Applications of Deep Learning Models in Financial Forecasting

Ahoora Rostamian

A thesis submitted for the degree of Doctor of Philosophy

Centre for Computational Finance and Economic Agents

School of Computer Science and Electronic Engineering

University of Essex

September 2023

Abstract

In financial markets, deep learning techniques sparked a revolution, reshaping conventional approaches and amplifying predictive capabilities. This thesis explored the applications of deep learning models to unravel insights and methodologies aimed at advancing financial forecasting. The crux of the research problem lies in the applications of predictive models within financial domains, characterised by high volatility and uncertainty. This thesis investigated the application of advanced deep-learning methodologies in the context of financial forecasting, addressing the challenges posed by the dynamic nature of financial markets. These challenges were tackled by exploring a range of techniques, including convolutional neural networks (CNNs), long short-term memory networks (LSTMs), autoencoders (AEs), and variational autoencoders (VAEs), along with approaches such as encoding financial time series into images. Through analysis, methodologies such as transfer learning, convolutional neural networks, long short-term memory networks, generative modelling, and image encoding of time series data were examined. These methodologies collectively offered a comprehensive toolkit for extracting meaningful insights from financial data. The present work investigated the practicality of a deep learning CNN-LSTM model within the Directional Change framework to predict significant DC events—a task crucial for timely decision-making in financial markets. Furthermore, the potential of autoencoders and variational autoencoders to enhance financial forecasting accuracy and remove noise from financial time series data was explored. Leveraging their capacity within financial time series, these models offered promising avenues for improved data representation and subsequent forecasting. To further contribute to financial prediction capabilities, a deep multi-model was developed that harnessed the power of pre-trained computer vision models. This innovative approach aimed to predict the VVIX, utilising the cross-disciplinary synergy between computer vision and financial forecasting. By integrating knowledge from these domains, novel insights into the prediction of market volatility were provided.

Dedication

To my mom and dad, Zari and Mansour

My sisters Atash and Aseman

and my brother Hoormazd

for their endless love, support, and devotion.

Acknowledgements

I would like to express my gratitude and appreciation for my supervisor Dr. John O'Hara whose guidance and support have been invaluable throughout my Ph.D.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 1955 at Dartmouth College in Hanover, New Hampshire, John McCarthy, a maths professor, first coined the term artificial intelligence. Two years later, prominent economist Herbert Simon predicted that computers would beat humans at chess within ten years. Artificial Intelligence (AI) is a broad topic today, and its popularity is on the rise thanks to the emergence of innovative technologies. Artificial intelligence alludes to computers' ability to execute tasks associated with the characteristics of human beings, such as reasoning, discovering meaning, generalising, and learning from experience [58]. Human intelligence is not characterised by just one trait but rather by a combination of diverse abilities. With that said, AI research focuses mainly on the aspects of intelligence: learning, reasoning, problem-solving, perception and using language [45].

Improvements in perception and cognition are the most significant advances. Within the category of perception, practical advances have been made in the area related to speech. Millions of people worldwide use the voice recognition of Google Assistant, Alexa, and Siri. Image recognition, on the other hand, has come a long way. Facebook and many other apps now recognise faces in photos and prompt users to tag them with their names. Self-driving cars used to misidentify pedestrians once in 30 frames: now, less than once in 30 million frames. The error rate of recognising images from the ImageNet dataset with several million images decreased from above 30% in 2010 to nearly 4% in 2016 [138]. Another significant improvement has been in cognition and problem-solving. AI has beaten humans at poker and Go, yet another achievement experts had predicted would take another decade.

Deep Mind machine learning improved the cooling efficiencies of Google's data centres by 15% [101]. Intelligent agents are being used to prevent money laundering, and cyber-attacks and automate claims processes. Many companies use machine learning to execute trades, credit scoring, fraud detection, and quantitative trading. Machine learning systems are ubiquitous and not only surpass older algorithms in many applications but are now better at many tasks that were once done best by humans [16].

Machine learning puts forward a fundamentally different approach to problem-solving: learning from examples rather than being explicitly programmed to obtain a particular outcome. For many years, information technology has focused on embedding existing human knowledge into machines. The principal shortcoming of this approach is the tacit nature of our knowledge, in many ways, it is not explainable. We recognise our friend's face but writing the instruction to that is nearly impossible. This noteworthy fact, Polanyi's Paradox, explains the cognitive phenomenon that there exist tasks we understand how to perform intuitively, although we cannot verbalise their rules and procedures [36, 17].

We are overcoming that limitation by building machines capable of learning from examples and using feedback to solve problems such as recognising faces. There are different flavours of artificial intelligence and machine learning. Machine learning is a statistical approach to learning from data. It is an umbrella term with various algorithms to understand relationships within data and make predictions and decisions on that data. Three main branches of machine learning are supervised, unsupervised, and reinforcement learning. However, most of the successes have been achieved in supervised learning [111]. In supervised learning, a machine is fed with many examples of correct answers to a problem, aiming to map from a set of inputs to a set of outputs. The training process of such a model almost always contains thousands, if not millions, of examples, labelled with correct answers. Following the training, the model can predict with a high accuracy rate.

Much of the success in supervised learning results from deep learning developments that employ neural networks [116, 81]. Neural networks have proven to have the upper hand compared to conventional machine learning algorithms. Training models with larger training examples would improve machine learning and deep learning models' performance. However, in the case of machine learning models, accuracy plateaus at a point, i.e., additional training data, will not lead to more accurate predictions. Although data labelling is a resource-expensive endeavour, it is a relatively straightforward task; hence making supervised machine learning systems more prevalent. Regression and classification models are the two common types of supervised learning algorithms.

Unsupervised learning is another common type of machine learning. It differs from supervised learning because it aims to extract information from unlabelled datasets. Developing such models is extremely difficult; not surprising that Yann LeCun compared supervised learning to the icing on the cake and unsupervised learning to the cake itself. The main branches of unsupervised learning are clustering, dimensionality reduction, and generative learning.

Reinforcement learning, inspired by behavioural psychology, is another growing and exciting area within the field of machine learning in which no data is given to the model. The machine performs tasks using an environment and an agent that tries to navigate in this environment. The agent has a goal or a set of goals, while the environment has rewards to help the agent reach its goal [94]. Systems trained to play Atari video games, and the game of Go use reinforcement learning techniques. Deep learning, which derives from machine learning, is a broad class of approaches and algorithms that are used to achieve human-like intelligence. Deep learning implies that the approaches attempt to solve problems in a general way—like spatial reasoning, computer vision, and speech recognition. Deep learning also concerns itself with supervised learning, unsupervised learning, and reinforcement learning. Deep learning approaches typically use multiple layers of artificial neural networks.

The use cases for artificial intelligence are not limited. Where there are data and problems to solve, AI can be applied innovatively to real-world problems. These techniques have been used in various domains, such as health care, cybersecurity, logistics, and telecommunications. One specific domain is finance and financial markets. With the growth of machine-readable data throughout the financial system and computing power, financial modelling has more than ever been affected. The financial crises of 2007-2008 resulted in regulatory bodies adopting data-based regulations. The collection and analysis of bank loan contractual terms and trading book stress-testing programs in the US and Europe are among the prominent examples of such data-driven regulations [41].

1.1 Research Background

Financial Markets and Data

Contrary to the conjured-up image of the bustling New York Stock Exchange floor, financial markets broadly refer to a marketplace where the trading of financial products takes place. These markets come in many forms, functioning in different ways, and whether formal or informal, they serve essential functions that are fundamental to the operation of modern economies. Some of these crucial functions include price setting, asset valuation, capital raising, investment, and risk management.

Financial markets provide price discovery, which involves the process of determining the relative price at which individuals' expectations and valuations align. These market prices, driven by the forces of supply and demand, offer valuable insights into the perceived value of assets, allowing investors and market participants to make informed decisions. Corporations utilise various financial instruments such as equities, bonds, and other securities to attract new funding, earn returns, and accumulate assets that will generate income in the future.

Equities, commonly known as stocks, represent ownership stakes in a company, providing investors with a claim on the company's profits and assets. Bonds, on the other hand, are debt instruments issued by corporations or governments to raise capital. Bondholders receive periodic interest payments and the return of their principal amount upon maturity. These financial products offer diverse investment opportunities and cater to the preferences and risk appetites of a wide range of investors.

Financial markets play an essential role in facilitating economies by efficiently allocating resources and generating liquidity for companies. When businesses need to raise capital for expansion or investment in new projects, they can issue stocks or bonds to the public through the primary market. Investors, in turn, purchase these securities, providing the necessary funds for the businesses' growth initiatives. Moreover, secondary markets enable investors to buy and sell previously issued securities, adding liquidity and fostering continuous trading and price discovery.

These markets are vital for both established enterprises and start-ups, providing them with access to the necessary funds for growth and expansion. For example, established companies may issue additional shares through rights offerings or public offerings to finance mergers and acquisitions or research and development projects. Start-up companies, with innovative ideas but limited capital, often turn to venture capital or IPOs[1] to attract investors and gain access to the financial markets.

Furthermore, it is important to highlight that financial securities play a crucial role in the modern economy, offering an array of investment opportunities. This intricate system not only enables individuals to accrue returns on their investments but also plays a pivotal role in channelling vital funds to businesses that require capital infusion for growth and development. This symbiotic relationship between investors and businesses underscores the significance of financial securities in fostering economic prosperity.

Savvy investors adeptly leverage financial markets to assemble portfolios finely tuned to their risk tolerance levels and overarching financial goals. This strategic diversification across a spectrum of assets, ranging from stocks and bonds to commodities, serves as a vital mechanism for risk management. By dispersing investments across various avenues, investors can mitigate the potential downsides associated with market fluctuations while simultaneously enhancing the prospects for favourable returns. This prudent approach to portfolio construction epitomises the sophisticated understanding of investors who appreciate the delicate balance between risk and reward.

In addition to traditional investment, derivatives are an important avenue of investing within the financial landscape. These intricate financial instruments, including futures, options, and other

[1] Initial Public Offering

derivative contracts, assume a dual role as tools for risk mitigation and speculative endeavours. Derivatives offer investors an opportunity to not only safeguard their portfolios from the impact of adverse price movements but also to venture into informed speculations regarding future asset prices. This dual functionality emphasises the versatility and relevance of derivatives in a dynamic and ever-evolving market environment.

To illustrate the practicality of derivatives, consider the case of a farmer relying on futures contracts to secure a stable income amidst market volatility. By locking in a predetermined price for the sale of their crops, the farmer shields themselves from the uncertainties that could erode profitability. This real-world application exemplifies the inherent power of derivatives in promoting stability and predictability for market participants across diverse sectors.

In essence, financial markets play a pivotal role in the smooth operation of capitalist economies, serving as the backbone of resource allocation, liquidity generation, and asset trading. They enable businesses and entrepreneurs to thrive, foster economic growth, and encourage investments in various sectors, thereby driving the progress of the entire economy. By ensuring efficient capital allocation and risk management, financial markets contribute significantly to the overall stability and prosperity of nations and global financial systems.

Financial Data

Trading opportunities are significantly influenced by the data that uncovers them. The frequency of the data plays a pivotal role in the number of arbitrage opportunities that come to light. In the pursuit of identifying profitable prospects, it is crucial to utilise data that offers the highest level of detail. Recent developments in micro-structure analysis and advancements in econometric modelling have fostered a common understanding of the unique attributes of tick data. Unlike regularly spaced low-frequency data, tick data comes in irregular, short intervals. These observed irregularities grant researchers and traders access to a trove of insights not accessible within low-frequency datasets. The time gaps between trades could potentially indicate shifts in market volatility, liquidity, and other pertinent factors. Moreover, the substantial data volume empowers researchers to generate statistically accurate conclusions.

Emphasised by Gençay et al. [42], larger datasets contain significantly wider ranges of input variables due to the expanded number of permissible degrees of freedom. The abundant amounts of tick data provide researchers with the opportunity to draw statistically meaningful conclusions about recent market changes using brief data snapshots. While a monthly collection of daily data might typically be considered too limited to yield statistically reliable forecasts, substantial volumes of tick data within the same monthly period can render such short-term predictions feasible.

However, it's important to consider frequency-related factors like intra-day seasonality when determining the necessary quantity of observations for assessment.

In a nutshell, finding good trading chances relies on having the right information. Getting data often and in detail is like having a treasure map. And recent improvements in understanding that kind of data, the tick data, help us read the map better. It's like solving a mystery where every little clue counts. The more clues we have, the better we can solve the puzzle of the financial markets.

Tick Data

Tick data, the highest-frequency data [3], is a collection of sequential ticks, the latest quote, trade, price, and volume information and usually with the following properties:

- **A timestamp**: This shows the exact time and date when the data was recorded.
- **A financial security identification code.**
- **An indicator of what information tick carries**:
  - Bid price: The highest price someone is willing to pay for the security.
  - Ask price: The lowest price at which someone is willing to sell the security.
  - Available bid volume: The amount of the security that people want to buy at the bid price.
  - Available ask volume: The amount of the security that people want to sell at the ask price.
  - Last trade price: The price at which the most recent trade took place.
  - Last trade size: The amount of the security traded in the most recent trade.
  - Option-specific data, such as implied volatility.
- **Information about the market value, such as the actual numerical value of the price, available volume, or size of the security.**

The timestamp signifies when a quote originated, either when released by the exchange, broker-dealer, or received by the trading system. This travel time can be as short as 20 milliseconds. Complex systems incorporate milliseconds for precision. Quoted data includes an identifier for financial security, often a ticker symbol or ticker with an exchange symbol for multi-exchange tickers. For futures, the identifier encompasses the underlying security, futures expiration date, and exchange code. The last trade price reflects the price of the latest cleared trade, which can differ from bid and ask prices. Differences arise when a customer's favourable limit order is instantly matched by the broker without broadcasting the customer's quote. The last trade size indicates the size of this executed trade. In the market, the bid quote is the highest available sale price, while the ask quote is the lowest buying price at a given time. Market participants provide these through limit orders. The highest bid from a yet-to-be-executed buy order and the lowest ask among sell orders form the market bid and ask. Available bid and ask volumes reveal total demand and supply at those prices.

High-frequency data is characterised by its large volume, with each tick generating a significant amount of daily observations. This data's sheer quantity doesn't necessarily equate to quality. Centralised exchanges provide accurate bid, ask, and volume data along with timely timestamps for transactions. However, detailed limit order book information is less accessible. In decentralised markets like foreign exchange and inter-bank money markets, market-wide rates are often unavailable. Participants in such markets are aware of the current price level, but individual institutions set prices based on purchase orders. Consequently, specific rates for financial instruments may differ among traders. To enhance the efficiency of decentralised markets, entities like Reuters, Telerate, and Knight-rider gather and distribute quotes from various traders. There are commonly perceived anomalies in price differences between traders.

Trader quotes often mirror their own inventory. For instance, a dealer who recently sold $100 million of USDCAD might seek to diversify risk and avoid further sales in USDCAD. Dealers typically must transact based on tradable quotes. To prompt clients to place sell orders on USDCAD, a dealer raises the bid quote temporarily. Simultaneously, the dealer increases the ask quote to discourage clients from placing buy orders. Consequently, dealers adjust bid and ask prices when they are short or long in a specific financial instrument. In anonymous marketplaces like dark pools, dealers and market makers might use indicative quotes to gauge demand or supply, often deviating significantly from previously quoted prices.

As observed by Gençay et al. [42], some trader quotes might lag behind actual market prices, with delays varying from milliseconds to one minute. Certain traders quote moving averages of others' quotes. Delayed quotes are sometimes used to enhance market presence in data feeds, particularly relevant when contracts are negotiated over the phone. However, the fast-paced electronic market discourages such delays, improving overall market quality.

The difference between the greatest price a buyer is willing to pay and the lowest price a seller is ready to accept, or the bid-ask spread (representing the cost of immediate buying and selling), is determined by several important elements in the financial markets. A key factor is liquidity; assets with higher levels of liquidity usually have smaller spreads since there are more buyers and sellers actively engaged in the market. Because assets with more frequent transactions typically have lower spreads, this is strongly related to trading volume. Another important consideration is volatility, with more volatile assets typically having larger spreads to offset the higher risk. The spread is also influenced by information availability and market transparency, since better information availability results in more aligned buying and selling prices. Another important factor is the degree of competition among market makers; higher levels of competition typically translate into narrower spreads. The spread may also be impacted by the asset's intrinsic qualities, such as the size and stability of the business in the case of equities. The bid-ask spread can also fluctuate due to outside variables such as the time of day and general market conditions, which might include times of economic duress. Lastly, by improving market efficiency, operational and technological developments, such as the switch to electronic trading, have typically served to lower spreads. A higher bid-ask spread necessitates greater security profit to cover the spread and other transaction costs. Bid-ask spreads are typically negligible during infrequent price changes. Conversely, with tick data, incremental price changes are smaller and equivalent to the bid squeeze spread.

Financial markets constitute a high-dimensional, complex, and noisy ecosystem where multiple factors interact to determine asset prices and investment outcomes. Recent technological advances have led to the creation of vast amounts of financial data. According to Jabbour et al. [69], nearly three billion bytes of data are generated daily from sensors, mobile devices, social networks, and online transactions. In the ever-changing financial landscape, accurate predictions of financial market movements enable participants to make informed trading and investment decisions. However, due to the large scale of today's financial data, innovative approaches and methodologies that better reflect the dynamic nature of financial markets are required.

Traditional forecasting models often rely on statistical approaches [53, 66, 5, 73]. The shortcomings of these conventional models lie in their unrealistic assumptions, which do not accurately capture the non-linear nature of financial data. Given the promising results of computer vision and Natural Language Processing algorithms in various fields, further investigation is needed to adapt these innovations to financial data. The pivotal role of the financial industry in global economies has given rise to various forecasting techniques ranging from time series models to machine learning algorithms [117, 91]. However, the complexity and dynamism of financial markets necessitate the use of highly sophisticated methodologies capable of effectively modelling nonlinear trends, sudden shifts, and intricate inter-dependencies among various variables.

1.2 Research Motivation and Objectives

Primarily, the objective of this research is to investigate the potential of deep learning models in improving financial forecasting accuracy. Specifically, this study aims to achieve the following objectives:

- **Examining the Suitability of Deep Learning Models in Directional Change Framework**: Explore the utilisation of deep learning models within the Directional Change Framework using high-frequency financial data. This will enable the model to capture rapid shifts in market direction, a crucial aspect for timely decision-making.
- **Evaluating Generative Models in Financial Forecasting**: Evaluate the suitability of deep generative models, i.e., Autoencoders and Variational Autoencoders, in forecasting financial time-series data.
- **Leveraging Computer Vision Models**: Analyse the potential of pre-trained computer vision models in financial prediction. Investigate how these models, originally designed for image analysis, can be adapted to extract valuable insights from financial data.

The following research questions guide this study:

1. **Directional Change Framework Performance**: How does a deep learning CNN-LSTM model perform within the Directional Change Framework for forecasting events, considering the inherent volatility of high-frequency financial data?
2. **Autoencoder and Variational Autoencoder Contribution**: Can an Autoencoder and a Variational Autoencoder be employed for financial time-series forecasting? If so, how would they perform compared to recurrent and convolutional models?
3. **Computer Vision Adaptation**: How can pre-trained computer vision models be harnessed for financial prediction?

The potential for transformation of decision-making processes within the financial industry through the utilisation of deep learning models in financial forecasting is significant. The provision of precise predictions, particularly during periods of instability, can result in the enhancement of risk management, investment strategies, and the overall stability of financial systems. The present research contributes to the progress of predictive analytics within the financial domain by addressing the shortcomings of conventional forecasting models.

The present thesis aims to address the research problem of enhancing the accuracy and adaptability of forecasting methods in the domain of finance. The conventional techniques of forecasting may encounter difficulties in comprehending the intricacies of financial data, which can ultimately result in sub-optimal predictions, particularly during times of heightened volatility and uncertainty. As the financial markets persist in their evolution, there is an escalating requirement for models that can proficiently adjust to changing conditions and augment forecasting accuracy.

1.3 Publications

In our published work in the Journal for Neural Computing and Applications [104], we investigated the effectiveness of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) within the Directional Change Framework. The main objective was to evaluate the predictive ability of the CNN-LSTM model in forecasting directional change event prices. To prepare the input data, we gathered tick prices for the currency pairs GBPUSD, EURUSD, USDCHF, and USDCAD, covering the period from January to August 2019.

Subsequently, we derived tick bars/candles from the collected tick prices for each currency pair. In addition to this, we generated DC-based summaries of the tick bars/candles. The summaries provided input data for the CNN-LSTM model, which effectively combines the feature extraction capabilities of CNN with the sequential data prediction capabilities of LSTM. To assess its performance, the CNN-LSTM model was evaluated and compared to alternative regression models, including Support Vector Regression and Random Forest Regression. The results demonstrated that the CNN-LSTM model exhibited superior performance compared to other regression models within the DC framework, thereby highlighting its predictive abilities in forecasting directional change event prices.

1.4 Roadmap

In the upcoming chapters, this study embarks on an exploration of deep learning's application in financial analysis and prediction. Chapter 2 gives the relevant literature review, summarising the key research findings and insights from previous studies in financial forecasting. Chapter 3 introduces the investigation of a CNN-LSTM model within the Directional Change framework for event prediction in high-frequency financial data, emphasising the use of Average True Range (ATR) as a threshold indicator. Chapter 4 delves into the challenges of financial market analysis, distinguishing between traditional statistical models and machine learning methods, particularly autoencoders, for denoising and forecasting financial data. Chapter 5 shifts the focus to volatility forecasting, particularly the VVIX index, and explores innovative techniques like Gramian Angular Fields and Markov Transition Fields alongside a hybrid ResNet-LSTM architecture to enhance accuracy. Finally, Chapter 6 concludes with a summary of the findings, contribution and significance, and the prospects of future research.

Chapter 2

Background and Literature Review

This chapter encompasses relevant background information. It commences with an overview of the utilisation of machine learning and deep learning within the field of finance. This succeeded by an introduction to the concept of financial times series forecasting and the subsequent introduction of the Directional Change framework. The narrative then proceeds to delve into the intricacies of Long Short-term Memory and Convolutional Networks, followed by a discourse on generative modelling and residual networks. Finally, it navigates through topics such as transfer learning, Gramian Angular Fields, Markov Chain and Markov Transition Fields.

2.1 Machine Learning and Deep Learning in Finance

Over the past two decades, the rapid evolution of information and communications technology has ushered in an era defined by an overwhelming abundance of collected data, laying the foundation for what is now commonly referred to as the era of Big Data [90]. This transformative landscape has naturally propelled machine learning and data science to the forefront of modern innovation. The surge in available Big Data and the increased affordability of computing power have synergistically catalysed the ascendance of machine learning and data science in recent times.

Currently, a sweeping trend can be observed across the financial sector, encompassing entities ranging from hedge funds and investment banks to retail banks and fintech companies, all of which are fervently embracing and heavily investing in machine learning and data science technologies. Below, a selection of the myriad applications of machine learning within the financial domain is elucidated.

Elevating Portfolio Management and the Emergence of Robo-Advisors

Portfolio management, a careful process involving choosing and keeping an eye on a mix of investments that match specific risk and return goals over a certain time period, has gotten a boost from the rise of artificial intelligence (AI). One significant outcome of AI in this area is the emergence of robo-advisors [49]. These are like real-time digital platforms that offer automated investment management based on maths and algorithms. They also provide clients with various financial services, adding a new layer to how portfolios are managed.

In this changing landscape, robo-advisors showcase how technology and finance can work together. By using the precision and efficiency of machine-guided decisions, robo-advisors combine investment oversight with financial strategies. This not only improves how investments are managed but also offers clients a range of financial services. This blend of technology and finance reshapes portfolio management, making it more precise and engaging for clients. This mix of technology and finance opens the door to a new era, changing how investment management happens [56].

Mitigating Fraud and Unveiling Money Laundering Patterns

The detection and prevention of fraudulent transactions or activities have been significantly strengthened by incorporating machine learning. Machine learning's inherent ability to analyse large datasets to uncover unusual patterns makes it especially effective in this context [107]. Another considerable challenge within the financial sector is the identification of indicators of money laundering. Money laundering involves hiding illegally obtained funds, and machine learning models are adept at examining transactional data to uncover signs of such illicit activities [99].

Enhancing Loans, Credit Card Services, and Insurance Underwriting

Underwriting processes can be significantly enhanced through the utilisation of machine learning, leading to streamlined risk assessment and more informed decision-making. Machine learning models possess the capability to analyse an array of data sources to evaluate clients' risk profiles systematically. This technological approach results in the provision of more precise pricing estimates, expedited approval timelines, and a reduction in operational expenditures [13, 88].

Revolutionising Risk Management Paradigms

The risk management field, these days, is undergoing a significant transformation, driven by the prevalent adoption of machine learning techniques. This domain encompasses a broad set of activities, including the identification of the most suitable lending limits for specific customers, the enhancement of regulatory compliance procedures, and the reduction of risks associated with models. Yu et al. [137] proposed a measure for systematic risk, known as the Financial Risk Meter (FRM) based on the penalisation parameter of a linear quantile lasso regression. It is calculated by taking the average of the penalisation parameters over the 100 largest US publicly traded financial institutions. In another research, Swankie and Broby [120] examined the relationship between Artificial Intelligence and banking risk management and concluded that the use of AI can add significant economic value to banking operations involving credit, operational, and liquidity risk.

Pioneering Sentiment Analysis in Finance

Financial Sentiment Analysis emerges as a dynamic field that delves into the intricate interplay between diverse financial resources and their profound influence on investment dynamics. At its core, it seeks to augment forecasting precision by harnessing the potency of deep learning methodologies [32]. This domain finds its roots in the recognition that the sentiments encapsulated within an array of sources, ranging from news articles to social media posts, wield tangible impact on stock valuations and the reputations of companies.

The growing impact of emotions has increased the importance of financial sentiment analysis for investors. By examining the emotional tone in news articles and tweets, investors gain a special advantage in predicting and understanding market shifts more accurately. As a result, researchers have tried using sentiment analysis on financial news to predict various financial events. This includes foreseeing changes in stock prices, predicting how companies will perform, and recognising patterns in both the Forex market and the wider global financial landscape [31, 27, 29, 93].

Essentially, financial sentiment analysis elucidates the intricate ties binding sentiments to financial outcomes, thereby furnishing investors with a dynamic toolset to navigate the complex and rapidly evolving landscape of investments. This research frontier continues to illuminate new pathways for the augmentation of forecasting accuracy, fostering a symbiotic interplay between sentiment analysis and deep learning methodologies.

Predicting Asset Prices with Machine Learning

The field of asset pricing revolves around the effort to figure out the real value of financial assets. Whether you're trading stocks, bonds, derivatives, or other financial tools, understanding how prices change is crucial. While various ways are used to value different assets, the main goal is always to balance risk and expected profits. One popular tool for this is the Capital Asset Pricing Model (CAPM), which is widely used to compare how risky an asset is and what returns it could bring compared to the overall market portfolio. Chen et al. [22] employed deep neural networks to estimate an asset pricing model for individual stock returns. They included that deep neural networks outperform benchmark approaches in asset pricing. Gu et al. [52] demonstrated the potential of machine learning in enhancing empirical asset pricing and identified the best-performing approaches.

Revitalising Derivative Pricing Strategies

A derivative is a financial product whose value is connected to one or more underlying assets. The process of determining the accurate value of a derivative is known as derivative pricing. Various types of derivatives have distinct methods for calculating their values. These financial agreements derive their worth from underlying assets, such as stocks, bonds, or commodities. The objective of derivative pricing is to establish an unbiased measurement of a derivative's actual value. This calculation varies depending on the specific derivative. For instance, futures contracts are valued by considering arbitrage opportunities, options are valued by using risk-neutral probabilities, and swaps' valuation relies on present values.

Traditional models for derivative pricing are constructed based on specific assumptions to replicate the observed connections between factors like strike prices, time until maturity, option type, and the market price of the derivative. The integration of machine learning techniques introduces novel possibilities within the field of derivative pricing. This encompasses various applications, such as creating datasets that incorporate product attributes, market trends, and model parameters to estimate model-based values. Moreover, machine learning aids in calibrating asset price models to real-world market data through simulations and regression analyses [136].

Furthermore, machine learning's potential extends to reducing the computational time needed to price complex derivatives and American options, employing advanced tree-based methodologies [118]. Lastly, machine learning facilitates the comparative evaluation of different algorithms for option pricing [68]. The advent of AI and machine learning in the domain of option pricing heralds a new era of precision and efficiency. These technologies, by harnessing advanced data analytics and pattern recognition capabilities, can dissect and interpret vast, multifaceted market data sets, revealing critical insights into market dynamics and future trends.

This is especially pivotal in options pricing, where understanding future volatility and price movements of underlying assets is crucial. AI/ML algorithms enhance traditional pricing models like Black-Scholes, adapting them to better reflect real-world market conditions and irregularities, such as changing volatility patterns and the unique features of different types of options, including American options with their early exercise characteristics [14, 28]. In risk management, AI/ML offers a more nuanced approach, providing deeper insights into the probability of diverse market scenarios, thus enabling more effective strategies for pricing and hedging options. For high-frequency trading, the speed and efficiency of AI/ML algorithms are unmatched, allowing for rapid analysis and execution of trades, capitalising on fleeting market opportunities. Beyond trading, these technologies streamline compliance and reporting processes in alignment with financial regulations, ensuring accuracy and reducing operational costs. In essence, machine learning offers a versatile toolkit for derivative pricing, encompassing the development of data-driven value estimates, efficient computation strategies, and the assessment of algorithmic performance.

Pioneering Algorithmic Trading Frontiers

Algorithmic trading uses complex algorithms to carry out trades autonomously, according to predefined directives for quick and objective decision-making. The incorporation of machine learning (ML) has greatly improved this practice, pushing algorithmic trading into more complex domains [26]. The implementation of increasingly complex and dynamic trading strategies is made possible by machine learning's ability to adapt and use data in real-time. Additionally, machine learning (ML) techniques open up a wider range of possibilities for deriving in-depth understandings of market dynamics, such as pattern recognition, predictive analytics, and adaptability to shifting market conditions.

The ability of ML algorithms to process and analyse massive amounts of data—such as historical price trends, news feeds, and social media sentiment—at unprecedented speeds is one way in which this advancement is most noticeable. With the help of these features, algorithms can anticipate future market trends and enhance risk management plans, which increases the efficacy and efficiency of trading decisions [18]. These technologies are essential for improving real-time trading tactics, even though many hedge funds and financial organisations keep their specific machine learning algorithms proprietary. They help create a more complex trading environment where judgements are made more quickly, based on more data, and less susceptible to biases and human mistakes.

It's crucial to understand, though, that these sophisticated algorithmic trading strategies have a unique set of drawbacks, such as the potential for over-fitting models to historical data and the requirement for constant regulatory standard adaption. Notwithstanding these difficulties, algorithmic trading's incorporation of machine learning marks a substantial advancement in the industry and gives players in the financial markets new options.

2.2 Time Series and Forecasting

In many domains of science, variables are measured sequentially over time and finance is not an exception. A series of data points that occur in successive order over a period. Historical or current price of listed stock, quarterly or annual earnings of a company, daily sales of a product or service and the monthly unemployment rate of a country are among some of the time series data. The principal characteristics of many time series are trends and seasonal variations which can be modelled deterministically with the mathematical functions of time. Another important characteristic is that close observations in time tend to be serially correlated. Time series forecasting is a method to predict future events or values based on time series data. Time series forecasting can help with decision-making, planning, and strategy in various fields.

Changes in the price of the financial assets are at their core, the result of market participants' perspective about the future. Biased and erroneous processes of human decision-making along with other macro and micro factors cause a large amount of noise in the form of new information in financial time series. In order to analyse financial time series, prices are recorded by sampling data points at fixed time intervals (Daily, weekly, monthly). In this method, researchers first, decide how often to sample the data, and then they take snapshots at the chosen frequency.

As a result of this convention, financial time series are unevenly spaced and discontinuous concerning the flow of physical time [43] due to several reasons:

- **Market Closure**: The financial markets are not always open. Usually, they are closed on weekends and holidays and only open during predetermined hours each day. For example, no data points are obtained while the market is closed when data is sampled daily. The time series has gaps over weekends and holidays as a result [64].
- **Non-Uniform Trading Activity**: Even within trading hours, trading volume can fluctuate significantly. Some trading hours or days can see an increase in trading volume due to economic news, market news or other factors. However, this fluctuation in trading volume within the interval will not be captured when sampling at fixed intervals. For example, at the end of the day.
- **Time Zone Differences and Global Markets**: Financial assets are traded in several markets across the globe, each operating in a different time zone. A daily sample in one time zone might not match a trading day in another, resulting in differences in how the data portrays market activity across different regions.
- **Irregular Events**: Events like stock splits, dividend payments, and earnings reports can cause big price movements. These events don't happen regularly and can introduce noise into a

time series that's been sampled at fixed intervals.

- **Discrete Nature of Sampling**: The sampling process itself, in which data points are measured at different time intervals, results in an inconsistent time representation. The constant flow of time, and the constant change in prices within each time interval, are compressed into a single point of data, resulting in a lack of information about price movements within a period.
- **Heteroscedasticity**: Heteroscedasticity is one of the most common characteristics of financial time series. This means that the variability of returns can vary over time. Fixed-interval sampling, however, may not capture these variations, particularly during periods of high or low market volatility.

Therefore, the interval-based summary of the price lacks realism since, in everyday life, history is recorded by identifying key events. To address the aforementioned shortcomings of the traditional approach of time series analysis, Guillaume et al. [54] proposed a new time scale. Intrinsic time is an alternative approach that replaces the notion of "physical time scale" and looks beyond the constraints of the physical time within financial data and constitutes an event-driven approach.

2.3 Deep Learning Applications in Financial Forecasting

Financial forecasting has always been an exciting research area in the financial industry. Many studies have been published on machine learning models with relatively better performances than classical time series forecasting techniques [113, 129, 38, 86, 123]. Artificial Neural Networks (ANNs), a sub-class of machine learning models are widely used for predictive data mining tasks. The applicability of artificial neural networks to stock market predictions was first hypothesised by White [133], with some indications of success by Saad et al. [106]. Artificial Neural Networks, in essence, mimic the structure of biological neural networks where neurons are interconnected and learn from experience. The architecture of ANNs consists of nodes or neurons, arranged in layers that are interconnected through a system of weights.

Various architectures of neural networks have been developed, and the most popular type that has been used to solve most of the problems is feed-forward neural networks. Such networks include an input layer, one or more hidden layers and an output layer [15]. Researchers explored non-linear neural networks and Support Vector Machines for time series prediction [85]. Zbikowski [139] enhanced classifier accuracy using Volume-Weighted Support Vector Machine for stock trading, while Choudhury et al. [25] used k-means and Support Vector Regression for market volatility and price prediction. Zhang [140] used neural networks and ARIMA for stock forecasting, highlighting the advantage of neural networks in non-linear prediction. Abu Hammad and Hall [1] explored the Jordanian stock market with multi-layer BP networks. Zhang et al. [141] proposed an LM-BP neural network for stock forecasting, and Wang et al. [130] introduced a wavelet neural network. Roondiwala et al. [103] achieved a root mean squared error of 0.0086 using LSTM for Nifty Index prediction.

A significant characteristic of feed-forward neural networks is that they have no memory meaning that each input is processed independently, with no state kept in between inputs. With such networks, a sequence or a temporal series of data points is treated as a single data point and processed in one go. Recurrent Neural Networks (RNN) [109], which are a powerful type of artificial neural network, process sequences by iterating through the sequence elements and maintaining a state containing information relative to previous states. One limitation of RNNs is a problem called vanishing/exploding gradient, which causes the model learning to slow down or stop altogether.

2.4 Directional Change Framework

Prediction of financial asset prices has long been a captivating pursuit for researchers and market analysts. Unlocking the underlying patterns and dynamics in an effort to gain a competitive edge in trading has led to extensive investigations into forecasting methodologies. However, this pursuit is far from trivial due to the intricate nature of financial markets. Financial markets present a distinct amalgamation of intricacies that differentiate them from conventional predictive domains. The inherent volatility of markets, which is dictated by a plethora of factors such as economic indicators, geopolitical events, investor sentiment, and even social media trends, engenders a constantly evolving landscape. Additionally, the non-linear correlation between events and price fluctuations poses a challenge to the efficacy of traditional linear predictive models [43].

Time-series analysis lies at the core of this endeavour to understand and forecast financial data [54]. Researchers have conventionally resorted to sampling price data at fixed intervals, such as daily, weekly, or monthly to uncover hidden trends and patterns hidden within the price fluctuations [43]. However, this approach carries inherent limitations that can hinder the accuracy of predictions. Fixed-interval sampling, while providing a structured representation of market data, inadvertently creates gaps in the timeline. These gaps can omit crucial events that occur between the sampling points. Imagine a scenario where a significant market-moving announcement is made just hours after the daily data has been captured. This announcement, which might drastically alter the trajectory of prices, would go unnoticed until the next sampling point, potentially leading to missed trading opportunities and flawed predictions.

Moreover, the interconnectedness of modern financial markets renders the concept of isolated, linear intervals inadequate. A market event occurring in one corner of the globe during non-trading hours can trigger a domino effect that influences prices in markets that are actively trading. This ripple effect transcends the confines of fixed intervals and necessitates an approach that acknowledges the interplay of events across both temporal and spatial dimensions [43].

In financial analysis, addressing these challenges requires a transformative paradigm shift that ushers in a new era of comprehending market dynamics. This shift is exemplified by the Directional Change (DC) framework, a novel approach that disrupts the conventions of traditional time-series analysis [54]. Departing from fixed intervals, the DC framework adopts an event-driven perspective to pinpoint significant shifts in market conditions, providing a more dynamic and precise portrayal of market trends.

Aiming to confront these challenges, a pioneering solution known as Directional Changes (DC) emerged [54]. This method redefines the concept of "physical time scale" by embracing an event-driven approach that transcends the temporal constraints of financial data. Instead of being confined to predetermined time intervals, it adopts a viewpoint centred around market events. With the surge in machine learning's prominence, researchers have harnessed various algorithms for predicting financial asset prices and movements [110]. Mehtab's work [92] stands as a testament, showcasing accurate CNN-based regression models tailored for predicting multivariate financial time series. Building upon this momentum, the present study introduces a deep learning-based regression model designed to predict currency pair prices within the dynamic Foreign Exchange (FX) market. Leveraging the innovative Directional Change framework, this model's performance is rigorously evaluated, both within the framework's scope and beyond.

The directional Change (DC) is an approach to summarise price movement by transforming a time series price curve into an intrinsic time curve [124]. Under the DC framework, a DC event is identified by a substantial change in the price of an asset, defined as a price change greater than a pre-defined threshold value θ. Following a DC event, an overshoot (OS) event happens until the next DC event in the opposite direction. Figure 2.1 illustrates a time series and the corresponding intrinsic time series for a $\theta = 0.05\%$. Based on the DC approach, the market is broken down into an alternating uptrend and downtrend. An upturn event indicates that the price change between the current market price p_t and the last low price p_l is greater than a threshold θ:

$$p_t \geq p_l(1+\theta) \tag{2.1}$$

As illustrated in Figure 2.1, the move from point A to B is an upturn DC event. By the same token, a downturn event is defined as an event where the difference between the current price p_t and the last high price p_h is lower than a fixed threshold θ [124]:

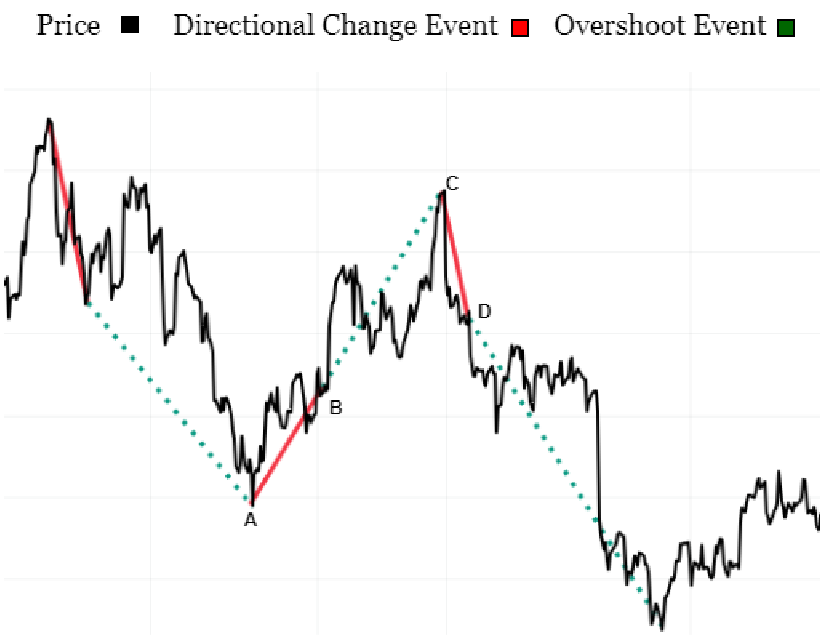


Figure 2.1: A share price and the corresponding intrinsic time curve for θ = 0.05%, for a selected time period

$$p_t \leq p_h(1-\theta) \tag{2.2}$$

A trend ends whenever a price change of the same threshold θ is observed in the opposite direction, see [7]. It should be noted that different thresholds generate different series of events. The notion of using different thresholds is that each threshold might be considered significant by a different trader. Smaller thresholds create more directional changes compared to larger ones. As it was mentioned above the value of the threshold needs to be predetermined when summarising price movements using the DC. It represents how big of a price change the observer considers significant.

Bakhach et al. [6] proposed a forecasting model, under the DC context, which aims to answer the question of whether the current trend will continue until a specific magnitude, expressed as a percentage, is reached before the trend ends. Golub et al. [46] suggested a novel approach to quantify FX market liquidity based on the DC framework. Their new approach seeks to model market dynamics to predict stress in financial markets. Few studies sought to develop trading strategies based on the DC framework.

In 2016, Bakhach et al. [7] introduced a DC-based trading strategy named "DBA". The so-called DBA strategy executes a trade when the magnitude of price change, during the OS event, reaches a particular threshold. DBA closes the position at the DC confirmation point of the next DC event. They applied DBA to three currency pairs. Experimental results showed that DBA earns enough return to compensate for the risk it took over the trading period. The results also showed that

Algorithm 1: Defining directional-change (DC) and overshoot (OS) events.

```
Result: DC-based event summaries
Initialisation: (event=upturn, $p_h = p_l = p_0$, $\theta \geq 0$, $t_0^{dc} = t_1^{dc} = t_0^{os} = t_1^{os} = t_0$);
if event is upturn then
    if $p_t \leq p_h * (1-\theta)$ then
        $event \leftarrow$ Downturn Event
        $p_l \leftarrow p_t$
        $t_1^{dc} \leftarrow t$ ;                        // Downturn event end time
        $t_0^{os} \leftarrow t+1$ ;                  // Downward OS event start time
    else
        if $p_h < p_t$ then
            $p_h \leftarrow p_t$
            $t_0^{dc} \leftarrow t$ ;                  // Downturn event start time
            $t_1^{os} \leftarrow t-1$ ;              // Downward OS event end time
        end
    end
else
    if $p_t \geq p_l * (1+\theta)$ then
        $event \leftarrow$ Upturn Event
        $p_h \leftarrow p_t$
        $t_1^{dc} \leftarrow t$ ;                          // Upturn event end time
        $t_0^{os} \leftarrow t+1$ ;                  // Upward OS event start time
    else
        if $p_l > p_t$ then
            $p_l \leftarrow p_t$
            $t_0^{dc} \leftarrow t$ ;                    // Upturn event start time
            $t_1^{os} \leftarrow t-1$ ;              // Downward OS event end time
        end
    end
end
```

DBA could generate positive returns of up to 14% within seven months, after deducting the bid-ask spread.

Kampouridis and Otero [72] proposed a trading strategy based on the DC framework, named “DC+GA”. It runs multiple DC summaries and thresholds simultaneously. For each DC summary, DC+GA tracks the identification of corresponding DC or OS events. It utilises these DC and OS events collectively with some trading parameters to implement buy or sell recommendations. DC+GA employs a Genetic Algorithm (GA) module to optimise the selection of thresholds and the values of the trading parameters of each threshold. This optimisation seeks to maximise the profits made, by DC+GA.

2.5 Long Short-Term Memory

Recurrent Neural Networks (RNN), are a robust type of artificial neural network which processes sequences by iterating through the sequence elements and maintaining a state containing information relative to previous states. Unlike the Feed-Forward neural networks, RNNs models can leverage the previous inputs’ sequential information through memory gates. The RNNs memory, which is called recurrent hidden state, enables the network to predict the next item in the input data sequence. Practically, however, the length of the sequential information is limited to only a few steps back. Although RNNs should theoretically retain information from previous time steps, such long-term dependencies are impossible to learn in practice.

A common problem among RNNs is vanishing gradient when the gradients' information vanishes while passing through a deep layered network. The gradient is the partial derivative of a function's output with respect to its inputs' changes. This problem prevents the network from learning long-term dependencies which causes the learning process to slow down or stop altogether. Conversely, there is the exploding gradient problem in which the gradient's information accumulates and results in a large gradient. In the "vanishing gradient" problem, the network assigns smaller values to the weight matrix, and in the "exploding gradient" problem, the opposite is true.

As mentioned earlier, RNNs are not capable of learning long-term dependencies. Long short-term memory architecture, was first proposed by Hochreiter and Schmidhuber [63] to address this problem. The LSTM models are an extension of RNNs and are designed to address the vanishing gradient problem. Generally, the LSTM model consists of three gates: forget, input, and output gates, as shown in Figure 2.2. The forget gate is responsible for deciding to preserve or remove the existing information. The input gate determines the extent to which the new information will be added to the memory, and the output gate controls whether the current value in the cell contributes to the output [63].

- **Forget Gate**: In the forget gate block of the LSTM layer, the information from the current input x_t and the previous hidden state h_{t-1} is passed through an activation function (e.g. sigmoid). The gate output f_t will be a value between 0 and 1, where zero implies removing the learned value while one means preserving the value. The output is computed as:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.3}$$

  where b_f is called the bias value.

- **Input Gate**: This gate which determines the additions of new information to the LSTM memory has two layers. A sigmoid layer decides which values need to be updated and the hyperbolic tangent layer generates a vector of new values that will be added to the memory. The output value of the input gate is computed through the following formulas:

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2.4}$$

$$\tilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2.5}$$

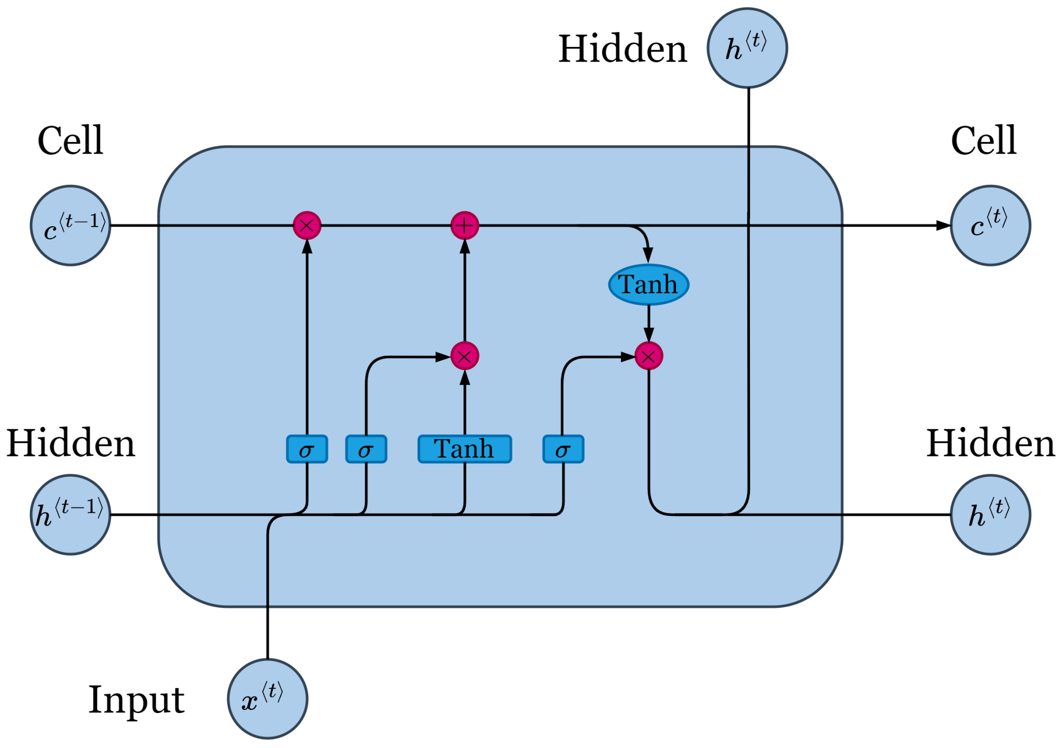


Figure 2.2: LSTM Block Architecture

Together, these two layers update the LSTM memory, forgetting the current value by multiplying the old value and adding a new value $i_t * \tilde{C}_t$. The following represents its equation:

$$\tilde{C}_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.6}$$

- **Output Gate**: Here the gate first uses a sigmoid function to determine which part of the LSTM memory contributes to the output. Subsequently, through the non-linear *tanh* function, it maps the values between -1 and 1.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.7}$$

$$h_t = o_t * tanh(C_t) \tag{2.8}$$

LSTM networks have also been used in the financial forecasting field. DiPersio and Honchar [37] compared three different RNN models (basic RNN, LSTM, and GRU) to predict the movement of Google stock prices. Hansson [57] used LSTM alongside other classical forecasting techniques to predict the trend of stock prices. Karmiani et al. [75] compared LSTM's performance to SVM, backpropagation, and Kalman filter, showing high accuracy and low variance. Fischer and Krauss [40] demonstrated LSTM's superiority in predicting S&P500. Nelson et al. [95] combined LSTM with technical indicators for stock movement prediction. Salis et al. [108] investigated LSTM's application in predicting gold price fluctuations. Zhuge et al. [142] predicted opening stock prices using LSTM. Hu [65] used CNN for time series prediction, showing moderate accuracy. Sezer and Ozbayoglu [112] classified daily prices of Dow 30[1] stocks and ETFs[2] using CNN.

2.6 Convolutional Neural Networks

Image data whether monochromatic or coloured is represented as a two-dimensional grid of pixels, where each pixel corresponds to one or multiple numerical values. The two-dimensionality of images is overlooked when images are treated as vectors of numbers, irrespective of the spatial relations of pixels. Flattening the images is a required procedure to prepare the data for a one-dimensional fully connected network. As these networks are insensitive to the arrangement of features, the outcomes remain consistent regardless of whether we maintain the order reflecting the spatial organisation of pixels or if we rearrange the columns in the design matrix before adjusting the parameters of the Multi-layer Perceptron (MLP) network. Ideally, we would exploit our prior understanding that neighbouring pixels often have a connection with each other to construct effective models for learning from image data. Convolutional Neural Networks [84] are powerful neural networks designed particularly to serve this purpose and are now ubiquitous in the computer vision field.

Convolutional Neural Networks have been shaped by drawing inspiration from various sources, including biology, group theory, and extensive experimentation. Apart from their effectiveness in producing accurate models with fewer samples, CNNs also demonstrate computational efficiency due to their reduced parameter requirements and the ease of parallelising convolutions across GPU cores [24]. As a result, practitioners widely employ CNNs whenever possible, and they have proven to be formidable contenders even in tasks with one-dimensional sequences like text [71] and time series analysis where recurrent neural networks have traditionally been utilised. Furthermore, clever adaptations of CNNs have enabled their application in handling graph-structured data [80] and recommender systems. As the name of the network implies, it uses a mathematical operation called convolution, a kind of linear operation. In other words, they use convolution at least in one of their layers in place of general matrix multiplication.

[1] Dow Jones Industrial Average (DJIA), commonly known as the Dow 30

[2] Exchange-Traded Fund

Convolution Operation

In mathematical terms, a convolution operation is derived from the integral of the product of two functions which expresses how the shape of one function is modified by the other function. The convolution is denoted by the asterisk symbol.

$$(f * g)(t) := \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau \tag{2.9}$$

The central building block of a Convolutional Neural Network (CNN) is the convolutional layer. Neurons in the first convolutional layer are connected only to pixels within their receptive field rather than to every single one in the input image. The key distinction between a fully connected layer and a convolutional layer is that the former learns global patterns in the input feature space, while the latter discovers local patterns. This characteristic gives convolutional networks two important properties:

- Translation Invariance: CNNs recognize learned patterns anywhere in the input data, making them robust to shifts or translations in the data.
- Learning Spatial Hierarchies: CNNs can learn complex patterns by building hierarchies of features, capturing low-level features in early layers and higher-level abstractions in deeper layers.

CNNs have shown high performance not only in image processing but also in natural language processing [76]. The CNN architecture consists of convolution and pooling layers, where each convolution layer contains different kernels. After the convolutional operations, the high-dimensional extracted features pass through a pooling layer to reduce dimensionality.

$$l_t = tanh(x_t * k_t + b_t) \tag{2.10}$$

In Equation 2.10, l_t represents the convolution's output, x_t is the input vector, k_t is the convolution kernel weights, and b_t is the bias. Although a Convolutional Neural Network was initially designed for image processing, it can be utilised for time series forecasting. The reduced number of parameters by the CNN improves the efficiency of the model [100].

Pooling and Down-sampling

In a neural network, a pooling layer serves to condense a group of neighbouring units from the previous layer into a single value. This pooling process is similar to a convolution layer, utilising a specific kernel size l and stride s, but unlike the convolution layer, the pooling operation remains

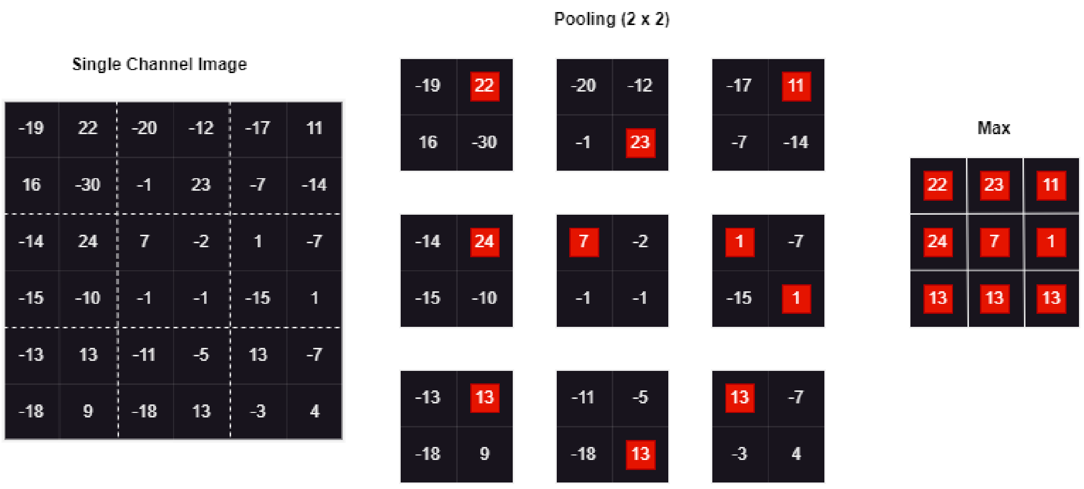


Figure 2.3: Max pooling

fixed and does not involve learning. Notably, pooling layers typically lack an associated activation function. There are two common forms of pooling:

- **Average-pooling** is a technique for down-sampling an image or feature map by summarising local regions into a single representative value. It helps reduce the spatial dimensions of the data while retaining essential features, making the computation more efficient. The process of average pooling involves moving a small window (often referred to as the "pooling window" or "kernel") across the input image or feature map. At each position, the average of the values within the window is computed, and the result is stored in the corresponding output position. The size of the pooling window and the stride (how far the window moves at each step) are typically specified beforehand.

- **Max-pooling** computes the maximum value of its l inputs. It is commonly used for down-sampling, but it differs somewhat in its semantics from average-pooling. For example, if we applied max-pooling to the hidden layer [3,7,4], the result would be a 7, indicating the presence of a darker dot somewhere in the input image that is detected by the kernel. In other words, max-pooling acts as a type of logical disjunction, suggesting the existence of a feature somewhere within the unit's receptive field.

In Figure 2.3, a max-pooling operation is performed on a single channel image with a kernel size of two. In this example, the stride is assumed to be the same size as the kernel.

2.7 Generative Modelling

A generative model, a powerful concept within the domain of artificial intelligence, describes how a dataset is generated based on a probabilistic model. By discerning the underlying patterns and structure from a given dataset, generative models facilitate the creation of new data samples that closely resemble the original examples. This capability has unlocked exciting possibilities across various fields, including computer vision, natural language processing, and even music composition. The application of generative models has garnered considerable attention in recent years due to their potential in data augmentation, fostering creativity, and aiding in problem-solving [59].

Generative models have found diverse applications in numerous fields. In computer vision, they are employed to generate realistic images of objects, faces, or even entire scenes. These synthetic images prove beneficial in augmenting training datasets, leading to more robust and precise computer vision systems. Additionally, in the realm of natural language processing, generative models are utilised to generate coherent and contextually relevant text, enabling applications such as language translation, chatbots, creative writing assistance, and even code snippet generation for programming tasks [20].

Beyond these traditional applications, generative models have shown promise in generating music and art. Researchers have explored using generative models to compose music across various genres, from classical symphonies to modern electronic beats. This technology has piqued the interest of the entertainment industry, as composers and producers experiment with generative music for video games and movies. Likewise, generative models have found application in the field of art, contributing to the creation of stunning paintings and digital art pieces, and showcasing their potential to inspire creativity among artists and designers [39].

Generative modelling has spearheaded a revolution in the finance industry, showcasing remarkable versatility by generating new data samples through probabilistic models. These varied applications have successfully addressed a range of challenges, significantly enhancing financial processes. A prominent achievement of generative models lies in fraud detection, as they glean insights from historical data to identify patterns associated with fraudulent activities. Financial institutions leverage these models to effectively detect and prevent fraudulent transactions, thereby safeguarding their customers and assets. Furthermore, generative models play a crucial role in risk management, simulating potential market scenarios, stress-testing portfolios, and predicting future risk exposures, facilitating prudent decision-making [61].

Investors have also experienced the benefits of generative models in portfolio optimisation. Drawing on knowledge from historical market data, these models generate simulated future scenarios, allowing investors to assess the potential performance of diverse investment portfolios under various market conditions. This empowers them to make well-informed decisions, achieving an optimal balance between risk and return. Credit risk assessment has witnessed substantial improvements through generative modelling. By scrutinising historical customer data and generating synthetic data representing potential borrowers, these models enhance the precision of credit risk evaluations [61].

Financial institutions can better assess creditworthiness and forecast default probabilities, improving the overall effectiveness of risk management processes. For instance, banks can use sophisticated algorithms to analyse a borrower's digital footprint, like online shopping habits or utility bill payment history, to better forecast their likelihood of default than traditional methods. Generative models also help to simulate financial market data, providing valuable insights into market trends and potential scenarios.

Generative models can simulate extreme market events, such as a sharp decline in the stock market or a sudden change in interest rates, to test how portfolios would respond. This helps to test financial models, validate assumptions, and gain a better understanding of how external factors influence market behaviour. For example, you can use generative models to simulate the impact of geopolitical events or fiscal policy changes on currency and stock prices. This provides a more reliable framework for making decisions. Generative models have also shown their ability to analyse customer transaction data to model behaviour patterns and preferences. By analysing transaction data patterns, these models can detect trends such as higher spending in specific categories or a shift to online transactions, allowing banks and financial services providers to provide personalised recommendations, products and services based on these behaviours. This model promotes customer satisfaction and builds loyalty by providing solutions that align closely with customers' evolving needs and preferences, for example, recommending savings plans or credit alternatives based on individual spending patterns.

There are several challenges and limitations associated with generative modelling. One of the biggest challenges is obtaining a sufficiently large and varied dataset to train the generative model efficiently. Techniques such as data augmentation are used to artificially enlarge the dataset, increasing its variability and quality. If there is insufficient or biased training data, the generated samples may be of poor quality or the output may be biased. Generating high-resolution images, or complex data samples, can also be computationally demanding and time-consuming. This is a common challenge that is often solved by using distributed computing and specialised hardware, such as GPUs, which speed up the training process.

Another important challenge is to ensure that the samples created are realistic and consistent. The design of the model architecture and the loss function play an important role here, which directs the model to learn to produce more realistic data. Generative models can suffer from mode collapse, producing limited variation by ignoring certain states of the data distribution. Spatial search algorithms are used to solve this. These algorithms modify the training process, especially for generative adversarial networks (GANs), by penalising the generator for producing similar samples, thus encouraging diversity. Architectural improvements also have a significant impact. The quality and variety of results can be improved by, for example, innovations such as Progressive GANs, which gradually increase the resolution of generated images, or StyleGAN, which differentiates between style and content.

Regularisation techniques such as adding noise to the inputs or using output layers are used to reduce spatial collapse while avoiding overturning. Furthermore, careful tuning of the hyper-parameters is critical to achieve a balance between the diversity and realism of the generated samples. Through a combination of these data strategies, computational techniques, architectural innovations, and algorithmic changes, generative models can be improved to produce more versatile, high-quality, and realistic results.

Several notable generative model architectures have been successful in different tasks. Autoencoders are unsupervised neural networks that aim to compress and reconstruct input data. By encoding the data into a lower-dimensional latent space and then decoding it back into its original form, autoencoders learn to represent data efficiently. They find applications in dimensionality reduction, denoising, and feature learning. Variational Autoencoder (VAE), a variation of autoencoders, introduce probabilistic modelling to the latent space. This enables VAEs to learn a statistical distribution of the data, allowing for interpolation and sampling of new data points. VAEs have been extensively used in image generation, data compression, and anomaly detection [4].

The Generative Adversarial Network (GAN), proposed by Goodfellow et al. [48], on the other hand, employ a unique adversarial framework, consisting of a generator and a discriminator. A generator and a discriminator, compete continuously. The job of the generator is to produce data instances that mimic real data as closely as possible. On the other hand, the Discriminator evaluates the authenticity of the information and differentiates between the real information and the imitation produced by the generator. This arrangement creates a dynamic competitive environment where both networks are constantly improving their functionality - the Generator learns to generate more convincing data and the Discriminator learns to better identify fakes [4].

During training, the goal of the Discriminator is to maximise the accuracy of distinguishing real data from fakes, while the goal of the generator is to produce data so convincing that the discriminator mistakenly considers it true. Training involves a back-and-forth process where the generator continuously tries to outsmart the discriminator. If the Discrimination becomes too good, the Generator may fail to improve due to a lack of meaningful feedback (a problem known as vanishing gradients). Conversely, if the generator is too powerful, the Discrimination can degenerate into a guess, leading to unstable training dynamics.

One significant challenge in training GANs is “mode collapse” where the Generator starts generating limited outputs, losing the diversity of the real data. Ensuring that both networks improve at a balanced rate is crucial to avoid this issue. Another challenge is maintaining stable training; due to the adversarial setup, it’s common for GANs to experience oscillations in performance, where the networks continually try to outdo each other without reaching an optimal state [21].

GANs have been utilised in a diverse range of domains, with a specific focus on the realm of image generation. These models have played a crucial role in the production of exceedingly lifelike images and artistic creations, thereby facilitating progress in many fields. In addition to their contributions to visual representation, GANs have also been employed in various other areas such as natural language processing, video generation, and the creation of music, thereby showcasing their versatility and adaptability.

Autoencoders

Autoencoders are unsupervised learning techniques that employ neural networks for the task of representation learning. An autoencoder aims to copy the inputs to the outputs. At its core, a hidden layer h describes a code used to represent the output. This hidden layer acts as a bottleneck that forces the compressed knowledge representation to the input features. If a pattern exists in the data, i.e., the correlation between the input features, this pattern can be learned. Figure 2.4 illustrates the autoencoder architecture. Autoencoders are crafted to be unable to learn to copy perfectly. If they did, they would act as an identity function, copying the input to the output without learning. Usually, they are restricted in a way that allows them to copy only approximately and only inputs that resemble the training data. To ensure the acquisition of meaningful knowledge by autoencoders, it is conventional to impose limitations upon them in one or multiple aspects of their operations:

- **Reduced Dimensionality**: The latent space, in which the encoder squeezes the data, frequently has a dimensionality that is less than that of the input space. Consequently, the autoencoder is compelled to acquire a condensed representation of the data, encompassing solely its most crucial characteristics.
- **Regularisation**: Autoencoders also incorporate regularisation techniques, such as imposing sparsity constraints, to mitigate the risk of the network solely memorising the input data. Consequently, the network is forced to assign priority to specific aspects of the data for encoding.

Due to these restrictions, autoencoders are incapable of copying the input to the output. Instead, they need to find the most efficient way to represent the input data in a smaller, constrained form.

The autoencoders' architecture 2.4 comprises two parts, an encoder $h = f(x)$ and a decoder $r = g(x)$. Encoders and decoders are perhaps one of the basic architectures in the domain of deep learning. Encoders and decoders are present in all the network structures. Hidden layers in a fully connected layer can be regarded as an encoding from the encoder and the output layer as a decoding from the hidden layer into the output.

Commonly, encoders encode the input into a vector of intermediate state and then the decoder decodes the vector into the output. A sequence-to-sequence (seq2seq) network used in machine translation is an established example of an encoder-decoder network. A sentence from a source language will be encoded into an intermediate vector representation, and then the decoder decodes the output sequence into the target language using the intermediate vector.

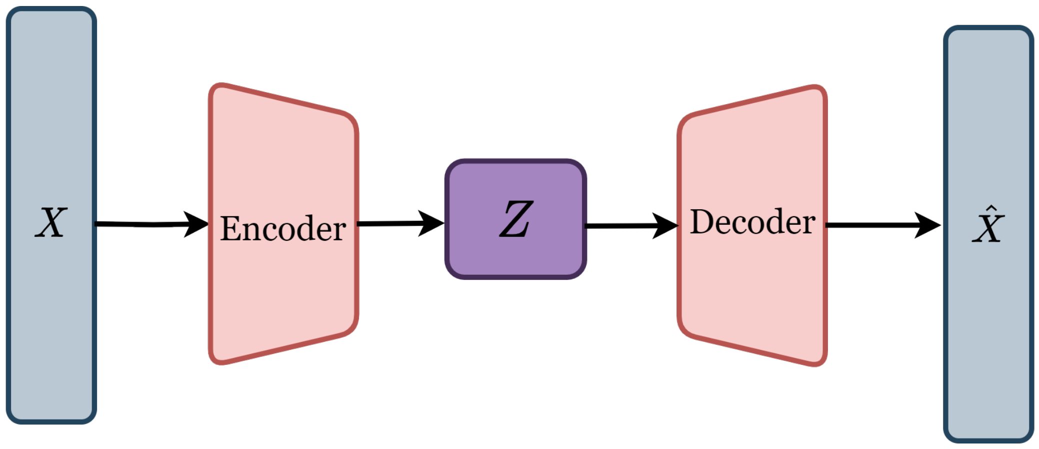


Figure 2.4: Autoencoder Architecture

With the advent of modern autoencoders, the idea of encoder and decoder has been generalised beyond deterministic functions to stochastic mappings. The training of autoencoders is a process of minimising the reconstruction loss $\mathscr{L}(x, \hat{x})$. Reconstruction loss is defined as the difference between the original features and their corresponding reconstructions. Undercomplete autoencoder is the one with a lesser code dimension. Undercomplete representation forces the model to capture the most outstanding features of the training data. In the learning process, the network minimises the loss function $L(x, g(f(x)))$ where L penalises $g(f(x))$ dissimilarity to x such as the mean squared error (MSE).

An autoencoder can be considered as the generalisation of the Principle Component Analysis (PCA), whereas PCA attempts to discover a lower dimensional hyperplane which describes the original data, autoencoders can learn non-linear manifolds. A manifold is a mathematical space that locally resembles Euclidean space like lines, planes, and other higher-dimensional spaces, but can have a more complex, global structure. Essentially, a manifold is a space where, if we zoom in closely enough at any point, the space looks like a flat Euclidean space, but the overall shape or structure might be curved or twisted. The non-linearity implies that the manifold can bend, curve, or have a more complex geometry. This non-linearity is in contrast to linear manifolds, where the structure is flat and can be fully described using linear equations.

For high-dimensional data, autoencoders can learn a complex representation of the data manifold which can be used to describe observations in a lower dimensionality and decoded into the original inputs. Undercomplete autoencoders are capable of learning the most salient features in the data distribution. Also, researchers observed that too much capacity for encoders and decoders prevents the model from learning valuable patterns [23]. The same shortcoming occurs if the hidden dimension is equal to the input, and in the overcomplete case with the hidden dimension larger than the input. In such cases, even a linear encoder and decoder learn only to copy the inputs to the outputs.

For optimal functionality in an autoencoder, a strategic selection of the code dimension and the capacity of the encoder and decoder is crucial. This selection should be carefully tailored to match the complexity of the data distribution that the model aims to represent. Instead of merely restricting the network's capacity through a shallow architecture and minimal code size, regularised autoencoders adopt an innovative approach. They employ a specialised loss function designed to encourage the model to learn beyond simple replication of the input. This loss function incentivises the autoencoder to develop additional capabilities, such as achieving a sparse representation, minimising the derivatives of the representation for smoother transitions, and enhancing robustness against noise or incomplete data [47]. These additional learning objectives enable the regularised autoencoder to extract more profound and valuable insights from the data, notwithstanding its nonlinear nature or overcomplete representation capabilities. This approach prevents the autoencoder from falling into the trap of learning a mere identity function, which would render it ineffective for meaningful data analysis. By focusing on these aspects, the regularised autoencoder becomes adept at understanding and representing the underlying data distribution in a more nuanced and detailed manner [47].

Generative and discriminative models represent two different approaches in the field of machine learning for tasks such as pattern recognition and classification. Generative models focus on understanding and reproducing the creation of data, to model the joint probability distribution of both input functions and output items. This approach facilitates learning the underlying distribution of each class in the data, capturing $P(X|Y)$ the probability of the inputs given the output class and $P(Y)$ the probability of each output class. By doing this, generative models such as naive Bayesian models, and hidden Markov models can not only classify new data but also create new data instances that resemble the training data. On the other hand, discriminant models, exemplified by logistic regression and support vector machines, go the other way. In the mentioned approaches, the decision boundary between the classes is directly modelled by estimating $P(Y|X)$ the probability of the class given the input features. Consequently, they are more efficient in classification tasks, since the differences between classes are targeted without delving into how each class's data is generated. Both approaches have their merits and are valuable tools [82].

The same holds in other disciplines, e.g., chemistry, biology, and economics. Modelling is almost always generative modelling. There are many reasons why generative modelling is essential. Many underlying laws and constraints can be expressed through generative by treating nuisance variables as noise. These resultant models are very intuitive and interpretable and can be tested against observations to confirm or reject our theories. Another reason to understand the generative process of data is that the causal relations can be expressed through a generative process. The benefit of causal relations is their generalisability to new situations.

The Bayes rule is needed to transform a generative model into a discriminator. Suppose there exists a generative model for events A and B. By comparing the capability of the two in describing the data, we can compute a probability of whether A or B happened. In contrast to a generative model, we directly learn a map in a direction we intend to make future predictions within discriminative models. As an example, it can be argued that an image is generated by first identifying the object, then generating it in three-dimensional space and subsequently projecting it into a pixel grid. Unlike a generative model, a discriminative model treats pixel values as input and maps them to labels. Generative models can better learn from data and make more robust hypotheses, causing a higher asymptotic bias when a model is wrong [9].

Generative modelling can be advantageous more generally, and it can be thought of as an auxiliary task. As an example, predicting the immediate may contribute to building valuable abstractions of the world that can be used for multiple subsequent prediction tasks. The search for meaningful, statistically independent and causal variation factors is known as unsupervised representation learning, and the variational autoencoder has been comprehensively used for that specific reason. On the contrary, viewing this process as a form of regularisation, the representations are forced to be meaningful, an inverse process that maps from inputs to representations and into a specific mould.

Variational Autoencoder

The variational autoencoder can be regarded as a two-joined yet independently parameterised model: the recognition model or the encoder and the generative model or the decoder. The recognition model provides the posterior approximation over the latent variables for the generative model, which is needed to update the parameters in the expectation-maximisation iteration. The generative model supports the recognition model to learn meaningful representations of data. According to the Bayes rule, the recognition model is the approximate inverse of the generative model. As it was mentioned earlier, generative models are primarily concerned with how the data is generated. It aims to model the joint probability distribution $P(X,Y)$, where X represents the observable data, and Y denotes the associated labels or outputs. Utilising the joint distribution, a generative model can generate new samples and infer the conditional probability $P(Y|X)$, which is the probability of the labels given the data. A recognition model, often used in the context of variational autoencoders is typically tasked with approximating the posterior distribution $P(Z|X)$, where Z represents latent variables and X is the observed data. The recognition model effectively tries to infer the latent structure of the data [47].

The benefit of the variational autoencoder framework, relative to the ordinary Variational Inference (VI), is that the recognition model is now a stochastic function of the input variables. This is contrary to VI, where each data case has a different variational distribution. The recognition model utilises a set of parameters called amortised inference to model the relationship between input and latent variables. In Variational Autoencoders (VAEs), the identification model (or encoder) is formulated as a probabilistic function of the input parameters. This implies that for any specified input, the model generates a probability distribution over the underlying variables, instead of a single estimation. The stochastic characteristics of this process assist the model in capturing uncertainty in the mapping from inputs to latent variables. In classical variational inference methodologies, each data point commonly has a distinct variational distribution. This technique generates an individualised yet fixed distribution for each data, resulting in a situation where the model must identify and optimise a separate set of variational coefficients for each data point. The variational autoencoder (VAE) inspired by Dayan et al. [34] was the first model to employ a recognition model. However, its algorithm was incapable of optimising a single objective. Alternatively, the variational autoencoder rules follow a single approximation to the maximum likelihood objective. Variational autoencoders are a combination of graphical models and deep learning.

Analogously, the recognition model is also a conditional Bayesian network of form $q(z|x)$ or as a hierarchy such as $q(z_0|z_1)\cdots q(z_L|X)$. Inside each conditional, there may be a deep neural network. Its learning algorithm is a classical expectation maximisation with a reparametrisation trick which back-propagates the embedding through the layers of the deep neural network. From the outset, the variational autoencoder framework has been extended into numerous directions, e.g., models with attention [50], dynamical models [79], models with multiple levels of stochastic latent variables [70], to name a few.

Another generative modelling paradigm, the generative adversarial network (GAN), has received particular attention [48]. GANS can generate images with high perceptual quality. As opposed to likelihood-based generative models, GANs lack full support over the data [51]. Generated samples of VAEs are more dispersed but are better models concerning likelihood criterion, similar to other likelihood-based models. The variational autoencoder framework provides a method for learning deep latent-variable models and inference models stochastic gradient descent. It has various applications, from generative modelling and semi-supervised learning to representation learning.

In 2013, the introduction of the Variational Autoencoder (VAE) by Diederik P. Kingma and Max Welling marked a significant advancement in generative modelling. The VAE framework distinguishes itself from traditional Variational Inference (VI) through the utilisation of a recognition model that operates as a stochastic process for the input variables. This is in contrast to VI, where each data case is associated with a unique variational distribution. In VAEs, a collection of parameters, referred to as amortised inference, is employed to establish the relationship between input and latent variables. This approach enables VAEs to generate disentangled representations, which offer reusable latent codes for applications such as transfer learning. Despite sharing similarities in structure with regular autoencoders, VAEs deviate in both their objectives and mathematical formulations [78, 77, 79].

The main objective of a variational autoencoder is to compress the input into a constrained multivariate latent distribution and then reconstruct it with high accuracy. This contrasts with the objective of a generative model, which seeks to approximate the distribution of input data using neural networks. The mathematical representation of this relationship is:

$$x \sim P_\theta(x) \tag{2.11}$$

In Equation 2.11, θ represents the parameters determined during training. Effective inference in this context requires establishing the joint distribution between inputs and latent variables, denoted as $P_\theta(x,z)$, which encompasses the distribution of input data points and their attributes. The marginal distribution of $P_\theta(x)$ is obtained through:

$$P_\theta(x) = \int P_\theta(x,z)dz \tag{2.12}$$

Equation 2.12 integrates all possible attributes to derive a distribution that characterises the inputs. However, this equation poses a challenge due to its intractability and non-differentiability with respect to the parameters, causing it unsuitable for neural network optimisation. An alternate formulation, based on Bayes' theorem, is:

$$P_\theta(x) = \int P_\theta(z \mid x) P(z) dz \tag{2.13}$$

In Equation 2.13, $P(z)$ is the prior distribution over z and remains unconditioned on observations. For discrete z and Gaussian $P_\theta(z \mid x)$, $P_\theta(x)$ is considered a mixture of Gaussian distributions. Constructing a neural network to approximate $P_\theta(x \mid z)$ without an appropriate loss function might lead to solutions that disregard z. To address this, variational inference introduces:

$$Q_\phi(z \mid x) \approx P_\theta(z \mid x) \tag{2.14}$$

Equation 3.2 presents $Q_\phi(z \mid x)$ as a parametric and tractable estimate of $P_\theta(z \mid x)$, which can be optimised by adjusting the parameter ϕ. This formulation significantly enhances the tractability of the VAE framework, solidifying its role in the field of generative modelling.

2.8 Residual Networks

Deep convolutional neural networks have brought significant advancements to image classification. These networks naturally combine low, mid, and high-level features as well as classifiers in a multi-layered manner, and the richness of the features improves with the depth of the stacked layers. Recent research highlights the critical importance of network depth, and the most successful results on challenging datasets, such as ImageNet, are achieved using "very deep" models with sixteen to thirty layers. Many other complex visual recognition tasks have also greatly improved by leveraging these deep models. The question is whether adding more layers is sufficient to improve the performance of a neural network. The answer is no. An obstacle called vanishing gradient hinders the network from converging [12, 44]. The vanishing gradient problem is a common issue when training deep neural networks. As the network becomes deeper, the gradients used to update the model's weights during back-propagation can become extremely small, effectively vanishing. This phenomenon makes it challenging for the network to learn from the earlier layers.

The introduction of Residual Networks [60], also known as ResNets, marked a significant breakthrough in overcoming the vanishing gradient problem and enabled the training of much deeper neural networks effectively. The skip connections in ResNets are crucial for their success. Traditional deep neural networks without skip connections attempt to learn the direct mapping from one layer to the next. As the depth of the network increases, the gradients can become extremely small during back-propagation, which leads to the vanishing gradient problem. When gradients are

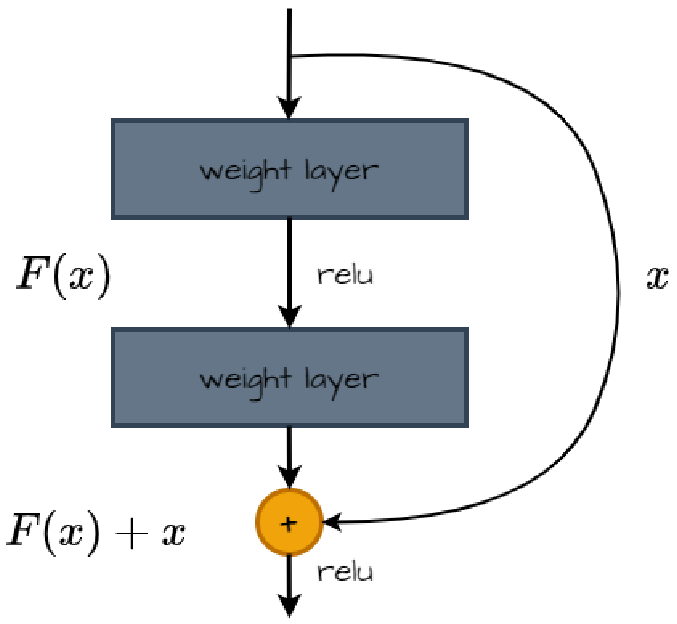


Figure 2.5: Residual Skip Connection.

too small, the network struggles to learn meaningful representations in the earlier layers, and the overall performance suffers. In contrast, ResNets introduce residual blocks that utilise skip connections to learn the residual mapping. These blocks take the input and attempt to learn the difference between the input and the desired output rather than learning the direct mapping. Mathematically, if $H(x)$ represents the mapping of a layer in a traditional neural network (without a skip connection), and $F(x)$ represents the mapping learned by the residual block, then the residual block learns $F(x) = H(x) - x$. The skip connection directly propagates the input x to the output of the residual block [122].

Consequently, the network can always fall back to the identity mapping if the optimal mapping learned by the residual block is close to the identity mapping. In this scenario, the gradient can flow smoothly through the skip connection without significant loss of information, alleviating the vanishing gradient problem. As a result of using residual blocks and skip connections, ResNets can be much deeper than traditional neural networks without encountering the vanishing gradient problem. This depth allows ResNets to capture increasingly complex patterns and features in the data, leading to better performance on challenging visual recognition tasks like image classification.

Furthermore, the impact of ResNets extends beyond image classification tasks. The idea of skip connections and residual learning has been applied to various other domains in deep learning, including natural language processing, speech recognition, and even reinforcement learning [87]. The fundamental concept of allowing the gradient to flow freely through the network and making it easier for the model to learn from earlier layers has proven to be a valuable technique in a wide range of applications.

ResNets offer a significant advantage by simplifying the optimisation process during training. The incorporation of skip connections permits the gradients to flow directly, thereby enabling the network to update the weights in earlier layers effectively. As a result, faster convergence and more secure learning are achieved, making this attribute particularly valuable in the face of complex datasets or limited training data. In such scenarios, the network's ability to generalise from earlier layers is critical to attaining optimal performance [11].

Moreover, Residual Networks have also provided impetus for the creation of other structures and fundamental units that endeavour to amplify the learning process of complex neural networks. For example, DenseNet, which is another widely-used architecture, takes the notion of skip connections to a greater extent by densely connecting each layer with all the other layers in a block. This further enhances the flow of gradients and reuse of features. Despite their success, ResNets are not without challenges. As the network becomes deeper, it can encounter other issues, such as over-fitting or increased computational complexity [74]. Designing the right architecture and hyper-parameters for a specific task and dataset is still an essential part of leveraging the full potential of ResNets. As the field of deep learning continues to evolve, researchers have not only focused on improving the depth of neural networks but also on addressing other challenges associated with training and optimising these models. One significant advancement that complements the success of ResNets is the use of transfer learning and pretraining on large-scale datasets.

In the field of finance, the application of ResNets and transfer learning is viable for examining intricate market data. ResNets possess a profound architecture that renders them suitable for capturing the convoluted patterns present in financial time series data. To illustrate, they can be employed in the prediction of stock prices, identification of market trends, or detection of anomalies within trading data. To utilise the feature extraction capability and computational efficiency of pre-trained image models e.g., ResNets having an image representation of financial time series data is necessary.

Transfer learning is the process of leveraging acquired knowledge from a task to boost performance on a related task. By utilising transfer learning and pre-trained models like ResNets, researchers can obtain highly effective feature extractors without the need to train extremely deep networks from scratch on smaller datasets. This approach not only saves computational resources and training time but also often leads to improved generalisation and better performance on the target task. Another direction in which deep learning has evolved is the development of attention mechanisms. Attention mechanisms allow the neural network to selectively focus on the most relevant parts of the input data, giving rise to models like Transformer networks.

Transformers have achieved remarkable success in natural language processing tasks, where sequences of words need to be understood and processed effectively. These models have been extended to computer vision tasks as well, where they can attend to relevant image regions and effectively handle tasks like image captioning and object detection. The combination of the mentioned advancements has led to a thriving field of deep learning with applications in diverse domains. From computer vision to natural language processing, speech recognition, and reinforcement learning, deep learning models are making a significant impact across various industries and scientific disciplines.

2.9 Transfer Learning

Can Machines Engage in Thought? This inquiry lays the groundwork for artificial intelligence (AI) and has spurred generations of researchers to delve into the design of intelligent machines. The trajectory of AI has seen its share of ups and downs. An enduring challenge that has posed obstacles on the AI journey revolves around how machines can glean knowledge from their surroundings. The quest to imbue machines with human-like cognitive processes has transitioned from rigidly imposing rule-based information to embracing data-driven learning.

This evolution has propelled machine learning into a pivotal industrial and societal tool, automating decision-making across domains like education, healthcare, finance, and commerce. This ascendancy owes much to machine learning's knack for bestowing machines with insight via labelled and unlabelled data. Nevertheless, achieving accurate predictions hinges on astute observations and a deep grasp of task domains. Often, training data comprises labelled samples, providing both observations and target outcomes. These examples are harnessed to craft machine learning models primed for forecasting novel data [105].

For instance, consider the illustrative domain of computer-based image analysis, specifically facial recognition. Imagine training a machine learning model on an expansive collection of photos. This model becomes adept at discerning whether a new image corresponds to someone within its training set. A practical application could be a building's security system that ascertains an individual's permission to enter. While these machine learning models are impressive, they are not immune to errors, especially when dealing with unfamiliar subjects or contexts. If, for instance, a model is trained exclusively on images of puppies, its recognition proficiency might plummet when presented with images of bunnies. This performance dip underscores the necessity for model updates and transfers in response to novel situations [135].

An overarching challenge in deploying machine learning algorithms is their tendency to falter in new task domains. This can be attributed to the scarcity of pertinent training data and shifts in circumstances and requirements. Medical imaging data serves as an example, where obtaining high-quality training data in novel situations can be a formidable hurdle. A machine learning model's efficacy relies on ample training data, yet accessing labelled data from a new realm often involves substantial resources, hampering the application of AI models in real-world contexts.

The need to develop transfer learning methodologies arises from several key factors:

- The success of machine learning hinges on copious labelled data, a resource that can be scarce, impeding generalisation to new domains.
- Traditional machine learning assumes uniformity between training and test data distributions. Yet, distributions can fluctuate over time, space, and scenarios, necessitating model adaptation.
- Personalising services to individual user preferences is vital, but gathering substantial personal data can be impractical in many scenarios.
- In certain cases, utilising multiple datasets with diverse owners is necessary, but privacy and security concerns limit data sharing. Extracting the core of each dataset for constructing a new model becomes imperative.

In essence, transfer learning stands as a machine learning paradigm where algorithms extract knowledge from diverse scenarios to bolster performance in target scenarios. In contrast to traditional machine learning, transfer learning doesn't demand copious, finely defined training data as input. Instead, it represents a fresh paradigm. This concept addresses data sparsity and cold start issues in numerous large-scale applications. Transfer learning empowers artificial intelligence in less technologically advanced sectors where labelled data is scarce.

A proficient model from one domain can be extended to similar domains using transfer learning techniques. To facilitate this, an accurate measure of domain task distance is crucial. This distance can be gauged in terms of the features used to describe data. For image analysis, features might be pixels, colours, or shapes, while in Natural Language Processing, words or phrases serve as features. Once domain comparability is established, AI models from well-established domains can be effectively utilised in less mature ones. This knowledge transfer broadens the scope of machine learning systems beyond their initial domain, rendering AI more robust and accessible in resource-scarce settings [128].

Various terms have been applied to transfer learning within machine learning, such as knowledge reuse, case-based reasoning (CBR), and learning by analogy. The notion of "transfer of learning" from education and learning psychology shares a similar essence in machine learning. It signifies the process of leveraging past experiences to shape future learning and performance in a target context. The learned knowledge or model is adapted for new tasks.

Despite the remarkable strides made by machine learning across practical problems, limitations persist in specific real-world scenarios. Conventional machine learning algorithms thrive on abundant labelled examples that match the test data distribution. However, attaining such training data is often expensive and time-consuming in reality. Semi-supervised learning, a related domain, mitigates this challenge by relying less heavily on extensively labelled data. It capitalises on sizeable unlabelled datasets to enhance learning accuracy. In this context, transfer learning emerges as a promising solution.

Regarding domain disparities, transfer learning is categorised into two types [132]: homogeneous and heterogeneous. Homogeneous transfer learning tackles cases where domains share the same feature space. However, this assumption doesn't hold in numerous instances. For instance, a word might possess distinct meanings in various domains, leading to context feature bias. Heterogeneous transfer learning [33] confronts domains with disparate feature spaces, necessitating adaptation of both distribution and feature space [67, 119].

Human beings innately excel at transferring knowledge across tasks. We naturally apply previously learned insights to new challenges. The more akin a new task is to our prior experience, the quicker we grasp it. On the contrary, conventional machine learning algorithms are engineered for isolated tasks. Transfer learning enriches classical machine learning by transposing knowledge acquired in source tasks to related target tasks. Transfer methods often extend machine learning algorithms, relying heavily on them.

For instance, within inductive learning, transfer learning involves extending classification, inference algorithms like neural networks, and Bayesian networks, as well as Q-learning and policy search in reinforcement learning. Transfer learning aims to leverage source task knowledge to bolster learning in target tasks. This enhancement can manifest through three dimensions:

- Initial attainable performance
- Training time
- Final attainable performance

Figure 2.6 depicts the conceptual representation of transfer learning where a developed model for one task could be utilised as the starting point for a model on a second task. The source refers to the source domain where the initial model has been trained. The source model has gathered certain knowledge from this domain. The target label represents the target domain to which the

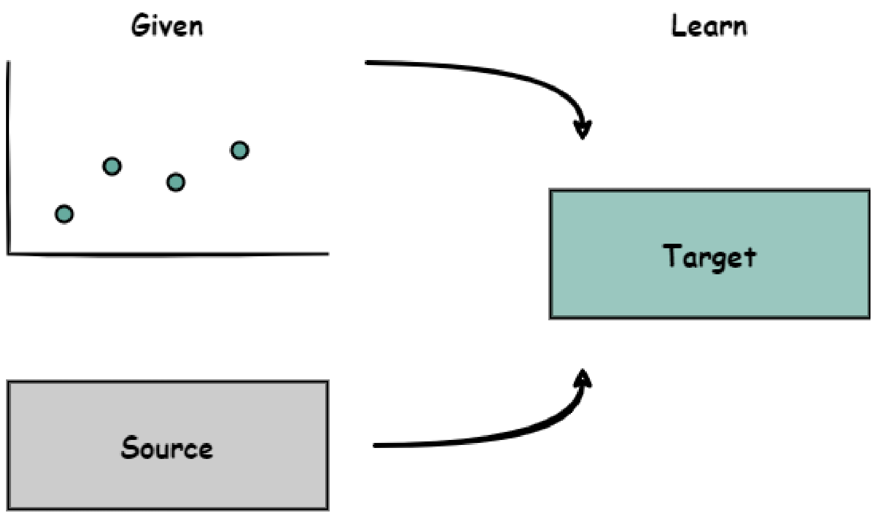


Figure 2.6: Transfer learning in machine learning

knowledge is being transferred. The target domain usually has less data available training data. The objective is to leverage the pre-existing knowledge gathered from the source domain to improve the performance in the target domain.

Figure 2.7 vividly illustrates how transfer learning contributes to improved learning through these dimensions. There's a marked elevation in initial performance, a steeper learning curve, and higher asymptotic performance towards the curve's end. Should a transfer lead to performance degradation, a negative transfer has occurred. Facilitating positive transfer between related tasks while sidestepping negative transfer poses a significant challenge in transfer method development. Often, mapping features from one task to another is required to establish correspondence when applying knowledge. While humans often provide these mappings, automated methods also exist.

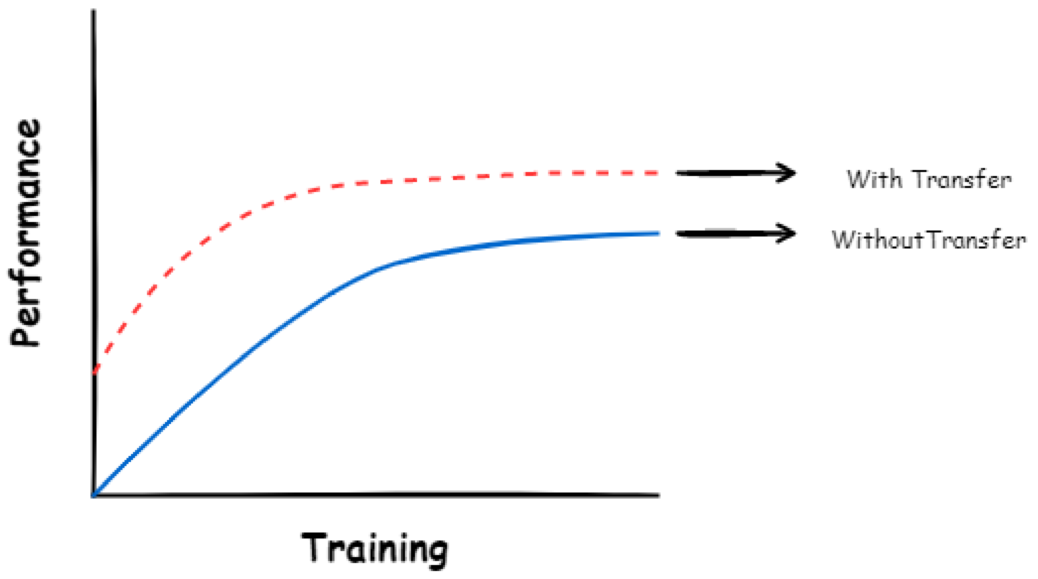


Figure 2.7: Contributions of transfer learning to improved learning

Transfer learning is aimed at rapid adaptation of systems to new scenarios, tasks, and environments. It equips machine learning systems with the capability to harness auxiliary data and models to solve target problems when scant data is available. This imbues these systems with reliability and resilience, preventing significant deviations from expected performance in the face of unforeseen changes. At an organisational level, transfer learning empowers knowledge reuse, enabling once-acquired expertise to be repeatedly deployed in the real world.

2.10 Gramian Angular Fields

Deep neural networks or deep learning techniques have deeply impacted many areas of artificial intelligence, such as natural language processing, speech recognition, and computer vision. One of Computer Vision's successful deep learning architectures is convolutional neural networks (CNN). CNN architecture takes advantage of translational invariance by extracting features through receptive fields and learning with weight sharing. Convolutional neural networks have become the go-to approach in various computer vision and image recognition problems. Accomplishments of supervised and unsupervised learning techniques in computer vision inspired Wang and Oates [131] to consider the problem of encoding time series as images to enable machines to recognise and learn patterns and structures visually.

Redefining time-series features as visual clues have raised attention in computer science and physics. Researchers have built different network structures from time series for visual inspection or designing distance measures. Silva et al. [115] utilised the compression distance to expand the recurrence plot paradigm for time series classification. Another approach to construct a weighted adjacency matrix is extracting transition dynamics from the first-order Markov matrix [19]. Despite demonstrating distinct topological properties amongst different time series, these maps are unclear about how they relate to the original time series since they lack exact inverse operations.

Gramian Angular Fields (GAF) imaging proposed by Wang and Oates [131] is a technique to encode time series into images. The reason for developing this approach is the possibility of using existing pre-trained models rather than training RNNs or employing one-dimensional CNN models. In order to generate GAF images, time series observations are required to be re-scaled.

Let $X = \{x_1, x_2, \cdots, x_n\}$ be the considered time-series with n observations, re-scaling to the $[-1, 1]$ interval is done using Equation 2.15 or within $[0, 1]$ interval using Equation 2.16:

$$\tilde{x_i} = \frac{(x_i - max(X)) + (x_i - min(X))}{max(X) - min(X)} \tag{2.15}$$

$$\tilde{x_i} = \frac{x_i - min(X)}{max(X) - min(X)} \tag{2.16}$$

Hence, the re-scaled series is denoted by $\tilde{X} = \{\tilde{x_1}, \tilde{x_2}, \cdots, \tilde{x_n}\}$. The re-scaled time series is transformed into a polar coordinates system by calculating the angular cosine of every single component of the scaled time series:

$$\begin{cases} \phi_i = \arccos(\tilde{x}_i), & -1 \le \tilde{x}_i \le 1, \tilde{x}_i \in \tilde{X} \\ r_i = \frac{i}{N}, & i \in N \end{cases} \tag{2.17}$$

Here r_i represents the radius for the time stamp i, i is the time stamp and N is a constant factor to regularise the span of the polar coordinate system. Contrary to the Cartesian coordinates, polar coordinates preserve temporal relations. Vidal and Kristjanpoller [127] proposed an LSTM added to the convolutional neural networks model (specifically, a pre-trained VGG16 network) to forecast the future volatility of Gold. Gramian Summation Angular Field (GASF) and Gramian Difference Angular Field (GADF) can be obtained by calculating the sum or difference between the points of the series:

$$GASF = [\cos(\phi_i - \phi_j)] = \tilde{X}^\top \cdot \tilde{X} - \sqrt{I - \tilde{X}^2}^\top \cdot \sqrt{I - \tilde{X}^2} \tag{2.18}$$

$$GADF = [\sin(\phi_i - \phi_j)] = \sqrt{I - \tilde{X}^2}^\top \cdot \tilde{X} - \tilde{X}^\top \cdot \sqrt{I - \tilde{X}^2} \tag{2.19}$$

I is the unit row vector $[1, 1, \cdots, 1]$.

2.11 Markov Chain

A Markov chain is a memory-less stochastic process, i.e., future actions are not dependent on previous states. This is called the Markov property. Simply put, a Markov chain characterises a process where a system can exist in a set of discrete states, and it moves from one state to another over time. The key characteristic is that the future behaviour of the system is only influenced by its current state and the probabilities of transitioning to other states from its current state. A Markov chain consists of the following components:

- **State Space** (S): This is defined as the set of all possible states.
- **Transition Probability Matrix** (P): This $n \times n$ matrix outlines the probabilities of transition-

ing from one state to another. The elements in the i^{th} row and j^{th} column represent the probability of transitioning from state i to state j.

- **Initial State Distribution**: This is the probability distribution representing the likelihood of the system starting in each of the possible states.

Following is an example of a transition probability matrix for a contrived weather model with three states: Sunny, Cloudy, and Raining.

$$P = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.3 & 0.4 & 0.3 \\ 0.2 & 0.3 & 0.5 \end{bmatrix}$$

This matrix indicates, for instance, that if it's currently sunny, there's a 70% chance that it will remain sunny the next day, a 20% chance it will become cloudy, and a 10% chance it will become rainy. Markov chains have various applications in many fields. They are employed to model processes where the outcome is uncertain and depends on the current state.

2.12 Markov Transition Fields

Markov Transition Field (MTF) is an image-like representation of transitions between different quantile-based states in a time series and offers insights into the temporal dynamics of the data. Given a time series X, it is divided into Q quantile bins to group data points by their values. Next, a $Q \times Q$ matrix is created where each element represents the probability of transitioning from one quantile bin to another. Subsequently, the matrix is normalised to ensure accurate representation. The resulting MTF matrix highlights patterns of transition probabilities. Consider the following time series of daily stock prices over a week:

$$X = [50, 52, 55, 48, 57, 49, 53] \tag{2.20}$$

To create simple MTF with $Q = 3$ quantile bins, the following steps are taken:

- **Quantile Assignment**:
  - Sort Data: $[48, 49, 50, 52, 53, 55, 57]$
  - Assign values to bins:
    - $q_1 : [48, 49, 50]$
    - $q_2 : [52, 53]$

    * $q_3 : [55, 57]$

- **Counting Transitions**:
  - $q_1 \to q_1$: 1 occurrence.
  - $q_1 \to q_2$: 1 occurrence.
  - $q_2 \to q_2$: 1 occurrence.
  - $q_2 \to q_3$: 1 occurrence.
  - $q_3 \to q_1$: 1 occurrence.
  - $q_3 \to q_2$: 1 occurrence.
- **Constructing Transition Probability Matrix** W: Using the transition counts, the transition probability matrix W is constructed. W_{ij} represents the probability of transitioning from quantile bin q_i to q_j.

$$W = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 0 & 1/2 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$

- **Normalisation**: Each row of the W matrix is normalised:

$$W_{normalised} = \begin{bmatrix} 0.33 & 0.33 & 0.33 \\ 0 & 0.25 & 0.25 \\ 0.5 & 0.5 & 0 \end{bmatrix}$$

- **Markov Transition Field Visualisation**: Visualise the normalised matrix W as a grid where the colour or intensity of each cell represents the transition probability. Darker cells indicate higher probabilities.

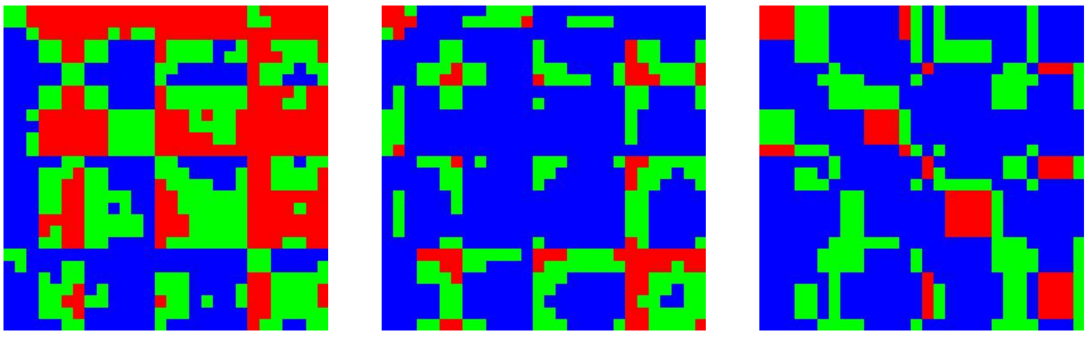

(a) Gramian Angular Difference Field.
(b) Gramian Angular Summation Field.
(c) Markov Transition Field Image.

Figure 2.8: GAF and MTF Images

Chapter 3

Event Prediction within Directional Change Framework using a CNN-LSTM Model

3.1 Introduction

The focus of this study, presented in this chapter is to investigate how a deep learning model i.e. CNN-LSTM model performs in the Directional Change framework considering the volatility of high-frequency financial data[1]. The chapter concludes by discussing the findings of the study and suggesting future research directions, including exploring other neural network architectures and determining the Directional Change threshold dynamically. Overall, the study highlights the effectiveness of the CNN-LSTM model within the DC framework for event prediction in high-frequency financial markets.

3.2 Methodology and Experimentation

The methodology begins with a brief introduction to Support Vector and Random Forest Regression, which will be used for comparison. The data in this study comprises tick prices of GBPUSD, EURUSD, USDCHF, and USDCAD currency pairs and tick bars are generated from these prices. Tick bars with the least auto-correlations are then identified using the Durbin-Watson statistic, and the

[1]This Chapter is based on a published work in the journal of Neural Computing and Applications.

ATR is computed for these tick bars to determine the θ value. DC summaries are generated based on this threshold value. The experiment is carried out by training and validating the CNN-LSTM model using the DC summaries within the framework and without the framework using raw tick bars.

The model's predictive capabilities are assessed through metrics such as mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination (R^2). The results indicate that the CNN-LSTM model performs significantly better within the DC framework for all tested currency pairs, showing improvements in MAE, RMSE, and R^2 metrics. Comparative analysis between Support Vector and Random Forest regression models, in conjunction with the CNN-LSTM model, demonstrates that the latter outperforms the former in terms of predictive accuracy within the DC framework.

Support Vector Machines and Random Forest

Support Vector Machines (SVM) are a sophisticated set of supervised learning algorithms that are primarily employed for classification and to a lesser extent for regression tasks. Developed by Vapnik [126], Support Vector Machines have achieved popularity due to their capability to efficiently address both linear and non-linear data. Fundamentally, an SVM model attempts to identify the optimum hyperplane that distinguishes different classes within the feature space. For problems with two-class classification, the SVM algorithm creates a line or hyperplane that partitions the data such that the margin of separation between the classes is maximised. This margin is specified by the closest points to the hyperplane, referred to as the support vectors. These support vectors play a crucial role in the dataset as they directly influence the position and orientation of the hyperplane. When dealing with data that is not linearly separable, SVMs employ the kernel trick, a mathematical technique that transforms the input space into a higher-dimensional space where a hyperplane can successfully separate the classes. The commonly used kernels encompass the linear, polynomial, radial basis function (RBF), and sigmoid kernels [102, 53, 75].

The Random Forest algorithm is a highly flexible and robust machine learning technique employed for both classification and regression. It is classified as an ensemble learning method, functioning by generating numerous decision trees during the training phase. In classification tasks, the output of Random Forest is the class that occurs most frequently among the individual trees. Conversely, in regression tasks, it predicts the mean output value. Technically, Random Forest is based on the principle of decision trees. The algorithm creates multiple decision trees, where each tree is trained on a random subset of the data. This training process, known as bagging or bootstrap aggregating, ensures that each tree in the ensemble has slight variations from the others. By doing so, the risk of over-fitting is minimised, leading to an enhanced overall accuracy of the model [134].

During the generation of the decision trees, the Random Forest algorithm incorporates an element of randomness. Rather than taking into account all characteristics when dividing nodes, the algorithm instead chooses a random subset of features. This element of randomness enhances the resilience of the model and makes it more robust and less prone to bias towards certain features. The Random Forest algorithm has gained recognition for its exceptional accuracy, robustness, and capacity to handle large and higher dimensional datasets. This method proves particularly effective in cases where over-fitting arises, as the ensemble approaches notably lessen this risk. Additionally, the algorithm manages missing data very well and is performant in feature importance, making it quite valuable for exploratory data analysis [2].

Random Forest has some drawbacks despite its strengths. The model can be complex and computationally intensive, especially when dealing with a large number of trees. This complexity can lead to longer training times. Moreover, the ensemble nature of Random Forest causes less interpretability compared to a single decision tree, which can be seen as a limitation where understanding the decision-making process of the model is necessary [97]. Random Forest has various applications in finance. Fraud detection and risk assessment are among the various applications of Random Forest in finance [134, 83]. It also has been used for predicting stock behaviour and creating recommendation systems.

Data Description

Financial data comes in a variety of shapes and forms. The four essential financial data types are fundamental data, market data, analytics, and alternative data. To apply machine learning algorithms to unstructured financial data, we need to parse it and extract valuable information, then store those extractions in a regularised format. The tabular representations of data used in ML algorithms (i.e. table rows) equate to what finance practitioners refer to as bar in bar charts [35]. Time bars which perhaps are the most popular among market practitioners and academics are generated through sampling price information at fixed time intervals. The information usually includes; timestamp, volume-weighted average price, open, high, low, close, and traded volume. Time bars unrealistically process information at a fixed time interval, leading to an exhibition of poor statistical properties [35].

In financial jargon, a tick refers to a change in the price of a security from one trade to the next. To generate tick bars, it is necessary to extract the aforementioned sample variables every time a predetermined number of transactions takes place. This will enable the synchronisation of sampling with a proxy indicator of the arrival of information. For instance, if we wish to generate 100-tick bars, we need to store the 100 price information and then extract the open, high, low, and close values from the observations. Mandelbrot and Taylor [89] found that sampling as the function of transaction numbers exhibits Gaussian distribution properties. In contrast, sampling over a fixed interval may follow a stable Paretian distribution, whose variance is infinite [35]. It should be mentioned that throughout the experiment, tick bars and tick candles are used interchangeably. The sole difference between the two is that the tick candles are colour-coded to reflect any increase or decrease in price.

Average True Range

The average true range (ATR) is a technical analysis indicator that measures market volatility. It decomposes the whole range of an asset price for a specific period. It is typically derived from a moving average of length 14 of a series of true range values and can be calculated on an intra-day, daily, weekly or monthly basis. If the high of the current period is greater than the high of the previous period and the low is lesser than the low of the previous period (referred to as an “outside day"), then the difference between the high and the low will be utilised as the True Range.

In addition, in the case of a gap when the previous close is greater than the current high or the previous close is lower than the current low, or an inside day (i.e. when the current high is below the previous high and the current low is above the previous low), current high less the previous close or the current low less the previous close will be used. The following equations represent the calculation of ATR using High, Low, and previous close ($H, L, C_{previous}$):

$$TR = \max\left[H - L, \left|H - C_{previous}\right|, \left|L - C_{previous}\right|\right] \tag{3.1}$$

$$ATR = \frac{1}{n}\sum_{i}^{n} TR_i \tag{3.2}$$

$$ATR\,\% = \frac{ATR}{current\ price} \tag{3.3}$$

where TR_i is the true range, and n is the time period. In Equation 3.3, ATR%, is the ATR division by the current price of the asset. In Table 3.1, asset price information and the calculation of TR and ATR are represented. Equation 3.3, %ATR, is the ATR division by the current price of the asset.

High	Low	Close	H-L	abs(H-C)	abs(L-C)	TR	ATR
48.70	47.79	48.16	0.91	-	-	0.91	-
48.72	48.14	48.16	0.58	0.56	0.02	0.58	-
48.90	48.39	48.75	0.51	0.29	0.22	0.51	0.67
48.87	48.37	48.63	0.50	0.12	0.38	0.50	0.53
48.82	48.24	48.4	0.58	0.19	0.39	0.58	0.53

Table 3.1: Average True Range Calculation

Experiment

This chapter's objective is to apply the CNN-LSTM network to the generated DC-based summaries of GBPUSD, EURUSD, USDCHF, and USDCAD tick prices to predict the next DC event. The initial dataset comprises of the currency pairs' tick prices from January to August 2019, in comma-separated variables (CSV) format. As we mentioned earlier, a tick price alludes to a change in an asset price from one trade to the next. To generate the tick bars, we will aggregate 50, 100, 200, 500, 1000 data points from the original tick prices of the GBPUSD, EURUSD, USDCHF, and USDCAD currency pairs.

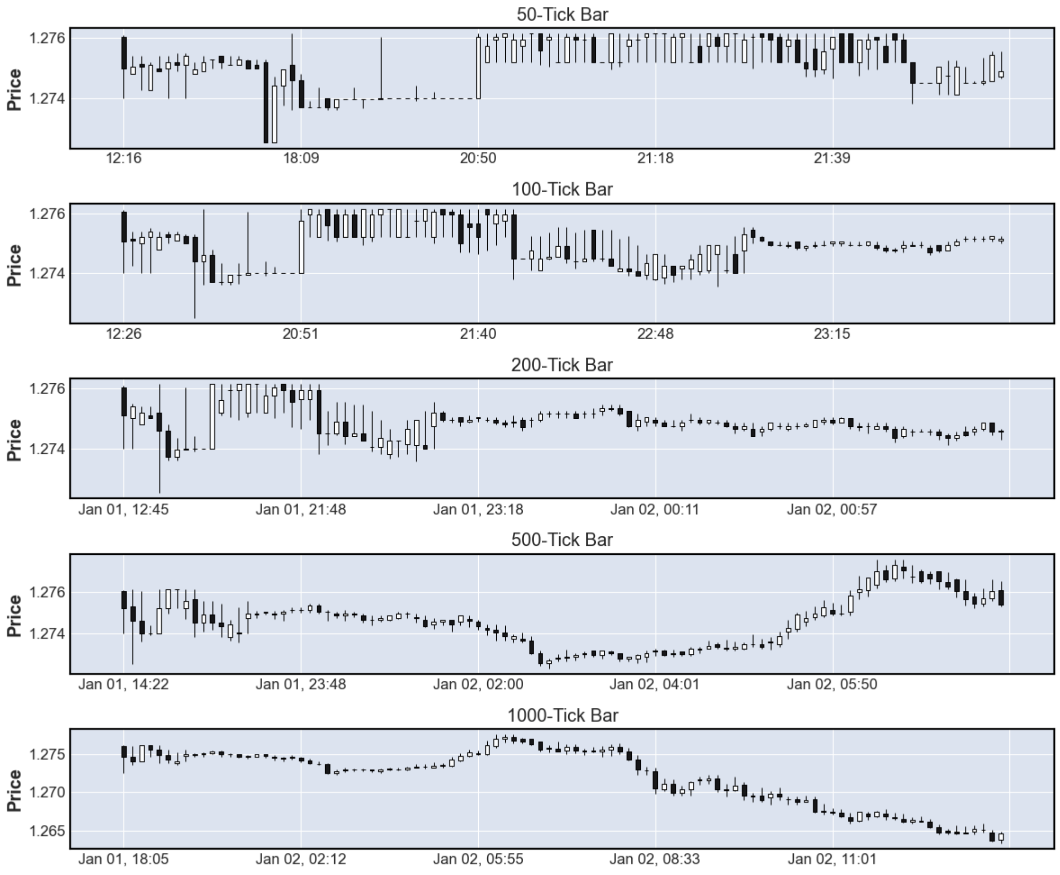


Figure 3.1: The first 100 observations of GBPUSD with a predefined number of tick prices.

Every tick bar has an open, high, low, and close price. The open and close prices correspond to the price of the first and last trade. The high and close prices are the maximum and minimum prices within the range of the predefined number of ticks. Figure 3.1 is the depiction of the generated tick bars/candles from the GBPUSD tick prices with the predefined number of ticks. The tick bar with the least auto-correlation will be used to generate the DC-based summaries. In order to obtain the least auto-correlated tick bar, the Durbin-Watson (DW) statistic was performed on all the currency pairs' tick bars. The DW test is calculated as follows:

$$DW = \frac{\sum_{i=2}^{n} (e_i - e_{i-1})^2}{\sum_{i=1}^{n} e_i^2} \tag{3.4}$$

Where:

- e_i represents the ith residual (difference between observed and predicted values).
- n is the number of observations.

The Durbin-Watson statistic ranges from 0 to 4. A value of around 2 indicates no auto-correlation, a value significantly less than 2 suggests positive auto-correlation and a value significantly greater than 2 suggests negative auto-correlation. Table 3.3 represents the Durbin-Watson results for the tick bars. As the results imply, the 1000 tick-bar has the lowest DW value for GBPUSD, EURUSD, and USDCHF and the 200 tick-bar for the USDCAD pair. The Average True Range will be calculated for the tick-bar with the smallest DW and will then be used as the Directional Change threshold θ.

The rationale for using ATR as the determinant for θ is anchored in its ability to reflect market volatility. Employing a fixed θ might not be appropriate to different market conditions. A market with low volatility might require a smaller θ to be able to capture price movements, whereas a high volatility environment might need a larger θ to filter out noise. ATR adjusts the threshold according to the market volatility and provides a more dynamic threshold. As previously mentioned, the Average True Range (ATR) is a market volatility measure and is typically calculated from the 14-day simple moving average of true range values. After the ATR calculation, DC-based summaries will be generated. Using the past five DC event prices we aimed to predict the next event price. The CNN-LSTM model, as its name implies, consists of a convolutional neural network layer and a long short-term memory layer. Figure 3.2 is the illustration of the employed model.

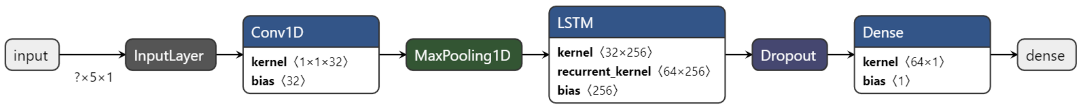


Figure 3.2: CNN-LSTM model

As illustrated in Figure 4.6, the inputs i.e., DC event prices are directed to the one-dimensional convolutional layer. To reduce dimensionality and capture the most significant features, a max-pooling layer is applied. Following the LSTM layer, a dropout layer is incorporated to mitigate the

Parameters	Value
Convolution layer filters	32
Convolution layer kernel size	1
Convolution layer padding	Same
Pooling layer pool size	3
Pooling layer padding	Same
Number of units in LSTM layer	64
LSTM activation function	$tanh$
Dropout rate	0.2
Optimiser learning rate	0.001

Table 3.2: CNN-LSTM Parameters

risk of over-fitting. This is necessary to enhance the generalisability of the model. The number of Convolutional filters, LSTM units and activation function, as well as the Dropout percentage and optimiser learning rate, were determined through hyper-parameter tuning with KerasTuner [96]. Table 3.2 presents the parameters' setting for the CNN-LSTM model.

The DC summaries of the currency pairs were divided into training, validation, and test sets, where 80% of data points constitute the training, and the remaining 20% is the test set. Moreover, 20% of the training set was used as the validation set to prevent data leakage. The training process was performed with the *Adam* optimiser and the mean squared error as the loss function. To evaluate the predictive performance of the model, the mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination (R^2) will be used. The following are the equations for the MAE, RMSE, and R^2 where y_i is the target value and $\hat{y}_i$ represents the prediction.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3.5}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{3.6}$$

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{3.7}$$

The CNN-LSTM model will be trained and validated with the DC summaries of GBPUSD, EURUSD, USDCHF, and USDCAD with an Early-Stopping of Keras callback API. Initially, DC summaries of the GBPUSD pair will be used to train and validate the model on the training and validation sets with respective 4,567 and 1,138 data points. Prediction on the test set, which is considered the out-of-sample set, resulted in a 0.0142 mean absolute error and a 0.0179 root mean squared error. Figure 3.4a represents the prediction of the model on the GBPUSD DC summaries.

As it is observable, the model has reached a reasonably good prediction throughout the summaries with a coefficient of determination of 0.985. The accuracy of prediction has dwindled near the end of the graph. To explore the predictive capability of the CNN-LSTM model within the directional change framework and on the raw tick bars, we applied the identical CNN-LSTM model on the close price of the 1000 tick bar dataset. Training and validating the CNN-LSTM model on the GBPUSD raw 1000 tick bar dataset with the respective number of 14,921 and 3,727 observations resulted in 0.0604 mean-absolute error (MAE), and 0.0697 root mean squared error (RMSE). We then utilised the trained model to perform predictions on the out-of-sample dataset. From Table 3.4b, in the absence of the DC Framework, the coefficient of determination has plummeted from 0.985 to 0.359. Figure 3.4b portrays this noticeable decline in the prediction accuracy of the model. The same steps were applied for EURUSD, USDCHF, and USDCAD currency pairs.

With the suggestion of Table 3.4 and the comparison of Figures 3.5a and 3.5b, an increase in the MAE and RMSE metrics from 0.0188 to 0.0294 and 0.0248 to 0.0368 is discernible. Furthermore, the coefficient of determination (R^2) for EURUSD has decreased from 0.972 to 0.946. Despite capturing the overall trend of the USDCHF, distinguished from Figures 3.6a and 3.6b, metrics altogether corroborate the substantial drop in the accuracy of the CNN-LSTM model. Both MAE and RMSE have risen from 0.0301 to 0.0466 and from 0.0387 to 0.0516. The R^2 has declined from 0.865 to 0.772. Figure 3.7a substantiates the prediction accuracy of the CNN-LSTM model within the DC framework. The model captured the overall trend correctly and predicted more than 6000 observations with a coefficient of determination (R^2) of 0.973. In Figure 3.7b the performance of the model in predicting nearly three times more observations without DC framework plummeted to 0.548. For the USDCAD, MAE and RMSE have surged from 0.0182 to 0.0989 and from 0.0221 to 0.1094. R^2 has plunged from 0.973 to 0.548.

We observed that the CNN-LSTM model, within the DC framework, outperforms itself by a considerable margin. Consequently, applying the CNN-LSTM model within the DC framework for the GBPUSD, EURUSD, USDCHF, and USDCAD currency pairs enhances the accuracy of the prediction in all performance metrics. It is concluded from the results that applying the CNN-LSTM architecture within the Directional Change framework improves the accuracy of prediction for high-frequency Forex data. Support Vector and Random Forest regression, two widely used machine learning techniques in financial forecasting, were also utilised to compare to the CNN-LSTM model. Both models' hyper-parameters were tuned with RandomisedSearchCV [98] and used in the same fashion as the CNN-LSTM with and without the DC framework.

It is inferred from Table 3.4 that Support Vector, and Random Forest regression failed to perform an acceptable prediction with significantly high error and negative coefficient of determination

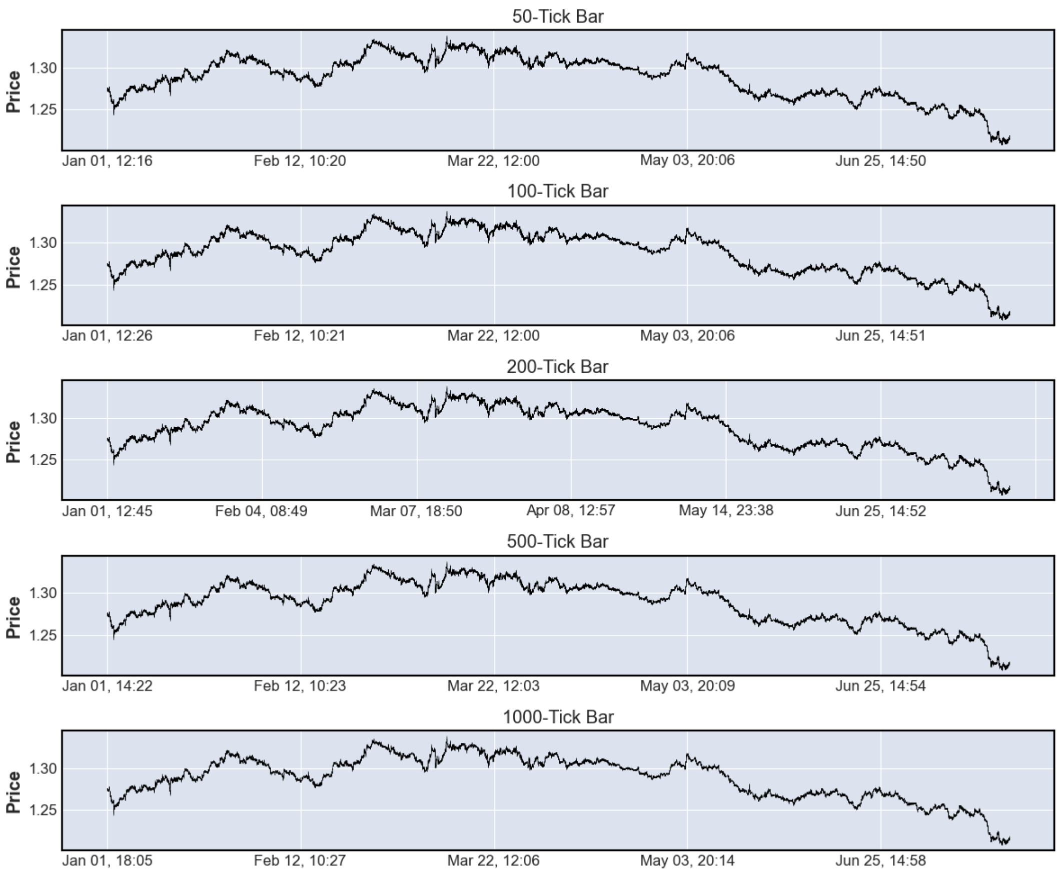
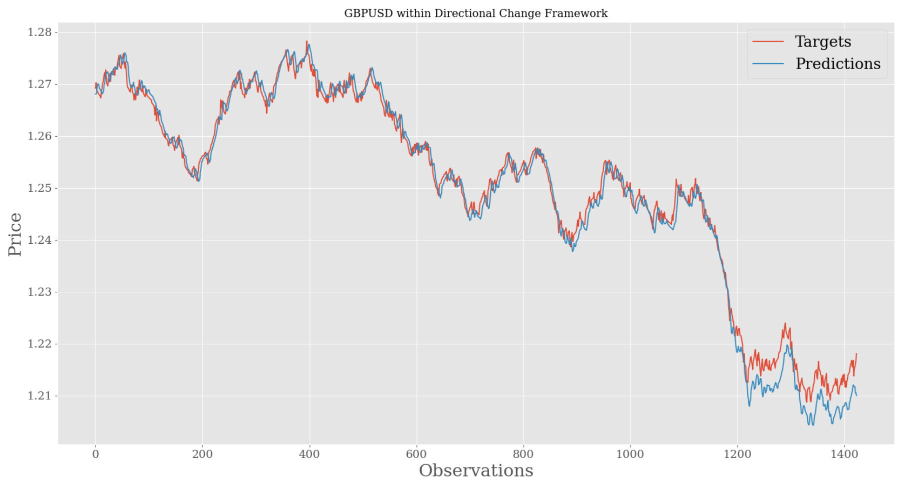


Figure 3.3: GBPUSD generated tick bars.

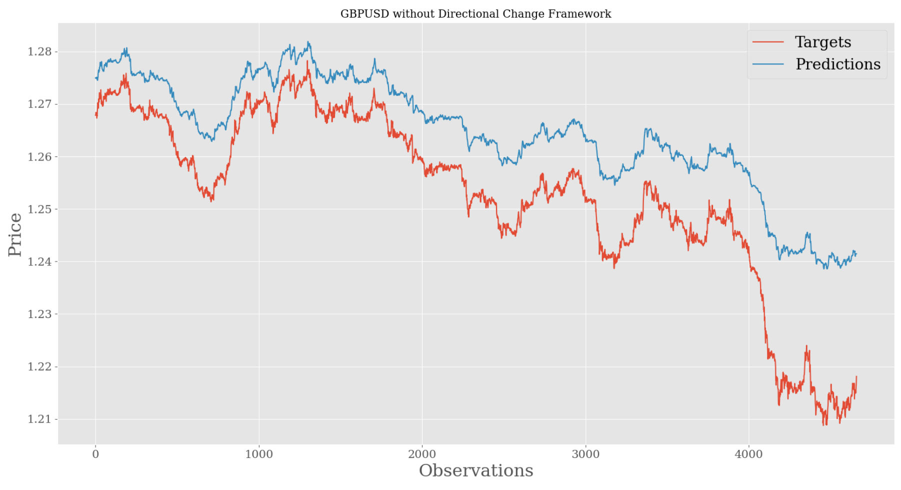
Currency Pair	50 Tick-bar	100 Tick-Bar	200 Tick-Bar	500 Tick-Bar	1000 Tick-Bar
GBPUSD	2.294	2.163	2.126	2.094	**2.092**
EURUSD	2.591	2.447	2.320	2.180	**2.131**
USDCHF	2.289	2.181	2.158	2.138	**2.112**
USDCAD	2.678	2.511	**2.379**	2.386	2.461

Table 3.3: Durbin-Watson Statistic of the Currency Pairs

(R^2). Summarily, the tick bars were created from raw tick prices and the least auto-correlated were determined using the Durbin-Watson statistic. Next, the least auto-correlated tick bars were used to calculate the ATR value, which then was used as the Directional Change threshold θ. Then, the DC summaries of the tick bars were generated. Finally, the proposed model was applied to the mentioned DC summaries of all the currency pairs as well as their raw tick bars to investigate the performance of the CNN-LSTM model with and without the DC framework.

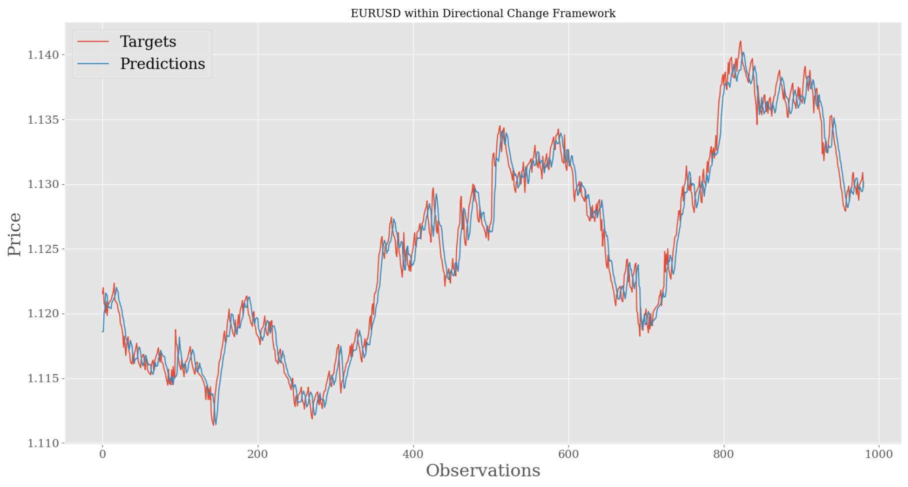


(a) CNN-LSTM prediction on GBPUSD DC summaries

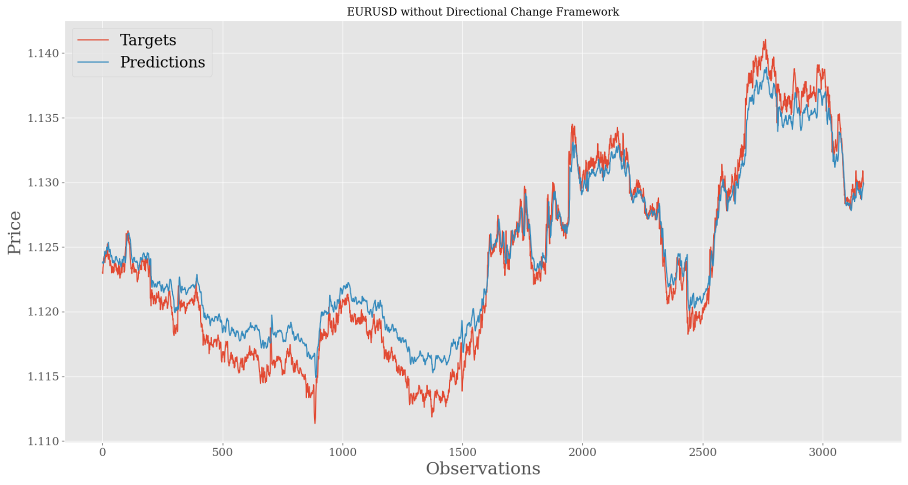


(b) CNN-LSTM prediction on GBPUSD 1000 tick bars

Figure 3.4: CNN-LSTM results within DC framework and on raw tick bars for GBPUSD

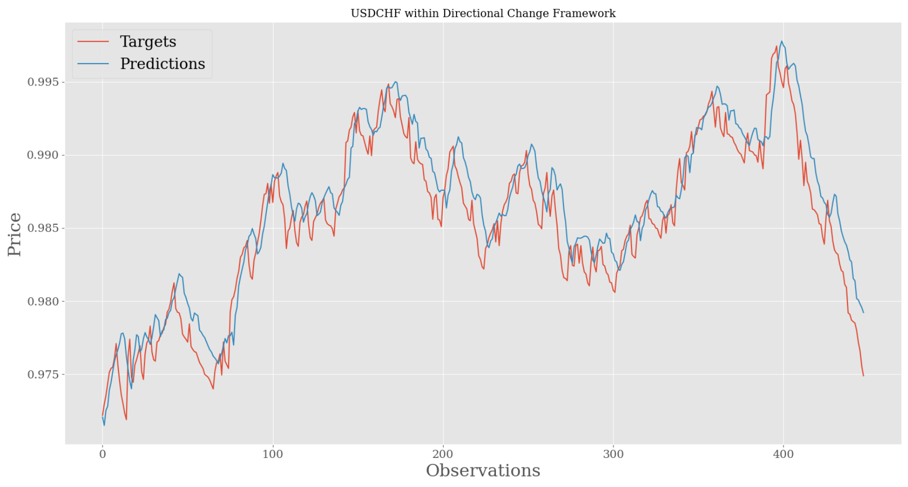


(a) CNN-LSTM prediction on EURUSD DC summaries

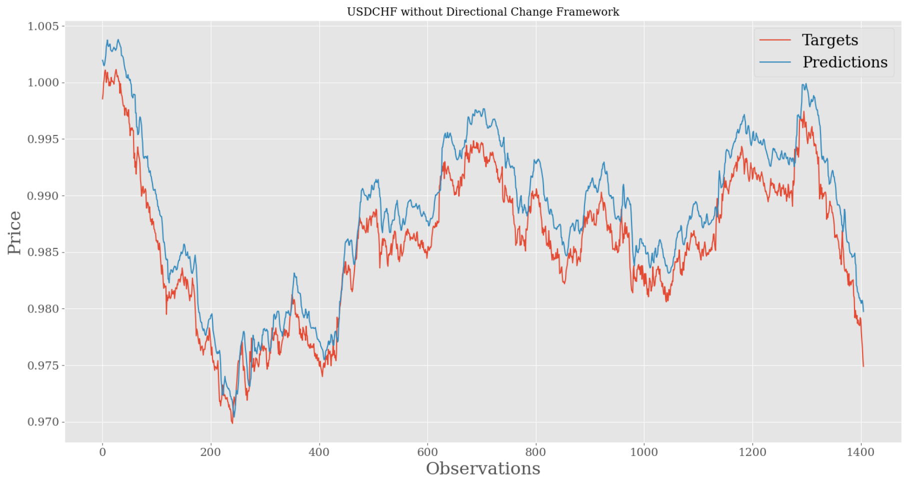


(b) CNN-LSTM prediction on EURUSD 1000 tick bars

Figure 3.5: CNN-LSTM results within DC framework and on raw tick bars for EURUSD

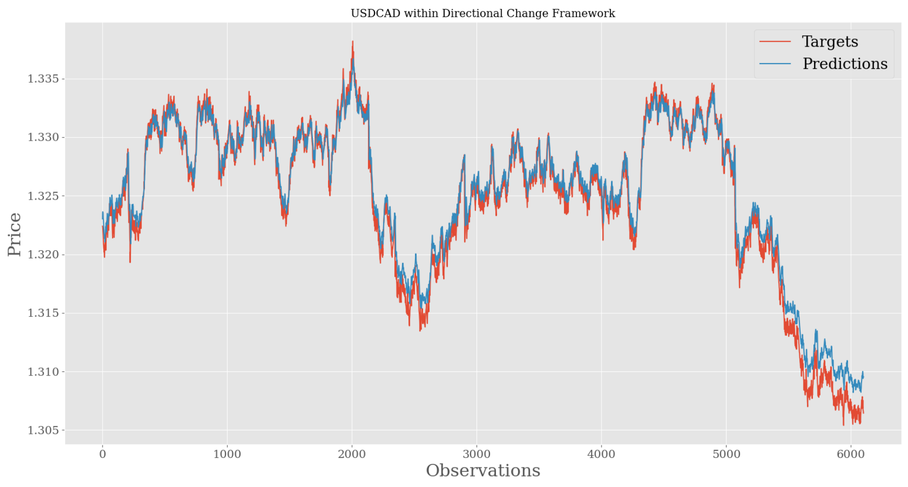


(a) CNN-LSTM prediction on USDCHF DC summaries

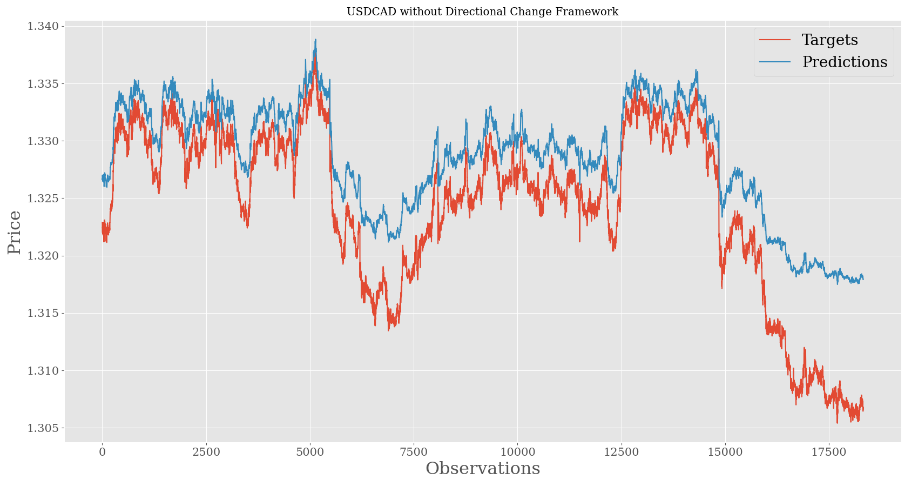


(b) CNN-LSTM prediction on USDCHF 1000 tick bars

Figure 3.6: CNN-LSTM results within DC framework and on raw tick bars for USDCHF



(a) CNN-LSTM prediction on USDCAD DC summaries



(b) CNN-LSTM prediction on USDCAD 200 tick bars

Figure 3.7: CNN-LSTM results within DC framework and on raw tick bars for USDCAD

	CNN-LSTM			Support Vector Regression			Random Forest Regression		
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2
GBPUSD	0.0142	0.0179	0.985	0.2909	0.3604	-1.8945	0.2841	0.3543	-1.7975
EURUSD	0.0188	0.0248	0.972	0.5746	0.6047	-8.3019	0.5993	0.6294	-9.0758
USDCHF	0.0301	0.0387	0.865	0.0917	0.1149	0.0035	0.1016	0.1220	-0.1219
USDCAD	0.0182	0.0221	0.973	0.5791	0.5914	-23.1734	0.5488	0.5579	-30.1266

(a) Results within DC Framework

	CNN-LSTM			Support Vector Regression			Random Forest Regression		
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2
GBPUSD	0.0604	0.0697	0.359	0.3435	0.3942	-3.1636	0.2501	0.3158	-1.6722
EURUSD	0.0294	0.0368	0.946	0.5107	0.5409	-6.9423	0.6003	0.6304	-9.7862
USDCHF	0.0466	0.0516	0.772	0.1029	0.1291	-0.0081	0.1133	0.1382	-0.1549
USDCAD	0.0989	0.1094	0.548	0.5812	0.5948	-21.071	0.5625	0.5725	-28.2334

(b) Results without DC Framework

Table 3.4: Prediction Accuracy Results

3.3 Results

The present section details a research endeavour that focuses on the prediction of events in financial markets through the utilisation of a CNN-LSTM model within the framework of Directional Change (DC). The DC framework is responsible for converting time series price data into an intrinsic time curve by utilising a predetermined threshold value known as theta (θ) to identify significant price changes called events. The threshold value is established with the aid of the Average True Range (ATR) indicator. The research methodology involves comparing the performance of the CNN-LSTM model within and without the DC framework, using raw tick bars.

The research study encompasses several essential steps, which include generating tick bars from tick prices of major currency pairs in the Forex market, identifying the least auto-correlated tick bars through the Durbin-Watson statistic, computing the ATR for these least auto-correlated tick bars to determine the threshold value (θ), generating DC summaries based on the threshold value, training and validating the CNN-LSTM model using the DC summaries and raw tick bars separately, and evaluating the predictive capabilities of the CNN-LSTM model using metrics such as mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination (R^2). The results of the research indicate that the CNN-LSTM model performs significantly better within the DC framework for all tested currency pairs, displaying improvements in MAE, RMSE, and R^2 metrics.

The model's accuracy is augmented when using DC summaries in comparison to raw tick bars. Comparative analysis with Support Vector and Random Forest regression models also validates the superiority of the CNN-LSTM model in terms of predictive accuracy within the DC framework.

Further research would apply other networks, e.g. GRU, and BiLSTM with a more complex architecture on different currency pairs and financial assets. It would be of importance and interest to explore ways to determine the Directional Change threshold dynamically.

Chapter 4

Deep-learning Driven Forecasting with Autoencoders and Variational Autoencoders

4.1 Introduction

The financial market analysis is a highly challenging task. The profit and success of the financial investments are closely tied to the accuracy of prediction within the market. However, the prediction of financial markets is a challenging task due to factors that affect the market's direction. Noise, time-varying distribution, and non-linearity are among the reasons that challenge the effort. The main two approaches to financial market analysis are trend and price prediction. The market trend prediction is considered a classification task, while price prediction is a regression problem. Two major approaches, statistical models and machine learning methods, are widely used to address stock price prediction.

Conventional statistical models and machine learning approaches can be fundamentally distinct when it comes to their underlying assumptions and adaptability in the context of financial time series. Classical statistical models, such as ARIMA, often are based on the assumption of linearity and stationarity in the data. On the contrary, machine learning methods are particularly designed to capture and model more complex, non-linear relationships in the data. This property has proven to be advantageous in financial time series analysis, where the market displays non-linear and non-stationary behaviours. For this reason, while not all financial time series adhere to the mentioned characteristics, the flexibility and robustness of machine learning methods in dealing with different patterns have significantly broadened the scope and effectiveness of financial analysis. Linear models ARMA and ARIMA have been widely used to predict financial time series [125]. Non-linear models such as neural networks, support vector regression, and hybrid algorithms have also been employed to predict stock prices with high accuracy [10], [55], [30]. At the forefront of innovative methodologies, autoencoders emerge as unsupervised learning algorithms designed to unravel complex data representations, offering potential solutions to the intricacies of financial market analysis.

In this chapter, we investigate the employment of autoencoders and variational autoencoders for denoising and forecasting financial data, comparing their performance with recurrent and convolutional models. The systematic exploration covers concepts of autoencoders, experimental data, methodology, and a comprehensive discussion of results. Our findings reveal that while autoencoders and variational autoencoders are effective in reducing noise from financial time series, CNN-LSTM (Convolutional Neural Network - Long Short Term Memory) and CNN-BiLSTM (Convolutional Neural Network - Bidirectional Long Short Term Memory) models demonstrate superior performance. These advanced neural network architectures, with their ability to capture both spatial and temporal dependencies, prove more adept at handling the complexities inherent in financial time series, outperforming in both denoising and forecasting accuracy.

4.2 Methodology and Experimentation

Autoencoders introduced by Hinton and Salakhutdinov [62], as it was described in 2.7, are unsupervised learning algorithms designed for representation learning that attempts to copy inputs to the outputs. Autoencoders are inherently unable to create perfect replications of their inputs. This limitation has become a strength since it forces the model to learn the most salient features of the data, rather than merely memorising it. As a consequence, autoencoders surpass tasks where the extraction of underlying patterns and features is critical. Autoencoders have found various applications across different fields. In dimensionality reduction tasks, they distil high-dimensional data into lower dimensions. In image processing, they are used to compress and denoise image data. Moreover, they are widely used in feature extraction, recommendation systems, and sequence-to-sequence prediction tasks.

Given the complexity of financial time series forecasting and the proven efficacy of deep learning models, we propose a deep learning framework tailored for time series prediction. Our model is designed to process sequences of financial time series data, each characterised by open, high, low,

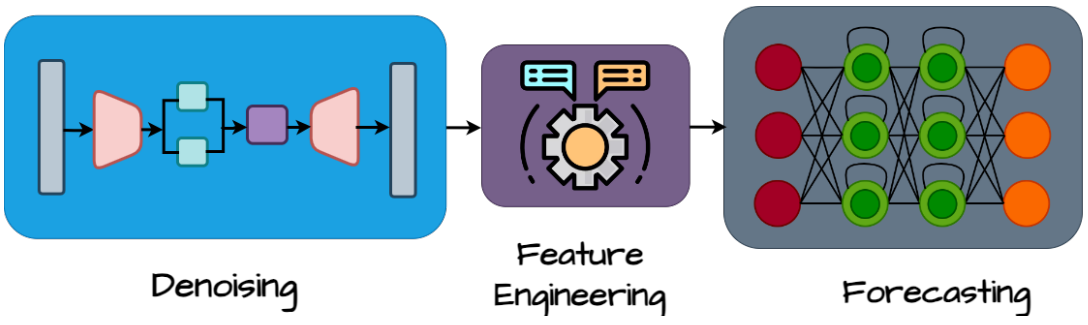


Figure 4.1: Proposed Model

close, and volume features. The output is a sequence where each time lag[1] to predict the closing price for the subsequent intervals. Our approach sets our model as a tool for multi-step-ahead forecasting, capable of handling and generating predictions from multivariate financial data.

Researchers use autoencoders in a variety of problems, including dimensionality reduction, image compression and denoising, feature extraction, recommendation systems and sequence-to-sequence prediction, to name a few. Considering the complexity of financial time series forecasting and deep learning applications in addressing such problems, we propose a deep learning framework to perform prediction. Our proposed model uses an input sequence of financial time series where each time lag has multivariate features: open, high, low, close and volume. The output of the model will be a sequence where each time lag corresponds to the closing price of the succeeding intervals. Consequently, the proposed model is considered to be a multi-step-ahead forecasting with multivariate data.

Data Description

In the experiment, the five companies' minute-by-minute price information, namely AAPL, GOOG, AMZN, FB, and EBAY are collected over one month from June 2021 to July 2021 from AlphaVantage API[2]. The price information consists of open, high, low, close prices as well as the volume. Open price is defined as the first price across all markets for the minute. Similarly, the high, low, and close prices are respectively the highest, lowest and close prices across all markets for the minute. The volume is the total volume of trades across all markets for the minute.

Before feeding the data into the forecasting model, it passes through the denoising block. This step denoises the input time series of the five stocks with a variational autoencoder network. A variational autoencoder is applied to the financial time series of data with open, high, low, close, and

[1] Time lag refers to the interval or delay between two consecutive data points in a time series. In financial analysis, it often represents the period used to predict future values based on past observations. For instance, a time lag of one day in stock price data would use the previous day's price to forecast today's price

[2] https://www.alphavantage.co/documentation/

volume as the input features to remove noise by generating the latent dimension from the original features. Consequently, following the application of the variational autoencoder, the financial time series will have the original and reconstructed features combined.

Data Normalisation

To preprocess the input financial time series, we first normalise the inputs using feature scaling to ensure that the multivariate input and multi-step predictions lie in the range (0, 1). Since the models in the forecasting block are designed with the recurrent neural network, Long Short-Term Memory (LSTM), the inputs are required to be three-dimensional data of shape $[n_s, n_t, n_f]$, where n_f is the number of training samples, n_t is the input window size and n_f is the number of input features. To normalise the inputs, we use the Scikit-Learn library. MinMaxScalar is a class for data normalisation and requires the input data to be two-dimensional.

To make the input data of shape $[n_s, n_t, n_f]$ compatible with the input size of the MinMaxScalar, we reshape the data of shape $[n_s, n_f]$ into separate groups for each feature. Following the reshaping, the number of separate groups will be equal to n_f, with two-dimensional data of shape $[n_s, 1]$. After normalisation, we merged them into their original two-dimensional shape of $[n_s, n_f]$. In this manner, we will have n_f unique scalar objects for each feature in the dataset. Consequently, it will already be normalised when the data is expanded to have the third dimension, i.e., the number of time steps.

Data Preparation

To prepare the normalised input data for the forecasting model, it's crucial to restructure it into an overlapping sliding window format. The normalised dataset, originally having the shape $[n_s, n_f]$, where n_s denotes the number of samples and n_f the number of features, is transformed to facilitate time series forecasting. Specifically, we utilise n past time steps and m time steps ahead to perform predictions. For instance, to predict the closing price of a stock at future times $t+1$, $t+2$, $t+3, \ldots,$ $t+m$, the input to the model comprises the past data points x_t, x_{t-1}, $\ldots$, x_{t-n}, while the model outputs future values y_{t+1}, y_{t+2}, $\ldots$, y_{t+m}. Consequently, this restructuring results in the dataset having the shape $[n_s, n_t, n_f]$.

In this project, n_t, the length of the sliding window, is set to seven minutes, meaning that we use data from seven past time steps to predict the closing price for the next seven steps. Detailed information about the datasets includes the source of the data, the range of dates covered, and any preprocessing steps such as normalisation or handling of missing values. Our preprocessing steps ensure that the data is clean, relevant, and structured in a way that optimises the model's ability to learn from temporal patterns. By providing this detailed information on dataset preparation, we aim to eliminate any ambiguities, allowing readers to fully grasp the methodology and context of the forecasting model's data structure.

Proposed Model

LSTM-VAE Denoising Block

For the purpose of noise removal from the input financial time series, we use a Long Short-Term Memory Variational Autoencoder (LSTM-VAE). A variational autoencoder, analogous to a standard autoencoder, consists of an encoder and a decoder network. The inputs at each time step are mapped into latent space with a Gaussian distribution with $\mu = 0$ and $\sigma = 1$. Then, the decoder network estimates the expected distribution of the inputs from the latent representation. In other words, the encoder in variational autoencoder first projects an instance into a mean and standard deviation of a latent variable and then samples from the latent distribution. Subsequently, the decoder decodes the samples into a mean and standard deviation of the output variables and generates samples from the output distribution. The reconstruction of the input features contributes to removing noise and anomalies. Therefore, collecting the original inputs and their reconstruction will create the input for the feature engineering block. Figure 4.2 and 4.3 are the illustrations of the encoder and decoder network of the LSTM-VAE network.

The encoder model comprises an input layer and two LSTM layers with a dropout layer in between. The first LSTM layer has 128 units, and the second one has 64 units. The dropout layer with a ratio of 0.1 prevents the encoder model from over-fitting. Next, there exist two dense layers for the mean and variance of the variational autoencoder—these two dense layers output to a Lambda layer. The ensemble of the mentioned layers creates the encoder model of the variational autoencoder. The output of the encoder network enters the repeat vector layer of the decoder network and then passes through two LSTM layers, connected with a dropout layer with the same ratio as the encoder. The decoder network mirrors the encoder model; therefore, the first LSTM layer has 64 units and the second one 128 units. Eventually, the decoder model ends with a Time-Distributed layer. We train the variational autoencoder with *Adam* [77] optimiser for 40 epochs. The denoised features will be concatenated with the original features forming ten features to feed into the feature engineering and selection block.

Feature Engineering Block

The output from the previous step is given to the feature engineering block to extract a new set of features. Since we deal with time-series data, we need to employ time-series feature extraction

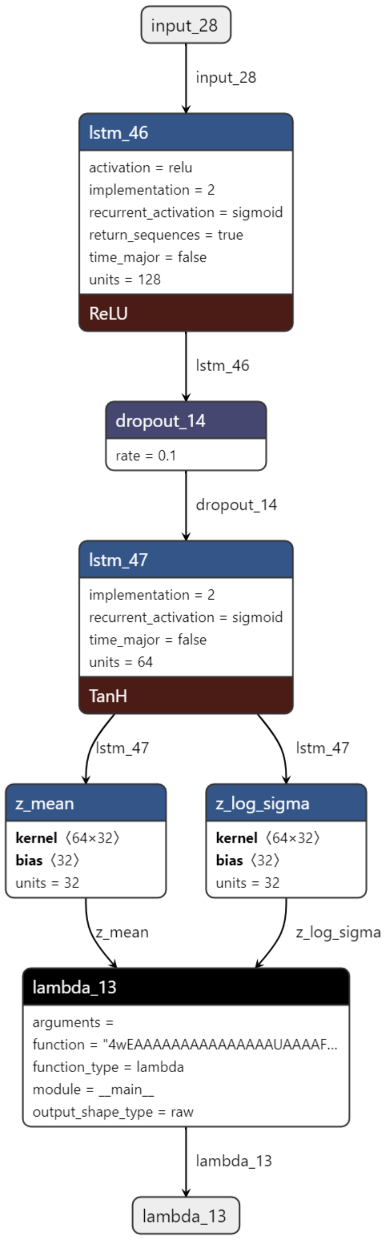


Figure 4.2: Denoising LSTM-VAE encoder

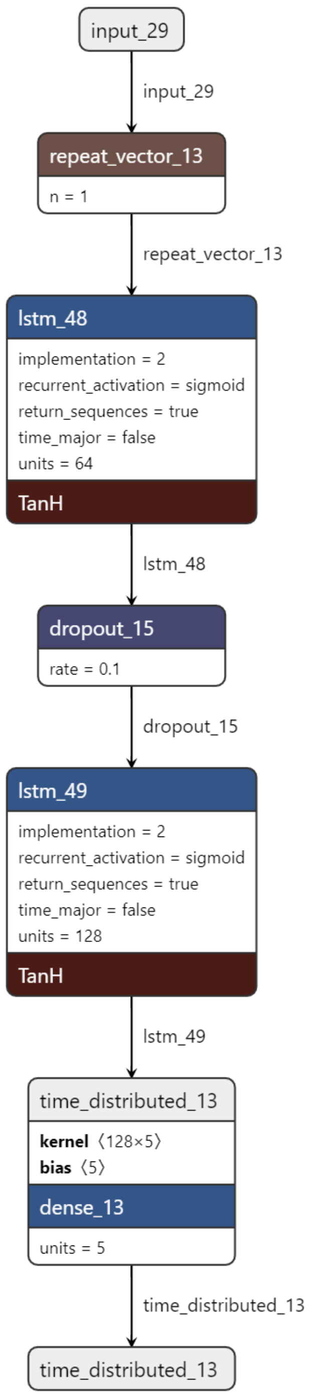


Figure 4.3: Denoising LSTM-VAE decoder

techniques to bring out informative features. Tsfresh [3] is a Python package that automatically computes a vast number of time series features. It has methods to evaluate the power and significance of such features for regression and classification problems. The extracted features involve basic and advanced characteristics of the time series.

Tsfresh extracted more than seven hundred features for each stock, and some extracted features include the mean, standard deviation, kurtosis, and skewness, to name a few. Before proceeding, we need to impute and replace the null values in the features. Next, we rank the features according to their importance using the SelectKBest, a Scikit-Learn's class for selecting features according to the k highest scores. Due to the minor improvement in accuracy measures, we only used the twenty features of the SelectKBest algorithm.

Forecasting Block

The forecasting block of the proposed model consists of a stacked LSTM autoencoder, a stacked BiLSTM autoencoder, a CNN-LSTM, and a CNN-BiLSTM model. The convolutional layers in CNN-LSTM and CNN-BiLSTM are one-dimensional convolutional neural networks. A one-dimensional CNN layer creates a convolutional kernel that convolves with the layer input over a single spatial or temporal dimension. The stacked LSTM autoencoder is the reference model in this work. The LSTM autoencoder is an encoder-decoder LSTM architecture suited for sequence-to-sequence problems.

A sequence-to-sequence (seq2seq) problem takes in a sequence and predicts an output sequence. The sequence-to-sequence prediction problems are challenging since the length of the input and output sequences are not fixed. LSTM autoencoders have proven to be effective for the seq2seq problems. In such a problem, the encoder compresses the inputs into a fixed-length representation, and the decoder uses it to predict the output sequences.

The detailed structure of the stacked LSTM autoencoder is represented in Figure 4.4. In the output layer, the TimeDistributed layer is a wrapper function to ensure that the Dense layer inside is applied independently to the output of each time step. Hence, the network's output dimension will be the same as the input dimension, i.e., $[n_s, n_t, n_f]$. Here, predicting the stock closing price for a seven-step-ahead prediction will output a sequence with the shape $[1, 7, 1]$. The stacked LSTM autoencoder takes a sequence of features with n lags and predicts the outputs of a sequence of length m. Each lag of the input sequences entails multiple features, including the open, high, low, close and volume and the features from the feature engineering step. Therefore, the task is to perform

[3] https://tsfresh.readthedocs.io/en/latest/

multi-step forecasting with multivariate input data. Additionally, the performance of the stacked LSTM autoencoder will be investigated against a stacked BiLSTM autoencoder, CNN-LSTM, and CNN-BiLSTM models. Figures 4.5, 4.6, and 4.7 illustrate the detailed structure of the forecasting models.

The structure of the stacked BiLSTM autoencoder is similar to the stacked LSTM autoencoder, except the LSTM layers are replaced with BiLSTM layers. Bidirectional LSTM or BiLSTM is an extension of the LSTM models described in 2.5, in which two LSTM layers are applied to the input data. In the first step, an LSTM layer is applied to the input sequences in the forward direction. In the second step, another LSTM layer is applied to the input sequence reversely, in the backward direction [114]. Employing the LSTM layer twice improves learning long-term dependencies and, consequently, improves the model's accuracy [8].

For this reason, we also investigate the advantage of the BiLSTM over the LSTM in our CNN-LSTM and CNN-BiLSTM forecasting models. The CNN layer in both CNN-LSTM and CNN-BiLSTM is a one-dimensional CNN with thirty-two filters and a kernel size of 1. The padding is set to 'same', and the 'use-bias' parameter is true. The CNN layer is followed by a 'MaxPooling1D' layer with a pool size of three. CNN-LSTM model has an LSTM layer with 32 units and a Rectified Linear Unit as the activation function. A similar setting is used for the CNN-BiLSTM. The final two layers are a dropout with a 0.2 rate and a dense layer with seven units corresponding to the seven step-ahead predictions.

Predictive Performance

The predictive performance of the proposed models has been validated with multiple experiments. For the experiment, we used the minute-by-minute historical prices of five companies in the top 50 stocks on the NASDAQ exchange to evaluate our model. The companies are AAPL, AMZN, GOOG, FB, and EBay. Separating the time-series data into training and validation sets for model evaluation and selection is a challenging task. Methods such as k-fold cross-validation are not applicable since they do not preserve the temporal nature of time-series data [121].

For this reason, at each iteration, the training data will be split into two sets so that the validation data chronologically succeeds the training data. In each iteration, the first m months constitute the training data and the final n form the validation data, where $n < m$ [114]. In order to evaluate the predictive performance of the proposed models, we used Root Mean Squared Error ($RMSE$), coefficient of determination (R^2), and the Mean Absolute Error (MAE). Following are the metrics equations.

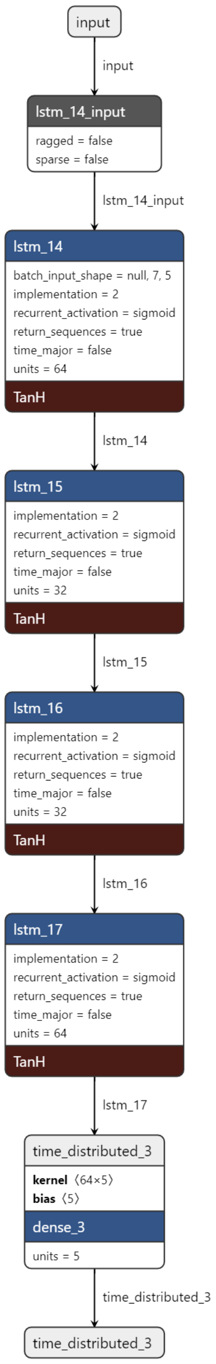


Figure 4.4: LSTM-AE forecasting model

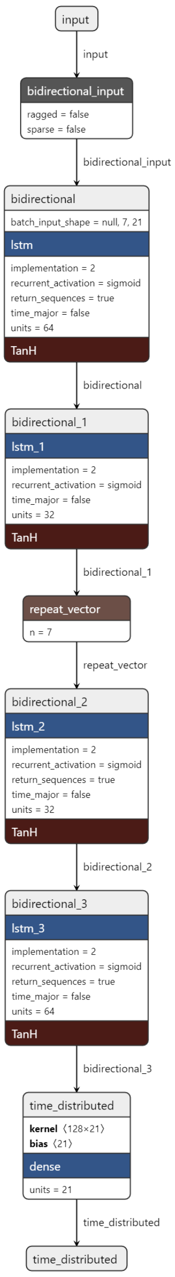


Figure 4.5: BiLSTM-AE forecasting model

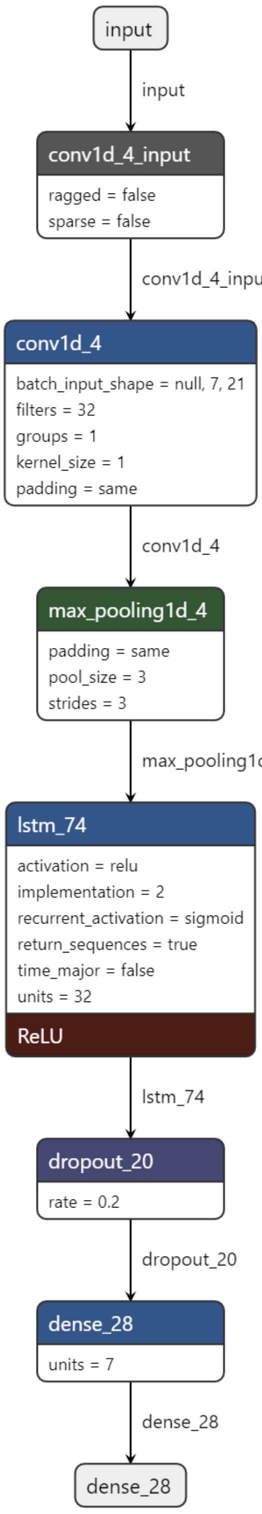


Figure 4.6: CNN-LSTM forecasting model

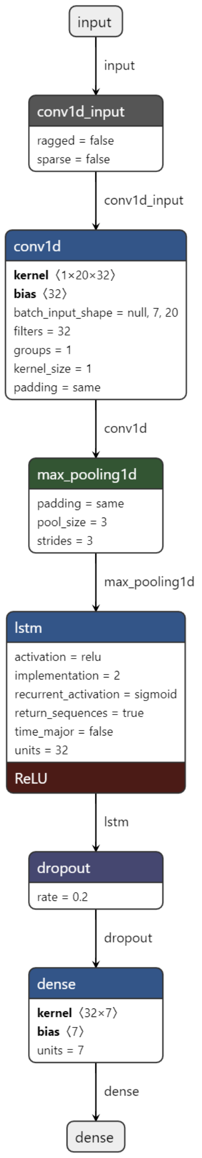


Figure 4.7: CNN-BiLSTM forecasting model

Ticker	Model	RMSE	MAE	R^2
AAPL	LSTM-AE	0.5337	0.0604	0.8081
	BiLSTM-AE	0.8594	0.0512	0.8423
	CNN-LSTM	0.5874	0.0483	0.8545
	CNN-BiLSTM	0.8965	0.0423	0.8978
GOOG	LSTM-AE	2.7854	0.0521	0.8147
	BiLSTM-AE	1.9676	0.0500	0.8565
	CNN-LSTM	2.5245	0.0489	0.8763
	CNN-BiLSTM	2.7595	0.0471	0.9158
AMZN	LSTM-AE	2.4194	0.0665	0.7965
	BiLSTM-AE	4.004	0.0623	0.8497
	CNN-LSTM	5.0073	0.0515	0.8521
	CNN-BiLSTM	3.5658	0.0557	0.9265
FB	LSTM-AE	0.8589	0.0587	0.8341
	BiLSTM-AE	0.8148	0.0564	0.8425
	CNN-LSTM	0.7653	0.5580	0.9331
	CNN-BiLSTM	0.8465	0.0478	0.9451
EBAY	LSTM-AE	0.0645	0.0658	0.7895
	BiLSTM-AE	0.0652	0.0549	0.8265
	CNN-LSTM	0.0614	0.0538	0.8897
	CNN-BiLSTM	0.0686	0.0529	0.9469

Table 4.1: Prediction Accuracy Results (Test set)

4.3 Results

Within the forecasting block, four distinct models were examined: a stacked LSTM autoencoder, a stacked Bidirectional LSTM autoencoder, a CNN-LSTM, and a CNN-BiLSTM. The LSTM-AE was trained using an Adam optimiser and mean-squared error loss, undergoing 200 epochs with early stopping on denoised and feature-engineered AAPL data. Post-training, the model achieved 0.0604 for Mean Absolute Error (MAE) and 0.8081 for Coefficient of Determination (R^2). The subsequent model, akin to LSTM-AE but incorporating Bidirectional LSTM, underwent similar training, resulting in MAE of 0.0512 and R^2 of 0.8423. The addition of the Bidirectional LSTM layer notably enhanced MAE from 0.0604 to 0.0512 and R^2 from 0.8081 to 0.8423. Remarkably, the one-dimensional Convolutional Neural Network showed to be a robust architecture for time-series data.

Upon training CNN-LSTM with AAPL data, it achieved 0.0483 for MAE and 0.8545 for R^2. The ultimate model in the forecasting block, CNN-BiLSTM, achieved MAE of 0.0423 and R^2 of 0.8978. The model's forecasting outcomes showcased CNN-BiLSTM's superiority over other models. Notably, consistent enhancement was observed across forecasting models: LSTM-AE, BiLSTM-AE, CNN-LSTM, and CNN-BiLSTM. Notably, the combination of Convolutional Neural Networks and Long Short-Term Memory demonstrated superior performance compared to the autoencoder model. This enhancement trend was mirrored across metrics for GOOG, AMZN, FB, and EBAY.

Chapter 5

Transfer Learning in Financial Forecasting, Encoding Time-series to Images

5.1 Introduction

In the intricate realm of financial markets, accurate forecasting stands as a pivotal cornerstone for making informed decisions, mitigating risks, and formulating effective strategic plans. Among the various dimensions of forecasting, the task of predicting volatility holds particular significance. This is especially true in the context of the Volatility of Volatility Index (VVIX), which offers insights into market sentiment and uncertainty dynamics. However, the domain of financial time series forecasting has lagged in adopting cutting-edge techniques compared to fields like Natural Language Processing and Computer Vision, which have experienced remarkable advancements.

While modern methodologies from Natural Language Processing (NLP) and CV have reshaped various domains, the intricate characteristics of financial time series data have posed unique challenges, hampering the direct transfer of techniques from related domains. The financial forecasting domain deals with highly complex, often non-linear data that is influenced by a myriad of unpredictable factors, including economic indicators, political events, and market sentiment. Additionally, the financial industry's strict regulatory environment and the high cost of inaccuracies in predictions add layers of complexity. These factors make the direct application of techniques successful in NLP and Computer Vision more challenging, necessitating a more cautious and tailored approach to innovation in financial forecasting.

This disparity between the potential of advanced techniques and their application in financial forecasting has created an urgent need for innovative approaches capable of bridging the gap between finance and modern deep learning methodologies.

The motivation driving this research is rooted in several fundamental observations within the realms of financial forecasting and the rapid progress in deep learning. The complexity inherent in financial sequential data has inhibited the evolution of forecasting methodologies, leaving it disconnected from the strides made in NLP and CV. This disconnection arises from the intricate nature of financial time series data, which requires tailored approaches due to its unique characteristics. In response, two pioneering approaches have emerged as potential solutions: Gramian Angular Fields and Markov Transition Fields. These techniques ingeniously transform time series data into image representations, capturing temporal information while aligning with the image-centric advancements in CV. Furthermore, the sophistication of computer vision models, such as Residual Neural Networks (ResNet), and the utility of pre-trained networks offer significant gains in feature extraction and generalisation.

Among the crucial topics in finance, volatility forecasting, especially regarding the VVIX index, holds a prominent position. Given its impact on risk assessment, trading strategies, and portfolio management, accurate VVIX forecasting becomes paramount. By delving into the convergence of time series image encoding, pre-trained networks, and deep learning techniques, a novel pathway emerges for refining VVIX forecasting accuracy. Such a synthesis possesses the potential to enrich the finance industry with more precise insights and predictive capabilities. Our research embarks on a comprehensive exploration of deep learning-based VVIX forecasting, powered by time series image encoding and a hybrid ResNet-LSTM model, guided by the following question:

- *To what extent does the hybrid ResNet-LSTM architecture capture temporal dependencies and feature hierarchies, contributing to more robust VVIX predictions?*

This chapter progresses to provide an intricate analysis of the methodology, experiment, and results that collectively elucidate the potential of our approach in the context of financial forecasting.

5.2 Methodology and Experimentation

VVIX Index

The CBOE[1] VVIX functions as an index that assesses the volatility of volatility. Its core objective is to gauge the anticipated volatility of the VIX's 30-day forward price. The forward price, in this context,

[1] Chicago Board Options Exchange

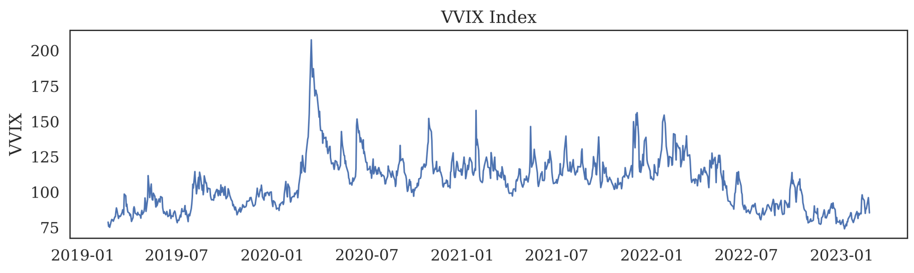


Figure 5.1: VIX Volatility

alludes to the hypothetical value of a VIX futures contract, which would expire in 30 days. The VVIX, while not precisely identical to the expected volatility of the VIX itself, shares a close relationship with the latter due to the tracking of nearby VIX futures to the VIX. The oscillation of VVIX between 2019 to 2023 is depicted in Figure 5.1. The methodology used to compute VVIX is analogous to that of VIX. It is obtained from the price of a portfolio of liquid at[2] and out-of-money[3] VIX options. This portfolio can be utilised to manage volatility risk linked to exposures to VIX and to exploit the risk premium between the expected and realised volatility of VIX forward prices. Although the availability of VIX options with a 30-day expiration is generally limited, the computation of VVIX-like values can be achieved by utilising VIX options that expire at two distinct dates within a 30-day time frame. The determination of VVIX is subsequently performed through interpolation founded on said values. Following is the calculation of VVIX from VIX options prices at each expiration using the VIX formula:

$$VVIX = \frac{2}{T}\sum \frac{\Delta K_i}{K_i^2} e^{RT}\, Q(k_i) - \frac{1}{T}\left[\frac{F}{K_0} - 1\right]^2 \tag{5.1}$$

where:

T	Time to expiration
F	Forward index level
K_0	First strike below the forward index level F
K_i	Strike price of i^{th}out-of-money option
ΔK_i	Interval between strike prices
R	Risk-free rate to expiration
$Q(k_i)$	Spread midpoint for option with strike k_i

[2]At the money are calls and puts whose strike price is at or very near to the current market price of the underlying security.

[3]Out of the money refers to options that do not have any intrinsic value; they only have extrinsic, or time value.

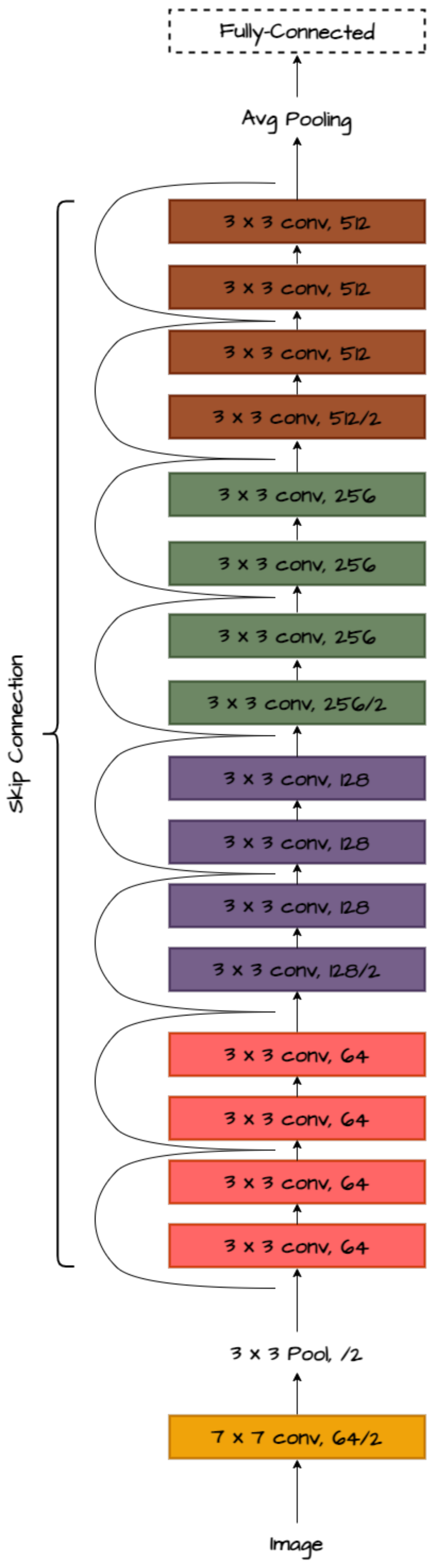


Figure 5.2: ResNet-18 Architecture.

ResNet-18

The ResNet-18 architecture comprises multiple stages and layers. Initially, the input of a fixed-sized image, typically 224 × 224 pixels undergoes a convolutional layer with 64 filters with a spatial size of 7 × 7, following a batch normalisation layer and Rectified Linear Unit (ReLU) activation resulting in dimension reduction of the input image. Thereafter, a 3 × 3 max-pooling layer is employed to reduce the spatial dimension. From here onward there exist four sets of layers each containing two residual blocks with convolutional layers with respective 64, 128, 256, and 512 filters. Following a global average pooling layer is applied to aggregate spatial information across the entire feature map leading to a fixed-size representation. Ultimately, the global average pooling layer output is passed to a fully connected layer. As for ResNet-18, the fully connected layer consists of 1000 units corresponding to the number of classes in the ImageNet dataset. The final classification probabilities are obtained through the application of a softmax activation function at the exit of this layer.

Dataset Description

The dataset used in this study comprises VVIX time series data spanning from 2006 to 2023, obtained from the Chicago Board Options Exchange website[4]. The Volatility of VIX (VVIX) measures the expected volatility for the S&P 500 options. It is a crucial indicator for market participants, including investors and traders, as it reflects the market's expectations of future volatility. High VVIX values signify increased market uncertainty and potential fluctuations in price, while low VVIX values suggest market stability and reduced volatility. Accurate VVIX predictions are of great importance for making informed decisions related to risk management, portfolio allocation, and trading strategies.

Proposed Hybrid Model

The proposed model in this experiment combines both an image pathway and a recurrent pathway to predict VVIX values.

Image Pathway

The image pathway leverages three representations: Gramian Angular Difference Field, Gramian Angular Summation Field, and Markov Transition Field. These representations convert the VVIX data into spatial images, which are then processed through ResNet blocks and convolutional layers. This enables the model to capture spatial patterns and features in the volatility data. To take advantage of pre-existing knowledge and feature representations, we employ a pre-trained ResNet-18

[4]https://www.cboe.com/us/indices/dashboard/vvix/

Recurrent Pathway	Input Dim	Hidden Dim
LSTM Layer 1	1	32
LSTM Layer 2	32	32

Table 5.1: Recurrent Pathway LSTM dimensions.

Image Pathway	Kernel
Convolutional Layer 1	128
Convolutional Layer 2	64
Convolutional Layer 3	32

*All convolutional layers have a kernel size of 3, a stride of 1, and a padding of 1, without the use of biases.

Table 5.2: Image Pathway kernel dimensions.

featuriser within the image pathway. This approach reduces data requirements, speeds up training, and improves performance. Three convolutional layers are then applied, with the final layer concatenating the outputs of the convolutional neural networks (CNNs). This process allows the model to capture spatial features and patterns present in the images.

Recurrent Pathway

In contrast to the image pathway, the recurrent pathway incorporates sequential information by passing the VVIX time series data through recurrent layers to capture temporal dependencies and trends. Two consecutive recurrent layers are used to process the recurrent sequences effectively.

Data Preparation

Before inputting the data into the model, several preprocessing steps were applied. For the recurrent pathway, sequences were constructed using three different sliding windows, corresponding to short-term, mid-term, and long-term time frames. Each sequence had a window length of 10, 21, and 41 working days as features, and the subsequent day was used as the label for making predictions. The window lengths are chosen arbitrarily to incorporate the impact of providing the model with more information. In addition to the recurrent sequences, three key representations were generated for the image pathway. These representations were used to create RGB[5] images depicting the volatility values for each time frame. Figure 5.4 displays a sample of the generated images, providing a visual representation.

[5]Red-Green-Blue

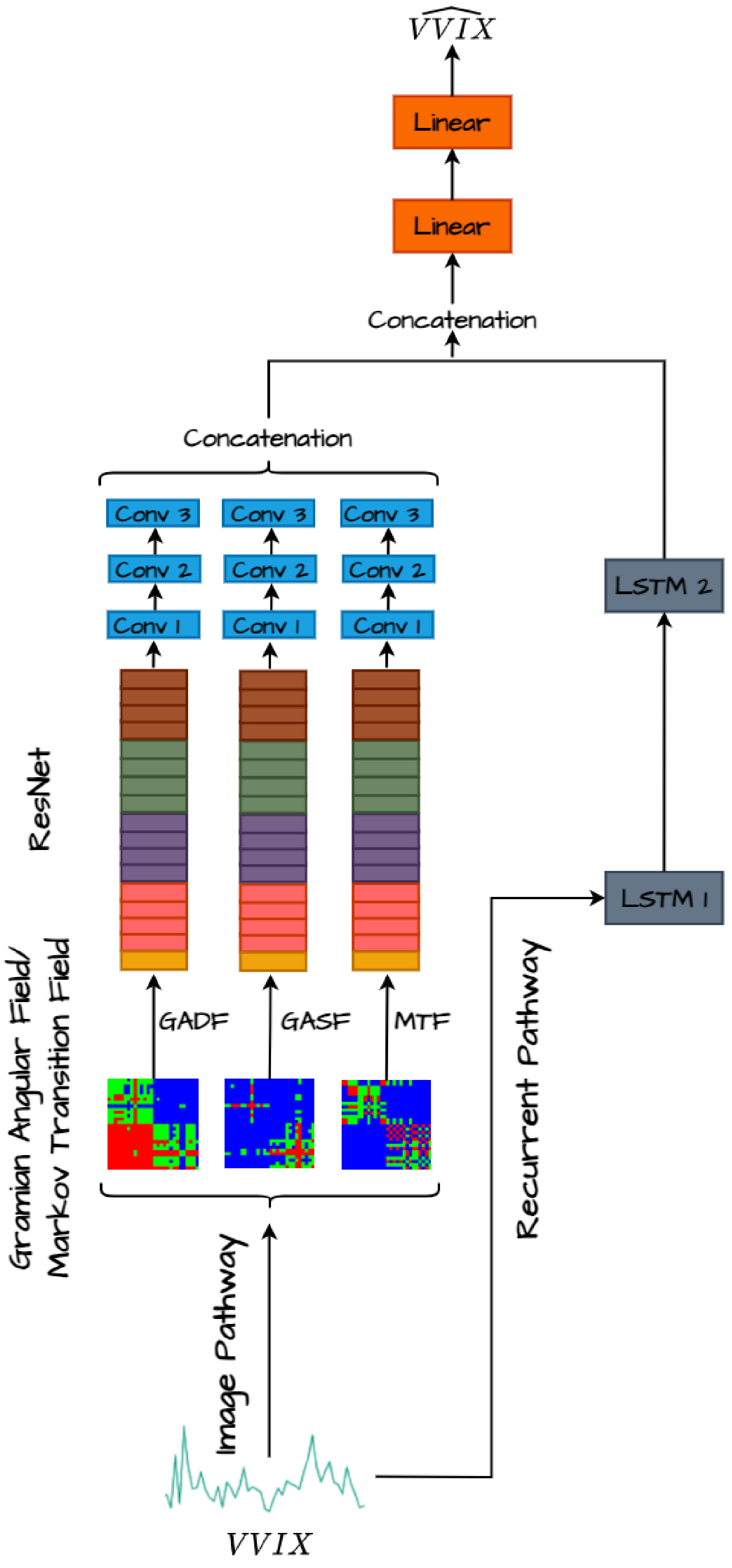


Figure 5.3: Proposed ResNet-LSTM Model

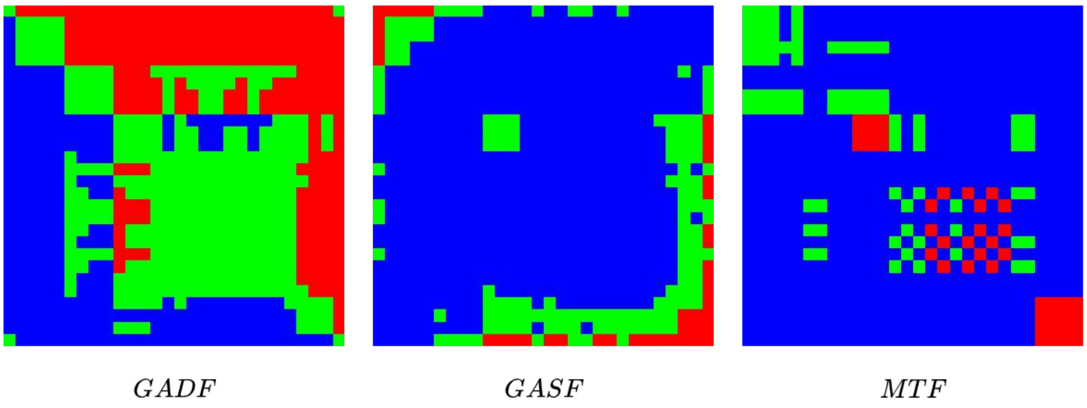


Figure 5.4: GAF, MTF Images of VVIX

Model Training

The model's weights and biases were optimised using the mean squared error (MSE) loss function and the Adam optimiser. To enhance the model's performance during training and avoid convergence to sub-optimal solutions, a decaying learning rate was implemented. If the loss plateaued after three consecutive epochs, the learning rate was reduced from the initial value of 0.01. After subjecting the proposed ResNet-LSTM model to three training runs using distinct time frames (short-term, mid-term, and long-term: 10x10, 21x21, and 41x41), it became evident that the mid-term time frame exhibited a more favourable balance in accuracy and demanded a reasonable duration for training. Combining the strengths from both the image and recurrent pathways within our model exhibits promising potential. It showcases the ability to predict VVIX values with enhanced precision, particularly highlighted by the mid-term time frame which demonstrated notably reduced errors on average when contrasted with the outcomes from short-term and long-term time frames.

	Image Size	LSTM Lag Length	Run 1 MSE	Run 2 MSE	Run 3 MSE
	10×10	10	0.008	0.051	0.034
ResNet-LSTM	21×21	21	0.014	0.087	0.007
	41×41	41	0.055	0.028	0.067

Table 5.3: ResNet-LSTM results for different image sizes and LSTM lag lengths.

5.3 Results

In this study, we proposed and examined a novel model for predicting VVIX values based on a combination of image and recurrent pathways. The VVIX dataset, spanning from 2006 to 2023, served as the foundation for our investigation into accurately forecasting market volatility. Our research is significant as it addresses the critical need for reliable VVIX predictions, enabling market participants, such as investors and traders, to make well-informed decisions regarding risk management, portfolio allocation, and trading strategies. The utilisation of both image and recurrent pathways in our proposed model allowed us to capture and incorporate both spatial patterns and temporal dependencies present in the VVIX time series data. The image pathway leveraged Gramian Angular Difference Field, Gramian Angular Summation Field, and Markov Transition Field to convert the raw VVIX data into spatial images.

By employing pre-trained ResNet-18 featuriser and convolutional layers within the image pathway, we effectively harnessed valuable feature representations while optimising computational efficiency. Concurrently, the recurrent pathway was adept at capturing sequential information, enabling the model to discern underlying trends and temporal dynamics in the VVIX time series. The combination of both pathways facilitated a comprehensive understanding of the volatility patterns, culminating in a reasonable predictive performance. The experimental results exhibited promising outcomes, with the mid-term time frame demonstrating a particularly balanced accuracy and training time. This finding emphasises the importance of choosing an appropriate time frame for VVIX prediction, which can significantly impact the practical applicability of the model.

Overall, our proposed ResNet-LSTM model showcased potential advantages for accurate VVIX value predictions, providing valuable insights to market participants seeking to navigate the complexities of financial markets. The successful amalgamation of spatial and temporal information in our model highlights the significance of incorporating multiple data modalities in predictive tasks, particularly in the domain of financial market forecasting. It is important to acknowledge the limitations of our study. While our proposed model exhibited promising results, further research is warranted to explore various architecture configurations, alternative data representations, and additional optimisation strategies. Moreover, the generalisability of the model to different market conditions and the influence of various external factors merit further investigation.

In conclusion, this research contributes to the growing body of knowledge in predictive analytics for financial markets, specifically in the context of VVIX value forecasting. Our proposed model serves as a stepping stone towards more accurate and robust predictions in financial domains, which are indispensable for making informed decisions and mitigating risk in today's dynamic and interconnected global markets. As the financial landscape continues to evolve, our model opens avenues for future studies and applications that can have a profound impact on the financial industry and its stakeholders.

Chapter 6

Conclusion and Future Work

This chapter will provide a conclusive overview of the investigation by succinctly summarising the key research findings in relation to the study's objectives and queries, as well as highlighting the worth and contribution thereof. Lastly, it will suggest prospects for further research.

6.1 Summary of Findings

In this comprehensive exploration of deep learning model applications in financial forecasting, our study delved into three distinct yet interconnected research questions, each contributing a unique perspective to our understanding of the field. The following research questions guided our study:

Project 1

- ***How does a deep learning CNN-LSTM model perform within the Directional Change Framework for forecasting events, considering the inherent volatility of high-frequency financial data?***

  To answer the first research question, in our first project, we summarised the price movement of the high-frequency tick bar data of four currency pairs i.e., GBPUSD, EURUSD, USDCHF, and USDCAD using the Directional Change framework. Within DC, an event is identified by a price change greater than a pre-defined threshold θ. We employed the Average True Range (ATR) indicator to determine θ based upon the oscillation of tick data. Subsequent to the generation of the DC events, a one-dimensional CNN-LSTM model was utilised to predict unseen events. Through this investigation, we observed that the CNN-LSTM model's performance transcends within the DC framework, demonstrating improvements in MAE, RMSE, and R^2 metrics, compared to its performance on raw tick bars. Comparative analysis with

Support Vector and Random Forest regression models also validated the out-performance of CNN-LSTM with respect to predictive accuracy.

Project 2

- ***Can an Autoencoder and a Variational Autoencoder be employed for financial time series forecasting? If so, how would they perform compared to recurrent and convolutional models?***

  Our exploration of the second research question led us to investigate the applications of a variational autoencoder to improve the accuracy of financial forecasting. In this study, we leveraged a Long Short-term Memory Variational Autoencoder (LSTM-VAE) in the denoising block to remove noise from the minute-by-minute price information of the five companies with ticker symbols AAPL, GOOG, AMZN, FB, and EBAY respectively. Following this step, the denoised data were used to extract a new set of features within the feature engineering and selection block. The outputs were used in the forecasting block which contains a stacked LSTM autoencoder, a stacked Bi-LSTM autoencoder, a CNN-LSTM and a CNN-BiLSTM. The outcomes exhibited by the Autoencoders could be utilised to forecast time-series data however the CNN-LSTM and CNN-BiLSTM performances exceeded in terms of MAE, RMSE, and R^2.

Project 3

- ***How can pre-trained computer vision models be harnessed for financial prediction?***

  The final research question shaped our third project. In this study, we explored the feasibility of employing pre-trained computer vision models in financial forecasting, for predicting VVIX values by merging images and recurrent sequences from the image and recurrent pathways. We utilised VVIX data spanning from 2006 to 2023 to generate RGB images with 10×10, 21×21, and 40×40 image sizes, representing the short-term, mid-term, and long-term periods. By incorporating both spatial patterns and temporal dependencies present in the VVIX time series, we utilised Gramian Angular Difference Field, Gramian Angular Summation Field, and Markov Transition Field to convert raw VVIX data into spatial images. The image pathway leverages a pre-trained ResNet-18 featuriser and convolutional layers to efficiently capture valuable feature representations. Meanwhile, the recurrent pathway captures sequential information, enabling the framework to comprehend underlying trends and temporal dynamics. The integration of pathways leads to improved predictive performance, particularly within the mid-term time frame. The ResNet-LSTM framework exhibits potential advantages for reasonable VVIX value projections, demonstrating the significance of incorporating multiple data modalities in financial market forecasting.

6.2 Synthesis of Findings

Collectively, the findings from these projects underscored the value of deep learning models in financial forecasting. Our research in the first project highlighted the application of the CNN-LSTM model in predicting Directional Change events and the contribution of the Directional Change framework to improve the accuracy of the mentioned model. In the second project, we emphasised the capability of autoencoders and variational autoencoders to remove noise in high-frequency minute-by-minute data as well as forecasting. Ultimately, we accentuated the potency of pre-trained computer vision models i.e., ResNet-18 within a multi-modal framework to forecast the volatility of the volatility index (VVIX).

6.3 Contribution and Broader Significance

The present thesis has made a contribution through its empirical exploration of various deep learning model applications in the realm of financial forecasting. Through addressing unique research questions and conducting meticulous comparative analyses, our work has enhanced the collection of tools accessible to both financial researchers and practitioners. Project 1's insights supported the beneficial application of the deep learning CNN-LSTM model in forecasting the Directional Change events as well as the potential of the Directional Change framework in improving the CNN-LSTM model's prediction accuracy. Project 2's emphasis on autoencoders and variational autoencoders highlighted their potential for noise reduction and forecasting, while Project 3's novelty in integrating pre-trained computer vision models and multi-modal models for volatility forecasting is noteworthy. These contributions have collectively contributed to the domain of financial forecasting with deep learning approaches.

6.4 Future Research

Although our investigation has shed light on several aspects of the utilisation of deep learning models for financial forecasting, there exist intriguing opportunities for further exploration.

1. Utilising denoising and forecasting capabilities of autoencoders and variational autoencoders in the Directional Change framework.
2. Investigating the utilisation of Gramian Angular Fields and Markov Transition Fields trans-

formations in Directional Change framework.

3. Determination and clustering of the patterns in high-frequency tick data using the Gramian Angular Fields and Markov Transition Fields transformations.

4. Applications of multi-modal approaches within Directional Change framework.

5. Improving the ResNet-LSTM model by employing different models. Replacing LSTM layers in the recurrent pathway with transformers models and extending the width by employing more inputs.

By venturing into these unexplored directions, future research has the potential to deepen our understanding of the intricate interplay between deep learning models and the dynamic landscape of financial forecasting.

Bibliography

[1] Ayman Abu Hammad and Ernest Hall. *Forecasting the Jordanian Stock Prices Using Artificial Neural Networks*. 01 2007.

[2] Md. Nasim Adnan. On reducing the bias of random forest. In Weitong Chen, Lina Yao, Taotao Cai, Shirui Pan, Tao Shen, and Xue Li, editors, *Advanced Data Mining and Applications*, pages 187–195. Springer Nature Switzerland, 2022. ISBN 978-3-031-22137-8.

[3] Irene E. Aldridge. *High-Frequency Trading: A Practical Guide to Algorithmic Strategies and Trading Systems*. Wiley Trading, 2009.

[4] Hamed Alqahtani, Manolya Kavakli-Thorne, and Gulshan Kumar. Applications of generative adversarial networks (gans): An updated review. *Archives of Computational Methods in Engineering*, 28:525 – 552, 2019.

[5] Daniel Bachman. Comparing forecasting methods: Why do traditional macroeconometric models remain popular? *Econometrics: Applied Econometrics & Modeling eJournal*, 2011.

[6] Amer Bakhach, Edward Tsang, and Hamid R. Jalalian. Forecasting directional changes in the FX markets. 12 2016.

[7] Amer Bakhach, Edward Tsang, Wing Lon Ng, and V L Raju Chinthalapati. Backlash agent: A trading strategy based on directional change. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–9, 2016.

[8] Pierre Baldi, Søren Brunak, Paolo Frasconi, Giovanni Soda, and Gianluca Pollastri. Exploiting the past and the future in protein secondary structure prediction . *Bioinformatics*, 15(11): 937–946, 11 1999.

[9] Arindam Banerjee. *An Analysis of Logistic Models: Exponential Family Connections and Online Performance*, pages 204–215.

[10] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS ONE*, 12, 2017.

[11] Irwan Bello, William Fedus, Xianzhi Du, Ekin D. Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies, 2021.

[12] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2:157–66, 1994.

[13] Akshay Bhalla. Enhancement in predictive model for insurance underwriting. 2012.

[14] Daniel Alexandre Bloch. Option pricing with machine learning. *EngRN: Electronic*, 2019.

[15] Imad Bou-Hamad and Ibrahim Jamali. Forecasting financial time-series using data mining models: A simulation study. *Research in International Business and Finance*, 51:101072, 2020.

[16] Joseph L. Breeden. Survey of machine learning in credit risk. *Electronic*, 2020.

[17] Bruce G. Buchanan. Can machine learning offer anything to expert systems? *Machine Learning*, 4:251–254, 2005.

[18] Nicholas Burgess. An introduction to algorithmic trading: Opportunities & challenges within the systematic trading industry. *Financial Crises eJournal*, 2019.

[19] Andriana S. L. O. Campanharo, M. Irmak Sirer, R. Dean Malmgren, Fernando Manuel Ramos, and Luis A. Nunes Amaral. Duality between time series and networks. *PLoS ONE*, 6, 2011.

[20] Mauro Castelli and Luca Manzoni. Special issue: Generative models in artificial intelligence and their applications. *Applied Sciences*, 2022.

[21] Yekun Chai, Qiyue Yin, and Junge Zhang. Improved training of mixture-of-experts language gans. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[22] Luyang Chen, Markus Pelger, and Jason Zhu. Deep learning in asset pricing. *Research Methods & Methodology in Accounting eJournal*, 2019.

[23] Shuangshuang Chen and Wei Guo. Auto-encoders in deep learningmdash;a review with new perspectives. *Mathematics*, 11(8), 2023. ISSN 2227-7390. URL `https://www.mdpi.com/2227-7390/11/8/1777`.

[24] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan M. Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *ArXiv*, abs/1410.0759, 2014.

[25] Subhabrata Choudhury, Subhajyoti Ghosh, Arnab Bhattacharya, Kiran Jude Fernandes, and

Manoj Kumar Tiwari. A real time clustering and svm based price-volatility prediction for optimal trading strategy. *Neurocomputing*, 131:419–426, 2014.

[26] Gil Cohen. Algorithmic trading and financial forecasting using advanced artificial intelligence methodologies. *Mathematics*, 2022.

[27] Sven F. Crone and Christian Koeppel. Predicting exchange rates with sentiment indicators: An empirical evaluation using text mining and multilayer perceptrons. *2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pages 114–121, 2014.

[28] Robert Culkin. Machine learning in finance: The case of deep learning for option pricing. 2017.

[29] Chester Curme, Harry Eugene Stanley, and Irena Vodenska. Coupled network approach to predictability of financial market returns and news sentiments. *International Journal of Theoretical and Applied Finance*, 18:1550043, 2015.

[30] Shom Prasad Das and Sudarsan Padhy. Support vector machines for prediction of futures prices in indian stock market. *International Journal of Computer Applications*, 41:22–26, 2012.

[31] Min-Yuh Day and Chia-Chou Lee. Deep learning for financial sentiment analysis on finance news providers. *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1127–1134, 2016.

[32] Min-Yuh Day and Chia-Chou Lee. Deep learning for financial sentiment analysis on finance news providers. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1127–1134, 2016.

[33] Oscar Day and Taghi M. Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4:1–42, 2017.

[34] Peter Dayan, Geoffrey E. Hinton, Radford M. Neal, and Richard S. Zemel. The helmholtz machine. *Neural Computation*, 7:889–904, 1995.

[35] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. John Wiley & Sons, 2018.

[36] Thomas G. Dietterich. What is machine learning? *Machine Learning and AI for Healthcare*, 2019.

[37] L. DiPersio and O. Honchar. Recurrent neural networks approach to the financial forecast of Google assets. 2017.

[38] Dongsong Zhang and Lina Zhou. Discovering golden nuggets: data mining in financial application. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 34(4):513–522, 2004.

[39] Ziv Epstein, Aaron Hertzmann, Laura Mariah Herman, Robert Mahari, Morgan R. Frank, Matthew Groh, Hope Schroeder, Amy Smith, Memo Akten, Jessica Fjeld, Hany Farid, Neil Leach, Alex Pentland, and Olga Russakovsky. Art and the science of generative ai. *Science*, 380:1110 – 1111, 2023.

[40] Thomas G. Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *Eur. J. Oper. Res.*, 270:654–669, 2018.

[41] Mark D. Flood, H. V. Jagadish, and Louiqa Raschid. Big data challenges and opportunities in financial stability monitoring. *Financial Stability Review*, pages 129–142, 2016.

[42] Ramazan Gençay, Michel M. Dacorogna, Ulrich A. Müller, Olivier V. Pictet, and Richard B. Olsen. *An Introduction to High-Frequency Finance.* 2001.

[43] J. Glattfelder, A. Dupuis, and R. Olsen. Patterns in high-frequency FX data: discovery of 12 empirical scaling laws. *Quantitative Finance*, 11:599 – 614, 2011.

[44] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[45] Ashok K. Goel and Kurt P. Eiselt. Mental models, text interpretation, and knowledge acquisition. *SIGART Bull.*, 2:75–78, 1991.

[46] A. Golub, G. Chliamovitch, A. Dupuis, and B. Chopard. Multi-scale representation of high frequency market liquidity, 2014.

[47] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016.

[48] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[49] Adam Grealish and Petter N. Kolm. Robo-advisory: From investing principles and algorithms to future developments. *Robotics eJournal*, 2021.

[50] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *ArXiv*, abs/1502.04623, 2015.

[51] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *AAAI*, 2018.

[52] Shihao Gu, Bryan T. Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *NBER Working Paper Series*, 2018.

[53] Bin Gui, Xianghe Wei, Qiong Shen, Jingshan Qi, and Liqiang Guo. Financial time series forecasting using support vector machine. *2014 Tenth International Conference on Computational Intelligence and Security*, pages 39–43, 2014.

[54] Dominique M. Guillaume, M. Dacorogna, R. R. Davé, Ulrich A. Müller, R. Olsen, and O. Pictet. From the bird's eye to the microscope: A survey of new stylized facts of the intra-daily foreign exchange markets. *Finance and Stochastics*, 1:95–129, 1997.

[55] Zhiqiang Guo, Huaiqing Wang, QUAN LIU, and Jie Yang. A feature fusion based forecasting model for financial time series. *PLoS ONE*, 9, 2014.

[56] Hamid Haddadian, Morteza Baky Haskuee, and Gholamreza Zomorodian. A hybrid artificial intelligence approach to portfolio management. *Iranian Journal of Finance*, 2022.

[57] Magnus Hansson. On stock return prediction with LSTM networks. 2017.

[58] Dinesh G. Harkut and K. N. Kasat. Introductory chapter: Artificial intelligence - challenges and applications. *Artificial Intelligence - Scope and Limitations*, 2019.

[59] GM Harshvardhan, Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray. A comprehensive survey and analysis of generative models in machine learning. *Comput. Sci. Rev.*, 38:100285, 2020.

[60] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[61] Pierre Henry-Labordère. Generative models for financial data. *Derivatives eJournal*, 2019.

[62] Geoffrey E. Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504 – 507, 2006.

[63] S. Hochreiter and J. Schmidhuber. LSTM can solve hard long time lag problems. In *NIPS*, 1996.

[64] Harrison G. Hong and Jiang Wang. Trading and returns under periodic market closures. *Journal of Finance*, 55:297–354, 2000.

[65] Y Hu. Stock market timing model based on convolutional neural network–a case study of Shanghai composite index. *Finance & Economy*, 4:71–74, 2018.

[66] Mei hua Zheng and Jia Miao. Comparing the forecastability of alternative quantitative models: a trading simulation approach in financial engineering. *Systems Engineering Procedia*, 4: 35–39, 2012.

[67] Jiayuan Huang, Alex Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. In *NIPS*, 2006.

[68] Codruț-Florin Ivașcu. Option pricing using machine learning. *Expert Systems with Applications*, 163:113799, 2021. ISSN 0957-4174.

[69] Charbel José Chiappetta Jabbour, Ana Beatriz Lopes de Sousa Jabbour, Joseph Sarkis, and Moacir Godinho Filho. Unlocking the circular economy through new business models based on large-scale data: An integrative framework and research agenda. *Technological Forecasting and Social Change*, 2017.

[70] Matthew J. Johnson, David Kristjanson Duvenaud, Alexander B. Wiltschko, Ryan P. Adams, and Sandeep Robert Datta. Composing graphical models with neural networks for structured representations and fast inference. In *NIPS*, 2016.

[71] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Annual Meeting of the Association for Computational Linguistics*, 2014.

[72] Michael Kampouridis and Fernando Otero. Evolving trading strategies using directional changes. *Expert Systems with Applications*, 73:145 – 160, 2017.

[73] Andreas S. Karathanasopoulos, Mitra Sovan, Chia Chun Lo, Adam Zaremba, and Mohammed Osman. Ensemble models in forecasting financial markets. *ERN: Forecasting Techniques (Topic)*, 2019.

[74] Sengim Karayalcin, Guilherme Perin, and Stjepan Picek. Resolving the doubts: On the construction and use of resnets for side-channel analysis. *Mathematics*, 11(15), 2023. doi: 10.3390/math11153265.

[75] Divit Karmiani, Ruman Kazi, Ameya Nambisan, Aastha Shah, and Vijaya Kamble. Comparison of predictive algorithms: Backpropagation, SVM, LSTM and Kalman filter for stock market. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 228–234, 2019.

[76] Byeong Soo Kim and T. Kim. Cooperation of simulation and data model for performance analysis of complex systems. *International Journal of Simulation Modelling*, 18:608–619, 12 2019.

[77] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[78] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[79] Diederik P. Kingma, Tim Salimans, and Max Welling. Improved variational inference with inverse autoregressive flow. *ArXiv*, abs/1606.04934, 2017.

[80] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2016.

[81] Anisha M. Lal, Bharath Reddy, and Aju D. Review on various machine learning and deep learning techniques for prediction and classification of quotidian datasets. 2020.

[82] Alex Lamb. A brief introduction to generative models. *ArXiv*, abs/2103.00265, 2021.

[83] Kruti Lavingia, Pimal Khanpara, Rachana Mehta, Karan Patel, and Niket Kothari. Predicting stock market trends using random forest: A comparative analysis. In *2022 7th International Conference on Communication and Electronics Systems (ICCES)*, pages 1544–1550, 2022. doi: 10.1109/ICCES54183.2022.9835876.

[84] Yann LeCun, Lawrence D. Jackel, Léon Bottou, A. Brunot, Corinna Cortes, John S. Denker, Harris Drucker, Isabelle M Guyon, Urs Muller, E. Sackinger, Patrice Y. Simard, and Vladimir Naumovich Vapnik. Comparison of learning algorithms for handwritten digit recognition. 1995.

[85] Jing Li, S. Pan, L. Huang, and Xi Zhu. A machine learning based method for customer behavior prediction. *Tehnicki Vjesnik-technical Gazette*, 26:1670–1676, 2019.

[86] Y. Li and W. Ma. Applications of Artificial Neural Networks in Financial Economics: A Survey. In *2010 International Symposium on Computational Intelligence and Design*, volume 1, pages 211–214, 2010.

[87] Fenglin Liu, Xuancheng Ren, Zhiyuan Zhang, Xu Sun, and Yuexian Zou. Rethinking skip connection with layer normalization in transformers and resnets, 2021.

[88] Marc E. Maier, Hayley Carlotto, Freddie Sanchez, Sherriff Balogun, and Sears A. Merritt. Transforming underwriting in the life insurance industry. In *AAAI Conference on Artificial Intelligence*, 2019.

[89] Benoît Mandelbrot and Howard M. Taylor. On the Distribution of Stock Price Differences. *Operations Research*, 15(6):1057–1062, 1967.

[90] Vivien Marx. Biology: The big challenges of big data. *Nature*, 498:255–260, 2013.

[91] Ricardo P. Masini, M. C. Medeiros, and Eduardo F. Mendes. Machine learning advances for time series forecasting. *ArXiv*, abs/2012.12802, 2020.

[92] Sidra Mehtab and Jaydip Sen. Stock Price Prediction Using Convolutional Neural Networks on a Multivariate Timeseries, 2020.

[93] Kostadin Mishev, Ana Gjorgjevikj, Irena Vodenska, Ljubomir T. Chitkushev, Wataru Souma, and Dimitar Trajanov. Forecasting corporate revenue by using deep-learning methodologies. *2019 International Conference on Control, Artificial Intelligence, Robotics & Optimization (IC-CAIRO)*, pages 115–120, 2019.

[94] Hanae Moussaoui, Nabil El Akkad, and Mohamed Benslimane. Reinforcement learning: A review. *International Journal of Computing and Digital Systems*, 2023.

[95] David Nelson, Adriano Pereira, and Renato de Oliveira. Stock market's price movement prediction with LSTM neural networks. pages 1419–1426, 05 2017.

[96] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner, 2019.

[97] Aakash Parmar, Rakesh Katariya, and Vatsal Patel. A review on random forest: An ensemble classifier. In Jude Hemanth, Xavier Fernando, Pavel Lafata, and Zubair Baig, editors, *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018*, pages 758–763. Springer International Publishing, 2019.

[98] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[99] G. Jaculine Priya and S. Saradha. Fraud detection and prevention using machine learning algorithms: A review. *2021 7th International Conference on Electrical Energy Systems (ICEES)*, pages 564–568, 2021.

[100] Lele Qin, Naiwen Yu, and Donghui Zhao. Applying the convolutional neural network deep learning technology to behavioural recognition in intelligent video. *Tehnicki Vjesnik*, 25:528–535, 04 2018.

[101] Yongyi Ran, Han Hu, Xin Zhou, and Yonggang Wen. Deepee: Joint optimization of job scheduling and cooling control for data center energy efficiency using deep reinforce-

ment learning. *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 645–655, 2019.

[102] Ibai Roman, Roberto Santana, Alexander Mendiburu, and José Antonio Lozano. In-depth analysis of svm kernel learning and its components. *Neural Computing and Applications*, 33: 6575 – 6594, 2020.

[103] Murtaza Roondiwala, Harshal Patel, and Shraddha Varma. Predicting stock prices using LSTM. 2017.

[104] Ahoora Rostamian and John G. O'Hara. Event prediction within directional change framework using a cnn-lstm model. *Neural Computing and Applications*, 34:17193 – 17205, 2022.

[105] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2020.

[106] E.W. Saad, D.V. Prokhorov, and D.C. Wunsch. Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks*, 9(6):1456–1470, 1998.

[107] Imane Sadgali, Nawal Sael, and Faouzia Benabbou. Performance of machine learning techniques in the detection of financial frauds. *Procedia Computer Science*, 2019.

[108] V. E. Salis, Akanksha Kumari, and Animesh Singh. Prediction of gold stock market using hybrid approach. *International journal of engineering research and technology*, 8:803–812, 2019.

[109] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[110] Jaydip Sen. Stock Price Prediction Using Machine Learning and Deep Learning Frameworks. 12 2018.

[111] Joel Serey, Luis E. Quezada, Miguel D. Alfaro, Guillermo Fuertes, Manuel Vargas, Rodrigo Ternero, Jorge Sabattin, Claudia Duran, and Sebastián Gutiérrez. Artificial intelligence methodologies for data management. *Symmetry*, 13:2040, 2021.

[112] Omer Berat Sezer and Ahmet Murat Ozbayoglu. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing*, 70:525–538, 2018.

[113] Shunrong Shen, Haomiao Jiang, and Tongda Zhang. Stock market forecasting using machine learning algorithms. 2012.

[114] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of LSTM

and BILSTM in forecasting time series. *2019 IEEE International Conference on Big Data (Big Data)*, pages 3285–3292, 2019.

[115] Diego Furtado Silva, Vinicius M. A. Souza, and Gustavo E. A. P. A. Batista. Time series classification using compression distance of recurrence plots. *2013 IEEE 13th International Conference on Data Mining*, pages 687–696, 2013.

[116] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1310–1315, 2016.

[117] Erik Solís, Sherald Noboa, and Erick Cuenca. Financial time series forecasting applying deep learning algorithms. *Information and Communication Technologies*, 2021.

[118] Jan De Spiegeleer, Dilip B. Madan, Sofie Reyners, and Wim Schoutens. Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. *Quantitative Finance*, 18: 1635 – 1643, 2018.

[119] Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60:699–746, 2008.

[120] George Daniel Brown Swankie and Daniel Broby. Examining the impact of artificial intelligence on the evaluation of banking risk. 2019.

[121] Sean J. Taylor and Benjamin Letham. Forecasting at scale. *PeerJ Prepr.*, 5:e3190, 2017.

[122] Aditya Thakur, Harish Chauhan, and Nikunj Gupta. Efficient resnets: Residual network design, 2023.

[123] Michal Tkáč and Robert Verner. Artificial neural networks in business: Two decades of research. *Applied Soft Computing*, 38:788 – 804, 2016.

[124] E. Tsang, Ran Tao, A. Serguieva, and S. Ma. Profiling high-frequency equity price movements in directional changes. *Quantitative Finance*, 17:217 – 225, 2017.

[125] Balakrishnan Umadevi, D. Sundar, and P. Alli. An effective time series analysis for stock trend prediction using arima model for nifty midcap-50. *International Journal of Data Mining & Knowledge Management Process*, 3:65–78, 2013.

[126] Vladimir Naumovich Vapnik. The nature of statistical learning theory. In *Statistics for Engineering and Information Science*, 2000.

[127] Andrés Vidal and W. Kristjanpoller. Gold volatility prediction using a cnn-lstm approach. *Expert Syst. Appl.*, 157:113481, 2020.

[128] Jindong Wang and Yiqiang Chen. *Introduction to Transfer Learning: Algorithms and Practice*. Machine Learning: Foundations, Methodologies, and Applications. Springer Singapore, 2023. doi: 10.1007/978-981-19-7584-4.

[129] Ju-Jie Wang, Jian-Zhou Wang, Zhe-George Zhang, and Shu-Po Guo. Stock index forecasting based on a hybrid model. *Omega*, 40(6):758 – 766, 2012.

[130] Pingan Wang, Yuanwei Lou, and Lei Lei. Research on stock price prediction based on BP wavelet neural network with mexico Hat wavelet basis. pages 99–102. Atlantis Press, 2017.

[131] Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation. *ArXiv*, abs/1506.00327, 2015.

[132] Karl R. Weiss, Taghi M. Khoshgoftaar, and Dingding Wang. A survey of transfer learning. *Journal of Big Data*, 3:1–40, 2016.

[133] White. Economic prediction using neural networks: the case of IBM daily stock returns. In *IEEE 1988 International Conference on Neural Networks*, pages 451–458 vol.2, 1988.

[134] Shiyang Xuan, Guanjun Liu, Zhenchuan Li, Lutao Zheng, Shuo Wang, and Changjun Jiang. Random forest for credit card fraud detection. In *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, pages 1–6, 2018. doi: 10.1109/ICNSC.2018.8361343.

[135] Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. *Transfer Learning*. Cambridge University Press, 2020.

[136] Tingting Ye and Liangliang Zhang. Derivatives pricing via machine learning. *Boston: Information Technology (Topic)*, 2019.

[137] Lining Yu, Wolfgang Karl Härdle, Lukas Borke, and Thijs Benschop. An ai approach to measuring financial risk. *Risk Management eJournal*, 2017.

[138] Li Yunsheng, Cao Jie, Chen Xuewen, Zhao Feng, and Li Jingling. Auto-recognition pedestrians research based on hog feature and svm classifier for vehicle images. *2020 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 304–309, 2020.

[139] Kamil Zbikowski. Using volume weighted support vector machines with walk forward testing and feature selection for the purpose of creating stock trading strategy. *Expert Syst. Appl.*, 42: 1797–1805, 2015.

[140] G.Peter Zhang. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, 50:159–175, 2003.

[141] Li Zhang, Fulin Wang, Bing Xu, Wenyu Chi, Qiongya Wang, and Ting Sun. Prediction of stock prices based on LM-BP neural network and the estimation of overfitting point by RDCI. *Neural Computing and Applications*, 30:1425–1444, 09 2018.

[142] Q. Zhuge, L. Xu, and G. Zhang. LSTM neural network with emotional analysis for prediction of stock price. *Engineering Letters*, 25:167–175, 01 2017.