UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

UNIVERSITÀ DEGLI STUDI DI BERGAMO

PH.D. THESIS
IN
TECHNOLOGY, INNOVATION AND MANAGEMENT

A DEEP FRAMEWORK FOR TIME SERIES FORECASTING

ANTONINO FERRARO

TUTOR: PROF. VINCENZO MOSCATO

COORDINATOR: PROF. RENATO REDONDI

XXXVI CICLO

Table of contents

Summary

Time series forecasting is a crucial task with applications across various domains, including finance and healthcare. Conventional methods often grapple with the complexities of time-varying data, leading to the adoption of deep learning techniques. However, these approaches tend to be specialized for particular tasks, resulting in ad-hoc solutions that lack versatility. In this Ph.D. Thesis, the author presents an innovative framework designed to overcome these limitations. This framework offers a general-purpose approach, emphasizing modularity and customization in each processing step, ensuring an efficient and effective processing pipeline. To validate the framework's practicality, it is applied to real-world domains such as "Industry 4.0" and "FinTech." This pragmatic approach recognizes the challenges of universal application due to the diversity of time series data in different domains. Within the realm of Industry 4.0, the focus is on Predictive Maintenance (PdM), with two significant papers presented: The Chapter 4 evaluates various time series coding techniques, including Recurrence Plot, Gramian Angular Field, Markovian Transition Field, and Wavelet Transform, combined with image classifiers based on convolutional neural networks (CNNs). The results demonstrate the superiority of CNNs and the advantages of data augmentation

techniques, including generative adversarial networks (GANs). The Chapter 5 introduces a deep learning approach for PdM, utilizing a multi-head attention mechanism. This approach achieves high accuracy in estimating Remaining Useful Life (*RUL*) while maintaining low memory requirements, making it suitable for implementation on equipment hardware. In the realm of FinTech, the research focuses on investment strategies, particularly stock market prediction, with two significant papers: The Chapter 6 introduces a framework that leverages historical data and user-generated content from Online Social Networks (OSNs) to enhance stock forecasting. The study highlights the importance of analyzing content from multiple OSNs and explores the influence of text data from Twitter and Reddit on stock prediction. The Chapter 7 addresses various research questions related to the impact of social and news data on the stock market. It conducts a benchmarking of machine learning and deep learning forecasting models, emphasizing the correlation between market and social/news data for specific stocks. In summary, this Ph.D. Thesis contributes a versatile framework for time series forecasting, validated in diverse real-world contexts, and sheds light on key considerations for enhancing analysis and prediction. The research underscores the potential of deep learning and user-generated content from OSNs in improving stock market forecasting.

Part I

Time Series forecasting

Time Series modeling represents a dynamically evolving field of research that has captured the attention of the scholarly community over recent decades [1]. The central objective of Time Series modeling is to meticulously gather, analyze, and investigate historical data in order to construct a suitable model that encapsulates the inherent structure of the temporal sequence [2]. This resultant model is subsequently leveraged to elucidate the underlying patterns within the Time Series and to project future values—a process commonly referred to as forecasting. Time Series forecasting stands as one of the most pervasive and widely employed tasks in the realm of learning [3]. On a daily basis, businesses harness Time Series forecasting to cater to a diverse spectrum of applications, encompassing the prediction of daily stock prices, fluctuations in foreign currency exchange rates, and the anticipation of unemployment rates [4, 5]. Meteorologists, in a similar vein, employ this technique to approximate wind speeds, daily temperature extrema, and precipitation levels [6, 7].

These myriad applications, among others, underscore the profound significance of Time Series analysis and the critical role played by accurate future estimations. Such estimations prove pivotal for businesses to effectively prepare for potential surges or dips in sales, thereby enabling proactive strategies, or to avert potential disasters through the interpretation of meteorological data.

Time Series forecasting, essentially, entails the thoughtful projection of future trends based on meticulous retrospective analysis. Given the indispensable nature of this task across a multitude of domains, including finance, meteorology, commerce, and the sciences, devising an appropriate model to conform to and subsequently predict the temporal sequence is far from a straightforward endeavor [8]. The uniqueness of each signal or sequence, with its distinct properties and dependencies on exogenous parameters, poses a challenge in accurately encapsulating it within a model.

Over the passage of time, a wealth of research endeavors and model formulations have been proposed by academics, statisticians, and economists alike, all aimed at augmenting the precision of predictive forecasting. Consequently, a spectrum of Time Series models has been developed and refined, although the copiousness of available models does not inherently translate into universal applicability [9]. Notably prevalent and enduring are statistical models like AutoRegressive Integrated Moving Average (ARIMA) and Exponential Smoothing (ES), alongside machine learning regression models, when suitably adapted for Time Series, such as Support Vector Regression (SVR) [10, 11]. However, in recent years, Deep Neural Networks (DNNs) have increasingly exhibited competitiveness vis-à-vis traditional methodologies, extending their potential for versatility beyond the aforementioned models. DNNs hold the capacity to accommodate a wide array of exogenous data, enriching the insights furnished by Time Series analysis [12].

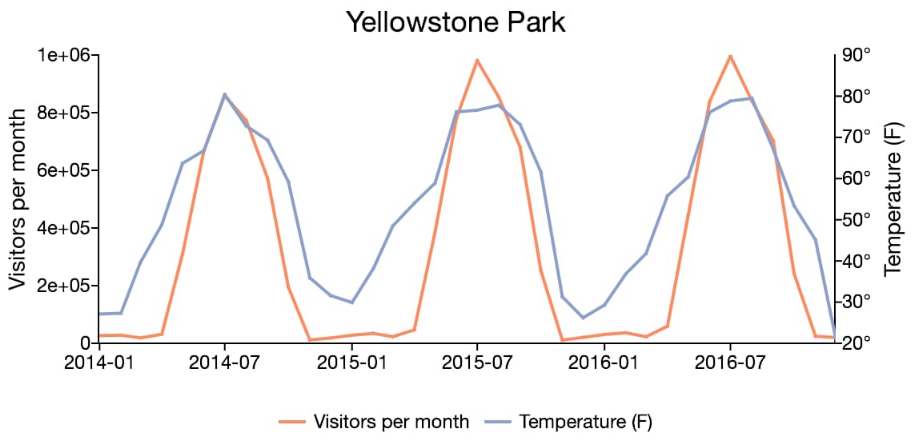


Figure 1: An example of a time series relating the number of monthly visitors to Yellowstone Park (USA) and the recorded environmental temperature, at several different times of the year.

1

Introduction to Time Series forecasting

Time series forecasting presents a distinct challenge compared to traditional regression tasks—temporal order. The sequential nature of data introduces a significant constraint for estimators striving to create a robust, enduring model. Patterns within the data may emerge temporarily and then vanish, or the entire data distribution might undergo changes. This challenge is particularly evident in stock price prediction, where regulatory shifts can fundamentally alter market behavior, presenting a substantial obstacle to accurate predictions in future periods (as illustrated in Figure 1.1).

Within the realm of time series analysis, the central aim is forecasting, a process rooted in the fundamental principle of leveraging historical data to predict future events [13]. The most suitable model, which accurately captures the underlying data patterns, forms the basis for making future predictions based on past observations. A forecasting model provides a functional representation that illuminates the behavior of a time series, serving as the framework for generating predictions.

A time series represents an ordered sequence of data points, typically measured over consecutive time intervals. In mathematical terms, it can be expressed as a set of vectors $x(t)$, where $t = 0, 1, 2, \ldots$ signifies the elapsed time

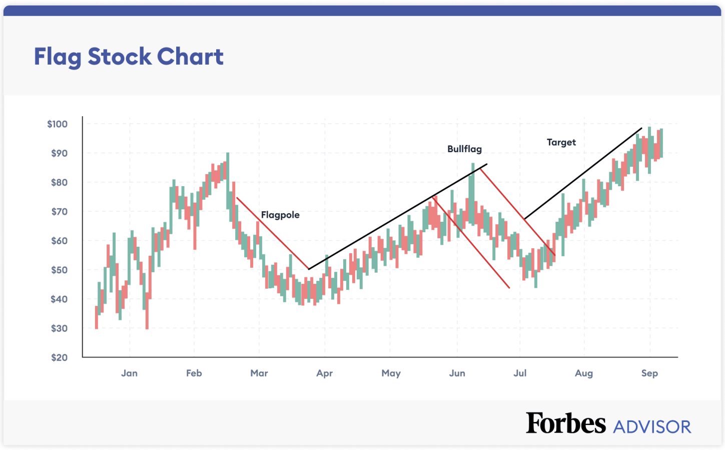


Figure 1.1: A pratical example of changes in data distribution: *Stock prices*. Market variation given by multiple factors (internal or external) is a strong obstacle to the accuracy of future forecasting.

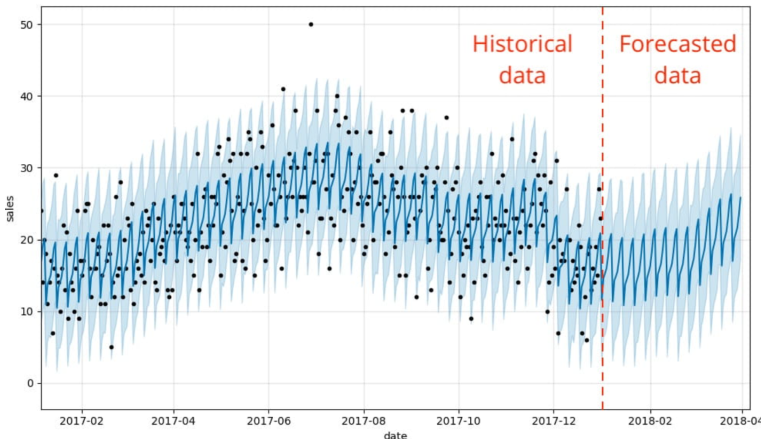


Figure 1.2: An example of Time Series Prediction: Predicting future sales, based on past historical observations.

[14, 15]. Each variable $x(t)$ is regarded as a stochastic variable. Importantly, the data points in a time series follow a systematic chronological order [2].

When a time series consists of records related to a single variable, it falls under the category of an *univariate* time series. Conversely, when the data includes records of multiple variables, it is referred to as a *multivariate* time series.

An univariate time series comprises a sequence of observations or data points recorded at regular time intervals, where each data point corresponds to a single variable [16]. In simpler terms, it involves a single stream of data points arranged in chronological order. This type of time series captures the temporal evolution of a single phenomenon or variable of interest. Analyzing univariate time series entails exploring patterns, trends, and dependencies within the data to gain insights and make predictions about its future behavior.

Within the domain of univariate time series, two primary categories of variables are relevant:

- Endogenous Variables: These encompass past values of the time series itself, forming a vital component in forecasting model construction.

- Exogenous Variables: Also known as explicative variables, these external factors are intrinsically linked to the time series value and exhibit correlation. Exogenous variables should ideally exhibit deterministic attributes, such as calendar-related data. It is imperative to ensure determinism to avert the scenario of utilizing other time series to predict the very series under consideration—a characteristic often observed in multivariate time series forecasting scenarios.

A multivariate time series, in contrast, comprises a set of observations recorded over time, with each observation containing multiple variables or attributes. In this context, each time point is represented by a vector of values, capturing the evolution of several interconnected variables simultaneously.

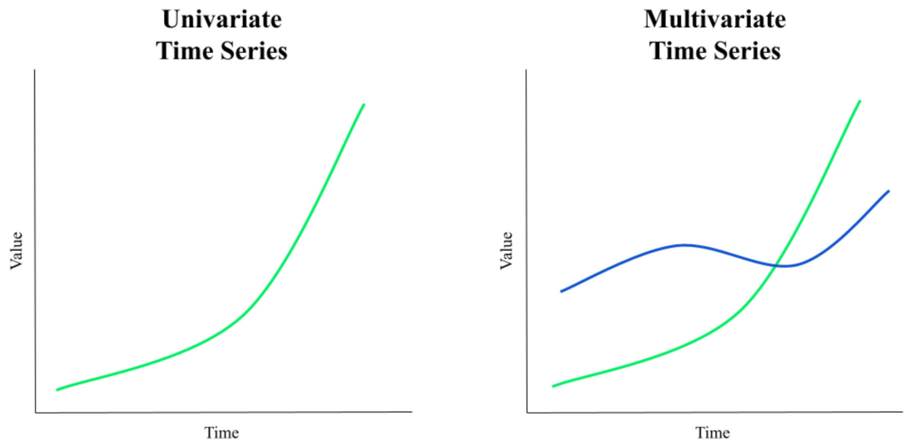


Figure 1.3: Representation of an Univariate and Multivariate Time Series.

Multivariate time series data often arise in situations where different variables are influenced by and interact with one another, resulting in intricate interdependencies. Analyzing multivariate time series entails investigating the relationships and interplay among these variables to comprehend their collective behavior and make predictions about future developments.

To summarize, the distinction between univariate and multivariate time series hinges on the number of variables considered at each time point. Univariate time series focuses on the temporal evolution of a single variable, whereas multivariate time series involves the concurrent monitoring of multiple interlinked variables over time.

Time series data comprises a sequential collection of observations denoted as x_t, recorded at discrete time points t. In the context of discrete time series, these time points belong to a set T with $t \in T$. Conversely, if observations are continuously measured over a time interval, they are referred to as a *continuous time series* [17].

Moreover, it is possible to transition from a continuous time series to a discrete one by aggregating data over predefined time intervals. This

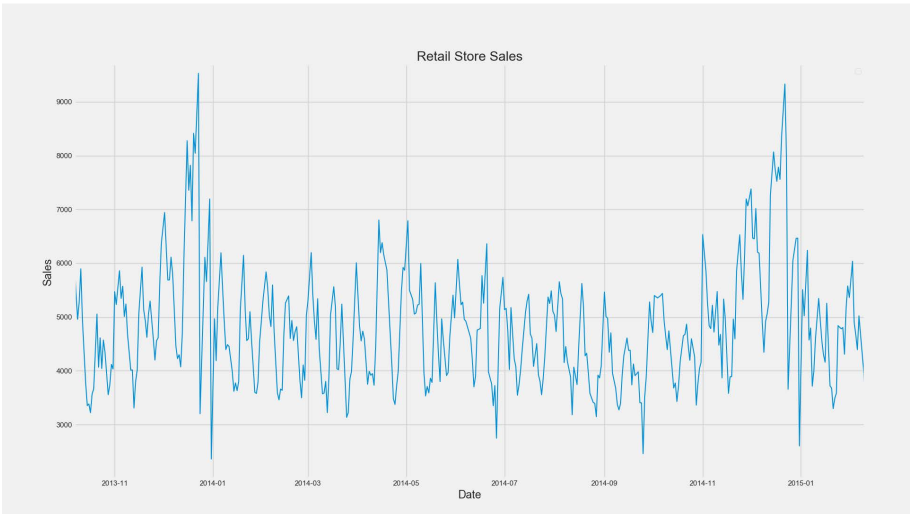


Figure 1.4: Time series related to the sales of a generic retail store.

transformation simplifies the continuous time series into a discrete form, making it more suitable for analysis and interpretation.

Time series data can be dissected into distinct components that capture specific underlying patterns [18]. These components include:

- T_t: The trend, representing persistent ascending or descending values.
- S_t: The seasonality, encompassing recurring short-term cycles with a predefined frequency.
- C_t: Cycles, akin to seasonality but characterized by less precise periodicity, often spanning durations exceeding two years.
- R_t: The residual component, encapsulating residual fluctuations.

Under the assumption of *additive decomposition*, the original time series can be expressed as the sum of its components, as shown in the equation:

$$y_t = T_t + S_t + C_t + R_t \tag{1.1}$$

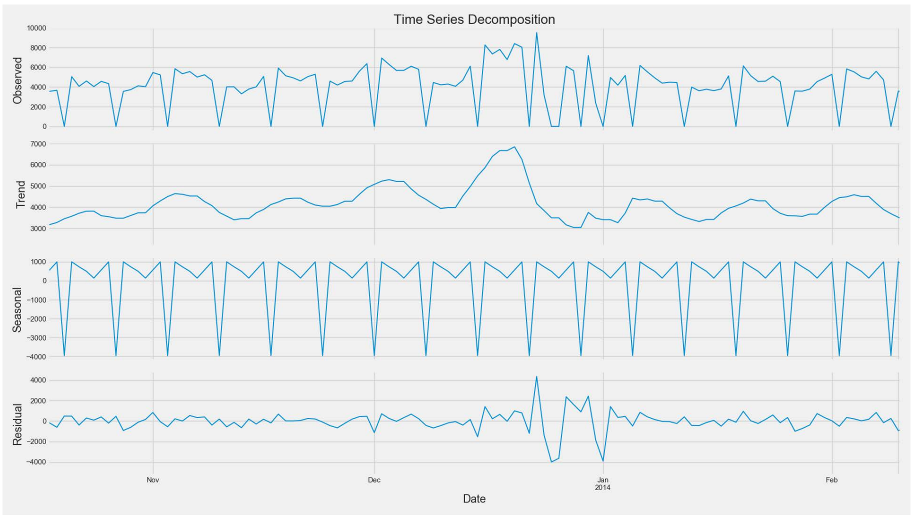


Figure 1.5: Decomposition of the time series in the figure 1.4.

The additive decomposition is suitable when the magnitude of fluctuations around the trend or the scale of seasonal variations aligns with the level (anticipated value) of the time series. However, situations where such alignment is not maintained require a different approach known as *multiplicative decomposition*, defined as:

$$y_t = T_t \cdot S_t \cdot C_t \cdot R_t \tag{1.2}$$

The multiplicative model assumes that the four constituent elements of a time series may have interdependencies, with their influences capable of interacting. In contrast, the additive model treats the four components as independent entities.

It's worth noting that in various decomposition techniques, the cyclical component is intertwined with the trend, highlighting the complex interplay between these constituents.

Residual or irregular fluctuations within a time series result from unforeseeable factors that lack a consistent pattern or repetitive nature. These variations stem from events such as conflicts, labor strikes, earthquakes, floods, revolutions, and other unpredictable occurrences. These irregular influences cannot be quantified using conventional statistical methods designed for time series analysis.

Time series data can be categorized based on various classification criteria [19]. In this context, we will focus on *Time Dependency* and *Stationarity*.

Time dependency refers to the influence of previous values on newly observed data points within the recorded variable or phenomenon. This distinction leads to the classification of time series into two categories:

- Long-term Memory Time Series
- Short-term Memory Time Series

Long-term Memory Time Series are characterized by a gradual decline in their auto-correlation function, indicating a prolonged change in behavior. These instances are often found in meteorological and geological data, with a notable example being the gradual evolution of mean planetary temperatures.

On the other hand, *Short-term Time Series* represent processes with rapid changes, and their auto-correlation function decreases quickly as the temporal distance from the present increases. This rapid decrease in correlation with time underscores their distinct nature. Financial data, such as stock prices, serves as a classic example of such time series.

Transitioning to the second classification criterion, *Stationarity*, a distinction is made between:

- Stationarity Time series
- Non-Stationarity Time series

Stationary Time Series represent processes characterized by statistical properties like mean and variance that remain unaffected by temporal fluctuations. In contrast, time series that deviate from this conventional description are labeled

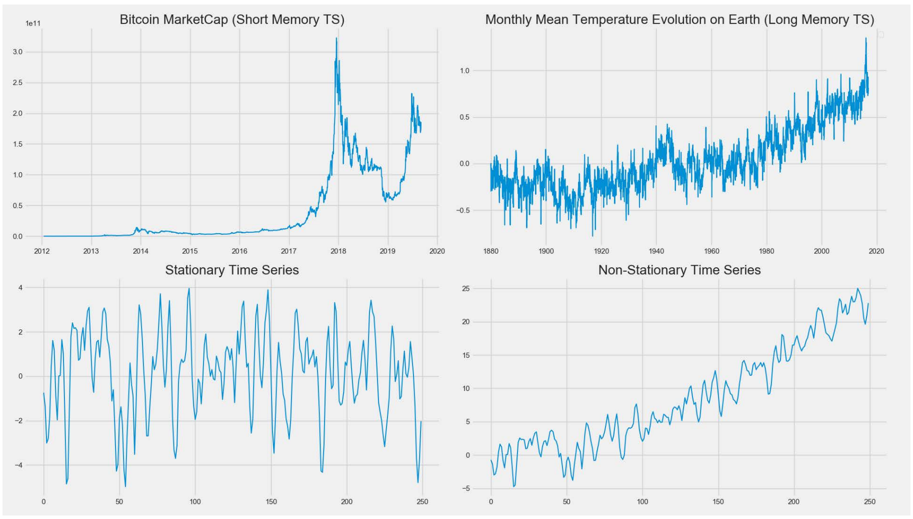


Figure 1.6: Distinct examples of different patterns of time series, starting from the top, from right to left: *Short-Term, Long-Term, Stationary* and *Non-Stationary*.

as *Non-Stationary*. Non-Stationary time series are frequently encountered in domains such as finance and retail. To tackle the forecasting challenges posed by Non-Stationary Time Series, a preprocessing approach is essential. This approach involves employing a variety of techniques to transform these processes into stationary ones, ultimately enhancing predictive accuracy.

1.1 From conventional statistical models torward Deep Neural Networks

In the previous Chapter 1, a comprehensive exploration of fundamental aspects related to time series modeling and forecasting was undertaken. The significance of selecting an appropriate model was emphasized, as it plays a pivotal role in unveiling the inherent structure of the time series, forming the basis for future predictions. Time series models are broadly categorized as

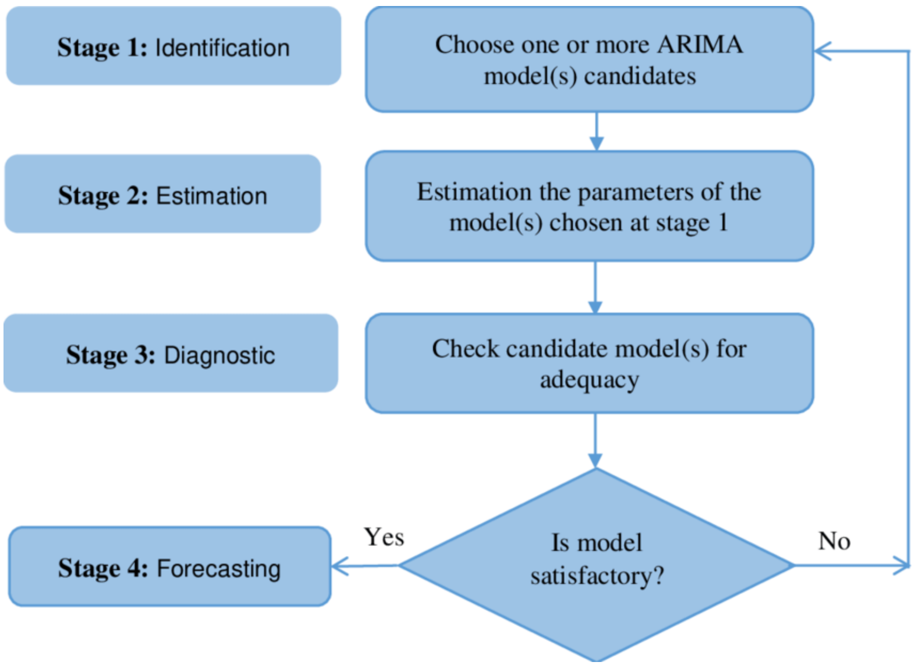

Stage 1: Identification
Choose one or more ARIMA model(s) candidates
Stage 2: Estimation
Estimation the parameters of the model(s) chosen at stage 1
Stage 3: Diagnostic
Check candidate model(s) for adequacy
Is model satisfactory?
Yes
No
Stage 4: Forecasting

Figure 1.7: The figure describes the Box-Jenkins principle, it is based on 4 steps: i) Identification ii) Estimation iii) Diagnostic and iv) Forecasting

either *Linear* or *Non-Linear*, depending on whether the current value of the series is a linear or non-linear function of past observations. Within the realm of time series data, a multitude of model forms exist [20], encompassing diverse *Stochastic* and *Deterministic* processes. In this landscape, two prominently employed linear time series models have garnered considerable attention: the *Autoregressive* (AR) and *Moving Average* (MA) models. Combining elements from these two paradigms, the *Autoregressive Moving Average (ARMA)* and *Autoregressive Integrated Moving Average* (ARIMA) models were introduced. To address seasonal time series forecasting, a variation of ARIMA, known as the *Seasonal Autoregressive Integrated Moving Average* (SARIMA) model, proves invaluable. The ARIMA model and its various derivations adhere to the *Box-Jenkins principle*, earning them the collective moniker of Box-Jenkins models, as illustrated in Figure 1.7.

In the realm of time series modeling, linear models have garnered prominence owing to their relative simplicity in terms of interpretation and implementation. However, it is worth noting that many real-world time series exhibit non-linear patterns. For instance, as highlighted by [21], non-linear models prove to be more suitable for forecasting volatility changes in economic and financial time series. Consequently, a spectrum of non-linear models has been proposed, including the widely recognized *Autoregressive Conditional Heteroskedasticity* (ARCH) model and its derivatives such as *Generalized ARCH* (GARCH) and *Exponential Generalized ARCH* (EGARCH) models.

The time series forecasting field has also embraced the *Deep Neural Networks* (DNNs) approach, which has gained substantial traction in recent years. This is evident in the following queries conducted on *Scopus*, focusing on papers published in the field of Time Series Prediction utilizing DNN-based methodologies. In particular, the following query in Scopus was employed:

TITLE-ABS-KEY (time AND series AND forecasting AND deep AND learning) AND PUBYEAR > 2006 AND PUBYEAR < 2024 AND (LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "cr") OR LIMIT-TO (DOCTYPE , "ch") OR LIMIT-TO (DOCTYPE , "re")) AND (LIMIT-TO (EXACTKEYWORD , "Time Series") OR LIMIT-TO (EXACTKEYWORD , "Forecasting") OR LIMIT-TO (EXACTKEYWORD , "Deep Learning") OR LIMIT-TO (EXACTKEYWORD , "Time Series Forecasting") OR LIMIT-TO (EXACTKEYWORD , "Time Series Analysis") OR LIMIT-TO (EXACTKEYWORD , "Machine Learning") OR LIMIT-TO (EXACTKEYWORD , "Deep Neural Networks") OR LIMIT-TO (EXACTKEYWORD , "Times Series") OR LIMIT-TO (EXACTKEYWORD , "Time Series Prediction") OR LIMIT-TO (EXACTKEYWORD , "Neural Networks") OR LIMIT-TO (EXACTKEYWORD , "Time-series Data") OR LIMIT-TO (EXACTKEYWORD , "Long Short-term Memory"))

Document type	
Article	*3,278*
Conference paper	*2,367*
Book chapter	*64*
Review	*56*
Total	***5,765***

Table 1.1: Summary of the type of documents retrieved on Scopus. A screening of the title and abstract of the papers was performed.

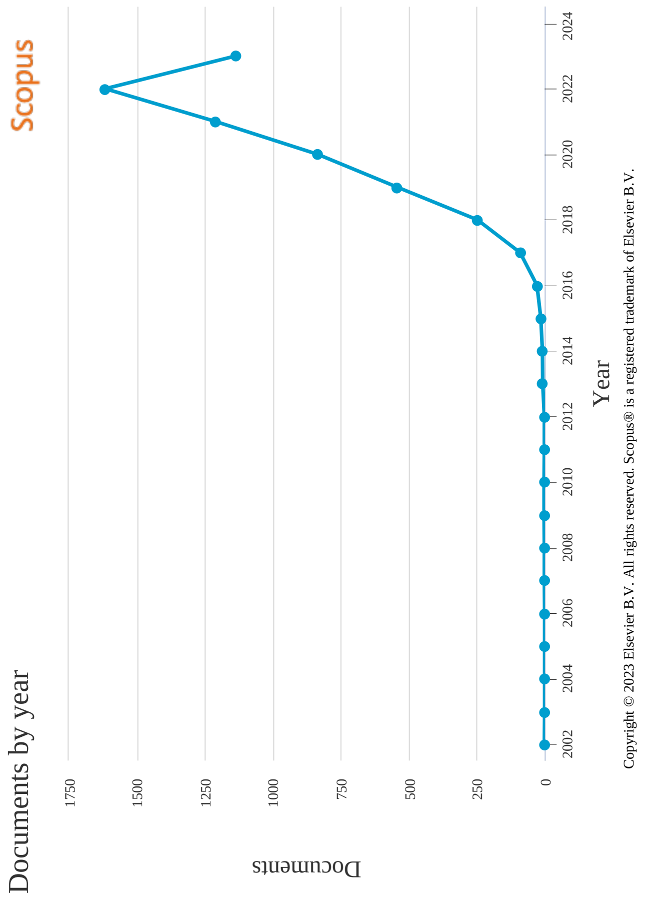


Figure 1.8: Papers published by year indexed on Scopus: we can say that the interest of the Scientific community in wanting to propose prediction models based on DNNs is registered from the year 2017.

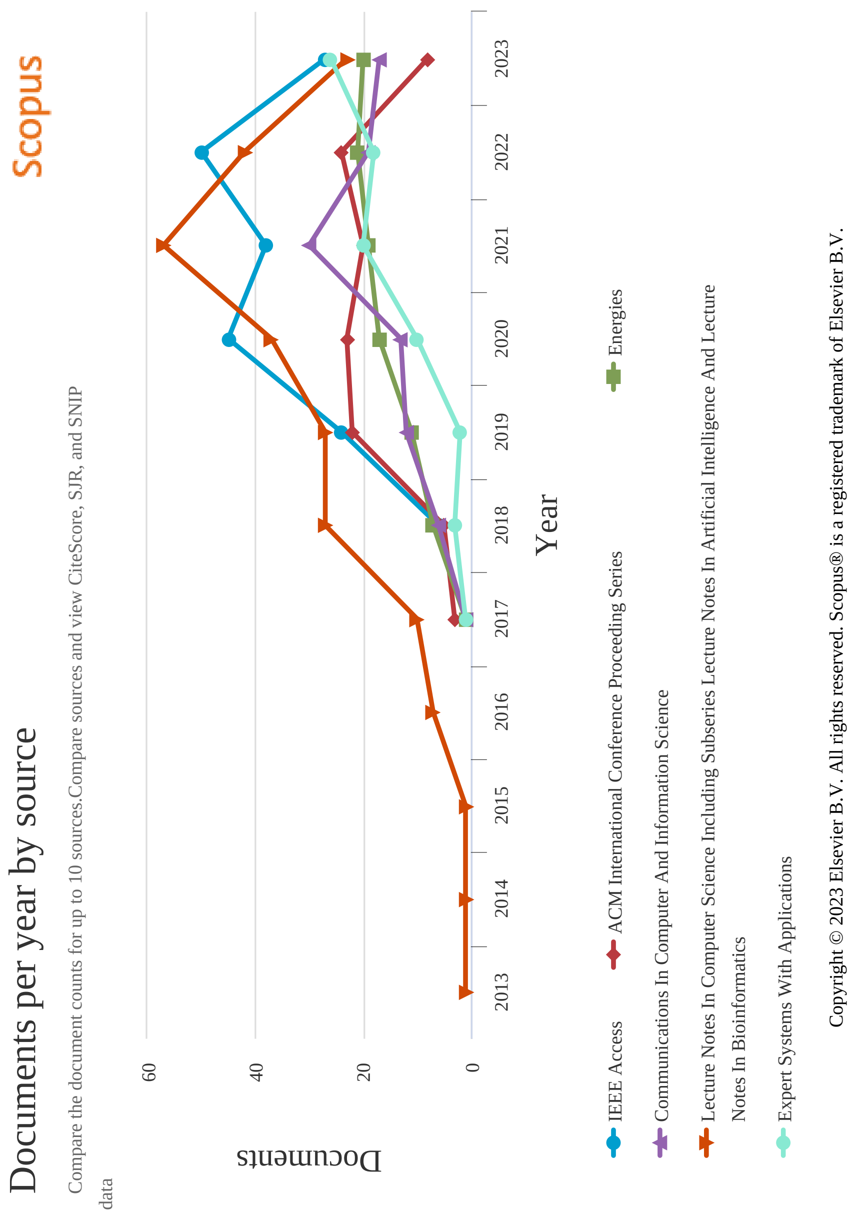


Figure 1.9: This plot shows the Top sources in terms of published papers in the field of time series prediction with Deep approaches.

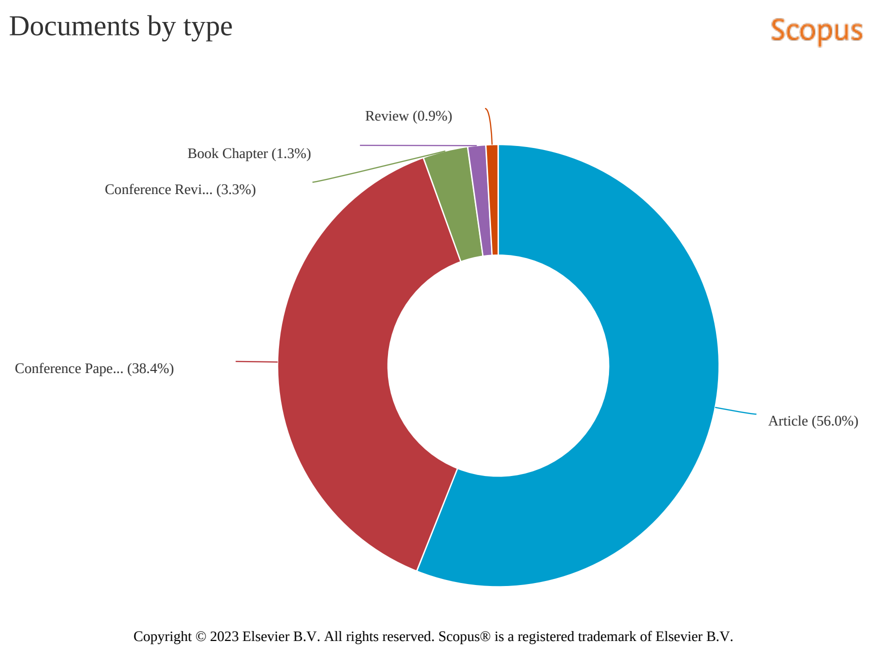


Figure 1.10: In this plot, on the other hand, we look at the types of articles published; we see that 56% are journal articles while around 38% are conference papers.

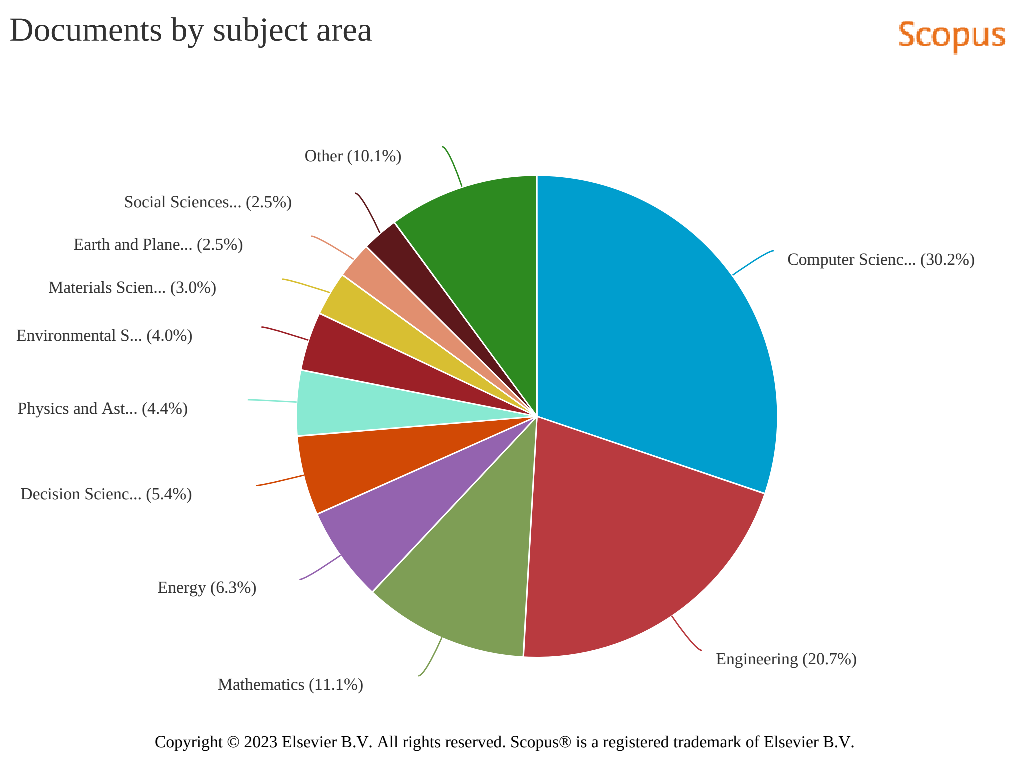


Figure 1.11: Finally, in the pie chart it is possible to observe the papers published according to the subject area, it is shows that more than 50% are from the Computer Science and Engineering area.

In the realm of artificial intelligence, Deep Neural Networks (DNNs) have emerged as a technology aiming to mimic human brain intelligence within machines [22, 23]. Similar to the human brain, DNNs excel at identifying intricate patterns within input data, learning from their experiences, and subsequently generating generalized outcomes based on acquired knowledge. While they draw inspiration from biological systems, DNNs have found diverse applications, with a particular emphasis on forecasting and classification tasks [24, 25].

The appeal of DNNs in time series analysis and prediction can be attributed to several key features:

Firstly, Deep Neural Networks (DNNs) are data-driven and adaptive in nature [26]. They do not necessitate a predefined model structure or assumptions about the data distribution. Instead, they dynamically adapt the model based on the inherent characteristics of the data. This adaptability proves invaluable in scenarios where theoretical guidance about data generation processes is lacking.

Secondly, DNNs inherently incorporate non-linearity, making them highly effective at capturing complex data patterns. This sets them apart from traditional linear methodologies like ARIMA [27]. Numerous instances demonstrate the superior analytical and forecasting capabilities of DNNs compared to linear models.

Finally, as elucidated by [28], DNNs serve as universal function approximators. Research has shown that neural networks can approximate continuous functions with any desired level of accuracy. DNNs leverage parallel processing to approximate a wide range of functions with exceptional precision. Moreover, they excel in handling scenarios involving erroneous, incomplete, or fuzzy input data [29].

In this section, a theoretical foundation of the stochastic and deterministic models commonly used in state-of-the-art time series prediction is provided, with a brief mention of the operation of DNNs, particularly Recurrent Neural

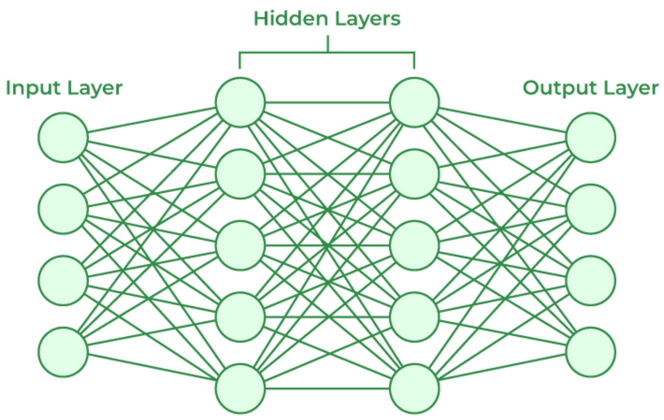


Figure 1.12: Example of a generic *Deep Neural Network* (DNN), unlike an *Artificial Neural Network* (ANN), it has more than one hidden layer.

Networks (RNNs). Detailed discussions about specific architectures are deferred to subsequent chapters (II and III).

1.1.1 Stochastic-based models

AR The Autoregressive (AR) model is a fundamental time series forecasting approach that predicts a variable's future value based on its past values. In this model, the current time series value is regarded as a linear combination of its preceding values, often referred to as "lags."

Mathematically, an AR(p) model of order p is expressed as:

$$x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + \varepsilon_t$$

Here:

- x_t denotes the time series value at time t.
- c is a constant term.

- $\phi_1, \phi_2, \ldots, \phi_p$ are autoregressive coefficients that indicate the impact of past values on the current one.

- ε_t represents white noise or random error at time t, accounting for unexplained variability and randomness.

The AR(p) model captures linear relationships between the current time series value and its lagged values up to order p, providing insights into temporal dependencies within the data.

MA The Moving Average (MA) model is another fundamental time series forecasting technique that focuses on modeling the relationship between the current value of a variable and its past forecast errors, also known as "residuals."

In an MA(q) model of order q, the current time series value is represented as a linear combination of the most recent q forecast errors, often referred to as "lags of residuals."

Mathematically, an MA(q) model is defined as:

$$x_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}$$

Here:

- x_t signifies the time series value at time t.

- μ is the mean or intercept term.

- ε_t stands for white noise or random error at time t.

- $\theta_1, \theta_2, \ldots, \theta_q$ are coefficients that determine the influence of past forecast errors on the current value.

- $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-q}$ represent past forecast errors.

The MA(q) model captures dependencies between the current value and recent forecast errors, which are assumed to be white noise. It is particularly effective in capturing short-term fluctuations or shocks in time series data.

ARMA The Autoregressive Moving Average (ARMA) model combines the Autoregressive (AR) and Moving Average (MA) concepts to create a versatile time series forecasting model that encompasses both the linear relationship between past values and the influence of past forecast errors.

An ARMA(p,q) model, with order p for the autoregressive component and q for the moving average component, is defined as:

$$x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}$$

Here:

- x_t represents the time series value at time t.
- c is a constant term.
- $\phi_1, \phi_2, \ldots, \phi_p$ are autoregressive coefficients that determine the impact of past values on the current value.
- ε_t is the white noise term or random error at time t.
- $\theta_1, \theta_2, \ldots, \theta_q$ are coefficients that determine the influence of past forecast errors on the current value.
- $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-q}$ represent past forecast errors.

The ARMA model integrates lagged values of the time series and past forecast errors to create a comprehensive forecasting model. The Autoregressive component captures temporal dependencies, while the Moving Average component accounts for the influence of past forecast errors. It is particularly useful for capturing both short-term fluctuations and the overall temporal structure of time series data.

ARIMA The Autoregressive Integrated Moving Average (ARIMA) model is a powerful and widely used time series forecasting model that combines the Autoregressive (AR), differencing (I), and Moving Average (MA) components to handle both trend and seasonality in time series data.

An ARIMA(p,d,q) model consists of three primary components:

1. Autoregressive (AR) component of order p.
2. Differencing (I) component of order d.
3. Moving Average (MA) component of order q.

Mathematically, an ARIMA(p,d,q) model can be defined as:

$$(1-\phi_1 B-\phi_2 B^2-\ldots-\phi_p B^p)(1-B)^d x_t = c+(1+\theta_1 B+\theta_2 B^2+\ldots+\theta_q B^q)\varepsilon_t$$

Here:

- x_t represents the value of the time series at time t.
- c is a constant term.
- $\phi_1, \phi_2, \ldots, \phi_p$ are the autoregressive coefficients that determine the impact of the previous values on the current value.
- d represents the order of differencing, which helps in making the time series stationary.
- B is the backshift operator, where $Bx_t = x_{t-1}$.
- $\theta_1, \theta_2, \ldots, \theta_q$ are the coefficients that determine the impact of the past forecast errors on the current value.

- ε_t is the white noise term (random error) at time t.

The ARIMA model effectively captures short-term dependencies, temporal structure, trend, and seasonality within time series data. The autoregressive and moving average components capture linear relationships with past values and forecast errors, while differencing addresses trend and seasonality, ensuring stationarity.

The ARIMA model is particularly useful for handling time series data with trend and seasonality while maintaining flexibility to model various data patterns.

SARIMA The Seasonal Autoregressive Integrated Moving Average (SARIMA) model extends the ARIMA model to address time series data with seasonal patterns. SARIMA combines Autoregressive (AR), Differencing (I), Moving Average (MA), and their seasonal counterparts to capture both temporal structure and seasonality.

The SARIMA(p,d,q)(P,D,Q)$_s$ model consists of three components for the non-seasonal part:

1. Autoregressive (AR) component of order p.
2. Differencing (I) component of order d.
3. Moving Average (MA) component of order q.

Additionally, it includes three seasonal components for the seasonal part:

1. Seasonal Autoregressive (SAR) component of order P.
2. Seasonal Differencing (SI) component of order D.
3. Seasonal Moving Average (SMA) component of order Q.

Mathematically, the SARIMA(p,d,q)(P,D,Q)$_s$ model can be defined as:

$$(1-\phi_1 B-\phi_2 B^2-\ldots-\phi_p B^p)(1-B)^d(1-\Phi_1 B^s-\Phi_2 B^{2s}-\ldots-\Phi_P B^{P\cdot s})$$

$$(1-B^s)^D x_t =$$

$$c+(1+\theta_1 B+\theta_2 B^2+\ldots+\theta_q B^q)(1+\Theta_1 B^s+\Theta_2 B^{2s}+\ldots+\Theta_Q B^{Q\cdot s})\varepsilon_t$$

Here:

- x_t represents the time series value at time t.
- c is a constant term.
- $\phi_1, \phi_2, \ldots, \phi_p$ are autoregressive coefficients for the non-seasonal part.
- d represents the order of non-seasonal differencing.
- B is the backshift operator.
- $\Phi_1, \Phi_2, \ldots, \Phi_P$ are autoregressive coefficients for the seasonal part.
- D represents the order of seasonal differencing.
- $\theta_1, \theta_2, \ldots, \theta_q$ are coefficients for the non-seasonal moving average part.
- $\Theta_1, \Theta_2, \ldots, \Theta_Q$ are coefficients for the seasonal moving average part.
- s is the seasonal period, indicating the number of time steps in a season (e.g., $s = 12$ for monthly data with yearly seasonality).

The SARIMA model excels in capturing short-term temporal relationships and longer-term seasonality present in time series data. The appropriate values for orders p, d, q, P, D, and Q are determined through statistical techniques like AIC or BIC. SARIMA is essential for forecasting time series data with recurring seasonal patterns.

	Seasonality	**Stationarity**	**Non-Stationarity**
AR	N	Y	N
MA	N	Y	N
ARMA	N	Y	N
ARIMA	N	Y	Y
SARIMA	Y	Y	Y

Table 1.2: Brief summary of the Auto-Regressive models analyzed (Y:Yes; N:No).

ARCH The Autoregressive Conditional Heteroskedasticity (ARCH) model is a statistical approach used for predicting time series data with varying levels of volatility or variance. It addresses the concept of *heteroskedasticity*, where the variability of error terms changes over time.

The ARCH model concentrates on modeling the conditional variance of the time series, implying that the variance of the error term at a given time depends on previous observations of the time series.

In mathematical terms, an ARCH(p) model can be expressed as:

$$\varepsilon_t = \sigma_t z_t$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 \varepsilon_{t-1}^2 + \alpha_2 \varepsilon_{t-2}^2 + \ldots + \alpha_p \varepsilon_{t-p}^2$$

Here:

- ε_t denotes the error term at time t.
- σ_t^2 represents the conditional variance of the error term at time t.
- z_t is a white noise term with a mean of zero and a variance of one.
- $\alpha_0, \alpha_1, \ldots, \alpha_p$ are the parameters of the ARCH model.
- $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-p}$ are past error terms.

The ARCH model accounts for changing volatility in the time series by adjusting the conditional variance based on past squared error terms. This adaptability is crucial for handling time series data with varying volatility levels.

GARCH The Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model is an extension of the ARCH model that offers a more flexible and advanced approach to modeling time series data with varying volatility. Similar to the ARCH model, the GARCH model centers on capturing the conditional variance of the time series.

The GARCH model introduces lagged values of both the conditional variance and squared past error terms to better capture the dynamics of changing volatility. It accommodates both short-term and long-term effects on the conditional variance.

The GARCH(p, q) model can be defined as:

$$\varepsilon_t = \sigma_t z_t$$

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2$$

Here:

- ε_t represents the error term at time t.
- σ_t^2 represents the conditional variance of the error term at time t.
- z_t is a white noise term with a mean of zero and a variance of one.
- ω is a constant term.
- $\alpha_1, \alpha_2, \ldots, \alpha_p$ are the autoregressive coefficients for squared past error terms.

- $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-p}$ are squared past error terms.
- $\beta_1, \beta_2, \ldots, \beta_q$ are the moving average coefficients for past conditional variances.
- $\sigma_{t-1}^2, \sigma_{t-2}^2, \ldots, \sigma_{t-q}^2$ are past conditional variances.

The GARCH model enhances the ARCH model by considering both squared past error terms and past conditional variances in calculating the conditional variance at each time step. This approach allows the GARCH model to capture the persistence of volatility changes over time, making it suitable for time series data with complex volatility patterns.

EGARCH The Exponential Generalized Autoregressive Conditional Heteroskedasticity (EGARCH) model is an extension of the GARCH model that accommodates asymmetric effects of positive and negative shocks on the conditional volatility of a time series. The EGARCH model is particularly effective at capturing the phenomenon of leverage, where negative shocks tend to increase volatility more than positive shocks of the same magnitude.

Mathematically, an EGARCH(p, q) model can be defined as:

$$\varepsilon_t = \sigma_t z_t$$

$$\log(\sigma_t^2) = \omega + \sum_{i=1}^{p} \alpha_i \left(\left| \frac{\varepsilon_{t-i}}{\sigma_{t-i}} \right| - \sqrt{\frac{2}{\pi}} \right) + \sum_{j=1}^{q} \beta_j \log(\sigma_{t-j}^2)$$

Here:

- ε_t represents the error term at time t.
- σ_t^2 represents the conditional variance of the error term at time t.
- z_t is a white noise term with a mean of zero

  and a variance of one.

- ω is a constant term.
- $\alpha_1, \alpha_2, \ldots, \alpha_p$ are the autoregressive coefficients for asymmetric effects.
- $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-p}$ are past error terms.
- $\beta_1, \beta_2, \ldots, \beta_q$ are the moving average coefficients for past conditional log-variances.
- $\log(\sigma_{t-1}^2), \log(\sigma_{t-2}^2), \ldots, \log(\sigma_{t-q}^2)$ are past conditional log-variances.

The distinctive feature of the EGARCH model is its inclusion of the absolute values of past standardized errors ($\frac{\varepsilon_{t-i}}{\sigma_{t-i}}$) in the conditional variance equation. This allows the model to effectively capture the asymmetric response of volatility to positive and negative shocks. The term $\sqrt{\frac{2}{\pi}}$ is included to maintain the stability of the conditional variance equation.

The EGARCH model's capability to model asymmetric volatility dynamics makes it a valuable tool for accurately modeling and forecasting time series data with these characteristics.

In general, models from the GARCH family are particularly useful for financial time series data, where negative shocks often lead to larger increases in volatility compared to positive shocks of the same magnitude.

The selection of appropriate orders (p, d, q, P, D, and Q) for these models depends on the data's characteristics and can be determined using statistical techniques like the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC).

1.1.2 Deterministic-based models

In this section, deterministic time series forecasting models are introduced:

SES The Simple Exponential Smoothing (SES) model is a fundamental time series forecasting model used for short-term predictions. It is ideal for time series data with a constant level and no significant trends or seasonality. The SES model calculates the forecast for the next period as a weighted average of the most recent observation and the previous forecast. These weights decrease exponentially as observations move further into the past, with more emphasis on recent data.

The formula for the SES model is represented as:

$$\hat{y}_{t+1} = \alpha \cdot y_t + (1 - \alpha) \cdot \hat{y}_t$$

Where: - $\hat{y}_{t+1}$ is the forecast for the next period. - y_t is the observed value at time t. - $\hat{y}_t$ is the forecast at time t. - α is the smoothing parameter (smoothing factor) that controls the weight assigned to the most recent observation, with a range between 0 and 1.

The SES model emphasizes recent observations, with the parameter α determining the extent of this emphasis. A smaller α assigns more weight to past forecasts, resulting in a smoother and less responsive forecast, whereas a larger α gives more weight to the most recent observation, resulting in a more responsive forecast.

SES is suitable for relatively stable time series data with minimal trends or seasonality. It provides a simple method for generating short-term forecasts, making it a valuable starting point for forecasting tasks. However, its simplicity may limit its effectiveness for more complex time series patterns.

DES Holt's Linear Trend model, also known as Double Exponential Smoothing (DES), extends SES by incorporating a linear trend component. It is designed for time series data with both a level component and a linear trend over time. The model introduces two smoothing parameters: one for the level (α) and another for the trend (β). It captures the current level and the rate of change in the time series, and forecasts the next period based on the current level and trend.

The mathematical representations for Holt's Linear Trend model are as follows:

Level equation

$$L_t = \alpha \cdot y_t + (1 - \alpha) \cdot (L_{t-1} + T_{t-1})$$

Trend equation

$$T_t = \beta \cdot (L_t - L_{t-1}) + (1 - \beta) \cdot T_{t-1}$$

Forecast equation

$$\hat{y}_{t+1} = L_t + T_t$$

Where:

- L_t is the estimated level at time t.
- T_t is the estimated trend at time t.
- y_t is the observed value at time t.
- $\hat{y}_{t+1}$ is the forecast for the next time period.
- α is the smoothing parameter for the level, which determines the weight assigned to the most recent observation.
- β is the smoothing parameter for the trend, which determines the weight assigned to the estimated trend.

Similar to SES, the choice of α and β influences the weight given to recent observations and trends, respectively. Adjusting these parameters allows the model to react more quickly or slowly to changes in the data. Holt's Linear Trend model is suitable for time series data with a consistent trend component but may not perform well with nonlinear trends or seasonality.

TES The Triple Exponential Smoothing (TES) model, also known as the Holt-Winters Exponential Smoothing model, extends Holt's Linear Trend by incorporating components for level, trend, and seasonality. It is appropriate for time series data with level, trend, and seasonal patterns. TES introduces three smoothing parameters: one for the level (α), one for the trend (β), and one for the seasonal component (γ).

The mathematical representations for the TES model are as follows:

Level equation

$$L_t = \alpha \cdot (y_t - S_{t-m}) + (1-\alpha) \cdot (L_{t-1} + T_{t-1})$$

Trend equation

$$T_t = \beta \cdot (L_t - L_{t-1}) + (1-\beta) \cdot T_{t-1}$$

Seasonal equation

$$S_t = \gamma \cdot (y_t - L_t) + (1-\gamma) \cdot S_{t-m}$$

Forecast equation

$$\hat{y}_{t+h} = L_t + h \cdot T_t + S_{t-m+h}$$

Where:

- L_t is the estimated level at time t.
- T_t is the estimated trend at time t.
- S_t is the estimated seasonal component at time t.
- y_t is the observed value at time t.
- $\hat{y}_{t+h}$ is the forecast for time $t+h$, where h represents the forecast horizon.

	Trend	**Seasonality**	**Multiplicative Seasonality**
SES	Y	N	N
DES	Y	N	N
TES	Y	Y	Y

Table 1.3: Brief summary of the Exponential-Smoothing models analyzed (Y:Yes; N:No)

- m is the number of seasons (seasonal periods) in a cycle.
- α is the smoothing parameter for the level.
- β is the smoothing parameter for the trend.
- γ is the smoothing parameter for the seasonal component.

The TES model accounts for interactions between the level, trend, and seasonality, allowing it to capture more complex patterns in the data. It is particularly useful for time series data with seasonality that repeats at regular intervals. However, TES may not perform well with data exhibiting irregular or changing patterns. Smoothing parameters (α, β, and γ) are typically estimated using optimization techniques to minimize forecast errors on historical data.

1.1.3 Deep-based models

Stochastic models, Deterministic models and DNNs are three prominent methodologies for time series forecasting. However, each approach possesses distinct limitations that researchers and practitioners must carefully consider when selecting the most appropriate methodology [30].

Stochastic Models: Stochastic models, often grounded in statistical methods, have been extensively employed for time series forecasting. These models assume a probabilistic distribution governing the data generation process. Notable stochastic models encompass ARIMA (AutoRegressive

Integrated Moving Average), GARCH (Generalized AutoRegressive Conditional Heteroskedasticity), and state space models.

Nevertheless, stochastic models entail several limitations [31]:

1. **Assumption Rigidity:** Stochastic models frequently impose strict distributional assumptions on the data, which may not hold true in real-world scenarios. Violations of these assumptions can lead to suboptimal forecasting outcomes.
2. **Complex Pattern Representation:** Stochastic models may struggle to capture intricate temporal patterns, particularly when confronted with nonlinear relationships or intricate interactions.
3. **Feature Engineering Demands:** Stochastic models often necessitate manual feature engineering, a process that can be time-intensive and might inadvertently overlook relevant data features.
4. **Memory Constraints:** Stochastic models usually incorporate only a finite history of past observations, potentially neglecting significant long-term dependencies.

Deterministic Models: Deterministic models, exemplified by Exponential Smoothing, offer an alternative paradigm for time series forecasting. These models emphasize capturing patterns through weighted averages of past observations, while incorporating trend and seasonality components.

However, deterministic models also present limitations [32]:

1. **Sensitivity to Initialization:** Exponential Smoothing models can be sensitive to initial parameter choices, potentially leading to divergent forecasts with slight variations in initialization.
2. **Pattern Complexity:** These models might struggle to capture intricate and nonlinear patterns inherent in certain datasets.

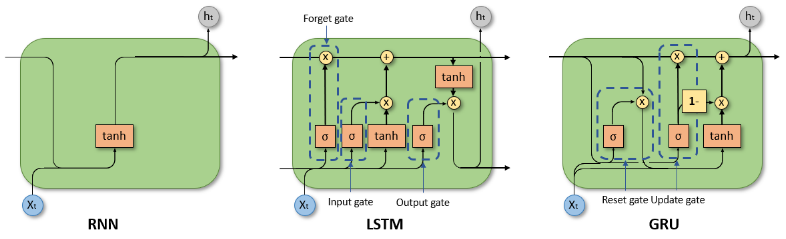


Figure 1.13: Architectures of Recurrent Neural Networks (RNNs), starting with *Vanilla*-RNN, and then exploring *Long Short Term Memory* and *Gated Recurrent Unit* (GRU).

3. **Limited Long-term Dependencies:** Deterministic models often prioritize short-term dependencies, potentially missing out on capturing longer-term trends and dependencies.

Deep Neural Networks: Deep neural networks, specifically Recurrent Neural networks (RNNs) and their variations like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), have garnered attention for time series forecasting due to their proficiency in learning intricate temporal relationships [33]. In particular, they are specifically designed to handle sequential data. RNNs possess feedback connections that allow them to maintain a form of memory about previous time steps, while LSTMs enhance this by incorporating gating mechanisms that control the flow of information through the network's memory cells. These architectures are adept at modeling and remembering dependencies over long sequences, making them well-suited for time series prediction tasks [34].

A Recurrent Neural Network (RNN) is a type of neural network architecture designed to process sequences of data while maintaining a memory of previous elements in the sequence. This memory enables RNNs to capture temporal dependencies and patterns in sequential data,

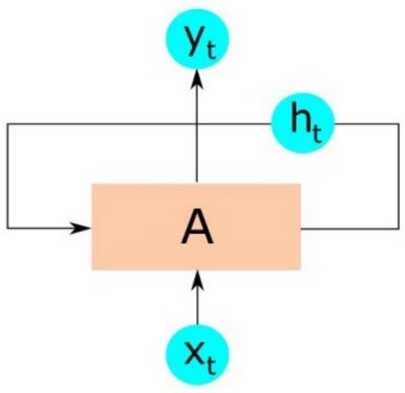


Figure 1.14: Simple RNN unit

making them well-suited for tasks like natural language processing, speech recognition, and time series analysis [35–37].

The core idea behind an RNN is the introduction of hidden states that store information about previous elements in the sequence. At each time step t, the RNN takes as input the current element of the sequence x_t and the previous hidden state h_{t-1}, and produces an output y_t and an updated hidden state h_t. This hidden state h_t serves as a memory that encodes information from past time steps and influences the network's predictions at the current time step [38].

The computations in an RNN can be described as follows:

Input to Hidden State:

$$h_t = \text{activation}\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

where:

- h_t is the hidden state at time step t.
- x_t is the input at time step t.

- W_{hx} is the weight matrix for the input-to-hidden transformation.
- W_{hh} is the weight matrix for the hidden-to-hidden transformation.
- b_h is the bias term for the hidden state transformation.
- activation is a non-linear activation function (e.g., tanh or ReLU).

Hidden State to Output:

$$y_t = \text{activation}\left(W_{yh}h_t + b_y\right)$$

where:

- y_t is the output at time step t.
- W_{yh} is the weight matrix for the hidden-to-output transformation.
- b_y is the bias term for the output transformation.

The RNN's parameters consist of the weight matrices W_{hx}, W_{hh}, and W_{yh}, as well as the bias terms b_h and b_y. These parameters are learned through the training process, typically using gradient-based optimization algorithms like backpropagation through time (BPTT).

One challenge with basic RNNs is the vanishing gradient problem, where gradients can become extremely small as they are backpropagated through time, leading to difficulty in learning long-range dependencies. This challenge has led to the development of more advanced RNN variants like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), which are designed to better capture and manage long-term dependencies in sequential data.

Yet, DNNs in general, also exhibit limitations [39]:

1. **Data Intensity:** Effective learning in DNNs often demands substantial data quantities, a requirement that can be challenging to fulfill in certain domains.

2. **Overfitting Vulnerability:** DNNs are susceptible to overfitting if not appropriately regularized, particularly when training data is limited.
3. **Computational Demands:** Training deep neural networks can entail significant computational expense and time consumption, necessitating substantial computational resources.
4. **Opaque Nature**: DNNs may be perceived as *black-box* models, hindering the interpretation of learned patterns and the comprehension of rationale behind specific forecasts.

In summary, the choice between stochastic models, deep neural networks, and deterministic models such as Exponential Smoothing should be driven by the characteristics of the time series data and the specific forecasting objectives. Each approach carries its own advantages and limitations, and an informed decision should be based on a comprehensive understanding of these factors.

In conclusion, neural networks are preferred to classical approaches, such as deterministic and linear methods, in time series prediction for several key reasons in the academic literature [40–42], including:

- It excel at capturing the intricate, nonlinear patterns found in time series data, which classical methods struggle to model effectively.
- It can automatically extract relevant features from the raw data, eliminating the need for manual feature engineering, a process often required by traditional methods.
- Time series data often show dynamic and evolutionary trends influenced by various factors, and possess the ability to capture changing patterns over time, making them suitable for modeling these trends.
- It skillfully handle multiple inputs, allowing interactions between variables to be incorporated. This ability proves invaluable in predicting time series influenced by multiple sources of information.

- Neural networks, especially those with recurrent layers, effectively capture long-term dependencies in time series data, unlike linear models that may fall short in this aspect.

In addition, given the increase in complex, high-dimensional time data sets, neural networks are more adept at handling the resulting data complexities. Moreover, it boasts a remarkable modeling capability, which enables them to learn from extensive data and adapt to different types of models, ultimately improving prediction accuracy [43]. Their adaptive learning nature facilitates continuous adjustment of internal parameters based on new data, which is crucial when dealing with nonstationary time series characterized by changing statistical properties [9]. Another important aspect is that neural networks can be integrated with ensemble methods or hybrid models to exploit the strengths of different techniques, resulting in increased prediction accuracy [44, 45]. Advances in neural network architectures and training algorithms, such as LSTM and attention mechanisms, have helped to significantly improve performance in time series forecasting tasks [46]. In conclusion, the ability of neural networks to capture complex, nonlinear relationships in time series data, their ability to adapt to changing patterns, and their suitability for high-dimensional data contribute to their superiority over classical methods in improving the accuracy of time series forecasting.

Lastly, in real-world scenarios, the use of neural networks for time series forecasting has proven to be highly advantageous over traditional deterministic and linear approaches; for further details, refer to Chapter 3.

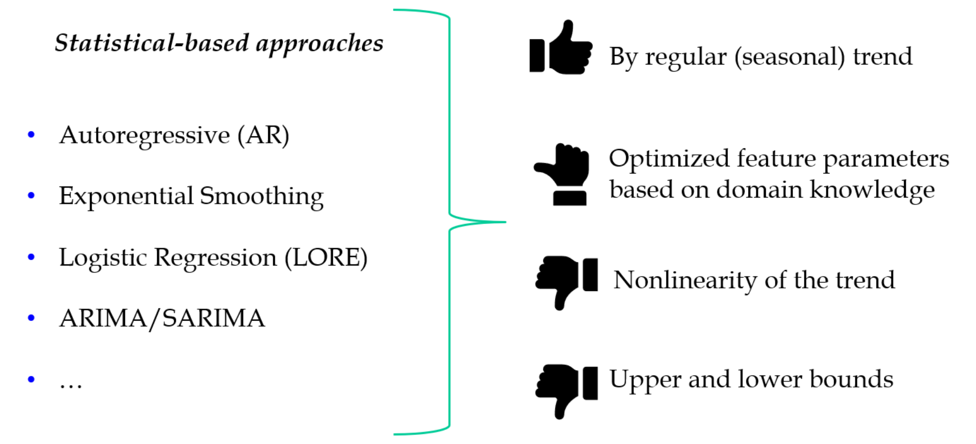

Statistical-based approaches
Autoregressive (AR)
Exponential Smoothing
Logistic Regression (LORE)
ARIMA/SARIMA
…
By regular (seasonal) trend
Optimized feature parameters based on domain knowledge
Nonlinearity of the trend
Upper and lower bounds

Figure 1.15: Summary of pros and cons on using Statistics-based approaches for time series prediction.

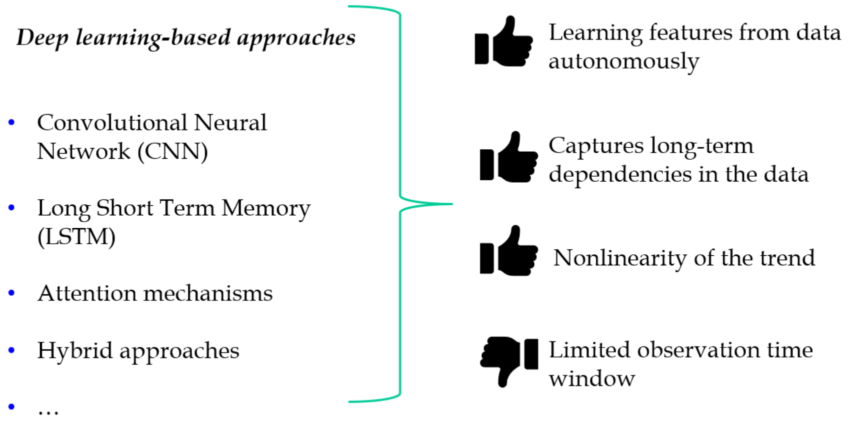

Deep learning-based approaches
Convolutional Neural Network (CNN)
Long Short Term Memory (LSTM)
Attention mechanisms
Hybrid approaches
…
Learning features from data autonomously
Captures long-term dependencies in the data
Nonlinearity of the trend
Limited observation time window

Figure 1.16: Summary of pros and cons on using Deep Learning-based approaches for time series prediction.

2 Proposed Framework

Time series forecasting is a fundamental challenge across various domains, from finance to healthcare and beyond. Traditional methods often struggle to capture the intricate patterns inherent in time-varying data, leading to the increasing adoption of deep learning techniques. However, the majority of existing approaches focus on specific forecasting tasks, resulting in *ad-hoc* solutions that lack flexibility and fail to address the broader landscape of time series prediction.

To overcome this limitation, a pioneering framework that encapsulates a general-purpose approach, modularity, and customization in each processing step for time series forecasting is presented in this dissertation to ensure an effective and efficient processing pipeline. Specifically, in this Chapter 2, the motivations, methodology applied and goals achieved by the proposed framework were explored in depth.

2.1 Motivations

The paramount objective of the proposed framework is to offer a versatile solution for time series forecasting across diverse domains. Unlike conventional methods that cater to distinct forecasting requirements, our framework adopts a holistic approach to accommodate forecasting tasks of any nature. This empowers researchers and practitioners to engage with a unified platform regardless of the domain-specific intricacies, thereby streamlining the forecasting process.

One of the distinguishing features of our framework is its modular design, where each component within the pipeline is treated as an independent module. This modularity enables users to selectively activate or deactivate modules, tailoring the processing pipeline to the specific needs of their forecasting task. The customizability extends to a granular level, encompassing functionalities such as pre-processing, feature engineering, data augmentation, and encoding strategies. This adaptability ensures that the framework caters to both novices seeking a plug-and-play solution and experts aiming to experiment with intricate preprocessing and modeling techniques.

The versatility of the framework is based on the ability to choose coding techniques for time series and deep neural network-based models for prediction. Instead of imposing a single algorithmic approach, the framework offers a spectrum of coding methodologies, allowing users to align their choices with the characteristics of the underlying data. In addition, the prediction network itself is non-prescriptive, allowing users to experiment with various deep learning architectures, from *Convolutional Neural Networks* (CNNs) to *Recurrent Neural Networks* (RNNs), based on the complexities of the data to be processed.

Instead, the framework transcends the conventional static nature of processing pipelines. Recognizing that different forecasting tasks demand tailored strategies, the framework introduces adaptive module activation. Depending on the specific challenges posed by a forecasting problem, users can dynamically enable or disable modules within the processing pipeline. This ensures that the framework remains both efficient and effective across a wide array of forecasting scenarios.

2.2 Methodology

The proposed framework revolutionizes time series processing and forecasting by introducing a meticulously designed pipeline comprising four distinct steps. Each step is characterized by its modular nature, enabling users to seamlessly tailor their approach to the intricacies of their forecasting task. The framework's overarching objective is to provide a unified solution for diverse time series forecasting challenges while upholding the principles of modularity and customization.

2.2.1 Pre-processor and Feature Engineer Module

The Pre-Processing and Feature Engineering phase stands as the bedrock of our framework, sculpting raw time series data into a form that optimally facilitates subsequent stages of analysis and forecasting. Through a combination of preparatory steps and strategic feature engineering, this phase ensures that the data is primed for extraction of relevant patterns and insights. The following techniques exemplify the comprehensive approach undertaken within this step.

Pre-Processing Techniques:

1. **Normalization:** To ensure uniformity and comparability across data points, normalization standardizes time series values within a common range. By mitigating the influence of varying scales, normalization enhances the framework's capacity to identify subtle patterns across the data.

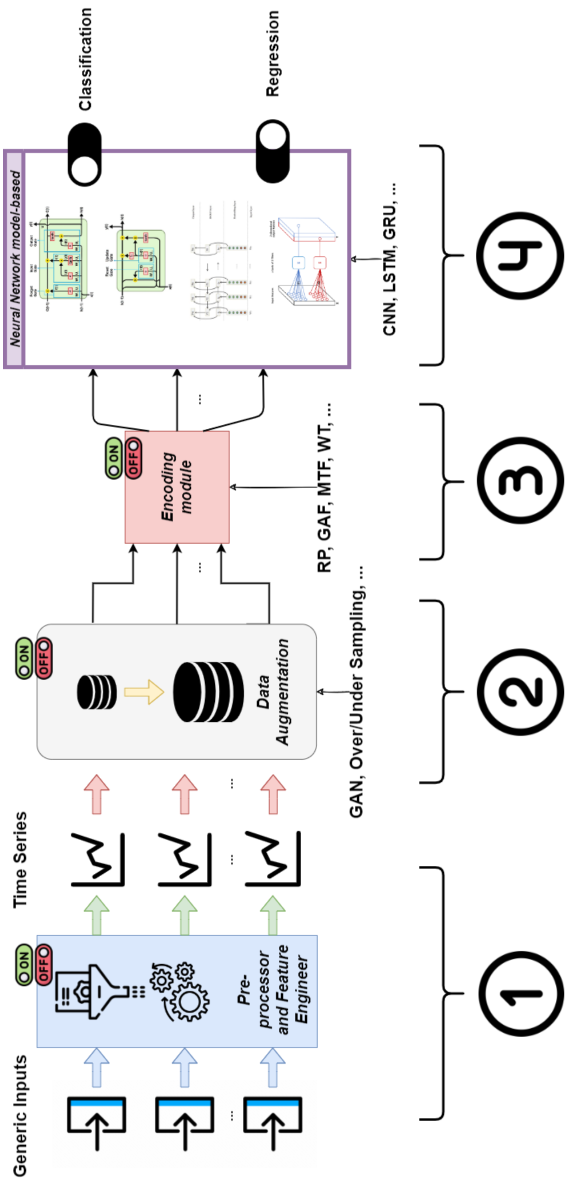


Figure 2.1: Illustration of the proposed Deep framework. It consists of 4 modules: ① Pre-processor and Feature Engineer, ② Data Augmentation, ③ Encoding and ④ Deep Model selector.

2. **Imputation:** Addressing missing values is vital to prevent data loss and ensure the integrity of subsequent analyses. Imputation techniques, such as linear interpolation or mean substitution, are employed to fill gaps and maintain continuity in the time series.

3. **Outlier Detection and Treatment:** Outliers can significantly skew forecasts and insights. Detection methods, like Z-score or percentile-based techniques, identify potential outliers, which can then be treated through transformation or removal.

Feature Engineering Strategies:

1. **Lag Features:** By introducing lagged versions of the time series data, temporal dependencies and trends are explicitly captured. These lag features enable the model to consider historical patterns, critical for accurate forecasting.

2. **Moving Averages:** Calculating moving averages smooths out noise and fluctuations, revealing underlying trends and cyclical patterns that might be obscured by noise.

3. **Statistical Measures:** Extracting statistical properties such as mean, variance, skewness, and kurtosis offers a comprehensive view of data distribution and behavior, facilitating the identification of distinct patterns.

4. **Time-Domain and Frequency-Domain Features:** These features capture both time-related characteristics (e.g., autocorrelation) and frequency-domain attributes (e.g., dominant frequencies) that contribute to the overall data profile.

The amalgamation of these pre-processing and feature engineering techniques transforms raw data into a refined format that enhances the framework's ability to capture relevant temporal patterns and characteristics. This foundational phase ensures that the subsequent modules receive data optimized for extracting valuable insights, thereby contributing to more accurate and insightful forecasting.

In the spirit of the framework's modularity, users have the autonomy to customize the pre-processing and feature engineering steps to align with their specific dataset characteristics and forecasting objectives. This adaptability empowers researchers and practitioners to cater to the unique intricacies of their time series data, a hallmark of the framework's comprehensive approach to time series analysis and prediction.

2.2.2 Data Augmentation Module

Building on the refined time series from Step ①, the Data Augmentation phase (Step ②) contributes a critical augmentation layer to the framework. Data augmentation, a technique widely recognized for its ability to alleviate data scarcity and improve model robustness, introduces synthetic variations to the existing dataset. Techniques like Generative Adversarial Networks (GANs), Over Sampling, and Under Sampling are integrated to create augmented data samples. GANs generate synthetic samples by modeling data distribution, while Over Sampling and Under Sampling address class imbalance concerns. By diversifying the dataset, the framework gains resilience against overfitting and better generalization capabilities, thereby enhancing forecasting accuracy.

Techniques for Data Augmentation:

1. Generative Adversarial Networks (GANs): GANs are a cutting-edge technique in deep learning that involves training a generator and a discriminator in tandem. The generator creates synthetic data samples that mimic the distribution of the real data, while the discriminator aims to differentiate between real and synthetic samples. The iterative interplay between these components leads to the generation of highly realistic synthetic data, enriching the dataset.

2. Over Sampling: Over Sampling targets class imbalance by replicating instances from the minority class, thereby achieving a more balanced distribution. This approach prevents the model from being biased towards the majority class and aids in capturing the subtleties of the minority class.

3. Under Sampling: Conversely, Under Sampling involves reducing instances from the majority class to balance class proportions. This technique is particularly effective when a substantial class imbalance is present, ensuring that both classes are adequately represented during model training.

4. Time-Series Specific Augmentation: Techniques like Time Warping, Noise Injection, and Magnitude Scaling introduce controlled variations to the time series. Time Warping, for instance, distorts the temporal structure, simulating scenarios where data might be temporally shifted or warped.

5. Synthetic Time Series Generation: Simulating various temporal patterns through mathematical functions or simulations can augment the dataset with diverse scenarios. This is particularly valuable for capturing uncommon patterns that might be underrepresented in the original dataset.

By infusing the dataset with synthetic variations, the Data Augmentation phase equips the framework with greater adaptability and resilience against overfitting. The resulting augmented dataset encapsulates a broader range of temporal dynamics, empowering the framework to generalize more effectively to unseen data and scenarios.

Customizability is once again a fundamental feature of this phase, allowing users to select and combine augmentation techniques according to the peculiarities of their dataset and forecasting requirements. This user-driven adaptability is integral to the overarching ethos of the framework, which prioritizes a tailored approach to time series forecasting across diverse domains.

2.2.3 Encoding Module

The Encoding Module (Step ③), a distinctive component of our framework, leverages mathematical transformations to bridge the gap between time series data and image-based deep learning architectures. Operating on the augmented time series generated in Step ②, this module employs transformations to convert time series data into images, preserving intricate temporal correlations and dependencies. The following mathematical transformations, each tailored to capture specific aspects of time series data, constitute the core of this module:

- Recurrence Plot (RP): RP is a technique that transforms a time series into a binary matrix by examining the recurrence of patterns in the data. Each element of the matrix indicates whether a pair of time points exhibits similarity, effectively highlighting temporal correlations and repeating patterns. RP excels at revealing cyclic behavior and capturing non-linear dependencies present in the time series.

- Gramian Angular Field (GAF): GAF leverages trigonometric functions to convert a time series into an image representation, portraying the relative angles between data points. This transformation encapsulates phase information and temporal correlations, enabling convolutional neural networks to identify intricate patterns and trends across the time series.

- Markovian Transition Field (MTF): MTF encodes transitions between sequential data points by mapping them onto a matrix. By quantifying the likelihood of transitioning from one state to another, MTF encodes both short-term and long-term dependencies present in the time series.

This technique is particularly effective for capturing transition patterns in data with varying temporal dynamics.

- Wavelet Transform: Wavelet Transform decomposes a time series into multiple frequency components, unveiling its multi-resolution structure. By analyzing the amplitude and frequency of each component across time, the transformed data retains both global trends and local variations. This approach is particularly suitable for capturing transient events and oscillatory patterns.

The incorporation of these mathematical transformations in the Encoding Module facilitates the conversion of time series data into images, effectively preserving temporal correlations and dependencies. These image representations enable the utilization of well-established convolutional neural network architectures, such as VGG-16 and others. The modularity of the framework extends to this module as well, allowing users to selectively adopt the transformation techniques that best align with the characteristics of their data and forecasting task.

In essence, the Encoding Module serves as a bridge between the domain of time series and image-based deep learning, unlocking a new realm of possibilities for accurate and nuanced forecasting. By harnessing the strengths of mathematical transformations, this phase imbues the framework with adaptability, enabling it to address a wide spectrum of forecasting scenarios across diverse domains.

2.2.4 Deep Model selector Module

The Choice of Deep Forecasting Model (Step ④), the culmination of our framework, offers a comprehensive selection of deep learning architectures tailored to both classification and regression tasks. Recognizing that time series forecasting encompasses diverse goals, from predicting discrete classes to continuous values, this phase provides a unified solution that accommodates both objectives seamlessly. The selection process is fortified by the modularity inherent to our framework, enabling users to align their choice with the intricacies of their forecasting challenge.

- **Supporting Classification and Regression:** For classification tasks, the framework offers a spectrum of architectures conducive to handling discrete outcomes. These encompass Convolutional Neural Networks (CNNs) with softmax outputs, tailored for class probabilities estimation, as well as LSTM and GRU networks with a final dense layer suited for multi-class classification.

For regression tasks, the framework deploys architectures optimized for continuous value prediction. Here, LSTM and GRU networks configured for regression purposes emerge as effective choices, capitalizing on their sequence-to-sequence mapping capabilities.

The evaluation of time series forecasting models is pivotal in quantifying their performance and aiding model selection. Several metrics, entrenched in the literature, serve as guiding standards [8, 47]:

- **Mean Squared Error (MSE):** This metric gauges the average squared difference between predicted and actual values. It provides insights into the overall model accuracy while heavily penalizing larger errors.
- **Mean Absolute Error (MAE):** MAE computes the average absolute differences between predicted and actual values. It offers a straightforward measure of the model's forecasting precision, less sensitive to outliers compared to MSE.
- **Root Mean Squared Error (RMSE):** RMSE extends MSE by taking the square root of the average squared differences. It presents errors in the original unit of the data, offering better interpretability.
- **Mean Absolute Percentage Error (MAPE):** MAPE quantifies the average percentage difference between predicted and actual values. It

is particularly useful for interpreting errors in the context of the data's magnitude.

- **Accuracy and F1-Score:** For classification tasks, accuracy and F1-score serve as pivotal metrics. Accuracy measures the ratio of correctly predicted instances to the total, while F1-score balances precision and recall to account for class imbalances.

- **Mean Directional Accuracy (MDA):** MDA assesses the correctness of the directional movement predicted by the model in time series forecasting. It evaluates whether the model's direction matches the observed direction of the target variable.

- **Quantile Loss:** When uncertainty estimation is crucial, quantile loss assesses the predictive quantiles of the model. It is particularly useful for probabilistic forecasting.

By offering a broad spectrum of architectures and embracing evaluation metrics tailored to both classification and regression tasks, our framework acknowledges the multifaceted nature of time series forecasting. The modularity and adaptability inherent in the framework extend to this phase as well, empowering users to select models and metrics aligned with their specific objectives and challenges. This approach resonates with the overarching philosophy of the framework, which strives to offer a unified solution that transcends domain-specific limitations in time series forecasting.

2.3 Goals

This Section 2.3 seeks to introduce the goals of the paradigm-shifting framework proposed in this dissertation, which is designed to push the field of time series analysis and forecasting to new heights. The core of this framework seeks to address the limitations inherent in traditional, specialized methodologies by embracing a comprehensive and adaptive approach. The overall goals of this effort are threefold:

1. To provide the scientific community with a versatile tool for time series analysis.
2. Instill modularity and customization as core principles.
3. To achieve excellent prediction performance through strategic calibration.

As the study unfolds, it charts the path to a future in which time series prediction enables practitioners to navigate the complex landscape of data analysis with high accuracy and effectiveness.

2.3.1 Advancing Time Series Forecasting Through General-Purpose Utility

The primary goal of this research endeavor is to contribute a versatile and comprehensive framework to the scientific community, offering an instrument of unparalleled utility for the analysis and prediction of time series data. This framework represents a groundbreaking departure from conventional, task-specific methodologies, striving to address the vast spectrum of time series forecasting challenges that span diverse domains.

2.3.2 Modularity and Customizability: Empowering Precision through Adaptability

An intrinsic tenet of our framework is its unwavering commitment to modularity and customizability. By embodying these principles, the framework seeks to empower researchers and practitioners with the freedom to orchestrate a tailored workflow that seamlessly integrates with their specific data and objectives. The modular architecture encompasses a palette of interdependent yet independent components, allowing for fine-grained control over the activation and deactivation of functionalities within the pipeline.

2.3.3 Unleashing Potential for Efficacy and Efficiency

The framework embodies the potential to achieve outstanding levels of effectiveness and efficiency, provided that the data scientist or data engineer can navigate the terrain of calibration and optimization. The optimization journey involves a judicious selection of pre-processing techniques, data augmentation strategies, encoding methodologies, and forecasting models. This carefully curated orchestration harmonizes the framework's capabilities with the intricacies of the forecasting task at hand.

In essence, this framework aspires to transcend the conventional boundaries of time series forecasting. By furnishing a platform that is both powerful and flexible, it aims to revolutionize the manner in which time series data is analyzed and forecasted. As it extends an open invitation to the scientific community, this framework embodies the essence of innovation and collaboration, poised to illuminate new paths in the realm of time series analysis.

3 Applications in Real-World scenarios

In the contemporary era of data-driven decision making, predictive analytics of time series data has emerged as a key tool in several areas. This methodology is particularly promising when exploited through deep learning approaches, which harness the power of neural networks to capture intricate temporal patterns and correlations within dynamic datasets. This introduction lays the groundwork for an exploration of concrete applications of deep learning-based time series prediction, encompassing several real-world contexts. . Notable cases range from climate monitoring to optimizing patient care trajectories, from forecasting stock market fluctuations to fine-tuning the maintenance of equipment in manufacturing processes.

Let's delve into a few concrete examples to better understand this context:

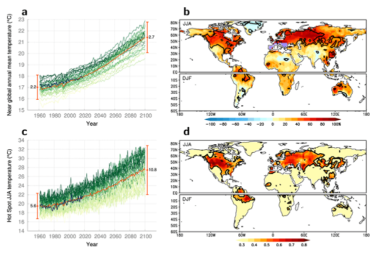
Climate Modeling: Deep neural networks have been harnessed to model complex climate patterns and predict future climatic trends. By ingesting historical data on various climatic variables, such as temperature, humidity, and atmospheric pressure, DNNs can capture intricate temporal relationships. This enables more accurate long-term climate predictions, helping researchers and policymakers anticipate shifts in weather patterns, potential natural disasters, and the impact of climate change on different regions [48, 49].

Stock Prediction: Deep neural networks are adept at analyzing large volumes of historical stock market data and identifying intricate patterns that influence stock prices. By considering factors like market sentiment, financial indicators, and historical stock prices, DNNs can make predictions that assist traders and investors in making informed decisions. Their ability to capture nonlinear relationships in financial data sets them apart from traditional linear models, contributing to more accurate short-term and long-term stock price forecasts [50, 51].

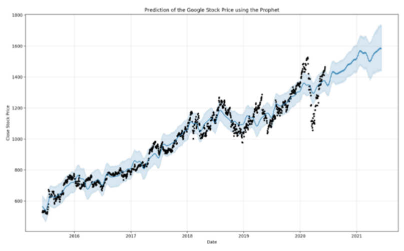
Predictive Maintenance and Fault Detection: In the realm of predictive maintenance, deep neural networks play a pivotal role in assessing the health of industrial equipment. By analyzing real-time sensor data, DNNs can detect subtle deviations from normal operating conditions that could indicate impending faults or failures. This allows maintenance teams to intervene proactively, reducing downtime and preventing costly breakdowns. DNNs excel at capturing complex patterns of equipment behavior and can adapt to changing conditions, making them a powerful tool for optimizing maintenance strategies [52, 53].

Health Forecasting: Deep neural networks have found application in forecasting healthcare-related outcomes, such as patient admission rates, disease outbreaks, and healthcare resource demands. By analyzing historical patient data, environmental factors, and public health policies, DNNs can predict disease prevalence and healthcare needs. This information aids healthcare providers, policymakers, and emergency response teams in allocating resources effectively and implementing timely interventions [54, 55].

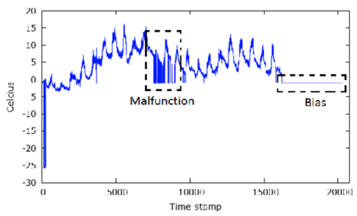
In these real-world scenarios, neural networks shine due to their ability to handle intricate and nonlinear patterns, adapt to changing conditions, and learn from complex data relationships. This effectiveness makes them a preferred choice over traditional deterministic and linear methods for accurate and reliable time series forecasting.



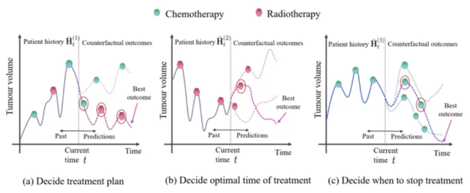
Climate modeling



Stock prediction



Predictive maintenance/
Fault detection



Health forecasting

Figure 3.1: Time-series prediction examples in real-world scenarios

Within the scope of this PhD thesis, the presented framework not only introduces a pioneering methodology but also endeavors to validate its efficacy through practical application in distinct real-world contexts. The validation efforts have notably extended to two pivotal domains: *Industry 4.0* and *FinTech*. This strategic selection stems from the recognition that attempting to apply the framework across the entire spectrum of conceivable scenarios would be a formidable undertaking. Given the ubiquity of time series data representation across diverse domains, such an all-encompassing application would inevitably encounter insurmountable challenges.

By focusing validation on specific sectors—Industry 4.0, emblematic of modern manufacturing, and Fintech, emblematic of financial technology—the thesis underscores the framework's adaptability to varying requirements and challenges. It is a pragmatic recognition that while the framework's potency is universal, the intricacies of distinct sectors necessitate tailored approaches. Through these selected real-world scenarios, the thesis exemplifies how the framework's modular and customizable architecture can be harnessed to yield profound insights and forecasts, even within contexts characterized by nuanced temporal dynamics. In the backdrop of this validation strategy, the framework emerges not merely as an abstract concept, but as a tangible asset poised to shape the landscape of practical time series analysis and forecasting.

3.1 Industry 4.0

The data generated by production processes have experienced exponential growth, driven by the widespread adoption of information and communication technologies. Analyzing and processing these vast datasets yields valuable insights into the production process, as well as the systems and equipment of interest [56]. According to industry experts, the industrial sector is currently undergoing the Fourth Industrial Revolution, commonly referred to as Industry 4.0. This revolution is primarily characterized by the profound interconnection between the physical and digital realms within a production system [57]. This correlation generates a wealth of data captured by various equipment distributed across different sectors of a company. Furthermore, Industry 4.0 enables seamless integration between machines, products, and personnel, resulting in enhanced and more efficient information exchange [58].

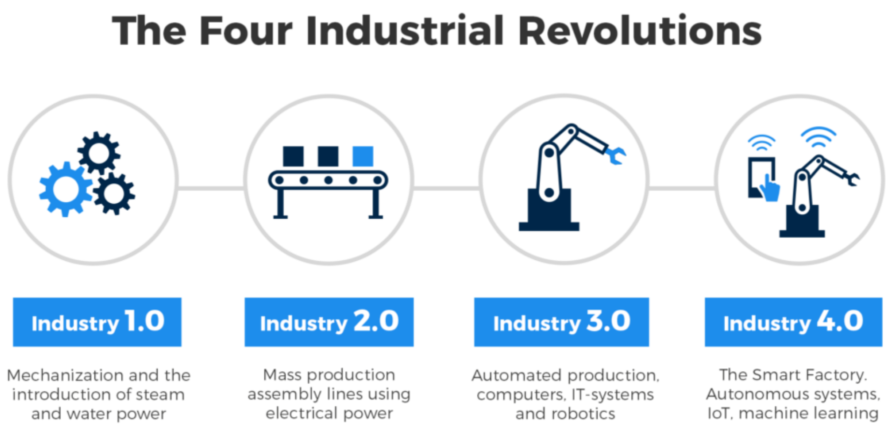

The Four Industrial Revolutions
Industry 1.0
Mechanization and the introduction of steam and water power
Industry 2.0
Mass production assembly lines using electrical power
Industry 3.0
Automated production, computers, IT-systems and robotics
Industry 4.0
The Smart Factory. Autonomous systems, IoT, machine learning

Figure 3.2: Outline of Industrial Revolutions, starting from the First Revolution toward Industry 4.0

This wealth of data collected by industrial systems encompasses events, alarms, and process information within the industrial chain. When analyzed and processed effectively, it can provide insights into production dynamics. For instance, it can help in:

- reducing production costs;
- making informed decisions about operational strategies;
- minimizing failures, costs, and repair times;
- enhancing the safety of human operators;
- optimizing profitability.

One of the most compelling challenges in the industrial domain is significantly improving the efficiency and operational timelines of various production systems. To achieve this, it is imperative to promptly identify potential faults and address them to prevent production interruptions. Different maintenance strategies are employed in the industrial context, including Run-to-Failure (R2F), Preventive Maintenance (PvM), and Predictive Maintenance (PdM). Notably, R2F, also known as corrective maintenance, is implemented only when equipment malfunctions. While it is a straightforward approach, it entails production stoppages, resulting in unexpected costs. PvM, on the other hand, involves scheduled or timed maintenance at predefined intervals to prevent breakdowns. It excels in averting failures but can lead to unnecessary corrective actions, thereby increasing maintenance costs [59].

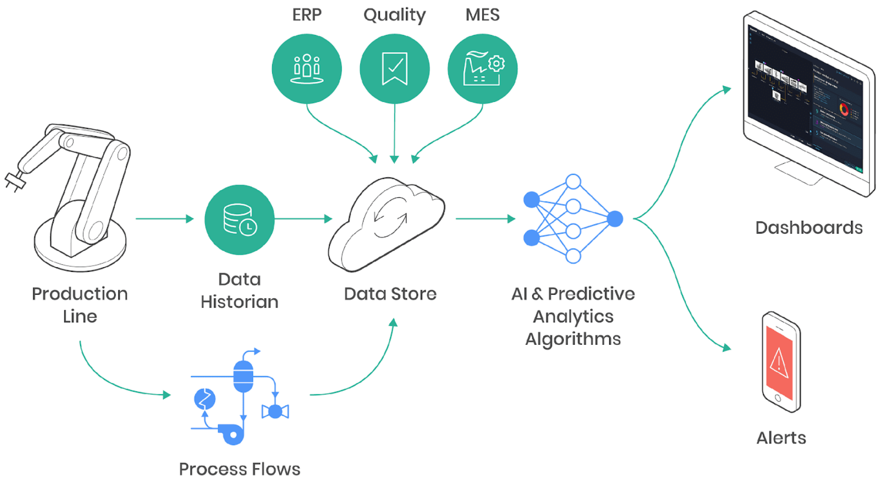


Figure 3.3: Benefits on the application of predictive maintenance in the context of production lines

In contrast, Predictive Maintenance (PdM) leverages predictive tools to determine the optimal timing for maintenance interventions. This approach necessitates continuous monitoring of the integrity of machinery or processes and intervening only when necessary [60]. Essentially, PdM aims to forecast failures before they occur, enabling maintenance precisely when needed. The advantage lies in early fault detection facilitated by predictive tools, which rely on historical data, integrity factors, engineering principles, and inferential statistics. Consequently, an optimal maintenance strategy reduces the likelihood of failures and minimizes maintenance costs, ultimately prolonging the lifespan of equipment. Undoubtedly, PdM is a highly promising strategy and a cornerstone of Industry 4.0, as it strives to minimize or delay maintenance actions while optimizing equipment operation and utilization, all by predicting potential failures before they manifest [61].

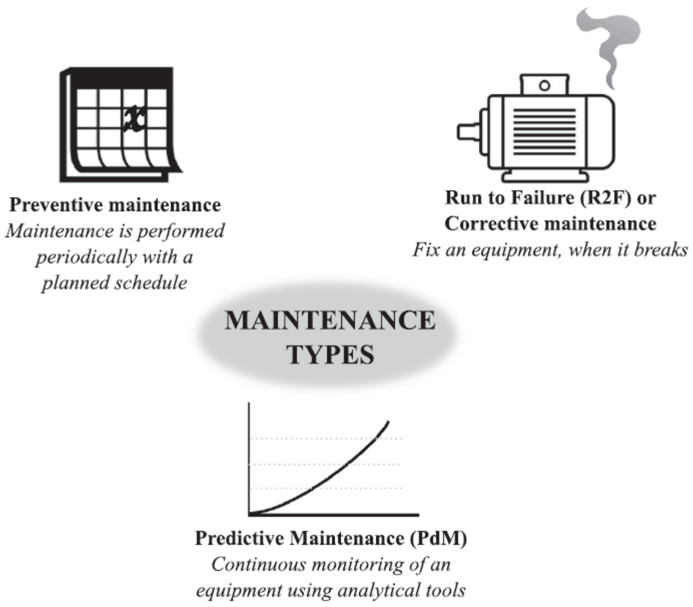

Preventive maintenance
Maintenance is performed periodically with a planned schedule
Run to Failure (R2F) or Corrective maintenance
Fix an equipment, when it breaks
MAINTENANCE TYPES
Predictive Maintenance (PdM)
Continuous monitoring of an equipment using analytical tools

Figure 3.4: Maintenance approaches: *Preventive Maintenance*, *Run to Failure* and *Predictive Maintenance*.

Recent literature highlights the efficacy of machine learning (ML) methods in PdM to prevent failures. The effectiveness of these methods hinges on the choice of ML algorithms employed. In the realm of artificial intelligence (AI), ML techniques enable the development of intelligent prediction algorithms capable of handling vast and multivariate datasets to uncover relationships that might remain hidden in complex industrial environments. Consequently, ML provides robust predictive approaches for the realm of

PdM. However, it is essential to note that the performance of predictive maintenance is closely tied to the selection of appropriate ML algorithms, as we will explore further in subsequent sections (II). Within the realm of regression-based PdM problems, several modeling challenges persist, including grappling with high-dimensionality [62], addressing data fragmentation, necessitating adaptive solutions [63, 64], and demanding interpretable models [65]. A particular open question in PdM modeling pertains to the treatment of input data, specifically equipment or process variables that exhibit time-series evolution, resulting in scalar-type outputs. The customary approach in such scenarios involves extracting a uniform set of features from each time series. However, this approach entails an inherent loss of information, given the inability to discern in advance which segments of a given time series, if any, exert influence on the output variable [66].

3.2 FinTech

Digitization has exerted a profound influence on the financial services sector, primarily attributed to the fact that financial products predominantly rely on information rather than physical components, such as payment transactions and credit contracts [67]. This transformation is further underscored by the increasing prevalence of digital processes that require little to no physical interaction, exemplified by online payment systems and stock trading [67].

Recent advancements in information technology (IT), including social computing, big data, the internet of things, and cloud computing, have not only led to process automation but have also instigated a significant restructuring of the financial services value chain. This restructuring has given rise to innovative business models such as robo-advisors and introduced new market participants like Apple [68].

Several drivers fuel this transformation:

1. **Changing Role of IT:** The convergence of IT capabilities empowers financial service providers to not only automate their existing operations but also to introduce entirely novel products, services, processes, and business models [69]. Crowdfunding and peer-to-peer insurance platforms stand as prominent examples of these new models [70].

2. **Changing Consumer Behavior:** The proliferation of electronic interaction channels among consumers necessitates a shift in how financial service providers engage with clients. This shift entails resizing branch and agent networks and transitioning towards hybrid client interaction and self-service options [71].

3. **Changing Ecosystems:** Traditional financial institutions have progressively outsourced their operations, resulting in a more specialized focus. This trend has extended from back-office functions to front-office operations, fostering the development of new ecosystems that include both incumbents and fintech startups, as well as external entities from outside the financial services sector [72].

4. **Changing Regulation:** Post-2008 financial crisis, regulatory measures have tightened across various facets of the financial services industry. However, several countries have initiated initiatives to lower entry barriers for fintech startups. Notable examples include London, Singapore, and Hong Kong, which have introduced fintech 'sandboxes' for experimenting with new products and services, specialized organizations for market development, and financial support (Monetary Authority of Singapore) [73].

Figure 3.5: Representation of the *FinTech* world and its component parts

The term 'FinTech' itself, a portmanteau of 'Financial Technology,' was first mentioned in the early 1990s by Citicorp's chairman John Reed in the context of the 'Smart Card Forum' consortium. It encompasses innovative financial solutions facilitated by IT and is commonly associated with startup companies that provide these solutions, along with traditional financial service providers like banks and insurers [74].

Furthermore, fintech is closely linked to the concept of 'financial innovation,' defined as the creation and popularization of new financial instruments, technologies, institutions, and markets [75]. These innovations manifest across primary categories, including products and services, organizational structures, processes, systems (e.g., blockchain), and business models (e.g., crowdlending), smart strategic investments (e.g. stock prediction), reflecting the multifaceted dimensions of fintech [76, 77]. Among these dimensions, stock market prediction has emerged as a topic of significant academic interest, particularly with the advent of deep neural network approaches and the utilization of heterogeneous data sources beyond traditional market data, such as sentiment analysis from social networks like Reddit and Twitter, as well as online news sources [78]. Certainly, let's delve deeper into stock market prediction and how social and news data are leveraged in this context within the fintech landscape:

Stock market prediction is a pivotal area of interest in fintech, as it seeks to forecast the future movements of financial markets, such as stock prices, indices, and commodities. Accurate predictions can provide investors, traders, and financial institutions with valuable insights for making informed decisions. Historically, stock market prediction has relied on various quantitative models, statistical analysis, and technical indicators. However, recent advancements in technology, particularly in the realm of artificial intelligence and deep learning, have introduced novel approaches that exhibit promise in improving prediction accuracy [79, 80].

One key approach in stock market prediction involves sentiment analysis of social media and news data. Sentiment analysis algorithms examine textual data from platforms like Twitter, Reddit, and online news articles to gauge the sentiment (positive, negative, or neutral) surrounding specific stocks or the overall market. For instance, if there is a surge of positive sentiment regarding a particular company on Twitter, it may indicate bullish behavior among retail investors [81, 82].

Thus, Natural Language Processing (NLP) techniques are integral in extracting valuable information from unstructured textual data. NLP algorithms can identify key phrases, opinions, and trends within social media posts and news articles. By analyzing the language used in these sources, predictive models can discern market sentiment and potential market-moving events [83].

In addition, Social and news data can be employed to detect significant events that may impact the financial markets. For instance, by monitoring news articles and social media posts for keywords related to geopolitical tensions, economic indicators, or corporate earnings reports, algorithms can identify events that could trigger market fluctuations [84, 85].

Deep neural network models, such as Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), are adept at processing sequential data like time series or textual information. Fintech researchers and practitioners integrate these models with sentiment analysis to build predictive models. These models learn from historical price data and associated sentiment signals to make future price predictions [86]. To enhance prediction accuracy, fintech applications often combine diverse data sources, including market data, social sentiment data, and news data, into a single integrated model. This data fusion approach allows for a holistic view of the factors influencing market movements [87].

Finally, the speed of social and news data processing is crucial for stock market forecasting. In fact, advanced fintech systems use real-time data feeds and fast processing algorithms to ensure that forecasts can be made quickly, enabling traders and investors to respond quickly to changing market conditions [88].

In summary, social media and news data play a pivotal role in enhancing stock market prediction within the fintech domain. By harnessing the power of sentiment analysis, natural language processing, and predictive modeling, fintech applications can provide valuable insights to market participants, helping them make informed decisions and manage risk effectively in the dynamic world of financial markets.

Part II

Time Series forecasting in Industry 4.0

In this Part II, two approaches applied in the context of Industry 4.0, specifically focusing on predictive maintenance (PdM), are presented.

In Chapter 4, the emphasis is placed on predictive maintenance (PdM) within industrial settings, addressing the challenge of data-intensive requirements inherent in machine learning and deep learning approaches. The paper introduces a comprehensive framework for assessing time series encoding techniques in conjunction with Convolutional Neural Network-based image classifiers for PdM tasks. Empirical evaluations are carried out using real-world datasets, with performance comparisons against state-of-the-art methods.

In Chapter 5, on the other hand, the exploration extends to the utilization of data-driven artificial intelligence techniques for PdM within the realm of Industry 4.0. This chapter introduces a novel deep learning (DL) approach specifically designed to address computational efficiency, tailored for Internet of Things (IoT) scenarios. The approach harnesses a multi-head attention mechanism to optimize the estimation of remaining useful life (RUL) and reduce model storage requirements. Experimental results demonstrate its superior performance in terms of both effectiveness and efficiency compared to existing techniques, positioning it as a viable solution for resource-constrained embedded AI applications. Prior to their introduction, a theoretical background is presented to facilitate comprehension for the reader.

Background

This section summarizes a combination of coding techniques and neural networks and introduces the concept of Attention Mechanism, used in the Chapters 4 and 5.

Recurrence Plot A Recurrence Plot (RP) [89, 90] is a visualization tool to explore an m-dimensional phase space trajectory through a 2-dimensional representation of its recurrences. The core idea is to reveal in which points some trajectories return to a previous state. Mathematically, this concept can be formulated as:

$$R_{i,j} = \theta(\varepsilon - ||\vec{s}_i - \vec{s}_j||), \quad \vec{s}(.) \in R^m, \quad i,j = 1,...,K \tag{3.1}$$

where K is the number of states $\vec{s}$, ε is a threshold distance, $||.||$ is the norm and θ is the Heaviside function. As a result R is a matrix. Fading to the upper left and lower right corners represents a trend, while vertical and horizontal lines indicate that some states do not change or change slowly.

Gramian Angular Field A Gramian Angular Field (GAF) [91] encoding produces an image representing a time series in a polar coordinate system rather than the typical Cartesian coordinates. Let $Y = \{y_1, y_2, \ldots, y_n\}$ be a time series having observation values scaled within the $[-1, 1]$ interval. Then, the scaled time series is represented in polar coordinates by encoding each value as the angular cosine, and the time stamp as the radius using the equation below:

$$\begin{cases} \phi = arccos(\tilde{x}_i) & -1 \leq \tilde{x}_i \leq 1, \ \tilde{x}_i \in \tilde{X} \\ r = \frac{t_i}{N} & t_i \in N \end{cases} \tag{3.2}$$

where t_i is the time stamp and N is a constant factor to regularize the span of the polar coordinate system. This transformation has three core properties: i) it is bijective with rescaled $[0, 1]$ time series data and it produces one and only map; ii) it is surjective with rescaled $[-1, 1]$ data and it produces one map, as the inverse image is not unique because of the ambiguity of $cos(\phi)$ when ϕ is in $[0, 2\pi]$; iii) unlike Cartesian coordinates, polar coordinates preserve absolute temporal relations. A GAF provides a different information granularity for classification tasks. After transforming the time series into polar coordinates, a GAF constructs a map by calculating the trigonometric sum (GASF), or the difference (GADF), between each point. GASF and GADF are calculated as follows:

$$\begin{cases} GASF = cos(\phi_i + \phi_j) \\ GADF = sin(\phi_i - \phi_j) \end{cases} \tag{3.3}$$

The GAF is defined as follow:

$$G = \begin{bmatrix} cos(\phi_1 + \phi_1) & \cdots & cos(\phi_1 + \phi_n) \\ cos(\phi_2 + \phi_1) & \cdots & cos(\phi_2 + \phi_n) \\ \vdots & \ddots & \vdots \\ cos(\phi_n + \phi_1) & \cdots & cos(\phi_n + \phi_n) \end{bmatrix} \tag{3.4}$$

As mentioned, the use of this encoding preserves temporal dependence and temporal correlations.

Markovian Transition Field In a Markovian Transition Field (MTF) [91], the encoding starts with a time series X and identifies its Q quartile bins by assigning to each x_i its corresponding bin q_j ($j \in [1,Q]$). After that, the adjacency matrix $W = Q \times Q$ is constructed, where each element $w_{i,j}$ represents the frequency with which a point in q_j is followed by a point in q_i. W is called the Markov transition matrix. Importantly, this step potentially leads to a loss of temporal information. In order to overcome this issue, the matrix is distributed. An MTF is thus defined as:

$$\begin{bmatrix} w_{ij|x_1 \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_1 \in q_i, x_n \in q_j} \\ w_{ij|x_2 \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_2 \in q_i, x_n \in q_j} \\ \cdots & \cdots & \cdots \\ w_{ij|x_n \in q_i, x_1 \in q_j} & \cdots & w_{ij|x_n \in q_i, x_n \in q_j} \end{bmatrix}$$

where each element $w_{i,j}$ represents the transition probability from quantile q_i to quantile q_j and the main diagonal is the special case of the self-transition probability from each quantile to itself. The sum of the elements of a row must have a value equal to 1. Like the GAF, the MTF is surjective and, starting from a time series X and fixing quantile bins Q, produces a single map. Note that the inverse image of the MTF is not unique.

Wavelet Transform The Wavelet Transform (WT) [92, 93] is an alternative to the more classic Fourier transform, decomposing a function into a set of wavelets. It provides high resolution in both the time and frequency domains, and it is thus suitable for analysing dynamic signals. An important property is that a wavelet exists for a finite duration.

There are two types of WT:

- **Discrete Wavelet Transform (DWT)**: The frequencies of the original signals are decomposed into *approximate coefficients* and *detail coefficients* (also called *wavelet coefficients*). Detail coefficients with larger amplitudes are considered significant, while those with smaller amplitudes are noise. A DWT used in combination with threshold denoising is a low-pass filter: it removes high-frequency noise and it is suitable for removing transient signals.

- **Continuous Wavelet Transform (CWT)**: It is based on the *mother wavelet*. One type of application requires a different mother, because each of them has a characteristic frequency band. The equivalent frequency is defined as:
$$F_{eq} = \frac{C_f}{s\delta t} \tag{3.5}$$
where C_f represents the center frequency, s is the wavelet scale and δt is the sampling interval. The output of a CWT are coefficients that are function of scale, frequency and time: the higher the number of scales considered, the finer is the scale discretization.

The DWT is often used for denoising and compression of signals because it can represent them with few coefficients. The CWT is instead often used in time-frequency analysis and filtering of time-localised frequency components.

Multi-Head Attention Attention mechanism can be described as mapping a query and a set of key-value pairs to an output. Queries, keys, values and outputs are vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is given by an arbitrary compatibility function of the query with the corresponding key. According to [94], the selected compatibility function, is the *Scaled Dot-Product Attention*.

Firstly, each of the timesteps in input is linearly projected to obtain its specific query, key and value vectors of dimension d_k. Next, given a query, the dot product of the query with all keys is computed. Then, these products are divided by $\sqrt{d_k}$. Finally, the softmax function is applied to obtain the weights on the values. These weights can be seen as scores, thus they represent the importance of the values (each corresponding to one multiplied key) with respect to the value corresponding to the query. Intuitively, a subset of more important times receives high weights, while useless ones receive lower weights. At this point, weighted values are summed up.

The explained calculation is valid only when there is a single query. In practice, as we have seen, there are a number of queries equal to the number of timesteps in the selected time window, i.e. T_w. Therefore, in order to speed-up the computation, the scaled dot-product attention is computed on a set of queries of queries simultaneously, packed together into a matrix Q. If we do the same with the keys and values, the output can be expressed as:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.6}$$

Multi-head attention mechanism (Figure 3.6) simply repeats the above computation a number of times equal to the chosen number of heads, h. More precisely, instead of calculating a single attention function with one set of queries, keys and values, this mechanism first creates h different sets of queries, keys and values and for each of them performs the attention function in parallel.

The outputs of each head are concatenated and the final result is linearly projected in order to obtain the matrix of shape (T_w, N_x).

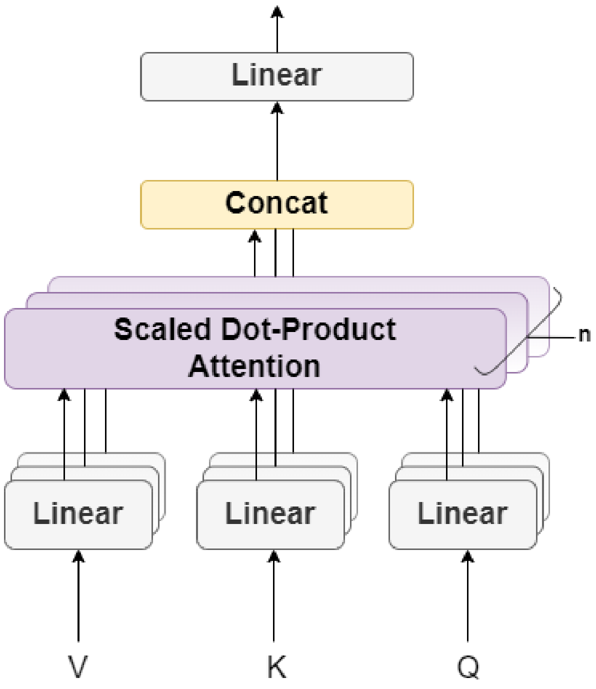


Figure 3.6: Multi-Head Attention.

Mathematically, multi-head attention is defined as:

$$MultiHead(Q,K,V) = Concat(head_1, ..., head_h)W^O \tag{3.7}$$

where

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{3.8}$$

At a higher level of abstraction, the multi-head attention sub-module computes a new representation of the input time window. In this representation, each timestep is enriched by the knowledge of the timesteps that precede or follow it in the sequence.

4 Time series encodings evaluation for predictive maintenance

4.1 Introduction

The individual rapidly navigating the digital technology landscape has been instrumental in transforming industrial processes due to the deep integration between physical and digital systems within production environments. In the present day, this individual possesses the capability to amass substantial volumes of data concerning the functioning of diverse equipment, all while facilitating targeted exchanges of information among individuals, products, and machines.

The increasing pace of transformative technological advancements has led experts to characterize this new phase of development as the "Fourth Industrial Revolution" or "Industry 4.0," linking it to high connectivity, the availability of rich data sources, and the capacity of technologies to explore the high dimensionality of such sources, thanks to enhancements in computational power and storage capacity [95]. For example, this individual can analyze in real-time the multitude of events unfolding along an industrial production line, establishing correlations between real-time data and past occurrences to identify and proactively mitigate potential structural failures, thereby averting prolonged downtime.

In a more general context, data that is rich and high-dimensional in real-time can yield valuable insights into the internal dynamics of intricate industrial systems. In this regard, this individual has uncovered the incredible potential of applying data analytics techniques to the industrial pipeline across various domains, including cost reduction in maintenance, minimization of machine faults, decreased repair stoppages, inventory reduction in spare parts, extended lifespan of spare parts, augmented overall production output, enhanced operator safety, validation of repair actions, overall profit augmentation, and numerous other applications [96, 97].

Notably, most of these challenges are intrinsically connected to the timely execution of efficient and effective maintenance procedures.

A particular area of interest for this individual undoubtedly lies in the realm of condition monitoring and diagnostics concerning mechanical components within industries such as avionics and automotive. This encompasses elements such as gearboxes, ball bearings, and rotating shafts [98]. In these domains, a sudden interruption in the production line carries costs associated with the loss of product that far surpass the expenses related to the component itself. Another intriguing domain where maintenance procedures hold significant importance is in the management of Information Technology (IT) infrastructures, particularly in the prediction of hard disk failures within large-scale data centers. The disruption of hard disks within these data centers directly impacts the reliability of the entire infrastructure, thereby adversely affecting the business's Service Level Agreement [99].

To address these challenges, Predictive Maintenance techniques have become indispensable for ensuring substantial improvements in business operations, harnessing the technological advancements of Industry 4.0 to minimize downtime and reduce equipment failure rates across diverse contexts [100–104].

As is often the case in fields dealing with vast and complex data, this individual has found that approaches leveraging machine learning and deep learning tools hold the most promise among the array of modern predictive maintenance techniques [101, 105, 106]. Typically, these approaches rely on historical datasets organized as labeled time series data concerning equipment operations, enabling the training of various regression and classification models. These models can then be utilized to predict potential failures in terms of "Remaining Useful Life" (RUL) estimation. It is evident that the effectiveness of these approaches is closely tied to the availability of comprehensive and reliable training data.

Given that this is not always the case in real-world scenarios, deep learning models have garnered attention as a means to address potential limitations arising from data scarcity [107]. Notably, many reliable deep learning architectures have been developed with a primary focus on image analysis. Consequently, in recent years, there has been a surge in research efforts aimed at encoding time series data as images and reimagining RUL prediction as an image classification task [108].

In the present work, this individual presents a framework for evaluating the performance of several widely used time series encoding techniques, including Recurrence Plot, Gramian Angular Field, Markovian Transition Field, and Wavelet Transform, in conjunction with image classifiers based on Convolutional Neural Networks (CNNs). These CNN models are subsequently compared with three benchmarking deep learning models using the PAKDD2020 Alibaba AI Ops Competition dataset, which provides data on hard disk status within a data center, and two state-of-the-art models using the NASA bearing dataset, consisting of vibration signals from bearings. The experimental evaluation highlights that the utilization of CNNs, with inputs generated through encoding techniques, delivers high effectiveness performance, surpassing most state-of-the-art models and demonstrating superior Memory Occupation parameters. The advantages of appropriate data augmentation techniques, including those based on Generative Adversarial Networks (GANs), are discussed, and the results underscore the benefits and drawbacks of various modeling and training choices. In particular, it is shown that while the incorporation of GANs enhances the training process and slightly improves performance, this advantage must be weighed against increased demands on training time and memory resources.

To the best of this individual's knowledge, this work represents one of the earliest comprehensive studies reporting a systematic benchmark of a variety of extensively used time series encoding techniques as viable models for predictive maintenance tasks.

In summary, the main contributions of this approach can be summarized as follows:

- Designing a comprehensive framework for assessing the performance of prevalent time series encoding techniques in predictive maintenance tasks.
- Utilizing two distinct CNN-based models for predicting equipment failures, with inputs generated through different encoding techniques.
- Conducting a comparative analysis of encoding-based techniques versus various state-of-the-art approaches using two real-world datasets (PAKDD2020 Alibaba AI Ops Competition and NASA bearing).
- Analyzing the performance impact of GANs as a data augmentation strategy.

In conclusion, this work sheds light on the challenges and opportunities in predictive maintenance within the context of Industry 4.0, offering valuable insights into the efficacy of different techniques and models in enhancing equipment reliability and reducing downtime.

The chapter's structure is as follows. Section 4.2 provides an overview of related research concerning predictive maintenance techniques, with particular emphasis on recent machine-based approaches. The 4.3 section outlines the proposal and evaluation of the proposed framework. Section 4.4 presents the findings from the experiments conducted to assess the various encoding techniques introduced in this chapter. Lastly, in Section 4.6, the chapter concludes by summarizing the results and discussing potential avenues for future research.

4.2 Related Work

Scheduling maintenance decisions is a critical task aimed at preventing unforeseen shutdowns of mechanical equipment, ultimately enhancing their reliability [105]. Predictive maintenance has gained significance in the industry by leveraging machine learning models to analyze large volumes of operational data, thereby improving equipment efficiency and reducing operational costs by using machine learning models [105, 109, 100, 110]. Advances in artificial intelligence (AI) have been pivotal in enhancing the reliability of predictive maintenance [111, 52, 112]. Deep learning techniques, which require labeled datasets consisting of equipment operation trajectories typically encoded as time series data, have been instrumental in estimating Remaining Useful Life (RUL) [101, 105, 106, 113].

Recent studies have successfully applied various models for RUL prediction across different domains. For instance, some have employed autoregressive models such as AR-RPF to predict the RUL of lithium-ion batteries [114]. Others have utilized support vector machine (SVM) models for fault detection and diagnosis in chillers [115]. Predicting RUL of bearings, common mechanical components in various equipment, has also been a focus [116, 117]. Early approaches relied on regression strategies [118, 119]. Some more recent approaches focused on deep learning models using vibration signals from bearings [120, 121]. Predicting Hard Disk Drive (HDD) RUL using S.M.A.R.T. attributes has been explored as well [122–124, 56].

Deep learning models' performance is highly dependent on the availability of extensive, consistent, and reliable training data [125]. Addressing data scarcity in real-world scenarios, some research has explored the use of deep learning models [107]. Given that extensive work on deep learning models has taken place in the domain of image analysis, recent work has experimented with encoding time series data as images [126].

Various encoding techniques have been developed to represent temporal correlations as images, which can be used to train deep learning models for a wide range of applications. Four prominent encoding methods include the Gramian Angular Field (GAF), Discrete Wavelet Transform (DWT), Markov Transition Field (MTF), and Recurrent Plot (RP). GAF encoding has been used for tasks such as solar irradiation forecasting, knowledge distillation, activity recognition, and financial forecasting [127–130]. In addition, GAF-based methods have been applied to predictive maintenance for conveyor motors [131]. DWT has been utilized for fault detection in gearboxes and arrhythmia detection using ECG signals [132–135]. MTF encoding has been employed for tasks such as candidate transient classification and arrhythmia classification [136, 137]. RP encoding has been used for human activity recognition and Parkinson's disease diagnosis [138–140].

Despite these successful applications in various domains, there is a noticeable gap in the literature concerning the use of encoding techniques for equipment failure prediction, especially for HDD and bearing health status prediction, which are commonly studied cases in predictive maintenance. This study aims to investigate the application of encoding strategies to predictive maintenance tasks with the goal of reducing memory and training time requirements while achieving performance comparable to state-of-the-art approaches. The research focuses on multidimensional time-series data typically generated in industrial settings and explores various encoding techniques, unlike most previous approaches that primarily analyze one-dimensional signals using a single encoding strategy.

Paper	Model	Time Series	Encoding	Application
[127]	Convolutional LSTM	One-dimensional	GAF	Solar Irradiation Forecasting
[128]	CNN	One-dimensional	GAF	Knowledge distillation
[129]	CNN	One-dimensional	GAF	Activity Recognition
[130]	CNN	One-dimensional	GAF	Financial forecasting
[131]	CNN	One-dimensional	GAF	Conveyor Motor maintenance
[59]	CNN	Multi-dimensional	GAF	Hard drives health status
[141]	CNN	Multi-dimensional	GAF, MTF	Sensor classification
[142]	CNN+SVM	Multi-dimensional	GAF	Electricity consumption forecasting
[143]	CNN	Multi-dimensional	GAF	Stock Market Forecasting
[132]	CNN	One-dimensional	DWT	Gearboxes fault diagnosis
[133]	CNN	One-dimensional	DWT	Gearboxes fault diagnosis
[135]	ESN	One-dimensional	DWT	Time-series forecasting
[134]	NCA+1NN classifier	One-dimensional	DWT	Arrhythmia detection
[136]	CNN	One-dimensional	MTF	Light curves classification
[137]	CNN	One-dimensional	MTF	Arrhythmia classification
[144]	CNN	One-dimensional	MTF	Anomalous energy consumption
[145]	CNN	One-dimensional	MTF	Online Fraud Detection
[138]	CNN	Multi-dimensional	RP	Activity Recognition
[139]	CNN	Multi-dimensional	RP	Disease identification
[140]	CNN	One-dimensional	RP	Classification task

Table 4.1: State-of-the art approaches classified on the basis of the neural network model adopted, the encoding method, and their application domain. Encoding methods are: Gramian Angular Field (GAF), Recurrence Plot (RP), Markovian Transition Field (MTF), and Wavelet Transform (WT). NCA and 1NN stand respectively for Neighborhood Component Analysis and 1-Nearest Neighborhood.

In the remainder of this paper, a framework is presented to evaluate four time series encoding techniques in the context of predictive maintenance tasks, integrating them with CNN-based image classifiers. The evaluation is conducted using the Alibaba and NASA bearing datasets, comparing the performance of CNN models to alternative machine learning architectures [126].

4.3 Framework

An overview of the training and evaluation framework is depicted in Figure 4.1. The primary objective is to establish a benchmark for various techniques that encode time-series data into images, with a specific focus on their performance in equipment failure prediction.

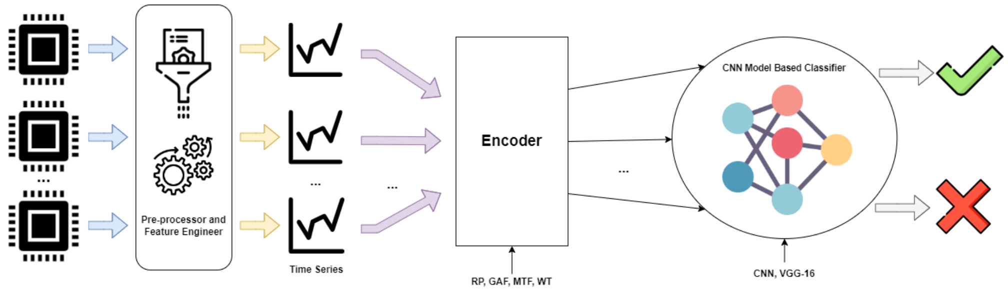

Pre-processor and Feature Engineer
Time Series
Encoder
RP, GAF, MTF, WT
CNN Model Based Classifier
CNN, VGG-16

Figure 4.1: Architectural overview of the proposed framework , that is composed by different phases. The initial step is devoted to pre-processing and feature engineering, the second step is to create the time series sequences and convert them into images, choosing the technique to be used in the encoding module.

The initial phase involves a series of pre-processing and feature engineering operations applied to the dataset. These operations encompass the removal of attributes containing missing values and columns that do not contribute to equipment failure prediction, such as capacity or manufacturing-related features. This results in a reduction in the overall number of features.

Additionally, various rebalancing and feature transformation techniques are applied using different methodologies, as detailed in Section 4.4.3.

In the subsequent phase, time-series sequences are constructed and fed into one of the image encoding methods outlined in Section II. This process generates the input data for the subsequent Convolutional Neural Network (CNN) designed for the specific time window under consideration. Two types of CNN models are trained to address the predictive maintenance task. Furthermore, a focused investigation is carried out to determine whether the utilization of a Generative Adversarial Network (GAN) has any impact on the overall performance.

4.3.1 CNN-based Classifiers

The dataset naturally yields time series of varying lengths, necessitating the creation of time series sequences based on fixed time windows (40 steps). These sequences are then suitable for input into one of the encoding techniques discussed in Section 4.4.3. This step generates image encodings, which serve as input data for a Convolutional Neural Network (CNN) model. Two distinct CNN models are considered for this purpose.

The first model comprises three convolutional layers utilizing the leaky ReLU activation function, each succeeded by a max-pooling layer with a specific filter size. At the top, there is a fully connected layer with softmax activation (Model 1; Figure 4.2).

Secondly, a CNN model based on the VGG-16 architecture [146] is considered. This pre-trained model is designed to process RGB images of dimensions $(224, 224)$. It comprises two convolutional layers with 64 and 128 filters, followed by a max-pooling layer. The third block consists of three convolutional layers with 256 filters and a max-pooling layer. At the top of this architecture, there are two fully connected layers, each containing 2048 neurons and employing softmax activation (VGG-like; Figure 4.3). This

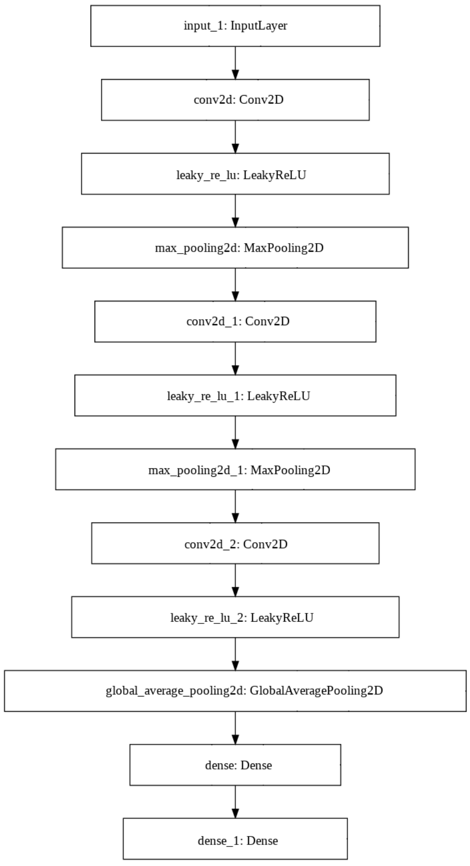


Figure 4.2: First CNN architecture

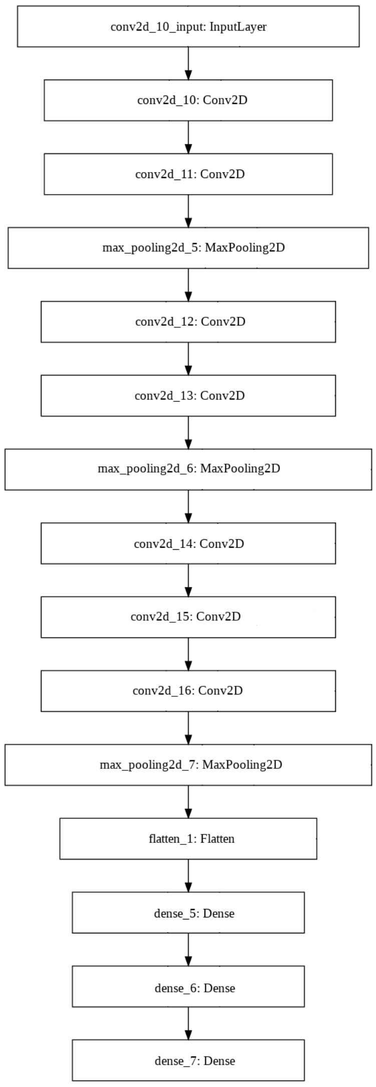


Figure 4.3: VGG-like architecture

choice of a shallower CNN allows for more efficient handling of smaller inputs, such as those sized 40×40.

Finally, the chosen loss function is the **log-loss** (also known as **cross-entropy loss**). This loss function returns a probability ranging from 0 to 1 for the two task classes, as described by equation 4.1.

$$L_i = -(y \cdot \log(p) + (1-y) \cdot \log(1-p)) \tag{4.1}$$

In this equation, y represents the correct label, p denotes the probability for the correct label, and L_i signifies the loss for the i-th element predicted by the classifier. Additionally, the *Adam* optimizer [147] is employed due to its ability to facilitate smoother gradient descent and prevent convergence to local optima. The Adam optimizer introduces two supplementary parameters known as the first and second moment. The former serves as a velocity term, reflecting a combination of historical information and the current value, while the latter acts as an energy term representing recent movements.

4.4 Experiments

Now that all the technical infrastructure is set up, the focus can shift to the heart of the evaluation task. The primary objective is to assess the performance of various methodologies with the aim of optimizing effectiveness, thereby minimizing both false positives and false negatives, while also considering efficiency. Consequently, different combinations of encoding methods and architectures are assessed for their effectiveness, using the metrics outlined in Section 4.4.4, and their efficiency, as measured by each model's memory usage and training time.

4.4.1 Experimental protocol

As a foundation for the evaluation, two distinct datasets were selected to explore the utilization of encoding methods in various industrial applications. The chosen approaches were those that yielded the highest effectiveness values.

The first dataset, provided by Alibaba for the PAKDD 2020 AIOps Competition[1], pertains to Hard Disk Drives (HDDs). Following a preprocessing and feature engineering phase, the dataset's size was reduced to approximately 420 MB. This dataset comprises over 150,000 tensors, each sized at 40×76, where 40 represents the window size, and 76 corresponds to the number of features. To enrich the dataset, additional features were generated by applying five different methods (Raw, Normalized, Shift, Absolute, and Relative) to the raw S.M.A.R.T. attributes. The dataset was partitioned such that 60% of it served as the training set, 20% as the validation set, and 20% as the testing set. Balancing was applied to ensure an equal number of healthy and failed disks.

The second dataset, referred to as the *NASA Bearing* dataset[2], comprises a total of 19,680,000 data points divided into 984 files, each containing 20,000 samples. After the preprocessing phase, a window of size five was applied to each file to generate encoded images. Consequently, the result of this processing is 984 images, each depicting the health state of a bearing at a specific moment during its operation. The dataset was divided into three parts using a *Stratified* approach: 60% and 30% for the training and test sets, respectively, and 10% for evaluating generalization performance (see Section 4.4.3 for further details).

4.4.2 Dataset

As previously mentioned, the predictive maintenance of equipment's health status holds paramount importance in large-scale industrial infrastructures,

Parameters	**Values**
Windows size	(1,3,5,7,10,15,20,30)
Epoch	[20 - 300]
Learning Rate	0.1,0.01,0.001
% Fake	25%,30%
% GAN module	Yes,No

Table 4.2: Hyper-parameters optimization phase.

as equipment failures can profoundly impact the overall reliability of such infrastructures. Consequently, the evaluation of the various methodologies discussed thus far is primarily focused on the task of predicting potential failures of equipment within a predefined time window (e.g., 30 days). This prediction relies on the analysis of time series data.

To facilitate this evaluation, two distinct datasets were selected, each representing predictive maintenance tasks in different industrial contexts. These contexts mainly revolve around bearings, which are subject to high-speed and high-pressure conditions, and Hard Disk Drives (HDDs), a critical component affecting the reliability of large-scale data centers. Specifically, a dataset from the *PAKDD2020 Alibaba AI OPS*[1] competition was leveraged. It comprises approximately 40 GB of samples collected from July 2017 to July 2018. The dataset includes S.M.A.R.T. attributes, providing both raw and normalized values for each disk per day, along with labels and failure timestamps.

Furthermore, the *NASA Bearing* dataset[2] was utilized, with details outlined in Table 4.3. This dataset primarily focuses on the analysis of vibration signals captured using accelerometers along the X-axis. These vibration signals were recorded with a 1-second time window at 10-minute intervals, with a sampling rate of 20 KHz. Subsequently, noise reduction and data

[1]https://tianchi.aliyun.com/competition/entrance/231775/introduction
[2]https://ti.arc.nasa.gov/c/3/

Bearing	**No.of Samples**	**No. of raw features**	**Conditions**
Bearing 1	984	20480	Outer race failure
Bearing 2	984	20480	No defect
Bearing 3	984	20480	No defect
Bearing 4	984	20480	No defect

Table 4.3: NASA Bearing dataset

normalization were performed, employing a moving average technique and *Min-max Normalization*, respectively.

4.4.3 Pre-processing and Feature engineering

The pre-processing of the Alibaba dataset involved two consecutive phases. Initially, a set of relevant features was selected, reducing the total from over 500 to 32. This selection process began by eliminating attributes with a missing value percentage exceeding 10% and a standard deviation of 0. Columns that were deemed non-critical for failure prediction were also dropped. Any remaining missing values were imputed using a moving average with a window size of 5 steps backward and 5 steps forward. Additionally, the dataset's overall balance between healthy and unhealthy disks was adjusted, shifting from 1% to 50% healthy disks by reducing the number of healthy disks.

Regarding the NASA Bearing dataset, a down-sampling operation was applied, reducing the sampling rate from 20KHz to 4KHz while employing a window size of five. Since this dataset lacks labels, a visual analysis was performed to identify the moment of failure, following the experimental procedure outlined in [148]. Vibration data was classified into three categories: *HIGH-RISK* for samples near the point of failure, *LOW-RISK* for samples representing normal behavior, and *MEDIUM RISK* for a state with a hypothetical medium risk of failure. In summary, the samples in the NASA Bearing dataset were categorized as follows:

- Samples from 2004-02-12 10:32:39 to 2004-02-17 10:52:39 are labeled as LOW-RISK
- Samples from 2004-02-17 10:52:39 to 2004-02-18 13:52:39 are labeled as MEDIUM-RISK
- Samples from 2004-02-18 13:52:39 to 2004-02-19 06:22:39 are labeled as HIGH-RISK

The prediction task was further designed as a binary problem, where the union of the classes *HIGH-RISK* and *MEDIUM-RISK* was considered as a single class.

Next, the raw features were converted into normalized features using the following methods:

- **Shift features (Shift)**: The original raw features ($V(n)$) were shifted by N days ($V(n-N)$), with different values of $N = 1,3,5,7,10,15,20,30$.
- **Relative comparison features (Diff)**: The difference between a raw feature ($V(n)$) and its corresponding shifted feature ($V(n-N)$) was computed as follows:

$$Relative(n)[N] = V(n) - V(n-N) \tag{4.2}$$

- **Absolute comparison features (Sum)**: The sum of a raw feature ($V(n)$) and its corresponding shifted feature ($V(n-N)$) was calculated as follows:

$$Absolute(n)[N] = V(n) + V(n-N) \tag{4.3}$$

- **Exponential moving average features (Exp)**: This transformation was applied to *S.M.A.R.T.* raw features for each disk, according to equation 4.4:

$$history(n) = 0.9 \times history(n-1) + 0.1 \times raw(n)\ history(-1) = 0 \tag{4.4}$$

where $raw(n)$ is the current value of $S.M.A.R.T._{raw}$ at the $n-th$ step and *history* is the cumulative weighted sum of historic data.

- **Division features (Div)**: These features represent the ratio between raw and normalized features, as shown in equation 4.5:

$$Division(n) = \frac{S.M.A.R.T.raw(n)}{S.M.A.R.T.Normalized(n) + \varepsilon} \tag{4.5}$$

where ε is a constant used to avoid division by zero.

It is important to note that the number inside the parenthesis in the next tables corresponds to the shifted feature in terms of number of days.

A grid search was performed using the hyper-parameters shown in Table 4.2 to identify the optimal ones for training the models. To validate the results statistically, a 10-cross validation [149, 150] was executed, reporting the mean and standard deviation of each experiment outcome. A stratified sampling strategy was also employed to split the dataset into training and test sets.

The evaluation framework was deployed on Google Colaboratory[3] using TensorFlow V2[4] and Keras[5] for building deep learning models. The preprocessing operations and running of the time series classification algorithms were performed using pyts[6].

[3]https://colab.research.google.com/
[4]https://www.tensorflow.org/
[5]https://keras.io/
[6]https://pyts.readthedocs.io/

	Model 1			**VGG-like**		
P	**F1-score**	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**
15	39.64±0.31	33.04±1.78	31.07±2.34	28.44±0.79	32.23±0.19	27.67±2.41
30	**59.24± 0.39**	**61.15±3.18**	**57.62±2.61**	**31.59±1.25**	**29.58±0.22**	**34.03±3.13**
45	47.94±1.41	42.77±3.47	48.08±2.89	46.67±2.09	43.01±1.74	48.13±3.85

Table 4.4: Evaluation of the both networks varying the P parameter.

4.4.4 Evaluation metrics

In this section, several metrics are described that are used to evaluate the efficiency of the proposed framework, which is defined as the ability to assess the equipment's health status within a 30-day interval. Specifically, a P-window (set to 30 days - further details in table 4.4) is defined as a fixed-size sliding window starting from the first moment in which a disk is predicted to fail.

Precision for P-window: the fraction of records that actually failed (TP) and the fraction expected overall ($TP+FP$):

$$Precision = \frac{TP}{TP+FP} \tag{4.6}$$

where TP and FP are respectively true and false positives.

Recall for R-window: the fraction of predicted failed disks that actually failed (TP) over the overall number of failed disks ($TP+FN$):

$$Recall = \frac{TP}{TP+FN} \tag{4.7}$$

where TP and FN are respectively true and false negatives.

F1-score is defined according to equation 4.8:

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{4.8}$$

4.5 Results

In this section, the obtained results on the *Alibaba HDD* and *NASA bearing* datasets are discussed. Finally, the performance of the discussed methodology when adopting a Generative adversarial Networks (GAN) is investigated.

4.5.1 Results on Alibaba HDD

The two CNN models described in Section 4.3.1 are compared. The aim is to understand how each model performs based on the best fit between different encoding techniques — (RP, GAF, MTF, WT) — and pre-processing approaches (see Section 4.4.3). To summarize, two different CNN networks (one custom and another pretrained) are compared to effectively exploit the images generated by using the encoding strategies. The goal is to compare two different strategies (pretrained vs custom) while varying the different encoding methods. Table 4.5 shows the performance of both models in terms of memory usage and overall training time, where Model 1 achieves the best results independently of the encoding method. This result is due to the larger number of parameters to be optimized within the VGG-like model, resulting in a larger increase in network training time.

Model	Memory(kB)	Training time(secs/epoch)
Model 1	**470**	**13.7±0.1**
VGG-like	73.000	37.4±0.2

Table 4.5: Memory usage and training time for both models (independently of encoding method)

The overall performance of Model 1, a CNN, is influenced by different combinations of encoding techniques and feature engineering approaches, as shown in Table 4.6. The highest precision is achieved by *GASF+Exp(15)*, but it also leads to a large number of True Negatives. On the other hand,

Technique	F1-score	Precision	Recall
RP + Sum(1)	22.96±1.34	16.52±0.56	37.61±4.34
MTF + Diff(1)	21.64±0.55	15.74±0.22	34.82±2.22
GADF + Diff(7)	**32.50±1.44**	24.98±0.64	**46.41±3.76**
GASF + Exp(15)	31.04±2.09	**28.03±0.46**	35.01±4.84
WV + Exp(30)	26.67±1.46	21.73±0.31	35.01±5.11

Table 4.6: Performance of Model 1, based on the different image encoding techniques and pre-processing approaches used to generate its input. It is important to note that the number inside the parenthesis in the next tables corresponds to the shifted feature in terms of number of days.

Technique	TP	FP	FN	TN
RP + Sum(1)	30	155	49	127
MTF + Diff(1)	28	148	52	133
GADF + Diff(7)	50	150	60	101
GASF + Exp(15)	30	75	57	199
WV + Exp(30)	27	100	54	180

Table 4.7: Model 1: Confusion matrices (median values over repeated tests)

GADF+Diff(7) produces the best results in terms of F1-Score, but with a high number of False Positives, as indicated in Table 4.7.

Moving on to the second CNN model, the results achieved by the VGG-16-like architecture for different combinations of feature engineering methods and encoding techniques are shown in Table 4.8. It is evident that the combination of *GASF+Exp(7)* yields the highest F1-score and Precision, although it identifies a large number of True Positives. On the other hand, *RP+Sum(1)* achieves the highest Recall score but returns a significant number of False Positives (Table 4.9).

Finally, the performances of both models on six different types of faults according to the tag field into the *PAKDD2020 Alibaba AI Ops Competition*[1] were investigated. The results of the investigation can be seen in Table 4.10 and 4.11. It should be noted that *GAF* achieved the highest results by using

Technique	F1-score	Precision	Recall
RP + Sum(1)	22.17±1.52	15.15±0.70	**41.68±5.46**
MTF + Diff(1)	20.87±0.82	14.04±0.28	41.29±4.40
GADF + Diff(7)	29.63±1.17	23.53±0.23	40.23±3.62
GASF + Exp(15)	**31.59±1.25**	**29.58±0.22**	34.03±3.13
WV + Exp(30)	26.29±1.85	22.32±0.27	32.38±5.04

Table 4.8: Performance of the VGG-like model, based on the different image encoding techniques and pre-processing approaches used to generate its input.

Technique	TP	FP	FN	TN
RP + Sum(1)	34	182	42	103
MTF + Diff(1)	29	183	42	107
GADF + Diff(7)	45	146	66	104
GASF + Exp(15)	31	74	61	195
WV + Exp(30)	27	95	58	181

Table 4.9: VGG-like architecture: Confusion matrices (median values over repeated tests)

difference and exponential features over 7 and 15 days respectively. This can be attributed to how the encoding method handles the distribution of features over time from different perspectives (GADF and GASF), which allows for the representation of time series data in multi-channel images.

The performance metrics of Model 1 on the six available fault types ($[0,5]$) based on various coding techniques and pre-processing approaches are displayed in Table 4.11.

It is evident from the results that Model 1 surpasses VGG-like, as anticipated, due to its optimization for the specific tasks required.

The comparison between our best-performing CNN model, using GASF as the image encoding method, and alternative machine learning approaches, including LSTM, GRU, XGBoost, ResNet-50, DenseNet-121, and VGG-16, was conducted to evaluate their effectiveness in the predictive maintenance task.

The LSTM and GRU models each consist of two layers with 64 units and a final softmax activation layer. These recurrent neural networks were chosen for their reported performance in similar tasks. The XGBoost model, on the other hand, is a gradient boosting algorithm based on decision trees, known for its effectiveness in machine learning tasks.

In addition to the original dataset features (S.M.A.R.T. raw and normalized), we also utilized some of the generated features (Shift, Relative, and Absolute) to maximize model performance.

To assess the performance of these models, we compared them against our CNN-based Model 1, which used GASF as the image encoding method. This combination yielded the best results in our evaluation. The results of this comparison are presented in Table 4.6.

Table 4.12 provides a comparison between our best model and the selected benchmark models. Notably, Model 1 outperforms the others in terms of F1-score and Precision, while the LSTM model exhibits the highest Recall.

Fault	Metrics	Technique				
		RP + Sum(1)	**MTF + Diff(1)**	**GADF + Diff(7)**	**GASF + Exp(15)**	**WV + Exp(30)**
0	**F1-score**	22.76±1.33	20.77±0.51	32.45±1.41	31.02±2.08	26.65±1.49
	Precision	16.26±0.55	15.82±0.19	24.98±0.62	28.09±0.43	21.71±0.32
	Recall	37.61±4.37	34.91±2.19	46.33±3.75	34.97±4.84	34.98±5.09
1	**F1-score**	23.67±1.35	22.69±0.59	32.52±1.47	31.06±8.10	26.76±1.45
	Precision	16.97±0.57	15.85±0.22	25.02±0.63	27.98±0.49	21.75±0.34
	Recall	37.51±4.30	34.67±2.25	46.57±3.79	35.08±4.86	35.07±5.13
2	**F1-score**	22.45±1.36	21.47±0.53	32.63±1.45	31.11±8.09	26.67±1.46
	Precision	16.65±0.58	15.62±0.23	24.68±0.64	28.01±0.48	21.76±0.33
	Recall	37.21±4.34	35.06±2.24	46.21±3.78	35.01±4.82	35.02±5.12
3	**F1-score**	22.75±1.32	21.76±0.56	32.42±1.43	31.38±2.11	26.59±1.46
	Precision	16.54±0.56	15.61±0.24	25.08±0.62	28.03±0.45	21.74±0.29
	Recall	37.72±4.33	34.62±2.22	46.63±3.74	35.02±4.83	35.01±5.10
4	**F1-score**	23.63±1.37	21.70±0.54	32.47±1.42	30.9±2.07	26.63±1.47
	Precision	16.84±0.53	15.87±0.21	25.03±0.66	28.02±0.45	21.69±0.31
	Recall	37.86±4.37	34.89±2.21	46.37±3.74	35.03±4.81	35.04±5.12
5	**F1-score**	22.44±1.31	21.45±0.57	32.51±1.46	30.77±2.09	26.72±1.43
	Precision	15.86±0.57	15.67±0.23	25.13±0.67	28.05±0.46	21.73±0.27
	Recall	37.75±4.33	34.77±2.21	46.35±3.76	34.95±4.88	34.94±5.10

Table 4.10: Performance of Model 1 according to six different HDD fault types, based on the different images technique and pre-processing approaches used to generate its input.

Fault	**Metrics**	**Technique**				
		RP + Sum(1)	**MTF + Diff(1)**	**GADF + Diff(7)**	**GASF + Exp(15)**	**WV + Exp(30)**
0	**F1-score**	22.13±1.56	20.86±0.66	29.60±1.15	31.57±1.25	26.22±1.85
	Precision	15.13±0.66	13.95±0.28	23.65±0.23	29.65±0.21	22.31±0.26
	Recall	41.54±5.44	41.26±4.15	40.26±3.65	34.02±3.12	32.33±5.02
1	**F1-score**	22.18±1.61	20.89±0.67	29.65±1.14	31.60±1.24	26.34±1.84
	Precision	15.18±0.67	13.97±0.31	23.67±0.21	29.67±0.22	22.36±0.28
	Recall	41.67±5.41	41.27±4.46	40.24±3.58	34.04±3.16	32.42±5.05
2	**F1-score**	22.21±1.48	20.87±0.71	29.67±1.22	31.56±1.23	26.24±1.83
	Precision	15.13±0.71	14.05±0.26	23.54±0.26	29.54±0.23	22.31±0.27
	Recall	41.79±5.61	41.29±4.55	40.22±3.74	34.02±3.14	32.42±5.07
3	**F1-score**	22.11±1.51	20.86±0.72	29.63±1.11	31.58±1.26	26.58±1.86
	Precision	15.14±0.72	14.09±0.27	23.52±0.27	29.62±0.20	22.35±0.29
	Recall	41.87±5.43	41.31±4.45	40.25±3.75	34.02±3.11	32.37±5.03
4	**F1-score**	22.22±1.53	20.91±0.69	29.61±1.22	31.61±1.24	26.17±1.84
	Precision	15.17±0.69	14.17±0.29	23.39±0.22	29.49±0.23	22.33±0.25
	Recall	41.66±5.45	41.36±4.37	40.21±3.56	34.01±3.12	32.36±5.04
5	**F1-score**	22.17±1.43	20.83±0.71	29.62±1.18	31.62±1.28	26.19±1.83
	Precision	15.15±0.71	14.01±0.27	23.41±0.25	29.51±0.23	22.32±0.27
	Recall	41.55±5.42	41.25±4.42	40.20±3.44	34.07±3.13	32.38±5.03

Table 4.11: Performance of VGG-like according to six different HDD fault types, based on the different images technique and pre-processing approaches used to generate its input.

Model	**F1-score**	**Precision**	**Recall**
XGBoost	40.19±0.60	30.03±0.41	60.85±1.02
LSTM	52.51±1.32	42.87±1.73	**67.79±2.24**
GRU	51.73±1.81	41.47±1.85	66.81±2.45
VGG-16	52.23±1.74	42.17±1.81	67.21±2.25
ResNet-50	51.91±1.71	41.38±1.75	66.92±2.32
DenseNet-121	51.22±1.83	41.16±1.87	66.24±2.47
CNN Model 1	**59.24±0.39**	**61.15±3.18**	57.62±2.61

Table 4.12: Performances of the CNN Model 1 with respect to six state-of-the-art ones.

Model	**TP**	**FP**	**FN**	**TN**
XGBoost	83	136	51	161
LSTM	96	127	44	166
GRU	89	122	50	170
VGG-16	90	121	51	169
ResNet-50	88	121	55	167
DenseNet-121	87	123	53	168
CNN Model 1	103	67	71	190

Table 4.13: Confusion matrices (median values over repeated tests)

Model	**Memory usage**	**Training time (seconds)**
XGBoost	7 MB	780 (2000 estimators)
LSTM	850 kB	6 s/epoch (best at 10-th epoch)
GRU	767 kB	**5 s/epoch** (best at 15-th epoch)
VGG-16	8 MB	12 s/epoch (best at 21-th epoch)
ResNet-50	11 MB	14 s/epoch (best at 19-th epoch)
DenseNet-121	15 MB	8 s/epoch (best at 26-th epoch)
CNN Model 1	**540 kB**	91 s/epoch (best at 25-th epoch)

Table 4.14: Memory usage and training time

Encoding Techniques	Accuracy	Precision	Recall	F1-Score
GAF+Diff(7)	0.80±0.02	0.84±0.01	0.70±0.02	0.75±0.02
MTF+Exp(7)	0.75±0.01	0.74±0.02	0.74±0.01	0.74±0.02
RP+ Exp(15)	**0.87±0.02**	**0.86±0.01**	**0.88±0.01**	**0.83±0.01**
WV+Exp(30)	0.79±0.02	0.76±0.02	0.73±0.01	0.75±0.02

Table 4.15: Performances | 3-class classification

Moreover, our CNN model achieves higher numbers of true positives and true negatives compared to the XGBoost, GRU, LSTM, VGG-16, ResNet-50, and DenseNet-121 models (Table 4.13).

Regarding memory usage and training time (Table 4.14), our model demonstrates better efficiency in memory usage compared to the benchmark models. However, the GRU model proves to be the fastest in terms of training time.

In summary, Model 1 excels in both efficacy and efficiency compared to the VGG-based network, mainly because the latter is a pre-trained network.

4.5.2 Results on NASA Bearing

In this section, the experimental results involving encoding techniques for generating input images for CNN classification are discussed. The results are presented for both labeling procedures, including binary and three-class classification.

Three Classes Classification Results

Starting with the analysis of three-class classification results, the CNN's output provides the probability of a sample belonging to one of the three classes. Table 4.15 clearly indicates that the *Recurrence Plot* technique consistently outperforms others in all evaluated metrics.

Moreover, the results presented in Table 4.15 are substantiated by the examination of confusion matrices generated from the model's predictions using different encoding techniques on the test set. These matrices are illustrated in Figure 4.4, 4.5, and 4.6.

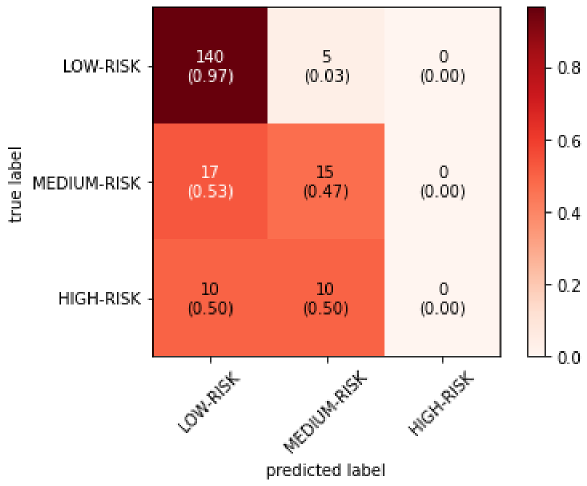


Figure 4.4: GAF Confusion Matrix | 3-class classification.

Specifically, it is evident that the network accurately identifies the first class (LOW-RISK) when employing the *RP* encoding technique. However, it struggles to distinguish between the other two classes (MEDIUM-RISK and HIGH-RISK), as these classes exhibit similarities in their vibration signal characteristics. This challenge arises due to the complexity of discerning differences between these closely related classes.

As observed in Figure 4.7, the loss curve exhibits a typical pattern with a plateau that converges to approximately 0.4 for the validation subset. This suggests that the network has been adequately trained, but the resulting predictions do not exhibit a high confidence level.

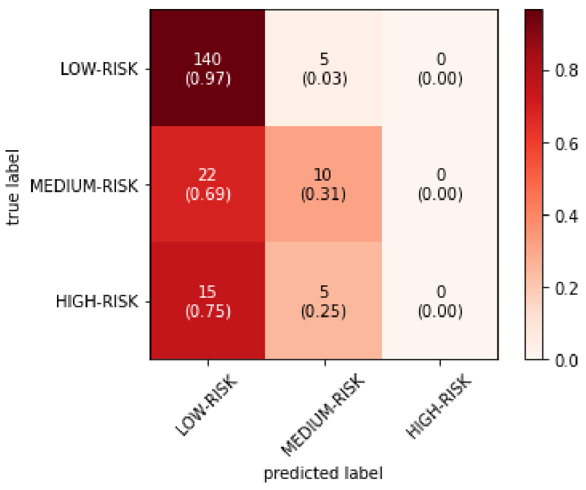


Figure 4.5: MTF Confusion Matrix | 3-class classification.

Encoding Techniques	Accuracy	Precision	Recall	F1-Score
GAF+Diff(7)	0.85±0.02	0.84±0.01	0.83±0.02	0.84±0.02
MTF+Exp(7)	0.81±0.01	0.80±0.02	0.79±0.01	0.80±0.02
RP+Exp(15)	**0.96±0.02**	**0.95±0.01**	**0.95±0.01**	**0.95±0.01**
WV+Exp(30)	0.83±0.01	0.82±0.02	0.81±0.01	0.81±0.01

Table 4.16: Performances | 2-class classification

Two Classes Classification Results

In this section, the results for predicting bearing health status in two classes using a CNN are analyzed by merging the MEDIUM-RISK and HIGH-RISK classes into a single class, creating a 2-Band classification model. Table 4.16 demonstrates that the **Recurrence Plot** technique consistently outperforms others in all metrics, even in this specific task.

Furthermore, the results presented in Table 4.16 are corroborated by the examination of confusion matrices generated from the model's predictions on the test set (see Table 4.17).

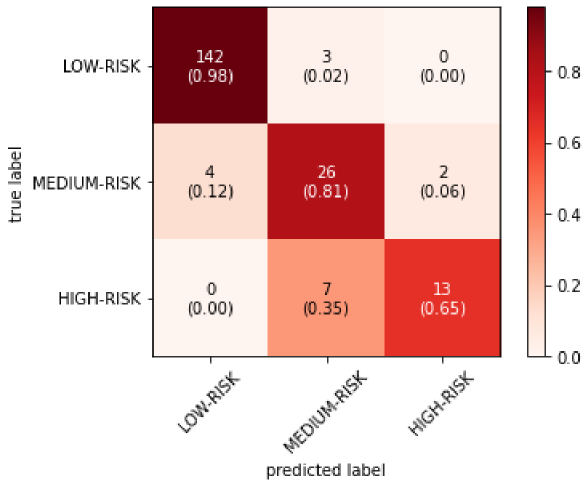


Figure 4.6: Recurrence Plot Confusion Matrix | 3-class classification.

Technique	TP	FP	FN	TN
GAF+Diff(7)	131	14	19	33
MTF+Exp(7)	138	7	17	35
RP+Exp(15)	140	5	6	46
WV+Exp(30)	139	8	16	34

Table 4.17: Confusion matrices (median values over repeated tests) | 2-class classification

It is evident that the developed network achieves outstanding performance when dealing with only two classes, enhancing model prediction reliability. Specifically, the confusion matrices indicate that both the numbers of *False Positives* and *False Negatives* are exceptionally low.

As evident from Figure 4.8, the loss curve maintains a typical shape, reaching a plateau at approximately 0.2 for the validation subset. This suggests that the network was trained effectively, and the predictions it generates exhibit a higher percentage of accuracy compared to the scenario with 3 classes.

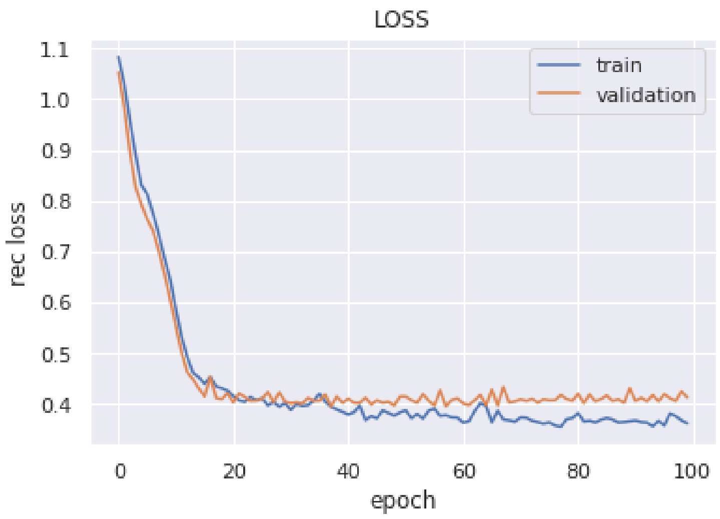


Figure 4.7: Model loss with RP | 3-class classification.

Comparison with different baselines

In this section, the designed model was compared to two reference models (see Table 4.18) for the binary classification task: the first model is an LSTM, a commonly used architecture for time series classification in predictive maintenance tasks. It consists of 2 layers, each containing 64 units, and a final layer with a softmax activation function. The second reference model is the one described in [151], which achieved the best performance on the classification task using the NASA bearing dataset.

Model	**Accuracy**	**F1-score**
[151]	**0.98±0.01**	**0.97±0.01**
LSTM	0.90±0.02	0.91±0.02
Proposed CNN	0.96±0.02	0.95±0.01

Table 4.18: Performances of the three compared models.

It is important to highlight that the proposed network does not surpass the performance of the model described in [151]. However, it achieved better

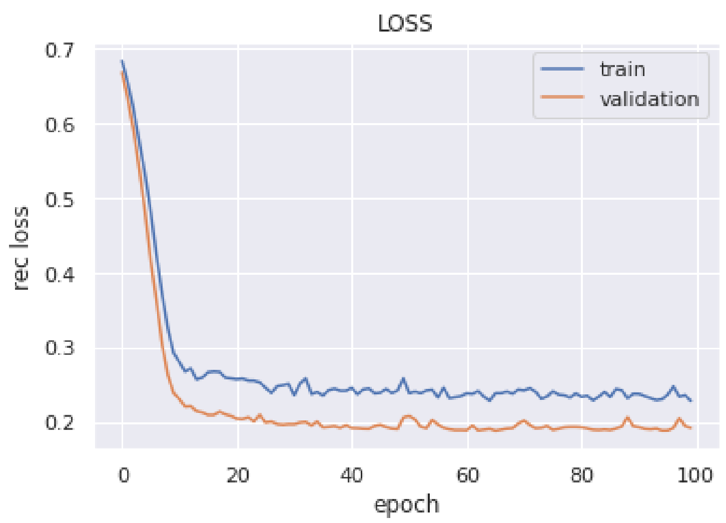


Figure 4.8: Model loss with RP | 2-class classification

results than the LSTM network in terms of both accuracy and F1-score. Table 4.19 presents the efficiency performance of the designed model compared to the two models examined in Table 4.18, considering factors such as model memory size and mean training time.

Model	**Memory usage**	**Training time (seconds)**
[151]	5 MB	60 s/epoch (b. at 60-th ep)
LSTM	850 kB	**10 s/epoch** (b. at 20-th ep)
Proposed CNN	**380 kB**	20 s/epoch (b. at 50-th ep)

Table 4.19: Memory usage and training time

It is important to highlight that the proposed network does not surpass the performance of the model described in [151]. However, it achieved better results than the LSTM network in terms of both accuracy and F1-score. Table 4.19 presents the efficiency performance of the designed model compared to the two models examined in Table 4.18, considering factors such as model memory size and mean training time.

4.5.3 Benefits of GAN

Despite various data augmentation strategies proposed, some, such as rotating and flipping, when applied to encoded images, can distort the time domain signal, rendering them unreasonable for use (see [152] for examples). Therefore, the performance of the discussed methodology is analyzed when adopting Generative Adversarial Networks (GANs). Specifically, Table 4.20 demonstrates that while GANs provide a slight performance improvement during training, this advantage must be weighed against increased demands on training time and memory resources.

To address the potential drawback of having failure labels still within the minority class, a GAN was developed to augment the number of samples in the minority class (see Figure 4.9). This GAN utilizes a CNN **discriminator** to differentiate between real and generated images created by another CNN **generator**, which samples from a Gaussian distribution.

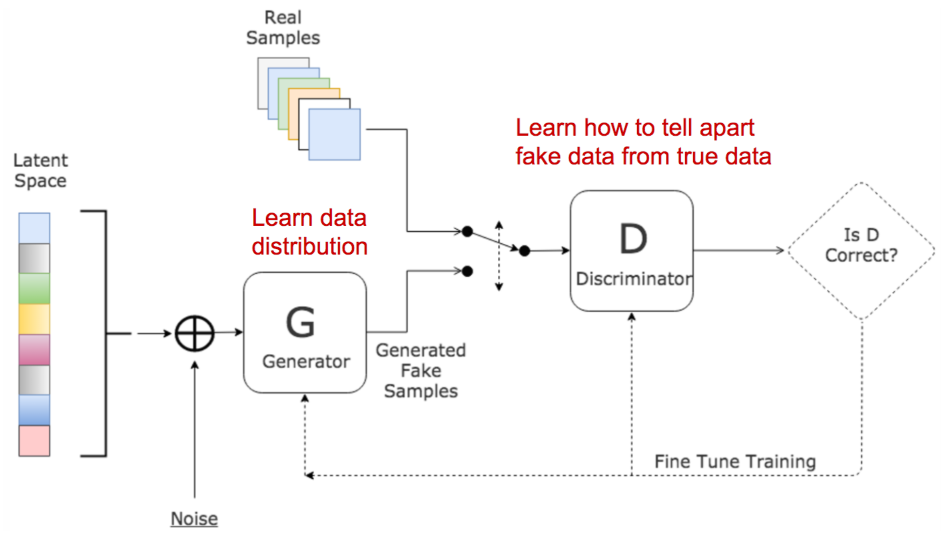


Figure 4.9: GAN architecture - It consists of two sub-models, Generator and Discriminator. The former is responsible for generating new plausible examples from the problem domain. The second one is used to classify examples as real (from the domain) or false (generated).

	Technique	F1-Score	Precision	Recall
Alibaba	Without GAN (GASF + Exp(15))	31.59±1.25	29.58±0.22	34.03±3.13
	With GAN (+25% fake tensors)	**34.47±2.13**	**32.46±0.22**	37.02±4.96
	With GAN (+50% fake tensors)	32.52±1.24	27.43±0.66	**40.16±3.91**
Bearing	Without GAN (RP + Exp(15))	0.95±0.01	0.95±0.01	0.95±0.01
	With GAN (+25% fake tensors)	**0.97±0.02**	**0.97±0.03**	0.96±0.02
	With GAN (+50% fake tensors)	0.96±0.01	0.96±0.02	**0.97±0.01**

Table 4.20: Results of data augmentation with GAN on Alibaba HDD and NASA Bearing datasets.

The GAN model operates by jointly training the discriminator and the generator (as illustrated in Figure 4.10 and 4.11). The discriminator is trained on a batch comprising half fake and half real samples, while the generator updates its parameters based on the loss of the frozen discriminator. The discriminator's role is to predict the probability of assigning a given input image to class '0' (fake) or '1' (real). Meanwhile, the generator aims to maximize the probability of the discriminator labeling artificially generated images as "truthful."

If the discriminator consistently predicts a low probability of truthfulness for the artificially generated images, it results in a substantial back-propagated error signal in the generator. Consequently, this error drives a relatively large feedback to the generator, improving its ability to generate more convincing "fake" samples in the subsequent batch.

The assumption about the GAN's behavior was further substantiated by examining the generator's loss. To illustrate this, we present the loss data for the HDD dataset in Figure 4.12. It's evident from the figure that the generator's loss stabilizes between batch numbers 2000 and 3000, indicating that the generator is functioning effectively.

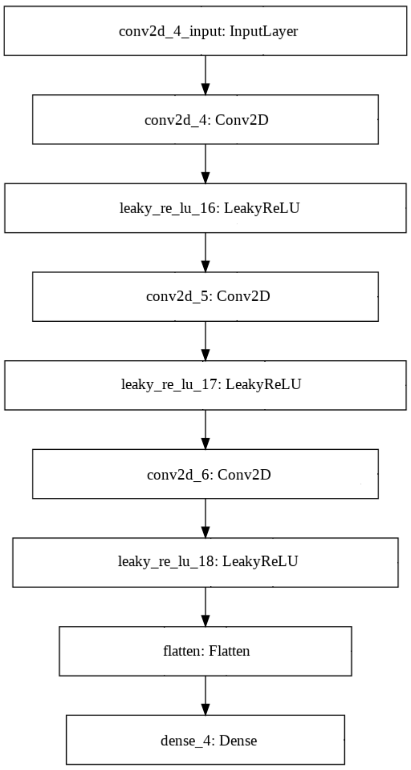

conv2d_4_input: InputLayer
conv2d_4: Conv2D
leaky_re_lu_16: LeakyReLU
conv2d_5: Conv2D
leaky_re_lu_17: LeakyReLU
conv2d_6: Conv2D
leaky_re_lu_18: LeakyReLU
flatten: Flatten
dense_4: Dense

Figure 4.10: Discriminative network

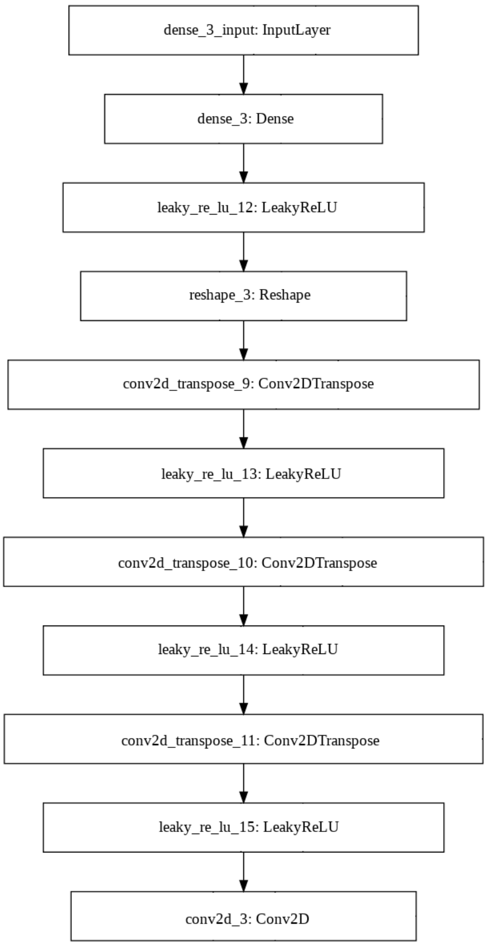

dense_3_input: InputLayer
dense_3: Dense
leaky_re_lu_12: LeakyReLU
reshape_3: Reshape
conv2d_transpose_9: Conv2DTranspose
leaky_re_lu_13: LeakyReLU
conv2d_transpose_10: Conv2DTranspose
leaky_re_lu_14: LeakyReLU
conv2d_transpose_11: Conv2DTranspose
leaky_re_lu_15: LeakyReLU
conv2d_3: Conv2D

Figure 4.11: Generative network

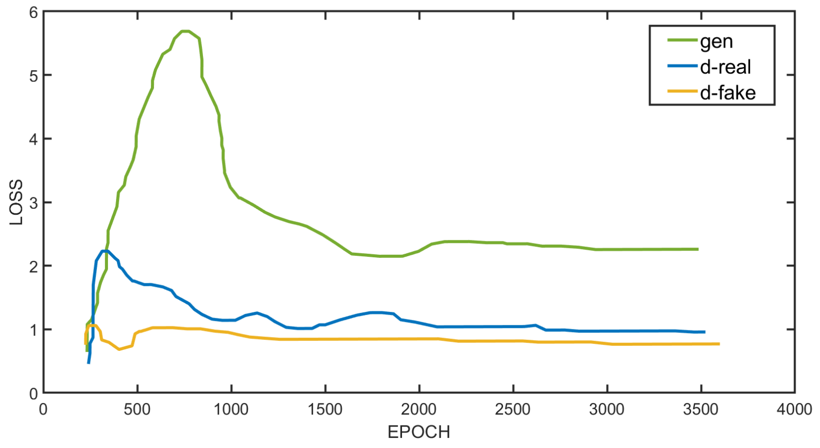


Figure 4.12: Loss plot for real and fake samples, and the generator

4.5.4 Combination of Encoding strategies

Recent advancements in the literature [153, 154] have led to the application of an ensemble of encoding techniques. This approach combines various strategies described in Section II to enhance the model's performance and reliability. For each sample, a single three-channel input volume ($40 \times 40 \times 3$) is generated by employing three different encoding strategies (GAF, MTF, RP).

The network parameters and training procedures remain consistent with the previous experiments, with the only modification being the network input. Table 4.21 presents the experimental results comparing the performance of the 1-channel and 3-channel networks. It indicates a slight increase in the confidence interval of the combination strategy, with the loss being statistically smaller in the second network. However, there is no statistically significant difference in accuracy and F1 performance.

Encoding Type	Accuracy	F1-score	Loss
1-Channel	0.96±0.02	0.95±0.01	0.06±0.01
3-Channel	0.96±0.01	0.94±0.01	**0.01±0.01**

Table 4.21: Performance of 3-channel encoding

4.6 Discussion & Conclusions

In the context of industrial systems, the need for preventive maintenance and effective monitoring techniques has grown significantly due to the increasing complexity of these systems. Predictive maintenance has emerged as a crucial tool for businesses and industrial settings, offering benefits such as cost reduction, data loss prevention, and minimized downtime, especially in data centers and mechanical component maintenance.

With the rapid digitization of industrial processes, vast amounts of real-time data are now available for analysis. This paper aims to develop and evaluate techniques that can harness this wealth of information effectively. The authors propose an evaluation framework that combines various time series encoding methods with Convolutional Neural Network (CNN) image classifiers for predictive maintenance tasks. CNNs have demonstrated their ability to handle challenges often encountered in predictive maintenance data, such as missing values and sparsity.

The study explores four encoding methods and assesses two different CNN models using the PAKDD2020 Alibaba AI Ops Competition dataset (focused on HDD health status in data centers) and the NASA Bearing dataset (capturing vibration signals from bearings). Additionally, the authors investigate the potential benefits of using a Generative Adversarial Network (GAN) to enhance model performance. While the GAN does provide a slight improvement in prediction performance, it comes at the cost of significantly increased training time and computational resources, leading to a discussion of the trade-offs involved.

In a subsequent evaluation, the authors compare their best-performing CNN model and encoding technique with three benchmarking neural network models on the Alibaba and NASA Bearing datasets. The benchmarks include LSTM and XGBoost models, with a particular focus on the competition-winning XGBoost model for the Alibaba dataset, as well as LSTM and a model from a previous study for the NASA Bearing dataset. The discussion delves into the trade-offs between computational resources and model performance across various evaluation metrics.

Overall, the results support the combined use of image encoding techniques with CNN models for predictive maintenance tasks, especially when compared to other models. However, the resource-intensive nature of CNN models, such as longer training times, should be carefully considered. The paper highlights the importance of conducting thorough cross-model evaluations for different tasks in the predictive maintenance domain.

In summary, the proposed approach achieves comparable or superior results to the state-of-the-art for both datasets, with a particular advantage in terms of efficiency. Future research directions may involve exploring tiled CNNs for improved computational efficiency, refining GAN-based approaches, investigating ensemble models for enhanced classification performance, and incorporating Explainable AI (XAI) techniques to aid practitioners in understanding misclassifications.

5 A predictive maintenance application in IoT scenarios

5.1 Introduction

The current era of Industry 4.0 is characterized by the integration of modern technologies like the Internet of Things (IoT) and Artificial Intelligence (AI) into manufacturing and industrial practices [155]. This integration involves the collection of vast amounts of data from smart equipment and sensors in production environments.

Smart sensors play a pivotal role by generating data on physical parameters and offering functionalities like self-monitoring and self-configuration. This data is analyzed for strategic decision-making, leading to benefits such as reduced maintenance costs, decreased machine faults, optimized spare parts inventory, and increased production efficiency.

Maintenance procedures in industrial settings are crucial, with industries investing in data-driven maintenance strategies [156, 157]. These strategies can be categorized as model-driven, data-driven, or hybrid-driven. Data-driven methods, which leverage collected data to predict machinery failures, are particularly promising for predictive maintenance [158, 159].

Predictive Maintenance (*PdM*) is a key focus, where maintenance actions are scheduled based on sensor-reported health status. Machine learning and deep learning techniques have become effective tools for PdM, utilizing historical data to train models for Remaining Useful Life (*RUL*) estimation and failure prediction [101, 105, 106].

However, challenges exist, including the need for computational efficiency and real-world IoT scenarios. Deep learning models often demand significant computational resources, limiting their deployment on equipment hardware. Embedded AI techniques are emerging to address these challenges [160].

The chapter introduces a deep learning approach for PdM that employs a multi-head attention mechanism. This approach offers high accuracy in *RUL* estimation while maintaining low memory storage requirements, making it suitable for direct implementation on equipment hardware. Experimental results on the NASA dataset demonstrate its superior effectiveness and efficiency compared to state-of-the-art techniques, making it a viable solution for real-world PdM scenarios [161].

The Chapter is organized as in the following. Section 5.2 reports the Related Work about PdM approaches and the related challenges, while Section 5.3 describes the proposed methodology together with the introduced deep architecture for PdM task. Sections 5.4 and 5.5 present the experimental protocol and the achieved results, respectively. Finally, Section 5.6 reports a final Discussion together with some Conclusions and Future Work.

5.2 Related Work

Predictive maintenance has emerged as a prominent research area, with a focus on leveraging Machine Learning (ML) and Deep Learning (DL) techniques, especially in the context of IoT and AI advancements. Recent studies, exemplified by [162] and [163], have explored data-driven methodologies that rely on time series analysis and mining. Additionally, [164] introduced a framework for Predictive Maintenance (PdM) that combines historical and real-time data to continually enhance performance.

Table 5.1 provides a summary of some notable proposals in this field. It's evident that a majority of these approaches are built upon Long-Short Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs). Within the realm of DL approaches, those utilizing the NASA Turbofan Engine dataset have garnered significant interest, particularly falling into two categories: Recurrent models and Hybrid models, as outlined in Table 5.1. The next sections will outline the criteria for organizing the literature review, delve into the discussed models, and elucidate how our proposed approach addresses pertinent limitations.

5.2.1 Selection Criteria

The literature review adhered to the guidelines outlined by [165] and [166], ensuring the systematic and high-quality conduct of research. This process involved several key steps: formulating the research question, identifying pertinent literature, synthesizing the findings, and presenting the results.

The primary objective was to retrieve relevant publications in the realm of Predictive Maintenance (PdM) employing Deep Learning (DL) approaches across various industrial contexts. Special attention was given to studies utilizing our dataset for experimentation, as outlined in Table 5.1. Notably, the focus was on works employing Recurrent and Hybrid network models. Ultimately, this literature review aimed to address the key research challenges in the PdM domain.

The search spanned the years 2017 to 2022 and encompassed articles from both Scopus and Google Scholar. Several keywords, individually and in combination, were employed, including "predictive maintenance," "deep learning," "recurrent model," "hybrid," "machine learning," "attention mechanism," and "multi-head attention."

Consequently, this process yielded a collection of 21 articles, comprising 17 from Scopus and 4 from Google Scholar. An initial screening was conducted based on abstracts and methodologies, followed by a comprehensive review of full chapters for articles that utilized the NASA Turbofan Engine Dataset.

Ref.	**Year**	**Deep Learning (DL) approach**
[167]	2020	Anomaly triggered Long Short-Term Memory (LSTM)
[168]	2019	Restricted Boltzmann Machine (RBM) + Long Short-Term Memory (LSTM)
[169]	2020	Long Short-Term Memory (LSTM) with attention
[170]	2019	Long Short-Term Memory (LSTM) + Convolutional Neural Network (CNN)
[171]	2020	Multi-Head Attention (MHA) + LSTM + CNN + Neural Turing Machine (NTM)
[172]	2020	Noisy Bidirectional Long Short-Term Memory (BLSTM) + CNN

Table 5.1: A summary of the most recent papers regarding NASA Turbofan Engine Dataset.

5.2.2 Recurrent models

[167] introduced a two-component framework for predictive maintenance. The initial component is responsible for identifying significant deviations from the normal (healthy) state. Following anomaly detection in streaming sensor data, the second component, an LSTM model, is activated for Remaining Useful Life (RUL) estimation. This approach continuously monitors for anomalies and initiates RUL estimation only upon anomaly detection.

A similar LSTM-based approach for RUL estimation was devised by [187]. This method uncovers concealed patterns through the analysis of sensor sequence data. Likewise, [188] employed an LSTM model with a data-driven approach to predict failures based on real operating conditions and dynamic loading.

[169] employed an Encoder-Decoder architecture based on LSTM, incorporating an attention mechanism to address lengthy sequences. Attention mechanisms, by prioritizing specific aspects of the input while minimizing

attention to others, enhance the training process and reveal relationships between input and output [189] (further details in Section 5.3). Similarly, [190] introduced another LSTM-based attention mechanism to enhance the model's capacity to analyze sequences of signals in survival analysis. Additionally, [191] proposed a predictive maintenance system based on an FPGA-adapted LSTM network, aiming to concurrently reduce power consumption and management costs.

A common challenge in supervised predictive maintenance applications is the scarcity of labeled data. [168] tackled this problem by exploring the impact of unsupervised pre-training in RUL predictions using a semi-supervised setup. The approach involved using a Restricted Boltzmann Machine (RBM) in the initial layer for unsupervised pre-training. The RBM learned abstract features from unlabeled raw input data and initialized weights, facilitating supervised fine-tuning with LSTM to capture long-term dependencies. Finally, a fully connected output layer was added for RUL prediction.

5.2.3 Hybrid models

[170] introduced a *Hybrid Deep Neural Network Model* (HDNN) consisting of two parallel paths, LSTM and CNN, followed by a fully connected layer that combines their outputs to predict the target Remaining Useful Life (RUL). This framework employs the LSTM path to extract temporal features, while concurrently using CNN to extract spatial features. An extension of this approach was developed by [172], introducing a dual-path deep learning architecture trained with noisy input data (Noisy Bidirectional LSTM - NBLSTM). They demonstrated that training on noisy data, particularly with Gaussian Noise, could enhance the model's robustness, leading to significantly improved generalization behavior.

Recently, [171] proposed a dual-stream architecture, comprising a *Multi-Head Attention* (MHA) module and a Neural Turing Machine module. In this approach, time-series data is divided into shorter windows and labeled using a piece-wise linear degradation model, with a fixed maximum RUL value of 125. These windows are then input into the MHA module, which identifies relationships between different sensor data to reveal hidden patterns. The MHA module's output is subsequently fed into the networks in each stream. The features extracted by the LSTMs in the first stream and the CNN in the second stream are concatenated with augmented features computed by the NTM module. Finally, two stacked feedforward networks map the extracted features to a sequence of RUL values.

[192] presented the *ConvNet* model, utilizing a CNN-LSTM network to estimate the RUL of a turbofan engine while reducing the number of parameters. On the other hand, [193] proposed a novel bi-directional gated recurrent unit with a temporal self-attention mechanism (BiGRU-TSAM) for RUL prediction.

Lastly, [194] introduced a hybrid multi-task deep learning approach that integrates the strengths of CNN and LSTM networks. CNN serves as a feature extractor, while LSTM captures long-term temporal dependency features.

5.2.4 Research Challenges in Predictive Maintenance

Despite the considerable progress made in developing predictive maintenance approaches in recent years, several limitations have been identified:

1. Recurrent Neural Networks (RNNs), due to their inherently sequential nature, lack parallelization within training examples.

2. The length of the time window used as input during training can pose challenges, especially concerning vanishing and exploding gradient problems. RNNs, including Long Short-Term Memory (LSTM) networks, may struggle to retain information from the early timesteps when processing lengthy sequences, leading to issues with very long-term dependencies.

3. Many existing frameworks are complex and have a substantial number of parameters, requiring significant storage space. This complexity is often not considered, even though it is crucial for deploying predictive maintenance models in resource-constrained hardware environments, where memory size and power consumption are critical factors.

Therefore, there is a need to design efficient analytics for critical predictive maintenance applications that can meet specific requirements, such as reliability, low latency, privacy, and power efficiency, as highlighted in [195].

Inspired by the success of Transformer models in Natural Language Processing (NLP), researchers have explored their applicability in other domains. For instance, [196] introduced the SAnD (Simply Attend and Diagnose) architecture for clinical time-series data, incorporating a masked, self-attention mechanism, positional encoding, and dense interpolation strategies to handle temporal order. Additionally, [176] used Transformers for time-series forecasting, a critical task in various scientific and engineering disciplines that involves predicting future trends based on historical data.

The main innovation of this work is the introduction of an efficient deep network based on multi-head attention for predictive maintenance tasks. This model maintains high accuracy performance while significantly reducing processing time and storage requirements. Notably, while previous literature often combined the attention mechanism with other types of networks (e.g., LSTMs or CNNs), our proposed model relies solely on the attention mechanism, making it an efficient solution for embedded AI applications.

5.3 Methodology

As we have seen from previous sections, tasks related to predictive maintenance can be modeled as regression and classification problems. In regression, the primary objective is to estimate the Remaining Useful Life (RUL) of machinery, while in classification, the goal is to predict the health state of the equipment. Data-driven techniques play a crucial role in optimizing maintenance procedures and preventing downtime.

The aim of this work is to design an AI model capable of estimating RUL from various types of equipment data. The general workflow for RUL estimation involves:

Selecting the most suitable RUL estimation model based on the available data and system knowledge. Training the estimation model using historical data. Using test data similar to historical data to estimate the RUL of the test component. In this section, the methodology is detailed, beginning with the task definition and then presenting the model architecture, with a particular focus on the introduced "Attention" module.

5.3.1 Model architecture

The high-level view of the proposed model architecture for the PdM task is depicted in Figure 5.1. The architecture comprises several key components, including positional encoding and an attention module.

Specifically, the model architecture consists of:

- **Positional Encoding:** This component addresses the sequential nature of the data by incorporating information about the relative or absolute position of time-steps in the input sequence. It helps capture temporal dependencies.

- **Attention Module:** The attention module is composed of two sub-layers with residual connections:

    1. **Multi-Head Attention Block:** This block enables the model to attend to different parts of the input sequence simultaneously, capturing complex relationships between features.
    2. **Fully Connected Network Module:** This module processes the multi-head attention outputs to produce the final predictions.

The positional encoding ensures that the model can account for the order and position of time-steps in the input sequence, which is crucial for detecting degradation trends. The attention module helps the model capture temporal dependencies effectively.

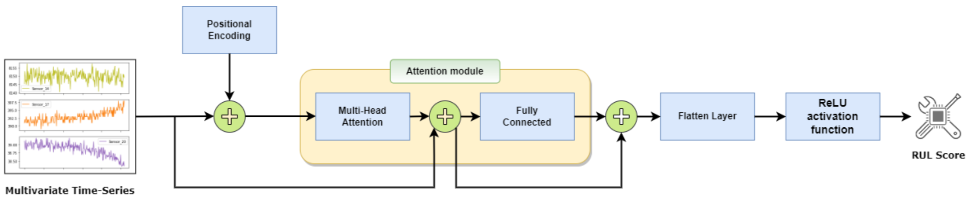


Figure 5.1: Proposed AI architecture.

5.3.2 Positional encoding

Since the proposed model lacks recurrences, it requires a mechanism to incorporate the order of time-steps. Positional encoding is introduced to achieve this. Each time-step in the input sequence is augmented with information about its position. This positional encoding is added to each input time-step to provide the model with a sense of order. The positional encoding is computed using sine and cosine functions with varying frequencies, resulting in a vector of length N_x. These functions introduce information about the position of the time-steps in the sequence, ensuring that the model can capture the order of events effectively. By combining positional encoding with the attention mechanism, the model can process sequential data efficiently. The proposed model architecture aims to address the challenges of predictive maintenance effectively while maintaining efficiency and accuracy. It leverages positional encoding and attention mechanisms to capture temporal dependencies and provide reliable RUL estimations.

5.4 Experimental Evaluation

The major innovation of this work lies in the introduction of an attention-based deep architecture for predictive maintenance, specifically designed for applications that require predictive models to be stored in memory-constrained devices. To demonstrate its effectiveness compared to common recurrent deep models, a comparison was conducted with a widely used architecture in terms of model storage size and accuracy performance, using specific metrics.

The chosen architecture for comparison is an LSTM network with two layers, each containing 128 units, and a final Dense layer with a ReLU activation function for the regression task.

The comparison was conducted using the Turbofan Engine Degradation Simulation Dataset, provided by NASA Ames Prognostics Data Repository [7]. This dataset is a well-known benchmark in the Prognostic and Health Management (PHM) field. It is generated by the C-MAPSS tool, simulating various degradation scenarios of engines of the same type. The dataset comprises four sub-datasets (FD001, ..., FD004) with different operating conditions and fault modes, each including training and testing datasets.

The training dataset consists of run-to-failure sequential data collected from 21 sensors [197]. The engines initially operate normally with varying degrees of initial wear, and the sensors record data until a system failure occurs.

Each row of the dataset includes 26 fields:

1. Engine ID.
2. Cycle index.
3. Three fields representing the engine's operating condition.
4. Twenty-one sensor readings.

[7]https://data.nasa.gov/Aerospace/Turbofan-engine-degradation-simulation-data-set/vrks-gjie

The test set data differs in that the engines start in an unknown deteriorated state, and the sensor readings terminate before a system failure occurs. Therefore, the objective is to predict the Remaining Useful Life (RUL) of each engine. For evaluation purposes, the true RUL values for the test trajectories are provided.

In the test dataset, sensory data of the system leading up to the system failure are recorded. The task is to estimate the RUL of the engine in the testing dataset, and the actual RUL of each data sample in the testing dataset is provided for result validation of the proposed method.

5.4.1 Hyperparameters

To provide more insight into the model's hyperparameters, a summary is presented:

1. *NUM_ENC*: This parameter indicates the number of stacked attention modules.

2. *NUM_HEADS*: It signifies the number of attention heads in the multi-head attention mechanism.

3. *KEY_DIM*: This parameter denotes the dimension of the query and key vectors. In this work, the value vector also has this dimension, although it is not mandatory.

4. *FFN_FACTOR*: It regulates the number of neurons in the first layer of the Fully Connected sublayer within each of the attention modules.

However, there are other hyperparameters, such as learning rate, batch size, and the number of epochs, which are not specific to the proposed model but do impact the training stage of the model.

The implementation of the proposed model was carried out in *Tensorflow 2.0* using the corresponding *Keras* layers.

Below, the analysis steps are summarized, followed by more detailed explanations of each:

1. Feature selection and normalization.
2. Definition of the RUL target function.
3. Creation of time windows.

5.4.2 Feature Selection and normalization

Feature selection activities were conducted, including the removal of constant columns. Specifically, the following columns were deleted: $sensor_1$, $sensor_5$, $sensor_{16}$, $sensor_{18}$, and $sensor_{19}$. Additionally, columns related to the operational condition were removed since it was consistent across all engines.

Figure 5.2 illustrates that $sensor_6$ and $sensor_{10}$ do not significantly contribute to detecting a degradation trend. Therefore, these features were also dropped. This holds true for the other engines as well.

The engine IDs and cycle numbers are not utilized during model training but are crucial for creating the time windows.

Following the feature selection process, 14 columns are retained.

Considering the varying ranges of sensor measurements, a normalization step is implemented to standardize the values and ensure unbiased contributions from each sensor reading. Specifically, min-max scaling is applied to each sensor, scaling their values to the [0, 1] range.

5.4.3 RUL target function definition

Data obtained from the NASA UCR Repository cannot be directly employed for training a model using supervised learning techniques since it lacks ground-truth information. To address this limitation, various approaches have been introduced in the literature.

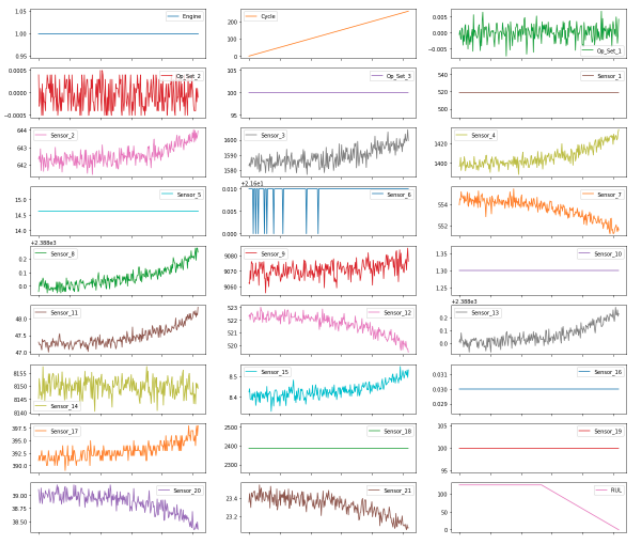


Figure 5.2: Features plot of the first engine.

In this work, the piece-wise linear degradation model is utilized. The fundamental concept behind this strategy is straightforward. As engine failure typically occurs gradually, it is not suitable to employ the real remaining useful life (RUL) as the label. Instead, a degradation threshold is established, and the period before the engine reaches this threshold is disregarded. Once the operating time surpasses the degradation threshold, the engine's remaining usable life monotonically decreases. To model this process, a piece-wise linear degradation model is adopted, based on the approach proposed by [198].

In accordance with the majority of related studies, a clip value of 125 is set.

Figure 5.3 illustrates the distinction between the two primary RUL target functions commonly employed.

To formalize this concept, let's introduce some variables. Consider n_i as the total number of cycles for the i-th engine, and let x_i represent the current cycle of that engine. The RUL value for this cycle can be determined as follows:

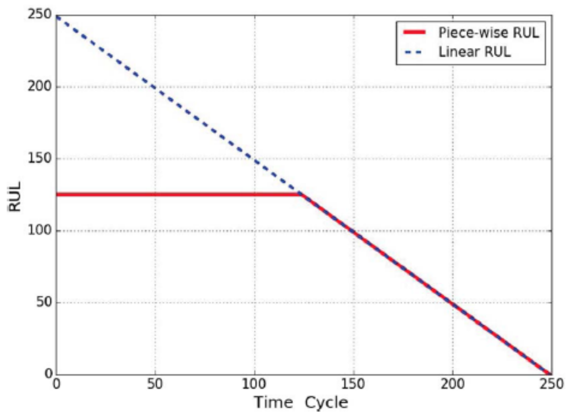


Figure 5.3: Linear and Piece-Wise Linear degradation model.

- If $n_i - x_i > 125$, then the RUL value is set to a fixed value of 125.
- Otherwise, the RUL value is calculated as $n_i - x_i$.

5.4.4 Time-windows creation

To summarize the data preparation phase, the proposed model requires input data in the form of sequential time windows. This is achieved using the sliding window method, as illustrated in Figure 5.4. With a given window size, W, the total number of cycles, T, and the window stride, s, it becomes possible to generate $T - W - s$ time windows for each engine. During training, the RUL value associated with a time window corresponds to the RUL of its last timestep (cycle). This study employs various time window sizes, including 10, 20, and 30, while keeping the stride s fixed at 1.

5.4.5 Performance metrics

In this section, the evaluation metrics employed for assessment are outlined. For the Turbofan Engine Degradation dataset, two objective metrics are

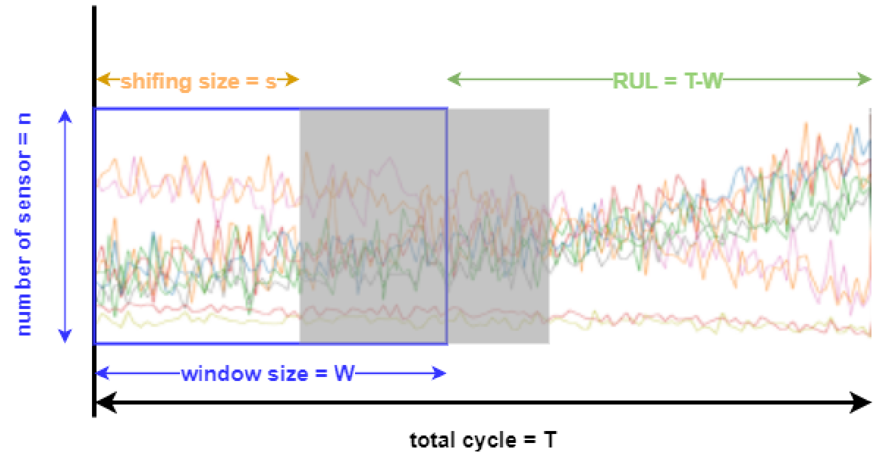


Figure 5.4: Time windows creation.

utilized to evaluate the model's performance: the Scoring Function and the Root Mean Square Error (RMSE), given that it is a regression problem.

Let RUL_i' and RUL_i represent the estimated and actual RUL, respectively, of the ith test engine (with a total of N engines). The RMSE is expressed as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(RUL_i' - RUL_i)} \tag{5.1}$$

The Scoring function, originally introduced in [197] and now commonly employed in Prognostic and Health Management (PHM) applications, is utilized. By defining $h_i = RUL_i' - RUL_i$, this function can be expressed as follows:

$$S = \sum_{i=1}^{N} s_i \tag{5.2}$$

where

$$s_i = \begin{cases} e^{\frac{-h_i}{13}} - 1 & if \quad h_i < 0 \\ e^{\frac{h_i}{10}} - 1 & if \quad h_i \geq 0 \end{cases} \tag{5.3}$$

The scoring function aims for lower values to indicate better performance. In Figure 5.5, a plot depicts the scoring function alongside the RMSE. Notably, the RMSE does not distinguish between early predictions (when the estimated RUL is lower than the actual) and late predictions. However, in PHM, having accurate RUL predictions that allow for timely maintenance before machine failure is crucial. This characteristic is evident in the plot: as the predicted RUL exceeds the actual, the scoring function increases exponentially. On the other hand, if the predicted RUL is lower than the actual, it is considered an error but with a lower rate of increase in the scoring function.

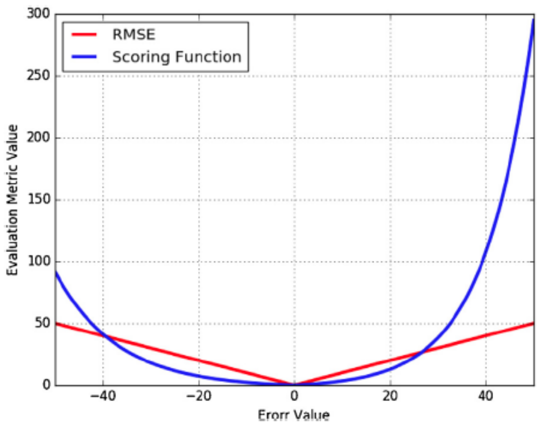


Figure 5.5: Scoring function.

5.5 Results

In this section, the results obtained from the experiments are presented and discussed. Specifically, Table 5.3 displays the performance metrics of the proposed model, with variations in the time window length. To ensure robust estimations, all tests were repeated 10 times, and the means and standard deviations were calculated. Similarly, Table 5.4 provides the performance metrics for the LSTM network under different time window lengths.

For the proposed model, the chosen hyperparameters are as follows: NUM_ENC 1, NUM_HEADS 8, KEY_DIM 28, and FFN_FACTOR 517. As for the LSTM model, it consists of 2 layers, each with 128 units. Both models were trained using the Adam optimizer, with a maximum of 300 epochs, a batch size of 128, and a learning rate of $10e-3$.

The first observation drawn from the results is that, as expected, both models benefit from an increase in the time window length.

Performance metrics achieve their highest scores with a time window length fixed at 30 cycles, which is reasonable as a larger number of samples can aid in extracting a degradation pattern in the engine.

Upon analyzing the reported results, there is no statistically significant difference between the two models in the score function when the time window length is equal to 20 cycles (391 and 375) and 30 cycles (279 and 262). However, this difference becomes statistically significant in favor of the proposed model when considering a time window length of 10 cycles.

Furthermore, the reported results should also be evaluated in terms of model complexity. Table 5.5 provides the number of parameters for both models. Notably, the proposed model achieves comparable performance to LSTM with approximately 86% fewer parameters.

This final aspect significantly impacts two critical aspects: model storage size and training time.

Table 5.6 reveals that the proposed model only requires 141 KB of memory, in stark contrast to the 2.5 MB needed by the LSTM network. This results in the proposed model being 94.36

To underscore the importance of limited memory occupancy in industrial contexts, it's worth referencing some published work in this domain. For instance, [199] devised a PdM strategy for embedded plant and machinery systems. Given the inherent limitations in embedded systems regarding computing resources, they had to strike a balance between memory occupancy, processing speed, and accuracy. Similarly, [200] introduced "TIP4.0," a modular framework for PdM in IoT, with a primary objective of offering a system capable of running on hardware with limited computational power and memory.

Table 5.7 includes the training times for the models with varying time window lengths. As demonstrated, the proposed model is approximately 20% faster to train compared to the LSTM model, even though their performance varies only slightly (an average of about 53 seconds for all tested window lengths). While this may seem like a modest improvement, it can be significant in real-world scenarios where data have high dimensionality. In such cases, even a small reduction in training time can have a substantial impact, particularly in terms of ensuring the system's scalability concerning dataset size, as highlighted in [201].

Table 5.8 provides a comparison with state-of-the-art approaches using the Turbofan Engine Degradation dataset as a benchmark. While our attention-based approach may not be the top-performing method, it remains competitive, particularly when considering the scoring function. The best results are highlighted in bold.

Figure 5.6 and Figure 5.7 display the prediction errors for the best runs of the proposed and LSTM models, respectively.

The horizontal axis represents the test engine's ID, ordered in decreasing order based on their actual RUL. For example, the engine with ID 82 in the test dataset has the lowest RUL, and so on. The vertical axis shows $RUL_i' - RUL_i$, which is denoted as "diff" in the plots, indicating the predicted RUL (RUL_i') for the i-th engine compared to its actual RUL (RUL_i).

The prediction error is consistently less than 10 when the actual RUL values are low. However, as the actual RUL increases, the prediction error tends to rise in both cases. This behavior may be attributed to the fact that when the actual RUL is low, the degradation process has already started, making it easier for the models to recognize the pattern. Conversely, when the actual RUL is very high, the engine is assumed to be in good condition, and the models struggle to differentiate between engines with true RULs of 80 and 105.

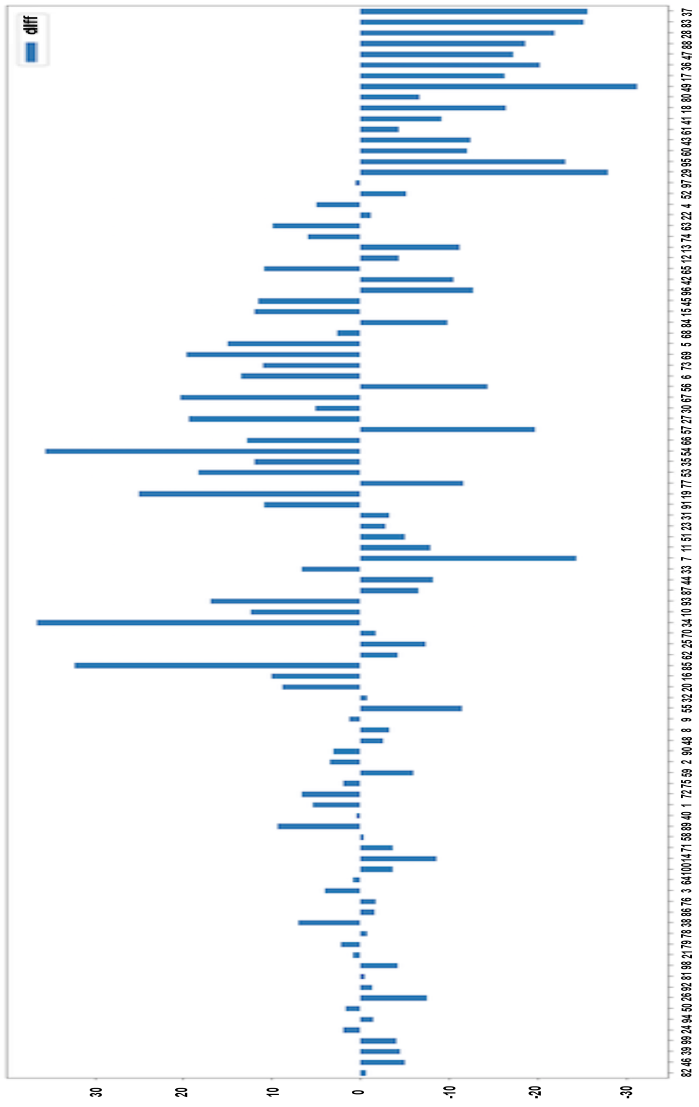


Figure 5.6: Best proposed model's prediction errors.

In summary, the analysis of the results provides several valuable insights:

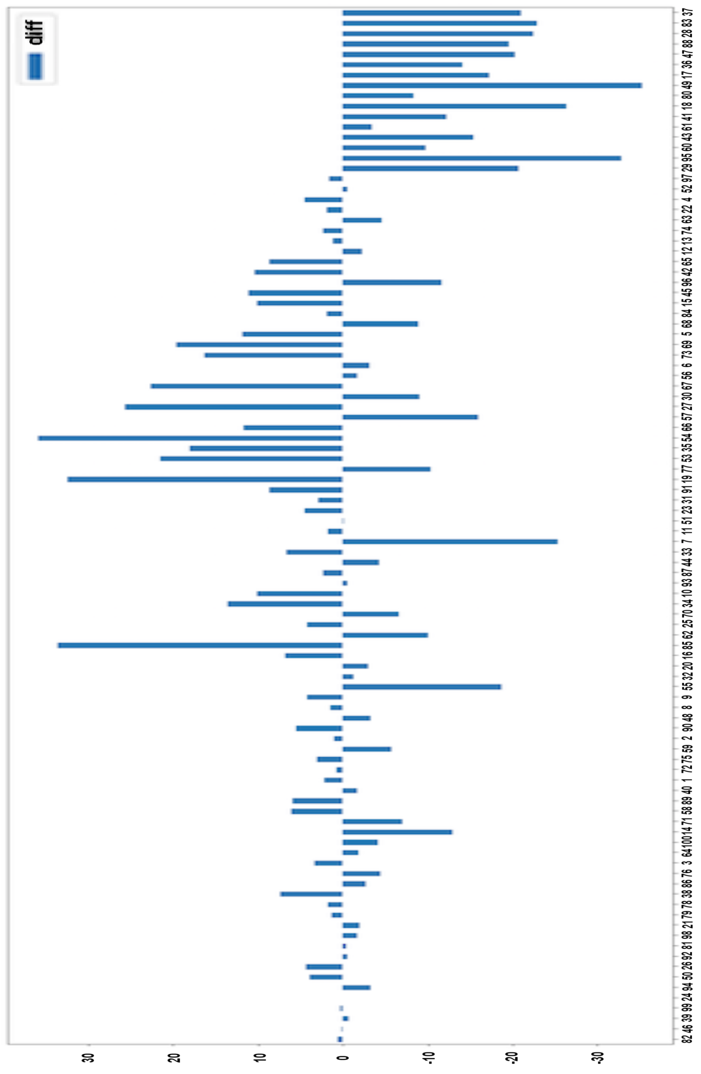


Figure 5.7: Best LSTM model's prediction errors.

- The proposed approach achieves a high level of accuracy when compared to the best-performing deep learning models in the literature. This indicates its effectiveness in estimating RUL.

- The model exhibits shorter training times and requires significantly less storage capacity, making it an efficient solution suitable for implementation on hardware with resource constraints.

- Despite its validation in a limited scenario, the model demonstrates promising results. This suggests that deploying it in a real-world Internet of Things (IoT) environment could meet the requirements of reliability, low latency, privacy, and low power consumption, as previously discussed in the literature [195].

5.6 Discussion and Conclusions

In the Industry 4.0 era, predictive maintenance (PdM) holds significant managerial and practical implications, offering the potential to reduce maintenance costs, prevent equipment failures, and optimize industrial operations. Data-driven approaches, particularly those employing machine learning algorithms, have gained prominence in recent years, enabling the estimation of machinery's remaining useful life (RUL) by analyzing historical operational data.

A notable trend in this domain is the adoption of deep learning models, which have demonstrated state-of-the-art results in fields like Computer Vision and Natural Language Processing, making them applicable to PdM tasks. However, many existing models are overly complex and ill-suited for critical applications where resource-constrained hardware is prevalent, notably in terms of memory capacity and power consumption. In such cases, relying on cloud-based processing is less desirable due to the stringent requirements of reliability, low latency, data privacy, and energy efficiency.

This work introduces a lightweight attention-based model designed to address these challenges. The attention mechanism, known for its success in Natural Language Processing tasks, is creatively adapted in this study to analyze time-series data. To validate the proposed model, the well-established

Turbofan Engine Degradation Dataset provided by NASA is used. The study includes comparisons with the latest state-of-the-art methods on this dataset, as well as an assessment of spatial and temporal complexity in comparison to a standard LSTM model.

The experimental results reveal that the proposed model achieves comparable performance in terms of RMSE and a PHM scoring function when compared to the recurrent LSTM model. However, the key advantage lies in the significantly reduced number of model parameters, resulting in a much smaller storage footprint and faster training times. This trade-off between efficiency and effectiveness is particularly valuable in industrial contexts, where optimizing the relationship between performance and resource allocation is crucial.

In summary, the findings demonstrate that the proposed approach meets the requirements of modern embedded AI applications. This holds significant implications for smart manufacturing systems, where reliability, low latency, privacy, and energy efficiency are paramount. Future research avenues may explore the application of the attention mechanism in various predictive maintenance scenarios and investigate methods for providing model explanations using eXplainable Artificial Intelligence (XAI) tools.

Reference	**Application domain**	**DL approach**	**Output of the model**	**Dataset**
[173]	Automobile	AE + DNN	Estimation of TBF of an automobile	Private
[174]	Railway Power Equipment	LSTM with residual connections	Prediction of the next failure time	Private
[175]	Conveyor Motor	GAF + CNN	A class corresponding to: No Fault, Minor Fault and Critical Fault	Private
[176]	Motor Bearing	LSTM	Health Status	NASA Bearing Dataset
[177]	Wind Turbine	Texture Signal Images + Multichannel CNN	Health Status or a specific fault	Synthetic
[178]	Rotating Machine	SAE + LSTM	/	Private
[179]	Rolling Bearing	CNN + Bid. GRU + Attention mechanism	RUL	Private
[180]	Helicopter	Time-series imaging + CAE	Anomaly detection	Airbus
[181]	Milling Machine Cutter	GAF + CNN	Tool wear class	Milling Machine Dataset (PHM10)
[182]	Rolling Bearing	GS + CNN	Tool wear class	NASA Bearing Dataset
[183]	Hard Disk	CNN-LSTM	A class indicating whether or not an HDD is going to fail	HDD Dataset
[184]	Hard Disk	LSTM	A class indicating whether or not an HDD is going to fail	ZTE's disk dataset
[185]	Hard Disk	LSTM	A class representing the health state of the HDD	HDD Backblaze dataset
[186]	Industrial equipment	Autoencoder	Anomaly detection	MIMII Dataset
[59]	Hard Disk	GAF + CNN	A class representing the health state of the HDD	HDD Backblaze dataset

Table 5.2: A summary of the most recent Chapters regarding Deep Learning models for predictive maintenance.

TW [Cycles]	RMSE	Score
10	$18,92 \pm 0,26$	1290 ± 42
20	$14,40 \pm 0,21$	391 ± 17
30	$13,50 \pm 0,30$	279 ± 23

Table 5.3: Performance metrics of the proposed model varying the Time Window length.

TW [Cycles]	RMSE	Score
10	$19,73 \pm 0,46$	1521 ± 44
20	$14,76 \pm 0,28$	375 ± 37
30	$13,11 \pm 0,36$	262 ± 20

Table 5.4: Performance metrics of the LSTM network varying the Time Window length.

TW [Cycles]	Proposed approach	Standard LSTM
10	28233	204929
20	28373	204929
30	28513	204929

Table 5.5: Number of parameters comparison.

Proposed approach	Standard LSTM
141 KB	2,5 MB

Table 5.6: Models' storage size.

TW [Cycles]	Proposed approach	Standard LSTM
10	$217,68 \pm 4,04$ [s]	$272,54 \pm 3,81$ [s]
20	$235,08 \pm 1,69$ [s]	$290,21 \pm 2,83$ [s]
30	$273,34 \pm 2,99$ [s]	$323,67 \pm 16,50$ [s]

Table 5.7: Comparison of training times by varying the time window (TW) between proposed model and LSTM network.

Authors	DL approach	RMSE	Score
[172]	**Noisy BLSTM + CNN**	$11,36 \pm 0,09$	226 ± 3
[169]	LSTM with attention	11,44	263
[168]	RBM + LSTM	12,10	251
[170]	LSTM + CNN	$12,22 \pm 0,04$	288 ± 4
[202]	CNN + LSTM	12,46	535
Standard LSTM	LSTM	$13,11 \pm 0,36$	262 ± 20
Proposed approach	**Attention-based**	$13,50 \pm 0,30$	279 ± 23
[167]	Anomaly triggered LSTM	17,63	424
[171]	MHA + LSTM + CNN + NTM	/	275

Table 5.8: Comparison with state-of-the-art approaches.

Part III

Time Series forecasting in FinTech

In this Part III, two approaches applied in the context of Fintech, specifically focusing on stock market prediction, are presented.

The Chapter 6 introduces a stock forecasting framework that integrates textual user-generated content and financial data as inputs into various deep learning models. A case study centered around GameStop is conducted, examining various deep learning models and assessing how textual information gleaned from prominent OSNs (Twitter and Reddit) can impact the stock prediction module's performance. The analysis underscores the significance of analyzing user-generated content sourced from diverse OSNs in the context of stock forecasting, even in the presence of dynamic user opinions that may pose challenges to the stock prediction task.

Instead, Chapter 7 developed a system dedicated to Stock Price Prediction, with the primary objective being the evaluation of market-derived data encompassing elements such as supply, demand, and exchanges as effective predictors of market trends, using Machine Learning (ML) and Deep Learning (DL) models. Furthermore, this investigation seeks to determine whether augmenting this data with sentiment analysis drawn from news sources and online communities can provide additional insights into stock price prediction. Prior to their introduction, a theoretical background is presented to facilitate comprehension for the reader.

Background

This Section summarizes a combination of Machine Learning algorithms and Neural Networks as predictive models, which were used in Chapters 6 and 7.

Random Forest Random Forest, proposed by Breiman [203] is a widespread ensemble learning method that combines the predictions of multiple decision trees to make more accurate predictions. The main idea behind Random Forest is to train many decision trees on random subsets of the training data and features, and then combine their predictions by averaging or taking the majority vote.

The individual decision trees in a Random Forest are trained on random subsets of the training data, which helps to reduce overfitting and improve generalization. Each decision tree is trained to predict the target variable based on a random subset of the input features, which helps to increase diversity among the trees and reduce correlation. The final prediction of the Random Forest is typically obtained by averaging or taking the majority vote of the predictions of the individual trees. Random Forest can handle both classification and regression tasks, and is robust to noisy and missing data.

To train a single decision tree, we recursively partition the input space into rectangular regions, based on the values of the input features. Each partition is defined by a decision rule of the form:

$$x_j \leq t_k \tag{5.4}$$

where x_j is the value of the jth feature, and t_k is a threshold value.

To find the optimal partition for each node of the decision tree, we choose the feature and threshold that minimize a cost function C, such as the Gini index or information gain:

$$C = \sum_{i=1}^{n} p_i(1 - p_i) \tag{5.5}$$

where n is the number of classes, and p_i is the proportion of training examples in class i in the current node.

To build a Random Forest, we train T decision trees on random subsets of the training data and features. For each tree t, we sample m features from the total p features, where $m << p$. This helps to increase diversity among the trees and reduce correlation.

$$m << p, \quad T \text{ decision trees trained on random subsets of data and features} \tag{5.6}$$

To make a prediction for a new example, we pass it through each decision tree in the forest, and obtain a set of T predictions. The final prediction is obtained by averaging (for regression) or taking the majority vote (for classification) of the individual predictions.

$$\text{Final prediction} = \begin{cases} \frac{1}{T}\sum_{t=1}^{T} f_t(x), & \text{regression} \\ \text{mode}(\{f_t(x)\}), & \text{classification} \end{cases} \tag{5.7}$$

where $f_t(x)$ is the prediction of the tth decision tree on the input example x.

In practice, Random Forest has been shown to be effective on a wide range of problems, including image classification [204], text classification [205], and regression analysis [206]. It has become a popular choice for machine learning tasks due to its robustness, scalability, and ease of use.

Gradient Boosting Gradient Boosting is a popular ensemble learning technique used in supervised learning tasks, particularly in regression and classification problems. It involves constructing a series of weak prediction models and iteratively refining them to improve the overall predictive performance of the model. The technique works by iteratively adding new models to the ensemble, with each new model attempting to correct the errors of the previous models.

The key idea behind gradient boosting is to optimize a loss function using gradient descent, where the gradient is computed with respect to the output of the current ensemble of models. This approach allows gradient boosting to effectively handle complex datasets with high-dimensional input features, noisy or missing data, and nonlinear relationships between the input features and the output variable.

The origin of gradient boosting can be traced back to the work of Friedman [207], who introduced the concept of boosting decision trees using gradient descent. Since then, gradient boosting has become a widely-used technique in machine learning, with a variety of implementations and extensions developed over the years.

The objective function to be minimized in gradient boosting is typically defined as a sum of loss functions over all training examples:

$$\mathscr{L} = \sum_{i=1}^{n} L(y_i, F(x_i)) \tag{5.8}$$

where n is the number of training examples, y_i is the true label of the ith example, x_i is the input feature vector of the ith example, $F(x_i)$ is the current prediction of the ensemble model on the ith example, and L is a loss function that measures the difference between the predicted and true labels.

The gradient of the objective function with respect to the output of the current ensemble model $F(x)$ can be computed as:

$$\frac{\partial \mathscr{L}}{\partial F(x_i)} = \sum_{j=1}^{n} \frac{\partial L(y_j, F(x_j))}{\partial F(x_i)} \tag{5.9}$$

To update the ensemble model, a new weak learner $h(x)$ is fit to the negative gradient of the loss function:

$$r_{ij} = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}, \quad \forall i = 1, \ldots, n \tag{5.10}$$

$$h_j(x) = \arg\min_{h \in H} \sum_{i=1}^{n} (h(x_i) - r_{ij})^2 \tag{5.11}$$

where H is the space of possible weak learners, and r_{ij} is the residual error between the predicted value and the true label for the ith example using the current ensemble model.

The weak learner is then added to the ensemble model by multiplying its predictions by a small learning rate ν and adding them to the current predictions:

$$F(x) \leftarrow F(x) + \nu h(x) \tag{5.12}$$

Extremely Randomized Trees (ExtraTrees) ExtraTrees is a ML algorithm that belongs to the ensemble family of methods. It was first introduced by Geurts et al. [208].

ExtraTrees is an extension of the popular Random Forest algorithm, which builds a collection of decision trees and aggregates their predictions to make a final prediction. In contrast, ExtraTrees introduces additional randomness by randomly selecting a subset of candidate features at each node of the decision tree and randomly selecting the splitting thresholds. This extra randomness leads to a more diverse set of decision trees and can improve the performance of the algorithm.

The training process of ExtraTrees can be summarized as follows. Given a training set $\{(x_1, y_1), \dots, (x_n, y_n)\}$, where x_i is a feature vector and y_i is a label, the algorithm generates a collection of T decision trees, where each tree is trained on a bootstrap sample of the training set and a randomly selected subset of features. For each node of each tree, a random subset of features is selected, and a splitting threshold is chosen at random. The best split among the randomly chosen ones is selected to split the node, and the process is repeated recursively until all the nodes are pure or until a stopping criterion is met.

The final prediction of the ExtraTrees algorithm is obtained by aggregating the predictions of the individual trees. For regression tasks, the predictions are simply averaged over the trees, while for classification tasks, the predictions are combined using voting or weighted voting.

The performance of ExtraTrees depends on the number of trees T, the number of randomly selected features, and the number of randomly chosen thresholds. Generally, increasing the number of trees and the number of features can improve the performance of the algorithm, but at the cost of increased computational complexity.

In summary, ExtraTrees can improve the performance of decision tree ensembles by introducing additional randomness in the tree building process. It has been shown to be effective in a wide range of applications, including image classification [209], speech recognition [210], and anomaly detection [211].

Least Absolute Shrinkage and Selection Operator (Lasso) Lasso is a regularization technique used in linear regression to prevent overfitting and select important features. It was first introduced by Tibshirani [212].

The Lasso technique adds a penalty term to the least squares cost function that encourages the coefficients of the less important features to be set to zero. This results in a sparse solution that selects only the most important features and avoids overfitting. The penalty term is defined as the L1 norm of the coefficient vector.

The optimization problem can be written as follows:

$$\min_{\beta_0,\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{5.13}$$

where y_i is the target variable, x_{ij} is the jth feature of the ith observation, β_j is the corresponding coefficient, β_0 is the intercept, and λ is a hyperparameter that controls the strength of the penalty. The first term is the least squares cost function, and the second term is the L1 penalty term.

The Lasso algorithm solves the above optimization problem using coordinate descent, which updates the coefficients sequentially while holding the other coefficients fixed. This makes the algorithm computationally efficient and allows for high-dimensional data.

The performance of the Lasso algorithm depends on the value of the hyperparameter λ, which controls the trade-off between the fit of the model and the sparsity of the solution. A larger value of λ results in a sparser solution, while a smaller value of λ allows for more features to be included in the model.

In summary, Lasso is a powerful regularization technique that can prevent overfitting and select important features in linear regression. It has been shown to be effective in a wide range of applications, including genetics [213], finance [214], and image processing [215].

Long Short-Term Memory (LSTM) LSTM is a type of recurrent neural network (RNN) that is capable of learning long-term dependencies and has become popular in various fields such as natural language processing, speech recognition, and image captioning. The LSTM architecture was introduced by Hochreiter & Schmidhuber in [216].

The key feature of LSTM is its ability to selectively remember or forget information from previous time steps. This is achieved through the use of memory cells and gates. The memory cell is a vector that stores information over time, and the gates are used to control the flow of information into and out of the cell.

The LSTM architecture consists of four main components: the input gate, the forget gate, the output gate, and the cell state. The input gate controls the flow of information from the input at the current time step into the cell state. The forget gate controls the flow of information from the previous cell state into the current cell state, allowing the LSTM to selectively forget or remember information from the past. The output gate controls the flow of information from the cell state to the output at the current time step.

The equations for the LSTM architecture are as follows:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{5.14}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{5.15}$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \tag{5.16}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{5.17}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{5.18}$$

$$h_t = o_t \odot \tanh(C_t) \tag{5.19}$$

where x_t is the input at time step t, h_{t-1} is the output of the LSTM at the previous time step, σ is the sigmoid activation function, $\odot$ denotes element-wise multiplication, W_f, W_i, W_C, W_o are weight matrices, and b_f, b_i, b_C, b_o are bias vectors.

The forget gate f_t and input gate i_t are computed using the sigmoid function and control the flow of information into and out of the cell state. The candidate values $\tilde{C}_t$ are computed using the hyperbolic tangent function and are used to update the cell state C_t. Finally, the output gate o_t controls the flow of information from the cell state to the output, and the output h_t is computed by taking the element-wise product of the output gate and the hyperbolic tangent of the updated cell state.

In summary, LSTM is a type of recurrent neural network that is capable of learning long-term dependencies by selectively remembering or forgetting information from previous time steps. The architecture uses memory cells and gates to control the flow of information and has become a popular choice for a wide range of applications [217].

Bidirectional LSTM Bidirectional Long Short-Term Memory (BiLSTM) is a type of neural network that combines the forward and backward information flow of two Long Short-Term Memory (LSTM) networks to model temporal dependencies in both forward and backward directions. BiLSTM was introduced by Schuster & Paliwal in [218].

The BiLSTM architecture consists of two LSTMs, one processing the input sequence in the forward direction and the other processing it in the backward direction. The outputs of both LSTMs are then concatenated at each time step to produce the final output. The use of both forward and backward information allows the BiLSTM to capture not only past dependencies but also future dependencies in the input sequence.

The equations for the BiLSTM architecture are as follows:

$$\overrightarrow{h_t} = \mathrm{LSTM}(x_t, \overrightarrow{h_{t-1}})\ \overleftarrow{h_t} \qquad = \mathrm{LSTM}(x_t, \overleftarrow{h_{t+1}})\ h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{5.20}$$

where x_t is the input at time step t, $\overrightarrow{h_{t-1}}$ is the hidden state of the forward LSTM at the previous time step, and $\overleftarrow{h_{t+1}}$ is the hidden state of the backward LSTM at the next time step. $[\cdot;\cdot]$ denotes concatenation of vectors.

The forward and backward LSTMs are identical to the standard LSTM, with the exception that the backward LSTM processes the input sequence in reverse order. The output of the BiLSTM at each time step is the concatenation of the forward and backward LSTM outputs.

BiLSTM has been shown to be effective in various applications such as speech recognition [219], natural language processing [220], and image captioning [221], where the modeling of both past and future dependencies is important.

Gated Recurrent Unit (GRU) Gated Recurrent Unit (GRU) is a type of recurrent neural network that is designed to address the vanishing gradient problem and allow for efficient modeling of long-term dependencies. GRU was introduced by Cho et al. in [222].

The GRU architecture is similar to Long Short-Term Memory (LSTM), but with fewer parameters and a simpler gating mechanism. The GRU unit has a reset gate and an update gate, which control how much information is passed from the previous time step and how much information is updated based on the current input. The update gate allows the GRU to selectively update the hidden state based on the input, while the reset gate allows the GRU to forget previous information that is no longer relevant.

The equations for the GRU unit are as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{5.21}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{5.22}$$

$$\tilde{h}t = \tanh(W_h x_t + U_h(r_t \odot ht - 1)) \tag{5.23}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{5.24}$$

where x_t is the input at time step t, h_{t-1} is the hidden state at the previous time step, σ is the sigmoid activation function, $\odot$ denotes element-wise multiplication, and $W_z, W_r, W_h, U_z, U_r, U_h$ are weight matrices to be learned.

The update gate z_t determines how much of the previous hidden state to keep and how much of the new candidate hidden state $\tilde{h}_t$ to incorporate. The reset gate r_t determines how much of the previous hidden state to forget and how much of the new input to use for the new candidate hidden state $\tilde{h}_t$.

GRU has been shown to be effective in various natural language processing tasks, such as language modeling [223], machine translation [224], and sentiment analysis [225], as well as in other applications such as image and speech recognition [226].

Bidirectional GRU Bidirectional Gated Recurrent Unit (BiGRU) is an extension of GRU that incorporates information from both past and future contexts in the input sequence. BiGRU was first introduced by Graves & Schmidhuber in [227]. BiGRU consists of two separate GRUs, one processing the input sequence forward and one processing the input sequence backward.

The output of each GRU is concatenated at each time step, providing a context window that includes information from both past and future. The forward and backward GRUs have their own set of weights, which are learned independently during training.

The equations for the forward GRU unit are as follows:

$$z_t = \sigma(W_z^f x_t + U_z^f h_{t-1}^f) \tag{5.25}$$
$$r_t = \sigma(W_r^f x_t + U_r^f h_{t-1}^f) \tag{5.26}$$
$$\tilde{h}_t^f = \tanh(W_h^f x_t + U_h^f (r_t \odot h_{t-1}^f)) \tag{5.27}$$
$$h_t^f = (1 - z_t) \odot h_{t-1}^f + z_t \odot \tilde{h}_t^f \tag{5.28}$$

The equations for the backward GRU unit are as follows:

$$z_t = \sigma(W_z^b x_t + U_z^b h_{t+1}^b) \tag{5.29}$$
$$r_t = \sigma(W_r^b x_t + U_r^b h_{t+1}^b) \tag{5.30}$$
$$\tilde{h}_t^b = \tanh(W_h^b x_t + U_h^b (r_t \odot h_{t+1}^b)) \tag{5.31}$$
$$h_t^b = (1 - z_t) \odot h_{t+1}^b + z_t \odot \tilde{h}_t^b \tag{5.32}$$

where x_t is the input at the time step t, h_{t-1}^f is the hidden state of the forward GRU at the previous time step, h_{t+1}^b is the hidden state of the backward GRU at the next time step, σ is the sigmoid activation function, $\odot$ denotes elemental multiplication, and $W_z^f, W_r^f, W_h^f, U_z^f, U_r^f, U_h^f$ are weight matrices for the forward GRU, and $W_z^b, W_r^b, W_h^b, U_z^b, U_r^b, U_h^b$ are weight matrices for the backward GRU.

The output of the Bi-GRU at each time step is the concatenation of the forward and backward hidden states:

$$h_t = [h_t^f, h_t^b] \tag{5.33}$$

BiGRU has been shown to be effective in various natural language processing tasks, such as part-of-speech tagging, named entity recognition, and sentiment analysis [220].

S-LSTM is a sequence of LSTM layers stacked on top of each other so that the output of each layer is the input of the next one. This allows the network to build a hierarchical representation of the input, where each layer is able to extract more abstract features from the data.

6 Forecasting the stock market leveraging social media data

6.1 Introduction

In recent years, there has been a growing interest in stock market analysis, driven by investors looking to profit from financial markets [228–230].

The stock market's fluctuations have a significant impact on the social and economic conditions of both individuals and countries [231]. It affects a wide range of industries, including communication, banking, home appliances, medicine, transportation, and civil aviation. However, the stock market is known for its high degree of nonlinearity and dynamism [232, 233], making investment decisions challenging. Additionally, the market's dynamic price fluctuations and chaotic behavior pose substantial challenges for price forecasting [234].

Over the past decade, researchers have developed various machine learning models to address stock market forecasting [235]. Notably, there has been a growing trend toward using deep learning models for stock forecasting, with promising results [236]. These models leverage historical financial data and have shown effectiveness in predicting stock market trends.

However, the stock market is also influenced by contextual events, such as financial news, user opinions, and investor decisions [229]. These factors contribute to the market's dynamic and nonlinear nature. To address this complexity, researchers have explored information fusion strategies for stock price and trend prediction [237].

The rise of Online Social Networks (OSNs) has had a profound impact on investor and consumer behavior in the stock market [228, 238, 239]. Recent studies [240, 241] have investigated the role of social media information in understanding investor behavior and predicting market trends. This became especially evident during the Gamestop (GME) squeeze event in early 2021, where online discussions and debates on platforms like Reddit influenced millions of novice traders, leading to significant market disruptions. The price of GME surged by a staggering 1700% during this period, causing substantial losses for major investors like Melvin Capital.

This chapter aims to explore how content shared on OSNs can impact investor decisions and, consequently, stock market forecasting. It does so by integrating historical financial data with social media data and evaluating their combined effect on prediction outcomes.

The chapter is structured as follows: Section 6.2 provides an overview of state-of-the-art approaches in stock prediction. Section 6.3 presents the proposed methodology, which integrates user-generated content from Online Social Networks (OSNs) with historical financial data. Section 6.4 outlines the experimental protocol, including hyperparameter optimization for training the stock forecasting model. Section 6.5 presents the obtained results and discusses key findings. Finally, Section 6.6 offers a summary of the methodology and results, along with insights into potential future research directions.

6.2 Related Work

In recent years, stock market prediction has become a prominent challenge in the finance domain [231, 242]. Despite the Efficient Market Hypothesis theory suggesting that consistently outperforming the market is impossible, researchers and practitioners have been actively developing approaches and methodologies to assist traders in achieving financial gains within the stock market [243].

Early stock forecasting approaches primarily relied on statistical methods [244, 245], which yielded subpar results due to the stock market's high volatility and nonlinearity [246, 232]. This inherent complexity posed significant challenges for investors seeking profits from their investments.

The proliferation of Artificial Intelligence (AI) models in the financial domain [247, 243] has offered opportunities for investors to gain profits. Machine learning models have been extensively developed for stock forecasting [228], with deep learning-based models gaining prominence due to the growing volume of financial data [235, 248, 249]. While deep learning models have shown promise in stock market prediction [236], their applications have limitations, as highlighted by Thakkar and Chaudhari [229].

The dynamic and nonlinear nature of the stock market necessitates the consideration of various data sources, such as financial news and content from Online Social Networks (OSNs) [234]. Some approaches have focused on analyzing news for stock forecasting but faced challenges due to the costly and time-consuming nature of news collection [250]. Others, like Zhai and Zhang [251], proposed LSTM-based forecasting models that combine historical data with features extracted from news using Deep Text Mining methods. Bayesian inference-based approaches, such as the one presented by Colasanto et al. [252], utilize polarity scores inferred from news through models like *AlBERTo*. In [253], authors propose an approach that investigates news related to homogeneous clusters of companies to integrate information from these clusters with the target company's data. Ray et al. [254] combined

Bayesian Structural Time Series with LSTM to investigate the influence of news sentiment on short-term stock price forecasting.

While news content can provide technical input, information shared on OSNs often exerts a more significant influence on user opinions [255]. Several approaches have integrated information from OSNs into stock prediction processes, as demonstrated by [238]. The rapid spread of information and related user opinions on OSNs has been observed across various domains [256, 257], emphasizing the importance of affective factors [258]. Jing et al. [259] designed an LSTM-based approach to predict stock prices by combining historical financial data with investor sentiment. In [240], authors integrated semantic features extracted using the BERT model, external structural characteristics inferred by Node2vec, and Tweet embeddings into a deep learning model with an attention mechanism, alongside historical stock prices. Lu et al. [241] used an Artificial Neural Network to predict stock market crises, combining market indicators, mixed-frequency investor sentiment, and historical data. Zheng et al. [260] investigated the collective behaviors of Reddit users, analyzing topics and user sentiment through interaction networks.

Despite the myriad approaches proposed in the literature, challenges persist due to the inherent nonlinear and dynamic behavior of the stock market. The key novelties relative to existing state-of-the-art analyses and approaches include:

- Adoption of a multi-modal approach, combining historical stock prices and sentiment time series, in contrast to [229], which focuses solely on analyzing AI models and heuristics using historical financial data.
- Correlation of information extracted from social networks with historical stock price data for stock prediction, while [260] primarily focuses on the analysis of social behavior during the GameStop squeeze event.
- Investigation of two different social networks to uncover potential correlation patterns, compared to [240, 260], which predominantly

focuses on a single social network (Twitter and Reddit, respectively), and [259], which concentrates on a specific investor forum.

- Extraction of three distinct time series for inferring daily sentiment (positive/neutral/negative) relative to a stock, as opposed to [240, 260], which amalgamate sentiment scores into a single value.

6.3 Methodology

In this section, the proposed framework for stock forecasting is described in detail, as depicted in Figure 6.1. The framework aims to combine content information from various Online Social Networks with historical financial data to predict the next-day stock opening price.

The framework comprises three main steps:

1. **Data Collection:** In the first step, information is collected from multiple data sources (represented as number ① in Figure 6.1).

2. **Pre-processing:** Subsequently, different pre-processing operations are applied to the collected data (number ② in Figure 6.1).

3. **Prediction Module:** The pre-processed data is then fed as input into the prediction module (number ③ in Figure 6.1), which is responsible for forecasting the next-day stock opening price.

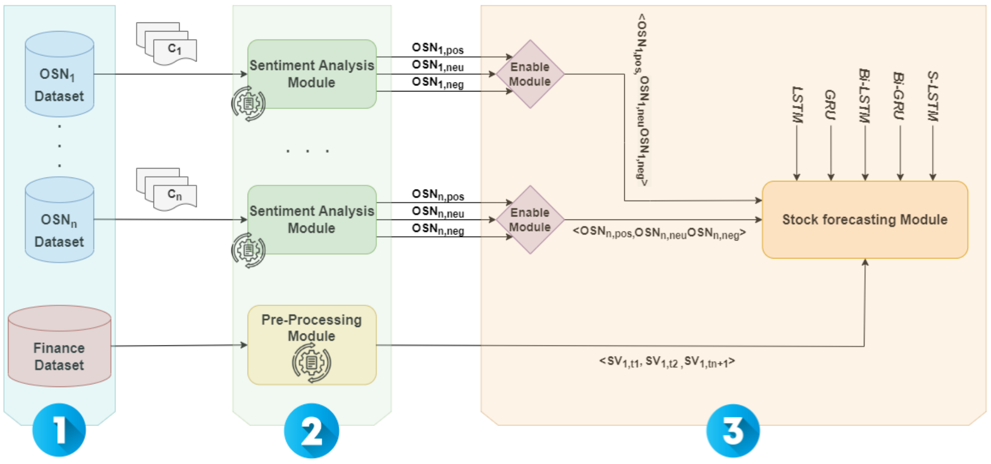


Figure 6.1: Overview of the proposed framework, which relies on three main steps: *Data ingestion*, *Pre-processing* and *Stock forecasting*.

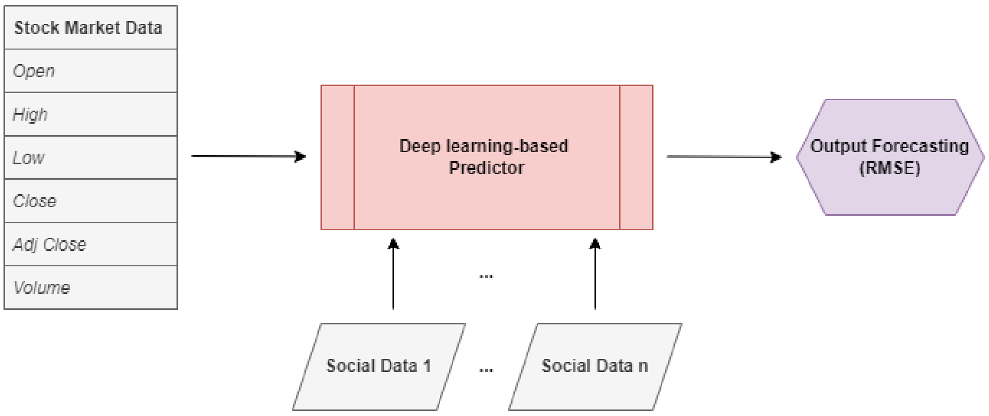


Figure 6.2: Description of the task in which different time series may be considered to forecast next day stock price.

The task involves utilizing diverse data sources, which include historical market data and textual content from various social media platforms [261].

The purpose of combining these different data sources is to enhance the accuracy of market predictions by taking into account its non-linear nature and high volatility. Figure 6.2 illustrates the heterogeneous data sources used in this task.

6.3.1 Data ingestion module

The first stage of the framework, as depicted in Figure 6.1, focuses on gathering data from various sources such as Online Social Networks and dedicated portals, in order to provide multiple perspectives on a single stock. To obtain user-generated text content from OSNs like comments, tweets, and posts, standard information retrieval methods are utilized. This involves the use of specific APIs (*snscrape* [8] and *pushshift pushshift*[9]) with different criteria based on the selected OSNs (e.g., keyword lists for Twitter or specific subreddits for Reddit).

Concurrently, financial data is crawled from dedicated portals, which contain historical market prices and transaction volume statistics. The historical financial information is collected using the *yfinance* API *yfinance*[10], and takes into account seven different features: open price, high price, low price, close price, adjusted price, close price, and volume.

6.3.2 Pre-processing module

The objective of the second module of the framework is to perform pre-processing and Sentiment analysis operations. First, special characters, symbols (e.g. @ , \, or #), and Uniform Resource Locators (URLs) are removed from user generated content as they do not affect user sentiment. Then, the *roBERTa* model is used to assign three sentiment scores (positive, neutral, and

[8]https://github.com/JustAnotherArchivist/snscrape
[9]https://github.com/pushshift/api
[10]https://pypi.org/project/yfinance/

negative) to each comment $c_i \in C$ (where C is the set of comments). Finally, the comments are aggregated daily to create three different time series which are then used as input for the Stock forecasting module.

Additionally, historical financial data is normalized using the *min-max* operation to scale the values in the range [0,1]. These normalized values are then used as input for the prediction module.

In summary, this module generates three distinct time series for each analyzed OSN, representing the daily sentiment scores for positive, negative, and neutral sentiments. It also produces a multivariate time series for the historical financial data as its output.

6.3.3 Stock forecasting module

The aim of the last module is forecasting the stock price for the next day. This is achieved by combining historical financial values and sentiment scores to enhance the accuracy of the prediction module. It should be noted that in Figure 6.1, one or more data sources can be chosen to assist the forecasting process. This process depends on five distinct deep learning models - Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), Bidirectional LSTM (Bi-LSTM), Bi-GRU, and Stacked-LSTM (S-LSTM) - to handle the given task.

6.4 Experimental analysis

In this section, a comprehensive description of the experimental analysis is provided, which was conducted to evaluate the approach across various datasets and time window lengths (refer to Section 6.4.1). Furthermore, the detailed process of hyper-parameter optimization employed to optimize each deep learning model used in the forecasting module is presented in Section 6.4.2.

6.4.1 Experimental protocol

The analysis's goal is to evaluate the efficiency and effectiveness of the proposed methodology. It also aims to investigate the impact of user-generated content published on Online Social Networks on stock prediction.

In summary, two analyses were conducted using different datasets and time window lengths:

- The efficiency of each deep learning model in terms of training time was assessed.
- The effectiveness of the proposed methodology was measured by comparing different deep learning models using Root Mean Square Error (RMSE), as defined in equation 6.1. Specifically, the cumulative RMSE for the validation set was calculated, while the test set was evaluated in terms of day-by-day RMSE.

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(Predicted_{\mathrm{i}} - Actual_{\mathrm{i}})^2}{N}} \tag{6.1}$$

Furthermore, each model has undergone optimization through the implementation of a greedy search, where the *learning rate* ($\in \{0.01, 0.001, 0.0001\}$) and *batch size* ($\in \{16, 32, 64, 128, 256\}$) were varied.

Various datasets have been constructed to investigate different real-world scenarios related to GameStop (GME) stock, based on the time span between January 1, 2021, and December 31, 2021. These datasets can be categorized as follows: i) only financial data obtained from *Yahoo! Finance*, ii) data obtained from both *Yahoo! Finance* and Twitter, iii) data obtained from both *Yahoo! Finance* and Reddit, and iv) consideration of data from all available sources (*Yahoo! Finance*, Twitter, and Reddit). The characteristics of these datasets are presented in Table 6.1.

Dataset	Amount	Data type	Average	STD
Twitter	893,400	Tweet	2247.67	652.26
Reddit	41,436	Forum comments	113.52	19.21
Yahoo! Finance	2,184	Open Price;High Price;Low Price; Close Price Adj Price; Close Price;Volume	1	1

Table 6.1: Dataset characterization of the collected dataset based on the time span between January 1, 2021 and December 31, 2021. It is worth to note that the average and standard deviation are daily computed.

The methodology has been implemented on Google Colaboratory [11], in which deep learning models have been built by using *Keras*[12], *TensorFlow V2*[13] and *pyts*[14] for time series analysis.

6.4.2 Hyperparameter optimization

In this section, the hyper-parameter optimization made for each model is described, with the aim of identifying the best parameters by performing a grid search. This search considers 15 different configurations, which are detailed in Table 6.2.

[11]https://colab.research.google.com/
[12]https://keras.io/
[13]https://www.tensorflow.org/
[14]https://pyts.readthedocs.io/

Configuration	**Hyperparameters**	
	Batch size	**Learning rate**
1	16	0.01
2	16	0.001
3	16	0.0001
4	32	0.01
5	32	0.001
6	32	0.0001
7	64	0.01
8	64	0.001
9	64	0.0001
10	128	0.01
11	128	0.001
12	128	0.0001
13	256	0.01
14	256	0.001
15	256	0.0001

Table 6.2: Hyper-parameter configurations used during the grid search strategy for each model.

The hyper-parameter optimization was performed by varying the time window size to evaluate its effect on stock forecasting. The results for time windows of 8 and 16 days are shown in Figure 6.3 and 6.4 (6.5 and 6.6). It is evident that configurations 1, 4, 5, 6, and 12 exhibit a balance between efficiency and efficacy. Furthermore, the models trained on 16 days outperformed those trained on 8 days, leading to a focus on the longer time window.

In terms of model efficacy, the GRU model generally had shorter training times compared to the other models. However, in the case of the dataset integrating data from both social networks and historical financial data, the LSTM model was the fastest but less effective than GRU. Configurations

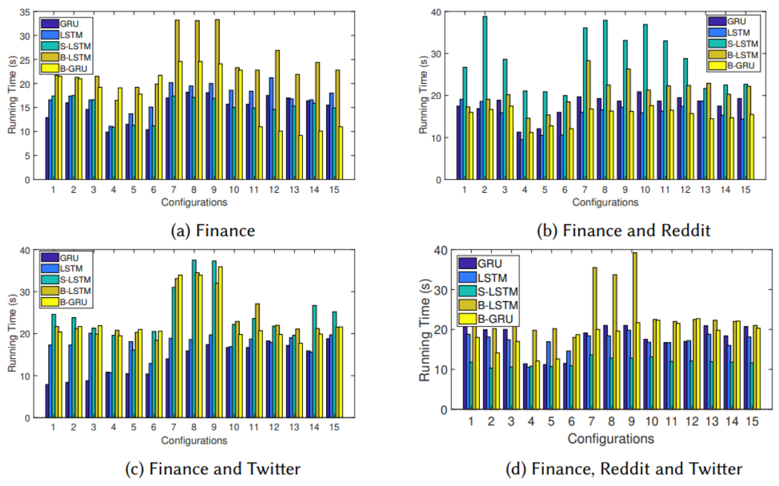


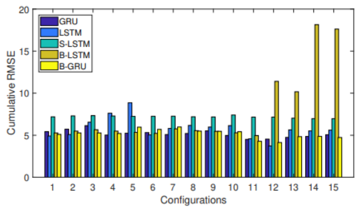
Figure 6.3: Running time analysis varying datasets and fixed time window equal to 8.

3, 4, and 5 achieved significantly faster training times with GRU, while configurations with a *batch size* of 16 showed similar efficiency. The best configurations considering both efficiency and effectiveness were GRU with a *batch size* of 16 and *learning rate* of 0.01 and LSTM with a *batch size* of 128 and *learning rate* of 0.0001. Both models were slightly better than GRU with a 16-day time window in terms of efficiency but significantly less effective.

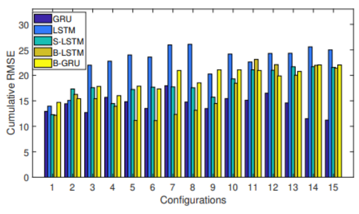
Lastly, Figure 6.7 illustrates the characteristics of each deep learning model at the end of the hyper-parameter optimization phase, including the number of layers and trainable parameters.

6.5 Results

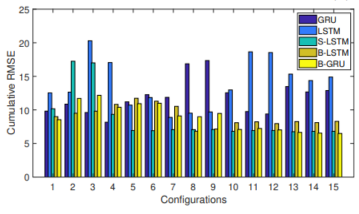
In this section, the effectiveness results once the hyper-parameter optimization of each model is performed, as detailed in Section 6.4.2, are discussed. The table 6.3 depicts the mean and standard deviation of RMSE evaluated on the test set.



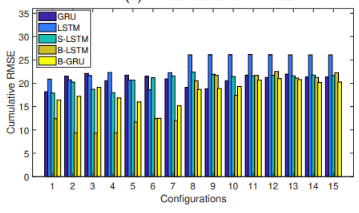
(a) Finance



(b) Finance and Reddit



(c) Finance and Twitter

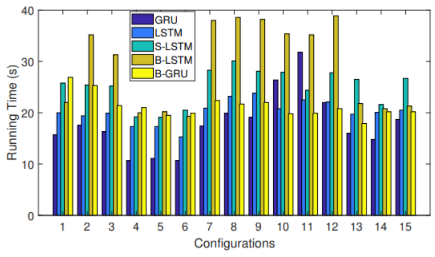


(d) Finance, Reddit and Twitter

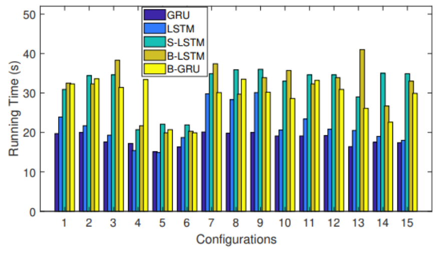
Figure 6.4: RMSE training set varying datasets and fixed time window equal to 8.

		YF		**YF+Reddit**		**YF+Twitter**		**YF+All**	
	Models	**Mean**	**STD**	**Mean**	**STD**	**Mean**	**STD**	**Mean**	**STD**
16 days	GRU	2.552	0.523	3.541	0.244	4.717	0.378	6.147	0.519
	LSTM	3.965	1.039	6.374	1.109	4.323	0.992	6.530	0.698
	S-LSTM	6.007	0.258	6.897	0.625	6.025	0.672	5.907	0.467
	B-LSTM	3.437	0.185	6.767	1.003	4.854	1.279	6.201	0.862
	B-GRU	3.410	0.469	5.661	0.241	4.240	0.314	4.894	0.128
8 days	GRU	4.570	0.738	6.723	1.599	4.688	1.040	8.819	0.292
	LSTM	3.834	0.151	7.694	0.624	5.100	0.429	10.101	1.229
	S-LSTM	5.858	0.776	6.476	0.089	5.401	0.169	8.588	0.904
	B-LSTM	4.036	1.073	5.201	0.160	4.396	0.092	4.649	0.124
	B-GRU	3.889	0.283	7.154	0.224	4.539	0.029	8.712	2.666

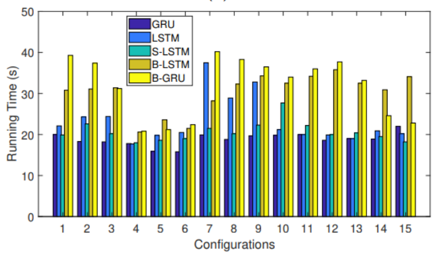
Table 6.3: Results of each deep learning model in terms of RMSE over the entire test set. In particular, YF and All stand for Yahoo Finance and both OSNs, respectively.



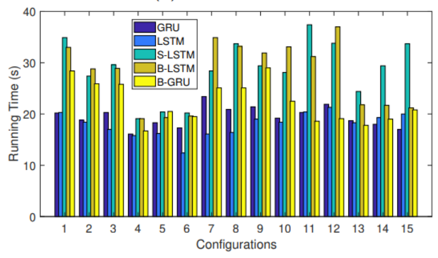
(a) Finance



(b) Finance and Reddit



(c) Finance and Twitter

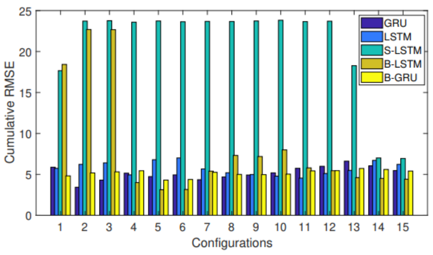


(d) Finance, Reddit and Twitter

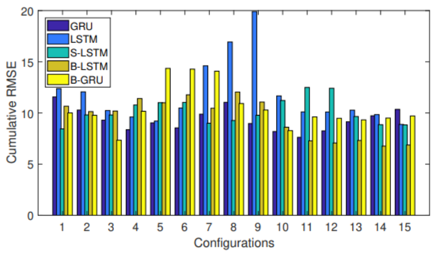
Figure 6.5: Running time analysis varying datasets and fixed time window equal to 16.

Despite the consideration of the influence of OSNs on different tasks by Zheng et al. [260], the best results in terms of RMSE on the entire set are achieved using only financial data. In terms of models, it is evident that GRU achieves better results when considering the 16-day window due to its ability to discard long-term information, while LSTM performs better on the 8-day window by considering the temporal relationship over the entire analysis period. When it comes to social networks, Reddit provides more information, as evidenced by the observed effect of squeeze in some subreddits [262, 263].

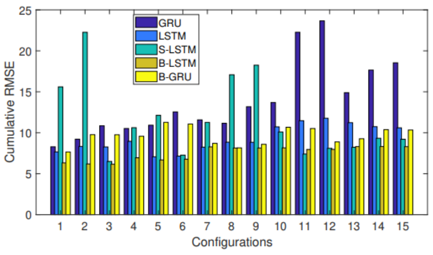
Furthermore, more insight about the obtained results has been provided by investigating the day-by-day RMSE evaluated on the test set considering all the possible datasets. Specifically, the number of tweets and comments that have been published on Twitter and Reddit in the days of the test set are shown in Figure 6.8, respectively.



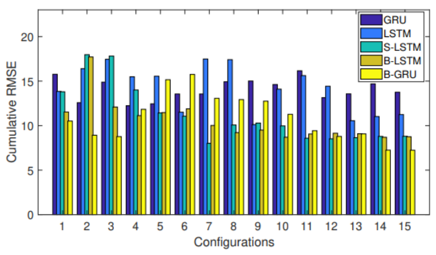
(a) Finance



(b) Finance and Reddit



(c) Finance and Twitter

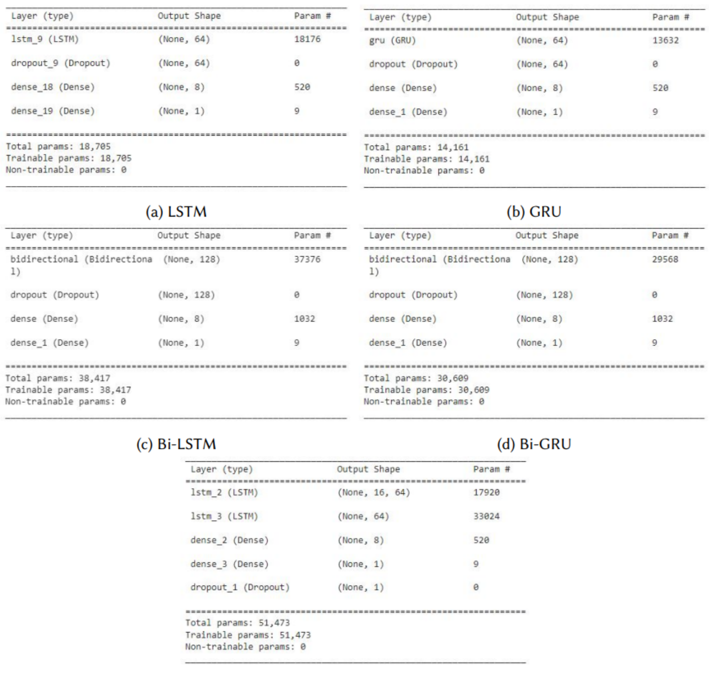


(d) Finance, Reddit and Twitter

Figure 6.6: RMSE training set varying datasets and fixed time window equal to 16.

It is observed that there are some days when there is a predominance of tweets compared to Reddit comments and vice versa. Specifically, the days with a high number of tweets and low number of Reddit comments are highlighted in blue, while the days with a high number of Reddit comments and low number of tweets are highlighted in orange. On the other hand, the days with a high number of information published on both social media platforms are highlighted in red. In terms of RMSE values, the days indicated in blue and orange show higher values when considering the dataset with Twitter and Reddit comments respectively. However, when considering financial data and data from both social media platforms, a higher RMSE is achieved on the days highlighted in red.

Despite previous studies highlighting the potential impact of social dynamics on stock prediction, our analysis reveals that the use of social data does not always improve the prediction task. This can be observed in Figure



```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_9 (LSTM)                (None, 64)                18176

dropout_9 (Dropout)          (None, 64)                0

dense_18 (Dense)             (None, 8)                 520

dense_19 (Dense)             (None, 1)                 9

=================================================================
Total params: 18,705
Trainable params: 18,705
Non-trainable params: 0
_________________________________________________________________
```

(a) LSTM

```
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 64)                13632

dropout (Dropout)            (None, 64)                0

dense (Dense)                (None, 8)                 520

dense_1 (Dense)              (None, 1)                 9

=================================================================
Total params: 14,161
Trainable params: 14,161
Non-trainable params: 0
_________________________________________________________________
```

(b) GRU

```
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional (Bidirectiona  (None, 128)               37376
l)

dropout (Dropout)            (None, 128)               0

dense (Dense)                (None, 8)                 1032

dense_1 (Dense)              (None, 1)                 9

=================================================================
Total params: 38,417
Trainable params: 38,417
Non-trainable params: 0
_________________________________________________________________
```

(c) Bi-LSTM

```
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional (Bidirectiona  (None, 128)               29568
l)

dropout (Dropout)            (None, 128)               0

dense (Dense)                (None, 8)                 1032

dense_1 (Dense)              (None, 1)                 9

=================================================================
Total params: 30,609
Trainable params: 30,609
Non-trainable params: 0
_________________________________________________________________
```

(d) Bi-GRU

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 16, 64)            17920

lstm_3 (LSTM)                (None, 64)                33024

dense_2 (Dense)              (None, 8)                 520

dense_3 (Dense)              (None, 1)                 9

dropout_1 (Dropout)          (None, 1)                 0

=================================================================
Total params: 51,473
Trainable params: 51,473
Non-trainable params: 0
_________________________________________________________________
```

(e) S-LSTM

Figure 6.7: Configuration of each deep learning model at the end of the hyper-parameter optimization.

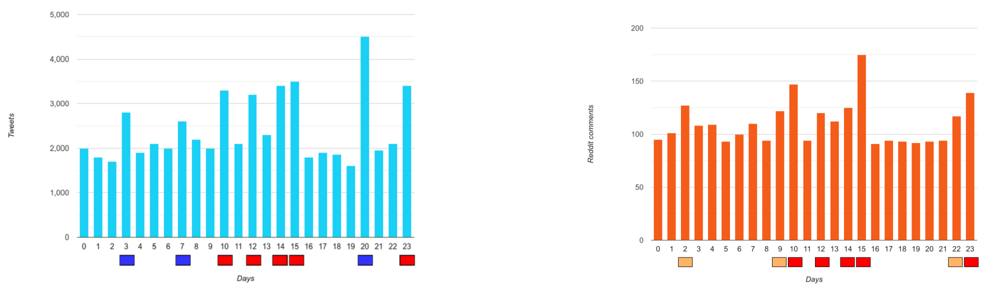


Figure 6.8: Number of Tweets and Reddit comments per days.

6.8, where the network's performance does not improve on days 4, 5, and 6 despite having over 2000 tweets. Therefore, it can be concluded that while social data has an impact on stock prediction, it is not always beneficial even when there is significant informational content.

6.6 Conclusion

In the realm of FinTech, stock market forecasting stands out as a significant challenge, attracting increasing attention due to the substantial daily trading volumes on stock exchanges [264]. This chapter introduces a framework that combines historical data with content derived from user-generated posts on Online Social Networks (OSNs) to enhance stock forecasting. The case study revolves around GME and assesses various deep learning models and the influence of text data extracted from Twitter and Reddit on stock prediction performance. The proposed approach introduces three key innovations: i) the development of a multimodal methodology that merges historical stock prices with sentiment scores from diverse OSNs, ii) the exploration of potential correlations between different social networks and historical data during the GameStop squeeze, and iii) the extraction of distinct sentiment scores for each day.

The analysis underscores the significance of examining user-generated content from multiple OSNs for effective stock forecasting, considering various deep learning models and time windows. Moreover, the investigation of distinct OSNs provides varied perspectives that may unveil correlations in data analysis. However, the ever-shifting sentiments of OSN users can adversely affect stock prediction tasks, particularly with the spread of misleading news, as demonstrated in previous studies [265–267].

Future research endeavors will broaden the scope to encompass a wider range of stocks and multimedia social networks like Instagram and Facebook.

Additionally, the integration of eXplainable Artificial Intelligence (XAI) tools will empower practitioners in their decision-making processes.

7

Benchmarking stock prediction models exploiting social data and news

7.1 Introduction

The GameStop squeeze, which took place in late January 2021, involved a group of amateur traders on the Reddit forum r/wallstreetbets. They coordinated the purchase of GameStop (GME) shares, a struggling video game retailer, with the aim of driving up the stock price. The goal was to trigger a "short squeeze," compelling investors who had bet against GameStop (by "shorting" the stock) to buy shares and limit their losses, causing the stock price to rise further.

This coordinated buying led to a significant increase in GameStop's stock price and resulted in substantial losses for hedge funds that had shorted the stock. The GameStop squeeze also had broader implications for the financial market, raising questions about market fairness and the influence of retail traders using social media coordination. Some called for increased market regulation, while others viewed it as a triumph for individual investors against Wall Street.

In summary, the GameStop squeeze marked a historic moment in finance, highlighting how social media and online communities can impact financial markets. Social media, particularly the Reddit forum r/wallstreetbets, played a pivotal role in this short squeeze by facilitating information sharing and coordinated buying. This collective action empowered individual investors and contributed to the squeeze's success.

Furthermore, social networks can influence the stock market in various ways, including information dissemination, collective action, shaping market sentiment, and spreading rumors. While they offer opportunities for information sharing and collective decision-making, caution is necessary due to the potential spread of misinformation.

Additionally, news from reputable sources can also influence investor decisions, impacting stock prices based on factors like earnings reports, regulatory issues, and macroeconomic news.

This Chapter aims to address research questions related to the influence of social and news data on the stock market, the most suitable data sources, combining such data with market data, forecasting techniques, model selection, performance metrics, and the correlation between market and social/news data for specific stocks. Specifically:

- RQ_1: Can social or news data impact the stock market by influencing investor decisions based on sentiment associated with such information?

- RQ_2: What are the optimal data sources, such as social media platforms (e.g., Twitter, Reddit) and news websites, for maximizing the utility of this data?

- RQ_3: How can social and news data from selected sources be effectively utilized and integrated with market data to enhance stock prediction? Do sentiment-related time series from social and news data exhibit distinct or similar temporal patterns compared to market data? How does data volume and availability over specific time intervals affect the quality of analysis?

- RQ_4: What forecasting techniques prove most effective when dealing with stock prediction in the presence of diverse data sources?
- RQ_5: Are Machine Learning (ML) and Deep Learning (DL) methods the most promising approaches? Which ML/DL models demonstrate the highest performance for our task?
- RQ_6: Which performance metrics are best suited for selecting the optimal model for stock prediction, considering the various types of data?
- RQ_7: In which basket of stocks is the correlation between market data and social/news data most pronounced? Can such data serve as a robust test case for the chosen models?

The Chapter is organized into sections covering related work (7.2), the proposed system overview (7.3), results and discussion (7.4), and conclusions and future work (7.5).

7.2 Related Work

The relationship between news sentiment, social network comments, and their influence on the stock market has garnered significant research attention in the financial field [268]. Recent research suggests that news sentiment and comments on social media platforms can exert a substantial impact on stock market trends and serve as valuable indicators of stock price movements [269, 270]. Machine Learning (ML) and Deep Learning (DL) techniques have been employed to analyze extensive data from various sources, including news articles and social media platforms, resulting in more accurate stock market predictions compared to traditional statistical models [271]. The superiority of these models is widely acknowledged in the realm of stock market prediction [272, 273].

Sentiment analysis plays a pivotal role in understanding the stock market. Analyzing sentiments from diverse sources provides insights into how the stock market responds to different categories of news over varying timeframes—short-term, medium-term, and long-term. As a result, sentiment analysis has emerged as a novel approach to assess the impact of news sentiment on financial markets [274].

Reddit and Twitter are frequently utilized as sources for sentiment analysis due to their popularity and the availability of publicly accessible data [275–277]. In these studies, major news websites such as Financial Times, The Wall Street Journal, Bloomberg, Reuters, Forbes, Yahoo Finance, and Business Insider are commonly used as news sources [270, 272, 278].

Khan and Ghazanfar [279] adopted a unique approach to sentiment analysis compared to previous studies. Their approach encompasses not only news articles but also posts from social media platforms, especially Twitter. The authors collected data from three primary sources: stock data from Yahoo! Finance, news articles from Business Insider, and tweets containing stock ticker symbols preceded by the dollar sign ($), such as '$NYSE' and '$HPQ'. These tweets provided crucial attributes: Source, TweetText, and Date. The study explored various Machine Learning algorithms, including Support Vector Machine, Logistic Regression, and Random Forest, revealing that some markets are more influenced by news sources, while others are more affected by social media platforms.

Another proposed method, S_I_LSTM [280], integrates multiple data sources, encompassing traditional historical stock data, technical indicators, stock-related posts, and financial news for stock price prediction. This approach combines sentiment index, technical indicators, and stock historical transaction data as features for stock price prediction and employs the Long Short-Term Memory (LSTM) network for predicting the China Shanghai A-share market. The study demonstrates the model's effectiveness in predicting stock closing prices with lower errors compared to using a single data source, as assessed through Mean Absolute Error (MAE), Mean Square Error (MSE), and Root Mean Square Error (RMSE).

In contrast, another study [281] underscores the challenges of predicting stock prices due to their multifaceted influences and inherent unpredictability. The authors propose a machine learning model based on the Long-Short Term Memory (LSTM) algorithm designed to predict stock price data. Their system operates in real-time using the Yahoo Finance API and incorporates Twitter sentiment analysis as a feature to provide public opinion insights on specific stocks. The model successfully generates prediction graphs with minimal error, particularly for Asian Paints data.

Conversely, a different study [282] introduces an approach reliant on Ensemble Learning and Sentiment Analysis. This approach utilizes a dataset comprising historical data, technical indicators, and sentiment features for 50 stocks in the Nifty 50 Index, spanning two different time horizons (1 year and 5 years). The dataset is augmented with sentimental features such as Wikipedia page hits and Google News mentions for each company. The study compares the performance of state-of-the-art machine learning models, including XGBoost, LSTM, ARIMA, and SVR, in predicting stock closing prices. The results highlight the superior performance of LSTM and XGBoost in terms of RMSE.

7.3 System Overview

This section presents the structural design of the system, as depicted in Figure 7.1. It will subsequently provide a comprehensive explanation of the key stages comprising the presented workflow.

The selected methodology revolves around the concept of information integration, specifically employing early feature fusion. This approach enables the merging of multiple data sources at the outset of the forecasting process. In the context of stock prediction, these data sources encompass financial data, news articles, and sentiment analysis from social media. By adopting early feature fusion, the predictive model gains the capacity to capture intricate relationships and dependencies among these diverse data sources, potentially enhancing its predictive accuracy [87].

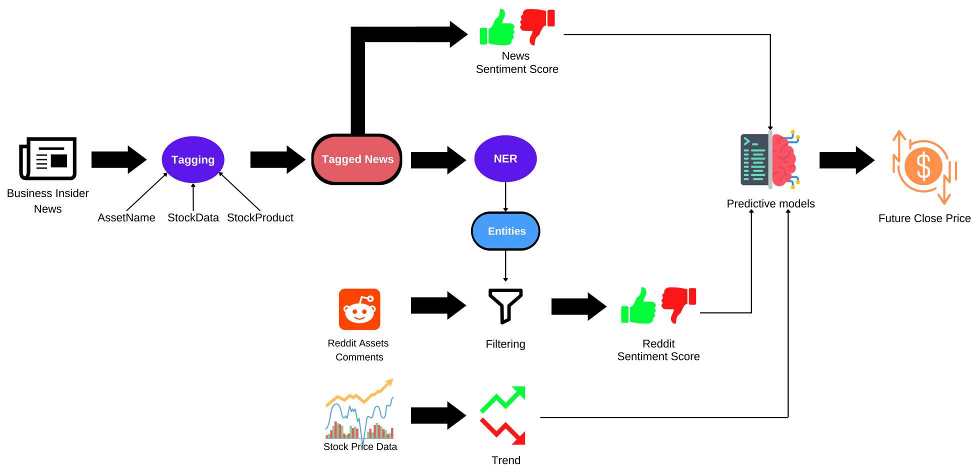


Figure 7.1: Overview of the proposed framework

Data Extraction As explained in Section 7.1, this research is focused on utilizing historical price data, stock news, and user comments from social media for precise stock prediction. The choice of these data sources was influenced by a comprehensive literature review, where previous studies frequently investigated stocks belonging to the S&P 500 index, particularly *Apple, Amazon, Tesla, Google*, and *Meta*. This index encompasses 500 companies, representing approximately 80% of the total market capitalization. To collect stock news, the commonly used platform Business Insider was selected, as it aggregates news articles from reputable sources such as Reuters and Financial Times, as documented in previous research [279, 283]. For extracting sentiment from social media, Reddit was chosen due to its active user community engaged in stock market discussions. All comments containing the stock name or full company name, regardless of the case (uppercase, lowercase, or mixed), were extracted for analysis. Additionally, historical market data was obtained from Yahoo! Finance. In this study, the data spanned from July 14, 2022, to August 23, 2022, covering the specified time period.

Data Processing After identifying the required data for constructing the dataset, the subsequent step involves processing the extracted data of various types. Regarding the price history data, specific attributes were derived for each security, including the opening price, highest price reached, lowest price touched, closing price, and traded volume on a given date.

Additionally, additional features of potential interest were calculated. One such feature is the Previous Trend attribute, which was defined and computed as follows:

$$PT_n = \begin{cases} \text{Positive}, & \text{if } C_n > C_{n-1} \\ \text{Negative}, & \text{if } C_n < C_{n-1} \end{cases}$$

Here, PT_n represents the Previous Trend on day n, C_n denotes the closing price on day n, and C_{n-1} refers to the closing price on day $n-1$. Furthermore, the Future Close Price was calculated as the closing price of the stock on the next day, which is the value to be predicted in the final stages of the process.

Different operations were performed on the news and Reddit comments. For the news data, an approach was sought to associate each news story with a specific asset. This was achieved by searching for the stock name and associated company in both the headline and the text of the news articles.

To enhance the accuracy of tagging unclassified news, two JSON files were created. These files contained, for each stock, the names of key individuals working in the associated companies (including competitors) and the products introduced to the market. This additional information facilitated a more precise search within the news to enable effective tagging.

Once all 3618 news items were labeled, the five equities with the highest number of news items were selected for further analysis. Specifically, these stocks are Meta (with 292 news items), Apple (with 264 news items), Amazon (with 262 news items), Tesla (with 89 news items), and Google (with 65 news items).

After determining the assets to be considered and their corresponding news items, an approach was developed to filter the Reddit comments to make them relevant. This was achieved by applying Name Entity Recognition (NER), which involves identifying and categorizing key information (entities) in a text. Entities can consist of individual words or phrases consistently referring to the same entity. Each identified entity is then classified into a predetermined category.

The NER procedure involved extrapolating entities from the news headlines and saving them in a JSON file. For each stock, this file contains the date along with the associated keywords extracted from the news published on that date.

Subsequently, the extracted keywords from the news stories were used to search for related comments within the Reddit dataset.

Sentiment analysis for Reddit comments and news The data required for this study were obtained as outlined in the previous section. It is now essential to determine the sentiment associated with the headlines, text content of news articles, and Reddit comments. Each news headline, article text, and Reddit comment received a sentiment score, with values ranging from -1, signifying negative sentiment, to 1, representing positive sentiment. Figure 7.2 provides an overview of the sentiment score calculation process.

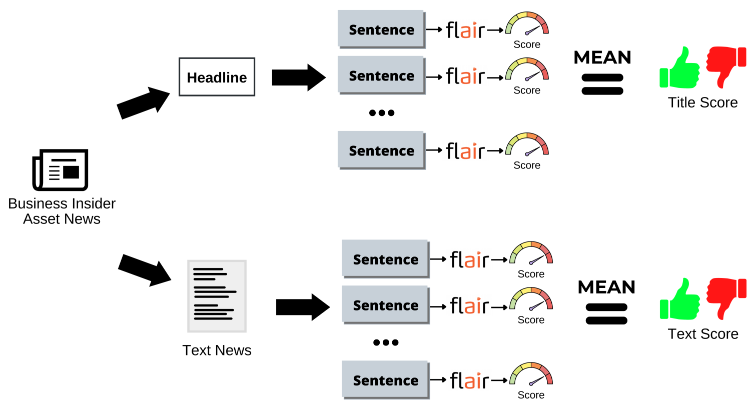


Figure 7.2: Extrapolation of the sentiment score

7.3.1 Dataset and Metrics

The dataset used for training predictive algorithms underwent several stages of processing to derive a subset from a larger dataset. Initially, a total of 3,618 news items were extracted from Business Insider, covering the period from July 14, 2022, to August 23, 2022. Each news item was associated with the following attributes:

- **ID:** An identifier generated from the last 16 characters of the SHA256 hash encoding of the news title.
- **Title:** The headline of the news.
- **Text:** The body of the news item.
- **Datasite:** The publication date of the news item on the website.

- **DataScraping:** The date when the news item was extracted by the crawler, useful when website publication dates were unavailable.
- **Source:** The link to the page containing the full news item.
- **Premium:** A boolean variable indicating whether the news content might be restricted to premium users.

Following this, the news items were labeled, and the analysis focused solely on the five stocks with the highest number of instances: Apple, Amazon, Google, Meta, and Tesla.

For the Reddit data, a total of $1,672,967$ comments were collected for these five stocks, and each instance was characterized by the following attributes:

- **author:** The username of the commenter.
- **body:** The content of the comment.
- **id:** The identifier associated with the user.
- **link:** The link to the comment.
- **subreddit:** The subreddit where the comment was posted.
- **subreddit_id:** The identifier of the subreddit.
- **subreddit_type:** Indication of whether the subreddit is public or private.
- **datetime:** The date the comment was made.

The Reddit posts were further refined using named entity recognition (NER) keywords, resulting in the following number of comments for each stock:

- **Apple:** 39,665 instances.

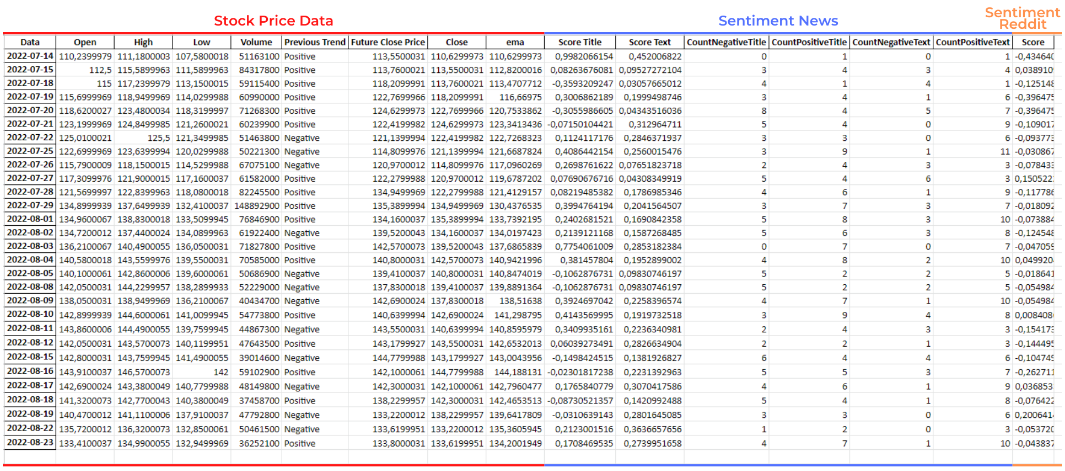
- **Amazon:** 19,675 instances.
- **Tesla:** 5,343 instances.
- **Google:** 92,390 instances.
- **Meta:** 16,283 instances.

The final dataset used for experiments combined data from both news items and Reddit comments, resulting in instances corresponding to the number of trading days between July 14, 2022, and August 23, 2022, for each stock. Given that the stock market is not open every day, each asset had 29 instances in the dataset.

During the experiments, the dataset was employed in four distinct configurations:

1. **Stock Price Data:** This configuration included data from Yahoo Finance.
2. **Stock Price Data + Sentiment News:** It incorporated data from Yahoo Finance, sentiment scores extracted from news, and the count of positive and negative news items.
3. **Stock Price Data + Reddit Sentiment:** In addition to the attributes from the Stock Price Data configuration, this setup included a sentiment score derived from Reddit posts for each stock.
4. **Stock Price Data + Sentiment News + Sentiment Reddit:** This comprehensive configuration merged all data from the previous setups.

Figure 7.3 provides a visual summary of the resulting final dataset, showcasing the attributes associated with each configuration.



Stock Price Data									Sentiment News						Sentiment Reddit
Data	Open	High	Low	Volume	Previous Trend	Future Close Price	Close	ema	Score Title	Score Text	CountNegativeTitle	CountPositiveTitle	CountNegativeText	CountPositiveText	Score
2022-07-14	110,2399979	111,1800003	107,5800018	51163100	Positive	113,5500031	110,6299973	110,6299973	0,9982066154	0,452006822	0	1	0	1	-0,43464
2022-07-15	112,5	115,5899963	111,5899963	84317800	Positive	113,7600021	113,5500031	112,8200016	0,08263676081	0,09527272104	3	4	3	4	0,03891
2022-07-18	115	117,2399979	113,1500015	59115400	Positive	118,2099991	113,7600021	113,4707712	-0,3593209247	0,03057665012	4	1	4	1	-0,12514
2022-07-19	115,6999969	118,9499969	114,0299988	60990000	Positive	122,7699966	118,2099991	116,66975	0,3006862189	0,1999498746	3	4	1	6	-0,39647
2022-07-20	118,6200027	123,4800034	118,3199997	71268300	Positive	124,6299973	122,7699966	120,7533862	-0,3055986605	0,04343516036	8	4	5	7	-0,39647
2022-07-21	123,1999969	124,8499985	121,2600021	60239900	Positive	122,4199982	124,6299973	123,3413436	-0,07150104421	0,312964711	5	4	0	9	-0,10901
2022-07-22	125,0100021	125,5	121,3499985	51463800	Negative	121,1399994	122,4199982	122,7268323	0,1124117176	0,2846371937	3	3	0	6	-0,09377
2022-07-25	122,6999969	123,6399994	120,0299988	50221300	Negative	114,8099976	121,1399994	121,6687824	0,4086442154	0,2560015476	3	9	1	11	-0,03086
2022-07-26	115,7900009	118,1500015	114,5299988	67075100	Negative	120,9700012	114,8099976	117,0960269	0,2698761622	0,07651823718	2	4	3	3	-0,07843
2022-07-27	117,3099976	121,9000015	117,1600037	61582000	Positive	122,2799988	120,9700012	119,6787202	0,07690676716	0,04308349919	5	4	6	3	0,15052
2022-07-28	121,5699997	122,8399963	118,0800018	82245500	Positive	134,9499969	122,2799988	121,4129157	0,08219485382	0,1786985346	4	6	1	9	-0,11778
2022-07-29	134,8999939	137,6499939	132,4100037	148892900	Positive	135,3899994	134,9499969	130,4376535	0,3994764194	0,2041564507	3	7	3	7	-0,01809
2022-08-01	134,9600067	138,8300018	133,5099945	76846900	Positive	134,1600037	135,3899994	133,7392195	0,2402681521	0,1690842358	5	8	3	10	-0,07388
2022-08-02	134,7200012	137,4400024	134,0899963	61922400	Negative	139,5200043	134,1600037	134,0197423	0,2139121168	0,1587268485	5	6	3	8	-0,12454
2022-08-03	136,2100067	140,4900055	136,0500031	71827800	Positive	142,5700073	139,5200043	137,6865839	0,7754061009	0,2853182384	0	7	0	7	-0,04705
2022-08-04	140,5800018	143,5599976	139,5500031	70585000	Positive	140,8000031	142,5700073	140,9421996	0,381457804	0,1952899002	4	8	2	10	0,04992
2022-08-05	140,1000061	142,8600006	139,6000061	50686900	Negative	139,4100037	140,8000031	140,8474019	-0,1062876731	0,09830746197	5	2	2	5	-0,018641
2022-08-08	142,0500031	144,2299957	138,2899933	52229000	Negative	137,8300018	139,4100037	139,8891364	-0,1062876731	0,09830746197	5	2	2	5	-0,05498
2022-08-09	138,0500031	138,9499969	136,2100067	40434700	Negative	142,6900024	137,8300018	138,51638	0,3924697042	0,2258396574	4	7	1	10	-0,05498
2022-08-10	142,8999939	144,6000061	141,0099945	54773800	Positive	140,6399994	142,6900024	141,298795	0,4143569995	0,1919732518	3	9	4	8	0,00840
2022-08-11	143,8600006	144,4900055	139,7599945	44867300	Negative	143,5500031	140,6399994	140,8595979	0,3409935161	0,2236340981	2	4	3	3	-0,15417
2022-08-12	142,0500031	143,5700073	140,1199951	47643500	Positive	143,1799927	143,5500031	142,6532013	0,06039273491	0,2826634904	2	2	1	3	-0,14449
2022-08-15	142,8000031	143,7599945	141,4900055	39014600	Negative	144,7799988	143,1799927	143,0043956	-0,1498424515	0,1381926827	6	4	4	6	-0,10474
2022-08-16	143,9100037	146,5700073	142	59102900	Positive	142,1000061	144,7799988	144,188131	-0,02301817238	0,2231392963	5	5	3	7	-0,262711
2022-08-17	142,6900024	143,3800049	140,7799988	48149800	Negative	142,3000031	142,1000061	142,7960477	0,1765840779	0,3070417586	4	6	1	9	0,03685
2022-08-18	141,3200073	142,7700043	140,3800049	37458700	Positive	138,2299957	142,3000031	142,4653513	-0,08730521357	0,1420992488	5	4	1	8	-0,07642
2022-08-19	140,4700012	141,1100006	137,9100037	47792800	Negative	133,2200012	138,2299957	139,6417809	-0,0310639143	0,2801645085	3	3	0	6	0,20064
2022-08-22	135,7200012	136,3200073	132,8500061	50461500	Negative	133,6199951	133,2200012	135,3605945	0,2123001516	0,3636657656	1	2	0	3	-0,05372
2022-08-23	133,4100037	134,9900055	132,9499969	36252100	Positive	133,8000031	133,6199951	134,2001949	0,1708469535	0,2739951658	4	7	1	10	-0,04383

Figure 7.3: Final Dataset

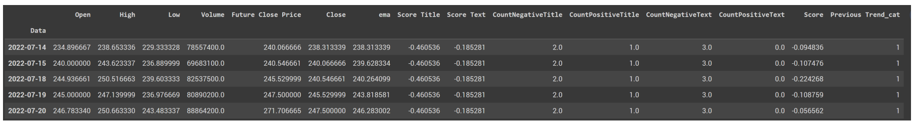
Before utilizing the derived data to train predictive models, a series of preprocessing steps were applied. Initially, adjustments were made to the Stock Price Data configuration. An additional attribute called Exponential Moving Average (EMA) was introduced, a commonly used indicator in stock trading. The EMA serves as a trend indicator by computing the average of daily closing prices over a specified period. EMA values were calculated based on the Close Prices using the following formula:

$$\text{EMA}_t = \text{EMA}_{t-1} + \alpha \times (\text{Close Price}_t - \text{EMA}_{t-1})$$

In this formula, EMA_t represents the EMA at time t, EMA_{t-1} represents the EMA at time $t-1$, Close Price_t represents the close price at time t, and α denotes the smoothing factor.

Additionally, an encoding operation was carried out on the Stock Price Data to convert the values in the "Previous Trend" column from strings ("Positive" and "Negative") to integers (1 and 0).

The resulting modified dataset, encompassing these changes, is visually represented in Figure 7.4.



	Open	High	Low	Volume	Future Close Price	Close	ema	Score Title	Score Text	CountNegativeTitle	CountPositiveTitle	CountNegativeText	CountPositiveText	Score	Previous Trend_cat
Data															
2022-07-14	234.896667	238.653336	229.333328	78557400.0	240.066666	238.313339	238.313339	-0.460536	-0.185281	2.0	1.0	3.0	0.0	-0.094836	1
2022-07-15	240.000000	243.623337	236.889999	69683100.0	240.546661	240.066666	239.628334	-0.460536	-0.185281	2.0	1.0	3.0	0.0	-0.107476	1
2022-07-18	244.936661	250.516663	239.603333	82537500.0	245.529999	240.546661	240.264099	-0.460536	-0.185281	2.0	1.0	3.0	0.0	-0.224268	1
2022-07-19	245.000000	247.139999	236.976669	80890200.0	247.500000	245.529999	243.818581	-0.460536	-0.185281	2.0	1.0	3.0	0.0	-0.108759	1
2022-07-20	246.783340	250.663330	243.483337	88864200.0	271.706665	247.500000	246.283002	-0.460536	-0.185281	2.0	1.0	3.0	0.0	-0.056562	1

Figure 7.4: Final Dataset

Once it was confirmed that the dataset comprised numerical data, a scaling operation was performed to enhance the quality of the results. The dataset underwent scaling using the StandardScaler, a transformation that adjusts the data to have a mean of zero and a standard deviation of one, conforming to the standard normal distribution (z-distribution). This scaling operation facilitates quicker convergence and improves the effectiveness of the statistical model. The scaling formula employed is as follows:

$$z = \frac{x - \mu}{\sigma}$$

Here, z represents the standardized value, x is the original value, μ denotes the mean, and σ signifies the standard deviation.

In summary, the dataset was meticulously prepared and preprocessed to be employed in various configurations, with the additional scaling step implemented to enhance result quality.

This study specifically focuses on three widely recognized evaluation metrics commonly utilized in the literature for stock prediction tasks:

Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) [284, 285, 234].

The adoption of these metrics serves the purpose of evaluating the predictive performance of stock prediction models that leverage both financial data and social data sourced from Reddit and news channels.

Reddit was chosen as a social data source due to its extensive usage as a platform for financial discussions and insights [286–288]. Furthermore, we incorporated news channels from Business Insider, renowned for its comprehensive coverage of financial and business news [270, 272, 278, 279]. By combining financial and social data, our study aims to explore the influence of market sentiment and news events on the performance of our stock prediction models.

Mean Absolute Error (MAE) In the realm of statistics, Mean Absolute Error (MAE), often referred to as L1 loss, serves as a metric for quantifying discrepancies between paired observations representing the same phenomenon. These pairs usually involve comparisons between predictions and corresponding observations, temporal disparities between later and earlier time points, or differences between a measurement technique and an alternative method. MAE is calculated by summing the absolute errors and dividing the result by the sample size, as expressed in the following formula:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

It represents the average value of the absolute errors, with each error term, denoted as e_i, signifying the absolute difference between the predicted value y_i and the true value x_i.

Mean Squared Error Mean Squared Error (MSE), also known as L2 Loss, is a statistical measure obtained by squaring the difference between the observed y value and its corresponding predicted value. MSE is calculated using the formula below:

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

Here:

- y_i represents the i^{th} observed value.
- $\hat{y}$ denotes the predicted value associated with y_i.
- n indicates the total number of observations.

The process of calculating MSE is similar to computing variance. To determine MSE, the observed value is subtracted from the expected value, and the resulting difference is squared. This procedure is repeated for all observations. Subsequently, the squared differences are summed, and the resulting sum is divided by the number of observations to obtain MSE.

Root Mean Square Error Root Mean Square Error (RMSE) serves as a metric for assessing the average discrepancy between predicted values generated by a model and the actual values in a given dataset. A lower RMSE value indicates better fitting of the model to the dataset.

The formula for RMSE calculation is as follows:

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

Here:

- y_i denotes the i^{th} observed value.
- $\hat{y}$ represents the corresponding predicted value.
- n signifies the total number of observations.

7.4 Results and Discussion

In this section, the results of the analysis are presented, addressing the research questions posed in Section 7.1.

Google considerations The analysis begins with an examination of the Google asset. Graphs comparing the performance of sentiment scores derived from news (Score Title and Score Text) and Reddit comments are provided.

The trends of both Score Title and Score Text are largely similar, but Score Title exhibits more abrupt changes due to the shorter length of the titles. Averaging the scores tends to smooth out the trends but may not capture nuanced sentiment in longer news articles. An interesting outlier occurred on July 26, 2022, where the trend of the scores deviated. This was caused by a news item with a headline mentioning "rejections," resulting in a negative score. Although the body of the article conveyed a positive sentiment, the headline's influence on the score underscores the importance of considering the entire news content.

Comparing the scores derived from Reddit, it is evident that the trend is much smoother, thanks to the higher volume of Reddit comments available for the Google asset. However, the sentiment in Reddit comments tends to lean more negative compared to news scores, possibly due to the use of stronger negative language in social media. Further filtering of Reddit comments based on news-related keywords could provide more accurate sentiment analysis.

Additionally, the analysis reveals that the news trend often precedes the Reddit trend by a day, suggesting that the release of news articles can impact subsequent social media discussions. To explore the correlation between sentiment trends and market trends, the closing price graph is examined. Notably, there are instances where the trends in news scores and Reddit scores align with the price trend, suggesting a potential correlation.

Amazon considerations The analysis continues with an examination of the Amazon asset. Similar to the previous case, the trends of Score Title and Score Text are comparable, with Score Title exhibiting more pronounced variations. However, in this instance, the trends of the Reddit scores and news scores do not consistently align, highlighting the divergent nature of sentiment expressed in news and Reddit comments for the Amazon asset.

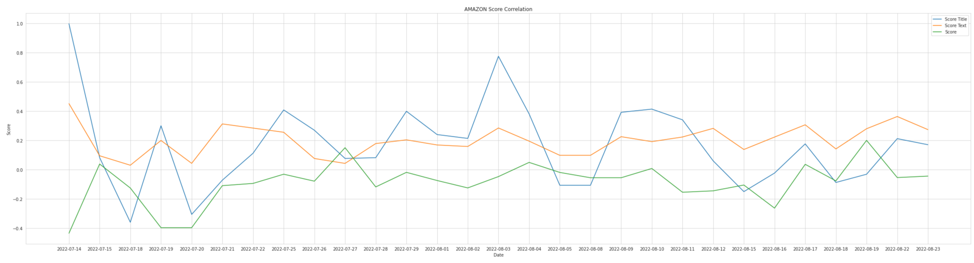


Figure 7.5: Amazon Score

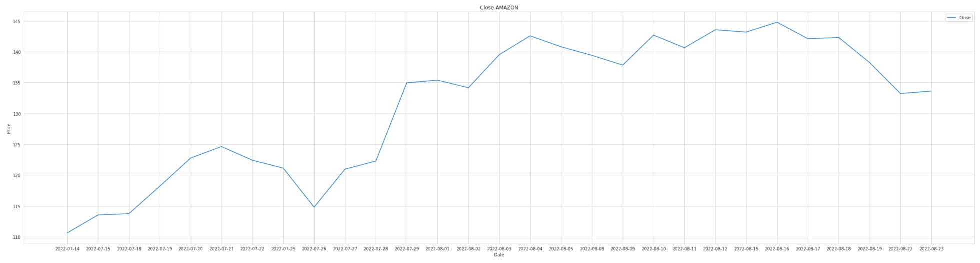


Figure 7.6: Amazon Close

Meta considerations Next, the Meta asset is examined. The trends in Score Title and Score Text are again similar, with Score Title displaying sharper variations. The analysis indicates that the trend of news scores often precedes the Reddit trend by a day, potentially indicating the influence of news articles on subsequent social media discussions.

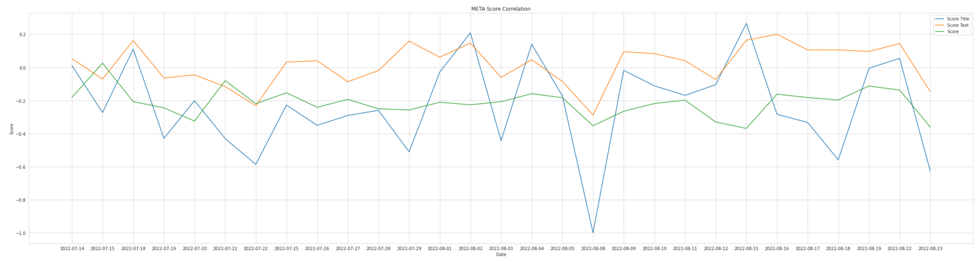


Figure 7.7: Meta Score

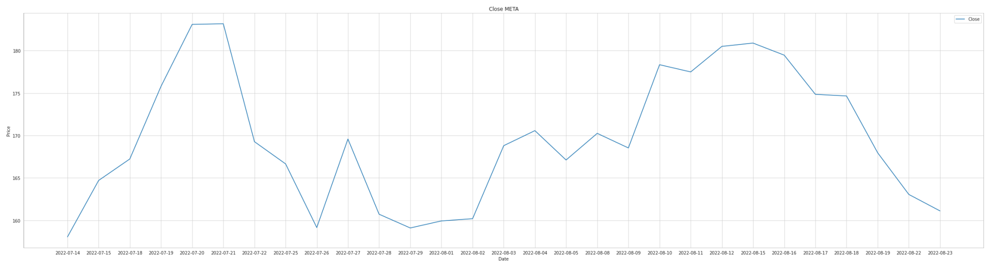


Figure 7.8: Meta Close

Apple considerations The analysis then moves to the Apple asset. The trends in Score Title and Score Text for Apple exhibit comparable patterns, with Score Title showing sharper fluctuations. It is notable that the trend of news scores aligns with the market trend until around August 9. However, after this point, the sentiment scores become negative while the price continues to rise. An interesting date to note is August 22, where all scores show a strong downward trend, coinciding with a decline in the market price.

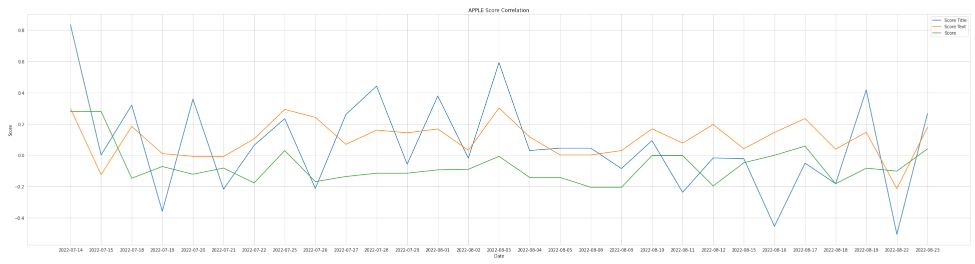


Figure 7.9: Apple Score

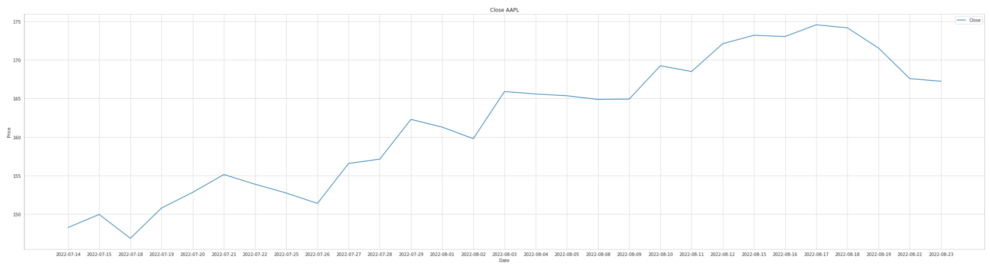


Figure 7.10: Apple Close

Tesla considerations Lastly, the Tesla asset is analyzed. The sentiment scores for Tesla, both in news and Reddit, tend to be more negative compared to the other assets. However, the negative sentiment does not reflect the market performance, as Tesla experiences substantial price fluctuations. It is worth noting that Tesla had the highest price variation among the assets studied, which may have impacted the performance of predictive models.

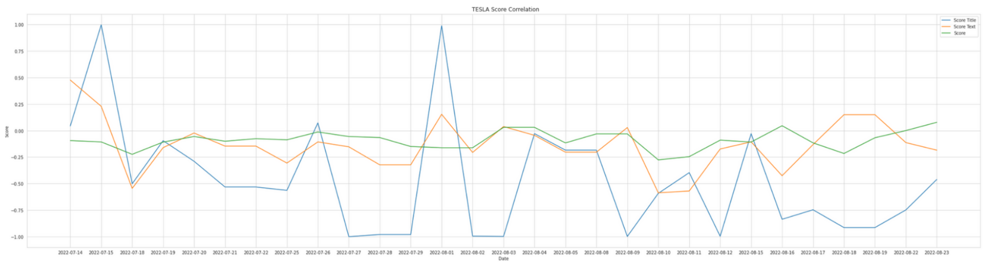


Figure 7.11: Tesla Score

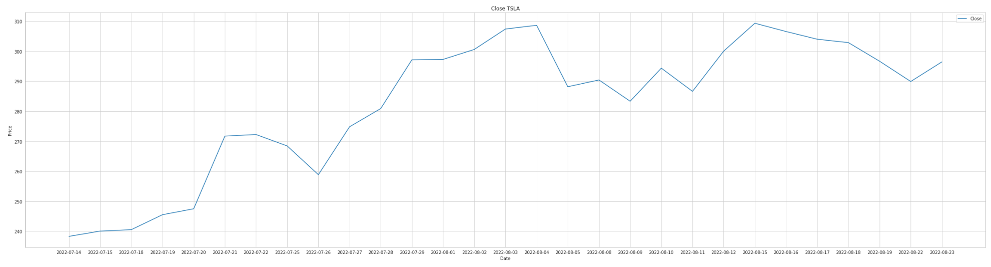


Figure 7.12: Tesla Close

7.4.1 Model Results

In this section, the results obtained from various algorithms are examined to address the remaining research questions. A comparison is made between supervised machine learning approaches (Random Forest, Gradient Boost, ExtraTrees, and Lasso) and deep learning approaches (LSTM, BiLSTM, GRU, and BiGRU).

Stock Price Data

The analysis begins by considering the results obtained solely from the Stock Price Data configuration, without incorporating sentiment from news or Reddit. The Mean Absolute Error (MAE) metric is used for evaluation. The Random Forest Regressor consistently outperforms other models for Google, Apple, and Amazon assets. However, for the Meta and Tesla assets, the GradientBoostingRegressor and ExtraTreesRegressor perform better, respectively. When considering the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) metrics, the deep learning models (LSTM, BiLSTM, GRU, and BiGRU) generally yield better results.

Stock Price Data + Sentiment News

The analysis then proceeds to the results obtained by incorporating sentiment from news in addition to the stock price data. The Random Forest Regressor performs the best across most assets, achieving the lowest MAE values. For MSE and RMSE, LSTM outperforms other models for Apple, Meta, and Tesla, while GRU and BiLSTM emerge as the best solutions for Google and Amazon, respectively.

Stock Price Data + Sentiment Reddit

The next set of results considers the inclusion of sentiment from Reddit comments along with stock price data. The Random Forest Regressor generally performs the best, with the lowest MAE values across most assets. However, when considering MSE and RMSE, no model stands out as the clear winner. Deep learning algorithms generally yield better results across all assets.

Stock Price Data + Sentiment News + Sentiment Reddit

Finally, the analysis encompasses the results obtained by combining sentiment from both news and Reddit comments with stock price data. The Random Forest Regressor and ExtraTreesRegressor exhibit the lowest MAE values for various assets. For MSE and RMSE, LSTM performs the best for Amazon,

Stock Price Dataset					
	MAE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	**2.80 ± 0.001**	**2.03 ± 0.0002**	**2.84 ± 0.004**	6.02 ± 0.004	5.05 ± 0.03
GradientBoostingRegressor	2.88 ± 0.14	2.85 ± 0.18	3.01 ± 0.08	4.88 ± 0.02	**4.16 ±0.19**
ExtraTreesRegressor	3.05 ± 0.004	2.20 ± 0.17	3.092 ± 0.002	5.38 ± 0.006	5.70 ± 0.01
Lasso	3.17 ± 0	2.78 ± 0	4.6 ± 0	5.26 ± 0	8.85 ± 0
LSTM	5.41 ± 10.87	13.22 ± 9.82	16.80 ± 19.80	2.9 ± 1.72	81.81 ± 378.50
BiLSTM	8.30 ± 10.28	15.07 ± 5.47	11.53 ± 0.86	6.67 ± 9.04	34.065432.98
GRU	8.86 ± 11.18	12.8 ± 1.67	7.75 ± 3.90	**1.93 ±3.04**	45.33 ± 1104.21
BIGRU	7.92 ± 6.03	11.60 ± 2.49	7.37 ± 13.05	5.83 ± 7.71	85.71 ± 2762.07
	MSE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	12.07 ± 0.09	6.72 ± 0.008	13.76 ± 0.30	45.25 ± 0.72	30.52 ± 2.34
GradientBoostingRegressor	12.40 ± 7.97	13.01 ± 11.88	15.98 ± 2.87	32.06 ± 4.13	25.75 ± 26.03
ExtraTreesRegressor	12.74 ± 0.17	8.62 ± 0.17	13.97 ± 0.14	43.50 ± 0.15	38.64 ± 0.43
Lasso	12.66 ± 0	10.14 ± 0	22.88 ± 0	36.29 ± 0	146.13 ± 0
LSTM	**2.03 ±0.21**	3.08 ± 0.09	3.35 ± 0.07	1.41 ± 0.17	7.47 ± 1.52
BiLSTM	2.27 ± 0.31	3.17 ± 0.08	2.89 ± 0.86	2.28 ± 0.17	**4.64 ±2.29**
GRU	2.46 ± 0.26	2.91 ± 0.03	2.29 ± 0.06	**1.04 ±0.36**	5.38 ± 4.74
BIGRU	2.24 ± 0.15	**2.73 ±0.03**	**2.25 ±0.31**	2.14 ± 0.42	7.79 ± 8.88
	RMSE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	3.47 ± 0.002	2.59 ± 0.0003	3.71 ± 0.006	6.73 ± 0.004	5.52 ± 0.02
GradientBoostingRegressor	3.50 ± 0.19	3.57 ± 0.22	3.99 ± 0.05	5.66 ± 0.03	5.05 ± 0.25
ExtraTreesRegressor	3.57 ± 0.003	2.93 ± 0.005	3.74 ± 0.002	6.60 ± 0.001	6.22 ± 0.003
Lasso	3.56 ± 0	3.18 ± 0	4.78 ± 0	6.02 ± 0	12.09 ± 0
LSTM	**1.42 ±0.024**	1.75 ± 0.007	1.83 ± 0.006	1.17 ± 0.04	2.72 ± 0.05
BILSTM	1.49 ± 0.04	1.78 ± 0.006	1.70 ± 0.003	1.5 ± 0.02	**2.13 ±0.11**
GRU	1.56 ± 0.03	1.70 ± 0.01	1.51 ± 0.006	**0.98 ±0.09**	2.28 ± 0.19
BIGRU	1.49 ± 0.02	**1.65 ±0.002**	**1.49 ±0.03**	1.45 ± 0.05	2.74 ± 0.27

Table 7.1: Comparison of experimental results, set the dataset (Stock Price Dataset), according to the considered stock and prediction model.

Stock Price & Sentiment News Dataset					
	MAE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	**1.54 ±0.001**	1.99 ± 0.01	**2.98 ±0.001**	**4.71 ±0.008**	**5.55 ±0.04**
GradientBoostingRegressor	1.62 ± 0.03	2.39 ± 0.11	3.72 ± 0.29	5.93 ± 0.88	7.52 ± 1.75
ExtraTreesRegressor	1.79 ± 0.002	**1.65 ±0.001**	3.02 ± 0.01	5.59 ± 0.04	6.22 ± 0.03
Lasso	2.25 ± 0	2.32 ± 0	4.78 ± 0	6.07 ± 0	6.77 ± 0
LSTM	8.32 ± 6.80	14.02 ± 6.92	5.34 ± 8.03	7.20 ± 6.37	52.07 ± 117.32
BILSTM	14.46 ± 26.69	17.65 ± 32.64	3.52 ± 2.22	36.03 ± 121.69	86.94 ± 2223.99
GRU	5.61 ± 0.41	16.14 ± 12.14	8.65 ± 4.00	20.71 ± 25.38	127.66 ± 10100.23
BIGRU	10.61 ± 11.62	17.41 ± 11.07	9.47 ± 16.84	47.31 ± 41.89	159.31 ± 2502.55
	MSE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	3.84 ± 0.01	5.78 ± 0.22	13.14 ± 0.06	43.31 ± 2.60	43.09 ± 5.29
GradientBoostingRegressor	4.17 ± 0.43	8.13 ± 4.97	24.12 ± 64.50	49.18 ± 30.18	82.45 ± 760.01
ExtraTreesRegressor	5.02 ± 0.04	5.03 ± 0.02	16.61 ± 0.10	41.58 ± 4.14	56.026 ± 14.05
Lasso	6.54 ± 0	8.56 ± 0	28.34 ± 0	35.21 ± 0	53.67 ± 0
LSTM	2.36 ± 0.14	**2.96 ±0.05**	1.86 ± 0.31	**2.24 ±0.10**	**6.19 ±0.50**
BILSTM	3.19 ± 0.47	3.39 ± 0.30	**1.53 ±0.16**	5.31 ± 0.94	7.62 ± 7.42
GRU	**2.22 ±0.01**	3.35 ± 0.16	2.25 ± 0.15	4.13 ± 0.32	9.40 ± 17.60
BIGRU	2.95 ± 0.15	3.40 ± 0.06	2.64 ± 0.40	6.27 ± 0.21	11.60 ± 4.04
	RMSE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	**1.96 ±0.001**	2.40 ± 0.01	3.62 ± 0.06	6.58 ± 0.02	6.56 ± 0.03
GradientBoostingRegressor	2.04 ± 0.02	2.82 ± 0.16	4.85 ± 0.62	6.93 ± 1.09	8.96 ± 2.25
ExtraTreesRegressor	2.24 ± 0.002	2.24 ± 0.001	4.07 ± 0.002	6.44 ± 0.03	7.48 ± 0.06
Lasso	2.56 ± 0	2.93 ± 0	5.32 ± 0	5.93 ± 0	7.33 ± 0
LSTM	1.53 ± 0.01	**1.72 ±0.004**	1.35 ± 0.05	**1.49 ±0.01**	**2.48 ±0.02**
BiLSTM	1.78 ± 0.03	1.87 ± 0.02	**1.22 ±0.03**	2.29 ± 0.05	2.71 ± 0.26
GRU	1.49 ± 0.002	1.83 ± 0.01	1.49 ± 0.02	2.03 ± 0.02	3 ± 0.43
BiGRU	1.71 ± 0.01	1.84 ± 0.004	1.61 ± 0.04	2.50 ± 0.008	3.39 ± 0.08

Table 7.2: Comparison of experimental results, set the dataset (Stock Price and Sentiment Datasets), according to the considered stock and prediction model.

Stock Price & Sentiment Reddit Dataset					
	MAE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	**2.23 ±0.002**	**2.23 ±0.008**	**2.93 ±0.007**	5.56 ± 0.007	**5.62 ±0.04**
GradientBoostingRegressor	2.36 ± 0.006	2.41 ± 0.36	3.97 ± 0.50	3.73 ± 0.04	6.10 ± 0.79
ExtraTreesRegressor	2.48 ± 0.0004	2.27 ± 0.006	2.99 ± 0.002	4.71 ± 0.005	5.95 ± 0.11
Lasso	2.66 ± 0	2.73 ± 0	4.6 ± 0	4.75 ± 0	14.88 ± 0
LSTM	8.32 ± 6.82	18.08 ± 52.39	18.17 ± 4.81	7.48 ± 32.27	47.44 ± 51.29
BiLSTM	5.59 ± 6.81	15.70 ± 66.91	14.05 ± 38.71	**1.11 ±0.16**	59.70 ± 98.70
GRU	7.11 ± 3.38	15.70 ± 35.90	13.11 ± 8.59	**1.01 ±0.39**	87.02 ± 1273.53
BiGRU	23.88 ± 194.19	16.83 ± 24.27	5.10 ± 4.87	4.44 ± 8.14	114.13 ± 244.15
	MSE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	7.52 ± 0.09	7.25 ± 0.07	13.46 ± 0.27	36.20 ± 1.21	36.06 ± 10.94
GradientBoostingRegressor	9.23 ± 0.66	9.71 ± 0.57	22.65 ± 46.47	18.26 ± 2.09	56.30 ± 188.99
ExtraTreesRegressor	9.52 ± 0.66	8.11 ± 0.17	13.36 ± 0.20	32.50 ± 1.29	40.78 ± 15.47
Lasso	9.75 ± 0	11.01 ± 0	22.80 ± 0	29.70 ± 0	334.00 ± 0
LSTM	2.36 ± 0.14	3.51 ± 0.07	3.68 ± 0.09	2.30 ± 0.73	**5.65 ±0.10**
BiLSTM	**2.31 ±0.01**	3.42 ± 0.11	3.12 ± 0.53	0.88 ± 0.03	6.28 ± 0.87
GRU	2.33 ± 0.007	3.43 ± 0.12	2.97 ± 0.05	**0.84 ±0.082**	8.38 ± 1.85
BiGRU	4.20 ± 2.61	**3.34 ±0.01**	**1.99 ±0.13**	1.67 ± 0.31	9.32 ± 0.75
	RMSE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	2.74 ± 0.003	2.69 ± 0.002	3.65 ± 0.005	6.02 ± 0.008	6.00 ± 0.08
GradientBoostingRegressor	3.03 ± 0.02	3.11 ± 0.01	4.70 ± 0.54	4.27 ± 0.03	7.45 ± 0.81
ExtraTreesRegressor	3.09 ± 0.0005	2.85 ± 0.006	3.66 ± 0.004	5.70 ± 0.01	6.38 ± 0.10
Lasso	3.12 ± 0	3.32 ± 0	4.78 ± 0	5.45 ± 0	18.28 ± 0
LSTM	1.53 ± 0.01	1.87 ± 0.005	1.92 ± 0.006	1.49 ± 0.08	**2.38 ±0.004**
BiLSTM	**1.52 ±0.001**	1.85 ± 0.009	1.75 ± 0.05	0.93 ± 0.008	2.50 ± 0.04
GRU	1.53 ± 0.0008	1.85 ± 0.009	1.72 ± 0.004	**0.91 ±0.02**	2.89 ± 0.05
BiGRU	2.01 ± 0.18	**1.82 ±0.01**	**1.41 ±0.02**	1.27 ± 0.05	3.05 ± 0.02

Table 7.3: Comparison of experimental results, set the dataset (Stock Price & Sentiment Reddit Dataset), according to the considered stock and prediction model.

Stock Price Data + Sentiment News + Sentiment Reddit					
	MAE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	**1.63 ±0.001**	**1.99 ±0.008**	3.50 ± 0.09	6.10 ± 0.02	**5.58 ±0.08**
GradientBoostingRegressor	1.63 ± 0.004	2.70 ± 0.36	4.58 ± 0.63	4.40 ± 0.16	6.41 ± 0.14
ExtraTreesRegressor	1.74 ± 0.001	2.35 ± 0.006	**3.10 ±0.01**	**3.55 ±0.004**	6.22 ± 0.11
Lasso	2.42 ± 0	2.78 ± 0	4.29 ± 0	3.98 ± 0	17.96 ± 0
LSTM	12.43 ± 50.74	22.56 ± 52.40	7.73 ± 9.48	6.69 ± 4.30	93.94 ± 207.07
BiLSTM	7.73 ± 1.67	29.28 ± 66.92	9.92 ± 10.11	42.20 ± 85.80	96.71 ± 3220.69
GRU	12.61 ± 104.56	27.57 ± 35.90	10.93 ± 0.71	23.72 ± 24.37	89.93 ± 1285.01
BiGRU	8.42 ± 4.03	16.46 ± 24.27	11.01 ± 15.10	23.12 ± 31.90	136.86 ± 2817.44
	MSE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	4.38 ± 0.03	5.21 ± 0.29	18.62 ± 5.94	41.94 ± 3.77	42.68 ± 9.04
GradientBoostingRegressor	4.42 ± 0.12	10.44 ± 16.72	37.08 ± 166.48	22.76 ± 18.70	56.79 ± 41.59
ExtraTreesRegressor	4.77 ± 0.009	8.13 ± 0.13	17.33 ± 1.19	38.17 ± 0.77	54.91 ± 13.59
Lasso	8.78 ± 0	10.71 ± 0	21.89 ± 0	25.09 ± 0	366.92 ± 0
LSTM	2.91 ± 0.83	3.97 ± 0.22	**2.30 ±0.33**	**2.07 ±0.06**	**7.88 ±0.45**
BiLSTM	**2.48 ±0.08**	4.39 ± 0.39	2.62 ± 0.23	6.05 ± 0.63	7.99 ± 6.94
GRU	2.97 ± 1.36	4.49 ± 0.32	2.77 ± 0.01	4.61 ± 0.26	8.14 ± 3.52
BiGRU	2.73 ± 0.11	**3.37 ±0.09**	2.74 ± 0.22	4.48 ± 0.43	10.02 ± 8.72
	RMSE				
	Google	Apple	Amazon	Meta	Tesla
Random Forest Regressor	2.09 ± 0.002	2.28 ± 0.01	4.31 ± 0.08	6.48 ± 0.02	6.53 ± 0.06
GradientBoostingRegressor	2.10 ± 0.007	3.17 ± 0.36	5.99 ± 1.21	4.75 ± 0.19	7.53 ± 0.19
ExtraTreesRegressor	2.19 ± 0.0005	2.85 ± 0.004	4.16 ± 0.02	6.18 ± 0.005	7.41 ± 0.06
Lasso	2.96 ± 0	3.27 ± 0	4.68 ± 0	5.01 ± 0	19.16 ± 0
LSTM	1.68 ± 0.07	1.99 ± 0.01	**1.50 ±0.04**	**1.43 ±0.007**	2.81 ± 0.02
BiLSTM	**1.57 ±0.008**	2.09 ± 0.02	1.61 ± 0.02	2.45 ± 0.03	**2.79 ±0.21**
GRU	1.69 ± 0.10	2.12 ± 0.02	1.66 ± 0.001	4.61 ± 0.01	2.83 ± 0.11
BiGRU	1.65 ± 0.01	**1.83 ±0.007**	1.65 ± 0.02	2.11 ± 0.02	3.12 ± 0.26

Table 7.4: Comparison of experimental results, set the dataset (Stock Price, Sentiment News & Sentiment Reddit Dataset), according to the considered stock and prediction model.

Meta, and Tesla, while BiLSTM and BiGRU emerge as the top solutions for Google and Apple, respectively.

7.4.2 Discussions

In the context of addressing the research questions introduced in 7.1, it can be stated as follows:

- RQ$_1$: In the context of addressing this research question, recent research, as discussed in the Related Work section (see Section 7.2), has indicated that news sentiment and social media comments can exert a substantial influence on stock market trends and serve as valuable predictors of stock price movements [269, 270]. Notably, there exists a discernible correlation between the sentiments expressed on social media and news platforms and the behavior of stock prices. For example, sentiment analysis of social media data has been employed to investigate the impact of news flows on cryptocurrency prices [289]. Consequently, sentiment data derived from social media and news sources can impact investors' decisions to buy or sell stocks, guided by the sentiment associated with such information [290].

The experimental findings further substantiate the notion that incorporating information from news sentiment or Reddit comments can result in substantial enhancements in model performance in certain cases. Interestingly, it is worth noting that combining both news sentiment and Reddit comments rarely leads to superior outcomes.

- RQ_2: Reddit and Business Insider have emerged as prominent data sources in stock prediction research for several reasons. Firstly, Reddit serves as a prominent social media platform with specialized communities (subreddits) dedicated to finance and investing. Researchers have recognized its value in stock prediction due to the abundance of user-generated content related to financial markets, offering real-time discussions, opinions, and sentiments expressed by investors and traders [287].

  Secondly, Reddit discussions contain valuable information reflecting the sentiment of retail investors and the broader market. Analyzing posts and comments related to specific stocks or market events enables researchers to gain insights into market sentiment, potential shifts in investor behavior, and the impact of social media on stock prices [286, 281].

Business Insider, a reputable news website covering financial news, stock market updates, and business-related topics, is another data source frequently utilized by researchers. Its reliable reporting and comprehensive coverage of financial events, along with accessible textual data, make Business Insider articles valuable for incorporation into stock prediction models [279, 278].

Both Reddit and Business Insider offer publicly accessible data, facilitating data collection for research purposes [288, 278]. Additionally, Reddit's community-driven nature allows researchers to tap into the insights shared by passionate and knowledgeable individuals regarding specific stocks, enabling aggregation and anonymized incorporation of such information into prediction models [287].

In conclusion, Reddit and Business Insider are favored data sources in stock prediction research due to their extensive user-generated content, sentiment insights, reputable reporting, and data accessibility. By leveraging the unique characteristics of these sources, researchers can enhance their understanding of market sentiment and improve stock prediction models. However, researchers should exercise caution in interpreting results, taking into account potential limitations and biases associated with these data sources.

- RQ$_3$: In the realm of stock prediction research, Reddit and Business Insider have risen to prominence as significant data sources for several compelling reasons. Firstly, Reddit serves as a prominent social media platform, replete with specialized communities (subreddits) dedicated to finance and investing. Researchers have keenly recognized its value in stock prediction due to the wealth of user-generated content pertaining to financial markets. This content provides real-time discussions, opinions, and sentiments expressed by investors and traders, as documented in [287].

Secondly, Reddit discussions contain a trove of valuable information that reflects the sentiment of retail investors and the broader market. The analysis of posts and comments related to specific stocks or market events empowers researchers to gain valuable insights into market sentiment, potential shifts in investor behavior, and the influence of social media on stock prices, as detailed in studies such as [286, 281].

Furthermore, Business Insider, a reputable news website renowned for its coverage of financial news, stock market updates, and business-related topics, represents another data source frequently harnessed by researchers. Its dependable reporting and comprehensive coverage of financial events, coupled with accessible textual data, render Business Insider articles highly valuable for integration into stock prediction models, as evidenced by references such as [279, 278].

Both Reddit and Business Insider offer publicly accessible data, thereby simplifying the data collection process for research endeavors [288, 278]. Additionally, Reddit's community-driven nature empowers researchers to tap into the insights shared by impassioned and knowledgeable individuals with respect to specific stocks. This enables the aggregation and anonymized incorporation of such information into prediction models, as elucidated in [287].

In summary, Reddit and Business Insider occupy a favored position as data sources in stock prediction research due to their extensive reservoirs of user-generated content, which offer valuable insights into sentiment, their credible reporting, and the ease of data access. By harnessing the distinctive attributes of these sources, researchers can enrich their comprehension of market sentiment and enhance the accuracy of their stock prediction models. Nevertheless, it is essential for researchers to exercise prudence in the interpretation of results, while remaining mindful of potential limitations and biases associated with these data sources.

- RQ$_4$: The comprehensive review of existing literature has firmly established that the analysis of diverse data sources, which includes news articles and social media platforms, has been effectively addressed through the utilization of Machine Learning (ML) techniques. Deep Learning (DL) methods have received particular emphasis and have demonstrated their capability in achieving highly accurate predictions in the realm of stock market forecasting, as corroborated by [271]. This superiority in predictive performance becomes evident when contrasting ML and DL models with traditional statistical approaches, as corroborated by references such as [272, 273, 291].

  Furthermore, the experiments conducted within the confines of this study have yielded significant insights into the consequences of incorporating information related to news sentiment or comments from Reddit into prediction models. Notably, the augmentation of such information has translated into substantial performance improvements in specific instances, as vividly illustrated by the enhancements observed in the case of the Google asset. Nevertheless, the outcomes have taken a divergent trajectory for other assets, such as Tesla, where the inclusion of sentiment or Reddit data has led to diminished performance. Of particular interest is the revelation that the concurrent incorporation of both sentiment and Reddit data has seldom resulted in superior outcomes.

- RQ$_5$: In the pursuit of their objectives, extensive exploration has been conducted into the domain of Machine Learning (ML) and Deep Learning (DL) techniques. Nevertheless, an intriguing observation has surfaced regarding the consistent underperformance of the Lasso algorithm when juxtaposed with other models, regardless of the asset or metric utilized. This observation underscores the inherent limitations of linear regression-based algorithms when confronted with time prediction problems, which frequently encompass intricate nonlinear relationships.

The judicious selection of the most proficient ML/DL model hinges upon the specific metric of interest and the degree of susceptibility to outlier errors. When considering the Mean Absolute Error (MAE), it becomes apparent that machine learning models generally outshine their counterparts. Conversely, when assessing the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), the supremacy of deep learning models becomes evident.

In the realm of predictive model accuracy evaluation, deep learning algorithms, with a particular spotlight on the Long Short-Term Memory (LSTM), have earned recognition as more fitting choices. LSTM, being a variant of recurrent neural network, has distinguished itself by excelling in the encapsulation of prolonged dependencies and sequential patterns, rendering it especially advantageous for temporal predictions.

Nevertheless, in scenarios where practical application centers around trading, with an uncompromising emphasis on the evasion of outlier errors, it becomes imperative to accord greater significance to MAE. In such circumstances, machine learning models emerge as the more prudent alternatives, largely attributable to their heightened capacity for managing outliers with efficacy.

To summarize succinctly, the optimal choice between ML and DL models is contingent upon the metric of interest and the specific context of its application. Machine learning models, primarily celebrated for their commendable MAE performance, reign supreme, whereas deep learning models, particularly the LSTM, shine when pinpoint accuracy assessments are the mandate. However, discretion is urged when deploying deep learning models in settings where the mitigation of outlier errors takes precedence.

- RQ_6: In the context of tasks related to stock prediction, which encompass the consideration of diverse data sources, the appropriateness of performance metrics for selecting the most suitable model can hinge on the specific data type being integrated. To elaborate, the consequences of amalgamating sentiment scores derived from sources such as news and Reddit display significant fluctuations, contingent upon the particular asset and metric employed for evaluation.

It is discernible that there is no consistent, uniform pattern of performance enhancement observed across various assets when incorporating sentiment scores. The outcomes diverge depending on the specific asset in question. For instance, when dealing with assets like Google and Apple, the inclusion of sentiment data extracted from news sources translates into improved performance. Conversely, for assets such as Meta, superior results are attained through the utilization of sentiment sourced from Reddit comments. However, the amalgamation of sentiment data for assets like Amazon and Tesla does not yield substantial performance improvements.

It is crucial to underscore that a combined approach, wherein sentiment scores from both news and Reddit sources are integrated, fails to emerge as the optimal solution for any of the assets subjected to analysis. Consequently, it is imperative to acknowledge that the impact of incorporating sentiment data is contingent upon the specific asset and does not uniformly influence performance outcomes.

- RQ_7: The research question aims to identify a group of stocks in which the correlation between market data and social/news data is particularly pronounced and assess whether this data can serve as a suitable testing ground for the selected models.

  In tackling this inquiry, it is essential to recognize that algorithm performance can exhibit significant variations across different assets. For Mean Absolute Error (MAE), the Random Forest Regressor consistently demonstrates superior overall performance across most assets. In contrast, when assessing Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), the Long Short-Term Memory (LSTM) model consistently achieves the lowest values for specific stocks such as Apple, Meta, and Tesla. However, for Google and Amazon stocks, the Bi-directional LSTM (BiLSTM) and Bi-directional Gated Recurrent Unit (BiGRU) models respectively yield better results.

Nonetheless, it is imperative to highlight that there is no single algorithm that consistently outperforms others across all assets. Performance disparities arise based on the particular stock being analyzed. Nevertheless, the Random Forest Regressor (for MAE) and LSTM (for MSE and RMSE) exhibit promising performance across multiple assets.

The correlation between market data and social/news data appears more evident in specific stock groupings, especially when employing the previously mentioned models. Consequently, employing such data as a testing ground for the chosen models has the potential to yield valuable insights into the relationship between market trends and social/news sentiments for particular assets. Nevertheless, further investigation is required to determine the extent of the models' suitability and their applicability across diverse stock groupings and market conditions.

In summary, when it comes to selecting the best model for stock prediction tasks that integrate various data sources, it is crucial to factor in the asset-specific influence of sentiment data when deciding on performance metrics. The choice of metrics should align with the specific asset under scrutiny and the type of sentiment data being integrated to ensure more precise and effective stock predictions.

Furthermore, the findings emphasize that algorithm choice, the incorporation of sentiment data, and the unique characteristics of assets all play substantial roles in determining performance outcomes. While the Random Forest Regressor, LSTM, and other deep learning models exhibit potential

	MAE			
	Stock Price	Stock Price and News	Stock Price and Reddit	Stock Price, News and Reddit
Google	2.80 $\pm$ 0.001	**1.54 $\pm$0.001**	2.23 $\pm$ 0.002	1.63 $\pm$ 0.001
Apple	2.03 $\pm$ 0.0002	**1.65 $\pm$0.001**	2.23 $\pm$ 0.008	1.99 $\pm$ 0.008
Amazon	**2.84 $\pm$0.004**	2.98 $\pm$ 0.001	2.93 $\pm$ 0.007	3.10 $\pm$ 0.01
Meta	2.9 $\pm$ 1.72	4.71 $\pm$ 0.008	**1.11 $\pm$0.16**	3.55 $\pm$ 0.004
Tesla	**4.16 $\pm$0.19**	5.55 $\pm$ 0.04	5.62 $\pm$ 0.04	5.58 $\pm$ 0.08
	MSE			
	Stock Price Data	Stock Price Data + News	Stock Price Data + Reddit	Stock Price Data + News + Reddit
Google	**2.03 $\pm$0.21**	2.22 $\pm$ 0.01	2.31 $\pm$ 0.01	2.48 $\pm$ 0.08
Apple	**2.73 $\pm$0.03**	2.96 $\pm$ 0.05	3.34 $\pm$ 0.01	3.37 $\pm$ 0.09
Amazon	2.29 $\pm$ 0.06	**1.53 $\pm$0.16**	1.99 $\pm$ 0.13	2.30 $\pm$ 0.33
Meta	1.04 $\pm$ 0.36	2.24 $\pm$ 0.10	**0.88 $\pm$0.03**	2.07 $\pm$ 0.06
Tesla	**4.64 $\pm$2.29**	6.19 $\pm$ 0.50	5.65 $\pm$ 0.10	7.88 $\pm$ 0.45
	RMSE			
	Stock Price Data	Stock Price Data + News	Stock Price Data + Reddit	Stock Price Data + News + Reddit
Google	**1.42 $\pm$0.024**	1.49 $\pm$ 0.002	1.52 $\pm$ 0.001	1.57 $\pm$ 0.008
Apple	**1.70 $\pm$0.01**	1.72 $\pm$ 0.004	1.82 $\pm$ 0.01	1.83 $\pm$ 0.007
Amazon	1.51 $\pm$ 0.006	**1.22 $\pm$0.03**	1.41 $\pm$ 0.02	1.50 $\pm$ 0.04
Meta	0.98 $\pm$ 0.09	1.49 $\pm$ 0.01	**0.93 $\pm$0.008**	2.11 $\pm$ 0.02
Tesla	**2.13 $\pm$0.11**	2.48 $\pm$ 0.02	2.38 $\pm$ 0.004	2.79 $\pm$ 0.21

Table 7.5: Summary of the top results achieved. varying the Dataset and stock considered.

for accurate predictions, careful deliberation is necessary when choosing the most appropriate approach for each asset and evaluation metric.

Now, let's provide an overview of the additional considerations regarding the obtained results. Firstly, it's important to highlight that the Google asset showed the highest level of similarity between the score chart and the closing price. Interestingly, it demonstrated improved performance when additional information was incorporated. In contrast, the stocks of Tesla and Amazon exhibited a different behavior, with their performance declining when supplementary data was included.

Another significant factor that deserves attention is the price variation within the evaluated time frame. As shown in Table 7.7 below, Tesla had the greatest price fluctuation, approximately twice as much as the other assets. Amazon also displayed a higher variation compared to the remaining assets. This aspect could have had an impact on the performance of the predictive

	MAE			
	Stock Price Data	Stock Price Data + News	Stock Price Data + Reddit	Stock Price Data + News + Reddit
Google	Random Forest Regressor	Random Forest Regressor	Random Forest Regressor	Random Forest Regressor
Apple	Random Forest Regressor	ExtraTreesRegressor	Random Forest Regressor	Random Forest Regressor
Amazon	Random Forest Regressor	Random Forest Regressor	Random Forest Regressor	ExtraTreesRegressor
Meta	LSTM	Random Forest Regressor	BiLSTM	ExtraTreesRegressor
Tesla	GradientBoostingRegressor	Random Forest Regressor	Random Forest Regressor	Random Forest Regressor
	MSE			
	Stock Price Data	Stock Price Data + News	Stock Price Data + Reddit	Stock Price Data + News + Reddit
Google	LSTM	GRU	BILSTM	BILSTM
Apple	BiGRU	LSTM	BiGRU	BiGRU
Amazon	GRU	BiLSTM	BiGRU	LSTM
Meta	GRU	LSTM	BiLSTM	LSTM
Tesla	BiLSTM	LSTM	LSTM	LSTM
	RMSE			
	Stock Price Data	Stock Price Data + News	Stock Price Data + Reddit	Stock Price Data + News + Reddit
Google	LSTM	GRU	BiLSTM	BiLSTM
Apple	GRU	LSTM	BiGRU	BiGRU
Amazon	GRU	LSTM	BiGRU	LSTM
Meta	GRU	LSTM	BiLSTM	LSTM
Tesla	BiLSTM	LSTM	LSTM	LSTM

Table 7.6: Summary of the top models in terms of efficacy, varying the Dataset and stock considered.

Close			
	Min	**Max**	**Max-Min**
Google	150.02	122.08	17.06
Apple	146.87	174.55	27.68
Amazon	110.63	144.78	34.15
Meta	158.05	183.17	25.12
Tesla	238.31	309.32	**71.01**

Table 7.7: Stock Price Table.

models. Notably, these two cases showed higher variance in the predictive models.

An intriguing phenomenon is observed in the case of the Meta asset. Specifically, it stands out as the only case where the evaluation of Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) consistently favored Deep Learning models as the optimal solutions. Furthermore, when examining the price trend graph, it is evident that Meta alone displayed a "repeating" pattern, while the other assets exhibited only an increasing pattern. This suggests a potential opportunity to identify repeating patterns, and it is plausible that the LSTM or BiLSTM model contributed to superior performance in this scenario. The image below illustrates the best-case scenario, where the Meta asset in the "Stock Price Data + Reddit configuration" outperformed other instances.

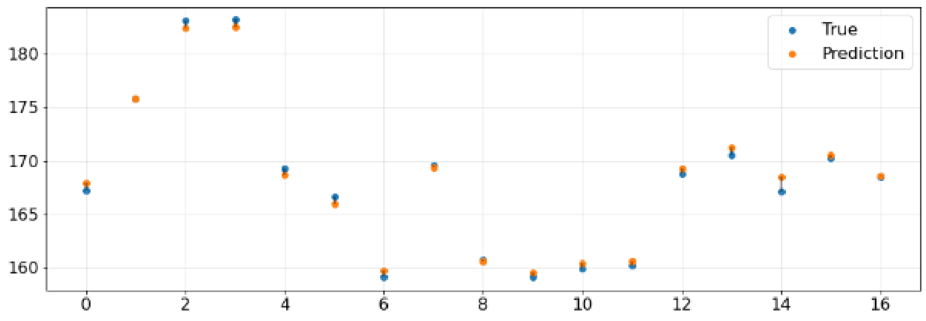


Figure 7.13: Meta Best Model

7.5 Conclusions and future work

This chapter concludes by providing an overview of the significant role that artificial intelligence has played in the financial sector, with a particular focus on a case study involving Stock Price Prediction. The study investigates the potential enhancements that web-extracted information can offer to existing predictive models found in the literature.

One noteworthy discovery is that the analysis of news from newspapers necessitates a deeper examination of article texts in addition to headlines to obtain more reliable sentiment values. Similarly, information extracted from social networks, especially Reddit comments, presents challenges due to the vast amount of data and diverse language usage, which complicates sentiment analysis.

Furthermore, the study underscores the substantial impact of metric choice on model performance. Machine Learning algorithms excel in achieving lower Mean Absolute Error (MAE), while Deep Learning models prove more accurate when optimizing for metrics like Mean Squared Error (MSE) and Root Mean Squared Error (RMSE).

The study also highlights the potential advantages of incorporating news sentiment or Reddit comments into predictive models. While the addition of such information significantly enhances model performance for assets like Google, it can lead to deteriorations in performance for assets like Tesla. Moreover, combining both types of information rarely results in improved outcomes.

To enhance this study, future research could expand the dataset by considering a broader range of dates. Automating the process of news scraping and labeling using scripts would facilitate this expansion, especially for assets within the S&P 500. Additionally, exploring the impact of news sentiment on equities within specific sectors, as categorized by Business Insider, could provide valuable insights into the relationship between information and price trends.

To further refine the analysis of Reddit comments, applying additional filtering based on keywords extracted from Named Entity Recognition (NER) in news articles could strengthen the correlation between news and social media sentiments. Moreover, manually labeling some Reddit comments and training a specialized model for sentiment assignment could enhance the accuracy of sentiment analysis.

Lastly, optimizing the neural networks utilized in the study holds promise for achieving even better performance results. Fine-tuning the configurations of these models could lead to significant improvements in predictive accuracy and overall effectiveness.

Discussions and Open Issues

In conclusion, the presented doctoral dissertation has introduced a comprehensive and adaptable framework for time series forecasting, addressing the ever-evolving challenges across various domains. This framework stands out due to its versatility, modularity, and customization capabilities, making it a valuable asset for both researchers and practitioners.

The primary goal of this framework is to offer a unified solution for time series forecasting, transcending the limitations of specialized methodologies. It achieves this by allowing users to tailor the processing pipeline to their specific forecasting needs. The modular design of the framework empowers users to selectively activate or deactivate components, including pre-processing, feature engineering, data augmentation, and encoding strategies. This adaptability caters to both novice users seeking simplicity and experts seeking advanced customization.

The framework's flexibility extends to the choice of coding techniques and deep learning models, providing a wide range of options to align with data characteristics. It goes beyond static pipelines, introducing adaptive module activation to address the unique challenges of each forecasting task dynamically.

In the context of *Industry 4.0*, the framework has been successfully applied to predictive maintenance tasks (Chapter 4). We explored various time series encoding techniques combined with Convolutional Neural Networks (CNNs) and evaluated their performance on real-world datasets, such as the *PAKDD2020 Alibaba AI Ops Competition* and *NASA Bearing* datasets. While the inclusion of Generative Adversarial Networks (*GAN*s) showed promise, it came at the cost of increased computational resources. In brief, the proposed approach achieves results comparable or superior to the state of the art for both datasets, with a particular advantage in terms of efficiency.

In another Industry 4.0 application, a light attention-based model was introduced in Chapter 5, demonstrating efficiency and effectiveness in handling time-series data with fewer parameters and faster training times compared to LSTM models. This trade-off between efficiency and accuracy is essential in industrial contexts.

In the realm of *Fintech*, a novel framework integrating historical stock data with sentiment scores from social networks was proposed (Chapter 6). The analysis revealed the potential impact of user-generated content on stock prediction, though the dynamic nature of social network discussions posed challenges. However, there is to be observed from the experimental results, that although social data has an impact on stock prediction, it is not always beneficial, even when the information content is significant. The final case study presented in Chapter 7 emphasized the role of AI in financial domains, specifically in stock price prediction. It highlighted the importance of considering both headlines and article texts for news sentiment analysis and the varying performance of machine learning and deep learning models based on different metrics. Incorporating web-extracted information from news and social networks into predictive models showed promising results, with significant enhancements for some assets. However, it also emphasized the need for careful consideration of data sources and metrics for different use cases. In summary, this doctoral dissertation has contributed a versatile framework for time series forecasting, demonstrating its applicability in Industry 4.0 and Fintech scenarios. The modularity and adaptability of the framework make it a valuable tool for addressing diverse forecasting challenges across various domains, ushering in a new era of forecasting possibilities.

In terms of open issues, the framework will be extended to address some of the limitations highlighted by the research findings. One avenue of exploration will involve integrating encoding techniques with tiled Convolutional Neural Networks (CNNs), as documented in the work by [292], which have demonstrated superior computational efficiency compared to standard CNNs. Additionally, there will be a heightened focus on enhancing the capabilities of GANs to further improve their effectiveness in the framework.

Furthermore, future research endeavors will investigate the potential advantages of adopting ensemble models in conjunction with CNNs, potentially leading to enhanced classification performance. Alongside this, there will be a concerted effort to explore *eXplainable Artificial Intelligence* (XAI) approaches, aimed at providing explanations for mis-classifications, thus aiding practitioners in their decision-making processes.

In the context of predictive maintenance applications, upcoming work will delve deeper into the application of the attention mechanism across various scenarios. Additionally, efforts will be made to interpret what the model learns, specifically identifying areas that garner greater attention, in order to facilitate the use of XAI tools.

Regarding the Fintech studies, future research will encompass a broader array of stocks and incorporate multimedia social networks such as Instagram and Facebook. Again, this expansion will also involve the integration of XAI techniques to support practitioners in making informed decisions.

Instead, forthcoming research will consider the inclusion of a more extensive dataset, including data spanning a wider range of dates. The process of news scraping and labeling will be automated using scripts, particularly for assets within the *S&P 500*. Additionally, exploring the impact of news sentiment on equities within specific sectors, as categorized by *Business Insider*, will provide valuable insights into the relationship between information dissemination and price trends.

To refine the analysis of *Reddit* comments, future work will involve additional filtering based on keywords extracted through *Named Entity Recognition* (NER) in news articles. Furthermore, a manual labeling process for select Reddit comments and the training of specialized models for sentiment assignment will be pursued to enhance the accuracy of sentiment analysis.

Lastly, optimization of the neural networks employed in the studies will be a priority, with fine-tuning of model configurations expected to yield significant improvements in predictive accuracy and overall effectiveness.

Bibliography

[1] James D Hamilton. *Time series analysis*. Princeton university press, 2020.

[2] Ratnadip Adhikari and Ramesh K Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*, 2013.

[3] Ming C Hao, Umeshwar Dayal, Daniel A Keim, and Tobias Schreck. Importance-driven visualization layouts for large time series data. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 203–210. IEEE, 2005.

[4] Sheikh Mohammad Idrees, M Afshar Alam, and Parul Agarwal. A prediction approach for stock market volatility based on time series data. *IEEE Access*, 7:17287–17298, 2019.

[5] Yakov Amihud. Illiquidity and stock returns: cross-section and time-series effects. *Journal of financial markets*, 5(1):31–56, 2002.

[6] Liang Lu, Hualiang Lin, Linwei Tian, Weizhong Yang, Jimin Sun, and Qiyong Liu. Time series analysis of dengue fever and weather in guangzhou, china. *BMC Public Health*, 9:1–5, 2009.

[7] Zahra Karevan and Johan AK Suykens. Transductive lstm for time-series prediction: An application to weather forecasting. *Neural Networks*, 125:1–9, 2020.

[8] José F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. Deep learning for time series forecasting: a survey. *Big Data*, 9(1):3–21, 2021.

[9] Changqing Cheng, Akkarapol Sa-Ngasoongsong, Omer Beyca, Trung Le, Hui Yang, Zhenyu Kong, and Satish TS Bukkapatnam. Time series forecasting for nonlinear and non-stationary processes: a review and comparative study. *Iie Transactions*, 47(10):1053–1071, 2015.

[10] Lucie Michel and David Makowski. Comparison of statistical models for analyzing wheat yield time series. *PLoS One*, 8(10):e78615, 2013.

[11] Kunhui Lin, Qiang Lin, Changle Zhou, and Junfeng Yao. Time series prediction based on linear regression and svr. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 1, pages 688–691. IEEE, 2007.

[12] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[13] Balpreet Singh, Pawan Kumar, Nonita Sharma, and KP Sharma. Sales forecast for amazon sales with time series modeling. In *2020 first international conference on power, control and computing technologies (ICPC2T)*, pages 38–43. IEEE, 2020.

[14] John H Cochrane. Time series for macroeconomics and finance, 1997.

[15] Thanapant Raicharoen, Chidchanok Lursinsap, and Paron Sanguanbhokai. Application of critical support vector machine to time series prediction. In *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS'03.*, volume 5, pages V–V. IEEE, 2003.

[16] IA Iwok and AS Okpe. A comparative study between univariate and multivariate linear stationary time series models. *American Journal of Mathematics and Statistics*, 6(5):203–212, 2016.

[17] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. Springer, 2002.

[18] G Peter Zhang and Min Qi. Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160(2):501–514, 2005.

[19] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[20] Theodore W Anderson. *The statistical analysis of time series*. John Wiley & Sons, 2011.

[21] Roberto Perrelli. Introduction to arch & garch models. *University of illinois optional TA handout*, pages 1–7, 2001.

[22] J Kihoro, Romanus Odhiambo Otieno, and C Wafula. Seasonal time series forecasting: A comparative study of arima and ann models. 2004.

[23] Joarder Kamruzzaman, Rezaul Begg, and Ruhul Sarker. *Artificial neural networks in finance and manufacturing*. IGI Global, 2006.

[24] Kasun Amarasinghe, Daniel L Marino, and Milos Manic. Deep neural networks for energy load forecasting. In *2017 IEEE 26th international symposium on industrial electronics (ISIE)*, pages 1483–1488. IEEE, 2017.

[25] Poonam Sharma and Akansha Singh. Era of deep neural networks: A review. In *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–5. IEEE, 2017.

[26] Guoqiang Zhang, B Eddy Patuwo, and Michael Y Hu. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62, 1998.

[27] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[28] Sho Sonoda and Noboru Murata. Neural network with unbounded activation functions is universal approximator. *Applied and Computational Harmonic Analysis*, 43(2):233–268, 2017.

[29] Marek Śmieja, Łukasz Struski, Jacek Tabor, Bartosz Zieliński, and Przemysław Spurek. Processing of missing data by neural networks. *Advances in neural information processing systems*, 31, 2018.

[30] Vincent Le Guen and Nicolas Thome. Deep time series forecasting with shape and temporal criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):342–355, 2022.

[31] Yeming Gong and René BM De Koster. A review on stochastic models and analysis of warehouse operations. *Logistics Research*, 3:191–205, 2011.

[32] Hui Liu, Chao Chen, Xinwei Lv, Xing Wu, and Min Liu. Deterministic wind energy forecasting: A review of intelligent predictors and auxiliary methods. *Energy Conversion and Management*, 195:328–345, 2019.

[33] Apeksha Shewalkar, Deepika Nyavanandi, and Simone A Ludwig. Performance evaluation of deep neural networks applied to speech recognition: Rnn, lstm and gru. *Journal of Artificial Intelligence and Soft Computing Research*, 9(4):235–245, 2019.

[34] Rui Fu, Zuo Zhang, and Li Li. Using lstm and gru neural network methods for traffic flow prediction. In *2016 31st Youth academic annual conference of Chinese association of automation (YAC)*, pages 324–328. IEEE, 2016.

[35] Jianqiong Xiao and Zhiyong Zhou. Research progress of rnn language model. In *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, pages 1285–1288. IEEE, 2020.

[36] George Saon, Zoltán Tüske, Daniel Bolanos, and Brian Kingsbury. Advancing rnn transducer technology for speech recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5654–5658. IEEE, 2021.

[37] Alper Tokgöz and Gözde Ünal. A rnn based time series approach for forecasting turkish electricity load. In *2018 26th Signal processing and communications applications conference (SIU)*, pages 1–4. IEEE, 2018.

[38] Danilo Mandic and Jonathon Chambers. *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley, 2001.

[39] Alaa Sagheer and Mostafa Kotb. Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, 323:203–213, 2019.

[40] Ray J Frank, Neil Davey, and Stephen P Hunt. Time series prediction and neural networks. *Journal of intelligent and robotic systems*, 31:91–103, 2001.

[41] Sui-Lau Ho, Min Xie, and Thong Ngee Goh. A comparative study of neural network and box-jenkins arima modeling in time series prediction. *Computers & Industrial Engineering*, 42(2-4):371–375, 2002.

[42] Tian Guo, Zhao Xu, Xin Yao, Haifeng Chen, Karl Aberer, and Koichi Funaya. Robust online time series prediction with recurrent neural networks. In *2016 IEEE international conference on data science and advanced analytics (DSAA)*, pages 816–825. Ieee, 2016.

[43] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[44] Ashu Jain and Avadhnam Madhav Kumar. Hybrid neural network models for hydrologic time series forecasting. *Applied Soft Computing*, 7(2):585–592, 2007.

[45] Durdu Ömer Faruk. A hybrid neural network and arima model for water quality time series prediction. *Engineering applications of artificial intelligence*, 23(4):586–594, 2010.

[46] Hossein Abbasimehr and Reza Paki. Improving time series forecasting using lstm and attention models. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–19, 2022.

[47] Ahmed Tealab. Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, 3(2):334–340, 2018.

[48] Eduardo Rocha Rodrigues, Igor Oliveira, Renato Cunha, and Marco Netto. Deepdownscale: A deep learning strategy for high-resolution weather forecast. In *2018 IEEE 14th International Conference on e-Science (e-Science)*, pages 415–422. IEEE, 2018.

[49] Aditya Grover, Ashish Kapoor, and Eric Horvitz. A deep hybrid model for weather forecasting. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 379–386, 2015.

[50] Ryo Akita, Akira Yoshihara, Takashi Matsubara, and Kuniaki Uehara. Deep learning for stock prediction using numerical and textual information. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6. IEEE, 2016.

[51] Mojtaba Nabipour, Pooyan Nayyeri, Hamed Jabani, Amir Mosavi, and Ely Salwana. Deep learning for stock market prediction. *Entropy*, 22(8):840, 2020.

[52] Oscar Serradilla, Ekhi Zugasti, Jon Rodriguez, and Urko Zurutuza. Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects. *Applied Intelligence*, 52(10):10934–10964, 2022.

[53] Youdao Wang, Yifan Zhao, and Sri Addepalli. Remaining useful life prediction using deep learning approaches: A review. *Procedia manufacturing*, 49:81–88, 2020.

[54] Francesco Piccialli, Fabio Giampaolo, Edoardo Prezioso, David Camacho, and Giovanni Acampora. Artificial intelligence and healthcare: Forecasting of medical bookings through multi-source time-series fusion. *Information Fusion*, 74:1–16, 2021.

[55] Abdelhafid Zeroual, Fouzi Harrou, Abdelkader Dairi, and Ying Sun. Deep learning methods for forecasting covid-19 time-series data: A comparative study. *Chaos, solitons & fractals*, 140:110121, 2020.

[56] Aniello De Santo, Antonio Galli, Michela Gravina, Vincenzo Moscato, and Giancarlo Sperli. Deep learning for hdd health assessment: an application based on lstm. *IEEE Transactions on Computers*, pages 1–1, 2020.

[57] Diego GS Pivoto, Luiz FF de Almeida, Rodrigo da Rosa Righi, Joel JPC Rodrigues, Alexandre Baratella Lugli, and Antonio M Alberti. Cyber-physical systems architectures for industrial internet of things applications in industry 4.0: A literature review. *Journal of manufacturing systems*, 58:176–192, 2021.

[58] Michael Sony and Subhash Naik. Industry 4.0 integration with socio-technical systems theory: A systematic review and proposed theoretical model. *Technology in society*, 61:101248, 2020.

[59] Antonino Ferraro, Antonio Galli, Vincenzo Moscato, and Giancarlo Sperlí. A novel approach for predictive maintenance combining gaf encoding strategies and deep networks. In *2020 IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application (DependSys)*, pages 127–132, 2020.

[60] R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002.

[61] Sule Selcuk. Predictive maintenance, its implementation and latest trends. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 231(9):1670–1679, 2017.

[62] Chanhee Park and Seoung Bum Kim. Virtual metrology modeling of time-dependent spectroscopic signals by a fused lasso algorithm. *Journal of Process Control*, 42:51–58, 2016.

[63] Adriaan Van Horenbeek and Liliane Pintelon. A dynamic predictive maintenance policy for complex multi-component systems. *Reliability engineering & system safety*, 120:39–50, 2013.

[64] Gian Antonio Susto and Alessandro Beghi. Dealing with time-series data in predictive maintenance problems. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4. IEEE, 2016.

[65] Ruben Sipos, Dmitriy Fradkin, Fabian Moerchen, and Zhuang Wang. Log-based predictive maintenance. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1867–1876, 2014.

[66] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):1–34, 2012.

[67] Alessandra Tanda and Cristiana-Maria Schena. *FinTech, BigTech and banks: Digitalisation and its impact on banking business models*. Springer, 2019.

[68] Thomas Puschmann and Rainer Alt. Sharing economy. *Business & Information Systems Engineering*, 58:93–99, 2016.

[69] Mariam H Ismail, Mohamed Khater, and Mohamed Zaki. Digital business transformation and strategy: What do we know so far. *Cambridge Service Alliance*, 10(1):1–35, 2017.

[70] Rainer Lenz. Peer-to-peer lending: Opportunities and risks. *European Journal of Risk Regulation*, 7(4):688–700, 2016.

[71] Rebecca Nüesch, Rainer Alt, and Thomas Puschmann. Hybrid customer interaction. *Business & Information Systems Engineering*, 57:73–78, 2015.

[72] Alessandra Tanda, Cristiana-Maria Schena, Alessandra Tanda, and Cristiana-Maria Schena. Bank strategies in the light of the digitalisation of financial activities. *FinTech, BigTech and Banks: Digitalisation and its Impact on Banking Business Models*, pages 51–81, 2019.

[73] Ian Pollari. The rise of fintech opportunities and challenges. *Jassa*, (3):15–21, 2016.

[74] Thomas Puschmann. Fintech. *Business & Information Systems Engineering*, 59:69–76, 2017.

[75] Peter Tufano. Financial innovation. *Handbook of the Economics of Finance*, 1:307–335, 2003.

[76] Christian Haddad and Lars Hornuf. The emergence of the global fintech market: Economic and technological determinants. *Small business economics*, 53(1):81–105, 2019.

[77] Mark A Chen, Qinxi Wu, and Baozhong Yang. How valuable is fintech innovation? *The Review of Financial Studies*, 32(5):2062–2106, 2019.

[78] SM Raju and Ali Mohammad Tarif. Real-time prediction of bitcoin price using machine learning techniques and public sentiment analysis. *arXiv preprint arXiv:2006.14473*, 2020.

[79] Isaac Kofi Nti, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. A novel multi-source information-fusion predictive framework based on deep neural networks for accuracy enhancement in stock market prediction. *Journal of Big data*, 8(1):1–28, 2021.

[80] Manas Ranjan Senapati, Sumanjit Das, and Sarojananda Mishra. A novel model for stock price prediction using hybrid neural network. *Journal of the Institution of Engineers (india): Series B*, 99:555–563, 2018.

[81] Thien Hai Nguyen and Kiyoaki Shirai. Topic modeling based sentiment analysis on social media for stock market prediction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1354–1364, 2015.

[82] Tejas Mankar, Tushar Hotchandani, Manish Madhwani, Akshay Chidrawar, and CS Lifna. Stock market prediction based on social sentiments using machine learning. In *2018 international conference on smart city and emerging technology (ICSCET)*, pages 1–3. IEEE, 2018.

[83] Om Mane et al. Stock market prediction using natural language processing–a survey. *arXiv preprint arXiv:2208.13564*, 2022.

[84] Zhaoxia Wang, Seng-Beng Ho, and Zhiping Lin. Stock market prediction analysis by incorporating social and news opinion and sentiment. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1375–1380. IEEE, 2018.

[85] Weiling Chen, Yan Zhang, Chai Kiat Yeo, Chiew Tong Lau, and Bu Sung Lee. Stock market prediction using neural network through news on online social networks. In *2017 international smart cities conference (ISC2)*, pages 1–6. IEEE, 2017.

[86] Polash Dey, Emam Hossain, Md Ishtiaque Hossain, Mohammed Armanuzzaman Chowdhury, Md Shariful Alam, Mohammad Shahadat Hossain, and Karl Andersson. Comparative analysis of recurrent neural networks in stock price prediction for different frequency domains. *Algorithms*, 14(8):251, 2021.

[87] Ankit Thakkar and Kinjal Chaudhari. Fusion in stock market prediction: a decade survey on the necessity, recent developments, and potential future directions. *Information Fusion*, 65:95–107, 2021.

[88] Ranjan Kumar Behera, Sushree Das, Santanu Kumar Rath, Sanjay Misra, and Robertas Damasevicius. Comparative study of real time machine learning models for stock prediction through streaming data. *J. Univers. Comput. Sci.*, 26(9):1128–1147, 2020.

[89] Jean-Pierre Eckmann, S Oliffson Kamphorst, David Ruelle, et al. Recurrence plots of dynamical systems. *World Scientific Series on Nonlinear Science Series A*, 16:441–446, 1995.

[90] Norbert Marwan, M. Carmen Romano, Marco Thiel, and Jürgen Kurths. Recurrence plots for the analysis of complex systems. *Physics Reports*, 438(5):237–329, 2007.

[91] Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 3939–3945. AAAI Press, 2015.

[92] Ali N Akansu, Richard A Haddad, Paul A Haddad, and Paul R Haddad. *Multiresolution signal decomposition: transforms, subbands, and wavelets*. Academic press, 2001.

[93] Paul S Addison. Wavelet transforms and the ECG: a review. *Physiological measurement*, 26(5):R155–R199, aug 2005.

[94] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[95] Klaus Schwab. *The fourth industrial revolution*. Currency, 2017.

[96] Weiting Zhang, Dong Yang, and Hongchao Wang. Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE Systems Journal*, 13(3):2213–2227, 2019.

[97] Morteza Alizadeh and Junfeng Ma. A comparative study of series hybrid approaches to model and predict the vehicle operating states. *Computers & Industrial Engineering*, 162:107770, 2021.

[98] Roberto M. Souza, Erick G.S. Nascimento, Ubatan A. Miranda, Wenisten J.D. Silva, and Herman A. Lepikson. Deep learning for diagnosis and classification of faults in industrial rotating machinery. *Computers & Industrial Engineering*, 153:107060, 2021.

[99] Chuan-Jun Su and Shi-Feng Huang. Real-time big data analytics for hard disk drive predictive maintenance. *Computers & Electrical Engineering*, 71:93–101, 2018.

[100] Tiago Zonta, Cristiano André da Costa, Rodrigo da Rosa Righi, Miromar José de Lima, Eduardo Silveira da Trindade, and Guann Pyng Li. Predictive maintenance in the industry 4.0: A systematic literature review. *Computers & Industrial Engineering*, 150:106889, 2020.

[101] Thyago P. Carvalho, Fabrízzio A. A. M. N. Soares, Roberto Vita, Roberto da P. Francisco, João P. Basto, and Symone G. S. Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024, 2019.

[102] Héctor Cañas, Josefa Mula, Manuel Díaz-Madroñero, and Francisco Campuzano-Bolarín. Implementing industry 4.0 principles. *Computers & Industrial Engineering*, 158:107379, 2021.

[103] Sujie Geng and Xiuli Wang. Predictive maintenance scheduling for multiple power equipment based on data-driven fault prediction. *Computers & Industrial Engineering*, 164:107898, 2022.

[104] Elisa Yumi Nakagawa, Pablo Oliveira Antonino, Frank Schnicke, Rafael Capilla, Thomas Kuhn, and Peter Liggesmeyer. Industry 4.0 reference architectures: State of the art and future trends. *Computers & Industrial Engineering*, 156:107241, 2021.

[105] Yongyi Ran, Xin Zhou, Pengfeng Lin, Yonggang Wen, and Ruilong Deng. A survey of predictive maintenance: Systems, purposes and approaches. *arXiv preprint arXiv:1912.07383*, 2019.

[106] Thomas Rieger, Stefanie Regier, Ingo Stengel, and Nathan L Clarke. Fast predictive maintenance in industrial internet of things (iiot) with deep learning (dl): A review. In *CERC*, pages 69–80, 2019.

[107] Jovani Dalzochio, Rafael Kunst, Edison Pignaton, Alecio Binotto, Srijnan Sanyal, Jose Favilla, and Jorge Barbosa. Machine learning and

reasoning for predictive maintenance in industry 4.0: Current status and challenges. *Computers in Industry*, 123:103298, 2020.

[108] Sajja Tulasi Krishna and Hemantha Kumar Kalluri. Deep learning and transfer learning approaches for image classification. *International Journal of Recent Technology and Engineering (IJRTE)*, 7(5S4):427–432, 2019.

[109] Weiting Zhang, Dong Yang, and Hongchao Wang. Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE Systems Journal*, 13(3):2213–2227, 2019.

[110] Jovani Dalzochio, Rafael Kunst, Edison Pignaton, Alecio Binotto, Srijnan Sanyal, Jose Favilla, and Jorge Barbosa. Machine learning and reasoning for predictive maintenance in industry 4.0: Current status and challenges. *Computers in Industry*, 123:103298, 2020.

[111] Adir Solomon, Mor Kertis, Bracha Shapira, and Lior Rokach. A deep learning framework for predicting burglaries based on multiple contextual factors. *Expert Systems with Applications*, 199:117042, 2022.

[112] Danilo Giordano, Flavio Giobergia, Eliana Pastor, Antonio La Macchia, Tania Cerquitelli, Elena Baralis, Marco Mellia, and Davide Tricarico. Data-driven strategies for predictive maintenance: Lesson learned from an automotive use case. *Computers in Industry*, 134:103554, 2022.

[113] Sebastian Schwendemann, Zubair Amjad, and Axel Sikora. A survey of machine-learning techniques for condition monitoring and predictive maintenance of bearings in grinding machines. *Computers in Industry*, 125:103380, 2021.

[114] Lijun Zhang, Zhongqiang Mu, and Changyan Sun. Remaining useful life prediction for lithium-ion batteries based on exponential model and particle filter. *IEEE Access*, 6:17729–17740, 2018.

[115] Hua Han, Xiaoyu Cui, Yuqiang Fan, and Hong Qing. Least squares support vector machine (ls-svm)-based chiller fault diagnosis using fault indicative features. *Applied Thermal Engineering*, 154:540–547, 2019.

[116] Dong Wang, Kwok-Leung Tsui, and Qiang Miao. Prognostics and health management: A review of vibration based bearing and gear health indicators. *IEEE Access*, 6:665–676, 2018.

[117] Zepeng Liu and Long Zhang. A review of failure modes, condition monitoring and fault diagnosis methods for large-scale wind turbine bearings. *Measurement*, 149:107002, 2020.

[118] David Siegel, Canh Ly, and Jay Lee. Methodology and framework for predicting helicopter rolling element bearing failure. *IEEE Transactions on Reliability*, 61(4):846–857, 2012.

[119] Theodoros H. Loutas, Dimitrios Roulias, and George Georgoulas. Remaining useful life estimation in rolling bearings utilizing data-driven probabilistic e-support vectors regression. *IEEE Transactions on Reliability*, 62(4):821–832, 2013.

[120] Junqiang Liu, Chunlu Pan, Fan Lei, Dongbin Hu, and Hongfu Zuo. Fault prediction of bearings based on lstm and statistical process analysis. *Reliability Engineering & System Safety*, 214:107646, 2021.

[121] Yi Qin, Dingliang Chen, Sheng Xiang, and Caichao Zhu. Gated dual attention unit neural networks for remaining useful life prediction of rolling bearings. *IEEE Transactions on Industrial Informatics*, 17(9):6438–6447, 2021.

[122] Preethi Anantharaman, Mu Qiao, and Divyesh Jadav. Large scale predictive analytics for hard disk remaining useful life estimation. In *2018 IEEE International Congress on Big Data (BigData Congress)*, pages 251–254, 2018.

[123] Sanchita Basak, Saptarshi Sengupta, and Abhishek Dubey. Mechanisms for integrated feature normalization and remaining useful life estimation using lstms applied to hard-disks. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 208–216, 2019.

[124] Fernando Dione S. Lima, Francisco Lucas F. Pereira, Lucas G. M. Leite, Joao Paulo P. Gomes, and Javam C. Machado. Remaining useful life estimation of hard disk drives based on deep neural networks. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2018.

[125] Huimin Zhao, Haodong Liu, Yang Jin, Xiangjun Dang, and Wu Deng. Feature extraction for data-driven remaining useful life prediction of rolling bearings. *IEEE Transactions on Instrumentation and Measurement*, 70:1–10, 2021.

[126] Antoine Guillaume, Christel Vrain, and Elloumi Wael. Time series classification for predictive maintenance on event logs. *arXiv preprint arXiv:2011.10996*, 2020.

[127] Ying-Yi Hong, John Joel F. Martinez, and Arnel C. Fajardo. Day-ahead solar irradiation forecasting utilizing gramian angular field and convolutional long short-term memory. *IEEE Access*, 8:18741–18753, 2020.

[128] Yang Liu, Keze Wang, Guanbin Li, and Liang Lin. Semantics-aware adaptive knowledge distillation for sensor-to-vision action recognition. *IEEE Transactions on Image Processing*, 30:5573–5588, 2021.

[129] Zhen Qin, Yibo Zhang, Shuyu Meng, Zhiguang Qin, and Kim-Kwang Raymond Choo. Imaging and fusing time series for wearable sensor-based human activity recognition. *Information Fusion*, 53:80–87, 2020.

[130] Silvio Barra, Salvatore Mario Carta, Andrea Corriga, Alessandro Sebastian Podda, and Diego Reforgiato Recupero. Deep learning and time series-to-image encoding for financial forecasting. *IEEE/CAA Journal of Automatica Sinica*, 7(3):683–692, 2020.

[131] Kahiomba Sonia Kiangala and Zenghui Wang. An effective predictive maintenance framework for conveyor motors using dual time-series imaging and convolutional neural network in an industry 4.0 environment. *IEEE Access*, 8:121033–121049, 2020.

[132] Renxiang Chen, Xin Huang, Lixia Yang, Xiangyang Xu, Xia Zhang, and Yong Zhang. Intelligent fault diagnosis method of planetary gearboxes based on convolution neural network and discrete wavelet transform. *Computers in Industry*, 106:48–59, 2019.

[133] Pengfei Liang, Chao Deng, Jun Wu, Zhixin Yang, Jinxuan Zhu, and Zihan Zhang. Compound fault diagnosis of gearboxes via multi-label convolutional neural network and wavelet transform. *Computers in Industry*, 113:103132, 2019.

[134] Turker Tuncer, Sengul Dogan, Paweł Pławiak, and U. Rajendra Acharya. Automated arrhythmia detection using novel hexadecimal local pattern and multilevel wavelet transform with ecg signals. *Knowledge-Based Systems*, 186:104923, 2019.

[135] Ruobin Gao, Liang Du, Okan Duru, and Kum Fai Yuen. Time series forecasting based on echo state network and empirical wavelet transformation. *Applied Soft Computing*, 102:107111, 2021.

[136] M. Bugueño, G. Molina, F. Mena, P. Olivares, and M. Araya. Harnessing the power of cnns for unevenly-sampled light-curves using markov transition field. *Astronomy and Computing*, 35:100461, 2021.

[137] K Vandith Sreenivas, M Ganesan, and R Lavanya. Classification of arrhythmia in time series ecg signals using image encoding and convolutional neural networks. In *2021 Seventh International conference on Bio Signals, Images, and Instrumentation (ICBSII)*, pages 1–6, 2021.

[138] Jianjie Lu and Kai-Yu Tong. Robust single accelerometer-based activity recognition using modified recurrence plot. *IEEE Sensors Journal*, 19(15):6317–6324, 2019.

[139] Luis C.S. Afonso, Gustavo H. Rosa, Clayton R. Pereira, Silke A.T. Weber, Christian Hook, Victor Hugo C. Albuquerque, and João P. Papa. A recurrence plot-based approach for parkinson's disease identification. *Future Generation Computer Systems*, 94:282–292, 2019.

[140] Ye Zhang, Yi Hou, Kewei OuYang, and Shilin Zhou. Multi-scale signed recurrence plot based time series classification using inception architectural networks. *Pattern Recognition*, page 108385, 2021.

[141] Chao-Lung Yang, Zhi-Xuan Chen, and Chen-Yi Yang. Sensor classification using convolutional neural network by encoding multivariate time series as two-dimensional colored images. *Sensors*, 20(1), 2020.

[142] S. Chan, I. Oktavianti, and V. Puspita. A deep learning cnn and ai-tuned svm for electricity consumption forecasting: Multivariate time series data. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0488–0494, 2019.

[143] Wei Chen, Manrui Jiang, Wei-Guo Zhang, and Zhensong Chen. A novel graph convolutional feature based convolutional neural network for stock trend prediction. *Information Sciences*, 556:67–94, 2021.

[144] Muhammad Fahim, Khadija Fraz, and Alberto Sillitti. Tsi: Time series to imaging based model for detecting anomalous energy consumption in smart buildings. *Information Sciences*, 523:1–13, 2020.

[145] Ruinan Zhang, Fanglan Zheng, and Wei Min. Sequential behavioral data processing using deep learning and the markov transition field in online fraud detection. *arXiv preprint arXiv:1808.05329*, 2018.

[146] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[147] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[148] Michał Markiewicz, Maciej Wielgosz, Mikołlaj Bocheński, Waldemar Tabaczyński, Tomasz Konieczny, and Liliana Kowalczyk. Predictive maintenance of induction motors using ultra-low power wireless sensors and compressed recurrent neural networks. *IEEE Access*, 7:178891–178902, 2019.

[149] Robert P.W. Duin. A note on comparing classifiers. *Pattern Recognition Letters*, 17(5):529–536, 1996.

[150] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[151] Mohendra Roy, Sumon Kumar Bose, Bapi Kar, Pradeep Kumar Gopalakrishnan, and Arindam Basu. A stacked autoencoder neural network based automated feature extraction method for anomaly detection in on-line condition monitoring, 2018.

[152] Jianjie Lu and Kai-Yu Tong. Robust single accelerometer-based activity recognition using modified recurrence plot. *IEEE Sensors Journal*, 19(15):6317–6324, 2019.

[153] Zeeshan Ahmad, Anika Tabassum, Ling Guan, and Naimul Mefraz Khan. Ecg heartbeat classification using multimodal fusion. *IEEE Access*, 9:100615–100626, 2021.

[154] Zeeshan Ahmad and Naimul Khan. Inertial sensor data to image encoding for human action recognition. *IEEE Sensors Journal*, 21(9):10978–10988, 2021.

[155] Emil Blixt Hansen and Simon Bøgh. Artificial intelligence and internet of things in small and medium-sized enterprises: A survey. *Journal of Manufacturing Systems*, 58:362–372, 2021.

[156] Alberto Petrillo, Antonio Picariello, Stefania Santini, Biagio Scarciello, and Giancarlo Sperlí. Model-based vehicular prognostics framework using big data architecture. *Computers in Industry*, 115:103177, 2020.

[157] Peng Liu, Lizhe Wang, Rajiv Ranjan, Guojin He, and Lei Zhao. A survey on active deep learning: from model-driven to data-driven. *ACM Computing Surveys (CSUR)*, 2021.

[158] Weichao Luo, Tianliang Hu, Yingxin Ye, Chengrui Zhang, and Yongli Wei. A hybrid predictive maintenance approach for cnc machine tool driven by digital twin. *Robotics and Computer-Integrated Manufacturing*, 65:101974, 2020.

[159] Ming Zhang, Nasser Amaitik, Zezhong Wang, Yuchun Xu, Alexander Maisuradze, Michael Peschl, and Dimitrios Tzovaras. Predictive maintenance for remanufacturing based on hybrid-driven remaining useful life prediction. *Applied Sciences*, 12(7):3218, 2022.

[160] Yyi Kai Teoh, Sukhpal Singh Gill, and Ajith Kumar Parlikad. Iot and fog computing based predictive maintenance model for effective asset management in industry 4.0 using machine learning. *IEEE Internet of Things Journal*, 2021.

[161] Marcelo Brandalero, Muhammad Ali, Laurens Le Jeune, Hector Gerardo Muñoz Hernandez, Mitko Veleski, Bruno da Silva, Jan Lemeire, Kristof Van Beeck, Abdellah Touhafi, Toon Goedemé, et al. Aitia: Embedded ai techniques for embedded industrial applications. In *2020 International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–7. IEEE, 2020.

[162] Guilherme Tortorella, Tarcisio Abreu Saurin, Flavio Sanson Fogliatto, Diego Tlapa, José Moyano-Fuentes, Paolo Gaiardelli, Zahra Seyedghorban, Roberto Vassolo, Alejandro Francisco Mac Cawley, V Raja Sreedharan, et al. The impact of industry 4.0 on the relationship between tpm and maintenance performance. *Journal of Manufacturing Technology Management*, 2022.

[163] Camilla Lundgren, Jon Bokrantz, and Anders Skoogh. A strategy development process for smart maintenance implementation. *Journal of Manufacturing Technology Management*, 32(9):142–166, 2021.

[164] Roberto Sala, Marco Bertoni, Fabiana Pirola, and Giuditta Pezzotta. Data-based decision-making in maintenance service delivery: the d3m framework. *Journal of Manufacturing Technology Management*, 32(9):122–141, 2021.

[165] Christian F Durach, Joakim Kembro, and Andreas Wieland. A new paradigm for systematic literature reviews in supply chain management. *Journal of Supply Chain Management*, 53(4):67–85, 2017.

[166] Rob Dekkers, Lindsey Carey, and Peter Langhorne. *Making Literature Reviews Work: A Multidisciplinary Guide to Systematic Approaches*. Springer, 2021.

[167] Gurkan Aydemir and Burak Acar. Anomaly monitoring improves remaining useful life estimation of industrial machinery. *Journal of Manufacturing Systems*, 56:463–469, 2020.

[168] André Listou Ellefsen, Emil Bjørlykhaug, Vilmar Æsøy, Sergey Ushakov, and Houxiang Zhang. Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, 183:240–251, 2019.

[169] Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, Ruqiang Yan, and Xiaoli Li. Attention-based sequence to sequence model for machine remaining useful life prediction. *Neurocomputing*, 466:58–68, 2021.

[170] Ali Al-Dulaimi, Soheil Zabihi, Amir Asif, and Arash Mohammadi. A multimodal and hybrid deep neural network model for remaining useful life estimation. *Computers in Industry*, 108:186–196, 2019.

[171] Alex Falcon, Giovanni D'Agostino, Giuseppe Serra, Giorgio Brajnik, Carlo Tasso, and Fondazione Bruno Kessler. A dual-stream architecture based on neural turing machine and attention for the remaining useful life estimation problem. 5(1):10–10, 2020.

[172] Ali Al-Dulaimi, Soheil Zabihi, Amir Asif, and Arash Mohammed. Nblstm: Noisy and hybrid convolutional neural network and blstm-based deep architecture for remaining useful life estimation. *Journal of Computing and Information Science in Engineering*, 20(2):021012, 2020.

[173] Chong Chen, Ying Liu, Xianfang Sun, Carla Di Cairano-Gilfedder, and Scott Titmus. Automobile maintenance prediction using deep learning with gis data. *Procedia CIRP*, 81:447–452, 2019. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

[174] Qi Wang, Siqi Bu, and Zhengyou He. Achieving predictive and proactive maintenance for high-speed railway power equipment with lstm-rnn. *IEEE Transactions on Industrial Informatics*, 16(10):6509–6517, 2020.

[175] Kahiomba Sonia Kiangala and Zenghui Wang. An effective predictive maintenance framework for conveyor motors using dual time-series imaging and convolutional neural network in an industry 4.0 environment. *IEEE Access*, 8:121033–121049, 2020.

[176] Haiyue Wu, Aihua Huang, and John W. Sutherland. Avoiding environmental consequences of equipment failure via an lstm-based model for predictive maintenance. *Procedia Manufacturing*, 43:666–673, 2020. Sustainable Manufacturing - Hand in Hand to Sustainability on Globe: Proceedings of the 17th Global Conference on Sustainable Manufacturing.

[177] Samira Zare and Moosa Ayati. Simultaneous fault diagnosis of wind turbine using multichannel convolutional neural networks. *ISA Transactions*, 108:230–239, 2021.

[178] Zhe Li, Jingyue Li, Yi Wang, and Kesheng Wang. A deep learning approach for anomaly detection based on sae and lstm in mechanical equipment. *The International Journal of Advanced Manufacturing Technology*, 103(1):499–510, 2019.

[179] Yuanhang Chen, Gaoliang Peng, Zhiyu Zhu, and Sijue Li. A novel deep learning method based on attention mechanism for bearing remaining useful life prediction. *Applied Soft Computing*, 86:105919, 2020.

[180] Gabriel Rodriguez Garcia, Gabriel Michau, Mélanie Ducoffe, Jayant Sen Gupta, and Olga Fink. Time series to images: Monitoring the condition of industrial assets with deep learning image processing algorithms. *arXiv preprint arXiv:2005.07031*, 2020.

[181] Giovanna Martínez-Arellano, German Terrazas, and Svetan Ratchev. Tool wear classification using time series imaging and deep learning.

The International Journal of Advanced Manufacturing Technology, 104(9):3647–3662, 2019.

[182] Luis A Pinedo-Sanchez, Diego A Mercado-Ravell, and Carlos A Carballo-Monsivais. Vibration analysis in bearings for failure prevention using cnn. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 42(12):1–17, 2020.

[183] Sidi Lu, Bing Luo, Tirthak Patel, Yongtao Yao, Devesh Tiwari, and Weisong Shi. Making disk failure predictions {SMARTer}! pages 151–167, 2020.

[184] Hongzhang Yang, Zongzhao Li, Huiyuan Qiang, Zhongliang Li, Yaofeng Tu, and Yahui Yang. Zte-predictor: Disk failure prediction system based on lstm. pages 17–20, 2020.

[185] Aniello De Santo, Antonio Galli, Michela Gravina, Vincenzo Moscato, and Giancarlo Sperlì. Deep learning for hdd health assessment: An application based on lstm. *IEEE Transactions on Computers*, 71(1):69–80, 2022.

[186] Harsh Purohit, Ryo Tanabe, Kenji Ichige, Takashi Endo, Yuki Nikaido, Kaori Suefusa, and Yohei Kawaguchi. Mimii dataset: Sound dataset for malfunctioning industrial machine investigation and inspection. *arXiv preprint arXiv:1909.09347*, 2019.

[187] Faatih Nuraliah Binti Sohaidan, Amgad Muneer, and Shakirah Mohd Taib. Remaining useful life prediction of turbofan engine using long-short term memory. In *2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pages 1–6. IEEE, 2021.

[188] Hadis Hesabi, Mustapha Nourelfath, and Adnène Hajji. A deep learning predictive model for selective maintenance optimization. *Reliability Engineering & System Safety*, 219:108191, 2022.

[189] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[190] Xingyu Li, Vasiliy Krivtsov, and Karunesh Arora. Attention-based deep survival model for time series data. *Reliability Engineering & System Safety*, 217:108033, 2022.

[191] Jeffrey Chen, Sehwan Hong, Warrick He, Jinyeong Moon, and Sang-Woo Jun. Eciton: Very low-power lstm neural network accelerator for predictive maintenance at the edge. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pages 1–8. IEEE, 2021.

[192] Chang Woo Hong, Min-Seung Ko, and Kyeon Hur. Convnet-based remaining useful life prognosis of a turbofan engine. In *2021 IEEE 4th International Conference on Knowledge Innovation and Invention (ICKII)*, pages 190–193. IEEE, 2021.

[193] Jiusi Zhang, Yuchen Jiang, Shimeng Wu, Xiang Li, Hao Luo, and Shen Yin. Prediction of remaining useful life based on bidirectional gated recurrent unit with temporal self-attention mechanism. *Reliability Engineering & System Safety*, page 108297, 2022.

[194] Maxim Shcherbakov and Cuong Sai. A hybrid deep learning framework for intelligent predictive maintenance of cyber-physical systems. *ACM Transactions on Cyber-Physical Systems*, 2022.

[195] Mehdi Mohammadi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys Tutorials*, 20(4):2923–2960, 2018.

[196] Huan Song, Deepta Rajan, Jayaraman Thiagarajan, and Andreas Spanias. Attend and diagnose: Clinical time series analysis using attention models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[197] A. Saxena, K. Goebel, D. Simon, N. Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. *2008 International Conference on Prognostics and Health Management*, pages 1–9, 2008.

[198] Emmanuel Ramasso. Investigating computational geometry for failure prognostics in presence of imprecise health indicator: Results and comparisons on c-mapss datasets. In *PHM Society European Conference*, volume 2, 2014.

[199] Carlo Concari and Giada Bettini. Embedded implementation of rainflow-counting for on-line predictive maintenance. In *2020 IEEE*

Energy Conversion Congress and Exposition (ECCE), pages 981–988. IEEE, 2020.

[200] Carlos Resende, Duarte Folgado, João Oliveira, Bernardo Franco, Waldir Moreira, Antonio Oliveira-Jr, Armando Cavaleiro, and Ricardo Carvalho. Tip4. 0: Industrial internet of things platform for predictive maintenance. *Sensors*, 21(14):4676, 2021.

[201] Lorenzo Gigoni, Alessandro Betti, Mauro Tucci, and Emanuele Crisostomi. A scalable predictive maintenance model for detecting wind turbine component failures based on scada data. In *2019 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5, 2019.

[202] J. Li, X. Li, D. He. A directed acyclic graph network combined with CNN and LSTM for remaining useful life prediction. *IEEE Access*, 7:75464–75475, 2019.

[203] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[204] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image classification using random forests and ferns. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. Ieee, 2007.

[205] Yanxiong Sun, Yeli Li, Qingtao Zeng, and Yuning Bian. Application research of text classification based on random forest algorithm. In *2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, pages 370–374. IEEE, 2020.

[206] Parveen Sihag, Sahar Mohsenzadeh Karimi, and Anastasia Angelaki. Random forest, m5p and regression analysis to estimate the field unsaturated hydraulic conductivity. *Applied Water Science*, 9:1–9, 2019.

[207] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[208] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

[209] Xin Zhou, Jianmin Pang, and Guanghui Liang. Image classification for malware detection using extremely randomized trees. In *2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, pages 54–59. IEEE, 2017.

[210] Panikos Heracleous, Yasser Mohammad, and Akio Yoneyama. Integrating language and emotion features for multilingual speech emotion recognition. In *Human-Computer Interaction. Multimodal and Natural Interaction: Thematic Area, HCI 2020, Held as Part of the 22nd International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part II 22*, pages 187–196. Springer, 2020.

[211] Chinedu I Ossai and Ifeanyi P Egwutuoha. Anomaly detection and extra tree regression for assessment of the remaining useful life of lithium-ion battery. In *Advanced Information Networking and Applications: Proceedings of the 34th International Conference on Advanced Information Networking and Applications (AINA-2020)*, pages 1474–1488. Springer, 2020.

[212] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[213] Omid Kohannim, Derrek P Hibar, Jason L Stein, Neda Jahanshad, Xue Hua, Priya Rajagopalan, Arthur W Toga, Clifford R Jack Jr, Michael W Weiner, Greig I De Zubicaray, et al. Discovery and replication of gene influences on brain structure using lasso regression. *Frontiers in neuroscience*, 6:115, 2012.

[214] Xi Nie, Guangming Deng, et al. Enterprise financial early warning based on lasso regression screening variables. *Journal of Financial Risk Management*, 9(04):454, 2020.

[215] Tom Goldstein, Min Li, and Xiaoming Yuan. Adaptive primal-dual splitting methods for statistical learning and image processing. *Advances in neural information processing systems*, 28, 2015.

[216] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[217] Kamilya Smagulova and Alex Pappachen James. A survey on lstm memristive neural network architectures and applications. *The European Physical Journal Special Topics*, 228(10):2313–2324, 2019.

[218] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[219] Li Yang, Ying Li, Jin Wang, and Zhuo Tang. Post text processing of chinese speech recognition based on bidirectional lstm networks and crf. *Electronics*, 8(11):1248, 2019.

[220] Rushali Dhumal Deshmukh and Arvind Kiwelekar. Deep learning techniques for part of speech tagging by natural language processing. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 76–81. IEEE, 2020.

[221] Cheng Wang, Haojin Yang, Christian Bartz, and Christoph Meinel. Image captioning with deep bidirectional lstms. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 988–997, 2016.

[222] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[223] Ben Athiwaratkun and Jack W Stokes. Malware classification with lstm and gru language models and a character-level cnn. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2482–2486. IEEE, 2017.

[224] Biao Zhang, Deyi Xiong, Jun Xie, and Jinsong Su. Neural machine translation with gru-gated attention model. *IEEE transactions on neural networks and learning systems*, 31(11):4688–4698, 2020.

[225] Muhammad Zulqarnain, Rozaida Ghazali, Muhammad Aamir, and Yana Mazwin Mohmad Hassim. An efficient two-state gru based on feature attention mechanism for sentiment analysis. *Multimedia Tools and Applications*, pages 1–26, 2022.

[226] Yuan Yuan, Chunlin Tian, and Xiaoqiang Lu. Auxiliary loss multimodal gru model in audio-visual speech recognition. *IEEE Access*, 6:5573–5583, 2018.

[227] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005: 15th International Conference, Warsaw, Poland, September 11-15, 2005. Proceedings, Part II 15*, pages 799–804. Springer, 2005.

[228] O Bustos and A. Pomares-Quimbaya. Stock market movement forecast: A Systematic review. *Expert Systems with Applications*, 156:113464, 2020.

[229] Ankit Thakkar and Kinjal Chaudhari. A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions. *Expert Systems with Applications*, 177:114800, 2021.

[230] Ge Zhang, Zhao Li, Jiaming Huang, Jia Wu, Chuan Zhou, Jian Yang, and Jianliang Gao. Efraudcom: An e-commerce fraud detection system via competitive graph neural networks. *ACM Transactions on Information Systems*, 40(3), mar 2022.

[231] T.O. Kehinde, Felix T.S. Chan, and S.H. Chung. Scientometric review and analysis of recent approaches to stock market forecasting: Two decades survey. *Expert Systems with Applications*, 213:119299, 2023.

[232] Dattatray P. Gandhmal and K. Kumar. Systematic analysis and review of stock market prediction techniques. *Computer Science Review*, 34:100190, 2019.

[233] Jing Zhang, Shicheng Cui, Yan Xu, Qianmu Li, and Tao Li. A novel data-driven stock price trend prediction system. *Expert Systems with Applications*, 97:60–69, 2018.

[234] Isaac Kofi Nti, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. A systematic review of fundamental and technical analysis of stock market predictions. *Artificial Intelligence Review*, 53(4):3007–3057, 2020.

[235] Mahinda Mailagaha Kumbure, Christoph Lohrmann, Pasi Luukka, and Jari Porras. Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications*, 197:116659, 2022.

[236] Weiwei Jiang. Applications of deep learning in stock market prediction: Recent progress. *Expert Systems with Applications*, 184:115537, 2021.

[237] Ankit Thakkar and Kinjal Chaudhari. Fusion in stock market prediction: A decade survey on the necessity, recent developments, and potential future directions. *Information Fusion*, 65:95–107, 2021.

[238] Sepideh Bazzaz Abkenar, Mostafa Haghi Kashani, Ebrahim Mahdipour, and Seyed Mahdi Jameii. Big data analytics meets social media: A systematic review of techniques, open issues, and future directions. *Telematics and Informatics*, 57:101517, 2021.

[239] Xiaolong Zheng, Xiao Wang, Zepeng Li, Rongrong Jing, Shuqi Xu, Tao Wang, Lifang Li, Zhenwen Zhang, Qingpeng Zhang, Huaiguang Jiang, Zhihua Guo, Xiaowei Zhang, and Fei-Yue Wang. Donald J. Trump's Presidency in Cyberspace: A Case Study of Social Perception and Social Influence in Digital Oligarchy Era. *IEEE Transactions on Computational Social Systems*, 8(2):279–293, 2021.

[240] Huihui Ni, Shuting Wang, and Peng Cheng. A hybrid approach for stock trend prediction based on tweets embedding and historical prices. *World Wide Web*, 24:849–868, 2021.

[241] Shan Lu, Chenhui Liu, and Zhensong Chen. Predicting stock market crisis via market indicators and mixed frequency investor sentiments. *Expert Systems with Applications*, 186:115844, 2021.

[242] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems*, 37(2), mar 2019.

[243] Noella Nazareth and Yeruva Venkata Ramana Reddy. Financial applications of machine learning: A literature review. *Expert Systems with Applications*, 219:119640, 2023.

[244] Sotirios P Chatzis, Vassilis Siakoulis, Anastasios Petropoulos, Evangelos Stavroulakis, and Nikos Vlachogiannakis. Forecasting stock market crisis events using deep and statistical machine learning techniques. *Expert systems with applications*, 112:353–371, 2018.

[245] Sarbjit Singh, Kulwinder Singh Parmar, and Jatinder Kumar. Soft computing model coupled with statistical models to estimate future of stock market. *Neural Computing and Applications*, 33:7629–7647, 2021.

[246] Rebecca Salles, Kele Belloze, Fabio Porto, Pedro H Gonzalez, and Eduardo Ogasawara. Nonstationary time series transformation methods: An experimental review. *Knowledge-Based Systems*, 164:274–291, 2019.

[247] Longbing Cao. AI in Finance: Challenges, Techniques, and Opportunities. *ACM Computing Surveys*, 55(3), feb 2022.

[248] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020.

[249] Yu Ma, Rui Mao, Qika Lin, Peng Wu, and Erik Cambria. Multisource aggregated classification for stock price movement prediction. *Information Fusion*, 91:515–528, 2023.

[250] Matin N. Ashtiani and Bijan Raahemi. News-based intelligent prediction of financial markets using text mining and machine learning: A systematic literature review. *Expert Systems with Applications*, 217:119509, 2023.

[251] Shuang (Sophie) Zhai and Zhu (Drew) Zhang. Read the News, Not the Books: Forecasting Firms' Long-Term Financial Performance via Deep Text Mining. *ACM Transactions on Management Information Systems*, 14(1), jan 2023.

[252] Francesco Colasanto, Luca Grilli, Domenico Santoro, and Giovanni Villani. AlBERTino for stock price prediction: a Gibbs sampling approach. *Information Sciences*, 597:341–357, 2022.

[253] Nohyoon Seong and Kihwan Nam. Predicting stock movements based on financial news with segmentation. *Expert Systems with Applications*, 164:113988, 2021.

[254] Paramita Ray, Bhaswati Ganguli, and Amlan Chakrabarti. A Hybrid Approach of Bayesian Structural Time Series With LSTM to Identify the Influence of News Sentiment on Short-Term Forecasting of Stock Price. *IEEE Transactions on Computational Social Systems*, 8(5):1153–1162, 2021.

[255] Jiyeon Im and Eunil Park. Effects of political orientation on sentiment features: the case of online news outlets in south korea. *Telematics and Informatics*, 74:101882, 2022.

[256] Young Anna Argyris, Victoria R. Nelson, Kaleigh Wiseley, Ruoyu Shen, and Alexa Roscizewski. Do social media campaigns foster vaccination adherence? A systematic review of prior intervention-based

campaigns on social media. *Telematics and Informatics*, 76:101918, 2023.

[257] Shinyoung Park and Jaemin Jung. The interplay between social media virality metrics and message framing in influence perception of pro-environmental messages and behavioral intentions. *Telematics and Informatics*, 78:101947, 2023.

[258] Hua Pang. Identifying associations between mobile social media users' perceived values, attitude, satisfaction, and ewom engagement: The moderating role of affective factors. *Telematics and Informatics*, 59:101561, 2021.

[259] Nan Jing, Zhao Wu, and Hefei Wang. A hybrid model integrating deep learning with investor sentiment analysis for stock price prediction. *Expert Systems with Applications*, 178:115019, 2021.

[260] Xiaolong Zheng, Hu Tian, Zhe Wan, Xiao Wang, Daniel Dajun Zeng, and Fei-Yue Wang. Game Starts at GameStop: Characterizing the Collective Behaviors and Social Dynamics in the Short Squeeze Episode. *IEEE Transactions on Computational Social Systems*, 9(1):45–58, 2022.

[261] Tengteng Liu, Xiang Ma, Shuo Li, Xuemei Li, and Caiming Zhang. A stock price prediction method based on meta-learning and variational mode decomposition. *Knowledge-Based Systems*, 252:109324, 2022.

[262] Anna Mancini, Antonio Desiderio, Riccardo Di Clemente, and Giulio Cimini. Self-induced consensus of reddit users to characterise the gamestop short squeeze. *Scientific reports*, 12(1):13780, 2022.

[263] Ehsan-Ul Haq, Tristan Braud, Lik-Hang Lee, Anish K. Vallapuram, Yue Yu, Gareth Tyson, and Pan Hui. Short, colorful, and irreverent! a comparative analysis of new users on wallstreetbets during the gamestop short-squeeze. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 52–61, New York, NY, USA, 2022. Association for Computing Machinery.

[264] Ehsan Hoseinzade and Saman Haratizadeh. CNNpred: CNN-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273–285, 2019.

[265] Lihong Wei, Jiankun Gong, Jing Xu, Nor Eeza Zainal Abidin, and Oberiri Destiny Apuke. Do social media literacy skills help in combating fake news spread? Modelling the moderating role of social media literacy skills in the relationship between rational choice factors and fake news sharing behaviour. *Telematics and Informatics*, 76:101910, 2023.

[266] Ruoyu Sun, Cong Li, Barbara Millet, Khudejah Iqbal Ali, and John Petit. Sharing news with online friends: A study of network homophily, network size, and news type. *Telematics and Informatics*, 67:101763, 2022.

[267] Yang Liu and Yi-Fang Brook Wu. Fned: A deep network for fake news early detection on social media. *ACM Transactions on Information Systems*, 38(3), may 2020.

[268] Oscar Bustos and Alexandra Pomares-Quimbaya. Stock market movement forecast: A systematic review. *Expert Systems with Applications*, 156:113464, 2020.

[269] Ali Derakhshan and Hamid Beigy. Sentiment analysis on stock social media for stock price movement prediction. *Engineering Applications of Artificial Intelligence*, 85:569–578, 2019.

[270] Wasiat Khan, Mustansar Ali Ghazanfar, Muhammad Awais Azam, Amin Karami, Khaled H Alyoubi, and Ahmed S Alfakeeh. Stock market prediction using machine learning classifiers and social media, news. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–24, 2020.

[271] Weiwei Jiang. Applications of deep learning in stock market prediction: recent progress. *Expert Systems with Applications*, 184:115537, 2021.

[272] Matin N Ashtiani and Bijan Raahmei. News-based intelligent prediction of financial markets using text mining and machine learning: A systematic literature review. *Expert Systems with Applications*, page 119509, 2023.

[273] Nusrat Rouf, Majid Bashir Malik, Tasleem Arif, Sparsh Sharma, Saurabh Singh, Satyabrata Aich, and Hee-Cheol Kim. Stock market prediction using machine learning techniques: A decade survey on methodologies, recent developments, and future directions. *Electronics*, 10(21), 2021.

[274] Anubhav Sarkar, Swagata Chakraborty, Sohom Ghosh, and Sudip Kumar Naskar. Evaluating impact of social media posts by executives on stock prices. In *Proceedings of the 14th Annual Meeting of the Forum for Information Retrieval Evaluation*, FIRE '22, page 74–82, New York, NY, USA, 2023. Association for Computing Machinery.

[275] Kingstone Nyakurukwa and Yudhvir Seetharam. The evolution of studies on social media sentiment in the stock market: Insights from bibliometric analysis. *Scientific African*, 20:e01596, 2023.

[276] Yili Wang, Jiaxuan Guo, Chengsheng Yuan, and Baozhu Li. Sentiment analysis of twitter data. *Applied Sciences*, 12(22):11775, 2022.

[277] Charlie Wang and Ben Luo. Predicting $ gme stock price movement using sentiment from reddit r/wallstreetbets. In *Proceedings of the Third Workshop on Financial Technology and Natural Language Processing*, pages 22–30, 2021.

[278] Matheus Gomes Sousa, Kenzo Sakiyama, Lucas de Souza Rodrigues, Pedro Henrique Moraes, Eraldo Rezende Fernandes, and Edson Takashi Matsubara. Bert for stock market sentiment analysis. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1597–1601. IEEE, 2019.

[279] Wasiat Khan, Mustansar ali Ghazanfar, Muhammad Awais Azam, Amin Karami, Khaled Alyoubi, and Ahmed Alfakeeh. Stock market prediction using machine learning classifiers and social media, news. *Journal of Ambient Intelligence and Humanized Computing*, 13, 07 2022.

[280] Shengting Wu, Yuling Liu, Ziran Zou, and Tien-Hsiung Weng. S_i_lstm: stock price prediction based on multiple data sources and sentiment analysis. *Connection Science*, 34(1):44–62, 2022.

[281] Priyadarshan Dhabe, Ayush Chandak, Om Deshpande, Pratik Fandade, Naman Chandak, and Yash Oswal. Stock market trend prediction along with twitter sentiment analysis. In Valentina Emilia Balas, Vijay Bhaskar Semwal, and Anand Khandare, editors, *Intelligent Computing and Networking*, pages 45–59, Singapore, 2023. Springer Nature Singapore.

[282] Tinku Singh, Siddhant Bhisikar, Satakshi, and Manish Kumar. Stock market prediction using ensemble learning and sentimental analysis.

In Rajesh Doriya, Badal Soni, Anupam Shukla, and Xiao-Zhi Gao, editors, *Machine Learning, Image Processing, Network Security and Data Sciences*, pages 429–441, Singapore, 2023. Springer Nature Singapore.

[283] Ljupco Todorovski and Savso Dzeroski. Combining classifiers with meta decision trees. *Machine Learning*, 50:223–249, 2004.

[284] Zexin Hu, Yiqi Zhao, and Matloob Khushi. A survey of forex and stock price prediction using deep learning. *Applied System Innovation*, 4(1), 2021.

[285] Dev Shah, Haruna Isah, and Farhana Zulkernine. Stock market analysis: A review and taxonomy of prediction techniques. *International Journal of Financial Studies*, 7(2), 2019.

[286] Dennis Huynh, Garrett Audet, Nikolay Alabi, and Yuan Tian. Stock price prediction leveraging reddit: The role of trust filter and sliding window. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1054–1060. IEEE, 2021.

[287] Arnav Machavarapu. Reddit sentiments effects on stock market prices. In *Smart Intelligent Computing and Applications, Volume 1: Proceedings of Fifth International Conference on Smart Computing and Informatics (SCI 2021)*, pages 75–84. Springer, 2022.

[288] Christopher N Broadhurst, Piotr Szczurek, et al. Data analytics on nasdaq stock prices: Reddit social media case study. In *2022 IEEE International Conference on Electro Information Technology (eIT)*, pages 053–060. IEEE, 2022.

[289] Román A. Mendoza-Urdiales, José Antonio Núñez-Mora, Roberto J. Santillán-Salgado, and Humberto Valencia-Herrera. Twitter sentiment analysis and influence on stock performance using transfer entropy and egarch methods. *Entropy*, 24(7), 2022.

[290] T Swathi, N Kasiviswanath, and A Ananda Rao. An optimal deep learning-based lstm for stock price prediction using twitter sentiment analysis. *Applied Intelligence*, 52(12):13675–13688, 2022.

[291] Jaimin Shah, Darsh Vaidya, and Manan Shah. A comprehensive review on multiple hybrid deep learning approaches for stock prediction. *Intelligent Systems with Applications*, 16:200111, 2022.

[292] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang Koh, Quoc Le, and Andrew Ng. Tiled convolutional neural networks. *Advances in neural information processing systems*, 23, 2010.

List of figures

List of tables