly, the algorithm computes the gradient of the loss with respect to the output $\mathbf{a}^{(L)}$. Equation (2.6) presents how the algorithm calculates the gradient loss.

$$\delta^{(L)} = \nabla_{\mathbf{a}^{(L)}} \mathcal{L} \odot \sigma'(\mathbf{z}^{(L)}) \quad (2.6)$$

From Equation (2.6), the $\odot$ denotes the element-wise multiplication and σ' is the derivative of the activation function. Equation (2.7) shows how the algorithm calculates the gradient loss for each layer l from $L - 1$ to 1.

$$\delta^{(l)} = (\mathbf{W}^{(l+1)})^T \delta^{(l+1)} \odot \sigma'(\mathbf{z}^{(l)}) \quad (2.7)$$

From Equation (2.7), the $\mathbf{W}^{(l+1)}$ are the respective layer weights. The last step is update the weights and bias. Equation (2.8) shows how the algorithm update the layers weights and bias.

$$\begin{aligned} \mathbf{W}^{(l)} &:= \mathbf{W}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \mathbf{W}^{(l)} - \eta \delta^{(l)} (\mathbf{a}^{(l-1)})^T \\ \mathbf{b}^{(l)} &:= \mathbf{b}^{(l)} - \eta \delta^{(l)} \end{aligned} \quad (2.8)$$

From Equation (2.8), the η is the learning rate hyperparameter. The learning rate determines the step size in each iteration while moving towards a minimum of the loss function. Backpropagation allows the neural network to update its weights in a way that minimizes the error in its predictions. This process is repeated iteratively for many epochs until the network's performance on the training data is satisfactory (Coppin 2004, Goodfellow et al. 2016, Bishop 2006).

2.2.3 Stochastic Gradient Descent

The Stochastic Gradient Descent (SGD) is an optimization algorithm used to minimize an objective function, often the loss function in machine learning models. It is particularly useful in training large-scale and high-dimensional models (Coppin 2004, Goodfellow et al. 2016). The algorithm lays on the basic concepts of objective function, and gradient descent. In machine learning, the objective function $\mathcal{L}(\theta)$ typically represents the loss, where θ denotes the model parameters. The gradient descent goal is to minimize $\mathcal{L}(\theta)$ by updating θ iteratively in the opposite direction of the gradient $\nabla \mathcal{L}(\theta)$. In full-batch gradient descent, the gradient is computed using the entire training dataset, $\theta := \theta - \eta \nabla \mathcal{L}(\theta)$, where η is the learning rate. Unlike full-batch gradient descent, which uses the entire dataset, SGD updates the model parameters using only a single training example (or a small mini-batch) at each iteration. This makes SGD faster and more suitable for large datasets (Coppin 2004, Goodfellow et al. 2016). Equation (2.9) shows how the SGD algorithm optimize the parameters to a given objective function.

$$\theta := \theta - \eta \nabla \mathcal{L}(\theta; x_i, y_i) \quad (2.9)$$

From Equation (2.9), we can assume a θ_0 that represents the initial parameters of the model. The algorithm randomly shuffle the training data, and for each training example (x_i, y_i) , update the parameters. Here, $\mathcal{L}(\theta; x_i, y_i)$ is the loss computed for the training example. The algorithm repeats this iteration over the dataset multiple times (epochs) until convergence. Equation (2.10) shows the SGD algorithm update rule over a given training dataset $\{(x_i, y_i)\}_{i=1}^n$, for each (x_i, y_i) .

$$\theta_{t+1} := \theta_t - \eta \nabla \mathcal{L}(\theta_t; x_i, y_i) \quad (2.10)$$

From Equation (2.10), θ_t are the parameters at iteration t , η is the learning rate hyperparameter, and $\nabla \mathcal{L}(\theta_t; x_i, y_i)$ is the gradient of the loss with respect to the parameters, computed using the training example (x_i, y_i) . The SGD algorithm updates are faster than full-batch gradient descent, especially for large datasets (Goodfellow et al. 2016). Also, the algorithm is suitable for online learning where data arrives in a stream. Lastly, it helps in escaping local minima, potentially finding better solutions.

2.3 Autoencoders

The concept of the autoencoder was first proposed in Yann LeCun's doctoral thesis (Lecun 1987). An autoencoder is an artificial neural network trained to replicate its input at its output (Goodfellow et al. 2016). This type of structure can be divided into two parts: the encoder and the decoder. These can be viewed as two functions $Z = h(X)$ and $\hat{X} = g(Z)$. Figure 2.7 presents a high-level representation of an autoencoder.

Figure 2.7 shows has the first function maps the input data x to the latent space Z , while the second function reconstructs the data, mapping Z from the latent space back to the input data space x (Zhai et al. 2018, Goodfellow et al. 2016). Autoencoders have generalized the idea of an encoder and a decoder beyond the deterministic functions previously described, to mappings in stochastic functions $p_{\text{encoder}}(Z | X)$ and $p_{\text{decoder}}(\hat{X} | Z)$. Here, $\hat{X}$ is the reconstruction of the input signal X . In real-world applications, it is not entirely desirable for the autoencoder to merely learn how to copy the input X . Therefore, constraints are applied so that learn to copy the inputs approximately (Zhai et al. 2018). Additionally, these structures are traditionally used for input dimensionality reduction and feature extraction/feature learning (Goodfellow et al. 2016).

Equation (2.11) shows how a autoencoder can be modeled when applied to a given training set $S = \{x_i | x_i \in \mathbb{R}^d\}$, where $1 < i < n$.

$$AE = \begin{cases} Z = h(w_e, b_e; X) \\ \hat{X} = g(w_d, b_d; Z) \end{cases} \quad (2.11)$$

From equation (2.11), $h(\cdot)$ is the encoder and $g(\cdot)$ is the decoder, which are typically artificial neural networks. The variables w_e and b_e are the parameters of the encoder, and w_d and b_d are the parameters of the decoder. For neural networks, these parameters

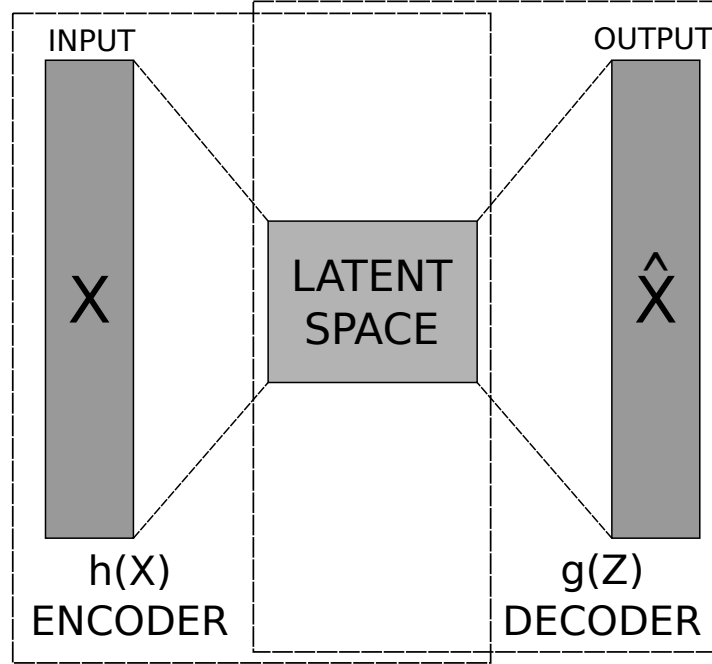


Figure 2.7: High-level representation of an autoencoder.

are the sets of weights and biases (Equation (2.1) and Equation (2.2)) of the encoder and decoder networks, respectively.

Training an autoencoder essentially involves optimizing (minimizing) the error of a loss function. Equation (2.12) shows the optimization performed in a autoencoder training.

$$J(\theta) = \frac{1}{N} \sum_{i=1}^n (X_i - \hat{X}_i)^2 \quad (2.12)$$

From Equation (2.12), J represents the chosen weights optimization algorithm. The letter $\theta = (w_e, b_e; w_d, b_d)$ is the set of the autoencoder weights and bias. In this case, the right equation part represents a chosen loss function, that calculate errors in the aproxi-mation of $\hat{X}_i$ to X_i . It is calculated as the average of the sum of the squared differences between each real value and its corresponding predicted value. The term $(X_i - \hat{X}_i)^2$ mea-sures the squared difference between the values. To solve the optimization problem in Equation (2.12), gradient descent or stochastic gradient descent algorithms are typically used.

Throughout time, several types of autoencoders have been proposed by different re-searchers. The Sparse Autoencoders (SAE) impose sparse regularization constraints on the latent variable Z to encourage sparsity using methods like Kullback-Leibler Diver-gence and the L_1 norm (Goodfellow et al. 2016, Zhai et al. 2018). Denoising Autoen-coders (DAE) are designed to reconstruct clean data from noisy inputs by training on cor-rupted data to predict the original input (Lecun 1987, Gallinari et al. 1987, Goodfellow et al. 2016). Convolutional Autoencoders (CAE) extend traditional AEs by using convolu-

tional neural networks (CNNs) for encoding and decoding, leveraging convolutional layers and pooling layers to process data, typically in image domains (Masci et al. 2011, Zhai et al. 2018). Contractive Autoencoders (CAE) add explicit regularization to the encoder’s output to minimize the sensitivity of the learned features to small input changes, using the Frobenius norm to penalize large derivatives (Rifai, Vincent, Muller, Glorot & Bengio 2011, Rifai, Mesnil, Vincent, Muller, Bengio, Dauphin & Glorot 2011, Goodfellow et al. 2016, Zhai et al. 2018). Lastly, Variational Autoencoders (VAE) incorporate variational inference to learn a probabilistic mapping of the data, optimizing the variational lower bound to balance reconstruction accuracy and the regularization of the latent space distribution (Kingma & Welling 2013, Rezende et al. 2014, Zhai et al. 2018, Goodfellow et al. 2016). Each variant enhances the basic AE framework to address specific challenges like noise reduction, feature sparsity, data dimensionality, and probabilistic representation, among others.

Chapter 3

Related Works

In the wake of the COVID-19 pandemic, accurate and reliable forecasting of the virus's spread has become paramount for public health planning and intervention. The complexity and unprecedented nature of the pandemic have spurred a vast array of research efforts aimed at predicting COVID-19 time series data. This chapter reviews the substantial body of work in this domain, categorizing it into three primary sub-groups: conventional machine learning models, artificial neural networks (ANNs), and autoencoder-based models. Each of these approaches offers unique strengths and insights, contributing to the evolving landscape of COVID-19 forecasting methodologies.

Conventional machine learning models have been extensively employed to predict COVID-19 time series, leveraging statistical techniques and well-established algorithms to discern patterns and trends in the pandemic's progression. In this direction, short-term forecasting has been provided, for example, using methods derived from smoothing model families. Petropoulos et al. (2022) presents a model designed to predict two key variables related to COVID-19: the cumulative number of confirmed cases and deaths. The data for this model was sourced from the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University, covering global and country-specific statistics since January 22, 2020. The model's performance was evaluated using different planning horizons and prediction intervals at 50%, 70%, and 90% confidence levels. The study also compared the model's accuracy to other publicly available forecasts and provided specific forecasts for Denmark, Norway, and Sweden, which implemented varying levels of interventions. The authors extended the study by Petropoulos & Makridakis (2020) to more rounds of forecasts and variables, analyzing the forecast accuracy and the performance of the prediction intervals. The focus is on the model's ability to provide accurate short-term forecasts and its value to policymakers and decision-makers.

The work proposed by Roccetti (2024) presents a *forecast by analogy* based on previously observed behavior peaks and declines of pandemic cases in Italy. The authors looked up two winter periods of 2022/2023 and 2023/2024 from COVID-19 infection. They extracted values of the weekly deaths and then computed the percentage of the change for each consecutive pair of values of the growth/decline rates of the series, comparing the most recent value in the series with the previous one. In conclusion, the authors claim the proposal is adequate and innovative, having a low percentage range of error.

Safi & Sanusi (2021) article proposes a hybrid model for COVID-19 forecasting, utilizing ARIMA, ETS, and ANN models. The neural network used is a feed-forward neural

network with a single hidden layer and lagged inputs. The data consists of total confirmed cases and recovery or death rates. The training dataset formulation involves fitting multiple individual model specifications, and the best results are obtained with ARIMA and ETS models. The input window size and output forecasting window are not explicitly mentioned in the article. In summary, the study presents a hybrid model combining ARIMA, ETS, and ANN for COVID-19 forecasting, with a focus on total confirmed cases and mortality rates, showcasing the effectiveness of ARIMA and ETS models in predicting pandemic trends.

The article of Aslam & Biswas (2023) discusses the use of various machine learning models to predict COVID-19 deaths, including linear regression, support vector machines, LASSO, and exponential smoothing. The data set used for the study included information from the US, India, Italy, and three continents: Asia, Europe, and North America. Due to missing values, some variables could not be included in the analysis. The study employed machine learning classifiers like decision trees, random forests, SVM, XGBoost, and others to understand the impact of temperature and humidity on COVID-19 spread. The random forest algorithm was found to be the most accurate. The results aim to assist the government in policy formulation and infrastructure management to reduce COVID-19 deaths.

The Güngör (2023) article compares Naive, Seasonal Naive, Drift, Linear Regression, Exponential Smoothing, ARIMA, and a Neural Network Autoregression (NNAR) time series forecasting methods for predicting COVID-19 cases. The NNAR model used is a feed-forward neural network with a single hidden layer. The data includes time series from ten of the most affected countries, with daily and weekly periods analyzed. Training data sizes for daily forecasting are 40, 180, and the largest possible size, which means, the complete data set sampled in days. While for weekly forecasting, the data sizes were 16, 52, and the largest possible size, which means the complete data set sampled in weeks. Test data sizes for daily forecasting are 1, 5, and 10 (days), and for weekly forecasting, they are 1, 2, and 4 (weeks). The study applies square root and logarithmic transformations for all methods except naive and seasonal naive. The performance of the models is evaluated using the mean absolute percentage error (MAPE) among all data combinations. The study finds that the reliability of time series forecasting for COVID-19 is highly sensitive to parameter selection and often deteriorates as the forecasting horizon extends. The study concludes that policymakers should exercise caution when using time series forecasts for decision-making during pandemics.

Kumar & Susan (2020) proposed another work in this direction using the ARIMA and Prophet models. The authors aimed forecasting the COVID-19 time-series spread for cumulative cases on the world total cases and on the ten most affected countries, showing the more effectiveness of the ARIMA model. From the above works, we can notice that the ARIMA model is frequently used to forecast time series due to its simplicity and composition, which was originally created to deal with this kind of problem.

Regarding another category based on machine learning models, we can find works that use artificial neural networks to solve the time series forecasting related to the COVID-19 pandemic. Artificial neural networks, with their ability to model complex and non-linear relationships, have shown significant promise in capturing the intricate dynamics of

COVID-19 time series data. Chimmula & Zhang (2020) proposes a forecasting model for the COVID-19 outbreak in Canada using Long Short-Term Memory (LSTM) networks. The time series dataset includes the number of confirmed cases, fatalities, and recovered patients until March 31, 2020. The data was split into 80-20% training and testing sets. The evaluation metrics used to assess the model's performance round about mean squared error (MSE), root mean squared error (RMSE) and accuracy of forecasting. The authors presented the 2, 4, 6, 8, 10, 12 and 14th day predictions for 2 successive days. The text does not provide specific details about the number of samples observed to make the predictions. The conclusions of the study indicate that the prompt and effective approaches taken by Canadian public health authorities have had a positive impact on minimizing human exposure to COVID-19, showing a linear trend in transmission rates compared to the exponential growth seen in the USA.

The article proposed by Tomar & Gupta (2020) presents a Long Short-Term Memory (LSTM) based technique for predict the possible number of confirmed and recovered cases of COVID-19 in India for the next 30 days. The data used spans from January 30, 2020, to April 4, 2020, with 80% allocated for training and 20% for testing and validation. A percentage error was calculated over the official data and the model prediction estimation. The training hyperparameters, evaluated metrics, and specific input window/look-back size are not explicitly detailed in the provided context. The results indicate that the model can approximately predict the number of new COVID-19 cases, aiding administrative preparations.

The study of Aragão et al. (2022) investigates the impact of environmental factors such as temperature, humidity, and air quality on predicting COVID-19 pandemic behavior using a data-driven approach. The researchers utilized a Long Short-Term Memory (LSTM) cells to model COVID-19 data from Brazil, including daily deaths, temperature, humidity, and air quality indices. The data spans from March 27, 2020, to June 3, 2021, covering 435 days or 62 weeks. The training dataset comprises 90% of the data (the first 56 weeks), and the remaining 10% (the last 6 weeks) forms the test dataset. To address trends and seasonality, the biweekly moving average for COVID-19 deaths and cases time series is used as input. The model employs a sliding window approach, with a window size of 1 week. The training configuration includes a grid search on hyperparameters such as the number of units per layer, batch size, and input normalization, followed by extensive training with 500 epochs and 50 trials. The root mean square error (RMSE) is the primary evaluation metric, with the best model showing strong forecasting capabilities for COVID-19 deaths in São Paulo, achieving minimal deviation from actual accumulated values. The input data window size and output forecasting period were key aspects of the model configuration. The multivariate approach tested different combinations of features, showing improved predictive skills as more features were added. In conclusion, the study highlights the significant influence of environmental factors on COVID-19 dynamics and suggests the need for incorporating these variables in predictive models to improve accuracy and reliability.

The article proposed by Car et al. (2020) presents a study on modeling the spread of COVID-19 infection using a Multilayer Perceptron (MLP) artificial neural network. The research utilizes a dataset containing geographical and time data of infected patients,

which is transformed into inputs and outputs for training the MLP. The dataset is split into five equal parts for training and testing, with a repeated 80%/20% distribution. The MLP models are trained using hyperparameters such as learning rates and regularization parameters, achieving high accuracy in predicting confirmed cases, recoveries, and deceased patients. The evaluated metrics in the study included the coefficient of determination (R^2) for confirmed cases, deceased patients, and recovered patients. The study emphasizes the importance of AI algorithms in capturing intricate data patterns for accurate regression modeling of infectious diseases. The authors' final conclusion highlighted the successful achievement of their aim to generate a model of corona virus disease spread on a global level using machine learning methods.

Borghi et al. (2021) presents a COVID-19 time series forecasting model based on a multilayer perceptron (MLP) artificial neural network (ANN). The proposed model employs a feedforward neural network with up to two hidden layers, each containing a maximum of ten neurons. The neurons in the hidden layers use a tangent sigmoid activation function, while the output layer neurons use a linear activation function. The Levenberg-Marquardt backpropagation method is utilized for training the network. The data for training and testing is derived from the top 30 countries with the highest number of confirmed COVID-19 cases as of May 11, 2020. The time series data includes the number of confirmed cases, new cases, accumulated deaths, and new deaths, all smoothed using a moving average filter with a window size of three days to reduce sudden fluctuations. The train, validation, and test datasets are divided in a 70%, 15%, and 15% ratio, respectively. The input window size is 20 days, while the forecasting window is six days. The performance of the model is evaluated using Mean Square Error (MSE) and Mean Absolute Error (MAE) metrics. The best-performing model configurations, identified as M1 and M2, include different numbers of neurons in the hidden layers, with M1 chosen as the optimal model based on its error measures. The study demonstrates the model's ability to generalize and predict COVID-19 trends effectively, despite the challenges of forecasting over longer periods.

A novel approach to forecasting COVID-19 cases using a modified multilayer artificial neural network (MMLNN) was proposed by Majhi (2023). The proposed model employs an activation function before summation tasks, which is unique compared to other models used for COVID-19 prediction. The data used for training and testing the ANN models are normalized and statistical features are extracted using a sliding window. The chosen window size was seven, which yields the best prediction performance. For training, 80% of the input patterns (66 out of 84) are randomly selected, while the remaining 20% (18 patterns) are used for testing. The last 18 patterns are specifically used for testing due to their exponential trend, which better evaluates the model's performance. The training hyperparameters include connection weights and bias weights arbitrarily placed in the range of $[-0.5, +0.5]$. The model's performance is evaluated based on its prediction accuracy, with the simulation study demonstrating robust and consistent results.

Sharma et al. (2023) proposes a model called AGLSTM that utilizes Convolutional Neural Networks (CNN) to process input training data and Long Short-Term Memory (LSTM) networks to capture long-term dependencies in time series data. The training dataset is used to optimize parameters for each task, while the test dataset optimizes pa-

rameters for the entire model. The model's performance is evaluated using metrics such as precision, mean squared error, and mean absolute error. The study focuses on data fusion and transfer learning to enhance prediction accuracy for COVID-19 outcomes. The model uses an adaptive gradient LSTM approach and achieves accurate predictions by integrating multivariate time series data. The input window size and output forecasting window are crucial parameters in the model's design, contributing to its successful prediction capabilities.

The work proposed by Mousavi et al. (2020) presents a methodology for predicting the number of confirmed COVID-19 cases using a Recurrent Neural Network (RNN) model. The data used includes four time series: temperature, humidity, the number of confirmed cases, and transmission rates, which reflect the effectiveness of control measures during different lockdown phases. The training and test datasets are formulated by systematically selecting features and labels, with the training process incorporating variational mode decomposition (VMD) to handle non-stationary signals. The best results are achieved by combining stationary and non-stationary features through VMD, enhancing the model's predictive accuracy. This approach demonstrates the potential of integrating environmental and control measure data to improve short-term COVID-19 case forecasting.

Mukhtar et al. (2022) discusses the use of Long Short-Term Memory (LSTM) neural networks for time series forecasting, highlighting their application in various domains such as predicting tourism flow, financial market assets, and disease growth. The data pre-processing involves normalizing numerical data to a "Min-Max" scale between 0 and 1, which is essential for the LSTM model to process the input values effectively. The study compares LSTM with traditional algorithms like ARIMA, finding that LSTM significantly reduces the error rate by 84-87%. The training and testing datasets are formulated by splitting the time series data, and the model's performance is evaluated using metrics such as Mean Absolute Percentage Error (MAPE). The best results are achieved with the AT-LSTM model, which outperforms standard LSTM due to its lower MAPE value. The input window size and output forecasting window are determined based on the specific application and data characteristics, although exact values are not provided in the context.

Haviluddin & Alfred (2023) proposes the use of various CNNs and their smoothed variants (Smoothing-CNN, Moving Average-CNN, Smoothed Moving Average-CNN, Moving Average Smoothed-CNN) for time series forecasting. The dataset was normalized with "Min-Max" scale. The experimental design includes five scenarios, with data processed directly using CNN in one of them. The window size of input information and the output forecasting window are part of the experimental setup, but specific details are not provided in the retrieved context. The authors used MAPE and MSE to measure the four models forecasting accuracy. In summary, the article evaluates the effectiveness of different CNN models and their smoothed variants for COVID-19 time series forecasting, using a structured experimental design with multiple datasets. The study includes detailed descriptions of data normalization, forecasting processes, and key performance indicators, with results and discussions highlighting the best-performing models. The research concludes with a summary of findings and potential areas for further investigation.

The work proposed by Mishra et al. (2024) presents the use of Long Short-Term Memory (LSTM) networks, for short-term forecasting in the context of COVID-19 dissemina-

tion. The authors claim chosen LSTM networks for their ability to handle long-term dependencies and maintain intrinsic memory cell status. The models used for comparison were VAR (Vector Autoregression), and SEIR. The data used includes new incidences and fatalities, and the model employs maximum likelihood estimation to estimate coefficients. The training and testing datasets are split 75%/25%, formulated from time-series data. They used the root mean square error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE) as evaluable metrics. After applying PCA, the data pre-processing steps included normalization and smoothing to transform the data. Additionally, the time series data was shifted to adopt appropriate lag values, making the dataset suitable for forecasting through supervised learning algorithms. The lag length used was thirty instances, to predicted infected incidences and fatalities for the next 24 hours. The best metric values were achieved by the proposed LSTM model, with the lowest median RMSE, and MAPE values. In summary, the article presents a model leveraging LSTM networks for short-term forecasting of COVID-19 spread, utilizing time-series data of incidences and fatalities.

Alifiah et al. (2023) discusses the development of an artificial neural network (ANN) model using K-fold validation, with one input layer, one output layer, and the ReLU activation function. The data is transformed using the "Standard" scaler to ensure the average observed value is 0 and the standard deviation is 1. In k-fold cross-validation, the dataset is divided into 'k' subsets of roughly the same size. For each of the 10 folds, 90% of the data is used for training and 10% is used for testing. This process is repeated 10 times, each time with a different subset as the test data. The model is trained using hyperparameters such as batch sizes of 16 and 32, epochs of 50 and 100, dropouts of 0.2 and 0.3, and learning rates of 0.001 and 0.01. The hyperameters-tuning process includes early stopping to optimize the model's performance. The input size for the model was 45 units, which corresponds to 45 days, to forecasting 7 days. The output layer of the model has 1 unit that indicates the model predicts one value at a time, likely the value for the next day. For forecasting multiple days, this process was repeated iteratively to predict each subsequent day. The metrics used to evaluate the model include accuracy, root mean square error (RMSE), and mean absolute percentage error (MAPE). The best results achieved an accuracy of 98.82% for confirmed cases, 98.52% for recovered cases, and 98.89% for death cases.

The article proposed by Chwila (2023) discusses the application of Long-Short Term Memory (LSTM) recurrent neural networks to forecast the dynamics of the Covid-19 pandemic. The authors mentioned the LSTM networks are chosen due to their ability to handle sequential data. The data used includes various pandemic-related metrics, and the concept of time series cross-validation is employed to formulate the train and test datasets. The data used for training spans from 11th April 2020 to 3rd March 2022, while the data used for testing is from 4th March 2022 to 10th March 2022 for the whole Poland. Others data formulation were used the '5'-fold time series cross-validation, with a variable train length of 117, 232, 347, 462, and 577, and a fixed test size of 114 samples. The compared machine learning models are Least Absolute Shrinkage and Selection Operator (LASSO) regression, Gradient Boosted (GB) regression trees, Support Vector Regression (SVR), and the proposed Long-Short Term Memory (LSTM). The study aims to predict

the daily number of new infections in Poland based on the data from the two previous days. Hyperparameters such as the number of iterations and learning rates are defined, with a specific mention of 500 iterations. The evaluation metrics include accuracy and the Root Mean Squared Error (RMSE), with the best results achieved by the SVR trained with a single split of data.

Rauf et al. (2023) proposed the application of deep learning models, specifically Long Short Term Memory networks (LSTM), Recurrent Neural Networks (RNN), and Gated Recurrent Units (GRU), for forecasting the number of COVID-19 cases. The dataset utilized for training and testing these deep learning models encompasses COVID-19 case data from January 22, 2020, to June 21, 2020, with the last 10 days' samples reserved for model prediction and evaluation. The input layer employs the ReLU activation function, while the Adam optimizer is applied in the final layer. The training process includes 100 epochs with a batch size of 1, incorporating early-stopping criteria. The evaluation metric used is accuracy, with an overall metric of 90% achieved by the models. The authors concluded that the LSTM model outperformed the other models in predicting COVID-19 case numbers. Additionally, they observed that the LSTM model provided more accurate predictions for the countries studied, including India, Pakistan, Afghanistan, and Bangladesh. The authors emphasized the significance of utilizing such models for improved preparation and response to future pandemics.

The study proposed by Shastri et al. (2020) also explores the use of three variants of LSTM-based Recurrent Neural Networks (RNN) for time series forecasting of COVID-19 datasets: Stacked LSTM, Bi-directional LSTM, and ConvLSTM. These models are trained to predict one-month future values of COVID-19 cases and deaths based on historical data from the USA and India. The dataset for confirmed cases spans from February 7, 2020, to July 7, 2020, for both countries. For deaths, the data spans from March 12, 2020, to July 7, 2020, for India, and from February 6, 2020, to July 7, 2020, for the USA. An 80-20 split was used for training and testing. The models were trained and tested rigorously, with hyperparameters carefully selected to optimize performance, including input and output window sizes and learning rates, among others. The evaluation metric employed was the Mean Absolute Percentage Error (MAPE), with the best model selected based on the lowest error and highest accuracy. The Convolutional LSTM (ConvLSTM) model outperformed the other two models, achieving an error rate ranging from 2.0% to 3.3% across all datasets.

Sperandio Nascimento et al. (2023) explores the application of deep neural networks, specifically Convolutional Neural Networks (CNN) and Convolutional Long Short-Term Memory (ConvLSTM) models, for predicting COVID-19 new cases and deaths worldwide. The proposed model leverages wavelet transform for data preprocessing, significantly enhancing prediction accuracy. The dataset comprises daily new cases and deaths from various countries, divided into training and testing sets. The training set spans data from December 2019 to December 2020, while the test set covers January 2021 to March 2022. The model architecture features multiple layers, including 1D convolutional layers, max-pooling layers, LSTM layers, and dense layers, utilizing activation functions such as TanH and ReLU. The training process incorporates hyperparameters like dropout rates and the number of neurons in each layer. The evaluation metrics include Mean Abso-

lute Error (MAE), Normalized Mean Squared Error (NMSE), Pearson R, and Factor of 2. The input window size and the output forecasting window are designed to predict the number of new daily cases and deaths up to 30 days ahead. The experiment determined that the optimal input window look-back for the best wavelet transform is of length 4, encompassing the current day plus three previous samples. This comprehensive approach underscores the potential of integrating wavelet transforms with deep learning models for effective time-series forecasting in the context of the COVID-19 pandemic.

Regarding the direct application to COVID-19 using Auto-Encoders, we can also find related works to solving the time series pandemic forecasting problem. Autoencoder-based models, known for their prowess in unsupervised learning and feature extraction, have been utilized to enhance the accuracy of COVID-19 time series forecasting by uncovering hidden structures within the data. Ibrahim et al. (2021) proposes a combination of a Variation Auto-Encoder (VAE) with LSTM networks. The proposed model is divided into two branches. The first branch is a self-attention autoencoder LSTM, which is fed with daily virus propagation data by country, government policies adopted per day and country, as well as urban characteristics per day and country. This first branch can also be divided into the self-attention LSTM sequence encoder and the LSTM sequence decoder. According to the authors' references, the self-attention mechanism enables the LSTM to understand the representation of its inputs by relating the positioning of each sequence. The second branch of the model operates in parallel with the self-attention LSTM mechanism. It consists of the VAE, which is fed with a spatial matrix of dimensions and repeated for the entire duration of the training with timestamps referring to date and time. The outputs of the two branches of the model are concatenated in the feature dimension and sent to the LSTM sequence decoder, which in turn returns the prediction values. This study aims to predict the spread (accumulated number of cases) of COVID-19 worldwide and for each country separately. Two models were trained, with the first predicting a value for three time samples, while the second was trained to predict ten days. The data used for training the models were collected from various datasets for the world and each country separately. Some of the countries tested included the USA, Italy, Spain, among others. The evaluation metrics used were RMSE and MSLE.

Zeroual et al. (2020) provide a comparative study with five different models of neural networks, including the RNN, LSTM, Bi-LSTM, GRU, and the Variational Auto-encoder. According to the authors, the last one achieved better results than the other models for predicting the number of new COVID-19 cases and of recovered cases. They propose to mix a Variational Auto-Encoder with LSTM networks, dividing the model into two branches. The first is a Self-Attention Auto-encoder LSTM, which receives data by daily and country virus propagation, government policies adopted, and urban characteristics. They also divide this first component into a self-attention LSTM Sequence Encoder and an LSTM Sequence Decoder. According to the authors, the self-attention mechanism makes the LSTM capable of understanding the representation of its inputs relating to the positioning of each sequence. The second branch of the model is a Variational Auto-Encoder (VAE) working in parallel with the self-attention LSTM mechanism. The VAE receives as data a dimensional, spatial matrix that has been repeated throughout the training duration with timestamps referring to the date and time. The outputs of the two model branches are con-

catenated in the feature dimension and sent to the LSTM sequence decoder, which returns the prediction values. This study attempts to predict the spread (accumulated number of cases) of COVID-19 worldwide and for each country separately. They trained two models, where the first forecasts three-time samples and the second predicts ten. The data used for training the models were collected from several world datasets and also for each country separately. Some countries tested were the USA, Italy, and Spain, among others. The RMSE and MSLE are used as evaluation metrics.

The Hawas (2020) paper presents a time-series prediction model for COVID-19 confirmed case for next 30 and 40 days in Brazil using recurrent neural networks (RNNs). The proposed model utilizes GRU (Gated Recurrent Unit) and incorporates a convolutional layer within a composite autoencoder neural network. The dataset consists of daily COVID-19 infection counts, and the data is divided into training and testing sets with a 70-30% split. The models were trained using the Adam optimizer with a learning rate of 0.001, and hyperparameters include 15 time-steps, 10 epochs, a batch size of 1024, and a validation split of 30%. Evaluative metrics for the models were based on three R-squared values: model accuracy over time-steps, over the whole evaluation period, and the sum of both. The best-performing model achieved an R-squared value of 0.5875 over the whole duration. The input window size is 15 time-steps, and the output forecasting size is 15 time-steps. The study's results demonstrate the effectiveness of using GRU with convolutional layers for time-series forecasting, showing substantial improvement in prediction accuracy and performance.

In summary, the forecasting of COVID-19 time series has seen a diverse array of methodological advancements, ranging from conventional machine learning models to the sophisticated realms of artificial neural networks and autoencoder-based approaches. Each of these methods brings its own set of advantages and challenges, contributing valuable insights to the predictive modeling landscape. As the pandemic continued to evolve, the integration and continuous refinement of these techniques were crucial in enhancing our understanding and response to future public health crises. The comparative analysis of these models underscores the importance of leveraging a multifaceted approach to improve the robustness and accuracy of COVID-19 forecasts, ultimately aiding in more effective decision-making and resource allocation.

Thus, our approach builds on existing advancements by proposing a multi-feature stacked autoencoder-based methodology for time series forecasting of COVID-19 cases and deaths. This approach leverages the strengths of autoencoders in capturing complex, non-linear representations of input data while integrating various features, such as epidemiological and environmental factors, to enhance predictive accuracy. While our methodology shares similarities with traditional autoencoder-based models, it distinguishes itself through its adaptability to smaller datasets and a careful balance between feature extraction and predictive performance. Challenges related to hyperparameter management, computational demands, and noise introduced during latent space generation were addressed through strategic fine-tuning and optimized configurations. Beyond COVID-19 forecasting, this architecture demonstrates potential for broader applications, such as image classification, embedding generation, robotics tasks, etc. By providing accurate and actionable predictions, the proposed model not only contributes to public

health decision-making but also establishes a foundation for its use in diverse domains.

Chapter 4

Problem and Proposal

In this chapter, the problem and the proposed solution will be presented in more details. Mathematical formulations regarding the time series prediction problem will be shown, as well as how this problem applies to the context of COVID-19. Finally, the proposed methodology and its functioning will be presented.

4.1 Time Series Prediction

In general, models for time series modeling can be univariate, where a single variable has temporal relationships, or multivariate, when there is more than one time-dependent variable. These models can have different architectures and may vary significantly when modeling a univariate or a multivariate system. However, most models that use artificial neural networks can handle both univariate and multivariate cases without distinction (Torres et al. 2021). Equation (4.1) simply defines how a univariate time series can be modeled.

$$y = y(t-L), \dots, y(t-1), y(t), y(t+1), y(t+2), \dots, y(t+h) \quad (4.1)$$

From Equation (4.1), we have L as the number of historical values in the series, where each $y(t-i)$ for $i = \{0, 1, \dots, L\}$ represents the value obtained from variable y at each time instant $t-i$. The prediction process is the estimation of the value for $y(t+1)$, where this estimate can be denoted as $\hat{y}(t+1)$. This process attempts to minimize the error between the actual $y(t+1)$ and the predicted $\hat{y}(t+1)$. This error is typically represented by $y(t+1) - \hat{y}(t+1)$. A prediction can also be made when the desired number of predictions h is greater than just one. That is, when the goal is to predict the next h values after $y(t)$ (the current time instant), thus, $y(t+i)$ with $i = \{1, 2, \dots, h\}$. For this type of situation, the best prediction value is achieved when the sum of the predicted values minus the actual ones is minimized (Torres et al. 2021). Equation (4.2) shows how a prediction approaches the actual value.

$$error = \sum_{i=1}^h y(t+i) - \hat{y}(t+i) \quad (4.2)$$

From Equation (4.2), it is possible to observe that when the error between $y(t+i)$

and $\hat{y}(t+i)$ for each value of i is close to zero, the prediction was accurate. With the definition of prediction for a uni-variate time series, this same concept can be extended to multivariate time series. Analogously, Equation (4.3) defines how a multivariate time series can be modeled.

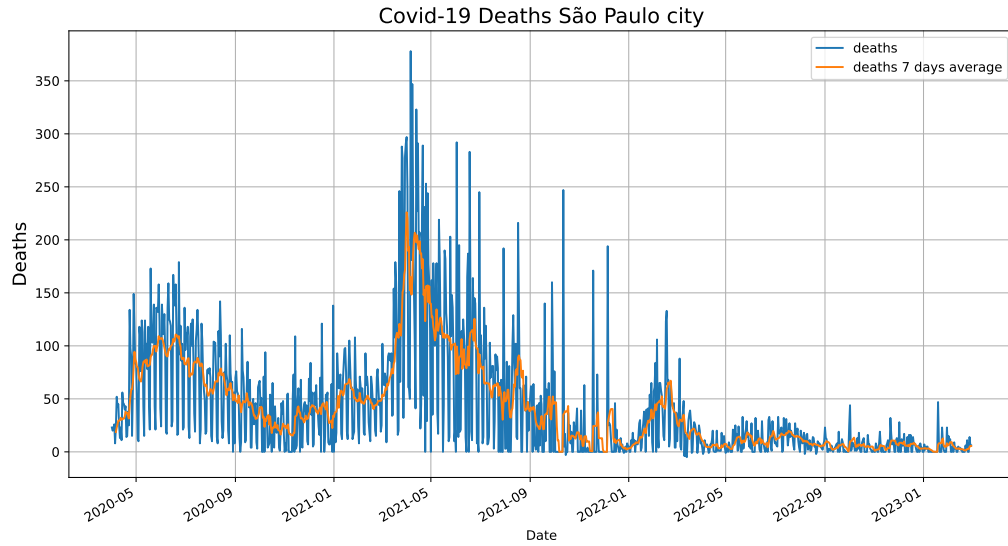
$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_T \end{pmatrix} = \begin{pmatrix} y_1(t-L) & \dots & y_1(t-1) & y_1(t) & y_1(t+1) & \dots & y_1(t+h) \\ y_2(t-L) & \dots & y_2(t-1) & y_2(t) & y_2(t+1) & \dots & y_2(t+h) \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ y_n(t-L) & \dots & y_n(t-1) & y_n(t) & y_n(t+1) & \dots & y_n(t+h) \end{pmatrix} \quad (4.3)$$

From Equation (4.3), we have $y_i(t-m)$, where $i = \{1, 2, \dots, n\}$ is the identifier of each time series within the set of time series. Additionally, $m = \{0, 1, \dots, L\}$ represents the historical data and the current value of the variables, and $m = \{-1, -2, \dots, -h\}$ represents the h future values. In multivariate prediction, there is the case where there is a target time series, the one to be predicted, which is not part of the set of series y_n , represented in this case by y_T on the left side of the equation. In this case, the remaining series are denoted as independent time series. Additionally, there are essentially two more cases in multivariate prediction: the N -to-1 and the N -to- N cases. The N -to-1 case is similar to the one already mentioned, but the target series to be predicted is part of the set of series y_n . In the N -to- N case, the targets for prediction are a number N being $n = \{0, 1, \dots, N\}$ of time series from the set y_n .

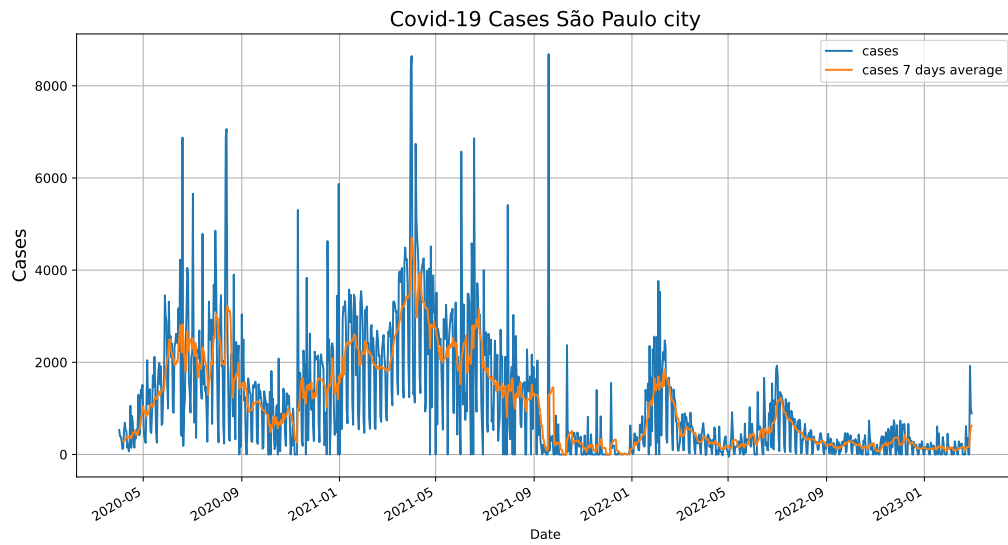
4.1.1 COVID-19 Time Series Prediction

As a real-world problem, the time series related to COVID-19 exhibit a behavior with a lot of variation and noise. Moreover, there are several additional factors that can influence the behavior of these series. Even with the end of the pandemic, these factors have not been fully understood by researchers. Some studies point to the influence of air quality and humidity as relevant factors in the behavior of the number of cases (and deaths) during the pandemic (Xu et al. 2020, Elsaid & Ahmed 2021). Similarly, social, economic factors, national holidays, and whether part of the population smokes have been evaluated to understand the spread of the virus (Aragão et al. 2021, Aabed & Lashin 2021, Mogi & Spijker 2021, COCCIA 2020). This work aims to predict the number of deaths and cases for COVID-19. Figure 4.1 shows the number of daily COVID-19 deaths and cases and their respective seven-day moving averages for São Paulo city in Brazil.

From Figures 4.1a and 4.1b, it is possible to observe the series in two representations. One representation shows the series for São Paulo city in real terms, that is, the number of deaths and cases that occurred on that day. Meanwhile, the other representation is a seven-day moving average for the series. Equation (4.4) shows how the moving average applied to a series can be obtained.



(a) Plot of the number of deaths and their moving average for the São Paulo city, Brazil, from 2020-04-01 to 2023-03-01.



(b) Plot of the number of cases and their moving average for São Paulo city, Brazil, from 2020-04-01 to 2023-03-01.

Figure 4.1: Plot of the target time series for prediction in this work.

$$m_i = \frac{1}{n} \sum_{j=1}^n d_{i+j} \quad (4.4)$$

From Equation (4.4), we have n as the number of samples over which the average is calculated, which, for the case of Figure 4.1, is $n = 7$. Due to its less noisy and more stable behavior (which is desirable), only series that have undergone a moving average will be used in this work.

As previously mentioned, the global COVID-19 pandemic problem can be influenced by various factors, characterizing it as a multivariate problem. In addition to the series representing the number of deaths and cases, the series representing air humidity, average ambient temperature, and air quality, known as AQI (Air Quality Index), were chosen. Thus, the problem can be formulated as shown in Equation (4.3). Considering this set of $S = \{s_1, s_2, s_3, s_4, s_5\}$ series that are directly or indirectly related to the variation in the count of COVID-19 cases and deaths, Equation (4.5) defines the problem explored in this work.

$$\begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{pmatrix} = \begin{pmatrix} s_1(t-L) & \dots & s_1(t-1) & s_1(t) & s_1(t+1) & \dots & s_1(t+h) \\ s_2(t-L) & \dots & s_2(t-1) & s_2(t) & s_2(t+1) & \dots & s_2(t+h) \\ s_3(t-L) & \dots & s_3(t-1) & s_3(t) & s_3(t+1) & \dots & s_3(t+h) \\ s_4(t-L) & \dots & s_4(t-1) & s_4(t) & s_4(t+1) & \dots & s_4(t+h) \\ s_5(t-L) & \dots & s_5(t-1) & s_5(t) & s_5(t+1) & \dots & s_5(t+h) \end{pmatrix} \quad (4.5)$$

Equation (4.5) shows the set S of selected series to achieve accurate predictions of the selected target series.

As the prediction model is a data-driven method, it must be able to return one or more time series forecasting using the set of time series defined as inputs. Figure 4.2 presents a high-level representation of how the model should function using the forecasting of two time series as example.

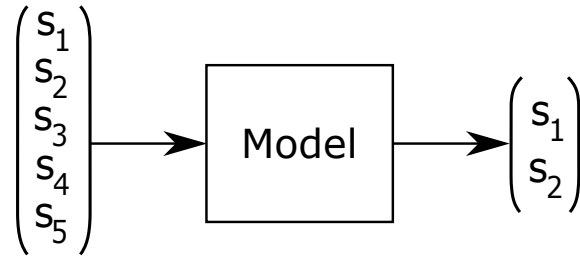


Figure 4.2: High-level representation of how the flow between input, a trained model, and prediction should function.

From Figure 4.2, it is possible to observe that the proposed model should receive a set of series $S = \{s_1, s_2, s_3, s_4, s_5\}$ relevant to the problem. Then, the model should be able to return the target variables for each desired prediction time step ahead, represented by s_1 and s_2 in the figure.

4.2 Expanded Latent Space Auto-encoder

We propose a multivariate input data-driven model applied for COVID-19 time-series forecasting task. The proposed model is a stacked auto-encoder that extends the baseline model initially presented by Pereira et al. (2020). Thus, to a better understanding of our model, it is primordial to firstly present the original proposed model.

4.2.1 Modified Auto-Encoder

Due to the short data size during the early COVID-19 outbreak, the authors proposed an auto-encoder based model for uni-variable forecasting daily cases, in their baseline approach (Pereira et al. 2020). Figure 4.3 shows a layer schematic representation of the originally proposed, named Modified Auto-Encoder.

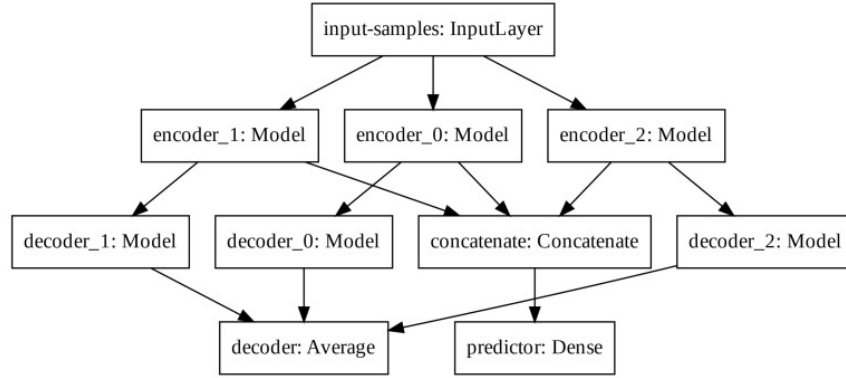


Figure 4.3: First proposed architecture named Modified Auto-Encoder model. Image used with author's authorization.

From Figure 4.3 we can see the model has one single input layer followed by three encoder layers that receive the same input data, where each encoder generates its own latent space. The three encoder latent spaces are concatenated before they are shown to the prediction layer. At the same time, each decoder generates its decoder output. The three decoded outputs have their average calculated to dimension match the input to error calculation. Finally, the prediction layer output has its values approximated to the desired value. This original propose has a three auto-encoders fixed architecture and a predictor tied to it. Our extended proposed aim to remove the internal auto-encoders quantity limitation, allowing it value be $n = 1, 2, \dots, N$. Figure 4.4 shows a representative diagram in flowchart form of how we present the inputs to the Expanded Latent Space Auto-encoder (ELSA) and how we use N internal auto-encoders to compose it.

Figure 4.4 shows the decoded mean vector has the same dimension as the input, and the expanded latent space vector has dimension $N \times \text{latent space size}$, where N is the numbers of internal auto-encoders. If the internal auto-encoders N value is greater than or equal to two, we denoted the concatenated encoders as the expanded latent space. Also,

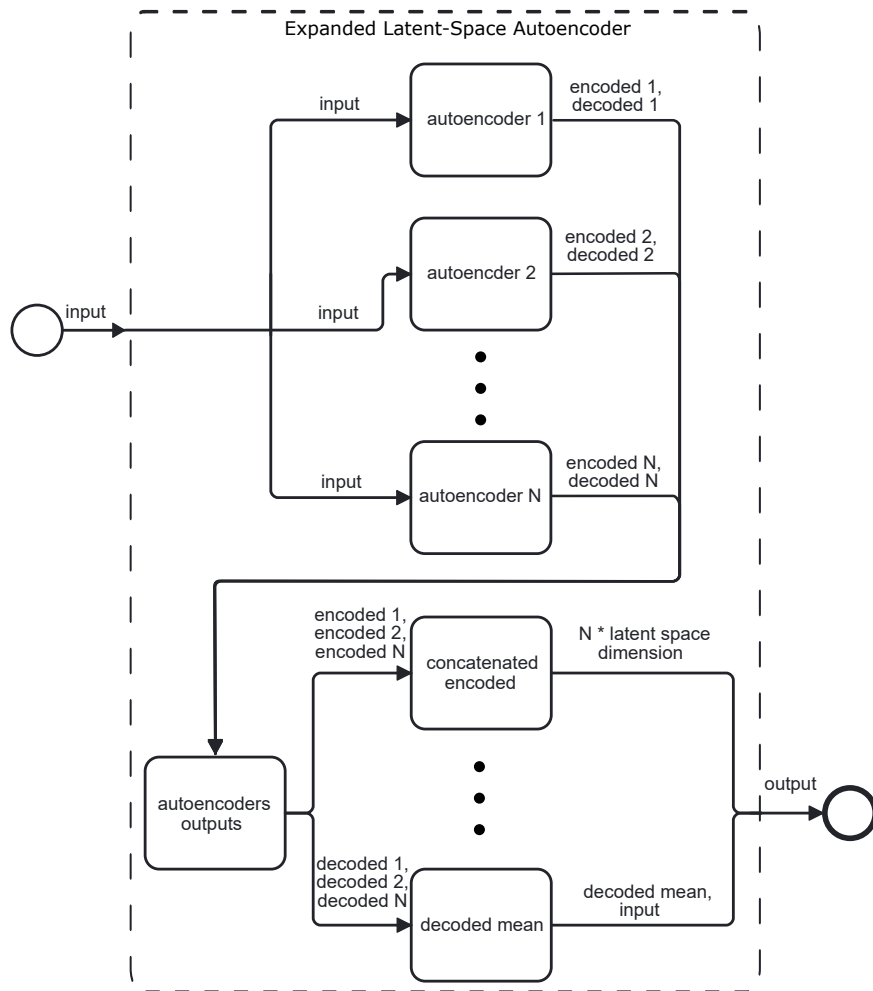


Figure 4.4: Representative figure of the sequence flowchart of entries into ELSA.

we do not tied any predictor to our architecture model at this step. The proposed network works only as a latent space generator, similar to a embedding model.

4.2.2 Multi-Variable ELSA

The reason we do not pre-attach any predictors to the ELSA network is because we want to use it to mimic an embedding model. Once we train an ELSA network for a specific series, the network can generate expanded latent space representations for each presented data. Then, all input data in the entire training dataset can pass through to their respective trained ELSA networks and generate an encoded dataset with the expanded latent spaces. With this final encoded dataset, it is possible to train a reduced number of neurons predictor networks. Consequently, the ELSA network can generate a dataset that makes training less costly and faster when speed is needed. Additionally, when a new series is added to the problem, it is not necessary to retrain all entire model, just a new ELSA network and the predictor network. Figure 4.5 shows the operational structure of a multi-variable model with ELSA networks.

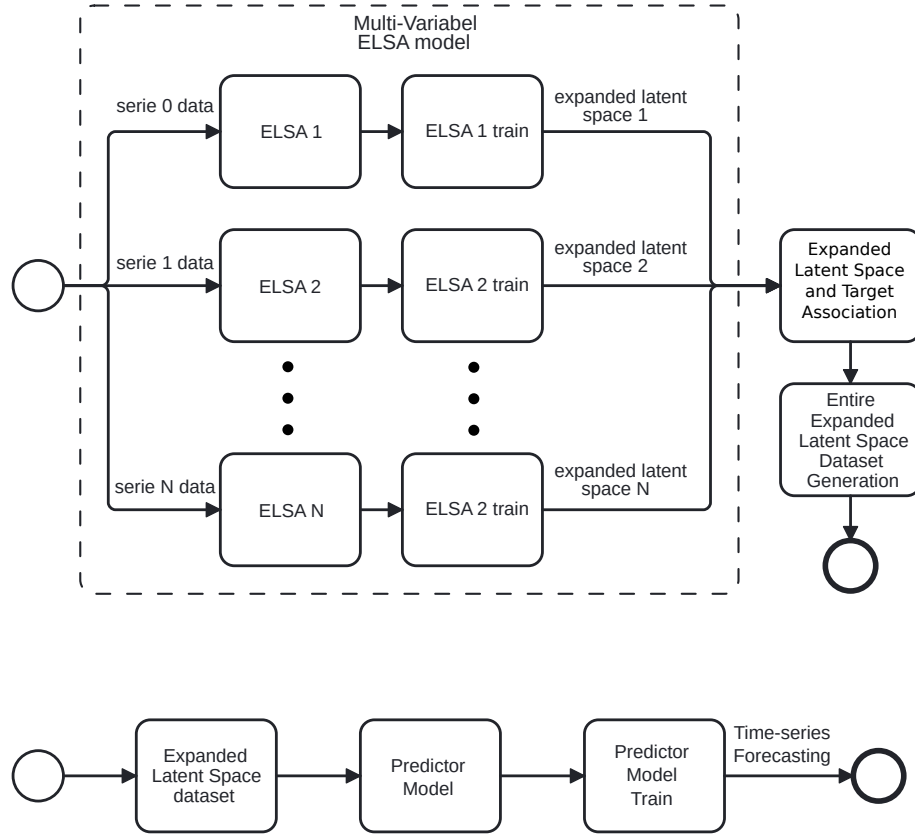


Figure 4.5: Flowchart representation of how a multi-variable ELSA network works.

From Figure 4.5, we can see that we can train N ELSA models to represent each input variable. With this, we can think of the ELSA network as multi-variable since it is sufficient to use an ELSA network for each variable separately and then create a

dataset with the expanded latent spaces (presenting all samples to their respective ELSA networks). Then, we can associate each expanded latent space with their respective target samples and use the expanded latent space values to train a predictor model.

In a previous work (de Oliveira, Aragão & Gonçalves 2024), we propose three modifications and extended variations for the original Modified Auto-Encoder model. We base these three model types on structural changes through the internal auto-encoder architecture, on the manner it manages the latent space, and on the approach to adjusting the model weights. From this study, it was possible to make an important decision regarding the model architecture. The proposed modification on its internal auto-encoder configuration did not promote the desired and computational cost reduction results. So, in this thesis we decided to come back to the original stacked auto-encoders proposed, arriving at the architecture showed in the previous subsection (4.2.2). From this same published article, we realized that the best way to train a predictor using the proposed networks, is training them in separated moments. The best overall results of our proposed model architecture were obtained from the variation that kept their weights unchanged during the training of the predictor network. For more details, please refer to our published work (de Oliveira, Aragão & Gonçalves 2024). This led us to the idea that it was possible for the proposed network to function only as generators of latent spaces that represented the inputs of a dataset. Thus, it is possible to generate a new dataset with latent spaces and from there train a predictor network. These observations obtained from this published article fully cover the choice of the current architecture and the training methodology proposed in this thesis and shown in Figure 4.5.

Chapter 5

Implementation

For a better understanding of the proposed model, it is essential to observe its behavior in various scenarios. Furthermore, the validations presented in this chapter aim to provide more robustness and clarity regarding the step-by-step implementation of the model.

5.1 Initial Validations

Following the introduction of the proposed model, we conducted several validations to better comprehend its functionality. Initially, a validation was performed to observe the behavior of the latent space values resulting from each of the internal autoencoders in a trained model, aiming to confirm the proposed concept. Subsequently, two more validations will be presented, where the first demonstrates the reproducibility in training an ELSA network, and the other allows us to observe the latent spaces for a set of networks trained with multivariate data. Although not the primary focus of this thesis proposal, all validations presented in this chapter utilized datasets composed of images. Classifying images is somewhat less complex than predicting time series. In classification tasks, each input can be considered independent of others since there is no connection between them. However, in time series prediction tasks, each sample or window of samples has a temporal connection, making it challenging to assess the robustness and obtain evaluative metrics of the proposed network in some validations. Therefore, for the first validation, we used the MNIST dataset, which contains grayscale images of handwritten arabic numerals from 0 to 9. This dataset comprises 60,000 training images and 10,000 test images, with a resolution of 28×28 pixels. In composing the validation datasets, images from the second half of the training set were utilized.

5.2 Proposal Validation

The premise of this validation is to demonstrate that training two (or more) autoencoders in parallel, with the same input, produces distinct latent spaces since the initial values for their parameters are different. For this purpose, a single ELSA network will be trained with a functional configuration to evaluate the generated latent spaces. Table 5.1 presents the hyperparameters values used in the network configuration for this validation.

Hyperparameter	Value
Training dataset size	600 (entries)
Validation dataset size	100 (entries)
Test dataset size	958 (entries)
ELSA network input size	784 ($28 * 28 * 1$) (neurons)
Size of each AE's latent space	28
Number of internal autoencoders	2
Expanded latent space size	56 ($28 * 2$)
Activation function	ReLU
Training epochs	50
Learning rate	0.005
Loss function	RMSE
Seed	40

Table 5.1: Table of hyperparameters used to demonstrate the creation of an expanded latent space.

In Table 5.1, the size of the ELSA network input is observed to be 784 neurons. This value is derived from the dimensions of the used image, where 28 represents the number of pixels for the x and y axes, and 1 denotes the number of color channels (1 for grayscale ranging from 0 to 255). Only entries corresponding to the digit 6 were selected (chosen randomly). Figure 5.1 depicts a graphical representation of the latent spaces obtained from the encoders of the individually trained model for a single (first) entry from the test dataset.

In Figure 5.1, it can be observed that the internal encoders generate different values for the same input. The Table 5.1 line “Size of each AE’s latent space” shows the latent space size of each internal autoencoder, i.e., we select 28 as the number of latent space generated samples. Statistical metrics such as mean and standard deviation can be extracted from these values. For this entry, the values for mean and standard deviation are 0.4121 and 0.4952 for encoder 1, and 0.4429 and 0.5359 for encoder 2, respectively. Subsequently, the entire test dataset was subjected to the model. For each presented entry, the mean and standard deviation of the values obtained in the latent spaces of each internal encoder were stored. Finally, a distribution of the means of each of the latent spaces, as well as the complete expanded latent space, was obtained. Algorithm 1 presents in details the step-by-step process followed to obtain the means and standard deviations that compose the distributions of each internal encoder in a trained ELSA network, based on the test dataset.

Following the steps described in Algorithm 1, it is possible to obtain a distribution of mean values generated from the 958 (see Table 5.1) entries in the test dataset. Figure 5.2 graphically illustrates the values obtained from the distribution of these values.

In Figure 5.2, it is noticeable that the values obtained for $encoder_1$ generate a different distribution compared to $encoder_2$. Quantitatively, the mean values were 0.5849 and 0.7218 for $encoder_1$ and $encoder_2$, respectively. Similarly, the obtained standard devia-

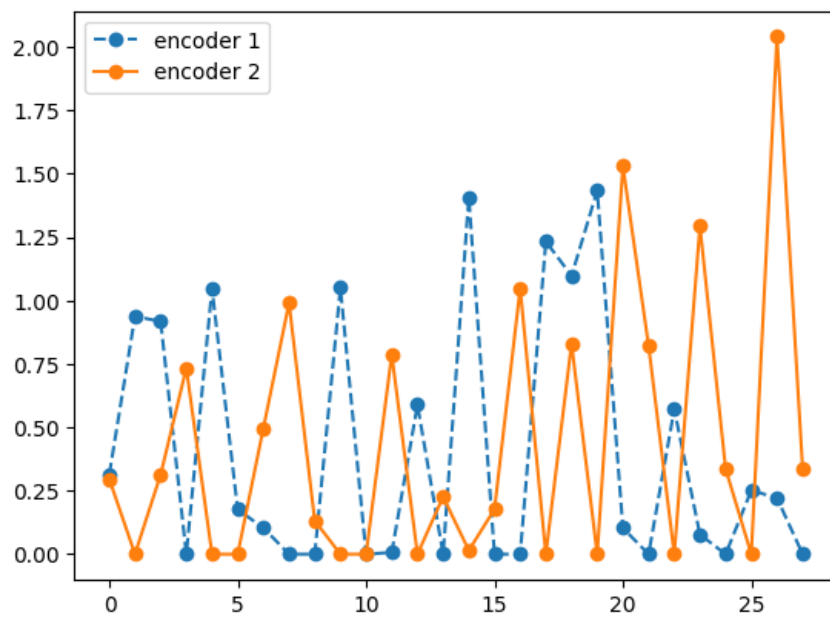


Figure 5.1: Image depicting the plot of generated values in the latent space for each of the internal encoders separately.

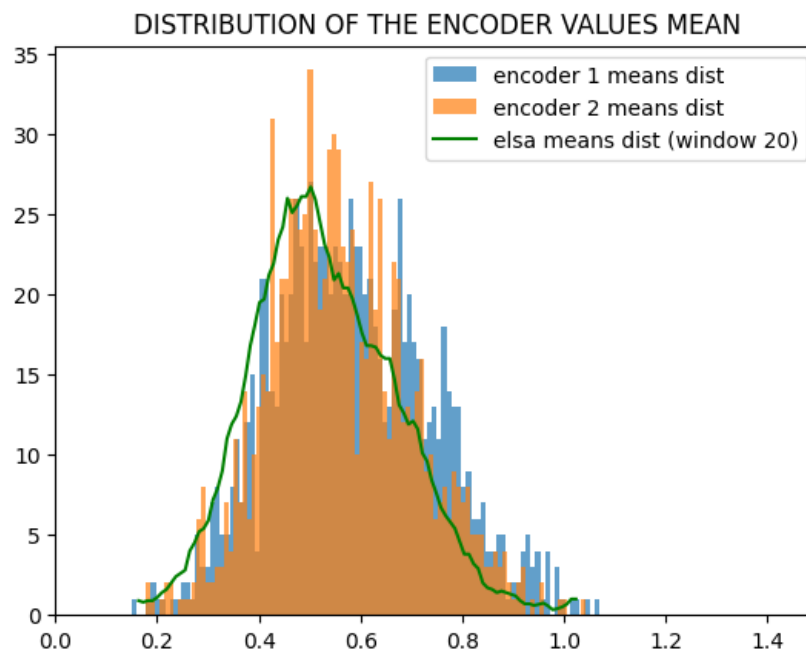


Figure 5.2: Image depicting the plot of distributions generated from the calculation of the mean for each of the values obtained in the internal encoders, along with a moving average plot of the distribution of values obtained in the expanded latent space.

Algorithm 1: Algorithm for obtaining the mean and standard deviation values generated from the mean and standard deviations for each of the internal encoders of a trained ELSA network, using the test dataset.

Data: X : test dataset; M : trained ELSA model

```

1   $data\_test \leftarrow X$ ;
2   $encoder_1 \leftarrow M.encoders[0]$ ;
3   $encoder_2 \leftarrow M.encoders[1]$ ;
   /* Empty vectors are created to store the values of each mean
   and standard deviation. */
4   $means_{e1} = vector[]$ ;
5   $stds_{e1} = vector[]$ ;
6   $means_{e2} = vector[]$ ;
7   $stds_{e2} = vector[]$ ;
8   $means_{ls} = vector[]$ ;
9   $stds_{ls} = vector[]$ ;
10 foreach  $input \in data\_test$  do
11    $encoded_1 \leftarrow encoder_1.forward(input)$ ;
12    $encoded_2 \leftarrow encoder_2.forward(input)$ ;
13    $encoded \leftarrow M.forward(input)$ ;
   /* Calculating the mean and storing the values in each
   vector respectively, for encoder 1. */
14    $mean_{e1} \leftarrow get\_mean(encoded_1)$ ;
15    $std_{e1} \leftarrow get\_std(encoded_1)$ ;
16    $means_{e1}.append(mean_{e1})$ ;
17    $stds_{e1}.append(std_{e1})$ ;
   /* Calculating the mean and storing the values in each
   vector respectively, for encoder 2. */
18    $mean_{e2} \leftarrow get\_mean(encoded_2)$ ;
19    $std_{e2} \leftarrow get\_std(encoded_2)$ ;
20    $means_{e2}.append(mean_{e2})$ ;
21    $stds_{e2}.append(std_{e2})$ ;
   /* Calculating the mean and storing the values in each
   vector respectively, for the expanded latent space. */
22    $mean_{ls} \leftarrow get\_mean(encoded)$ ;
23    $std_{ls} \leftarrow get\_std(encoded)$ ;
24    $means_{ls}.append(mean_{ls})$ ;
25    $stds_{ls}.append(std_{ls})$ ;
26 end
   /* Finally, from the mean and standard deviation vectors, the
   final means and standard deviations are calculated for each
   of the internal encoders and also for the expanded latent
   space. */
27  $mean_{e1f} \leftarrow get\_mean(means_{e1})$ ;
28  $std_{e1f} \leftarrow get\_std(stds_{e1})$ ;
29  $mean_{e2f} \leftarrow get\_mean(means_{e2})$ ;
30  $std_{e2f} \leftarrow get\_std(stds_{e2})$ ;
31  $mean_{lsf} \leftarrow get\_mean(means_{ls})$ ;
32  $std_{lsf} \leftarrow get\_std(stds_{ls})$ ;

```

tions were 0.5618 for $encoder_1$ and 0.7020 for $encoder_2$. This practically demonstrates that the architecture of a stacked autoencoder, receiving the same input, generates different latent space values. Thus, concatenating these latent spaces creates an expanded latent space that may capture more intrinsic characteristics of an input, making the basic premise of the proposed model true.

5.3 Reproducibility

The reproducibility of a proposal is a fundamental and essential characteristic for a scientific work. Thus, experiments conducted in a study can be reassessed and improved by other researchers. Studies related to artificial intelligence and machine learning sometimes fall short in this regard. Since the proposed work deals with a network that utilizes the variability of the network's initial parameters at initialization, it is crucial to demonstrate that once a result is obtained, it can be easily reproduced. For this purpose, a validation was conducted aiming to show that two different models, albeit with the same configuration and seed, result in identical trained models. We assigned a unique identifier to each model, with the same configuration as presented in Table 5.1, except for the seed, which was changed to the integer value of 35 (solely for differentiation from the previous validation). Figure 5.3 displays the distribution values obtained using the same methodology as the previous validation (Section 5.2).

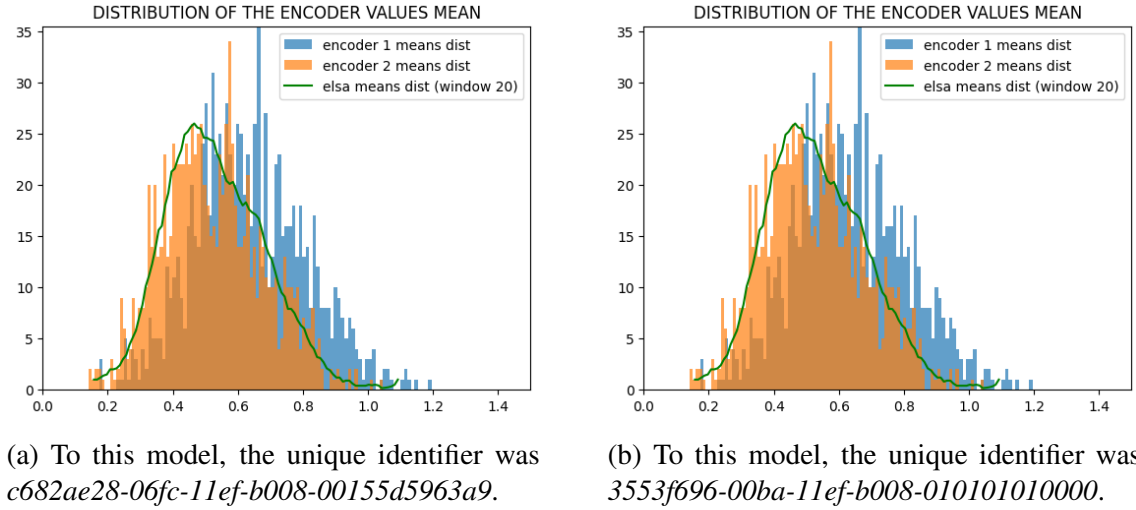


Figure 5.3: Plot of the distributions obtained from training a model equal to that presented in the previous validation, with a change in the value of the seed generator.

From Figure 5.3, it can be visually observed that the two trained models generate identical distributions. The quantitative values for the distributions generated from the two models are identical: 0.6340 for the mean and 0.7546 for the standard deviation of the distribution of $encoder_1$, and 0.52443 for the mean and 0.69763 for the standard deviation of $encoder_2$. Thus, it can be concluded that one model is identical to the other as long as they have the same hyperparameter configuration and seed.

Observing the graphical (Figures 5.2 and 5.3) and numerical results of the validations presented in Sections 5.2 and 5.3, the premise can be assumed: two models with identical hyperparameter configurations but different seed values result in different models.

5.4 Identical Internal Autoencoders

An approach that directly opposes the proposal is to use two internal autoencoders of the network with the same initial parameter values (weights). To observe such a situation, an ELSA network was generated with the same configurations as presented in Table 5.1. However, *autoencoder*₂ was assigned the same initial parameters (weights) as *autoencoder*₁, meaning the internal autoencoders are identical. Figure 5.4 presents the same graphical visualization of the distributions of the means generated for the test dataset using the same methodology previously presented in Algorithm 1.

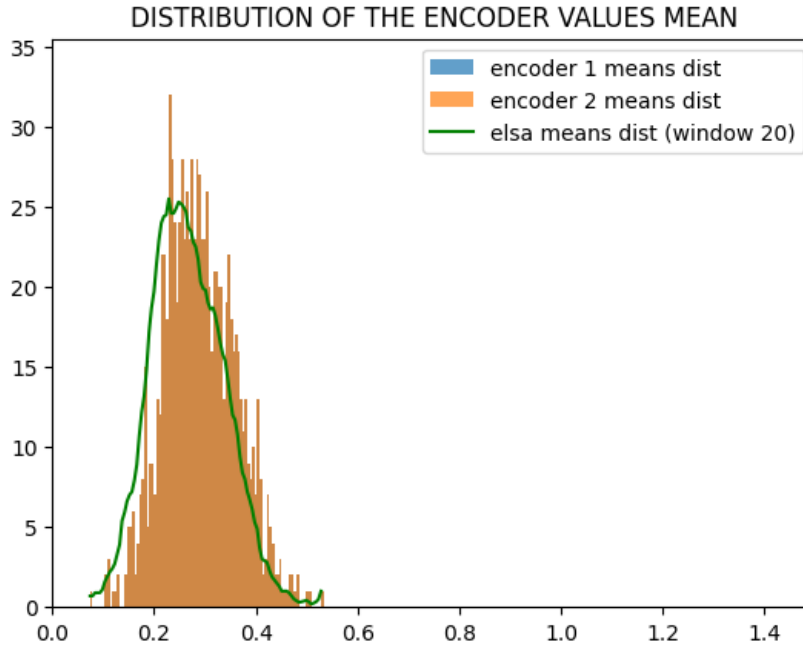


Figure 5.4: Image with the plot of distributions generated from the calculation of the mean for each of the values obtained in the internal encoders, along with a plot of moving average of the distribution of values obtained in the expanded latent space for a network with two identical internal autoencoders.

From Figure 5.4, it can be observed that there is no visual distinction between the distributions generated from *encoders*₁ and *encoders*₂. Quantitatively, the values obtained for mean and standard deviation for the distribution of both autoencoders were 0.28984 and 0.46347, respectively. From this validation, it can be noticed that using identical internal autoencoders (in initial parameters) would not bring any advantage to the resulting latent space, as both generate the same encoded information. Thus, it is important

to emphasize that the proposed stacked autoencoders architecture uses the variability of initial parameter values in artificial neural networks to achieve the expected results. However, this variability does not mean that after being trained, the models cannot be easily reproduced, as shown in Section 5.3.

5.5 Different Datasets

The previous validations were executed on only one dataset referring to the digit 6 present in the MNIST dataset. Here, the behavior of the distributions obtained from a multi-label dataset will be observed both visually and quantitatively. The data for digits 0, 3, 6, and 9 from the MNIST dataset will be used (also arbitrarily chosen). Analogously, the values presented in Table 5.1 were used as hyperparameters for each of the four networks, with only the difference in the quantity of samples in the test dataset. For the dataset referring to digit 0, the number of entries in the test dataset is 980. In datasets 3, 6, and 9, the number of entries is 1010, 958, and 1009, respectively. The methodology presented in Section 4.2.2 was used for training the networks. The set of Figures 5.5 shows the visual representation of the application of the methodology presented in Section 5.2 for the datasets referring to digits 0, 4, 6, and 9.

It can be observed that each of the figures presented in the set of Figures 5.5 has different behaviors for the distributions generated from their latent spaces. This applies both individually for each internal encoder in each of the different labels, as well as in the expanded latent spaces. From here, it is reasonable to assume that each set of generated latent spaces is different for each of the labels. Thus, a trained ELSA model can generate expanded latent spaces that represent a certain input, provided the input corresponds to the label of the trained model. Finally, it is possible to present a dataset to a properly trained ELSA model and obtain a set of expanded latent spaces that represent it. In this way, a classifier network can be trained with inputs being the expanded latent spaces of each label.

In the next validations, the influence of using expanded latent spaces as input for a classifier model will be evaluated. The first validation aims to compare the performance of the classifier when trained with inputs being expanded latent spaces generated by ELSA networks compared to regular inputs, i.e., non-latent spaces. The second validation aims to observe if the representation in the expanded latent space can promote better classifier performance compared to a conventional latent space representation of the same size in terms of neuron quantity.

5.6 Influence on Classification

This validation aims to assess the performance of a simple classifier model when trained with inputs generated from a model composed of ELSA networks and inputs without any prior encoding. For this validation, the entire MNIST image dataset, previously detailed at the beginning of Chapter 5, will be used. Table 5.2 presents the values for each of the hyperparameters used for training each ELSA network.

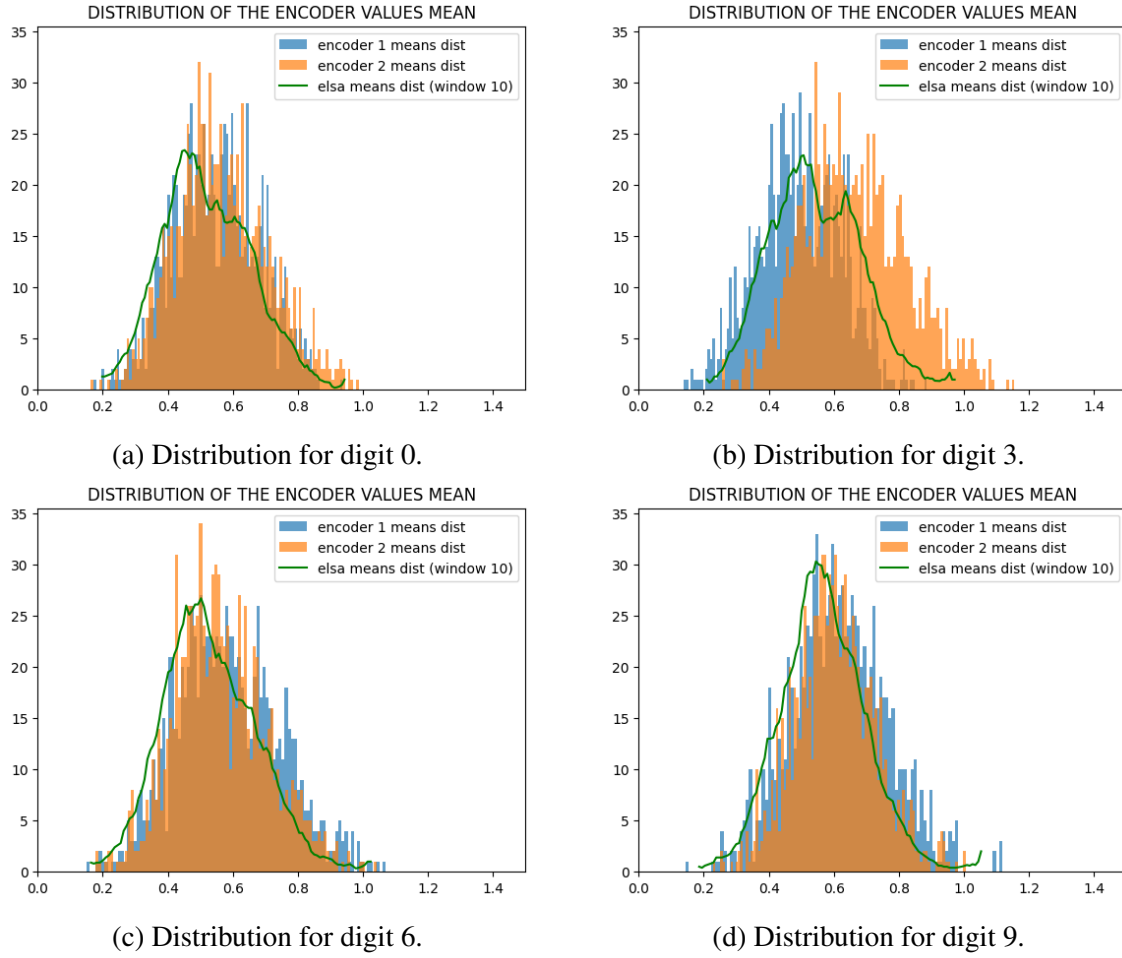


Figure 5.5: Visual representation of the distribution of means generated from $encoders_1$, $encoders_2$, and expanded latent space for each trained model.

Hyperparameter	Value
Training dataset size	600 (entries)
Validation dataset size	100 (entries)
Input size in the ELSA network	784 ($28 * 28 * 1$) (neurons)
Size of latent space in each AE	10
Number of internal autoencoders	2
Size of expanded latent space	20 ($20 * 2$)
Activation function	ReLU
Training epochs	50
Learning rate	0.005
Loss function	RMSE
Seed generator	40

Table 5.2: Table of hyperparameters used in each ELSA network for classifier training.

For each set of images corresponding to one of the digits in the MNIST dataset, the number of samples in the test dataset varies. Table 5.3 shows the sample sizes of each of the test datasets for MNIST.

Digit	Test dataset size
0	980
1	1135
2	1032
3	1010
4	982
5	892
6	958
7	1028
8	974
9	1009

Table 5.3: Table of sample sizes (images) of the datasets for training each ELSA network for the MNIST dataset.

Because the proposed model is based on the variability of the initial parameters of the internal autoencoders, when comparing it with another approach, it is good practice to vary the values of the seed generators and observe the variation of the evaluative metric in a distributed manner. Thus, 20 multi-label ELSA models were trained, with $S = \{0, 5, 10, \dots, 85, 90, 95\}$ being the variation of their seed generators. Along with the training of each of these models, a classifier network was trained, which in this case had the seed generator fixed at the value 42. For each of these classifiers, the input size (first layer of the network) is equal to the size of the expanded latent space, with 20 neurons. Similarly, 20 classifier networks were trained, in this case with seeds varying in $S = \{0, 5, 10, \dots, 85, 90, 95\}$ and their inputs being directly the pixel values of the image (without any encoding), resulting in the input size being 784. Finally, it is possible to obtain a distribution composed of the 20 values of the selected metric and, from there, extract components such as mean, median, and interquartile range of the results. Then, the results obtained from the distributions are compared, which generates greater robustness in the comparison, since the result of a training may be directly linked to the initialization of the parameters. It is also important to highlight that the networks comprising the ELSA model for each digit were trained using only 600 images out of the 60,000 available in the training set. This represents just 1% of the complete dataset, which, in this case, may be sufficient for the models to extract the relevant information needed for the classification task. The entire training dataset was presented only once to their respective ELSA models, thus, the training set in the expanded latent space representation was obtained and then presented to the classifier. In this validation, the chosen metric for evaluating the models is the final accuracy in classifying the all digits test dataset, previously detailed with more information in Table 5.3. Table 5.4 shows the values of each of the hyperparameters used in training the classifier networks for both trainings.

Classifier Hyperparameter	Value
Training dataset size	60000 (entries)
Test dataset size	10000 (entries)
Input size for classification	20 or 784
Number of classes	10
Training epochs	5
Learning rate	0.009
Loss function	NLL (Negative Log Likelihood)
Seed generator	42 or $S = \{0, 5, 10, \dots, 85, 90, 95\}$

Table 5.4: Table of hyperparameters used for the classifier networks.

In Table 5.4, the values of the hyperparameter "Input size for classification" are set to 20 for the classifiers using the expanded latent spaces as inputs and 784 for those using the images without any processing or encoding. Similarly, the values of the hyperparameter "Seed generator" are fixed at 42 for the 20 classifiers that had the expanded latent spaces as inputs and vary in $S = \{0, 5, 10, \dots, 85, 90, 95\}$ for the 20 classifiers that used the images without any processing or encoding. The rest of the hyperparameters remained the same for all classifiers. Figure 5.6 presents the distributions in *boxplot* form obtained from the described validation.

From Figure 5.6, it is observed that the distribution generated from the measured accuracy data referring to the *boxplot* represented in the figure by the mark "No" (no ELSA) does not have visually significant variation in its values. Numerically, the values for mean, median, and interquartile range are 91.075, 91.085, and 0.062, respectively. The low value for the interquartile range indicates a statistically insignificant variation in the accuracy values obtained by the initial parameters variation for this classifiers. The *boxplot* represented by the mark "N=2" (number of internal autoencoders) has visually significant variation in its values. Numerically, the values for mean, median, and interquartile range are 95.345, 95.241, and 3.397, respectively. The variation in the values obtained in this result indicates that the values of the network parameters at their initialization have considerable consequences on the final model performance. Finally, based on the values shown, it is evident that a classifier achieves better accuracy when receiving inputs generated from the codifications of an ELSA model. In the next validation, we will assess how the performance of a classifier varies concerning the use of inputs in conventional latent space compared to inputs composed of the proposed expanded latent space.

5.7 Conventional Latent Space $\times$ Expanded Latent Space

This validation aims to assess whether there are advantages in using an expanded latent space compared to a conventional latent space generated by a network composed of only one autoencoder and not N . This is equivalent to using the ELSA network with the quantity of internal autoencoders $N = 1$, but with its latent space value multiplied by N times the quantity of neurons in the model for comparison, to maintain fairness in terms

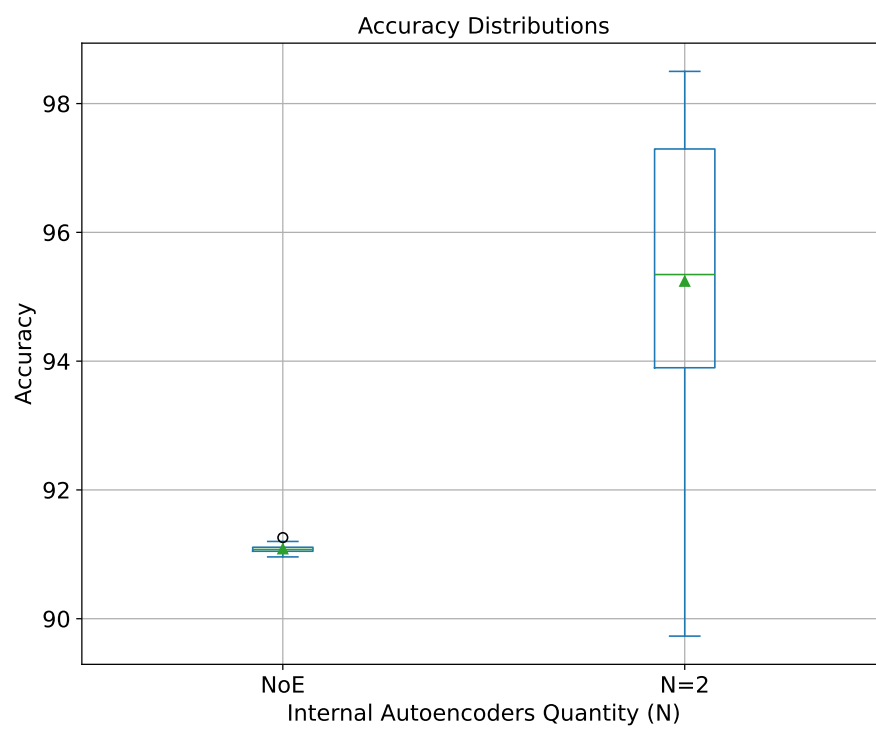


Figure 5.6: Boxplot image of the distributions generated from extracting the accuracy of the test dataset for the described validation.

of network complexity. Table 5.5 shows the hyperparameter values in the comparison.

Hyperparameter	Conv. L. Space	Exp. L. Space
Training set size	600	600
Validation set size	100	100
Input size in ELSA network	784	784
Latent space size	20	20
Activation function	ReLU	ReLU
Quantity of internal autoencoders	1	2
Training epochs	40	40
Learning rate	0.005	0.005
Loss function	RMSE	RMSE
Seed generator	$S = \{0, 5, \dots, 90, 95\}$	$S = \{0, 5, \dots, 90, 95\}$

Table 5.5: Table of hyperparameters used in each ELSA network with expanded latent space and conventional latent space for classifier training.

In Table 5.5, it is observed that the model hyperparameters are comparable since the final latent space size is the same, one being generated from the concatenation of two stacked autoencoders (expanded latent space), and the other being a conventional latent space. Tables 5.3 and 5.4 show the values for the test datasets and classifier (values related to the ELSA model) used in this validation. It is important to reaffirm that all the methodology explained in the validation presented in Section 5.6 was maintained. Figure 5.7 presents in boxplot form the values of the distributions obtained from the described validation using the MNIST dataset.

From Figure 5.7, it is possible to observe that the distribution obtained from the networks trained using the dataset generated by an expanded latent space has visually a distribution with a smaller interquartile range and higher mean and median in classification accuracy. Quantitatively, the distribution referring to the “N=1” label, with a conventional latent space, presented values of 91.985 for the median, 90.708 for the mean, and 6.675 for the interquartile range. On the other hand, the distribution obtained from the networks trained with the expanded latent space, with “N=2” labeling, presented values of 95.345, 95.241, and 3.397 for median, mean, and interquartile range, respectively. For this comparison, high mean and median values, together with smaller interquartile ranges, indicate that the models had a more resistant behavior to changes in the seed generator values and consequently to changes in the initial parameters’ values of the networks. To further assess the consistency of the results obtained in this validation, the same methodology was applied to the FashionMNIST dataset. This dataset is similar to MNIST but contains images of 10 different types of clothing items, namely, T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. In addition to this change, this dataset is perfectly balanced, where the quantity of entries for each of the labels is exactly the same, 6000 for training and 1000 for testing. Finally, totaling 60000 entries for training and 6000 for testing. In addition to the change in the dataset, the value in the final latent space size was also varied to observe if there would be any change in the scenario. The chosen

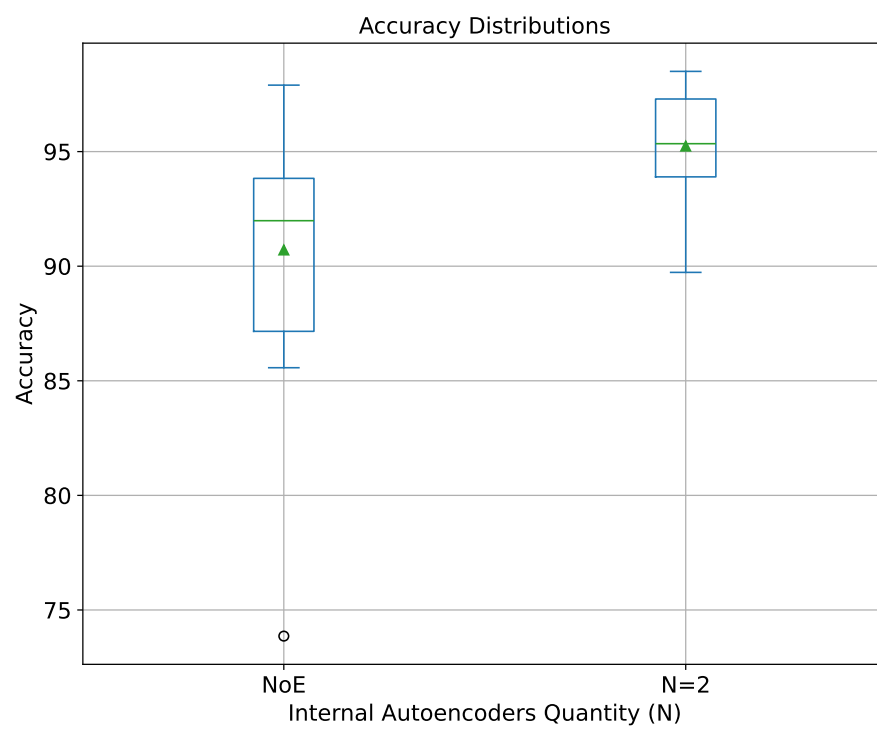


Figure 5.7: Boxplot image of the distributions generated from the extraction of test dataset accuracy for the described validation.

space values were incremented by 20, twice, thus being 20, 40, 60. Figure 5.8 presents in boxplot form the values of the distributions obtained from the described validation using the FashionMNIST dataset.

From Figure 5.8, it's observable that the distributions generated from the expanded latent space visually exhibit a distribution with a smaller interquartile range and higher mean and median in classification accuracy, reinforcing the results obtained in the previous validation with MNIST. Table 5.6 presents the values for mean, median, and interquartile range for each of the validation combinations.

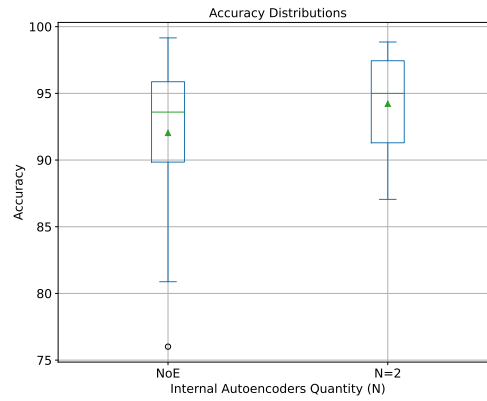
N	Median	Mean	Interquartile Range	Expanded Latent Space Size
1	93.595	92.029	6.028	20
2	94.995	94.217	6.152	20
1	98.455	98.158	1.067	40
2	99.710	99.673	0.118	40
1	99.365	99.321	0.413	60
2	99.870	99.877	0.070	60

Table 5.6: Table of quantitative values obtained from the distributions generated in the comparison validation for the FashionMNIST dataset.

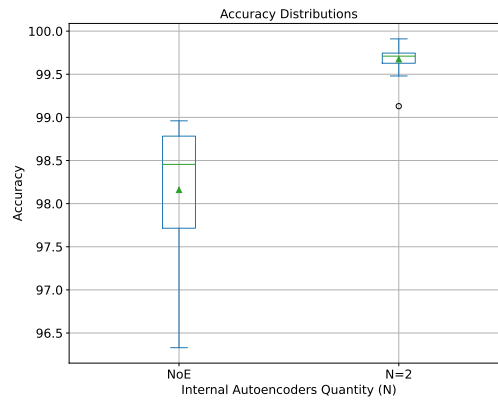
Table 5.6 numerically demonstrates that for different seed generator values of the network's initial weights and different sizes of latent spaces, classifiers trained using expanded latent spaces as inputs consistently show an improvement in the comparison metric used. The best results obtained were highlighted in italic, where only to the latent space size of 20, the interquartile range has a slightly improvement to "N=1" concerning "N=2". This validation suggest that the expanded latent space can make the classification of a simple MLP model more accurate compared to a conventional latent space of the same size.

5.8 Variation of N

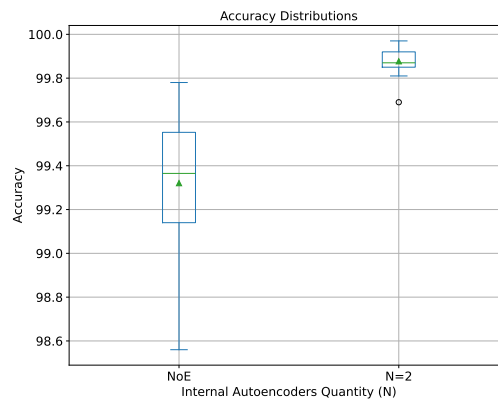
Based on the results obtained in the validations conducted so far, it can be assumed that the proposed approach holds advantages for the task of encoding a dataset used as input for a simple image classifier. The aim of the following validation is to observe the behavior of the distributions obtained when varying the quantity of internal autoencoders within the networks. Therefore, this validation will be divided into two parts. Initially, we evaluate this variation while keeping the values of the latent spaces of each autoencoder fixed, only varying their quantity and the size of the expanded latent space. The same configuration and dataset from the previous validation (Validation 5.7) were used for $N = \{2, 3, 4, 5\}$. Thus, the expanded latent spaces for each distribution had sizes of 20, 30, 40, and 50, respectively. These values result from multiplying the quantity of neurons in the last layer of the internal encoders of each autoencoder by 10, and the quantity N of internal autoencoders. Figure 5.9 presents, in boxplot form, the values of the distributions obtained from the described validation using the previous used FashionMNIST dataset.



(a) Distributions for the latent space with 20 neurons.



(b) Distributions for the latent space with 40 neurons.



(c) Distributions for the latent space with 60 neurons.

Figure 5.8: Visual representation in boxplot form of the distributions generated from the extraction of test dataset accuracy for the described validation.

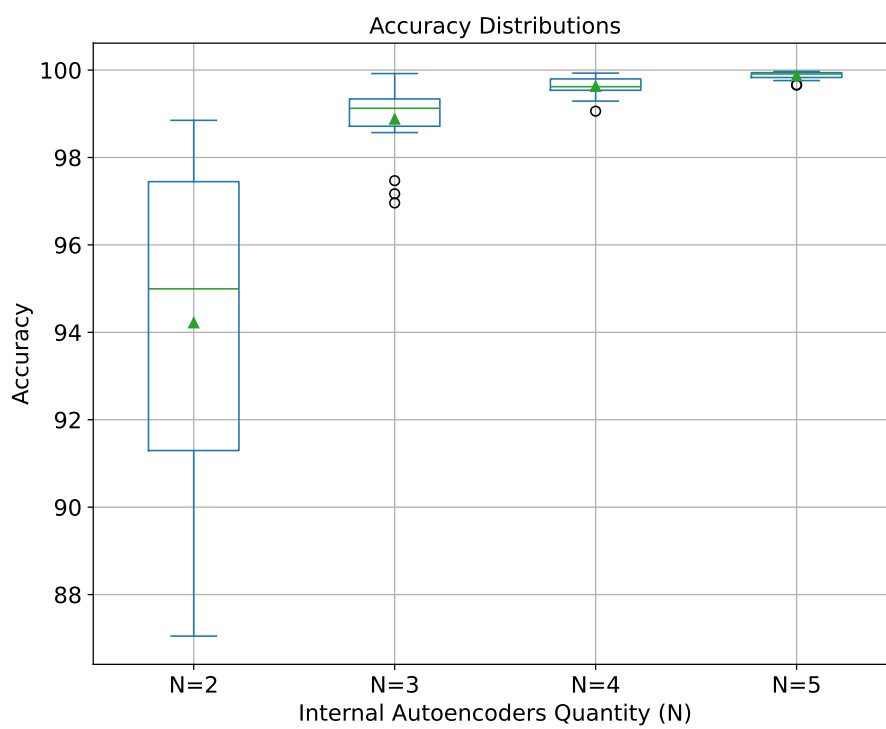


Figure 5.9: Boxplot representation of the distributions generated from the extraction of accuracy from the test dataset for the described validation.

Figure 5.9 visually demonstrates that as the quantity of internal autoencoders increases, the average accuracy values and medians tend to increase, while the interquartile range decreases. Table 5.7 presents the values for mean, median, and interquartile range for each combination of the validation.

N	Median	Mean	Interquartile Range	Expanded Latent Space Size
2	94.995	94.217	6.152	20
3	99.125	98.878	0.625	30
4	99.620	99.628	0.260	40
5	99.910	99.869	0.110	50

Table 5.7: Table of quantitative values obtained from the distributions generated in the comparison validation for the FashionMNIST dataset.

The values obtained in the validation presented in Table 5.7 demonstrate that the model's tendency is to increase classification accuracy as the value of N increases. However, this can also be attributed to the increase in the size of the expanded latent space, which indeed provides the classifier with more quantitative information. As a second part of this validation, we also evaluate the increase in the value of N , but with the size of the expanded latent space kept constant. To achieve this, the least common multiple of the values 2, 3, 4, 5 was found to be 60. Thus, the quantity of each internal autoencoder in the networks is the ratio between 60 and N . Since 6 is also a common denominator of 60, it was also added to the validation. Finally, the latent space values for each of the internal autoencoders are 30, 20, 15, 12, and 10 for N values of 2, 3, 4, 5, and 6, respectively. Additionally, for this validation, a dataset composed of images that can be considered more complex was used, aiming to make the classification more challenging for the models. In this case, the CIFAR10 dataset was used. This dataset has a similar number of entries to the previously described datasets (MNIST and FashionMNIST), with the difference being that the images have higher resolution (32x32) and three channels, derived from colors (RGB). CIFAR10 contains images of Airplanes, Automobiles, Birds, Cats, Deer, Dogs, Frogs, Horses, Ships, and Trucks. The fact that the images are not contained in any specific group, such as digits and pieces of clothing, may make the classification a more difficult task for the models. In this second part of the validation, the same configurations were also used for the new dataset and network architectures. Figure 5.10 presents, in boxplot form, the values of the distributions obtained from the described validation using the CIFAR10 dataset.

From Figure 5.10, it is visually apparent that for the values of $N = \{2, 3, 4\}$, the means and medians of the accuracies increase, while there is no considerable visual difference in the interquartile range. For the values of $N = \{5, 6\}$, the average and median accuracy values tend to decrease. This behavior suggests that there is a trade-off between the size of the latent space of each internal autoencoder and its quantity in composing the expanded latent space. Thus, depending on the size at which the internal latent space of each autoencoder is set, the best option is to decrease the quantity of autoencoders and increase the size of the latent space for each of them. Table 5.8 presents the values for

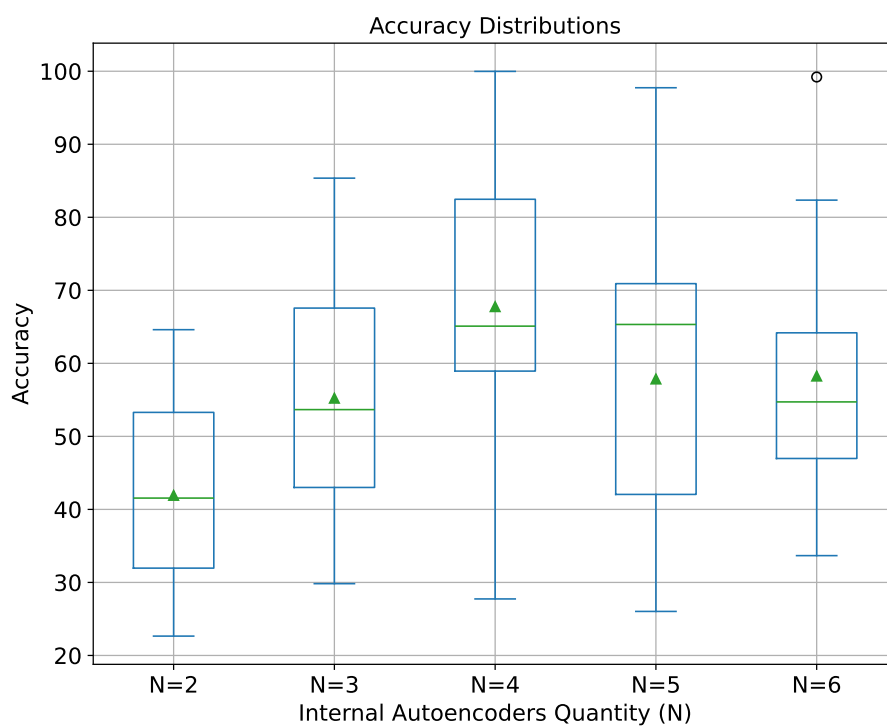


Figure 5.10: Boxplot representation of the distributions generated from the extraction of accuracy from the test dataset for the described validation.

mean, median, and interquartile range for each combination of the validation.

N	Median	Mean	Interquartile Range	Expanded Latent Space Size
2	41.550	41.924	21.328	60
3	53.665	55.228	24.562	60
4	65.095	67.765	23.530	60
5	65.325	57.859	28.862	60
6	54.720	58.260	17.208	60

Table 5.8: Table of quantitative values obtained from the distributions generated in the comparison validation for the CIFAR10 dataset.

The values obtained in the validation presented in Table 5.7 demonstrate that the model tends to improve classification accuracy as the value of N increases up to $N = 4$, where at $N = \{5, 6\}$, the values tend not to improve or even worsen. It is also noteworthy the low values for final accuracy in classification, with means and medians around 40 to 60 for all tested combinations. This indicates that the dataset exhibits higher complexity in its samples, which can be addressed by increasing the complexity of the networks as well, by adding an internal layer with a value equal to the expanded latent space in each of them. We performed previously validations where we used this configuration (adding an internal layer to each autoencoder), however, the obtained accuracy values were high, showing no significant variation within the distribution, thus not contributing to what is desired to observe in this validation. Therefore, as it was not easily possible to observe a variation in high accuracy values for more complex networks, we decided to decrease the complexity of the input data. Thus, the validation was redone, but the images underwent preprocessing; all of them were converted to grayscale, thus having 32x32 resolution pixels and only one channel (gray) instead of three (RGB). Figure 5.11 presents, in boxplot form, the values of the distributions obtained from the modifications made in the described validation.

Figure 5.10 visually demonstrates that for high accuracy values, from “ $N=2$ ” to “ $N=3$ ”, the means and medians of the accuracies increase, while there is a considerable visual difference in the interquartile range. Moving from “ $N=3$ ” to “ $N=4$ ”, a small visual difference between the mean values and the interquartile distances is observable. For “ $N=5$ ” and “ $N=6$ ”, a slight improvement in the metric values is noted compared to the other boxes, however, almost no modification between themselves, except for the interquartile distance which increases slightly from “ $N=5$ ” to “ $N=6$ ”. Table 5.9 presents the values for mean, median, and interquartile range for each combination of the validation.

The values obtained for this validation presented in Table 5.9 show that the model’s tendency is to increase classification accuracy as the value of N increases. In this validation, we highlight in *italic* the best values for each of the metrics, where $N = 5$ achieved better values for two of the three collected metrics. As the objective of this validation is not to obtain a better configuration for the model, but rather to observe its behavior in the variation of the value of N , it is possible to reiterate the partial conclusion of the previous validation, where there is a trade-off between the size of the internal latent space of each

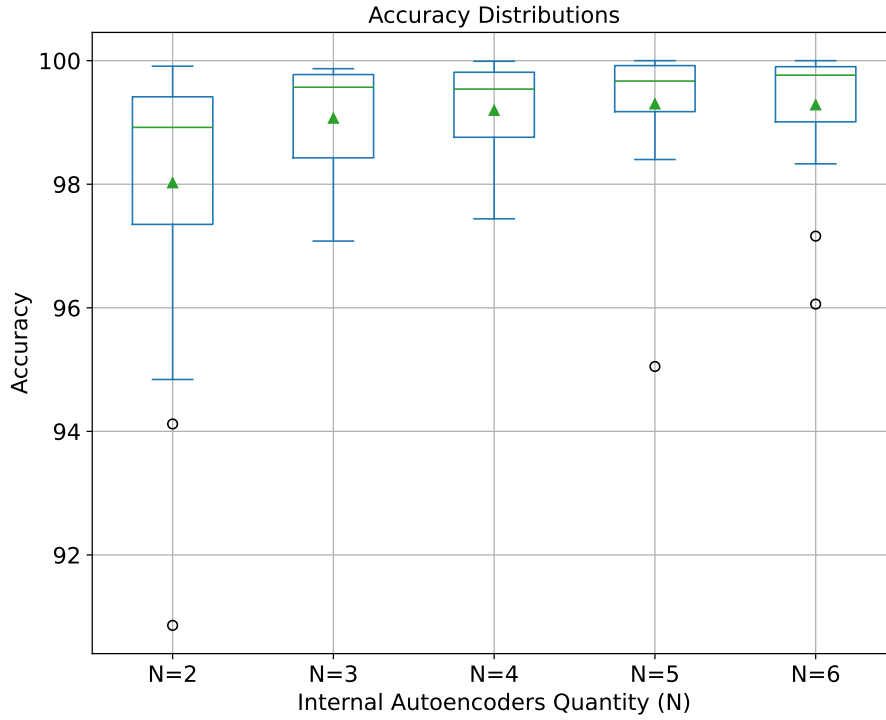


Figure 5.11: Boxplot representation of the distributions generated from the extraction of accuracy from the test dataset for the described validation.

N	Median	Mean	Interquartile Range	Expanded Latent Space Size
2	98.920	98.023	2.065	60
3	99.570	99.069	1.348	60
4	99.540	99.196	1.052	60
5	99.670	99.300	0.745	60
6	99.765	99.285	0.893	60

Table 5.9: Table of quantitative values obtained from the distributions generated in the comparison validation for the CIFAR10 dataset.

autoencoder and the quantity of autoencoders present in the model for generating the expanded latent space. Additionally, it is also possible to reinforce the conclusions of all the validations described in this chapter, where the use of the expanded latent space can offer a better representation of an input than a conventional latent space, as the increase in the quantity of internal autoencoders tends to increase the classification accuracy.

Chapter 6

Experiments

This chapter presents the experiments conducted to evaluate the information absorption capacity of the proposed model within the context of COVID-19 time series prediction. Initially, the acquisition and significance of the dataset will be described. Subsequently, an experiment will be proposed to evaluate the trained models for short, medium, and long-term forecasting. Based on the determination of the model's best fit within these perspectives, the most suitable models will be selected for a more detailed analysis of the prediction task. Finally, related works from the literature will be chosen for a direct comparison with the proposed model, aiming to position our work within the state of the art.

6.1 COVID-19 Dataset

For the following experiments, we used a COVID-19 time series dataset compiled by our research group. This dataset consists of time series containing information on the number of deaths and cases for the city of São Paulo, Brazil. These two initial series include data directly associated with the pandemic, previously visualized in Figures 4.1a and 4.1b. These series were provided daily by Wesley Cota¹, who voluntarily collected information from official government agencies and made this data available in an organized manner for most Brazilian municipalities. With the Brazilian Ministry of Health's shift to weekly data updates, his primary data source ceased updates from March 18, 2023. To avoid inconsistencies, data updates were discontinued from that date in the repository. For this reason we used data from April 1, 2020, to February 28, 2023, in our experiments.

In addition to the time series for the number of deaths and cases in São Paulo, Brazil, we utilized three more time series that aggregate environmental information. The selected series represent values for temperature, humidity, and the Air Quality Index (AQI). These values are collected through sensors placed in specific environmental locations. The World Air Quality Index ²³ is a non-profit organization that positions such sensors in various regions worldwide and stores environmental information, including certain air

¹<https://github.com/wcota/covid19br>

²<https://waqi.info/>

³<https://aqicn.org/>

pollutants. The AQI is calculated based on the **maximum** value recorded among the measured pollutants, which include No₂ (Nitrogen Dioxide), O₃ (Ozone), PM₁₀ (inhalable particles with diameters less than 10 micrometers), PM_{2.5} (inhalable particles with diameters less than 2.5 micrometers), Co (Carbon Monoxide), and So₂ (Sulfur Dioxide). Therefore, if the highest value measured among these pollutants is for Co, that value represents the AQI for that day. Besides AQI, the dataset also provides daily average values for temperature and humidity. Figure 6.1 shows the values for humidity, temperature, and AQI along with their respective seven-day moving averages for the city of São Paulo, Brazil.

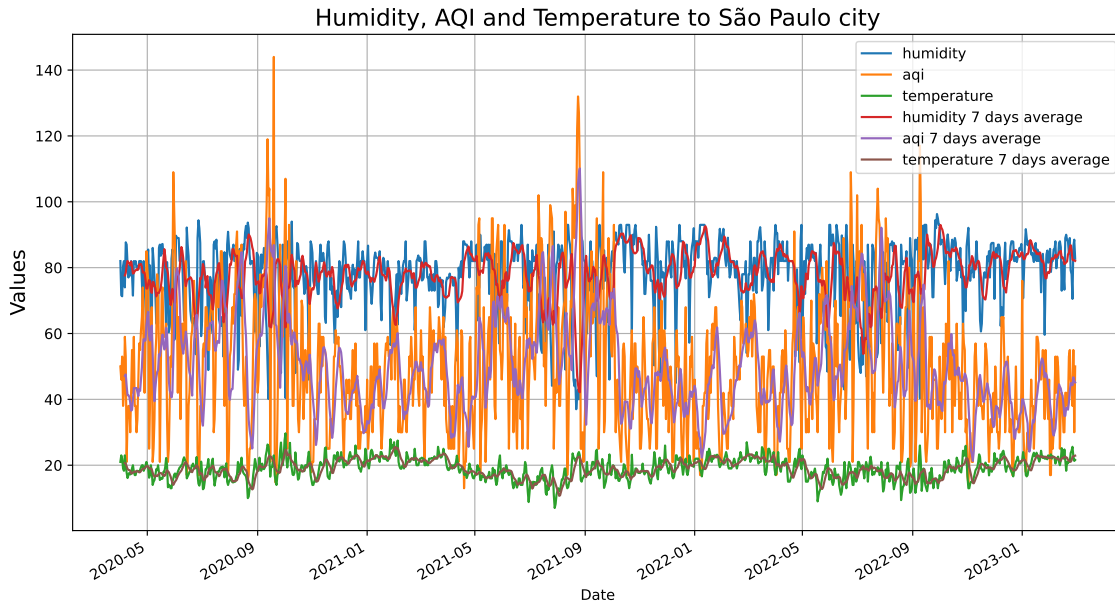


Figure 6.1: Plot of the number of humidity, temperature, aqi, and their moving average for the São Paulo city, Brazil, from 2020-04-01 to 2023-02-28.

Figure 6.1 and Figures 4.1a and 4.1b visually compose the dataset used for this first experiment.

6.2 Short-medium-long term forecasting

The first method we chose to evaluate the proposed model was in the task of short, medium, and long-term prediction for time series related to COVID-19. For the short term, we chose to use the values of 7 and 14 days (samples), equivalent to one and two weeks of data. For the medium term, we defined the prediction as 28 and 35 days, which correspond to four and five weeks. For the long term, we defined the prediction as 49 and 56 days, which are seven and eight weeks, respectively. The five previously described time series (deaths, cases, humidity, temperature, and AQI) were used in their seven-day moving average forms. Additionally, a second preprocessing step was performed on

the data. The data were normalized using the Standard Scale method. Equation (6.1) demonstrates the application of the Standard Scale to a set of series $S = \{S_1, S_2, \dots, S_i\}$.

$$S_i = \frac{S_i - S_{mean}}{S_{std}} \quad (6.1)$$

From Equation (6.1), S_i represents each series i in the dataset S , and S_{mean} and S_{std} are the mean and standard deviation of the series, respectively. This type of operation results in a new normalized dataset, where the mean value is zero and the standard deviation S_{std} remains unchanged.

For the task of time series prediction, sample windowing is commonly used to maintain the temporal relationships that samples have among them. These windows function as independent temporal regressors, fulfilling the objective of preserving these connections between the closest samples. The training dataset was constructed so that each input window has an overlap of $(w - 1)$ samples, meaning that only the sample corresponding to the current time stamp is new relative to the window at the previous time stamp. In this case, w is the prediction window size. Therefore, for the prediction of 7 days, there will be an overlap of 6 samples between the input at time t and the input at time $t + 1$, the same is applied to the others selected windows size.

For this experiment, we used data from the five input time series from April 1, 2020, to December 31, 2023, for training and validation. The windowing for the training set was as previously described, while for the validation set, the same data were used, but there was no overlap of samples between inputs. For the test set, we used data from January 1, 2023, to February 25, 2023, totaling 56 days. This value was intentionally chosen as it is the maximum possible value for a long-term prediction window. Thus, in this experiment, we will evaluate whether the employed model is more accurate in making multiple shorter predictions for the same test dataset than making fewer longer predictions. In short-term predictions, such as 7 and 14 days, for example, the model will have to make more predictions to cover all 56 samples of the test set. Conversely, at the extreme window value of 56 samples, only one prediction is made. The final evaluation metric used to quantify the models was the average Root Mean Squared Error (RMSE) in predicting the five input series. The RMSE is calculated for the prediction of the 56 test samples for the series separately, and then the average of these errors is computed and stored. As in Chapter 5, we used distributions generated from the extraction of the chosen metric by varying the values of the seed generators for the parameters of the ELSA networks. Table 6.1 presents the selected hyperparameters for the ELSA networks in this experiment.

From Table 6.1, the values for the set of windows W represent the input size for each ELSA network. As described in Chapter 5, one ELSA network is trained for each input series, totaling five in this experiment. Each network has a latent space size corresponding to the row “Size of latent space in each AE” and includes 4 internal autoencoders, resulting in a total of 80 values for the expanded latent space. These networks are trained for 50 epochs, with a learning rate of 0.004, using ReLU activation functions and the RMSE loss function. Consequently, 20 multivariate ELSA models were trained, with $S = \{0, 5, 10, \dots, 85, 90, 95\}$ as the variation in their seed generators. Alongside the train-

Hyperparameter	Value
Input Size	$W = \{7, 14, 28, 35, 49, 56\}$
Hidden AE layer size	160 (80 * 2)
Size of latent space in each AE	20
Number of internal autoencoders	4
Size of expanded latent space	80 (20 * 4)
Activation function	ReLU
Training epochs	50
Learning rate	0.004
Loss function	RMSE
Seed generator	$S = \{0, 5, 10, \dots, 90, 95\}$

Table 6.1: Table of hyperparameters used in each ELSA network.

ing of each of these models, one predictor network was also trained, with its seed generator fixed at the value 45. Table 6.2 shows the selected hyperparameters for the predictor networks in this experiment.

Predictor Hyperparameter	Value
Input size for forecasting	400 (80 * 5)
Hidden layers size	[100, 80]
Forecasting layer window size	$W = \{7, 14, 28, 35, 49, 56\}$
Training epochs	200
Input features quantity	5
Internal predictors quantity	5
Learning rate	0.001
Loss function	RMSE
Seed generator	45

Table 6.2: Table of hyperparameters used for the predictor networks.

From Table 6.2, the predictor network consists of 5 internal predictors (Internal predictors quantity), each responsible for predicting one of the input series (Input features quantity). Each of these internal predictors has the configuration specified by the hyperparameters in the rows “Input size for forecasting”, “Hidden layers size”, and “Forecasting layer window size”. The remaining rows present other hyperparameters used in the training of the predictor network. It is important to note that, unlike the training of the ELSA networks, where each one is trained at a distinct time, all internal predictors are trained at the same time, i.e., their parameters are adjusted simultaneously during the training process. As discussed in Chapter 5, keeping the seed generator for this network constant within the experiment assigns the attribution of a good or bad result to the extent to which the ELSA networks can absorb relevant information from the dataset. All hyperparameters showed in Tables 6.1 and 6.2 were defined based on Chapter 5 validations and our

expertise with the proposed model.

Ultimately, the expectation for this experiment is to have 6 distributions composed of the prediction errors (RMSE) for the 56 samples of the test set. Each distribution represents one of the selected prediction windows, allowing us to evaluate which scenario the proposed model best fits within the context of time series prediction related to the COVID-19 pandemic.

6.3 Works Comparison

In this experiment, we will directly compare our proposed model with two selected works from the existing literature. The objective is to situate our model within the context of validated works that have undergone rigorous evaluation by the scientific community. Identifying closely related literature is challenging due to the unique methodology we employ. This challenge is further compounded by the decreased relevance of pandemic-related research over the past two years. Consequently, our comparison will disregard the publication date, relevance, and citation count of the articles.

The first work we selected was the article “COVID-19 Pandemic Forecasting Using CNN-LSTM: A Hybrid Approach” (Zain & Alturki 2021). The work presents a method for predicting COVID-19 cases using a combination of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. The hybrid model incorporates two convolutional layers, each with 64 filters and a kernel size of 3, followed by a max pooling layer, a dropout layer, an LSTM layer with 200 units, and a fully connected layer. The dataset comprises daily confirmed COVID-19 cases from January 4, 2020, to August 14, 2020, with training data spanning January 4, 2020, to July 17, 2020, and test data from July 18, 2020, to August 14, 2020. The authors also use a 4 weeks window from August 15, 2020 until September 11, 2020 for forecasting the subsequent week (September 12, 2020–September 18, 2020) to evaluate the proposed approach. The model’s input utilizes a 7-day window size to forecast the next 7 days of confirmed cases. The hyperparameters were optimized using the Optuna framework, with the best results achieved using 472 epochs, a batch size of 22, and a verbose level of 1. The model’s performance was evaluated using Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), and Relative Root Mean Squared Error (RRMSE). The CNN-LSTM model demonstrated superior accuracy with a RMSE of 8780.7, to the forecasting set and 13275.0 to test data, outperforming other models such as CNN1D and standalone LSTM, which had higher RMSE values. All evaluated model were trained using the MinMaxScaler to scale the data, where else to calculate the metrics the data was unscaled. These findings underscore the effectiveness of the CNN-LSTM hybrid approach in accurately forecasting COVID-19 cases over a short-term period. This work has a particular resemblance with our proposed model due it use a specific neural network architecture to extract information from data and another one to perform the forecasting. That was the closest work to ours that we found in our search of the literature.

To ensure a fair and accurate comparison with the methodology used in the referenced study, we reformulated the dataset to match exactly. We assembled a training set from January 4, 2020, to July 17, 2020, as proposed. This dataset was scaled using “Min-

MaxScaler” without any additional data processing and used to train the ELSA network. Table 6.3 presents the hyperparameter values used for training the ELSA network.

Hyperparameter	Value
Input Size	7
Size of latent space in each AE	64
Number of internal autoencoders	2
Size of expanded latent space	128 (64 * 2)
Training epochs	100
Learning rate	0.004
Loss function	RMSE
Seed generator	$S = \{0, 1, 2, \dots, 198, 199\}$

Table 6.3: Table of hyperparameters used in the comparison ELSA network.

From Table 6.3, we note that no hidden layers were used within the internal autoencoders of the ELSA network. To ensure a fair comparison, we used a similar number of internal autoencoders and the same size for their latent space, with 64 neurons for each of the 2 autoencoders, resulting in an expanded latent space of 128 values. The network was trained for 100 epochs, with a learning rate of 0.004 and RMSE as the loss function, using an input window of 7 samples. All these values were defined based on the comparison work, the Chapter 5 validations and our expertise with the proposed model. It is important to note that for this experiment, the model is not multivariate, as the comparison study does not evaluate this approach. Following our model’s proposal, we also trained a predictor MLP network. Table 6.4 presents the hyperparameter values used for training the predictor network.

Predictor Hyperparameter	Value
Input size for forecasting	128
Hidden layer size	60
Forecasting layer window size	7
Training epochs	250
Input features quantity	1
Internal predictors quantity	1
Learning rate	0.004
Loss function	RMSE
Seed generator	45

Table 6.4: Table of hyperparameters used for the predictor network.

From Table 6.4, we see that the predictor network comprises a single predictor with an input size matching the expanded latent space, followed by a hidden layer of size 60, and finally, the forecasting layer with the desired forecast window size. It is important to highlight that due to the nature of our model, the input data size cannot be altered, being

7 (1 week) in this case. Thus, in the evaluation part of the comparison proposal where the authors use a 4-week input window from August 15, 2020, to September 11, 2020, to predict from September 12 to 18, we used an input of the preceding 7 days for prediction. In other words, to predict from September 12 to 18, we used the input from September 5 to 11, 2020. The predictor network was trained for 250 epochs, with a learning rate of 0.004, RMSE as the loss function, and a seed generator value of 45.

Similarly to the previous experiment, to find the best model, we varied the seed generator value for the ELSA network to leverage the impact of initial weight generation. The set $S = \{0, 1, 2, \dots, 198, 199\}$, represented in the last row of Table 6.3, shows the values used. We trained 200 ELSA models linked to an identical predictor network described in Table 6.4. During the training of each model, we stored the RMSE values for the test dataset and the final prediction period (September 12 to 18). From these values, we selected the seed generators that obtained the lowest RMSE simultaneously for both datasets. Finally, we averaged the two RMSEs for these models and chose the one that obtained the lowest average.

The second work we selected was the article “Optimal Neural Network Model for Short-Term Prediction of Confirmed Cases in the COVID-19 Pandemic” (Milić et al. 2022) presents a approach utilizing Time Delay Neural Networks (TDNN) for forecasting COVID-19 cases. The model leverages a window size of nine previous observations to predict the immediate next value, employing a shifting mechanism for each new prediction. The study uses time series data from 2021 for six countries: Chile, the Netherlands, Spain, Canada, South Africa, and Malaysia. The data undergoes preprocessing steps including shifting, normalizing, and smoothing. The TDNN configurations, with hidden layer neurons ranging from 3 to 10, are trained using a variation of the steepest-descent minimization algorithm. The training process aims to minimize prediction error, with the sigmoid activation function for hidden layers and a linear activation function for the output layer. Evaluation metrics such as Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and the coefficient of determination (R^2) demonstrate the model’s accuracy and reliability in short-term forecasting. The results indicate that the proposed TDNN-based model effectively predicts the trends of confirmed COVID-19 cases across different datasets, enhancing the accuracy of short-term predictions through the averaging of multiple TDNN configurations. The authors acquired the the proposed model RMSE final results from the forecast of all training dataset. Those final results were 88.67, 253.80, 393.74, 786.68, 429.43, 192.30 for Chile, the Netherlands, Spain, Canada, South Africa, and Malaysia, respectively. This work was selected to point out our model concerning to a multi-variable/multi-feature forecasting.

To ensure a fair and accurate comparison with the methodology used in the referenced study, we reformulated the dataset to match exactly. We assembled a training set from January 1, 2021, to December 31, 2021, as proposed. This dataset was not scaled, only pass through a mean average of 7 days window, as in the proposed comparison. Table 6.5 presents the hyperparameter values used for training the ELSA network.

From Table 6.5, we note that no hidden layers were used within the internal autoencoders of the ELSA network. We used 20 neurons as the final layers in the internal autoencoders for each of the 3 autoencoders, resulting in an expanded latent space of 60

Hyperparameter	Value
Input Size	9
Size of latent space in each AE	20
Number of internal autoencoders	3
Size of expanded latent space	60 (20 * 3)
Training epochs	60
Learning rate	0.004
Loss function	RMSE
Seed generator	$S = \{0, 1, 2, \dots, 198, 199\}$

Table 6.5: Table of hyperparameters used in the comparison ELSA network.

values. The network was trained for 60 epochs, with a learning rate of 0.004 and RMSE as the loss function, using an input window of 9 samples. All these values were defined based on the comparison work, the Chapter 5 validations and our expertise with the proposed model. It is important to note that for this experiment, the model is multivariate, unlike Comparison 1, which was not. Following our model's proposal, we also trained a predictor MLP network. Table 6.6 presents the hyperparameter values used for training the predictor network.

Predictor Hyperparameter	Value
Input size for forecasting	60
Hidden layer size	60
Forecasting layer window size	1
Training epochs	1000
Input features quantity	6
Internal predictors quantity	6
Learning rate	0.0003
Loss function	RMSE
Seed generator	45

Table 6.6: Table of hyperparameters used for the predictor network.

From Table 6.6, we see that the predictor network comprises a six-feature predictor, which each one has an input size matching the expanded latent space, followed by a hidden layer of size 60, and finally, the forecasting layer with the desired forecast window size of 1. The predictor network was trained for 1000 epochs, with a learning rate of 0.0003, RMSE as the loss function, and a seed generator value of 45. This high value of epochs and lower to learning rate was used due multi-feature forecasting nature of the model, once we are make predictions of six countries, so the training step has to carefully change the model parameters to correctly adjust them to fit all predicting time-series at the same time.

Similarly to the previous experiment, to find the best model, we varied the seed gener-

ator value for the ELSA network to leverage the impact of initial weight generation. The set $S = \{0, 1, 2, \dots, 198, 199\}$, represented in the last row of Table 6.5, shows the values used. We trained 200 ELSA models linked to an identical predictor network described in Table 6.6. During the training of each model, we stored the RMSE values for the training dataset of each country, where this RMSE was calculated without any adjustment of gradients on this specific calculation step. From these values, we calculate the mean of the countries forecasting and selected the seed generator that obtained the lowest mean RMSE.

Chapter 7

Results and Analysis

This chapter presents the results obtained from the experiments described in the previous chapter, which aim to test the information absorption capacity of the proposed model in the context of predicting COVID-19 related time series. Initially, we subjected the proposed structure to an evaluation of its ability to make short, medium, and long-term predictions. Subsequently, through a comparative experiment, we compare the proposed model within methodologies employed in works found in the literature, aiming to position our work within the state of the art among models who use feature extraction networks together with prediction networks, and models that predict multiple time series simultaneously.

7.1 Results from Short-medium-long-term Forecasting

In the first described and executed experiment, we subjected the proposed methodology and models to an evaluation to determine the forecasting scenario in which they best fit. The scenarios are observed from the perspective of prediction window size. We chose prediction windows of 7, 14, 28, 35, 49, and 56 days for this evaluation. The first two represent short-term predictions, while 28 and 35, 49, and 56 represent medium and long-term predictions, respectively. Figure 7.1 presents the distribution values obtained from the experiment in a boxplot format.

From Figure 7.1, it is visually showed that models with an observation window of 7 samples obtained the lowest RMSE metric values for the test set. Following this, the distribution obtained from the 14-sample window indicates that the proposed model is notably better for short-term predictions, as expected. For medium-term prediction windows, the 28-sample window had a distribution similar to the 14-sample window, while the 35-sample window had a distribution considerably different from the previous. For long-term prediction windows, both demonstrated increasing values in their distributions as the observation window size increased, which is undesirable. It is important to recall that the metric was extracted for 56 samples for all tested models, with the only difference being the number of predictions each model had to make relative to the observation window size. Table 7.1 presents the distribution values quantitatively in terms of median, mean, and interquartile range.

From Table 7.1, it can be observed that the distribution obtained from the 7-sample

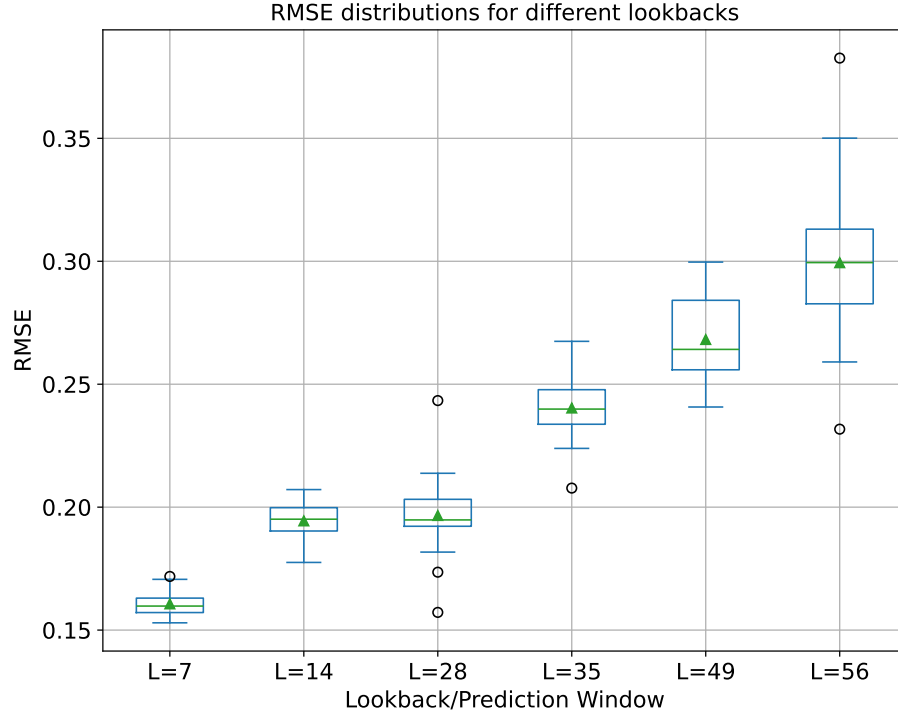


Figure 7.1: Boxplot representation of the distributions generated from the extraction of RMSE from the test dataset for the short-medium-long-term forecasting experiment.

L	Median	Mean	Interquartile Range
7	0.160	0.161	0.006
14	0.195	0.194	0.009
28	0.195	0.197	0.011
35	0.240	0.240	0.014
49	0.264	0.268	0.028
56	0.299	0.299	0.030

Table 7.1: Table of quantitative values obtained from the distributions generated in the short, medium, and long-term prediction evaluation experiment.

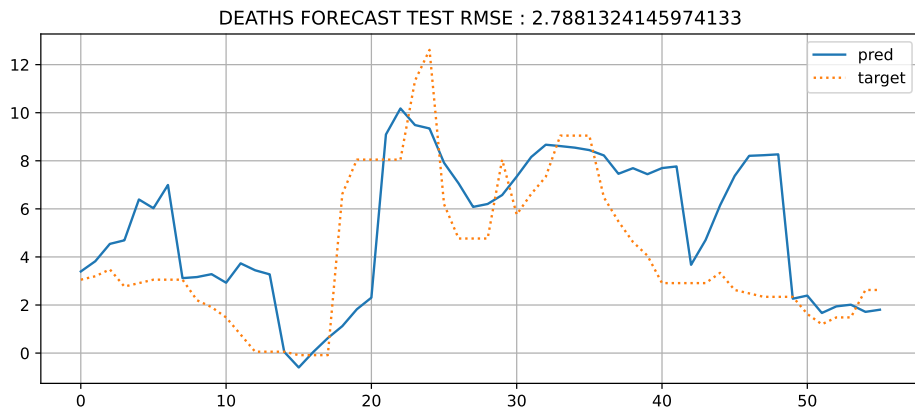
observation window had the lowest median, mean, and interquartile range values among all distributions. The distributions obtained from the 14 and 28-sample windows have relatively small differences between their values, being almost identical in median and mean with a small difference in interquartile range. The rest of the distributions exhibit a consistently increasing trend, indicating that the proposed model does not perform better when making longer-term predictions, even with a larger observation window. This behavior can be explained by the accumulated error for predicting a larger output window. Finally, it can be concluded that the proposed model performs better for short-term and medium-term predictions up to one month (or 28 days). It is also important to note that the values presented in Table 7.1 were obtained from normalized series. Table 7.2 shows the non-normalized values obtained for the average RMSE of the models trained individually for each input series.

L	humidity	temperature	aqi	cases	deaths
7	2.261	0.689	5.506	88.535	2.824
14	2.759	0.771	6.689	80.22	5.48
28	<i>2.15</i>	1.077	<i>4.714</i>	175.74	5.249
35	3.151	1.142	6.252	225.018	4.954
49	3.421	1.632	5.494	146.095	8.522
56	4.235	1.006	6.059	453.916	9.011

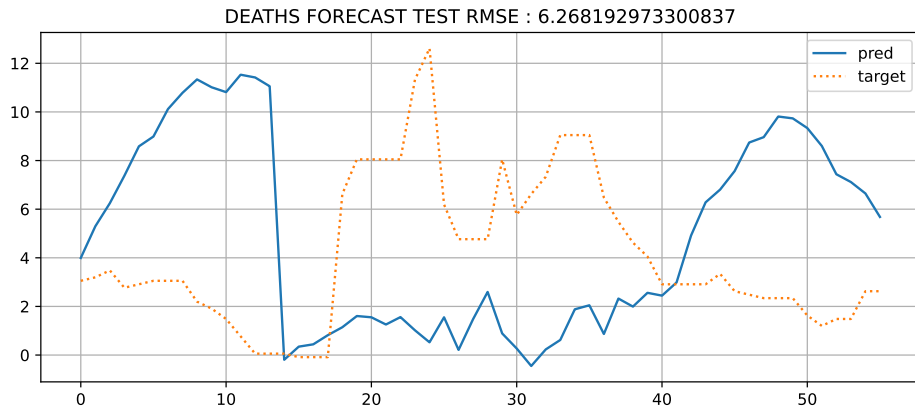
Table 7.2: Table of quantitative values obtained from the average RMSE for each input series in the short, medium, and long-term prediction evaluation experiment.

From Table 7.2, we can see that the models trained for the $L = 28$ window had lower RMSE values for the humidity, and AQI series. For predicting COVID-19 cases, the prediction window with the lowest error was $L = 14$, highlighted in italic format. Finally, the $L = 7$ window, in addition to having the best average performance for predicting all series (as shown in in Table 7.1) as previously observed, also has the best individual RMSE for predicting temperature, and the number of deaths. The number of deaths has more weight to this observation, once it can be considered the most relevant indicator of the pandemic spread behavior to society. The other windows did not show any other relevant metric values. To conclude this experiment, we used the best models for each of the distribution windows and plotted their predictions for the test set related to the death series. Figure 7.2 visually shows the plot of the original and predicted series for deaths in the short-term prediction experiments.

From Figure 7.2, we can visually observe that the $L = 7$ window achieved a better approximation of the predicted values compared to the $L = 14$ window. This observation strengthens the numerically observed conclusions. It is also important to recall that the prediction obtained from the model generated with the shown seeds had the lowest error for the average prediction of all series in the dataset. In other words, the relatively high RMSE value and poor visual approximation for the $L = 14$ window, even for the “best mode”, can be explained because the seed resulted in the best model for the average prediction of all series and not the best for the number of deaths in isolation. Another



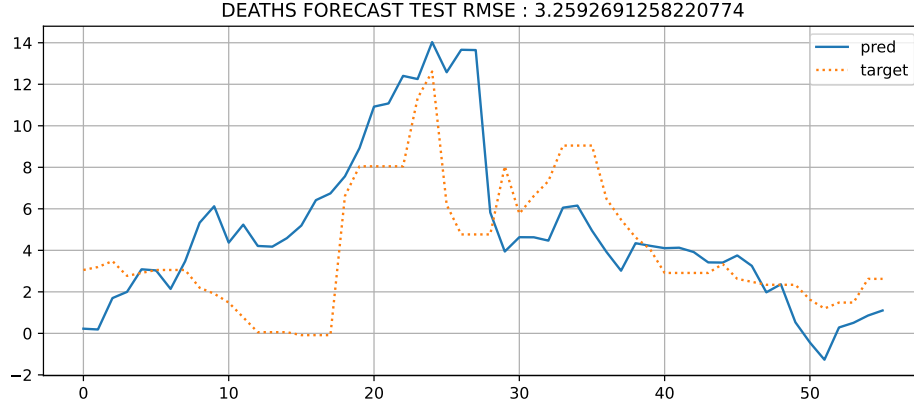
(a) Plot for the best model trained with the $L = 7$ window using seed $S = 45$.



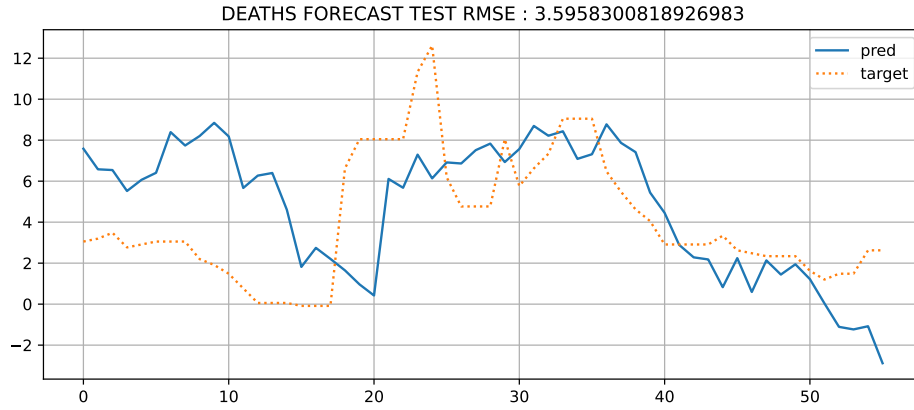
(b) Plot for the best model trained with the $L = 14$ window using seed $S = 35$.

Figure 7.2: Plot of the prediction for the time series related to the number of deaths in São Paulo - Brazil from 2023-01-01 to 2023-02-25.

pertinent observation is the possible negative values for the series samples. This occurs because, during data processing, we first normalized the series before applying the moving average filter. Figure 7.3 visually shows the plot of the original and predicted series for deaths in the medium-term prediction experiments.



(a) Plot for the best model trained with the $L = 28$ window using seed $S = 50$.

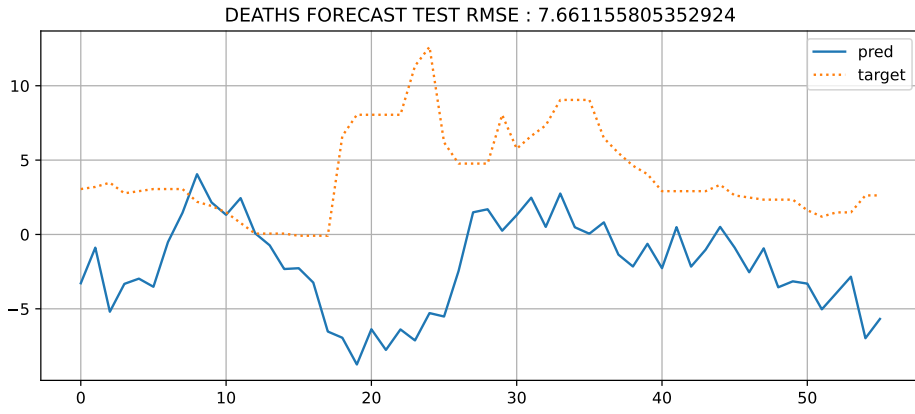


(b) Plot for the best model trained with the $L = 35$ window using seed $S = 40$.

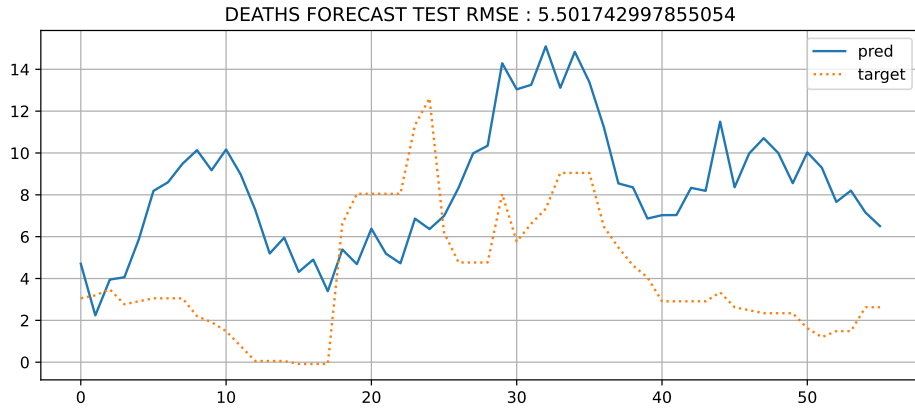
Figure 7.3: Plot of the prediction for the time series related to the number of deaths in São Paulo - Brazil from 2023-01-01 to 2023-02-25.

From Figure 7.3, we can visually observe that the $L = 28$ window achieved a slightly closer approximation to the predicted and actual values compared to the $L = 35$ window, with RMSE values being very close. In fact, both approximations are superior to the $L = 14$ short-term prediction model, both visually and numerically. This can also be explained by the previously made observation, where the seed resulting in the best model for the average prediction of all series is not necessarily the best for the number of deaths in isolation. Finally, Figure 7.4 visually shows the plot of the original and predicted series for deaths in the long-term prediction experiments.

From Figure 7.4, we can visually observe that both long-term prediction windows



(a) Plot for the best model trained with the $L = 49$ window using seed $S = 30$.



(b) Plot for the best model trained with the $L = 56$ window using seed $S = 55$.

Figure 7.4: Plot of the prediction for the time series related to the number of deaths in São Paulo - Brazil from 2023-01-01 to 2023-02-25.

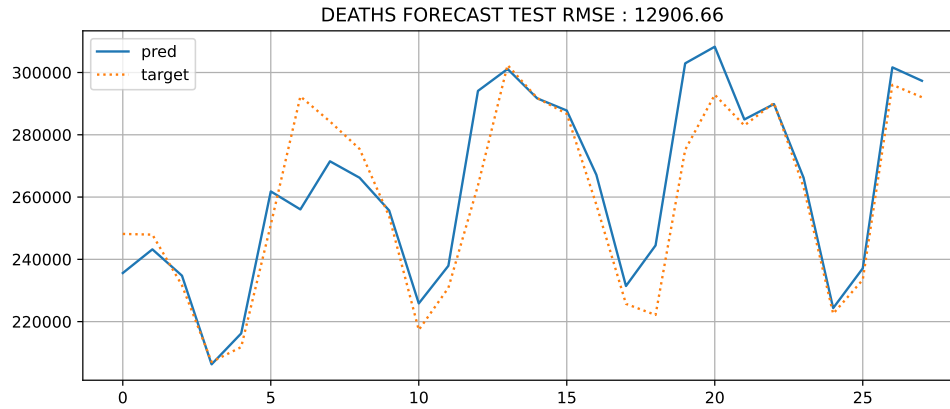
showed difficulty in following a possible abrupt increase and decrease in values. This can be better seen between samples 20 and 30, where there is a peak in the target series, and both models were unable to follow this peak. This can be explained by the characteristic of the model not being suitable for long-term predictions, where ELSA networks are not capable of extracting sufficiently good features that preserve information for the predictor to execute an accurate prediction. This behavior also strengthens the conclusions obtained from the numerical analysis at the beginning of this experiment. Finally, the proposed model, composed of ELSA networks and a simple MLP predictor, demonstrates better capacity for predicting COVID-19 related time series in a short-term scenario.

7.2 Results from Works Comparison

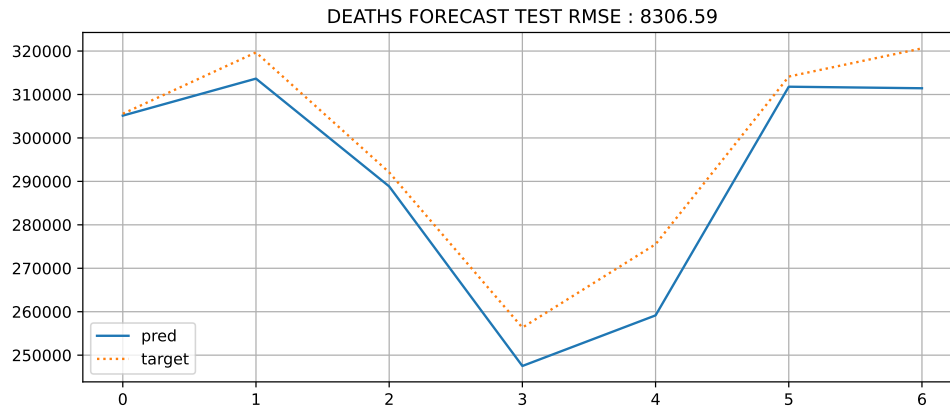
For the second described and executed experiment, we subjected the proposed methodology and models to a comparative evaluation against works present in the literature. The first work we used for comparison (Zain & Alturki 2021) employs a relatively similar approach to the one proposed here. The methodology uses a convolutional network to extract information from the inputs and subsequently employs an LSTM network for prediction. We used the same datasets for training and testing to extract evaluative metrics from our model, with the objective of directly comparing it with those of other authors. Based on the methodology previously explained in Chapter 6, we extracted the results of the model that presented the lowest average RMSE for the two datasets used to evaluate the proposed models in the comparison work. Figure 7.5 visually shows the plot of the original and predicted series for the cases in the first comparative experiment.

From Figure 7.5, we can visually observe that the series predicted by our proposed model for the set of 28 samples from 2020-07-18 to 2020-08-14 closely approximates the real series. Notably, the only instance where the predicted values did not follow the real values was between samples 5 and 10, where the predicted series failed to reach the same peak as the real series. Quantitatively, the prediction achieved an RMSE of 12906.66 compared to the real series, which surpasses the value of 13275.00 obtained in the comparison work. For the set of 7 samples from 2020-09-12 to 2020-09-18, as highlighted in Chapter 6, we did not use the same 4-week input as in the comparison study, but only the previous 7 days to predict the following 7 days. It is visually observable that the predicted values closely follow the curve generated by the real values. Quantitatively, the prediction achieved an RMSE of 8306.59 compared to the real series, which surpasses the value of 8780.71 obtained in the comparison work. These values correspond to an improvement of 2.77% and 5.39%, respectively, for the datasets. This experiment, in addition to demonstrating the predictive capability of the proposed model, extends the potential for univariate predictions. Another point to be noted is that our model only uses MLP networks in its operation, not employing more complex networks, which is an advantage suggesting that the methodology employed is efficient and independent of the network's capacity and complexity.

The second work we used for comparison (Milić et al. 2022) employs a not similar approach to the one proposed here. The methodology uses a approach with Time Delay Neural Networks (TDNN) for forecasting COVID-19 cases. We used the same



(a) Plot for the best model trained with a window size of $L = 7$ and a seed value of $S = 125$ using the test set to predict from 2020-07-18 to 2020-08-14.



(b) Plot for the best model trained with a window size of $L = 7$ and a seed value of $S = 125$ using the test set to predict from 2020-09-12 to 2020-09-18.

Figure 7.5: Prediction plot for the time series related to the number of global COVID-19 cases.

datasets for training and evaluate our model, with the objective of directly comparing it with those of other authors. Based on the methodology previously explained in Chapter 6, we extracted the results of the model that presented the lowest average RMSE for the six-country forecasting used at the proposed models in the comparison work. Figure 7.6 visually shows the plot of the original and predicted series for each country in the second comparative experiment.

From Figure 7.6, we can visually observe that the series predicted by our proposed model for the set of samples from 2021-01-01 to 2021-12-31 closely approximates the real series. Notably, there are small sample intervals where the predicted values deviate slightly from the actual values. These differences are visually more pronounced at steeper peaks and valleys in the time series. Quantitatively, the prediction achieved an RMSE of 62.59, 187.35, 244.78, 515.75, 390.11, 143.49 for Chile, the Netherlands, Spain, Canada, South Africa, and Malaysia, respectively. Table 7.3 shows the percentage improvement to the RMSE obtained by our proposed model for each country.

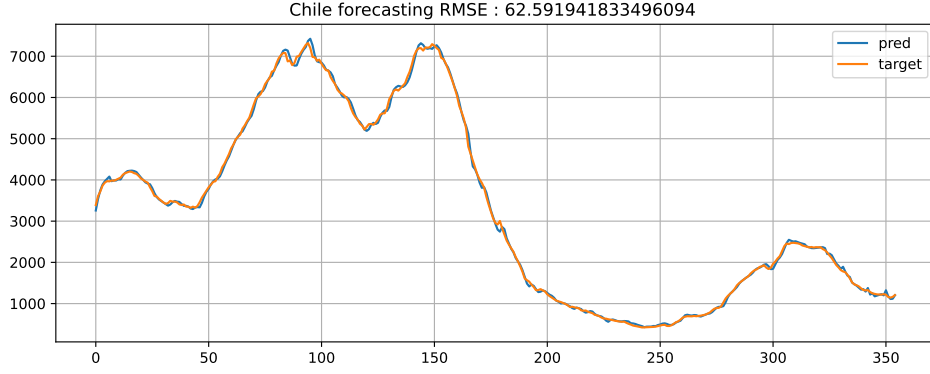
Country	Our model	Comparison Model	Percentage Improvement
Chile	62.59	88.67	29.41%
Netherlands	187.35	253.8	26.18%
Canada	244.78	393.74	37.83%
Spain	515.75	786.68	34.44%
South Africa	390.12	429.43	9.15%
Malaysia	143.50	192.3	25.38%

Table 7.3: Table of quantitative values obtained from the RMSE for each input country series in the prediction evaluation comparison experiment.

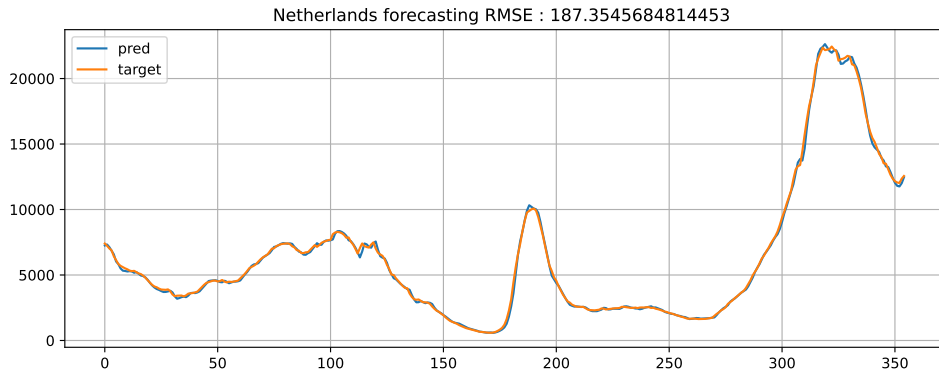
From Table 7.3 we have the RMSE forecasting values acquired by our proposed model for each comparison country time-series. Those values surpasses the values obtained in the comparison work. From "Percentage Improvement" column, the most improvement value is highlighted in italic, which was 37.83% for Canada. Others high improvements were acquired for Spain and Chile, 34.44% and 29.41%, respectively. Our proposed model was able to obtained a 27% of improvement average. This experiment, in addition to demonstrating the predictive capability of the proposed model using the possible minimal short-term forecasting (1 sample), reaffirms the proposed model potential for multi-variable predictions. Finally, it is important highlight the predictor model follows the same architecture of internal networks modules to each input time-series separately, as previous described at Chapter 4 and Chapter 5.

7.3 Discussion

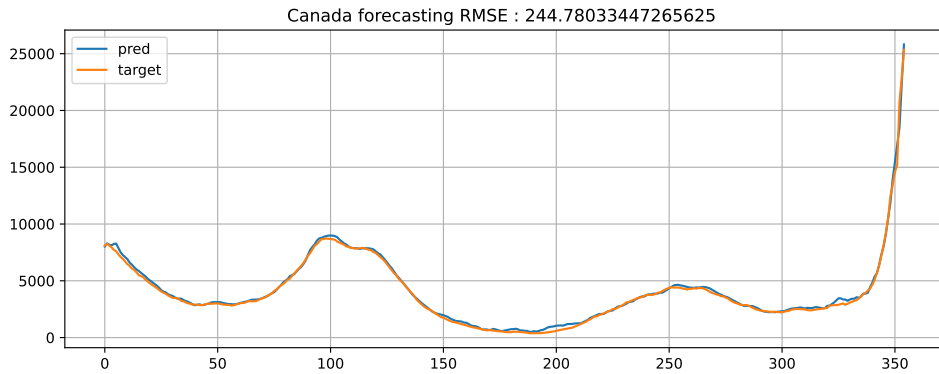
The experiments and results obtained with the proposed model demonstrate its value in the field of time series forecasting, especially in scenarios requiring resource scalability. However, it is crucial to acknowledge that these are mathematical models, representing



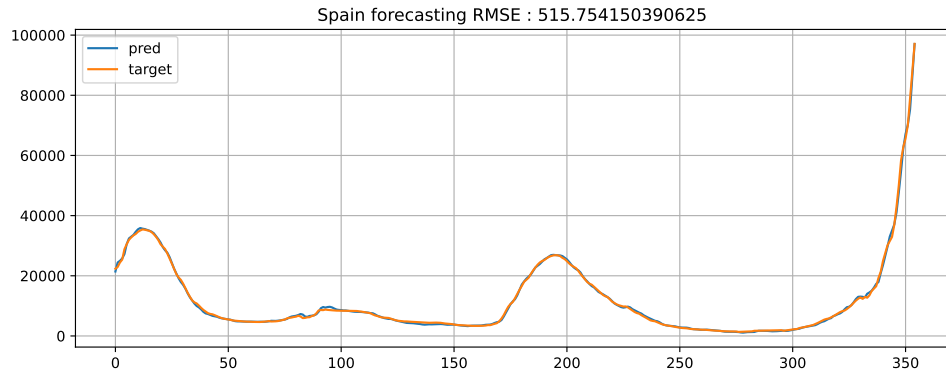
(a) Plot for the best model trained with a window size of $L = 9$ for Chile one day forecasting. The seed value $S = 170$ was used in the train set to predict from 2021-01-01 to 2021-12-31.



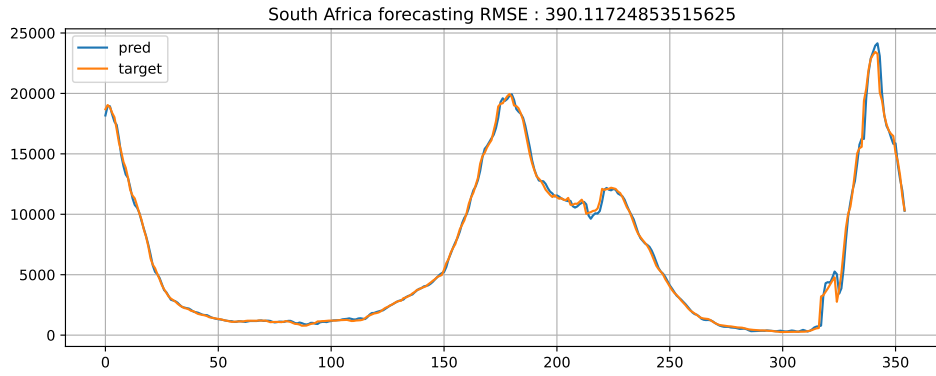
(b) Plot for the best model trained with a window size of $L = 9$ for Netherlands one day forecasting. The seed value $S = 170$ was used in the train set to predict from 2021-01-01 to 2021-12-31.



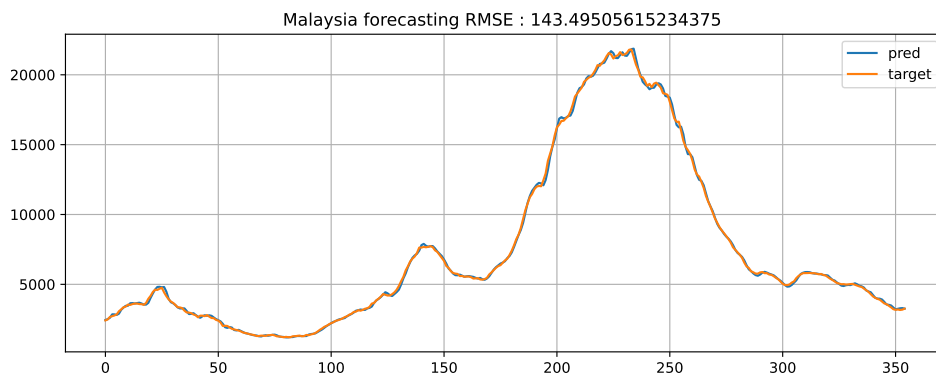
(c) Plot for the best model trained with a window size of $L = 9$ for Canada one day forecasting. The seed value $S = 170$ was used in the train set to predict from 2021-01-01 to 2021-12-31.



(d) Plot for the best model trained with a window size of $L = 9$ for Spain one day forecasting. The seed value $S = 170$ was used in the train set to predict from 2021-01-01 to 2021-12-31.



(e) Plot for the best model trained with a window size of $L = 9$ for South Africa one day forecasting. The seed value $S = 170$ was used in the train set to predict from 2021-01-01 to 2021-12-31.



(f) Plot for the best model trained with a window size of $L = 9$ for Malaysia one day forecasting. The seed value $S = 170$ was used in the train set to predict from 2021-01-01 to 2021-12-31.

Figure 7.6: Prediction plot for each country time series related to the number of COVID-19 cases.

approximations rather than exact solutions to complex problems like the COVID-19 pandemic. The issue at hand is multifaceted, affecting various systems such as healthcare, politics, and economics over time. Consequently, extensive work is necessary to gather substantial features that can be used as input data for the proposed model. To do this effectively, it is essential to fully engage with the environment in which the model will be applied, understand how the training data was collected, and, in many cases, incorporate domain knowledge when available data is insufficient.

Moreover, improving prediction accuracy requires minimizing noise interference that could hinder the model's generalization capabilities. The model consists of two primary stages during which noise may be introduced. The first involves concatenating latent spaces to generate an expanded latent space used as input for the predictive network. The second stage includes not only reconstructing the input information but also averaging the sequences generated in the reconstruction of each autoencoder, which introduces additional noise. Consequently, there is an increase in error in the second stage, both in the integrated establishment of expanded latent spaces and in the reconstruction of input data. Additionally, while adding neurons in the hidden layers of the internal autoencoders and increasing the number of these autoencoders can enhance the model's capabilities, these adjustments are only effective to a certain extent. This was briefly explored and analyzed during the validation of the proposed architecture.

Overall, the proposed architecture, composed of a network responsible for feature extraction and another for prediction, introduces a substantial number of hyperparameters that must be managed, in addition to the accumulated error. However, this is a common challenge across most artificial neural network-based models. There are methodologies for hyperparameter tuning available in the literature and implemented in well-known training frameworks within the community. A complicating factor in the proposed model is the presence of N stacked autoencoders, which can significantly increase complexity depending on the chosen value for N , the number of hidden layers, and their size (L). Furthermore, the initial conception of the proposed architecture, as outlined by Pereira et al. (2020), was intended for univariate prediction and designed to extract as much information as possible from smaller datasets. In applications and case studies involving larger datasets, this can become problematic, as the use of GPUs for processing may not be worth over CPU processing, leading to longer training times. However, this characteristic also offers an advantage; the model is designed to extract more information from small datasets, allowing for potentially reduced training times by not using an entire dataset for model training, as observed in the conducted validations.

Regarding the comparative experiments, the main challenges encountered were also related to model fine-tuning. Even when training a set of models with variations only in their initial random seed, it was necessary to pre-train some models to define the set of hyperparameters, which was time-consuming. As mentioned earlier, varying some hyperparameters could significantly increase the training time, making it impractical to conduct a more in-depth analysis of the models' response to these variations. Therefore, for the comparative experiments, we always started with a set of known hyperparameters based on our experience from the validations and short-medium-long-term prediction experiments. We then made small adjustments to key hyperparameters, observed whether there

was a significant improvement in predictions, and, if so, determined whether we were in a favorable gradient region to take advantage of variations in the initial seeds. From there, we conducted the search for the seed that produced the best results, as described in the experiments.

In summary, this is our outlined strengths and limitations of the proposed model, emphasizing its applicability to time series forecasting under various conditions. While the model's structure, consisting of feature extraction and predictive components, introduces complexities related to hyperparameter management and computational demands, it also offers distinct advantages in handling smaller datasets effectively. The challenges encountered, particularly in fine-tuning and noise management, highlight areas for careful consideration in practical applications. This analysis provides a nuanced understanding of the model's capabilities and potential pitfalls, setting the stage for a deeper exploration of its utility in diverse scenarios.

Chapter 8

Conclusion

In the event of a global pandemic, the scientific community plays an essential role in supporting decision-makers with data-driven insights. Researchers can analyze various factors, including environmental conditions that may facilitate the spread of the virus, as well as variables that influence the number of cases and fatalities. They also develop predictive models that observe trends in pandemic-related data to forecast changes in key indicators.

Accurate forecasting of COVID-19 cases and deaths is crucial for public health planning and response. Reliable predictions enable authorities to anticipate healthcare needs, allocate resources effectively, and implement timely interventions to curb the spread of the virus. Additionally, precise forecasts can guide policy decisions, such as the imposition or relaxation of lockdown measures, ensuring that actions taken are proportional to the projected severity of the outbreak. In this context, the methodology proposed in this thesis aims to provide more accurate and actionable insights to future pandemics, ultimately supporting efforts to save lives and minimize societal disruption.

This thesis presents a new approach using a multi-feature stacked Auto-Encoder-based methodology designed for the time-series forecasting of COVID-19 cases and deaths. This approach leverages the power of autoencoders to learn complex, non-linear representations of the input data, capturing essential features that may influence the trajectory of the pandemic. By stacking multiple autoencoders, the model is able to progressively refine these representations, ultimately enhancing its predictive accuracy. This methodology integrates various features, such as epidemiological data and environmental factors, to provide a comprehensive analysis and forecast, aiming to offer precise and actionable insights for public health decision-making.

To validate the proposed model, we initially subjected it to several validations to ensure its coherence in application. It was confirmed that, according to the artificial intelligence and machine learning frameworks utilized, two autoencoders generated sequentially with the same seed have different initial parameters, resulting in distinct autoencoders after training. Subsequently, the reproducibility of the model was demonstrated, along with the assurance of distinct distribution behaviors for different datasets (features). Furthermore, the advantage of using a stacked autoencoder architecture over a single autoencoder was highlighted. We also evaluated the optimal number of internal autoencoders that provides the best cost-benefit ratio in terms of the size of each autoencoder and the latent space generated. To assess the proposed architecture, we conducted an

experiment comparing the performance of networks trained on different input sizes in a distributed manner. This experiment aimed to evaluate the proposed model in the task of multivariate prediction and to determine its effectiveness in short, medium, and long-term predictions. Finally, we directly compared the proposed architecture with two studies found in the scientific literature, positioning the proposal among works that have undergone expert scrutiny. In these comparative experiments, we achieved improvements of 2.77% and 5.39%, respectively, for predicting the datasets presented in the first compared work. Additionally, we obtained an average of 27% improvement in prediction accuracy using the datasets and time series proposed in the second comparison work.

The discussion of this work highlighted the proposed model's capability in time series forecasting, where the proposed architecture, which leveraging feature extraction and prediction through stacked autoencoders, demonstrated both strengths and challenges. While it effectively handles small datasets, the model's complexity necessitates careful management of hyperparameters and noise. The discussion also addressed the introduction of errors during latent space generation and data reconstruction. Additionally, the scalability and practical implications of computational demands were considered, especially when expanding the model's architecture. This comprehensive evaluation provides valuable insights into the model's performance and sets a foundation for future improvements and applications.

The main contributions of the research presented in this thesis include, theoretically, the proposal of an autoencoder-based artificial neural network architecture potentially applicable to various problems in the literature, such as prediction, classification, and embedding. Indirectly, a superficial analysis of how the latent spaces of stacked autoencoders function was also conducted, based on mean distributions. Practically, this work contributes by providing a network architecture capable of making accurate time series predictions related to COVID-19, which can assist in providing insights for public health decision-making in the event of future epidemics or pandemics. By extension, the proposed model can be applied in other fields beyond time series prediction, as shown in the validation experiments.

These findings encourage further exploration for improvements, such as incorporating various types of neural network neurons and increasing the number of layers in the autoencoder. As previously discussed, the proposed architecture primarily utilized simple MLP (Multilayer Perceptron) neural networks to maintain simplicity and accurately assess the impact of an expanded latent space on prediction tasks. However, since the proposed architecture is not limited to this type of network, future work could explore the use of more complex neural networks, which may require greater computational resources, to determine if there is a worthwhile trade-off between complexity and performance. Additionally, other types of layers, such as dropout and normalization layers, could be incorporated into the architecture. Moreover, the development of a variational architecture is also considered feasible; however, based on our experience with the training processes, the computational cost, which affects training time, may not justify its use compared to other well-established networks and architectures in the scientific community. Finally, future research should investigate the limitations, complexity, and scalability of the proposed model.

Finally, we note that this work is part of a broader collaborative effort at the Natalnet Associate Laboratories of UFRN, where various aspects of the project are being explored. These include identifying the most effective machine learning approaches, determining the optimal database solutions for efficient data access by researchers, and making the system available to managers and the public. This availability is particularly important for addressing future pandemics, managing ongoing endemics such as dengue, and other public health challenges, including monitoring seasonal flu trends, tracking the spread of emerging infectious diseases, and optimizing vaccination distribution strategies. Additionally, the system can be applied to other sectors, such as environmental monitoring, disaster response, and urban planning, where predictive analytics can inform critical decision-making. As a final remark, we would like to note that our team has also contributed to other works, which have been recognized in the literature and have added to our academic contributions (de Oliveira, Aragão & Gonçalves 2024, Aragão et al. 2022, Pereira et al. 2022, de Oliveira, Aragão & Gonçalves 2024, Oliveira et al. 2022)

Bibliography

- Aabed, Kawther & Maha MA Lashin (2021), 'An analytical study of the factors that influence covid-19 spread', *Saudi Journal of Biological Sciences* **28**(2), 1177–1195.
- Alifiah, Nur, Dian Kurniasari, Amanto Amanto & Warsono Warsono (2023), 'Prediction of covid-19 using the artificial neural network (ann) with k-fold cross-validation', *Journal of Information Systems Engineering and Business Intelligence* **9**(1), 16–27.
- Anastassopoulou, Cleo, Lucia Russo, Athanasios Tsakris & Constantinos Siettos (2020), 'Data-based analysis, modelling and forecasting of the covid-19 outbreak', *PloS one* **15**(3), e0230405.
- Aragão, Dunfey Pires, Davi Henrique Dos Santos, Adriano Mondini & Luiz Marcos Garcia Gonçalves (2021), 'National holidays and social mobility behaviors: Alternatives for forecasting covid-19 deaths in brazil', *International Journal of Environmental Research and Public Health* **18**(21), 11595.
- Aragão, Dunfey P., Emerson V. Oliveira, Arthur A. Bezerra, Davi H. dos Santos, Andouglas G. da Silva Junior, Igor G. Pereira, Prisco Piscitelli, Alessandro Miani, Cosimo Distanto, Jordan S. Cuno, Aura Conci & Luiz M.G. Gonçalves (2022), 'Multivariate data-driven prediction of covid-19 dynamics: Towards new results with temperature, humidity and air quality data', *Environmental Research* **204**, 112348.
- Aslam, Humaira & Santanu Biswas (2023), 'Analysis of covid-19 death cases using machine learning', *SN Computer Science* **4**(4), 403.
- Bishop, Christopher M (2006), 'Pattern recognition and machine learning', *Springer google schola* **2**, 1122–1128.
- Borghi, Pedro Henrique, Oleksandr Zakordonets & João Paulo Teixeira (2021), 'A covid-19 time series forecasting model based on mlp ann', *Procedia Computer Science* **181**, 940–947.
- Car, Zlatan, Sandi Baressi Šegota, Nikola Anđelić, Ivan Lorencin & Vedran Mrzljak (2020), 'Modeling the spread of covid-19 infection using a multilayer perceptron', *Computational and mathematical methods in medicine* **2020**.
- Chimmula, Vinay Kumar Reddy & Lei Zhang (2020), 'Time series forecasting of covid-19 transmission in canada using lstm networks', *Chaos, Solitons & Fractals* **135**, 109864.

- Chwila, Adam (2023), 'The prediction of new covid-19 cases in poland with machine learning models', *Statistics in Transition. New Series* **24**(2), 59–83.
- COCCIA, Mario (2020), 'How do environmental, demographic, and geographical factors influence the spread of covid-19', *Journal of Social and Administrative Sciences* **7**(3), 169–209.
- Cooper, Ian, Argha Mondal & Chris G Antonopoulos (2020), 'A sir model assumption for the spread of covid-19 in different communities', *Chaos, Solitons & Fractals* **139**, 110057.
- Coppin, B. (2004), *Artificial Intelligence Illuminated*, Jones and Bartlett illuminated series, Jones and Bartlett Publishers.
URL: <https://books.google.com.br/books?id=LcOLqodW28EC>
- Cui, Jie, Fang Li & Zheng-Li Shi (2019), 'Origin and evolution of pathogenic coronaviruses', *Nature reviews microbiology* **17**(3), 181–192.
- Cyranoski, David (2021), 'Alarming covid variants show vital role of genomic surveillance.', *Nature* **589**(7842), 337–339.
- de Oliveira, Emerson Vilar, Dunfrey Pires Aragão & Luiz Marcos Garcia Gonçalves (2024), 'Elsa: Expanded latent space autoencoder for image feature extraction and classification', *Proceedings Copyright* **703**, 710.
- de Oliveira, Emerson Vilar, Dunfrey Pires Aragão & Luiz Marcos Garcia Gonçalves (2024), 'A new auto-regressive multi-variable modified auto-encoder for multivariate time-series prediction: A case study with application to covid-19 pandemics', *International Journal of Environmental Research and Public Health* **21**(4).
URL: <https://www.mdpi.com/1660-4601/21/4/497>
- Djaparidze, Levan & Federico Andres Lois (2020), 'Sars-cov-2 waves in europe: A 2-stratum seirs model solution', *medRxiv*.
- Drosten, Christian, Stephan Günther, Wolfgang Preiser, Sylvie Van Der Werf, Hans-Reinhard Brodt, Stephan Becker, Holger Rabenau, Marcus Panning, Larissa Kolesnikova, Ron AM Fouchier et al. (2003), 'Identification of a novel coronavirus in patients with severe acute respiratory syndrome', *New England journal of medicine* **348**(20), 1967–1976.
- ECDC (n.d.), 'Non-pharmaceutical interventions against covid-19', Available at <https://www.ecdc.europa.eu/en/covid-19/prevention-and-control/non-pharmaceutical-interventions> (2023/03/01).
- Elsaid, Ashraf Mimi & M Salem Ahmed (2021), 'Indoor air quality strategies for air-conditioning and ventilation systems with the spread of the global coronavirus (covid-19) epidemic: Improvements and recommendations', *Environmental Research* p. 111314.

- Fernandes, João C, GBD 2017 Causes of Death Collaborators et al. (2018), 'Erratum: Global, regional, and national age-sex-specific mortality for 282 causes of death in 195 countries and territories, 1980–2017: a systematic analysis for the global burden of disease study 2017 (the lancet (2018) 392 (10159)(1736–1788)(s0140673618322037)(10.1016/s0140-6736 (18) 32203-7))', *The Lancet* **392**(10160), 2170–2170.
- Freitas, Edmilson D, Sergio A Ibarra-Espinosa, Mario E Gavidia-Calderón, Amanda Rehbein, Sameh A Abou Rafee, Jorge A Martins, Leila D Martins, Ubiratan P Santos, Mariangeli F Ning, Maria F Andrade et al. (2020), 'Mobility restrictions and air quality under covid-19 pandemic in são paulo, brazil'.
- Gallinari, Patrick, Yann Lecun, Sylvie Thiria & F Fogelman Soulie (1987), Mémoires associatives distribuées: une comparaison (distributed associative memories: a comparison), *em* 'Proceedings of COGNITIVA 87, Paris, La Villette, May 1987', Cesta-Afcet.
- Goodfellow, Ian, Yoshua Bengio, Aaron Courville & Yoshua Bengio (2016), *Deep learning*, Vol. 1, MIT press Cambridge.
- Güngör, Murat (2023), 'Time series forecasting of the covid-19 pandemic: a critical assessment in retrospect', *Alphanumeric Journal* **11**(1), 85–100.
- Haviluddin, Haviluddin & Rayner Alfred (2023), 'Multi-step cnn forecasting for covid-19 multivariate time-series', *International Journal of Advances in Intelligent Informatics* **9**(2), 176–186.
- Hawas, Mohamed (2020), 'Generated time-series prediction data of covid-19's daily infections in brazil by using recurrent neural networks', *Data in brief* **32**, 106175.
- Haykin, Simon (2007), *Neural networks: a comprehensive foundation*, Prentice-Hall, Inc.
- He, Shaobo, Yuexi Peng & Kehui Sun (2020), 'Seir modeling of the covid-19 and its dynamics', *Nonlinear dynamics* **101**, 1667–1680.
- Ibrahim, Mohamed R, James Haworth, Aldo Lipani, Nilufer Aslam, Tao Cheng & Nicola Christie (2021), 'Variational-lstm autoencoder to forecast the spread of coronavirus across the globe', *PloS one* **16**(1), e0246120.
- Ismail, Iqbal MI, Muhammad Imtiaz Rashid, Nadeem Ali, Bothinah Abdullah Saeed Altaf & Muhammad Munir (2021), 'Temperature, humidity and outdoor air quality indicators influence covid-19 spread rate and mortality in major cities of saudi arabia', *Environmental Research* p. 112071.
- James, Spencer L, Degu Abate, Kalkidan Hassen Abate, Solomon M Abay, Cristiana Abbafati, Nooshin Abbasi, Hedayat Abbastabar, Foad Abd-Allah, Jemal Abdela, Ahmed Abdelalim et al. (2018), 'Global, regional, and national incidence, prevalence, and years lived with disability for 354 diseases and injuries for 195 countries

- and territories, 1990–2017: a systematic analysis for the global burden of disease study 2017', *The Lancet* **392**(10159), 1789–1858.
- Jang, Won Mo, Deok Hyun Jang & Jin Yong Lee (2020), 'Social distancing and transmission-reducing practices during the 2019 coronavirus disease and 2015 middle east respiratory syndrome coronavirus outbreaks in korea', *Journal of Korean Medical Science* **35**(23).
- Kingma, Diederik P & Max Welling (2013), 'Auto-encoding variational bayes', *arXiv preprint arXiv:1312.6114*.
- Kong, Jude Dzevela, Edward W Tekwa & Sarah A Gignoux-Wolfsohn (2021), 'Social, economic, and environmental factors influencing the basic reproduction number of covid-19 across countries', *PloS one* **16**(6), e0252373.
- Ksiazek, Thomas G, Dean Erdman, Cynthia S Goldsmith, Sherif R Zaki, Teresa Peret, Shannon Emery, Suxiang Tong, Carlo Urbani, James A Comer, Wilina Lim et al. (2003), 'A novel coronavirus associated with severe acute respiratory syndrome', *New England journal of medicine* **348**(20), 1953–1966.
- Kumar, Naresh & Seba Susan (2020), Covid-19 pandemic prediction using time series forecasting models, *em* '2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)', IEEE, pp. 1–7.
- Lambrou, Anastasia S (2022), 'Genomic surveillance for sars-cov-2 variants: predominance of the delta (b. 1.617. 2) and omicron (b. 1.1. 529) variants—united states, june 2021–january 2022', *MMWR. Morbidity and mortality weekly report* **71**.
- Lechien, Jerome R, Carlos M Chiesa-Estomba, Sammy Place, Yves Van Laethem, Pierre Cabaraux, Quentin Mat, Kathy Huet, Jan Plzak, Mihaela Horoi, Stéphane Hans et al. (2020), 'Clinical and epidemiological characteristics of 1420 european patients with mild-to-moderate coronavirus disease 2019', *Journal of internal medicine* **288**(3), 335–344.
- Lecun, Yann (1987), 'Phd thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models)'.
- Majhi, Babita (2023), 'A modified artificial neural network (ann)-based time series prediction of covid-19 cases from multi-country data', *Journal of The Institution of Engineers (India): Series B* **104**(2), 335–350.
- Masci, Jonathan, Ueli Meier, Dan Cireşan & Jürgen Schmidhuber (2011), Stacked convolutional auto-encoders for hierarchical feature extraction, *em* 'International conference on artificial neural networks', Springer, pp. 52–59.
- Milić, Miljana, Jelena Milojković & Miljan Jeremić (2022), 'Optimal neural network model for short-term prediction of confirmed cases in the covid-19 pandemic', *Mathematics* **10**(20), 3804.

- Mishra, Suryanshi, Tinku Singh, Manish Kumar & Satakshi (2024), 'Multivariate time series short term forecasting using cumulative data of coronavirus', *Evolving Systems* **15**(3), 811–828.
- Mogi, Ryohei & Jeroen Spijker (2021), 'The influence of social and economic ties to the spread of covid-19 in europe', *Journal of Population Research* pp. 1–17.
- Mousavi, Mohsen, Rohit Salgotra, Damien Holloway & Amir H Gandomi (2020), 'Covid-19 time series forecast using transmission rate and meteorological parameters as features', *IEEE Computational Intelligence Magazine* **15**(4), 34–50.
- Mousavizadeh, Leila & Sorayya Ghasemi (2021), 'Genotype and phenotype of covid-19: Their roles in pathogenesis', *Journal of Microbiology, Immunology and Infection* **54**(2), 159–163.
- Mukhtar, Harun, Reny Medikawati Taufiq, Ilham Herwinanda, Doni Winarso & Regiolina Hayami (2022), 'Forecasting covid-19 time series data using the long short-term memory (lstm)', *International Journal of Advanced Computer Science and Applications* **13**(10).
- Oliveira, Emerson V, David H do Santos & Luiz MG Goncalves (2022), Auto-regressive multi-variable auto-encoder, *em 'Conference on Graphics, Patterns and Images (SIBGRAPI)'*, SBC, pp. 147–152.
- Organization, World Health et al. (2020), 'Novel coronavirus (2019-ncov): situation report, 11'.
- Pereira, Igor Gadelha, Andouglas Gonçalves da Silva Junior, Dunfrey Pires Aragão, Emerson Vilar de Oliveira, Arthur Andrade Bezerra, Francinaldo de Almeida Pereira, Julio Gustavo FS Costa, Jordan Salas Cuno, Davi Henrique dos Santos, Joris Michel Guerin et al. (2022), 'Epidemiology forecasting of covid-19 using ai—a survey', *Computational Intelligence for COVID-19 and Future Pandemics: Emerging Applications and Strategies* pp. 89–120.
- Pereira, Igor Gadelha, Joris Michel Guerin, Andouglas Gonçalves Silva Júnior, Gabriel Santos Garcia, Prisco Piscitelli, Alessandro Miani, Cosimo Distanto & Luiz Marcos Garcia Gonçalves (2020), 'Forecasting covid-19 dynamics in brazil: a data driven approach', *International Journal of Environmental Research and Public Health* **17**(14), 5115.
- Petropoulos, Fotios & Spyros Makridakis (2020), 'Forecasting the novel coronavirus covid-19', *PloS one* **15**(3), e0231236.
- Petropoulos, Fotios, Spyros Makridakis & Neophytos Stylianou (2022), 'Covid-19: Forecasting confirmed cases and deaths with a simple time series model', *International journal of forecasting* **38**(2), 439–452.

- Rauf, Hafiz Tayyab, M Ikram Ullah Lali, Muhammad Attique Khan, Seifedine Kadry, Hanan Alolaiyan, Abdul Razaq & Rizwana Irfan (2023), 'Time series forecasting of covid-19 transmission in asia pacific countries using deep neural networks', *Personal and Ubiquitous Computing* pp. 1–18.
- Rezende, Danilo Jimenez, Shakir Mohamed & Daan Wierstra (2014), Stochastic back-propagation and approximate inference in deep generative models, *em* 'International conference on machine learning', PMLR, pp. 1278–1286.
- Rifai, Salah, Grégoire Mesnil, Pascal Vincent, Xavier Muller, Yoshua Bengio, Yann Dauphin & Xavier Glorot (2011), Higher order contractive auto-encoder, *em* 'Joint European conference on machine learning and knowledge discovery in databases', Springer, pp. 645–660.
- Rifai, Salah, Pascal Vincent, Xavier Muller, Xavier Glorot & Yoshua Bengio (2011), Contractive auto-encoders: Explicit invariance during feature extraction, *em* 'Icml'.
- Rocchetti, Marco (2024), 'Drawing a parallel between the trend of confirmed covid-19 deaths in the winters of 2022/2023 and 2023/2024 in italy, with a prediction', *Mathematical Biosciences and Engineering* **21**(3), 3742–3754.
- Safi, SK & OI Sanusi (2021), 'A hybrid of artificial neural network, exponential smoothing, and arima models for covid-19 time series forecasting. model assisted stat. appl. 16, 25–35'.
- Sharma, Shruti, Yogesh Kumar Gupta & Abhinava K Mishra (2023), 'Analysis and prediction of covid-19 multivariate data using deep ensemble learning methods', *International Journal of Environmental Research and Public Health* **20**(11), 5943.
- Shastri, Sourabh, Kuljeet Singh, Sachin Kumar, Paramjit Kour & Vibhakar Mansotra (2020), 'Time series forecasting of covid-19 using deep learning models: India-usa comparative case study', *Chaos, Solitons & Fractals* **140**, 110227.
- Shumway, Robert H, David S Stoffer & David S Stoffer (2000), *Time series analysis and its applications*, Vol. 3, Springer.
- Sperandio Nascimento, Erick Giovani, Júnia Ortiz, Adhvan Novais Furtado & Diego Frias (2023), 'Using discrete wavelet transform for optimizing covid-19 new cases and deaths prediction worldwide with deep neural networks', *Plos one* **18**(4), e0282621.
- Tomar, Anuradha & Neeraj Gupta (2020), 'Prediction for the spread of covid-19 in india and effectiveness of preventive measures', *Science of The Total Environment* **728**, 138762.
- Torres, José F, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez & Alicia Troncoso (2021), 'Deep learning for time series forecasting: A survey', *Big Data* **9**(1), 3–21.

- Xu, Hao, Chonghuai Yan, Qingyan Fu, Kai Xiao, Yamei Yu, Deming Han, Wenhua Wang & Jinping Cheng (2020), 'Possible environmental effects on the spread of covid-19 in china', *Science of the Total Environment* **731**, 139211.
- Yang, Zifeng, Zhiqi Zeng, Ke Wang, Sook-San Wong, Wenhua Liang, Mark Zanin, Peng Liu, Xudong Cao, Zhongqiang Gao, Zhitong Mai et al. (2020), 'Modified seir and ai prediction of the epidemics trend of covid-19 in china under public health interventions', *Journal of thoracic disease* **12**(3), 165.
- Zain, Zuhaira M & Nazik M Alturki (2021), 'Covid-19 pandemic forecasting using cnn-lstm: a hybrid approach', *Journal of Control Science and Engineering* **2021**(1), 8785636.
- Zaki, Ali M, Sander Van Boheemen, Theo M Bestebroer, Albert DME Osterhaus & Ron AM Fouchier (2012), 'Isolation of a novel coronavirus from a man with pneumonia in saudi arabia', *New England Journal of Medicine* **367**(19), 1814–1820.
- Zeroual, Abdelhafid, Fouzi Harrou, Abdelkader Dairi & Ying Sun (2020), 'Deep learning methods for forecasting covid-19 time-series data: A comparative study', *Chaos, Solitons & Fractals* **140**, 110121.
- Zhai, Junhai, Sufang Zhang, Junfen Chen & Qiang He (2018), Autoencoder and its various variants, *em '2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)'*, IEEE, pp. 415–419.