FINAL YEAR PROJECT

Comparative Analysis of LSTM and GRU Models for Stock Index Prediction in Developed and Emerging Markets

TAN TIAN QING

TP068266

APU3F2408BAF(FT)

Bachelor in Banking and Finance (Hons) with specialism in Financial Technology

School of Accounting and Finance,

Asia Pacific University of Technology and Innovation

Supervised by

DR. AHMAD DANIAL BIN ZAINUDIN

30th April 2025

Table of Contents

List of Tables

List of Figures

Acknowledgements

This graduation project has been an undoubtedly challenging yet a rewarding journey. Completing this report came with its fair share of stressful days, late nights, and moments of self-doubt. But looking back, it's all been worth it as the sense of accomplishment pays off.

Sincerely, I would like to express my uttermost appreciation to my Final Year Project supervisor, Dr. Ahmad Danial Bin Zainudin. Throughout this journey, he has been nothing short of amazing: always supportive, approachable (even in weekends!), and quick to respond to my queries as well as review my progress. His expertise in the field of Robo-advisor, which beautifully blends finance knowledge with Python-based analytical toolkits, provided me with countless insightful advice that shaped the direction and depth of my project. Not only that, he always encouraged us to challenge ourselves and think critically, because he said it was his role to make us show proficiency both academically and professionally. It has truly been an honour to work under his guidance, and I am genuinely grateful for all the encouragement and inspiration he has given me. Without his dedicated support, this project wouldn't have been completed effectively. I would say, it's a pleasure having him as my FYP supervisor!

I also want to give a huge shoutout to the emergence of Generative AIs, especially ChatGPT and Claude. Their arrival has disruptively lowered the barriers to interdisciplinary learning, making it easier than ever to explore topics that beyond our own specialism. They are really powerful serving as personalized tutors whenever I hit a wall, fragmenting intricate concepts into simple and understandable pieces in a patience manner. ChatGPT has been amazing at illustrating new ideas in a way that even a layman could digest, while Claude is extraordinarily good at coding task. Their occurrence really delivers borderless and inclusive learning, as these powerful models are even accessible without subscription! Candidly speaking, without their assistance, it would have been much tougher to push through the technical challenges of learning, building and training deep learning algorithms.

A special thanks should be credited to Google Colab as well, because it equips me with its integrated GPUs, so that I was able to train the models far more efficiently than I ever could on my personal device, without the need to acquire any expensive hardware on my workstation.

Wrapping up, while it was not an easy path, but every step has been worth it. I am grateful for all the beauty along the way that I had yet to discover!

ABSTRACT

Purpose: This study investigates the predictive accuracy of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural networks on stock market index prediction, comparing their performance across emerging and developed markets. It also examines critiques and shortcomings on the Efficient Market Hypothesis (EMH) and Randon Walk Theory (RWT).

Design: Historical daily closing prices from 10 stock indexes over a 15-year period were collected and used to train LSTM and GRU models with fixed hyperparameters to maintain comparability. The 10 indexes were geographically diverse countries, equally divided between developed and emerging markets. Model performance was evaluated using RMSE, MAE, R^2, and Coefficient of Variation (CV). Each model is reiterating 10 times to obtain the averaged result due to the non-deterministic nature in deep learning algorithms.

Findings: Both LSTM and GRU achieved high predictive performance, with R^2 consistently above 0.95 and normalized errors below 2%. GRU outperformed LSTM marginally in both accuracy and consistency, indicated by slightly lower error metrics and more stable results across multiple runs. Predictive accuracy between market types differed insignificantly, though emerging markets showed greater stability. These results challenge the Weak Form EMH and RWT by demonstrating that historical prices alone can explain a large portion of future price movements. No substantial difference in predictability between developed and emerging markets was observed, suggesting comparable levels of weak-form efficiency.

Key words: Stock Market Index Prediction, Financial Time Series Forecasting, Deep Learning, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Developed and Emerging Market, Efficient Market Hypothesis, Behavioural Finance.

CHAPTER ONE: INTRODUCTION

1.1 Background of Study

Stock market is a foundational pillar of the economy, serving as a global hub where capital flows between entities domestically and internationally. Over decades, the total market capitalization of stock market worldwide has grown tremendously, rising from $2.5 trillion in 1980 to $115 trillion by the end of 2023, with a compound annual growth rate (CAGR) of approximately 9.3% (Sunny et al., 2020; Sifma Research, 2024). Even during challenging periods like financial crises or the COVID-19 pandemic, stock markets have always remained functional, playing indispensable roles in stabilizing as well as supporting economic activities. Their impact can be seen in both short and long term, as they drastically influence and transform a country's economic growth and investment dynamics (Azam et al., 2016; Masoud, 2013). A burgeoning stock market will appeal investors to engage in the market participation, which aid in increasing trading volume, accumulating capital, and creating opportunities for companies to expand for their growth. This, in turn, strengthens the country's economic development and highlights the importance of a stable stock market for sustained growth (Pradhan, 2018).

Stock market index, on the other hand, is a tool used to measure the value of a specific section of the stock market, often reflecting the overall market trends (Bruni, 2016; Helmenstein & Haefke, 2015). These indices have historically been computed by taking a weighted average of companies' market values, however, newer methodologies have been developed that accommodate the dynamic relationships between constituent stocks, offering better insights into market stability and spotting potential bubbles (Zatlavi et al., 2014). Also, stock market indexes serve as benchmarks for evaluating the performance of individual stocks as well as investment portfolios and provide a representation of the gross market sentiment (Kapoor, 2018). These indexes are closely tied to economic health, with factors like economic growth, interest rates, and monetary policy are important determinants in shaping long-term stock index movements, whereas market volatility, which is frequently triggered by global events like the Russia-Ukraine conflict, can lead to sharp fluctuations and disrupt stability in short term (Bhowmik & Wang, 2020; Sanoyo et al., 2024).

Stock market indexes are widely used in technical analysis to predict price directions and assist traders in identifying favourable trading opportunities (Bruni, 2016). To support these analyses, a branch of time series analysis, which is the financial time series analysis, emerges and is used extensively in predicting the movement of future price levels. Over time, various approaches have been developed to predict stock price trends.

In the past few decades, traditional forecasting models like ARIMA and GARCH have been commonly applied to predict returns and volatility. While these models are useful, they often struggle to capture the non-linear and complex patterns present in financial data, especially in the contemporary era of data and information explosion further increasing the complexity of financial markets (Bhowmik & Wang, 2020; Hassan, 2025). In order to overcome these limitations, researchers have explored advanced methods such as neural networks, human brain-inspired deep learning algorithms, which have emerged as highly effective methods due to their ability to learn from and adapt to the non-linear and complex behaviour of financial data, make them particularly useful in stock market prediction.

With these capabilities, it can help both individual and institutional market participants to formulate more enlightened actions, such as (1) deciding whether to buy, hold, or sell stocks, enabling them to manage risks more effectively, (2) allocating portfolio, (3) mitigating risks, as well as (4) maximizing profits (Althelaya et al., 2021; Touzani & Douzi, 2021; Tashakkori et al., 2024). Research by Paul (2024), Ta et al. (2020), and Lundqvist (2019) highlighted the combination of deep learning approaches with dynamic asset allocation models is a breakthrough to model portfolio management. The attractiveness of the potential benefits that predicting market movements could bring has driven researchers and practitioners to develop and explore myriad methods for stock price prediction (Singh & Srivastava, 2016). Notwithstanding, how can the stock market be accurately predicted under different contexts still remain a problem worth exploring (Gao et al., 2020).

1.2 Problem Statement

The question of whether the stock market has room for prediction has long been a topic of debate among scholars and practitioners alike (Aminimehr et al., 2022). Stock market indexes are usually characterized by volatile and changing persistently and irregularly, which creates challenges for researchers trying to capture their regularity (Jain, 2019). Over years, researchers have conducted extensive research in this area, however, many still argue that predicting non-linear and non-stationary financial time series data, remains an unfeasible mission. Despite numerous mathematical and statistical models have been proposed, yet the results either fall short of expectations or possess noticeable limitations (Patel et al., 2015).

In many literatures, Artificial Neural Network (ANN) models have been broadly employed against traditional linear models for stock prediction. They have also been compared with different data mining classification algorithms, but the comparison suggests that ANN models offer better results (Huang et al., 2008; Teixeira & De Oliveira, 2010; Guresen et al., 2011). However, some of the studies highlight considerable drawbacks in using ANNs for stock prediction, pointing out their unsuitability as the stock market data has enormous noise and complex dimensionality, which makes it difficult to deliver consistent and reliable outcomes (Shen et al., 2011). Not only that, many ANN models rely on shallow architectures with only one hidden layer, which can lead to issues like suboptimal training strategies and overfitting when handling multidimensional and noisy data (Larochelle et al., 2009).

Yet, advanced architectures deliver promising alternatives in overcoming these issues (Larochelle et al., 2009). Unlike traditional neural networks, they are capable of approximate complicated non-linear functions, and therefore, achieve better performance (Le Roux & Bengio, 2010; Sutskever & Hinton, 2008). Researches show that deep learning architectures, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have yielded promising performance in various fields including language, speech and image, but it is of interest whether they can achieve similar performance in the financial market (Hinton & Salakhutdinov, 2006; Yu et al., 2009; Mohamed et al., 2011; Krizhevsky & Hinton, 2012; Zuo & Wang, 2014).

In light of this, more recent studies have increasingly explored the use of them for financial prediction, such as the use of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), advanced variants of RNNs. However, despite the growing body of research, significant gaps persist. Much of the existing works concentrated on either developed markets or single region, leaving cross-market comparisons largely unexplored. Moreover, while LSTM and GRU are frequently used for stock prediction, direct comparisons of their effectiveness across different market contexts, especially between developed and emerging economies, are still scarce, leaving a gap in the literature.

Hence, the aim of this study is to fill this gap by examining the performance of these two models across ten countries, including both developed and emerging markets, distributed equally, which will contribute to a better understanding of their forecasting potential in varying market environments, and ultimately ensure that it can aid in industry players in making optimal decision.

1.3 Research Objectives

RO1: To compare the forecasting accuracy and consistency of LSTM and GRU models.

RO2: To compare the performance of LSTM and GRU models across emerging and developed markets.

RQ3: To examine the predictability of stock market index.

1.4 Research Questions

RQ1: Which model can predict the stock index movements more accurately and consistently?

RQ2: To what extent does the predictive performance differ across developed and emerging markets?

RQ3: Whether the market is always efficient and leaving no room for prediction?

1.5 Significance of Study

1.5.1 Academic Contribution

From an academic perspective, this study contributes to the growing body of knowledge on stock market prediction by focusing on a comparative analysis of LSTM and GRU models in both emerging and developed markets. While prior research has explored the comparisons among these models, comprehensive cross-market evaluations remain rare. By addressing the gap in cross-market studies, this research provides new insights into the predictability of stock index across diverse economic contexts. Moreover, this study offers a replicable research framework that open for future scholars to build upon. The findings of the research will also enrich the academic discourse on financial time series prediction by assessing how these models behave under different degrees of market maturity within the predetermined timeframe (which include both bull and bear markets), thus, contributing to a body of literature on future stock market prediction research in both developed and developing markets.

1.5.2 Theoretical Contribution

From the theoretical aspect, this study advances understanding of deep learning models' applicability to non-linear and non-stationary time series data in financial prediction. It evaluates the efficacy of LSTM and GRU under a consistent architecture, extending to the existing literature on deep learning investigation. Furthermore, this research challenges traditional finance theories such as the Efficient Market Hypothesis and Random Walk Theory, which suggest that stock prices follow an unpredictable path and are not forecastable using historical data as they are already priced-in for all available information. By demonstrating the capacity of advanced neural networks to capture hidden patterns and temporal dependencies in stock index movements, the study provides a counter perspective to these classical theories and supports the growing body of evidence that market inefficiencies are exploitable to certain extent through these non-linear models. The results also guide that models may yield varying results depending on market characteristics even under a consistent architecture.

1.5.3 Practical Contribution

From a practical standpoint, this research has meaningful implications for market participants including investors and traders, practitioners including portfolio managers and hedgers, and other stakeholders including policymakers. By offering a comparative evaluation of LSTM and GRU models in predicting stock index trends across markets of varying maturity, it equips stakeholders with data-driven tools to assist decision-making processes. This could be either a trading or investment decision.

The findings can help optimize portfolio management strategies, implement risk mitigation techniques, optimize portfolio allocation, improve timing of market entry or exit, and most importantly, potentially enhance returns and reduce losses. For instance, if a high-performing model predicted a potential sharp downturn in a specific index based on its previously learned patterns, traders or investors can treat it as an indicator supporting them to implement corresponding strategies such as stop-loss orders, hedging, or early profit-taking. Additionally, a portfolio manager may utilize the prediction as a reference to proactively adjust their portfolios by reducing the weightage of components that are expected to have higher possibility of falling and instead, switching to the promising ones.

Ultimately, the study bridges the gap between academic research and real-world application, promoting the integration of advanced deep learning models into the financial industry’s analytical toolkit.

1.6 Scope of Study

This study is concerned with the comparative analysis of deep learning models of LSTM and GRU for stock index prediction. Specifically, this investigation has targeted ten selected stock indexes, five from developed markets and five from emerging markets, reflecting the balance in geographical and economic distribution. The chosen stock indexes include the S&P 500 Index from the United States, FTSE 100 Index from the United Kingdom, DAX from Germany, Hang Seng Index from Hong Kong, and ASX 100 Index from Australia from developed markets, and Bovespa Index from Brazil, Mexican IPC Index from Mexico, Tadawul All Share Index from Saudi Arabia, Kuala Lumpur Composite Index from Malaysia, and Shanghai Stock Exchange Composite Index from China.

The period chosen for the study spans an all-inclusive 15-year period using daily data, extending from the beginning of 2010 to the end of 2024. This extensive time frame strategically captures multiple significant economic cycles and global events, including the aftermath of the 2008 global financial crisis, the unprecedented economic disruption caused by the COVID-19 pandemic, and the subsequent market recoveries after the pandemic. It also encompasses periods of geopolitical tensions, monetary policy shifts, and technological advancements that have influenced global financial markets. By including such a diverse economic landscape, the research aims to test the strength and adaptability of the designated deep learning models across varying market conditions and economic environments.

Methodologically, the study employs a quantitative approach, where data collection is through the Yahoo Finance API, and data processing as well as model construction is done through Python programming language, which is particularly suited for data analysis mission.

The scope is intentionally designed to maintain an equal representation between developed and emerging markets to provide a cross-market comparative analysis that addresses the lacuna in existing literature. By examining the predictive performance of LSTM and GRU across developed and emerging markets, the study seeks to uncover hidden insights for to the field of financial time series forecasting. Lastly, in essence, this study focuses on the finance perspective rather than the territory of data science, computer science, or applied mathematics. Meaning that, the findings of this project will not play around with the parameters and hyperparameters of models. It pays majority attention to market predictability.

1.7 Limitations

Despite its contributions, this study has several limitations that must be acknowledged. Firstly, even though the research focuses on ten countries' stock index, split between developed and emerging markets, representing a symmetrical geographical and economic distribution, and while this allows for cross-market comparison, the findings may not be generalizable to all other markets. Future research could expand the scope to include more diverse markets. Also, the 15-year period, though covering multiple economic cycles, may not fully capture all the market transformations or emerging economic patterns.

Methodological limitations are inherent in the deep learning approaches employed. Stock indexes are influenced by numerous external factors including geopolitical events, economic policies, market sentiments, and global economic trends, which cannot be fully captured by mathematical models. It is because, unexpected things happen all the time, one of the significant examples is that the newly imposed tariffs by the United States in April 2025. Also, the non-deterministic and probabilistic nature of stock market prediction means that while the models provide valuable insights, they always cannot guarantee absolute predictive accuracy. Thus, while LSTM and GRU models are powerful, they also subject to potential overfitting and may struggle with extreme market conditions or unprecedented economic events. The models' predictive capabilities are fundamentally retrospective, meaning their performance is based on historical patterns and may not perfectly anticipate future market disruptions or black swan events.

Lastly, only two deep learning models are employed in the study. While these models are among the most prominent for financial time series prediction, employing other advanced architectures for comparisons may offer additional insights with noticeable discrepancies. By acknowledging these limitations, this study provides a foundation for future research to address these gaps and build upon its findings.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

In this chapter, we present a thorough review of the existing literature relevant to stock index predictions, adopting a structured approach that progressively narrows from broader contextual understanding to specialized research techniques. The chapter begins by discussing underpinning theories for stock market, following by the exploration of the broader context – stock market index and market heterogeneity, offering reader a general picture of the financial framework. Subsequently, the chapter introduces neural networks and deep neural networks, which serve as the foundation of this study. The description continues with an organized review of financial time-series forecasting applying machine learning and deep learning approaches while exploring their suitability and limitations. Finally, the focus converges on the most specialized area of focus, which is the empirical studies on Long Short-Term Memory and Gated Recurrent Unit models for stock index prediction across various markets. This structured progression, moves from general concepts to specialized methodological approaches, provides a cohesive narrative that illuminates the research context, ensuring a clear understanding of the research landscape.

2.2 Theories and Theoretical Grounding

In the early research relating to stock market prediction, Fama (1970) proposed the Efficient Market Hypothesis (EMH) and Horne and Parker (1967) introduced the Random Walk Theory (RWT). These theories argued that stock prices are influenced by factors beyond historical data, making accurate prediction seemingly unattainable.

2.2.1 Efficient Market Hypothesis

The EMH has been under academic and professional consideration for many years, yet it is still an important part of modern finance theory (Degutis & Novickytė, 2014). This hypothesis suggests that the price of a stock is fully determined by available market information and thus, any new information leads to an immediate adjustment in stock prices as a reaction of the newly released information (Fama, 1970). This theory suggests that stocks are always traded at their fair value, leaving no opportunity for trader to neither buy nor sell at a discounted price to make profit from undervalued or overvalued prices. Consequently, traders can only increase returns by taking on excessive risk. Furthermore, in an efficient market, asset prices are unpredictable, and any excess return is often attributed to a success rather than a precise forecasting. Allen, et al. (2011) defined efficiency in a market as the inability to earn returns above the market average. Similarly, Mishkin and Eakins (2012) stated that efficient markets fully reflect all available information in asset prices.

Generally, the concept of market efficiency rests on two main pillars: (1) All available information is already incorporated into stock prices; (2) Investors cannot achieve risk-adjusted excess returns (Degutis & Novickytė, 2014). According to Fama (1970), EMH can differentiate three different forms of market efficiency:

(1) Weak form, considers only historical data, asserting past prices were already accounted in current prices and negating the utility of technical analysis;

(2) Semi-strong form, includes both historical and publicly available current information, asserting neither fundamental nor technical analysis works; and

(3) Strong form, incorporates private and confidential information (known as insider news), leaving no room for informational advantage.

Empirical research has widely supported the weak form of market efficiency across various markets (Palan, 2004). In turn, studies on the semi-strong form have produced mixed results, while strong form efficiency remains less explored, but often shows inefficiencies in markets, according to Mishkin & Eakins, (2012).

2.2.2 Random Walk Theory

When it comes to EMH, the RWT by Horne and Parker (1967), stating that the movement of stock prices are in random behaviour and do not follow any patterns and thus past price movements have no bearing on future changes, always come along discussion. This theory also claims that it is meaningless attempting to predict future stock values using historical movements. Based on the combination of EMH and RWT, the price of a stock in a market not only reflected all previous prices and information but also movements are entirely random without following a pattern, therefore, the likelihood of predicting the price fluctuations accurately must not surpass a half.

2.2.3 Critique to Efficient Market Hypothesis and Random Walk Theory

Nevertheless, contrary to these earlier theories, many contemporary studies have shown that stock price movements can be predicted to certain extent. Traditionally, two primary approaches have emerged in the field of finance for predicting future stock prices:

(1) Fundamental analysis, evaluates a company's financial health and economic conditions, including factors such as interest rates, return on assets, revenue, costs, and price-to-earnings ratios. The objective is to assess long-term sustainability and quantify future prospect for investment purposes; and

(2) Technical analysis, focuses on historical price trends and patterns using time-series data. Technical analysis assumes that even though stock prices appear randomly at times, they exhibit historical trends that may repeat over time. Therefore, the key elements for technical analysis including moving averages, price movements, historical patterns, and stock related information. This method is better suit for short-term predictions (Moukalled et al., 2019; Selvin et al., 2017).

More recently, studies have seen the application of time series forecasting especially using machine learning and deep learning techniques in stock price prediction, and has been proven effective to certain extent and able to draw generalized pattern, contradicting to these theories. There is a growing body of research presenting strong evidence of abnormal market behaviours that appear inconsistent with both the EMH and RWT, supported by the rise of Behavioural Finance, which challenges its core assumptions. The EMH is grounded in the idea that investors behave rationally at all times (Wong et al., 2022). However, many studies have shown that in most of the time, the investors being the social beings, with a brain and a heart full of emotions, behave in an irrational way, in spite of having accessible to relevant and accurate information. Instead of maintaining a rational mindset, they tend to display biases in various situations and overlook rationality attitude, causing market inefficiency (Sharma, 2014). One such behaviour is known for herd behaviour, where individuals follow the actions of others rather than making decisions based on their own analysis. This sometimes leads to suboptimal decisions. Another common phenomenon is the formation of speculative bubbles, where asset prices rise sharply and reach unrealistic levels, driven more by collective enthusiasm than by fundamental values, mainly due to psychological factors (investor frenzy). These bubbles eventually burst, causing sharp price declines. Such anomalies indicate that price movements are influenced by more than just new information, suggesting a degree of predictability in stock prices that contradicts EMH (Wong et al., 2022). As a result, behavioural finance models have gained traction for their ability to explain and predict the phenomenon compared to the traditional theories in the literature with definite shortcomings.

Empirical studies including but not limited to Roy (2018), Das et al. (2018), Dias et al. (2020), Vuković et al. (2024) further strengthen the fact that markets are not always efficient, and movements are not always random, and there's potential failure in both EMH and RWT, leaving room for prediction, at least for short-term.

2.3 Review of Relevant Literature on Developed and Emerging Markets

Emerging markets refer to the stock markets of developing countries that are undergoing rapid growth and development but generally have lower per capita income compared to developed countries. These markets are usually characterized by their transition from planned economies to free-market systems (Mody, 2003). While emerging markets encompass some of the world's largest economies, including China and India, they exhibit distinct characteristics that set them apart from developed markets. According to Bekaert & Harvey (1997), emerging markets are known for higher volatility, which often results in greater average returns compared to their developed counterparts.

Empirical studies like Harvey (1995) and Risso (2009) consistently found the higher level of predictability within emerging markets as a result of higher degree of market inefficient. Supporting this view, Haque et al. (2004) observed that most Asian emerging markets display significant levels of predictability, with results decisively rejecting the weak form of market efficiency. This contrast is usually attributed to structural differences, because developed markets are beneficial from advanced infrastructure, greater technology, therefore, faster information dissemination. All of which contribute to higher market efficiency in developed markets. Nevertheless, Sharma and Thaker (2015) pointed out that despite these disparities, global markets tend to exhibit weak form efficiency in the long run. This in turn, suggests that predictability may diminish over extended periods. After reviewing past research on predictability different levels of market maturity, in the next section, we are going to explore existing literature relating to predictive models within the financial landscape.

2.4 Current State of Knowledge

2.4.1 Embedding Neural Networks Concept into Stock Price Prediction

Neural Networks (NNs) are a machine learning approach inspired by how animal brains function biologically. In a natural neuron, dendrites receive input signals, which are processed in the cell body, and then an axon transmits the output signal. The axons of one neuron typically connect to the dendrites of another, facilitating the exchange of signals. Depending on the type of input received, the neuron may either generate an action potential (excitation, where electrical activity is transmitted along its axon) or remain inactive (inhibition). The strength of the response between neurons can adjust over time, enabling the brain to learn and store information (Ashwell, 2012; Zhu & Zhaoxiang, 2017). Individually, while a single biological neuron performs a straightforward process, receiving inputs, integrating them, and deciding whether to generate an action potential, the interconnected network of neurons can accomplish remarkable tasks such as recognizing faces, interpreting language, writing poetry, solving problems, producing art, and even making scientific discoveries, which showcased the strength of NNs.

Artificial Neural Networks (ANNs) are computational models that therefore designed to replicate the functionality of biological neurons. Similar to natural neurons, artificial neurons have input nodes (which is the dendrite in biological terminology), where each input is multiplied by a weight that signifies its influence. The artificial neuron then combines these weighted inputs with an additional value called bias, sums them, and applies a non-linear activation function. This function determines the neuron's output (biologically an axon) and allows ANNs to identify complex patterns and relationships within data (Zurada, 1992). Without this feature, an ANN would only be capable of modelling linear behaviours, akin to linear regression. Although a single artificial neuron is underwhelming simple, connecting many neurons across layers can create a powerful system capable of learning from data (Hornik et al., 1989).

ANNs can be employed in both supervised and unsupervised learning contexts. In supervised learning, labelled datasets guide the network to predict specific outputs, whereas in unsupervised learning, the network uncovers patterns in data without predefined labels (Bishop, 2006). The foundational work on artificial neurons began with McCulloch and Pitts (1943), who developed a computational model based on threshold logic. The field advanced significantly in the 1950s and 1960s with theoretical developments (Hush & Horne, 1993), but a major breakthrough occurred in 1975 when Paul J. Werbos introduced the backpropagation algorithm in his PhD thesis, a method for training ANNs by minimizing the error between predicted and actual outputs (Werbos, 1990; Rumelhart et al., 1986; Hecht-Nielsen, 1987; Hinton, 1987; Hopfield & Tank, 1986; Hopfield, 1982). Backpropagation enables the efficient training of a NN from a set of examples and uses gradient descent to adjust the network's weights iteratively, enhancing its learning capabilities. However, although this algorithm was a significant advance, this algorithm has notable challenges, which is that it suffers the vanishing gradient problem. This issue arises when small gradients prevent effective weight updates, hindering the network's ability to learn, especially in deep architectures (Hochreiter, 1998).

Additionally, backpropagation can lead to overfitting, where a case that the network memorizes the training data but performs poorly on new, unseen data (Tzafestas, 1996). A neural network is said to be overfitted when it "memorizes" the training dataset and is able to correctly predict the "desired" outputs for it but performing poorly when exposed to different data exterior to the training set. This is because, the NN has learned the training data's noise and specific details instead of generalizable patterns. Overfitting is typically observed when the gap between the training error and test error becoming widen as training progresses, but it can be controlled by limiting the number of training epochs (Goodfellow et al., 2016). Besides, the likelihood of overfitting increases as the topology of the network becoming more complex. This means that the learning capacity of NN is inherently limited, since in practice, the backpropagation algorithm struggles to effectively train complex network topologies (Andres et al., 2021).

However, significant theoretical along with computational advancements in NN research has resolved the challenges of traditional backpropagation in the recent couple of decades. This includes the introduction of new non-linear activation functions which has mitigated the vanishing gradient problem (Clevert et al., 2015). New regularization techniques, including dropout, batch normalization, and data augmentation, have reduced the risk of a NN to overfitting (Zaremba et al., 2014). Furthermore, improvements in stochastic gradient descent methods have also enhanced the optimization process for weight updates (Johnson & Zhang, 2013). These advancements, coupled with the increased computing power of modern hardware, have enabled the training of larger and more complex networks, leading to the emergence of Deep Neural Networks (DNNs).

DNN is a type of NN that are characterized by having multiple hidden layers between the input and output layers, represent a significant evolution in neural network design. These networks have produced a revolution in the field of machine learning due to its great capacity for generalization and learning. Their applications range from image and speech recognition to drug discovery and genomics, making them a cornerstone of contemporary artificial intelligence research (LeCun et al., 2015). Within the financial landscape, DNN have demonstrated exceptional capabilities in financial time-series forecasting. Studies comparing Recurrent Neural Networks (RNNs) and traditional ANN models consistently show that DNNs provide superior performance for financial market predictions (Singh & Srivastava, 2016).

Recently, deep learning techniques have gained prominence in being on top of the prediction of stock market trends among the global financial market (Shahi et al., 2020). While the efficient market hypothesis suggests that price movements follow random paths, making predictions theoretically unfeasible, but empirical works challenge this view. Henrique et al. (2019) suggests that factors such as psychological influences and the immature nature of some markets enable the identification of predictable patterns. Conventionally employed financial forecasting methods, namely technical and fundamental analysis, have also been complemented by classic time-series models like the most popular ARIMA (Jiang, 2021; Kumar & Thenmozhi, 2014). Yet, the non-linear, unstable, and noisy characteristics of financial time series data have further thrived the evolution of deep learning techniques for more accurate forecasting (Hsu et al., 2016; Bezerra & Albuquerque, 2017; Zhang et al., 2017; Shah & Zulkernine, 2019).

Kim et al. (2021) compared linear regression, non-linear regression, support vector machines (SVM), random forests, and neural networks for forecasting corporate bond yield spreads. Their findings indicate that neural networks (deep learning models) outperform all other machine learning methods. Similarly, Gao and Chai (2018) concluded that RNNs deliver the most accurate stock index predictions. The versatility of neural networks has also led to the development of hybrid models that amalgamate various models. Sun et al. (2019) proposed a model combining the auto-regressive moving average (ARMA), generalized auto-regressive conditional heteroskedasticity (GARCH), and neural networks to analyze high-frequency data from the US stock market, while Huynh et al. (2017) introduced a bidirectional gated recurrent unit (BGRU) model to explore the relationship between investor sentiment and stock prices.

LSTM and GRU have also garnered significant attention for financial time-series forecasting due to their ability to handle both long and short- term temporal dependencies. Generally speaking, LSTMs and GRUs are advanced forms of RNNs that able to effectively address the challenges posed by the non-linear nature of financial markets. As a result, they are frequently employed in predicting securities prices and market indexes (Zhang et al., 2018; Kamal et al., 2020).

2.4.2 Empirical Studies on LSTM and GRU in Stock Market Prediction

Over the course of this decade, with advances in hardware and cloud conditions, there has been a proliferation of research utilizing deep learning models for stock index prediction.

In one notable study, He & Zhang (2024) benchmarked four deep learning models: LSTM, MLP, CNN, and GRU, against 14 internationally representative stock indexes. These indexes including but not limited to the Dow Jones, S&P 500, Nasdaq, Germany's Dax, and China's SSE A Index. Their experimental design incorporated short-term (20 trading days), mid-term (60 trading days), and long-term (250 trading days) prediction horizons. Notably, their findings revealed that the GRU model consistently delivered outstanding performance in both predictive accuracy and generalization capabilities.

Echoing those results , Dey et al. (2021), reached similar conclusions in their analysis comparing simple RNN, LSTM, and GRU models using data from three different companies between June 2000 and July 2020. Forecasting over one-day, three-day, and five-day intervals, they observed that both LSTM and GRU models surpassed simple RNN in performance, with GRU yielding the lowest error rates, particularly for short-term predictions. Clearly, when it comes to capturing fast-changing market behaviour, GRU holds a distinctive advantage.

Meanwhile, in a study focused on Nepal's financial sector, Saud & Shakya (2020) assessed Vanilla RNN, LSTM, and GRU architectures applied to the two most prominent commercial banks listed on the Nepal Stock Exchange (NEPSE). Their findings reinforced the performance of GRU, which recorded the lowest average MAPE values of 4.74 and 4.71, signalling its effectiveness in modelling financial time series within this regional context.

Shen et al. (2018) contributed another layer to the GRU success story. They applied two GRU-based models to forecast trading signals for indexes such as HSI, DAX, and the S&P 500, working with data spanning from 1991 to 2017. Their findings revealed that the GRU models outperformed Support Vector Machine (SVM) and other traditional models in prediction accuracy, contributing another piece of literature on GRU’s promising performance.

That said, not all studies agree GRU as the dominant model. Nabipour et al. (2020) provided a contrasting perspective with their investigation of the Tehran Stock Exchange. Using a decade’s worth of data (2009–2019), along with ten technical indicators, they evaluated LSTM, ANN, and RNN models across forecast windows ranging from 1 to 30 days. In this case, LSTM emerged as the clear winner, outperforming its peers in every error metric, recording an RMSE of 0.0093, MAPE of 0.60, and MAE of 6.70.

Another strong case for LSTM was presented by Maiti and Shetty (2020), who employed a deep LSTM model composed of four hidden layers to predict stock movements and closing prices in the Turkish Stock Exchange. Their data, collected at five-minute intervals between 2014 and 2019, was enriched with variables such as open, close, high, low prices, volume, and technical indicators, using a window size of 50. The results were exceptional: price movement prediction accuracy ranged between 97.53% and 98.91%, and RMSE for closing price forecasts varied from just 0.024 to as low as 0.0048. These findings highlighted LSTM's capacity to effectively capture the complexities of stock market data.

Nikou et al. (2019) in contrast, directly compared LSTM with other machine learning models such as MLP, SVR, and Random Forest Regression, using data from the MSCI United Kingdom stock index from 2015 to 2018. With a 10-day window for input features, LSTM was again the strongest contender. It reported the lowest values across all major error metrics: MAE of 0.210, MSE of 0.094, and RMSE of 0.307.

A similar conclusion was reached by Samarawickrama and Fernando (2017), who evaluated simple RNN, LSTM, and GRU models to forecast stock prices for three companies on the Colombo Stock Exchange. The input variables were limited to closing, high, and low prices over the preceding two days—yet the results still favoured LSTM compared to other architectures. While feedforward networks achieved forecasting accuracy as high as 99%, LSTM emerged as the most accurate among the recurrent models. GRU, by contrast, produced comparatively higher forecasting errors.

Moving on to empirical studies on ensemble models, Sulistio et al. (2023) conducted research to evaluate the performance of six deep learning algorithms for predicting stock closing prices using individual models like CNN, LSTM, and GRU and hybrid models. Their work focused on stocks in the energy sector listed on the Indonesian Stock Exchange. By comparing individual models against hybrid combinations, they found that the CNN-LSTM-GRU hybrid algorithm delivered the best results among all, even when it is compared with other hybrid algorithms such as CNN-LSTM. Specifically, it achieved a 14% reduction in RMSE (by 1.100 points), a 13.4% decrease in MAE (by 0.798 points), and a 3.9% increase in R^2 (to 0.957). Their findings suggest that the CNN-LSTM-GRU hybrid is a more appropriate tool for investors than relying on single algorithm to make investment decision.

Looking at novel configurations, Karim & Ahmed (2021) proposed a new deep learning-based forecasting framework called the Bidirectional Gated Recurrent Unit (BiGRU) model, enhanced with an external activation layer to leverage both forward and backward propagation features of the RNN. They compared the performance of their proposed BiGRU model with the widely used Bidirectional Long Short-Term Memory (BiLSTM) model. The models were implemented on three different datasets collected from the NIFTY-50 index. Using five different evaluation metrics, the results showed that the BiGRU model consistently outperformed the BiLSTM model across various hidden layer configurations and datasets. Additionally, BiGRU demonstrated better stability, had fewer trainable parameters, and was able to forecast stock prices accurately for longer prediction windows, including successfully predicting sudden spikes up to 1000 days ahead.

Zulqarnain et al. (2020), on the other hand, introduced a hybrid model that combines the strengths of CNN and GRU architectures. Their approach first extracts features through CNN layers before capturing temporal dependencies with GRU layers, effectively addressing challenges like vanishing and exploding gradients commonly found in standard RNNs. The stock indexes chosen is HSI, DAX, and S&P500, using historical data spanning from 2008 to 2016, and employing a 250-day moving window length. The GRU-CNN model achieved an accuracy of over 50% for all indexes, scoring 56.2% for HSI, 56.1% for DAX, and 56.3% for the S&P 500, slightly surpassing the baseline.

Financial news sentiment (which is also known as sentiment analysis) alongside stock features to predict trends and prices for the S&P 500 Index, were integrated in the proposed two-stream GRU model by Minh et al. (2018). They achieved an overall accuracy of 66.32%. This approach demonstrated superior performance compared to other models, including the standard GRU and LSTM models. Interestingly, the authors noted the notable shortcoming that the two-stream GRU model demanded extensive computational resources and longer training times due to its augmented complexity.

Thoroughly reviewing the existing literature, it is found that despite the widespread investigation relating to stock market prediction, significant gaps remain in cross-market index comparisons (developed and emerging markets). Multiple studies have only been concerned with either developed markets or single market regions independently, or altogether without separating developed and developing ones. Despite the fact that LSTM and GRU models frequently appeared in stock prediction studies, direct comparisons of their effectiveness across developed and emerging markets like this research are rather scarce. This lack of comprehensive analysis hinders a more profound perspective of how these models perform in varying economic and financial contexts, leaving the unexplored area in the literature that deserves further investigation. By addressing this gap, this study aims to addresses the missing information regarding the comparative strengths of LSTM and GRU models for stock index prediction across ten countries, equally distributed into developed and emerging countries.

Table 1: Literature Review Matrix

Reference	Model Choice	Dataset	Findings
He & Zhang (2024)	LSTM, MLP, CNN, GRU	14 global stock indices (e.g., Dow Jones, S&P500, Nasdaq, SSE A Index); Short (20 days), Mid (60 days), Long (250 days) horizons.	GRU consistently showed outstanding performance in predictive accuracy and generalization capabilities across all horizons.
Dey et al. (2021)	Simple RNN, LSTM, GRU	Stock prices from 3 companies (June 30, 2000 - July 21, 2020); 1-day, 3-day, 5-day intervals.	LSTM & GRU outperformed Simple RNN. GRU produced the lowest error rates, especially for shorter horizons, indicating superior efficiency for short-term dynamics.
Saud & Shakya (2020)	Vanilla RNN, LSTM, GRU	Two strongest commercial banks on Nepal Stock Exchange (NEPSE).	GRU outperformed Vanilla RNN and LSTM, achieving the lowest average MAPE (4.74 & 4.71).
Shen et al. (2018)	Two GRU models	HSI, DAX, S&P 500 Index data (1991 - 2017); Predicting trading signals.	GRU models outperformed SVM and other traditional models in prediction accuracy.
Nabipour et al. (2020)	LSTM, ANN, RNN	Tehran Stock Exchange data (2009–2019); Input: 10 technical indicators + stock prices; Timeframes: 1-30 days.	LSTM was the most effective model across all forecast durations, achieving lower RMSE (0.0093), MAPE (0.60), and MAE (6.70) compared to ANN and RNN.
Maiti & Shetty (2020)	LSTM (4 hidden layers)	Turkish Stock Exchange data (2014-2019), 5-minute intervals; Input: OHLC, volume, technical indicators; Window size: 50.	High accuracy (97.53%-98.91%) in predicting price movements; Excellent performance (RMSE 0.024-0.0048) in forecasting close prices. Highlighted LSTM's capacity.
Nikou et al. (2019)	LSTM, MLP, SVR, RFR	MSCI United Kingdom stock index data (2015-2018); 10-day window size.	LSTM outperformed MLP, SVR, and RFR in accuracy, achieving the lowest

			MAE (0.210), MSE (0.094), and RMSE (0.307).
Samarawickrama & Fernando (2017)	Simple RNN, LSTM, GRU	Three companies on Colombo Stock Exchange (CSE); Input: Closing, high, low prices (previous 2 days).	LSTM outperformed other architectures (lower errors, ~99% accuracy for best feedforward nets). GRU yielded comparatively higher forecasting errors.
Sulistio et al. (2023)	CNN, LSTM, GRU (individual); CNN-LSTM, CNN-LSTM-GRU (hybrid)	Energy sector stocks on Indonesian Stock Exchange; Predicting closing prices.	CNN-LSTM-GRU hybrid delivered the best results (lower RMSE/MAE, higher R^2 by 14%, 13.4%, 3.9% respectively) compared to individual models and other hybrids.
Karim & Ahmed (2021)	BiGRU (proposed), BiLSTM	Three datasets from NIFTY-50 index.	Proposed BiGRU consistently outperformed BiLSTM across metrics, configurations, datasets. More stable, fewer parameters, better long-term forecasting (up to 1000 days).
Zulqarnain et al. (2020)	CNN-GRU hybrid	HSI, DAX, S&P500 data (2008-2016); 250-day moving window.	Hybrid model achieved accuracy >50% (56.2% HSI, 56.1% DAX, 56.3% S&P 500), slightly surpassing baseline. Addresses gradient issues.
Minh et al. (2018)	Two-stream GRU (stock features + news sentiment)	S&P 500 Index; Predicting trends and prices.	Achieved 66.32% accuracy, outperforming standard GRU and LSTM. Noted high computational cost and training time.

2.5 Summary of Literature Review

This chapter of literature review first introduces the theoretical grounding of the research, which is the EMH and RWT, revealing the existing critiques with the support of Behavioural Finance. Subsequently, the chapter discusses literature on the predictability between developed and emerging markets, highlighting the fact that emerging markets typically more inefficient, thus, have forecasting potential. Next, the chapter converge in how neural networks are being embedded into stock market predictions, introduce their origin. After that, it reviews empirical studies that employed LSTM and GRU models for stock market prediction, and reveals remarkable gaps in cross-market comparative analysis for stock market index prediction with these two models. While existing studies have predominantly focused on single markets or isolated regions, there is a considerable absence of systematic investigation that explore how these deep learning models adapt to different economic backgrounds. Thus, this research addresses a scholarly need for improved understanding of LSTM and GRU model performance across established and developing markets.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

At the beginning, this section details the approach, methodology, and procedures used by us to collect and utilize data, and to train the prediction models subsequently. This section is particularly important because it assures that the study is conducted rigorously and systematically, allowing further scholars to replicate the study and achieving similar results by following the same procedures with identical data. The prediction models are implemented using Google Colab, a cloud-based platform provided by Google LLC that provides Python development tools and parallel computing across devices. Besides, this section also explains the various hyperparameters selected throughout the experiments with underlying justifications.

3.2 Research Methodological Choices

Similar to other predictive modelling studies, this research focuses on the prediction of time series data using mathematical methods, highlighting its quantitative nature. Using a quantitative approach, this research leverages ample financial data to construct deep learning models and enable the assessment of prediction accuracy through statistical metrics.

Besides, rather than employing primary data collection method, this study relies on secondary data sourced uniformly from Yahoo Finance, a highly recognized and reputable financial data provider. This choice allows access to extensive historical daily stock index data, which is publicly available and verifiable. Utilizing such secondary data enhances the reliability and replicability of the study, as the data is widely accessible for other researchers or practitioners aiming to validate the findings. Also, to ensure the study's reliability and practicality, the most up-to-date available data is chosen.

3.3 Data Collection, Diagnostics and Processing

The development of a prediction model begins with gathering historical daily stock index data for selected stock indexes through the Yahoo Finance API, which is accessible using Python in the environment of Google Colab. One of the rationales of using Google Colab is that it offers an easy-to-configure setup and free access to T4 GPUs, which enables the efficient execution of deep learning algorithms (Khalil & Bakar, 2023; Chollet, 2021).The Google Colab is embedded with the latest Python version, Python 3.10.12.

The downloaded raw datasets contain multiple fields: Open, Close, High, Low, Volume, Dividends, and Stock Splits. However, only the "Close" prices are needed for training the models as they represent the final price at which the stock traded every day. Therefore, the other irrelevant data columns would be discarded through the filter function embedded in Python to ensure the data remains relevant and manageable, reducing unnecessary.

In term of sample size, the selected time frame spans 15 years, from the beginning of 2010 to the end of 2024, covering key global market events such as the outbreak of COVID-19 pandemic, a major bear market troughed early 2020, and two significant bull markets—one post-2008 global financial crisis and another following the post-COVID-19 recovery in 2020. This broad time frame, incorporating both bear and bull markets, provides a balanced market circumstance and can potentially make the model capable to generalize across diverse economic cycles which may enhance predictive accuracy, according to Bhandari et al. (2022).

Apart from that, to achieve a balanced representation of global markets, the stock indexes chosen include equal distribution across diverse geographical regions, including America (AMER), Europe, the Middle East, and Africa (EMEA), and Asia Pacific (APAC), between developed and emerging economies based on MSCI classifications, as Table (2) below shows. This approach of selecting developed and emerging markets was also used in study by Sharma & Thaker (2015), since MSCI has long been a reputable provider of critical decision support tools and services for the global investment community.

Table 2: 10 Stock Indexes Selected Across Developed and Emerging Markets

Regions	**Developed Markets**	**Emerging Markets**
AMER	(1) United States (S&P 500 Index)	(1) Brazil (IBOVESPA), (2) Mexico (IPC Mexico)
EMEA	(2) United Kingdom (FTSE 100 Index), (3) Germany (DAX)	(3) Saudi Arabia (Tadawul All Shares Index)
APAC	(4) Hong Kong (Hang Seng Index), (5) Australia (ASX 200)	(4) Malaysia (FTSE Bursa Malaysia KLCI), (5) China (SSE Composite Index)

Before the data processing step, to better understand the patterns of the time series, diagnostics testing, such as the Augmented Dickey-Fuller (ADF) test that used to test the stationarity of the time series should be applied. To determine the result, the test statistic is compared against critical values at the most recognized confidence level of 5%. Hence, if the p-value from the ADF test falls below the chosen threshold, the null hypothesis is rejected, indicating that the data can be regarded as stationary (Alamu & Siam, 2024). Despite deep learning models do not need to ensure that the time series is stationary, a stationary time series is said to likely yield better results even with drastic change, as it is easier for the model to learn (Amiri et al., 2025).

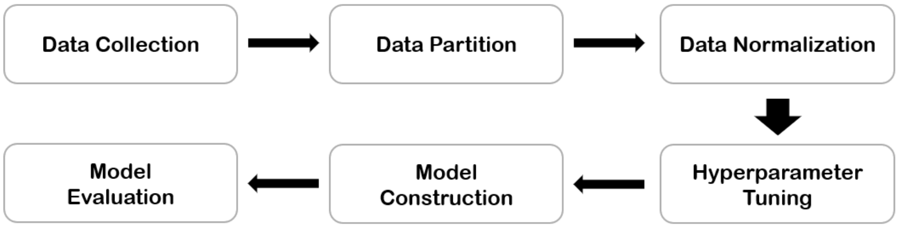
Moving on to data processing, the dataset is split into two subsets, which is the training and testing set. The initial 80% of the data is allocated for training in order to provide the model with ample data for learning, while the remaining 20% portion is reserved to assess their predictive accuracy. This division is a standard practice in time series modelling, as it ensures that the model is trained on a sufficient portion of historical data while reserving unseen data to objectively assess its generalization performance. This split ratio of 4:1 is suggested by Zulqarnain et al. (2020) who was working in a similar research direction to us and verified that this ratio could yield the best result.

Next, to facilitate effective model training, the closing price data is normalized and scaled into a range of (0, 1) using MinMaxScaler function. In essence, this data normalization approach named min-max normalization, and it is used to rescales the input data range, where the maximum value of the data will be transferred to 1, and the minimum value will be transferred to 0. This step is indispensable in preprocessing the data for training deep learning models, as neural networks require the use of activation functions, which usually scales the value between 0 and 1. The min-max normalization techniques can be described through the following equation:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where x' is the scaled data after normalization, while x is the original data input. The minimum and maximum value of the input is represented by x_{min} and x_{max}, respectively (Bhandari et al., 2022; Rahman et al., 2019). The reason of choosing min-max normalization is because it has advantage over other techniques in preserving the relationship in the original data values, as mentioned by Hadhood (2022).

Figure 1: Flowchart of Process of Implementation



3.4 Construction of Prediction Models

This section is going to describe in detailed how the prediction models are constructed. First of all, the tuning of hyperparameters, which are external configurable settings whose values are set before the learning process begins, should be done.

3.4.1 Hyperparameter Tuning and Experimental Setup

This stage is critical in this study due to the nature that deep learning networks are very sensitive to hyperparameter. Different hyperparameter choices yield varying results which could potentially lead to opposite conclusions (Mateus et al., 2021; Sutarna et al., 2024). It is worth restating that this project is not to identify which combination of hyperparameter settings can produce the most favourable predictive accuracy. Instead, the primary focus is on comparing predictive performance across developed and emerging markets, as well as between LSTM and GRU models. Thus, it is of paramount to maintain consistent hyperparameter tuning throughout the research to ensure fair and reliable comparisons.

Following the earlier stage of data processing including normalization and partition, a moving window (or sliding window) size of 250-day has been chosen in this research. In this experimental setup, each input sequence consists of 250 consecutive data (or commonly referred to as one year) will be utilized to predict the stock index level on the following day (the 251st day). This selection of window size is guided by Zulqarnain et al. (2020), whose work also informs our data partitioning ratio previously. The design allows the model to identify implicit patterns over each 250-day period, which it then uses to predict the next value. By setting the window to 250 days, the model would be able to capture long-term dependencies in stock index movements, which is essential for learning meaningful patterns in deep learning. Nonetheless, from a research perspective, the choice of window size represents a trade-off: despite larger windows may incorporate more historical information, but there is risk of introducing noise, whereas smaller windows may overlook important trends (Saeed & Yin, 2025).

As for the choice of prediction length, we referred to the prior study by He & Zhang (2024), whose research direction is similar to us as well, because Zulqarnain et al. (2020) did not mention explicitly the length of predictions they used in their research. The findings by He & Zhang (2024) suggest that both the prediction results of the length of 20-day (roughly a month) and 60-day horizons produced only minimal deviations in results for LSTM and GRU models, which were negligible. Comparatively, the 250-day prediction length resulted in higher forecast errors. Therefore, we select **20 days** as our prediction length, as it is considered a common sense in finance that longer periods often associate with higher uncertainties. That's why 30 years government bond usually yielded better returns compared to 10 years one. Also, a 3% threshold is set to define poor predictive performance, which we shall refer to during the discussion of experimental results in Chapter 5 later on (He & Zhang, 2024).

Another important consideration is the inherent non-deterministic nature of neural networks. Due to random initialization in neural networks and the way GPUs compute gradients, as discussed by Hoang and Laskemoen (2020), Goodfellow (2016), and Chollet (2021), models rarely produce same results even if the exact same input data is provided along with constant hyperparameters. In order to address this issue, each model in this study will be trained 10 times for every stock index (so there will be 2x10x10 = 200 runs in total). This approach was also adopted by Liu et al. (2021) and Bilevich (2023), who reiterated the model 10 times and averaged the results to obtain more robust and reliable value as the final prediction errors. Up to this point, we have preset all the external hyperparameters used in the model. Subsequently, we have to set the "internal hyperparameters" or known as the model's architecture. However, before setting them, we should first know how the model operates for clarity purpose.

3.4.2 Mechanism of LSTM and GRU Operation

Since in essence, both LSTM and GRU are kind of RNNs, we had better introduce them before further diving. The foundational concept on which the RNN was developed is their 'memory' feature, which retains information from preprocessing steps that is essential for accurately predicting subsequent values (Hochreiter & Schmidhuber, 1997). Simply speaking, RNNs process sequential data by repeatedly applying the same operation to each element in the sequence, and the output of each step depends not only on the current input but also on information from previous steps. In the context of financial markets, this design theoretically enables the model to learn patterns from past stock prices and use them to predict future movements. This memory mechanism is managed through a technique called backpropagation through time (BPTT), which adjusts model weights based on prediction errors, a complicated algorithm used in RNNs to calculate gradients and update network weights (Manjunath et al., 2021). At each timestep t, RNN reads a new index input, X_t, and updates its hidden state h_t (Chung et al., 2015). The value of the hidden state of RNN is often expressed by Equation (1), where h_{t-1} represents the hidden state from the previous time step:

(1) $$h_t = f(X_t, h_{t-1})$$

Nevertheless, RNNs is known for struggling in handling long-term dependencies. In predicting stock market trends based on historical data, simple RNNs often underperform due to their inability to retain information over long sequences, which is caused by the vanishing gradient problem during backpropagation (Ryu, 2024; Kratzert et al., 2019). This is one of the significant shortcomings of traditional RNNs, which is their short-term memory characteristic (Hochreiter et al., 2007). Owing to the gradient vanishing, the training does not occur properly because the model parameters are not updated, since gradients used during training become too small to influence weight updates (Ryu, 2024). Therefore, in order to address the limitations, two models, the LSTM and GRU were introduced as solutions, which is also the models that are employing for the current study (Kratzert et al., 2019; Cho et al., 2014).

3.4.2.1 Long Short-Term Memory

LSTM was introduced by Hochereiter et al. (2007) to specifically address the issue of long-term dependencies in RNNs. Over time, it has been further refined and gained popularity among researchers. Sherstinsky (2020) pointed out that why LSTMs are well-suited for handling long-term dependencies. It is because, they perform exceptionally well across a wide range of tasks and selectively retain information from earlier sequences. Like other RNNs, LSTMs have a chain-like structure in which information is passed sequentially from one node to the next, forming loops to retain past data for predicting future outcomes. However, unlike traditional RNNs with a single layer, LSTMs incorporate multiple layers that interact to filter and regulate information, keeping only the relevant data and discarding the rest (Greff et al., 2016).

What makes LSTMs distinguishable is their key feature of the cell state. Acting like a conveyor belt, the cell state maintains a "memory" that travels through the network, connecting past time steps with future ones. This setup allows information to move through the network with minimal alteration, effectively addressing long-term dependencies. LSTMs achieve this through a structure of four key components, known as gates. These gates selectively allow certain information to pass to the cell state while hindering irrelevant data (Greff et al., 2016).

Forget Gate Layer

In stock market data, not all past information is equally important. For instance, a market correction from a year ago might be less relevant than recent volatility. Thus, filtering out irrelevant noise is indispensable. The initial interaction with the cell state begins through the forget gate layer, which decides what information should be retained and what should be discarded. As data enters the first network layer, it is processed using a sigmoid activation function. For each previous cell state C_{t-1}, the sigmoid function evaluates both the prior hidden state h_{t-1} and the current input x_t to assess relevance. It assigns a probability score indicating whether the information should be preserved in the cell state or disregarded. An output of close to zero signifies discarding the information, while a result close to one implies that it should be kept (Van Der Westhuizen & Lasenby, 2018). Equation (2) showed the equation of the forget gate layer.

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

Input Gate Layer

The input gate determines which new information from the current market state should be added to the cell's memory. For example, a sudden spike in trading volume or an unexpected interest rate cut might be considered highly relevant and should influence the model's forecast.

This process involves three steps. First, the previous hidden state h_{t-1} and current input passed through a sigmoid function, which generates a probability between 0 and 1 based on the relevance of the incoming data. It's like generating an "importance score" for the new input. The second step involves creating a candidate value vector C_t through the hyperbolic tangent (*tanh*) activation function, which constrains values between -1 and 1, that represents the potential impact of the new market data. Finally, the output from the sigmoid function is multiplied by the candidate vector produced by the *tanh* output C_t, thereby incorporating the newly selected information into the cell state (Greff et al., 2016). This can be written as:

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

(4) $$C'_t = tanh(W_C[h_{t-1}, x_t] + b_C)$$

where W_i and W_C are the parameters.

Cell-state (Long-term Memory)

At this stage, through the actions of the forget gate layer f_t and the input gate layer i_t, the network decides which information to discard from the previous cell state C_{t-1} and which to add to the current cell state C_t. However, these decisions only take effect when the previous cell state is multiplied by the forget gate output f_t. If the output value is close to zero, the cell state will lose that information. Then, the result from the input gate layer i_t is added to the updated cell state C_t, completing the update (Hochreiter et al., 2007).

(5) $$C_t = f_t * C_{t-1} + i_t * C'_t$$

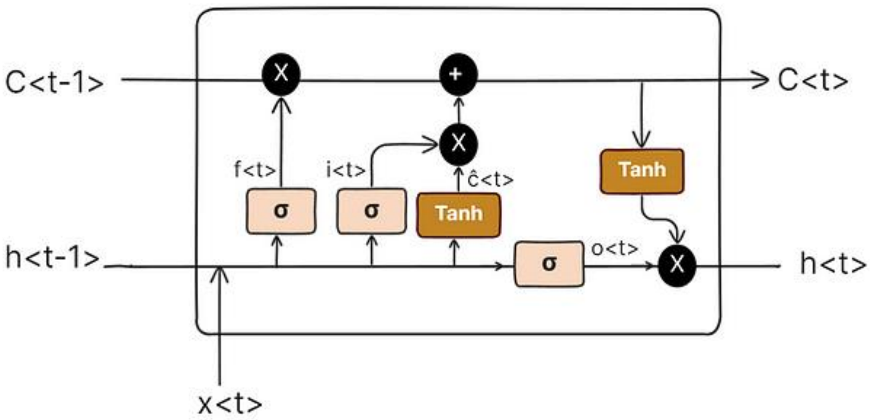
Output Gate

Once the cell state has been updated, the final operation is that the output gate helps determine the information to propagate through the hidden state h_t. This step blends the newly updated memory with a filter that highlights the most useful information for making the next prediction. To compute the hidden state, the prior hidden state and current input process through a sigmoid function. Meanwhile, the updated cell state is copied to a *tanh* function, which outputs values for the hidden state without altering C_t itself. The outputs of the sigmoid and *tanh* functions are then multiplied, producing the hidden state to be passed to the subsequent time step (Hochreiter et al., 2007).

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

(7) $$h_t = o_t + tanh(b_f)$$

Figure 2: Illustration of LSTM Model



In essence, the LSTM model functions like a financial analyst who continuously reviews historical data, filters out noise, highlights recent impactful events, and uses a blend of long- and short-term indicators to form a market outlook.

3.4.2.2 Gated Recurrent Unit

GRU, introduced by Cho et al. (2014), in contrast, is another recurrent neural network which quickly gained attention in the scientific community as a simplified, more computationally efficient version of the LSTM model. Like LSTM, GRU operates on similar configurations, but it integrates the cell state and hidden state into a single "conveyor belt" and includes only two gates, which is the update gate and the reset gate. The update gate controls the retention of past information, while the reset gate determines the extent to which prior information is forgotten, making it akin to LSTM but with some subtle differences (Hadhood, 2022).

GRU's learning algorithm is closely mirrors that of LSTM's, processing each input x_t and deriving the hidden state from the previous one h_{t-1}, then applying the reset and update gates. Finally, the updated hidden state h_t is output and passed to the next step in the sequence.

The GRU process primarily consists of two stages, with the first involving the computation of the candidate hidden state, as illustrated in Equation (8).

(8) $$h'_t = tanh(W_r(h_{t-1}r_t) + W_f x_t + b_h)$$

To compute the candidate hidden state, GRU multiplies the previous hidden state h_{t-1} with the reset gate's output r_t and combines it with the current input. This result is then passed through a *tanh* activation function, producing the candidate hidden state. The reset gate value r_t plays a critical role here:

- when r_t is 1, all information is preserved;
- when r_t is 0, information from the previous hidden state is ignored (Cho et al., 2014).

In the context of financial market where recent news or macroeconomic events may render past patterns temporarily irrelevant, this process is useful. For instance, during earnings season or interest rate announcements, past stock price movements might lose predictive power. The reset gate ensures the model does not over-rely on stale information, enabling it to adapt to abrupt market shifts. The second phase involves creating the current hidden state by using the update gateto balance between historical information from h_{t-1} and the newly generated candidate hidden state (Cho et al., 2014).

- If the update gate is set to 0, the current hidden state h_t becomes the candidate hidden state, excluding previous information;
- If the update gate is set to 1, the candidate hidden state is disregarded, and h_t relies entirely on h_{t-1} (Cho et al., 2014).

(9) $$h_t = (1 - z_t) * (h_{t-1}) + Z_t h'_t$$

The mechanism here plays similar roles to how technical analysts may weigh long-term moving averages against more recent price action. Theoretically, in stable markets, retaining older trends (larger z_t) may benefit the prediction, while volatile markets require rapid adaptation to new data (smaller z_t), which GRUs can handle dynamically.

Reset Gate

Moving on, the reset gate is of important in capturing short-term dependencies within sequences, with its sigmoid activation function producing an output value r_t between 0 and 1. Similarly, a value closer to 1 keeps the information, while a value closer to 0 disregards it (Cho et al., 2014). This functionality mimics how short-term traders might ignore last month's stock movement in favour of today's market

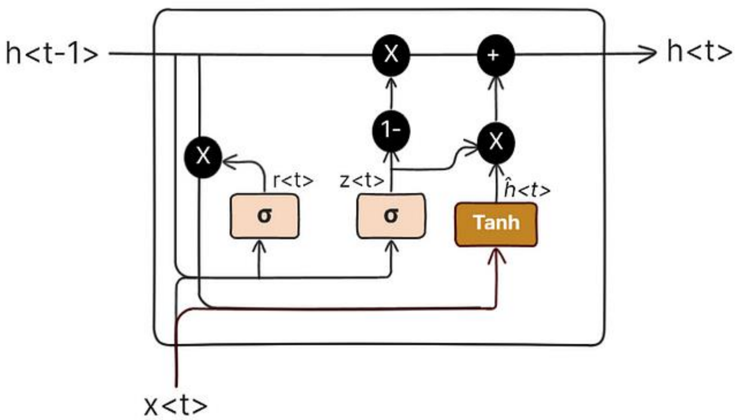
(10) $$r_t = \sigma(W_r h_{t-1} + W_r x_t + b_r)$$

Update Gate

On the flip side, the update gate is responsible for capturing long-term dependencies, as it decides whether information from h_{t-1} should be retained or replaced by the candidate hidden state when passing information forward to the next time step h_{t+1} (Cho et al., 2014).

(11) $$Z_t = \sigma(W_Z h_{t-1} + W_Z x_t + b_Z)$$

Figure 3: Illustration of GRU Model



3.4.3 Architecture of LSTM and GRU

Following by the comprehensive description of the working mechanism of LSTM and GRU, the final step for model construction is to set the architecture of models. In this project, we are going to construct a three-layer LSTM, each consisting of 128 unit of neurons and is configured to return sequences, while performing dropout regularization at a dropout rate of

0.2. This is designed to prevent the likelihood of overfitting while improving generalization ability. Similarly, each of the second and third LSTM layers also include 128 neurons, the different is that the second layer return sequences while the third layer does not return sequences, preparing the output for the following dense layers. The model ends with two dense layers: one with 8 units and the final output layer with 1 unit, designed to output the prediction for the stock price for the next day.

GRU is built upon an exactly similar manner (as our intention is make comparison under same conditions), the first two layers comprises 128 units, returns sequences, and applies a dropout rate of 0.2. It is then followed by the third GRU layer contains 128 units with 0.2 dropout rate but does not return sequences, concluding with two dense output layers, one with 8 neurons and one with a single neuron. The rationale of selecting these hyperparameters, again, is referred to He & Zhang (2024).

The compiled model applies the Adaptive Moment Estimation (Adam) optimizer with Mean Squared Error (MSE) as its loss function. Adam is a popular choice for optimization as they can adaptively adjust learning rates and is generally regarded as being fairly robust to the choice of hyperparameters according to Goodfellow et al. (2016), while MSE is a common choice for continuous regression tasks as it effectively penalizes larger errors more than small ones (the existence of “square” significantly magnified the error). Many prior studies have applied the identical combination, one of which is He & Zhang (2024). Dropout regularization is also incorporated into all LSTM (and GRU) layers, helping to mitigate overfitting by randomly omitting a fraction of connections during training.

Apart from that, an early stopping mechanism with a patience of 10 epochs is also implemented to prevent overfitting during training. This means that the training will be terminated once the validation performance ceases to improve for 10 consecutive epochs, while also restoring the weights from the best-performing epoch. Thus, without significantly sacrificing accuracy, the number of epochs required for training reduced (Anam et al., 2024). The initial epoch size is set to be 1000, which defines the maximum number of epochs the model will undergo during training (Chollet, 2021; Goodfellow et al., 2016).

Upon completion of training, the model generates predictions on the test data and transformed back to original price scale (as they were normalized at the initial stage) using the inverse transformation, allowing them to be compared directly to the actual prices.

Table 3: Summary of Hyperparameters

Hyperparameter/Setup	Setting/Value	Notes
Moving Window	250 days	Used as the input sequence length for prediction.
Prediction Length	20 days	The horizon for stock index level prediction.
Poor Predictive Performance Threshold	3%	Used for discussing experimental results.
Number of Training Runs per Model & Index	10 times	To address the non-deterministic nature of neural networks; results are averaged.
Optimizer	Adam	Common for continuous regression tasks, suggested by prior studies.
Loss Function	Mean Squared Error	Common for continuous regression tasks, suggested by prior studies.
Early Stopping Patience	10 epochs	Training stops if validation performance doesn't improve for this many epochs, to avoid overfitting.
Initial/Maximum Epoch Size	1000	The upper limit for the number of training epochs.

Table 4: Summary of LSTM and GRU Model Architectures

Architecture	Setting/Value	Notes
Number of Layers	3	Suggested by prior studies.
Units (Neuron) per Layer	128	Consistent across all three LSTM layers.
Return Sequence (Layer 1 & 2)	True	Output sequence is passed to the next layer.
Return Sequence (Layer 3)	False	Prepares output for dense layers.
Dropout Rate	0.2	Applied for regularization.
Number of Dense Layers	2	Suggested by prior studies.
Units in First Dense Layer	8	Suggested by prior studies
Units in Output Dense Layer	1	Predicts the stock price for the next day.

3.5 Model Evaluation Metrics

Evaluating model performance is a vital element in any research relating to prediction, since it determines the reliability and precision of the predicted results (Saboor et al., 2023). In this study, model performance is assessed using three standard statistical evaluation metrics commonly applied, which include Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and the Coefficient of Determination (R^2). The equation of these three metrics can refer to (12) to (14) stated below. These metrics serve as key indicators to measure each model's predictive accuracy by quantifying the prediction error over the observation periods. RMSE, MAE, and R^2 are well-regarded in stock market prediction studies due to their ability to effectively gauge the precision of model forecasts based on the discrepancy between predicted and actual values (Bustos & Pomares-Quimbaya, 2020; Gandhmal & Kumar, 2019).

(12)
$$RMSE = \sqrt{\frac{1}{N}\sum_{d=1}^{N}(a_d - p_d)^2}$$

(13)
$$MAE = \frac{1}{N}\sum_{d=1}^{N}|a_d - p_d|$$

(14)
$$R^2 = 1 - \frac{\sum_{d=1}^{N}(a_d - p_d)^2}{\sum_{d=1}^{N}(a_d - a)^2}$$

In detail, RMSE calculates the square root of the average of squared differences between actual and predicted values, which penalizes larger errors more heavily (because there is a squared), so that researchers can understand the model's sensitivity to extreme deviations. MAE, on the other hand, represents the mean of the absolute differences, providing a straightforward measure of prediction error while avoiding weighted prioritization of large error magnitudes.

Lastly, R^2 is used to measure the goodness of fittings between the actual and predicted values. It quantifies the proportion of variance in the stock index data that is captured by the model. A higher R^2 value suggests that the model's predictions are closely aligned with the real stock index movements, which in turn, indicating the model's ability to accurately learn and reproduce complex temporal patterns from the data (Khalil & Bakar, 2023, Shahi et al., 2020).

Ergo, the model with the lowest RMSE and MAE values alongside with highest R^2 value is considered the most accurate one, demonstrating superior predictive performance compared to models with larger error (Khalil & Bakar, 2023, Shahi et al., 2020).

3.6 Summary of Research Methodology

In summary, this chapter introduced that the study will be employing a quantitative approach to predict stock index movements using LSTM and GRU models with Python programming language in the Google Colab platform. Secondary data was collected via from Yahoo Finance API, covering a 15-year period of daily stock index data from 2010 to 2024, which included both bull and bear markets across global. The study selected ten stock market indexes representing developed and emerging markets (classified by MSCI) across America, EMEA, and APAC regions, and they are distributed equally. The data processing steps included data normalization through MinMaxScaler along with data partition that training data taking up 80% and testing data taking up 20% of the total dataset. Stationarity test is conducted to understand the time series patterns. Moving window size is selected as 250 days, with a 20-day prediction length. MSE is utilized as loss function, Adam serves as the optimizer, and a maximum of 1000 epochs is set. An early stopping callback mechanism is also introduced to prevent overfitting during the training process. Lastly, this chapter ends with choosing three standard statistical metrics, namely RMSE, MAE, and R^2 as the evaluation metrics.

CHAPTER 4: DATA PRESENTATION AND ANALYSIS

4.1 Introduction

First of all, this section is going to visualize the collected dataset and analyze their patterns through descriptive methods. Section 4.2 will first introduce the dataset by presenting a snapshot on its format, the number of observations, movement of all indexes, as well as some critical descriptive analyses relating to these movements. Following that, Section 4.3 is going to conduct diagnostics testing on the time series, to assess whether the time series is stationary, with Section 4.4 interpreting and analyzing the descriptive statistics results, including measures such as the mean, standard deviation, coefficient of variation, median, minimum, maximum, skewness, and kurtosis. This is to provide a clearer picture of the dataset characteristics. Lastly, Section 4.5 will explain in detail how the deep learning models are constructed through Python code, module by module, to ensure the model can be comprehensively understood and replicated by future researchers who are interested to validate or expand further from this research.

4.2 Data Description

Table (6) and (7) below present a snapshot of the input data format for the LSTM and GRU models to make prediction. These tables display the first five and the last five trading day figures for each index to give an overview of the dataset structure. Each row represents the closing price of the respective index on a particular date. This format was adopted to ensure consistency across different markets.

Additionally, for better readability throughout this and the following chapters, all selected indexes will be referred to by their common short abbreviation in paragraphs, and their corresponding ticker symbols on Yahoo Finance in tables and diagrams, as shown in Table (5).

Table 5: List of Stock Indexes with Corresponding Abbreviation

Index Name	Abbreviation	Ticker on Yahoo Finance
S&P 500 Index	SPX	^SPX
FTSE 100 Index	FTSE	^FTSE
DAX	DAX	^GDAXI
Hang Seng Index	HSI	^HSI
ASX 100 Index	ASX	^AXJO

IBOVESPA	BVSP	^BVSP
IPC Mexico	MXX	^MXX
Tawadul All Shares Index	TASI	^TASI.SR
Kuala Lumpur Composite Index	KLCI	^KLSE
SSE Composite Index	SSE	000001.SS

Table 6: Snapshot of Input Data for Developed Markets

Date	**Developed Markets**				
	^SPX	**^FTSE**	**^GDAXI**	**^HSI**	**^AXJO**
4/1/2010	1132.990	5500.300	6048.300	21823.279	4876.300
5/1/2010	1136.520	5522.500	6031.860	22279.580	4924.300
6/1/2010	1137.140	5530.000	6034.330	22416.670	4921.400
7/1/2010	1141.690	5526.700	6019.360	22269.449	4899.400
8/1/2010	1144.980	5534.200	6037.610	22296.750	4912.100
...	...	...	...	...	...
25/12/2024	NaN	NaN	NaN	NaN	NaN
26/12/2024	6037.590	NaN	NaN	NaN	NaN
27/12/2024	5970.840	8149.800	19984.320	20090.461	8261.800
29/12/2024	NaN	NaN	NaN	NaN	NaN
30/12/2024	5906.940	8121.000	19909.141	20041.420	8235.000
Observations	**3773**	**3785**	**3806**	**3691**	**3783**

Table 7: Snapshot of Input Data for Emerging Markets

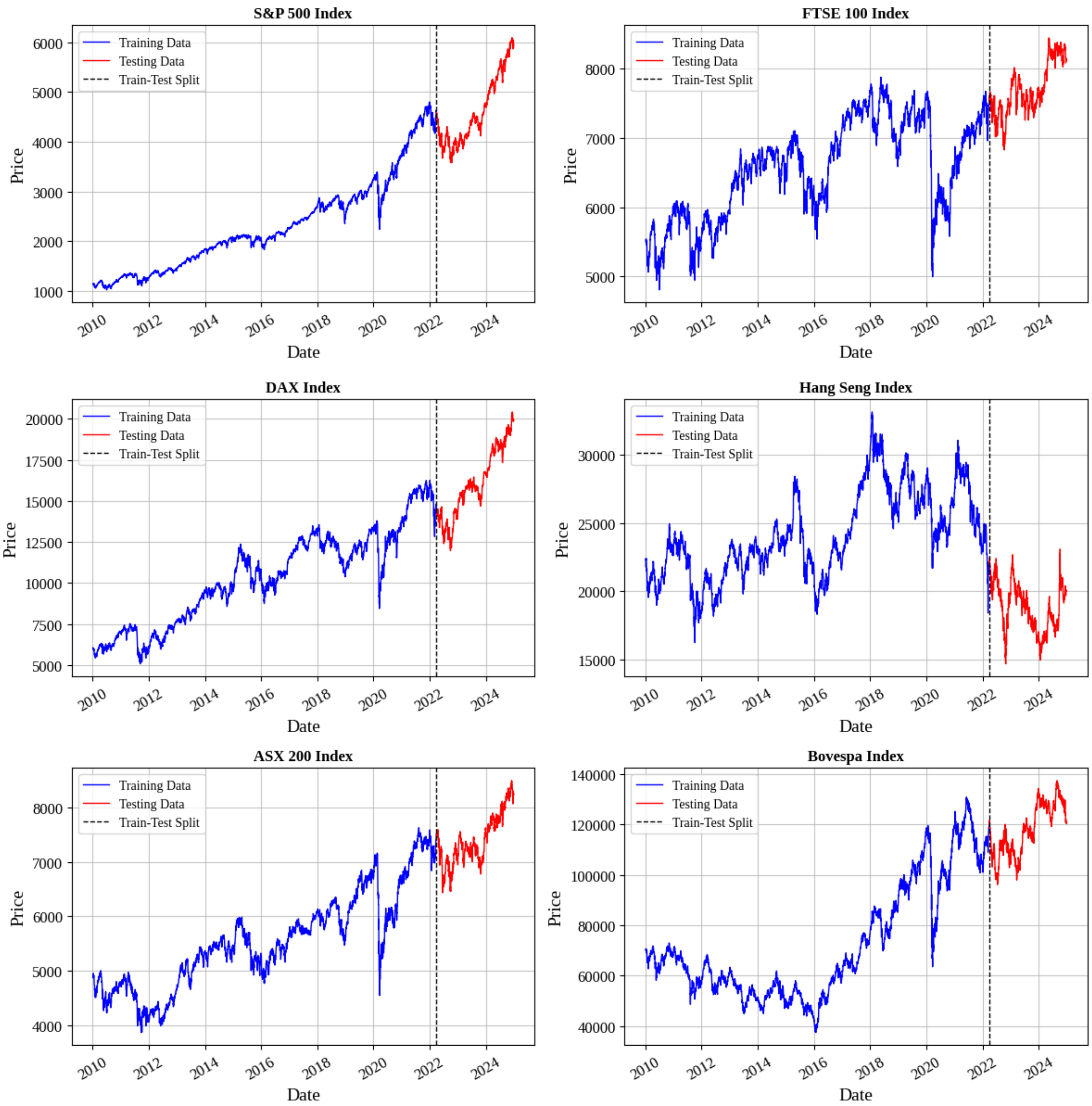
Date	**Emerging Markets**				
	^BVSP	**^MXX**	**^TASI.SR**	**^KLSE**	**000001.SS**
4/1/2010	70045.000	32758.529	6201.760	1275.750	3243.760
5/1/2010	70240.000	32732.760	6239.100	1288.240	3282.179
6/1/2010	70729.000	32830.160	6260.900	1293.170	3254.215
7/1/2010	70451.000	33064.570	6260.900	1291.420	3192.776
8/1/2010	70263.000	32892.039	NaN	1292.980	3195.997
...	...	...	...	...	...
25/12/2024	NaN	NaN	11892.320	NaN	3393.350
26/12/2024	121078.000	49535.578	11859.470	1613.700	3398.077
27/12/2024	120269.000	49290.578	NaN	1628.140	3400.142
29/12/2024	NaN	NaN	11892.750	NaN	NaN
30/12/2024	120283.000	48837.719	12000.920	1637.680	3407.326
Observations	**3715**	**3760**	**3280**	**3673**	**3638**

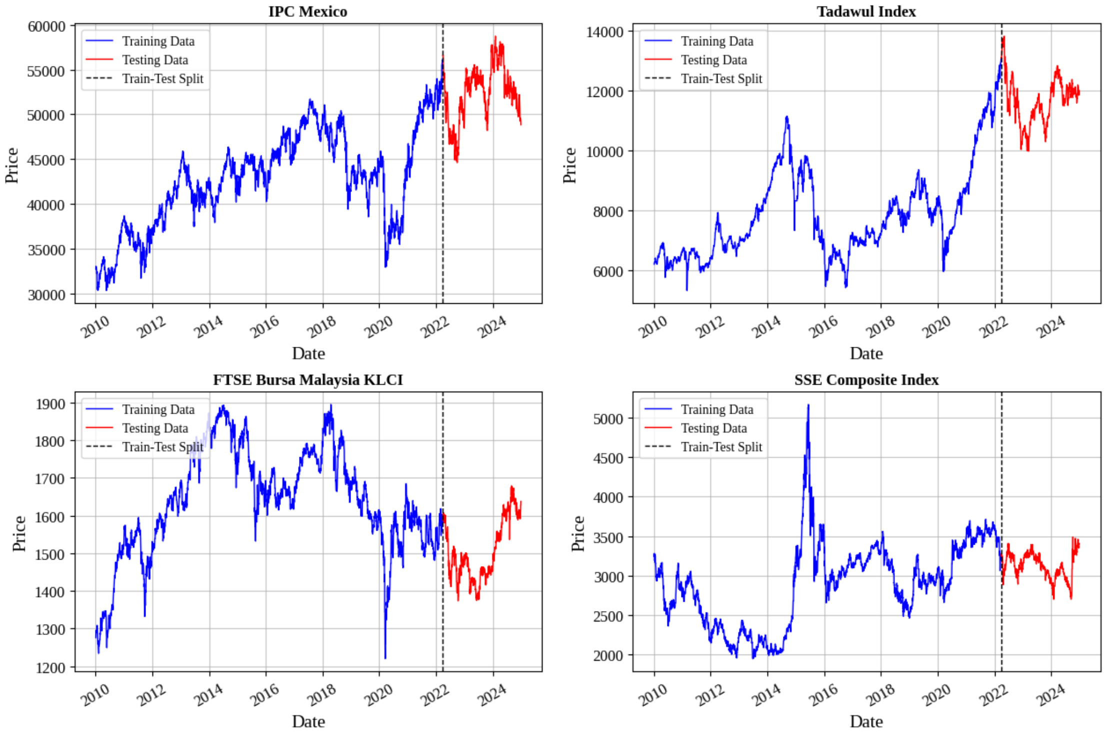
The number of observations for each index ranges from 3,280 (TASI) to 3,806 (DAX). These sample sizes reflect daily trading data across the full study horizon. The presence of “NaN” values indicates non-trading days for specific indexes, which may still be trading days for others. These gaps typically occur due to weekends or national public holidays. Such NaN values are excluded and will not be used during the training process later.

However, it is worth mentioning that TASI appears to be a significant outlier, with only 3,280 counts. In comparison, the second-lowest count is from SSE with 3,638 entries, while the remaining indexes range approximately between 3,600 and 3,800.

Next, as mentioned in the methodology section, the dataset is divided into two segments: 70% for training (represented in blue colour) and 30% for testing (represented in red colour). A snapshot of this division and the overall stock index trends are shown below in Figure (4), visualized using Python's Matplotlib library:

Figure 4: Snapshots of Training and Testing Dataset





From the snapshots above, it can be seen that most of the indexes exhibit an overall uptrend upon the selected 15-year period. However, the HSI, KLCI and SSE do not follow this pattern.

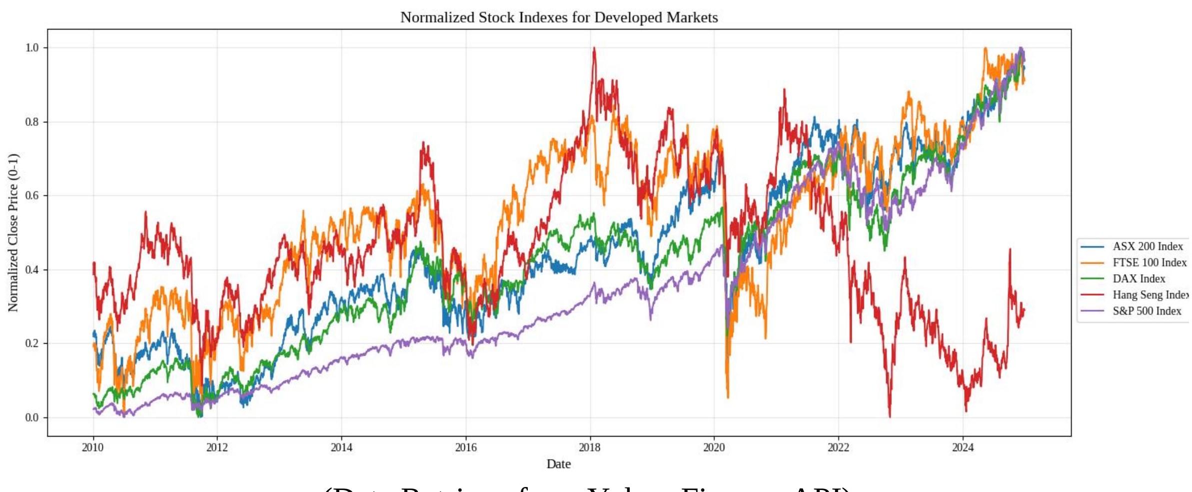
The KLCI, despite being in a non-uptrend category, closed at 1,637.68 by the end of 2024, marking a 28.37% increase from its level 15 years earlier (1,275.75). However, it experienced a notable decline from its historical peaks around 1,895 reached in 2014 and 2018. Besides, HSI stands out as the only index with a negative return over the period, falling by 8.16% from 21,823.28 in 2010 to 20,041.42 in 2024. This decline may be linked to the enactment of the Hong Kong National Security Law in mid-2020, which appears to coincide with a significant downward pivot. Yet, this research is not going to investigate such geopolitical factors in detail. Meanwhile, the SSE underwent a sharp rise in 2014 and reached its peak at 5,166.35 in 2015, and subsequently entered a sideways trend, fluctuating between 2,500 and 3,500 from 2016 onward, representing the second weakest performance over the 15-year horizon.

Interestingly, though the underperforming indexes are a mix of developed and emerging markets, all three of them are under the Asia-Pacific region.

Among all indexes, only the SPX and the DAX posted triple-digit returns, which is 421.36% and 229.17%, respectively. The remaining six indexes recorded returns in the range of 0% and 99%, naming sequentially from low to high: KLCI (28.37%), FTSE (47.65%), MXX (49.08%), ASX (68.88%), BVSP (71.72%), and TASI (93.51%). As such, SPX is known as the top performer, while HSI was the weakest. Notably, both of them are from developed markets. Averagely, developed markets achieved a cumulative return of 151.78%, whereas emerging markets only 49.55%, approximately one-third of the developed counterparts.

Another interesting finding found is that the extent of drawdowns from their historical peaks. Except for HSI, which declined 39.55% from its peak, all other developed market indexes only saw minor drawdowns of around 3%: SPX (3.01%), FTSE (3.85%), DAX (2.53%), and ASX (3.06%). Contrariwise, most emerging markets experienced steeper declines from their peaks, with BVSP falling 12.42%, MXX 16.82%, TASI 13.16%, and KLCI 13.59%, averaged 14%, but SSE was the exception in this group, with the highest drawdown of 34.05%. Figure (5) presented below, which is an aggregated chart illustrating the normalized movement of developed market stock indexes (to be further introduced in subsection 4.4.4), shows the obvious distinction between the HSI and other indexes.

Figure 5: Normalized Stock Index Movements for Developed Markets



(Data Retrieve from Yahoo Finance API)

Table (8) below presents detailed statistics that complement the preceding analysis. The Cumulative Index Returns represents the aggregated return of each index over the 15-year period, calculated as the difference between the final and initial close prices, divided by the initial value. Meanwhile, the Peak Corrections indicates the extent of correction from each index's peak, derived by the difference between the final and maximum close prices, divided by the maximum value.

Table 8: Detailed Statistics of Index Returns and Peak Corrections

Ticker	Initial Close Price	Final Close Price	Maximum Close Price	Cumulative Index Returns (%)	Peak Corrections (%)
^SPX	1132.990	5906.940	6090.270	421.36	-3.01
^FTSE	5500.300	8121.000	8445.800	47.65	-3.85
^GDAXI	6048.300	19909.141	20426.270	229.17	-2.53
^HSI	21823.279	20041.420	33154.121	-8.16	-39.55
^AXJO	4876.300	8235.000	8495.200	68.88	-3.06
^BVSP	70045.000	120283.000	137344.000	71.72	-12.42
^MXX	32758.529	48837.719	58711.871	49.08	-16.82
^TASI.SR	6201.760	12000.920	13820.350	93.51	-13.16
^KLSE	1275.750	1637.680	1895.180	28.37	-13.59
000001.SS	3243.760	3407.326	5166.350	5.04	-34.05

4.3 Diagnostic Testing

To complement an additional layer of analysis from above, the Augmented Dickey-Fuller (ADF) test is conducted to test for the stationarity of selected stock indexes, the results showed below:

Table 9: Augmented Dickey-Fuller Test Results

Ticker	ADF Statistics	p-value	Critical Value (5%)	Stationarity
^SPX	1.4265	0.9972	-2.8623	FALSE
^FTSE	-2.3967	0.1427	-2.8623	FALSE
^GDAXI	-0.3925	0.9114	-2.8623	FALSE
^HSI	-2.3975	0.1425	-2.8623	FALSE
^AXJO	-0.8848	0.7929	-2.8623	FALSE
^BVSP	-1.1405	0.6987	-2.8623	FALSE
^MXX	-2.3783	0.1480	-2.8623	FALSE
^TASI.SR	-1.2223	0.6639	-2.8624	FALSE
^KLSE	-2.9005	*0.0453	-2.8623	TRUE
000001.SS	-2.5287	0.1087	-2.8623	FALSE

Note: * indicates p-value < 0.05 (5% significance level)

The general rule for determining whether a time series is stationary using the ADF test is: if the ADF statistic is less than the critical value along with the p-value is less than the significance level 0.05, we reject the null hypothesis of a unit root and conclude that the time series is stationary, and vice versa (Raudys & Goldstein, 2022, Alamu & Siam, 2024). Therefore, based on the ADF test results presented above, the majority of the stock indexes were examined (except for KLCI) appear to be statistically significant of non-stationary. This suggests that these time series have a tendency to drift and their statistical properties change over time (Raudys & Goldstein, 2022).

However, given that LSTM and GRU were essentially designed to handle non-stationary and non-linear time series, explicit preprocessing like differencing or detrending is not required before the full-scale run. We are conducting this test is because this result may provide insightful information for the subsequent discussion in Chapter 5.

4.4 Descriptive Statistics Analysis

The descriptive statistics for the selected market indexes are summarized in Table (10) above.

Table 10: Descriptive Statistics of Dataset

Ticker	Mean	STDEV	CV	Median	Min	Max	Skewness	Kurtosis
^SPX	2727.44	1264.12	0.4635	2431.77	1022.58	6090.27	0.66	-0.55
^FTSE	6763.96	792.42	0.1172	6834.30	4805.80	8445.80	-0.21	-0.81
^GDAXI	11394.42	3507.35	0.3078	11584.63	5072.33	20426.27	0.20	-0.66
^HSI	23166.06	3594.41	0.1552	22956.57	14687.02	33154.12	0.23	-0.54
^AXJO	5894.03	1086.06	0.1843	5748.40	3863.90	8495.20	0.26	-0.91
^BVSP	81108.80	27391.14	0.3377	70423.00	37497.00	137344.00	0.39	-1.31
^MXX	44557.34	6241.71	0.1401	44558.19	30368.08	58711.87	-0.11	-0.67
^TASI.SR	8699.44	2070.48	0.2380	8110.08	5323.27	13820.35	0.53	-1.03
^KLSE	1614.67	141.87	0.0879	1611.49	1219.72	1895.18	-0.14	-0.50
000001.SS	2941.38	496.82	0.1689	3009.11	1950.01	5166.35	0.14	0.82

Across the indexes, both the mean values and medians vary significantly, reflecting differences in market scales and base units. For instance, BVSP records the highest mean value at 81,108.80, whereas KLCI shows the lowest at 1,614.67. Accordingly, due to its large base unit, BVSP also exhibits the highest standard deviation of 27,391.14. In contrast, KLCI, which operates within a smaller scale, has the lowest standard deviation of 141.87, aligning with its relatively narrower fluctuations.

As for median values, with no surprise, BVSP again ranks the highest (70,423.00), while KLCI remains the lowest (1,611.49), consistent with their respective base units. However, it is worth noting that only BVSP and SPX show a considerable gap between the mean and median. BVSP's median is 13.17% lower than its mean, while SPX's median is 10.84% below its mean, both suggesting a positive skew in distribution. The other indexes fall within a range of -6.78% (TASI) to 2.30% (SSE), indicating relatively symmetric distributions.

The Coefficient of Variation (CV), which standardizes the standard deviation relative to the mean, offers a more direct comparison of relative volatility rather than the raw figures. This is shown that despite not having the highest raw figures, SPX records the highest CV at 0.4635, derived by dividing its standard deviation of 1,264.12 by its mean of 2,727.44. This suggests that SPX experiences a higher degree of relative dispersion, meaning its data points are more widely spread around the mean. Financially, this indicates a more volatile market. In contrast, KLCI shows the lowest CV with only 0.0879, implying greater stability relative to its average value. On average, the overall CV across all markets is approximately 22%, but more specifically, emerging markets averaging 19.45% and developed markets averaging a higher 24.56%, suggesting higher volatility.

The minimum and maximum figures highlight the range of index levels throughout the sample period. SPX demonstrates the broadest range, where its maximum value (6,090.27) is 5.96 times greater than its minimum (1,022.58), underscoring substantial variability in the American market. On the other hand, KLCI shows the narrowest range, with the maximum value (1895.18) only 1.55 times higher than the minimum (1219.72), reflecting its relatively stable performance.

Skewness values range from negative to positive across the indexes. A positive skew indicates a right-tailed distribution, while a negative skew suggests a left-tailed one. Most indexes appear approximately symmetric, as their skewness values fall within the common threshold of -0.5 to 0.5. Only SPX (0.66) and TASI (0.53) exceed this threshold, but even these are considered only moderately skewed. This indicates that while extreme outliers on one side of the distribution are present, they are not overly dominant (Pandey, 2024).

Regarding kurtosis, all indexes show values below 3, which classifies them as platykurtic, indicating distributions with flatter peaks and thinner tails, where extreme values are less frequent. In detailed, SSE is the only index with a positive kurtosis (0.82), suggesting it has slightly heavier tails compared to the others. Conversely, BVSP has the lowest kurtosis at -1.31, indicating the lightest tails and a more uniform distribution around the mean (Pandey, 2024).

4.5 Implementation and Illustration of Model Architectures

4.5.1 Importing Essential Libraries

```
import yfinance as yf
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import Sequential
from sklearn.metrics import mean_squared_error,
mean_absolute_error, r2_score
```

To begin with, the implementation relies on a set of Python libraries that facilitate various stages of the data-driven modelling process. Therefore, the script above imports several key Python libraries, each serving a specific purpose in stock price prediction. Specifically, `yfinance` is utilized to extract historical financial data directly from Yahoo Finance, which provides everyone with convenient access to time-series stock prices. *pandas* and `numpy`, two fundamental data science libraries, are employed for efficient data manipulation and numerical operations. In particular, *pandas* enables structured handling of time-indexed datasets, while *numpy* supports the transformation of raw arrays into formats compatible with deep learning models.

Moreover, to ensure that the model processes input features effectively, `MinMaxScaler from sklearn.preprocessing` is used to normalize the price data within a range of 0 to 1, as ML algorithms perform more optimally when inputs are standardized (Bhandari et al., 2022).

For model evaluation, several statistical metrics are imported from `sklearn.metrics`, including mean squared error (MSE), mean absolute error (MAE), and the R-squared coefficient (R^2), which collectively assess prediction accuracy and model fit. Furthermore, the construction and training of the LSTM model are supported by the `tensorflow.keras` framework, with essential components such as `Sequential, LSTM, Dense, Dropout`, and `EarlyStopping`. Tensorflow and Keras are Python platform and API that support the construction and training of machine learning as well as deep learning models (Chollet et al., 2021).

4.5.2 Defining Hyperparameter and Initializing Results Storage Structure

```
DS_SPLIT = 0.8
MOVING_WIN_SIZE = 250
PREDICTION_DAYS = 20
EPOCHS = 1000
metrics_table = []
```

Subsequently, the script specifies several key hyperparameters that govern the structure and learning process of the predictive model. The first hyperparameter is the dataset splitting ratio, denoted by `DS_SPLIT = 0.8`, which indicates that 80% of the data is allocated for training and the remaining 20% for testing.

Besides, another hyperparameter is the moving window size, defined as `MOVING_WIN_SIZE = 250`. This parameter dictates the number of past observations used to generate each training sample. It is called a moving window because as new input data comes in, it will remove the oldest data during the training process, like a moving window forward through time. According to Saeed & Yin (2025), the window size is a hyperparameter that noticeably impacts the performance of deep learning models. The prediction length is on the other hand defined as `PREDICTION_DAYS = 20`.

`EPOCHS = 1000` specifies the maximum number of times the entire training dataset will be passed through the model during training. However, the inclusion of an `EarlyStopping` callback at below helps prevent overfitting by halting the training process once the validation loss ceases to improve for a predetermined number of epochs, which is set to be 10 in this research (Afaq & Rao, 2020; Goodfellow et al., 2016).

Additionally, in preparation for systematic performance evaluation, the script initializes a results table using a simple list structure: `metrics_table = []`. This structure is intended to store the output of each model run, which is the RMSE, MAE, and R^2. As each experiment is completed, the corresponding performance values are appended to this list. Eventually, the collection of metrics is converted into a pandas data frame, offering a better readable and interpretable format for subsequent analysis and comparison.

4.5.3 Retrieving Historical Stock Price Data

```
tickers = ["^SPX", "^FTSE", "^GDAXI", "^HSI", "^AXJO", "^BVSP",
"^MXX", "^TASI.SR", "^KLSE", "000001.SS"]
df = yf.Ticker(ticker).history(start="2010-01-01", end="2024-12-
31")
df = df.filter(["Close"])
```

Following the setup of parameters and libraries, the script proceeds with data acquisition and preparation. Specifically, the script here utilizes the `yfinance` library to retrieve historical price data from January 1, 2010, to December 31, 2024, for the target stock indexes. The dataset is filtered to retain only the "Close" prices, as closing values are typically used in financial forecasting due to their stability and representativeness of the value for each trading day, as they integrate all market activities throughout the day and are less susceptible to intraday noise.

4.5.4 Normalizing Data

```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_prices = scaler.fit_transform(df.values)
```

Subsequently, the data undergoes normalization using `MinMaxScaler`, which transforms the raw price values into a scaled range between 0 and 1. This scaling process, which transforms the data to a standard range, minimizes and standardizes variations caused by different price ranges across indexes and enhances the model training stability.

4.5.5 Sequence Generation for the Models

```
for j in range(len(scaled_prices) - MOVING_WIN_SIZE):
    x = scaled_prices[j:j + MOVING_WIN_SIZE]
    y = scaled_prices[j + MOVING_WIN_SIZE]
    all_x.append(x)
    all_y.append(y)
```

The entire sequence of scaled prices is then restructured into overlapping sub-sequences, which have lengths matching the moving window size. Each sub-sequence is paired with the subsequent price value as the prediction target. The line `x = scaled_prices[j:j + MOVING_WIN_SIZE]` is that for each iteration, it creates an input sequence (x) of length `MOVING_WIN_SIZE`, which is 250 days. For example, when j=0, x contains days 0 to 249;

when j=1, x contains days 1 to 250, and so on. Each "x" represents a window of 250 consecutive price points.

Besides, the line `y = scaled_prices[j + MOVING_WIN_SIZE]` selects the single price value that comes immediately after the input window, which is the target value we want the model to predict. `all_x.append(x)` and `all_y.append(y)` add each created window (x) and its corresponding target (y) to lists. So that by the end of the loop, `all_x` contains all possible 250-day windows from the data, whereas `all_y` contains all corresponding "next day" values. This sliding window technique effectively reformulates the time series into a supervised learning format, suitable for model training.

4.5.6 Model Architecture

```
model = Sequential([
    LSTM(units=128, return_sequences=True, input_shape=
(train_x.shape[1], 1)),
    Dropout(0.2),
    LSTM(units=128, return_sequences=True),
    Dropout(0.2),
    LSTM(units=128),
    Dropout(0.2),
    Dense(units=8),
    Dense(units=1)
])
```

Following next, this part of the script defines a sequential neural network model consisting of stacked LSTM layers and dropout regularization. The model has three of these LSTM layers stacked on top of each other, each with 128 neuron units that learn different aspects of stock price behaviour. Between these layers are dropout layers that *randomly deactivating some neurons during training*. This prevents the model from memorizing the training data too perfectly and helps it generalize better to new data. The `return_sequences=True` argument is set for the first two LSTM layers to ensure that the full sequence output is passed on to the next layer, allowing the model to learn hierarchical temporal patterns. After processing through these layers, the information flows through two regular (`Dense`) layers that narrow down all this complex pattern recognition into a single number – the ultimately predicted stock price level. The architecture of GRU is exactly the same as the LSTM, just to replace the “LSTM” with “GRU” when running the code. Importantly, the “LSTM” in the liabrary block should also be replaced with “GRU”.

4.5.7 Model Compilation and Training

```
model.compile(optimizer="adam", loss="mean_squared_error")
callback = EarlyStopping(monitor="val_loss", patience=10,
restore_best_weights=True)
model.fit(train_x, train_y, validation_split=0.2, epochs=EPOCHS,
callbacks=[callback], verbose=0)
```

`model.compile()` sets up the learning process by specifying the `"adam"` as the optimizer choice, with `"mean_squared_error"` as the loss function choice. An `EarlyStopping` callback is implemented with a patience of 10 epochs, aimed to monitor validation loss to prevent overfitting. `restore_best_weights=True` indicates that the models' weights will be restored to the best values observed during training if the validation loss doesn't improve for 10 consecutive epochs. Finally, `model.fit()` trains the model using the `train_x` input data and `train_y` target data. It sets aside 20% of the training data for validation to monitor performance on unseen data during training.

4.5.8 Model Evaluation and Metrics Computation

```
rmse = np.sqrt(mean_squared_error(test_y_actual, preds))
mae = mean_absolute_error(test_y_actual, preds)
r_squared = r2_score(test_y_actual, preds)
```

Upon training, the trained model is used to generate predictions on the test set. These predicted values are inverse-transformed using the same scaler applied during normalization to return them to their original price scale. The model's predictive accuracy is assessed using three key statistical metrics as discussed in Methodology: RMSE, MAE, and R^2 score. In short, RMSE provides a penalized measure of large deviations, MAE reflects the average magnitude of prediction errors, and R^2 quantifies the proportion of variance in the target variable that is predictable from the input features.

These metrics are collected for each run and appended to the `metrics_table`. This enables consistent performance monitoring across different test iterations or configurations, supporting a comprehensive empirical evaluation.

4.5.9 Predicting the Next 20 Days

```
last_window = df[-MOVING_WIN_SIZE:].values
last_window_scaled = scaler.transform(last_window)
X_test = np.array([last_window_scaled])
predicted_prices = []

for j in range(PREDICTION_DAYS):
    pred_price = model.predict(X_test)
    predicted_prices.append(pred_price[0, 0])
    pred_price_reshaped = pred_price.reshape(1, 1, 1)
    X_test = np.concatenate((X_test, pred_price_reshaped), axis=1)
    X_test = X_test[:, 1:, :]

    predicted_prices = np.array(predicted_prices).reshape(-1, 1)
        predicted_prices =
scaler.inverse_transform(predicted_prices)
```

In addition to evaluating historical prediction accuracy, the last part of the script is where the model looks into future to predict the stock index levels for the next 20 days (as justified earlier in the Methodology section), which is approximately the following month. In simple terms, it begins by taking the most recent 250 data points of stock index closing prices (as determined by `MOVING_WIN_SIZE`) from the dataset and scaling these values to the 0 to 1 range used during the model's training. The trained model uses the current `X_test` to predict the next day's stock index close price (`pred_price`), appended to the `predicted_prices` list. The predicted price is then reshaped to be compatible for concatenation with the existing `X_test`. The newly predicted price is added to the end of `X_test`, effectively extending the window of data used for the subsequent prediction. To maintain a consistent window size, the oldest price in `X_test` is removed. Once the sequence of future close prices is generated, the predicted values are inverse-transformed to their original scale. The resulting `predicted_prices` array contains the model's forecast for the stock index levels for the next 20 days. Here marks the end of the code execution.

4.6 Summary of Data Preparation and Analysis

This chapter details all aspects related to the dataset used for prediction. The chapter structure consists of three parts, including (1) data description and visualization with analytical findings, (2) diagnostics testing and descriptive statistics analysis, and (3) explanation of model implementation. Among all indexes, TASI reported the least observation of 3,280 throughout the selected period, remarkably lower than all others that have around 3,600 to 3,800. Besides, the HSI and KLCI together with SSE exhibit different patterns alongside upward trends observed in most indexes, with HSI the solely index that has negative returns, falling at 8.16% during the period. The financial returns for developed markets exceeded those of emerging markets by 102.23% with their 151.78% average performance, as opposed to emerging markets' 49.55% average returns. The developed market indexes showed small movements from their peak points averaging 3% although emerging market indexes declined sharply with SSE experiencing the deepest drop reaching 34.05% according to drawdown analysis. Section 4.3 shows that KLCI appears to be the only time series that is stationary, while Section 4.4 reveals that developed markets generally exhibited higher volatility with the averaged CV to be 24.56%, compared to emerging markets' 19.45%, opposing to the traditional perception that emerging markets are rather more volatile. Lastly, this chapter offers step-by-step guidance on the implementation of both the LSTM and GRU model, from import essential libraries in Python to retrieving historical data, normalizing data, defining hyperparameters, sequence generating, model training and evaluating, and ultimately, predicting future values.

CHAPTER 5: RESULTS AND CONCLUSIONS

5.1 Introduction

This chapter marks the end of our Final Year Project. In this chapter, we first discuss the results we recorded through running the models and making predictions, and critically analyze them to discover meaningful findings, reverting our initial research questions stated in Chapter 1. In addition, we point out the implications of our study that how it can be used practically, as well as suggest directions for future research.

5.2 Discussions on Experimental Results

Table 11: Mean of LSTM and GRU Results

Ticker	LSTM			GRU		
	RMSE	MAE	R^2	RMSE	MAE	R^2
^SPX	75.658	60.395	0.987	61.321	48.420	*0.992
^FTSE	59.606	45.226	0.975	58.922	44.007	0.975
^GDAXI	179.723	141.590	*0.992	162.154	126.304	*0.994
^HSI	339.261	262.711	0.960	352.401	267.014	0.960
^AXJO	62.741	48.317	0.981	62.483	48.395	0.981
^BVSP	1587.096	1293.333	0.975	1489.813	1203.392	0.978
^MXX	519.505	401.773	0.965	519.659	402.581	0.971
^TASI.SR	103.378	79.681	0.976	93.827	72.211	0.981
^KLSE	9.529	6.951	0.986	9.432	6.884	0.986
000001.SS	35.461	24.862	0.950	34.080	23.958	0.954

Note: * indicate $R^2 \geq 0.990$.

Table 12: Standard Deviation of LSTM and GRU Results

Ticker	LSTM			GRU		
	RMSE	MAE	R^2	RMSE	MAE	R^2
^SPX	8.176	7.112	0.003	7.412	6.260	0.002
^FTSE	1.796	2.387	0.002	1.714	2.334	0.001
^GDAXI	25.084	23.041	0.002	7.637	8.036	0.001
^HSI	30.951	20.361	0.004	18.869	23.766	0.004
^AXJO	1.933	1.871	0.001	1.414	1.745	0.001
^BVSP	40.049	36.311	0.001	51.754	44.508	0.002
^MXX	9.777	8.699	0.017	19.504	17.815	0.002
^TASI.SR	9.035	7.490	0.004	1.827	1.445	0.001
^KLSE	0.199	0.185	0.001	0.140	0.173	0.000
000001.SS	2.086	1.664	0.006	0.871	0.984	0.002

Table 13: Coefficient of Variation of LSTM and GRU Results

Ticker	LSTM			GRU		
	RMSE	MAE	R^2	RMSE	MAE	R^2
^SPX	**0.108	0.003	**0.118	**0.121	0.002	**0.129
^FTSE	0.030	0.002	*0.053	0.029	0.002	*0.053
^GDAXI	**0.140	0.002	**0.163	0.047	0.001	*0.064
^HSI	*0.091	0.004	*0.078	*0.054	0.004	*0.089
^AXJO	0.031	0.001	0.039	0.023	0.001	0.036
^BVSP	0.025	0.001	0.028	0.035	0.002	0.037
^MXX	0.019	0.017	0.022	0.038	0.002	0.044
^TASI.SR	*0.087	0.004	*0.094	0.019	0.001	0.020
^KLSE	0.021	0.001	0.027	0.015	0.000	0.025
000001.SS	*0.059	0.006	*0.067	0.026	0.002	0.041

Note: * indicate CV ≥ 0.05, ** indicates CV ≥ 0.10

5.2.1 Performance Comparison Among Models

Upon initial inspection of the Mean Results table, both LSTM and GRU models demonstrate exceptionally high R^2 values across all indexes, consistently exceeding 0.95 and some (SPX, GRU; DAX, both models) even reaching 0.99. This would, at first glance, suggests that the predictive accuracy on the test datasets is capable of explaining a very high proportion of the variance in future stock index movements, because the interpretation of R^2 is that how many percent of the variance in future stock price level can be explained by the models. Comparing both models, we see the GRU model achieve a very negligible advantage in R^2 mean, stands at 0.977, while LSTM is 0.975, which is almost no difference.

To expand, when we juxtapose the error metrics (RMSE, MAE) for both LSTM and GRU models, it is evident that GRU consistently marginally edges out LSTM across nearly every index, with almost all (except HSI and MXX) RMSE and MAE values calculated by the GRU model lower than their corresponding values from LSTM. This bring out an observation that GRU tends to produce predictions that are closer to the actual values than LSTM on average. In other words, GRUs have higher accuracy compared to LSTM networks.

Now, turning to the Coefficient of Variation (CV) results, which is a measurement of the variability of performance across the 10 runs (Reilly & Brown, 2012). These numbers provide a sneak peek on the model's stability. From Table (13), since MAE's CVs are generally very low for both models, almost all below 1%, it's not meaningful to discuss them. However, RMSE, a metric that penalize large errors, and R^2 values show more variation that worth examine. We see that GRU's runs are more stable in terms of range of variability, as the CV of RMSE for GRU hovers between approximately 0.019 and 0.121, while LSTM's CV ranges from 0.019 up to 0.140. In average terms, GRU's averaged RMSE CV achieve 0.041, one third lower than LSTM's 0.061.

Across all ten indexes, the highest variability for LSTM occurs on the DAX (CV ≈ 0.140 or 14%) and for GRU on the SPX (CV ≈ 0.121, 12.1%), both larger than 10%, which is considered inconsistent in prediction results. Also, for several tickers, especially DAX, HSI, TASI, and SSE, GRU exhibits significantly lower (more than or approximate to half) CVs for RMSE and R^2 compared to LSTM. This in turn, suggests that GRU's performance is more consistent across different random initializations and data splits over the 10 runs, making it a more reliable model for achieving similar levels of accuracy and variance explanation repeatedly. LSTMs, showed higher variability in these cases, implying its performance is more sensitive to the stochastic aspects of the training process.

Taken together, these perspectives reveal that GRU not only produces lower average errors but does so more stably compared to LSTM. Reverting to our RQ1, we conclude that GRU can predict the stock index movements relatively accurate and consistent. This finding aligns with He & Zhang (2024), Dey et al. (2021), and Saud & Shakya (2020) who also compared LSTM with GRU, and find that GRU outperformed LSTM. However, that's not the end of our discussion. There are areas that we yet to explore – how good are the models at predicting developed and emerging markets? This is another critical part of the project, addressing the RQ2, and would be covered in the following subsection.

5.2.2 Performance Comparison Among Markets

One way to verify a models' predictive power is to make comparisons, because a number alone cannot tell many stories. However, with our experimental results, conducting a direct comparison between RMSE and MAE values across different stock indexes is somewhat meaningless, because these metrics are scale-dependent. An error of '100' points on the DAX is very different in relative terms compared to an error of '100' points on a smaller index like the KLCI. Therefore, normalizing these metrics is a necessary step for a meaningful comparative analysis across indexes with vastly different value ranges. To do so, we will take the RMSE and MAE of each indexes and divided them by their corresponding final close price. This step not only provide a sense of the prediction error at the end of the prediction period but also allow us to understand the average magnitude of the error relative to the scale of each index on an equal footing. The normalized results are shown below:

Table 14: Normalized LSTM and GRU Results with Final Close Price

Ticker	Final Close Price	LSTM		GRU	
		nRMSE	nMAE	nRMSE	nMAE
^SPX	5906.940	1.28%	1.02%	1.04%	0.82%
^FTSE	8121.000	0.73%	0.56%	0.73%	0.54%
^GDAXI	19909.141	0.90%	0.71%	0.81%	0.63%
^HSI	20041.420	1.69%	1.31%	1.76%	1.33%
^AXJO	8235.000	0.76%	0.59%	0.76%	0.59%
^BVSP	120283.000	1.32%	1.08%	1.24%	1.00%
^MXX	48837.719	1.06%	0.82%	1.06%	0.82%
^TASI.SR	12000.920	0.86%	0.66%	0.78%	0.60%
^KLSE	1637.680	0.58%	0.42%	0.58%	0.42%
000001.SS	3407.326	1.04%	0.73%	1.00%	0.70%

Across developed markets, nRMSE ranges from 0.73% (FTSE, both models) to 1.76% (HSI, GRU), and nMAE from 0.54% (FTSE, GRU) to 1.33% (HSI, GRU). Within developed markets, GRU generally shows slightly lower normalized errors than LSTM, except for HSI where GRU has slightly higher errors. However, the idiosyncrasy of the HSI is not strange, as we have already exposed its "peculiarity" in Chapter 4 under Section 4.2 that it is the only stock index with negative aggregated returns over a 15-year period and one of the two that isn't in an uptrend throughout the period. Therefore, it is not surprise that it yielded unusual results.

Meanwhile, across emerging markets, nRMSE ranges from 0.58% (KLSE, both models) to 1.32% (BVSP, LSTM), and nMAE from 0.42% (^KLSE, both models) to 1.08% (BVSP, LSTM). Within emerging markets, GRU also generally shows slightly lower normalized errors than LSTM, except for MXX and KLCI where performance is considered identical.

Comparing across market types based on normalized errors, there is no strong indication that one market type is consistently associated with higher or lower predictive accuracy than the other. In average terms, we find that for LSTM, developed markets exhibit an averaged nRMSE of about 1.074% and nMAE of about 0.838%, whereas emerging markets average roughly 0.973% and 0.743%, both of which only with a very marginal of around 0.10% difference that shall not be perceived as meaningful. Similarly, for GRU, developed markets' averaged nRMSE and nMAE stands at 1.019% and 0.783% respectively, whereas emerging markets are 0.932% and 0.710%. Thus, the question of to what extent does the predictive performance differ across developed and emerging markets is said to be ≈ 0.10% in accuracy and is negligible.

Moving forward, examining the context of CVs presented in Table (13), on average, there shows considerably higher (more than double) CVs for RMSE in developed markets (8% for LSTM and 5.5% for GRU) compared to emerging markets (4.2% for LSTM and 2.6% for GRU), indicating less stable performance for developed markets. This is an interesting yet meaningful finding, as in addition to effectiveness (less errors), model's reliability is another important consideration should be taken when making prediction. Because, if a model that its results often deviate from the mean, it is generally referred to as "random". Notably, both LSTM and GRU in emerging markets have better stability compared to developing markets.

Interestingly, another remarkable result comes from KLCI. We've unveiled in Section 4.3 that KLCI is the only stationary time series among all indexes, with a p-value fall below the 5% significance level. As suggested by Raudys & Goldstein (2022), stationary time series is theoretically assumed to outperform those non-stationary one, since it is easier for deep learning models or even linear models to learn stationary data. This statement is thus proven correct in our research as KLCI achieved the lowest nRMSE, nMAE, and also have very low CV values, showing the fact that it is of highest accuracy and one of the most stable ones. Hence, we conclude that KLCI is of the top performer among all.

Synthesizing these findings, while both LSTM and GRU achieve high average prediction performance (low normalized errors with high R^2) in both developed and emerging markets, GRU generally holds a minor edge in both accuracy and stability across both market types. However, there is only very weak evidence based on these metrics to conclude that one market classification is inherently more predictable than the other. Our findings denied the statement by Bekaert & Havery (1997) that emerging markets have higher predictability.

5.2.3 Deepening the Connection to Underpinning Theories

The ability of both LSTM and GRU models to achieve very high R^2 values alongside very low normalized errors (all below the preset threshold of 3% in Chapter 3) using only historical price data compels a discussion regarding the Weak Form of the Efficient Market Hypothesis and the Random Walk Theory. The results presented here, demonstrating that models trained purely on historical closing prices can explain over 95% of the variance in future prices and keep average errors below 2% of the index value, appear to contradict a strict interpretation of these theories. Such a high level of predictability suggests that historical price patterns are not fully random or immediately incorporated into prices, and there are rooms for prediction, at least for short-term. This aligns with the Behavioural Finance theories we discussed earlier during Section 2.2.3.

Regarding the comparison between developed and emerging markets in relation to EMH, in general, it is often posited that emerging markets might be less weak-form efficient than developed markets. In other words, develop markets should be more efficient than emerging markets and follow a random behaviour. If this were consistently true, we might expect to see higher R^2 values and/or lower normalized errors in emerging markets, indicating greater predictability from historical prices. Notwithstanding, the results here do not provide strong support for this differential efficiency. Both market types exhibit similarly high R^2 values and comparable ranges of normalized errors. This could imply that, at least based on prediction solely from historical closing prices using these models, the level of weak-form efficiency is similar across the studied developed and emerging markets.

5.3 Key Findings and Conclusion

Stock market prediction is always an area captivated strong interest among all market participants across the globe, as people find it a potential pathway to wealth accumulation. However, precise and consistent stock price prediction is always a difficult task because they were impacted by so many factors, including but not limited to fundamental factors like macroeconomic data and geopolitical events, and also technical signals as well as psychological factors. It is almost impossible for a single model to account for all these elements simultaneously. Still, this does not imply that prediction is entirely futile. Rather, it becomes a matter of understanding the degree to which accuracy can be reasonably achieved.

This comparative analysis reveals several key findings. LSTM and GRU models demonstrate high predictive accuracy in forecasting stock index movements, with R^2 values consistently above 0.95 and normalized errors below 2% for all. In response to RQ1, queries about "*Which model can predict the stock index movements more accurately and consistently?*", the answer is GRU marginally outperforms LSTM in both accuracy and consistency. They yield slightly lower RMSE and MAE values and exhibits more stable performance across multiple runs, as reflected in their lower CVs. This indicates that GRU's performance is less sensitive to the stochastic elements of the training process.

Addressing RQ2, "*To what extent does the predictive performance differ across developed and emerging markets?*", the comparative analysis between developed and emerging markets indicates only minimal differences in predictive accuracy. Although emerging markets exhibit slightly lower normalized errors, the differences are somewhat not meaningful, suggesting limited market heterogeneity in predictability. However, the CV analysis revealed a notable difference in stability: developed markets showed considerably higher CVs for RMSE compared to emerging markets, indicating less stable performance in developed markets.

Regarding RQ3, "*Whether the market is always efficient and leaving no room for prediction?*", definitely, the simple answer is no. The consistently high R^2 values alongside low normalized errors achieved by models trained solely on historical price data challenge the strict interpretation of the Weak Form EMH and the RWT. The ability of both models to explain over 95% of price variance using only historical data suggests that markets are not entirely efficient and that historical price patterns retain some degree of predictability. This finding aligns well with concepts from Behavioural Finance perspective. Notably, from experiments, the observed similarity in predictive performance between developed and emerging markets does not support the assertion that emerging markets are significantly less weak-form efficient than developed ones, as discussed in Section 2.3.

5.4 Research Implications

Practically speaking, our predictive results can serve as a systematic compass for near-term investment decisions on a quantitative edge. Let’s visit an illustrative S&P500 example: at a closing level of 6,000 points, the mean RMSE of 62.4 points or 1.04% (with a standard deviation of 7.55) implies that roughly 95% (within two standard deviations) of a single run forecasts will err by somewhere between the ballpark of 62.4 ± 2(7.55) ≈ 47.3 to 77.5. This is equivalent to about ± 0.79% to ± 1.29% of the index level, serving as the lower and upper bound. Translating this back to the S&P500 if our model predicts 6,100 for the next session, we can reasonably expect the true closing value to lie within roughly 6,023 to 6,177.

With this level of accuracy, portfolio managers can integrate these error bounds into a Tactical Asset Allocation (TAA), which is an approach of actively adjusts portfolio based on short-term market forecasts, overlay atop their Strategic Asset Allocation (SAA). Essentially, the objective of TAA is to systematically exploit inefficiencies and temporary imbalances in the market, according to Stockton & Shtekhman (2010) from Vanguard Research. Illustratively, if the forecast band suggests a modest upside (say, +1%), a manager might overweight equities by a proportion calibrated to that magnitude, while capping exposure if volatility spikes or the model indicates a downward bias. Embedding normalized RMSE thresholds into stop-loss or take-profit rules also helps ensure that positions are unwound before errors exceed the model's typical uncertainty range.

In a robo-advisor context, the model can be integrated into a robo-advisory system and derive algorithm-driven recommendations: automatically rebalancing client portfolios or signaling tactical tilts when predicted deviations exceed a user's risk tolerance. This automated approach minimizes emotional biases and achieves emotion-free by adhering to pre-defined forecast confidence intervals. Similarly, quantitative hedge funds can fold the model's output into systematic trading rules: for instance, initiating long positions when the model's upper bound surpasses a threshold, and short positions when the lower bound dips below it.

Nevertheless, the margin of error remains significant enough that forecasts should be used with caution, especially in volatile markets or when managing leveraged positions. Overall, individuals and practitioners should treat model outputs only as directional guides, not precise price targets and not certainties, because investment actions should account for the expected range of forecast uncertainty. There is always no guaranteed in return!

From a theoretical perspective, this study contributes to another piece of research that discovers the potential for predicting market movements using historical data, thereby discovering further evidence suggesting the dysfunctionality of the Efficient Market Hypothesis and Random Walk Theory, and the significance of Behavioural Finance Theory.

5.5 Recommendations for Future Work

There remains considerable room for future improvement and expansion beyond this study. While this research has demonstrated the relative superiority of the GRU model in predicting stock index movements, it is important to acknowledge several limitations. As discussed in earlier sections, this study primarily aimed to compare the predictive power of LSTM and GRU models across developed and emerging markets under identical conditions. It offers a perspective that questions the strict interpretation of the EMH, showing that short-term predictability may still exist.

However, the study did not explore how model performance might vary with different combinations of hyperparameters. Parameters such as the number of epochs, hidden layers, neuron counts, lookback periods (or "moving windows"), learning rates, batch sizes, loss functions, and weight initialization schemes were kept fixed to ensure comparability from a finance perspective. From a computer science perspective, optimizing these could potentially lead to much better results. Future research is encouraged to conduct hyperparameter tuning and sensitivity analyses to uncover optimal configurations.

Moreover, researchers can also consider expanding the scope of input features. Incorporating a richer dataset, including macroeconomic data, technical indicators, intraday high/low prices, and sentiment variables (e.g., social media trends, earnings reports, and political news), could potentially improve both accuracy and generalizability. Additionally, hybrid models that combine deep learning with traditional methods, or ensemble techniques that aggregate multiple models, may enhance predictive stability.

It is also worth noting that models like LSTM and GRU tend to produce slightly different results each time due to internal randomness. Future researchers may consider setting a fixed random seed to increase reproducibility and control for variability across different runs.

REFERENCES

Abu-Mostafa, Y. S., & Atiya, A. F. (1996). Introduction to financial forecasting. *Applied intelligence*, 6, 205-213.

Afaq, S., & Rao, S. (2020). Significance of epochs on training a neural network. *International Journal of Scientific and Technology Research*, *9*(06), 485-488.

Alamu, O. S., & Siam, M. K. (2024). Stock Price Prediction and Traditional Models: An Approach to Achieve Short-, Medium-and Long-Term Goals. *arXiv preprint arXiv:2410.07220*.

Allen, F., Brealey, R., Myers, S. (2011). *Principles of Corporate Finance*. New York: McGraw-Hill/ Irwin.

Althelaya, K. A., Mohammed, S. A., & El-Alfy, E. S. M. (2021). Combining deep learning and multiresolution analysis for stock market forecasting. *IEEE Access*, *9*, 13099-13111.

Aminimehr, A., Raoofi, A. Aminimehr, A. & Aminimehr, A. (2022). A Comprehensive Study of Market Prediction from Efficient Market Hypothesis up to Late Intelligent Market Prediction Approaches. Springer, *Computational Economics*, *60*(2), 781–815. https://doi.org/10.1007/s10614-022-10283-1

Amiri, S. B., Amidian, A., & Fasihfar, Z. (2025). Intelligent Stock Price Prediction Using LSTM, GRU, ARIMA, and ARIMAX Models: Analysis and Performance Comparison. *Accounting and Auditing with Applications*, *2*(2), 109-121.

Anam, M. K., Defit, S., Haviluddin, H., Efrizoni, L., & Firdaus, M. B. (2024). Early Stopping on CNN-LSTM Development to Improve Classification Performance. *Journal of Applied Data Sciences*, *5*(3), 1175-1188.

Andres, A. R., Otero, A., & Amavilah, V. H. (2021). Using deep learning neural networks to predict the knowledge economy index for developing and emerging economies. *Expert Systems with Applications*, *184*, 115514.

Ashwell, K. W. (2012). *The brain book: development, function, disorder, health*. Firefly Books.

Azam, M., Haseeb, M., Samsi, A.B. & Raji, J.O. 2016. Stock market development and economic growth: Evidences from Asia-4 countries. *International Journal of Economics and Financial Issues 6*(3): 1200-1208.

Bekaert, G., & Harvey, C. R. (1997). Emerging equity market volatility. *Journal of Financial economics*, *43*(1), 29-77.

Bezerra, P. C. S., and Albuquerque, P. H. M. (2017). Volatility forecasting via SVR–GARCH with mixture of Gaussian kernels. *Computational Management Science 14*, 179–196. DOI: 10.1007/s10287-016-0267-0

Bilevich, A. L. (2023). *A Study on Time Series Predictions using Multivariate Neural Networks and External Sources* (Doctoral dissertation, University of Applied Sciences Technikum Wien).

Bhandari, H. N., Rimal, B., Pokhrel, N. R., Rimal, R., Dahal, K. R., & Khatri, R. K. (2022). Predicting stock market index using LSTM. *Machine Learning with Applications*, *9*, 100320.

Bhowmik, R., & Wang, S. (2020). Stock market volatility and return analysis: A systematic literature review. *Entropy*, *22*(5), 522.

Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer google schola*, *2*, 1122-1128.

Bruni R. (2016). Stock Market Index Data and indicators for Day Trading as a Binary Classification problem. *Data in brief*, *10*, 569–575. https://doi.org/10.1016/j.dib.2016.12.044

Bustos, O., & Pomares-Quimbaya, A. (2020). Stock market movement forecast: A systematic review. *Expert Systems with Applications*, *156*, 113464.

Cao, J., Li, Z., & Li, J. (2019). Financial time series forecasting model based on CEEMDAN and LSTM. *Physica A: Statistical mechanics and its applications*, *519*, 127-139.

Chatterjee, S., & Simonoff, J. S. (2013). *Handbook of regression analysis*. John Wiley & Sons.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Fethi, B.; Holger, S. & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chollet, F. (2021). Deep learning with Python (Second Edition). *Shelter Island: Manning Publications*.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., & Bengio, Y. (2015). A recurrent latent variable model for sequential data. *Advances in Neural Information Processing Systems*, *28*.

Clevert, D. A. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Dare, J., Patrick, A. O., & Oyewola, D. O. (2022). Comparison of stationarity on Ljung box test statistics for forecasting. *Earthline Journal of Mathematical Sciences*, *8*(2), 325-336.

Das, S. R., Mokashi, K., & Culkin, R. (2018). Are markets truly efficient? Experiments using deep learning algorithms for market movement prediction. *Algorithms*, *11*(9), 138.

Degutis, A., & Novickytė, L. (2014). The efficient market hypothesis: A critical review of literature and methodology. *Ekonomika*, *93*(2), 7-23.

Dey, P., Hossain, E., Hossain, M. I., Chowdhury, M. A., Alam, M. S., Hossain, M. S., & Andersson, K. (2021). Comparative analysis of recurrent neural networks in stock price prediction for different frequency domains. *Algorithms*, *14*(8), 251.

Dias, R., Teixeira, N., Machova, V., Pardal, P., Horak, J., & Vochozka, M. (2020). Random walks and market efficiency tests: evidence on US, Chinese and European capital markets within the context of the global Covid-19 pandemic. *Oeconomia Copernicana*, *11*(4), 585-608.

Fama, E. F. (1970). Efficient capital markets. *Journal of finance, 25*(2), 383-417.

FasterCapital. (2025). *Autocorrelation: Autocorrelation and Multicollinearity: Untangling the Ties in Time Series Analysis*. https://fastercapital.com/content/Autocorrelation--Autocorrelation-and-Multicollinearity--Untangling-the-Ties-in-Time-Series-Analysis.html

Fuller, E., Yerramalla, S., Cukic, B., & Gururajan, S. (2005, July). An approach to predicting non-deterministic neural network behavior. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* (Vol. 5, pp. 2921-2926). IEEE.

Gandhmal, D. P., & Kumar, K. (2019). Systematic analysis and review of stock market prediction techniques. *Computer Science Review, 34*, 100190.

Gao, P., Zhang, R. & Yang, X. (2020). The Application of Stock Index Price Prediction with Neural Network. *Mathematical and Computational Applications, 25*(3), 53. DOI:10.3390/mca25030053.

Gao, T., and Chai, Y. (2018). Improving Stock Closing price Prediction Using Recurrent Neural Network and Technical Indicators. *Neural Computation, 30*(10), 2833–2854. DOI:10.1162/neco_a_01124.

Gershonson, C. (2003). Artificial neural networks for beginners. *arXiv preprint cs/0308031*.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1, No. 2). Cambridge: MIT press.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222 2232.

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert systems with Applications, 38*(8), 10389-10397.

Hadhood, H. (2022). *Stock trend prediction using deep learning models LSTM and GRU with non-linear regression* (Master's thesis, University of Eastern Finland).

Harvey, C. R. (1995). Predictable risk and returns in emerging markets. *The review of financial studies*, *8*(3), 773-816.

Hassan, T. (2025). Augmented Analytics in Financial Forecasting Enhancing Risk Management and Investment Strategies through AI-Driven Insights. *Available at SSRN 5179431*.

Haque, M., Hassan, M. K., & Zaher, T. (2004). Stability, predictability and volatility of asian emerging stock markets. *Indian Journal of Economics and business, 3*, 121-146.

He, Z., & Zhang, H. (2024). A Comparative Study on Deep Learning Models for Stock Price Prediction. In *Image Processing, Electronics and Computers* (pp. 540-551). IOS Press.

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception* (pp. 65-93). Academic Press.

Helmenstein, C. & Haefke, C. (2015). Stock Market Indices. *Wiley Encyclopedia of Management, 1-2. Volume 4,* Finance. https://doi.org/10.1002/9781118785317.weom040072

Henrique, B. M., Sobreiro, V. A., and Kimura, H. (2019). Literature Review: Machine Learning Techniques Applied to Financial Market Prediction. *Expert Systems with Applications, 124*, 226–251. DOI: 10.1016/j.eswa.2019.01.012

Hinton, G. E. (1987). Learning translation invariant recognition in a massively parallel networks. In *International conference on parallel architectures and languages Europe* (pp. 1-13). Berlin, Heidelberg: Springer Berlin Heidelberg.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504-507.

Hoang, P. T., & Laskemoen, J. (2020). *Predicting stock prices with Long Short-Term Memory based models using a combination of data sources* (Master's thesis, NTNU).

Hochreiter, S. (1997). Long Short-term Memory. *Neural Computation MIT-Press*.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6*(02), 107-116.

Hochreiter, S., Heusel, M., & Obermayer, K. (2007). Fast model-based protein homology detection without alignment. *Bioinformatics, 23*(14), 1728–1736.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences, 79*(8), 2554-2558.

Hopfield, J. J., & Tank, D. (1986). Computing with neural circuits. *Science, 233*, 625–633.

Horne, V. J. C., & Parker, G. G. (1967). The random-walk theory: an empirical test. *Financial analysts journal, 23*(6), 87-92.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks, 2*(5), 359-366.

Hsu, M.-W., Lessmann, S., Sung, M.-C., Ma, T., and Johnson, J. E. V. (2016). Bridging the divide in Financial Market Forecasting: Machine Learners vs. Financial Economists. *Expert Systems with Applications, 61*, 215–234. DOI: 10.1016/j.eswa.2016. 05.033

Huang, C. J., Yang, D. X., & Chuang, Y. T. (2008). Application of wrapper approach and composite classifier to the stock trend prediction. *Expert Systems with Applications, 34*(4), 2870-2878. DOI: 10.1016/j.eswa.2007.05.035

Hush, D. R., & Horne, B. G. (1993). Progress in supervised neural networks. *IEEE signal processing magazine, 10*(1), 8-39.

Huynh, H. D., Dang, L. M., and Duong, D. (2017). "A New Model for Stock Price Movements Prediction Using Deep Neural Network," in *Proceedings of the Eighth International Symposium on Information and Communication Technology, 57–62*. DOI: 10.1145/3155133.3155202

Jain, S. (2019). *Analysing effect of Twitter, Oil Prices, Gold Prices and Foreign Exchange on S&P500 Using Machine Learning*. (Doctoral dissertation, Dublin, National College of Ireland).

Jiang, W. (2021). Applications of Deep Learning in Stock Market Prediction: Recent Progress. *Expert Systems with Applications, 184*, 115537. DOI: 10.1016/j.eswa.2021.115537

Johnson, R., & Zhang, T. (2013). Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems, 26*.

Kamal, I. M., Bae, H., Sunghyun, S., and Yun, H. (2020). DERN: Deep Ensemble Learning Model for Short- and Long-Term Prediction of Baltic Dry Index. *Applied Science 10*(4), 1504. DOI: 10.3390/app10041504

Karim, M. E., & Ahmed, S. (2021, October). A deep learning-based approach for stock price prediction using bidirectional gated recurrent unit and bidirectional long short term memory model. In *2021 2nd Global Conference for Advancement in Technology (GCAT)* (pp. 1-8). IEEE.

Kapoor, L. (2018). Relationship of Stock Market Indices and Individual stocks: A Descriptive Study. *Psychology and Education Vol. 55* No. 1 (2018): Volume 55 No. 1 (2018). ISSN: 1553-6939. DOI:10.48047/pne.2018.55.1.65.

Khairunisa, N. K., & Hendikawati, P. (2024). Long Short-Term Memory and Gated Recurrent Unit Modeling for Stock Price Forecasting. *Jurnal Matematika, Statistika dan Komputasi, 21*(1), 321-333..

Khalil, M. R. A. & Bakar, A. A. (2022). A Comparative Study of Deep Learning Algorithms in Univariate and Multivariate Forecasting of the Malaysian Stock Market. *Sains Malaysiana 52*(3) (2023): 993-1009. http://doi.org/10.17576/jsm-2023-5203-22.

Kim, J.-M., Kim, D. H., and Jung, H. (2021). Applications of Machine Learning for Corporate Bond Yield Spread Forecasting. *The North American Journal of Economics and Finance*, *58*, 101540. DOI: 10.1016/j.najef.2021.101540

Kratzert, F., Klotz, D., Herrnegger, M., Sampson, A. K., Hochreiter, S., & Nearing, G. S. (2019). Toward improved predictions in ungauged basins: Exploiting the power of machine learning. *Water Resources Research*, *55*(12), 11344–11354.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems, 25*.

Kumar, M., and Thenmozhi, M. (2014). Forecasting Stock index Returns Using ARIMA-SVM, ARIMA-ANN, and ARIMA-Random forest Hybrid Models. *International Journal of Banking, Accounting and Finance, 5*(3), 284–308. DOI:10.1504/IJBAAF.2014.064307

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of machine learning research*, *10*(1).

Le Roux, N., & Bengio, Y. (2010). Deep belief networks are compact universal approximators. *Neural computation*, *22*(8), 2192-2207.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436-444.

Liu, M. D., Ding, L., & Bai, Y. L. (2021). Application of hybrid model based on empirical mode decomposition, novel recurrent neural networks and the ARIMA to wind speed prediction. *Energy Conversion and Management*, *233*, 113917.

Lundqvist, L. (2019). A Deep Learning Model in a Tactical Asset Allocation Framework. DOI: 10.6084/m9.figshare.11671767.

Maiti, A. (2020). Indian stock market prediction using deep learning. *2020 IEEE REGION 10 CONFERENCE (TENCON)* (pp. 1215-1220). IEEE.

Manjunath, C., Marimuthu, B., & Ghosh, B. (2021). Deep Learning for Stock Market Index Price Movement Forecasting Using Improved Technical Analysis. *International Journal of Intelligent Engineering & Systems, 14(*5).

Masoud, N.M. (2013). The impact of stock market performance upon economic growth. *International Journal of Economics and Financial Issues 3*(4): 788-798.

Mateus, B. C., Mendes, M., Farinha, J. T., Assis, R., & Cardoso, A. M. (2021). Comparing LSTM and GRU models to predict the condition of a pulp paper press. *Energies*, *14*(21), 6958.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*, 115-133.

Minh, D. L., Sadeghi-Niaraki, A., Huy, H. D., Min, K., & Moon, H. (2018). Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. *IEEE Access*, 6, 55392-55404.

Mishkin, F. S. & Eakins, S. G. (2012). *Financial Markets and Institutions*. Pearson Education Inc. ISBN 10: 0-13-213683-X.

Mody, A. (2003). What is an emerging market. *International Monetary Fund Working Paper, 200*4.

Mohamed, A. R., Sainath, T. N., Dahl, G., Ramabhadran, B., Hinton, G. E., & Picheny, M. A. (2011, May). Deep belief networks using discriminative features for phone recognition. *2011 IEEE international conference on acoustics, speech and signal processing* (ICASSP) (pp. 5060-5063). IEEE.

Mokalled, W. E. H. M., & Jaber, M. (2019). Automated stock price prediction using machine learning. In *Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019)* (pp. 16-24).

Nabipour, M., Nayyeri, P., Jabani, H., Mosavi, A., & Salwana, E. (2020). Deep learning for stock market prediction. *Entropy*, *22*(8), 840.

Nikou, M., Mansourfar, G., & Bagherzadeh, J. (2019). Stock price prediction using DEEP learning algorithm and its comparison with machine learning algorithms. *Intelligent Systems in Accounting, Finance and Management*, *26*(4), 164-174.

Pandey, P. (2024). *Comprehensive Guide: Distribution, Skewness, and Kurtosis*. Medium.com. https://medium.com/@pujapandey73020/comprehensive-guide-distribution-skewness-and-kurtosis-f4c71dbd9da2

Palan, S. (2004). *The efficient market hypothesis and its validity in today's markets*. Master thesis. Graz: Faculty of Social and Economic Sciences, Karl-Franzens University, Graz.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert systems with applications*, *42*(1), 259-268.

Paul, J. (2024). Dynamic Asset Allocation Using Recurrent Neural Networks (RNNs).

Pradhan, R. P. (2018). Development of stock market and economic growth: The G-20 evidence. *Eurasian Economic Review*, *8*, 161-181.

Rahman, M. O., Hossain, M. S., Junaid, T. S., Forhad, M. S. A., & Hossen, M. K. (2019). Predicting prices of stock market using gated recurrent units (GRUs) neural networks. *International Journal of Computer Science and Network Security*, *19*(1), 213-222.

Raudys, A., & Goldstein, E. (2022). Forecasting detrended volatility risk and financial price series using lstm neural networks and xgboost regressor. *Journal of Risk and Financial Management*, *15*(12), 602.

Reilly, F. K. & Brown K. C. (2012). *Investment analysis & portfolio management*. Tenth Edition. South-Western, Cengage Learning.

Risso, W. A. (2009). The informational efficiency: The emerging markets versus the developed markets. *Applied Economics Letters*, *16*(5), 485-487.

Roy, S. (2018). Testing random walk and market efficiency: A cross-stock market analysis. *Foreign Trade Review*, *53*(4), 225-238.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533-536.

Ryu, G. Y. (2024). An Empirical Study on the Comparison of LSTM and GRU Forecasts using Stock Closing Prices. *International journal of advanced smart convergence*, *13*(4), 1-10.

Saboor, A., Hussain, A., Agbley, B. L. Y., Li, J. P., & Kumar, R. (2023). Stock Market Index Prediction Using Machine Learning and Deep Learning Techniques. *Intelligent Automation & Soft Computing*, *37*(2).

Saeed, B., & Yin, W. (2025). Optimizing Stock Price Prediction for South Asian Markets Using LSTM, GRU, CNN with Greedy Algorithm. https://doi.org/10.21203/rs.3.rs-58a47917/v1

Samarawickrama, A. J. P., & Fernando, T. G. I. (2017). A recurrent neural network approach in predicting daily stock prices an application to the Sri Lankan stock market. *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)* (pp. 1-6). IEEE.

Sanoyo, A. M., Hidayat, A. M., & Firmialy, S. D. (2024). IMPACT OF THE RUSSIA-UKRAINE CONFLICT ON INDONESIAN COMMODITY STOCKS: A SYSTEMATIC LITERATURE REVIEW. *Journal of Humanities, Social Sciences and Business, 3*(4), 1036-1049.

Saud, A. S., & Shakya, S. (2020). Analysis of look back period for stock price prediction with RNN variants: A case study on banking sector of NEPSE. *Procedia Computer Science*, *167*, 788-798.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)* (pp. 1643-1647). IEEE.

Shah, D., Isah, H., & Zulkernine, F. (2019). Stock Market Analysis: A Review and Taxonomy of Prediction Techniques. *International Journal of Financial Studies, 7*(2), 26. DOI:10. 3390/ijfs7020026

Shahi, T. B., Shrestha, A., Neupane, A., & Guo, W. (2020). Stock price forecasting with deep learning: A comparative study. *Mathematics, 8*(9), 1441.

Sharma, A. J. (2014). The behavioural finance: A challenge or replacement to efficient market concept. *The SIJ Transactions on Industrial, Finance & Business Management (IFBM)*, *2*(6), 1-5.

Sharma, A., & Thaker, K. (2015). Market efficiency in developed and emerging markets. *Afro-Asian Journal of Finance and Accounting*, *5*(4), 311-333.

Shen, G., Tan, Q., Zhang, H., Zeng, P., & Xu, J. (2018). Deep learning with gated recurrent unit networks for financial sequence predictions. *Procedia computer science, 131*, 895-903.

Shen, W., Guo, X., Wu, C., & Wu, D. (2011). Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. *Knowledge-Based Systems, 24*(3), 378-385. DOI: 10.1016/j.knosys.2010.11.001

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena, 404*, 132306.

Sifma Research. (2024). *2024 Capital Markets Fact Book*. Securities Industry and Financial Markets Association. https://www.sifma.org/wp-content/uploads/2023/07/2024-SIFMA-Capital-Markets-Factbook.pdf

Singh R. & Srivastava, S. (2016). Stock prediction using deep learning. *Multimedia Tools and Applications (2017)* 76:18569–18584. DOI 10.1007/s11042-016-4159-7.

Stockton, K. A., & Shtekhman, A. (2010). A primer on tactical asset allocation strategy evaluation.

Sulistio, B., Warnars, H. L. H. S., Gaol, F. L., & Soewito, B. (2023). Energy sector stock price prediction using the CNN, GRU & LSTM hybrid algorithm. In *2023 International Conference on Computer Science, Information Technology and Engineering (ICCoSITE)* (pp. 178-182). IEEE.

Sun, J., Xiao, K., Liu, C., Zhou, W., and Xiong, H. (2019). Exploiting Intra-day Patterns for Market Shock Prediction: A Machine Learning Approach. *Expert Systems with Applications, 127*, 272–281. DOI: 10.1016/j.eswa.2019.03.006

Sunny, M. A. I., Maswood, M. M. S., & Alharbi, A. G. (2020). Deep learning-based stock price prediction using LSTM and bi-directional LSTM model. In *2020 2nd novel intelligent and leading emerging sciences conference (NILES)* (pp. 87-92). IEEE.

Sutarna, N., Tjahyadi, C., Oktivasari, P., Dwiyaniti, M., & Tohazen, T. (2024). Hyperparameter tuning impact on deep learning bi-LSTM for photovoltaic power forecasting. *Journal of Robotics and Control (JRC)*, *5*(3), 677-693.

Sutskever, I., & Hinton, G. E. (2008). Deep, narrow sigmoid belief networks are universal approximators. *Neural computation*, *20*(11), 2629-2636.

Ta, V. D., Liu, C. M., & Tadesse, D. A. (2020). Portfolio optimization-based stock prediction using long-short term memory network in quantitative trading. *Applied Sciences*, *10*(2), 437.

Tashakkori, A., Erfanibehrouz, N., Mirshekari, S., Sodagartojgi, A., & Gupta, V. (2024). Enhancing stock market prediction accuracy with recurrent deep learning models: A case study on the CAC40 index. *World Journal of Advanced Research and Reviews, 23*(1), 2309-1. https://doi.org/10.30574/wjarr.2024.23.1.2156

Teixeira, L. A., & De Oliveira, A. L. I. (2010). A method for automatic stock trading combining technical analysis and nearest neighbour classification. *Expert systems with applications*, *37*(10), 6885-6890.

Touzani, Y. & Douzi, K. (2021). An LSTM and GRU based trading strategy adapted to the Moroccan market. *Journal of Big Data, 8*(1), 126. https://doi.org/10.1186/s40537-021-00512-z.

Tzafestas, S. G., Dalianis, P. J., & Anthopoulos, G. (1996). On the overtraining phenomenon of backpropagation neural networks. *Mathematics and computers in simulation*, *40*(5-6), 507-521.

Van Der Westhuizen, J., & Lasenby, J. (2018). The unreasonable effectiveness of the forget gate. *arXiv preprint arXiv:1804.04849*.

Vuković, D. B., Radenković, S. D., Simeunović, I., Zinovev, V., & Radovanović, M. (2024). Predictive patterns and market efficiency: A deep learning approach to financial time series forecasting. *Mathematics*, *12*(19), 3066.

Wang, J. J., Wang, J. Z., Zhang, Z. G., & Guo, S. P. (2012). Stock index forecasting based on a hybrid model. *Omega*, *40*(6), 758–766.

Wang, Y., Liu, Z., Wang, J., & Sun, Y. (2019). Comparative study of LSTM and GRU for stock price prediction on three major stock indices. *2019 IEEE International Conference on Big Data (Big Data)* (pp. 1763-1770). IEEE.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550-1560.

Wong, W. K., Yin, W. K., Kwong, A. W., HON Tai Yuen, K., & McAleer, M. (2022). Market efficiency, behavioural finance, and anomalies.

Yu, D., Deng, L., & Wang, S. (2009). Learning in the deep-structured conditional random fields. In *Proc. NIPS Workshop* (pp. 1-8).

Zaremba, W. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zatlavi, L., Kenett, D. Y. & Ben-Jacob, E. (2014). The design and performance of the adaptive stock market index. *Algorithmic Finance 3*(2014) 189–207 DOI 10.3233/AF-140039.

Zhang, N., Lin, A., and Shang, P. (2017). Multidimensionalk-nearest Neighbor Model Based on EEMD for Financial Time Series Forecasting. *Physica A: Statistical Mechanics and its Applications, 477*(1), 161–173. DOI: 10.1016/j.physa.2017.02.072

Zhang, Y., Xiong, R., He, H., and Pecht, M. G. (2018). Long Short-Term Memory Recurrent Neural Network for Remaining Useful Life Prediction of Lithium Ion Batteries. *IEEE Transactions on Vehicular Technology, 67*(7), 5695–5705. DOI: 10.1109/TVT. 2018.2805189

Zhu, G., Zhang, Z., Zhang, X. Y., & Liu, C. L. (2017). Diverse Neuron Type Selection for Convolutional Neural Networks. In *IJCAI* (pp. 3560-3566).

Zulqarnain, M., Ghazali, R., Ghouse, M. G., Hassim, Y. M. M., & Javid, I. (2020). Predicting financial prices of stock market using recurrent convolutional neural networks. *International Journal of Intelligent Systems and Applications, 13*(6), 21.

Zuo, Z., & Wang, G. (2014). Learning discriminative hierarchical features for object recognition. *IEEE Signal Processing Letters, 21*(9), 1159-1163.

Zurada, J. (1992). *Introduction to artificial neural systems*. West Publishing Co.

APPENDIX

[1]: Complete Table of Evaluation Metrics

Table 15: Complete Table of Evaluation Metrics

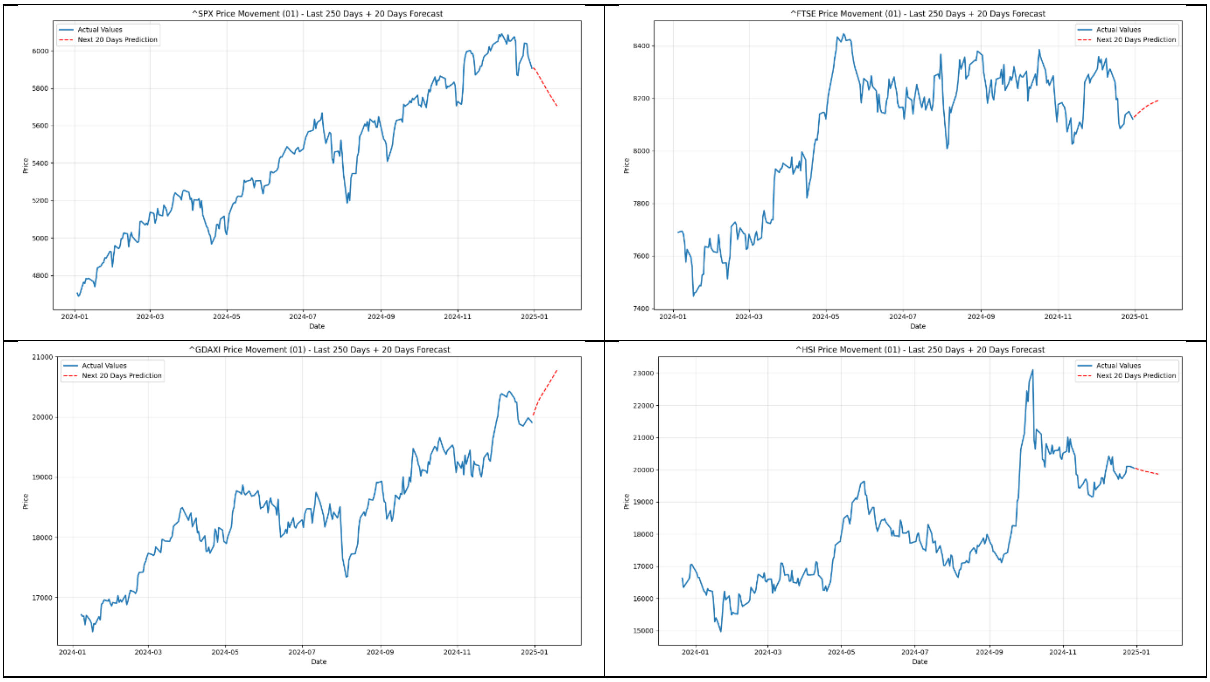
Ticker	Run	RMSE	MAE	R^2	Ticker	Run	RMSE	MAE	R^2
	01	67.628	52.175	0.990		01	1637.584	1343.912	0.974
	02	92.817	73.849	0.981		02	1623.523	1339.962	0.974
	03	75.719	58.894	0.988		03	1569.177	1272.006	0.976
	04	75.232	59.221	0.988		04	1529.812	1239.185	0.977
	05	70.468	53.596	0.989		05	1551.278	1265.246	0.976
	06	75.534	62.136	0.988		06	1633.457	1324.087	0.974
SPX	07	80.174	65.701	0.986	BVSP	07	1564.521	1273.600	0.976
	08	71.201	57.849	0.989		08	1542.896	1259.582	0.977
	09	64.705	52.548	0.991		09	1615.273	1313.386	0.974
	10	83.101	67.979	0.985		10	1603.438	1302.368	0.975
	AVERAGE	**75.658**	**60.395**	**0.987**		**AVERAGE**	**1587.096**	**1293.333**	**0.975**
	STDEV	**8.176**	**7.112**	**0.003**		**STDEV**	**40.049**	**36.311**	**0.001**
	CV	**0.108**	**0.118**	**0.003**		**CV**	**0.025**	**0.028**	**0.001**
	01	62.100	48.347	0.973		01	526.749	406.003	0.970
	02	59.064	44.660	0.975		02	513.442	396.160	0.972
	03	58.569	43.737	0.976		03	510.465	395.455	0.972
	04	62.611	49.219	0.972		04	522.432	405.359	0.971
	05	57.082	41.867	0.977		05	520.343	399.643	0.971
	06	61.062	47.296	0.974		06	511.714	396.304	0.972
FTSE	07	58.337	43.625	0.976	MXX	07	541.366	422.694	0.968
	08	59.448	45.243	0.975		08	513.358	395.439	0.972
	09	58.094	43.004	0.976		09	510.647	394.745	0.918
	10	59.690	45.258	0.975		10	524.536	405.931	0.970
	AVERAGE	**59.606**	**45.226**	**0.975**		**AVERAGE**	**519.505**	**401.773**	**0.965**
	STDEV	**1.796**	**2.387**	**0.002**		**STDEV**	**9.777**	**8.699**	**0.017**
	CV	**0.030**	**0.053**	**0.002**		**CV**	**0.019**	**0.022**	**0.017**
	01	198.088	161.607	0.991		01	107.320	84.946	0.975
	02	164.360	129.092	0.994		02	95.271	73.162	0.980
	03	226.077	182.363	0.988		03	99.314	75.595	0.978
	04	151.775	115.682	0.995		04	111.504	85.091	0.972
	05	180.131	140.241	0.992		05	102.108	78.829	0.977
	06	201.638	163.219	0.991		06	98.913	76.370	0.978
GDAXI	07	198.480	157.414	0.991	TASI.SR	07	102.456	78.343	0.977
	08	161.024	125.917	0.994		08	98.483	75.116	0.979
	09	160.950	121.806	0.994		09	94.092	72.339	0.980
	10	154.710	118.556	0.994		10	124.314	97.017	0.966
	AVERAGE	**179.723**	**141.590**	**0.992**		**AVERAGE**	**103.378**	**79.681**	**0.976**
	STDEV	**25.084**	**23.041**	**0.002**		**STDEV**	**9.035**	**7.490**	**0.004**
	CV	**0.140**	**0.163**	**0.002**		**CV**	**0.087**	**0.094**	**0.004**
	01	387.957	308.949	0.951		01	9.601	7.094	0.986
	02	342.074	258.230	0.962		02	10.016	7.382	0.985
	03	265.900	284.263	0.957		03	9.373	6.773	0.986
	04	338.675	249.463	0.963		04	9.392	6.820	0.986
HSI	05	348.856	263.143	0.961	KLSE	05	9.637	7.027	0.986
	06	330.967	244.083	0.965		06	9.534	6.939	0.986
	07	332.053	244.562	0.964		07	9.397	6.799	0.986
	08	363.990	268.740	0.957		08	9.356	6.828	0.987
	09	344.920	256.376	0.961		09	9.548	6.981	0.986

	10	337.221	249.299	0.963		10	9.432	6.861	0.986
	AVERAGE	**339.261**	**262.711**	**0.960**		**AVERAGE**	**9.529**	**6.951**	**0.986**
	STDEV	**30.951**	**20.361**	**0.004**		**STDEV**	**0.199**	**0.185**	**0.001**
	CV	**0.091**	**0.078**	**0.004**		**CV**	**0.021**	**0.027**	
AXJO	01	61.822	46.494	0.982	000001.SS	01	34.161	23.577	0.954
	02	67.357	52.289	0.978		02	35.284	24.504	0.951
	03	61.938	47.977	0.982		03	37.999	27.661	0.943
	04	62.115	49.172	0.982		04	34.343	24.131	0.954
	05	63.258	49.746	0.981		05	34.659	23.874	0.953
	06	61.965	47.432	0.982		06	33.769	23.755	0.955
	07	63.106	47.725	0.981		07	34.629	24.025	0.953
	08	59.727	45.552	0.983		08	40.328	28.203	0.936
	09	62.808	47.794	0.981		09	34.098	24.028	0.954
	10	63.316	48.992	0.981		10	35.335	24.864	0.951
	AVERAGE	**62.741**	**48.317**	**0.981**		**AVERAGE**	**35.461**	**24.862**	**0.950**
	STDEV	**1.933**	**1.871**	**0.001**		**STDEV**	**2.086**	**1.664**	**0.006**
	CV	**0.031**	**0.039**	**0.001**		**CV**	**0.059**	**0.067**	

[2]: Snapshot of Predicted Results

(Note: Since there are 200 runs in total and is not really feasible to display them all, we here only show the first run's result of both LSTM an GRU model across all indexes)

Figure 6: Snapshot of LSTM's First Run Prediction Results



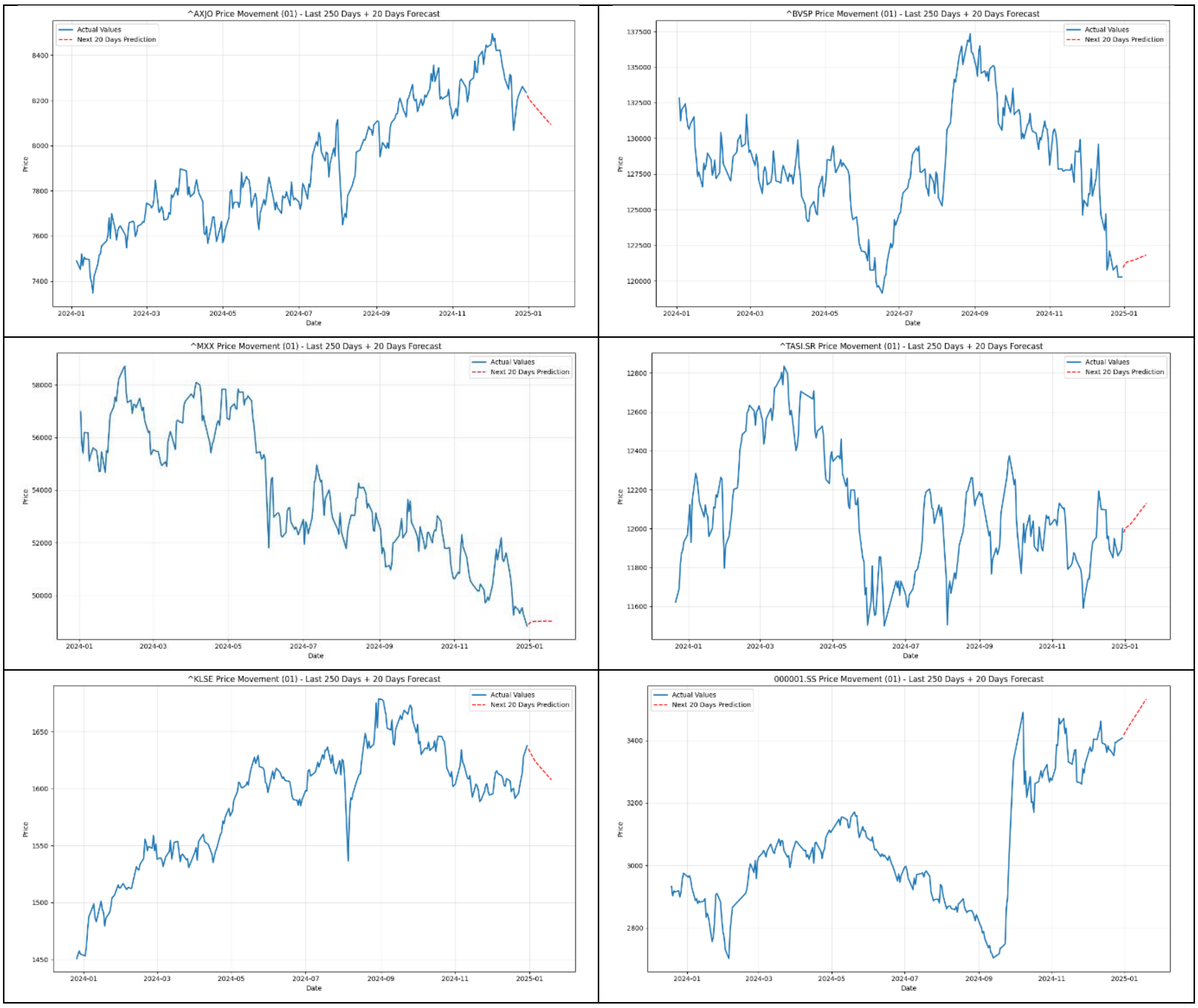
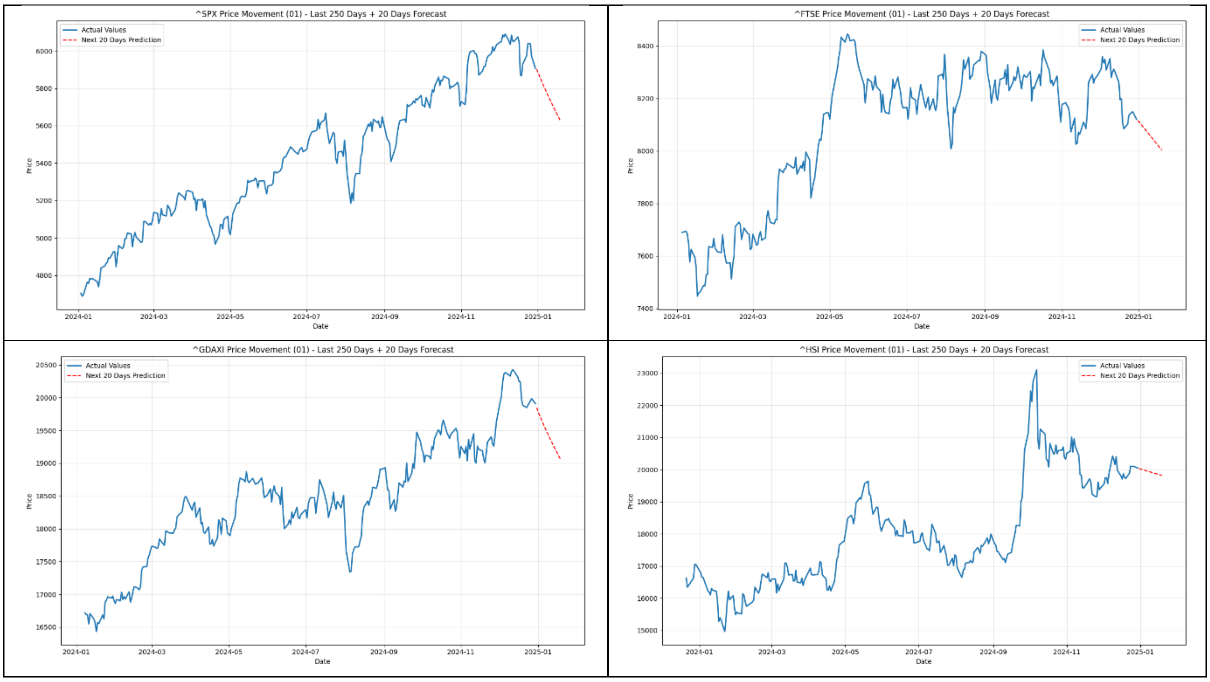
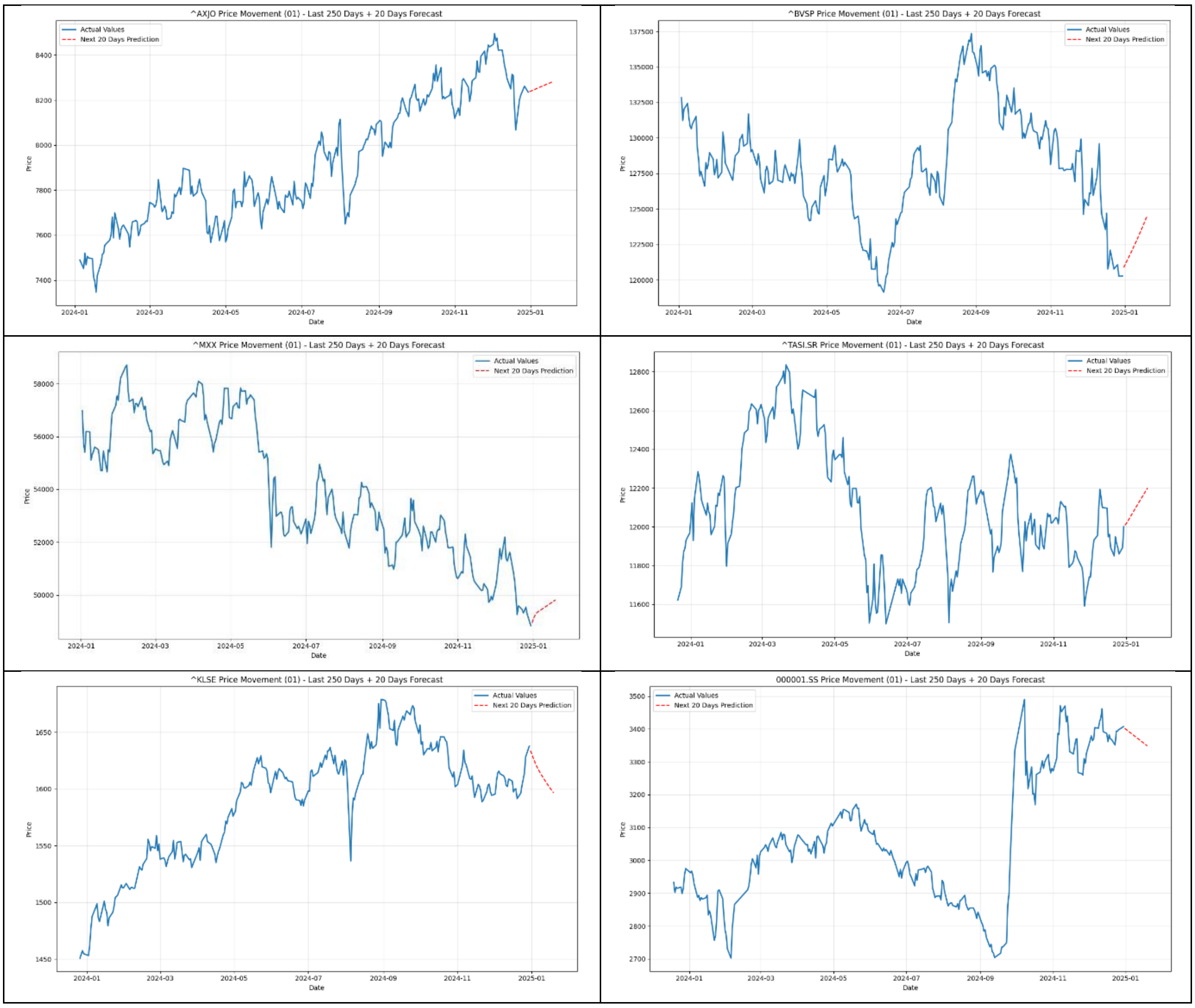


Figure 7: Snapshot of GRU's First Run Prediction Results

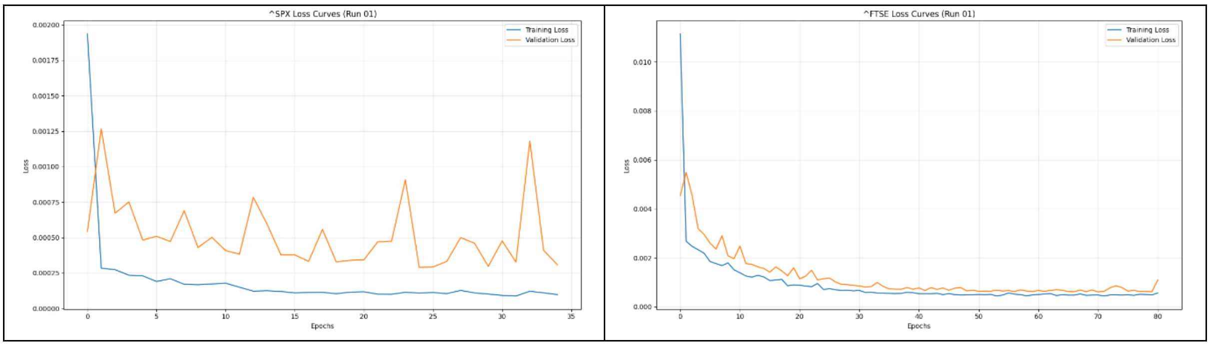




[3]: Snapshot of Loss Function over Epochs

(Note: Since there are 200 runs in total and is not really feasible to display them all, we here only show the first run's result of both LSTM an GRU model across all indexes)

Figure 8: Snapshot of LSTM's First Run Training and Validation Loss



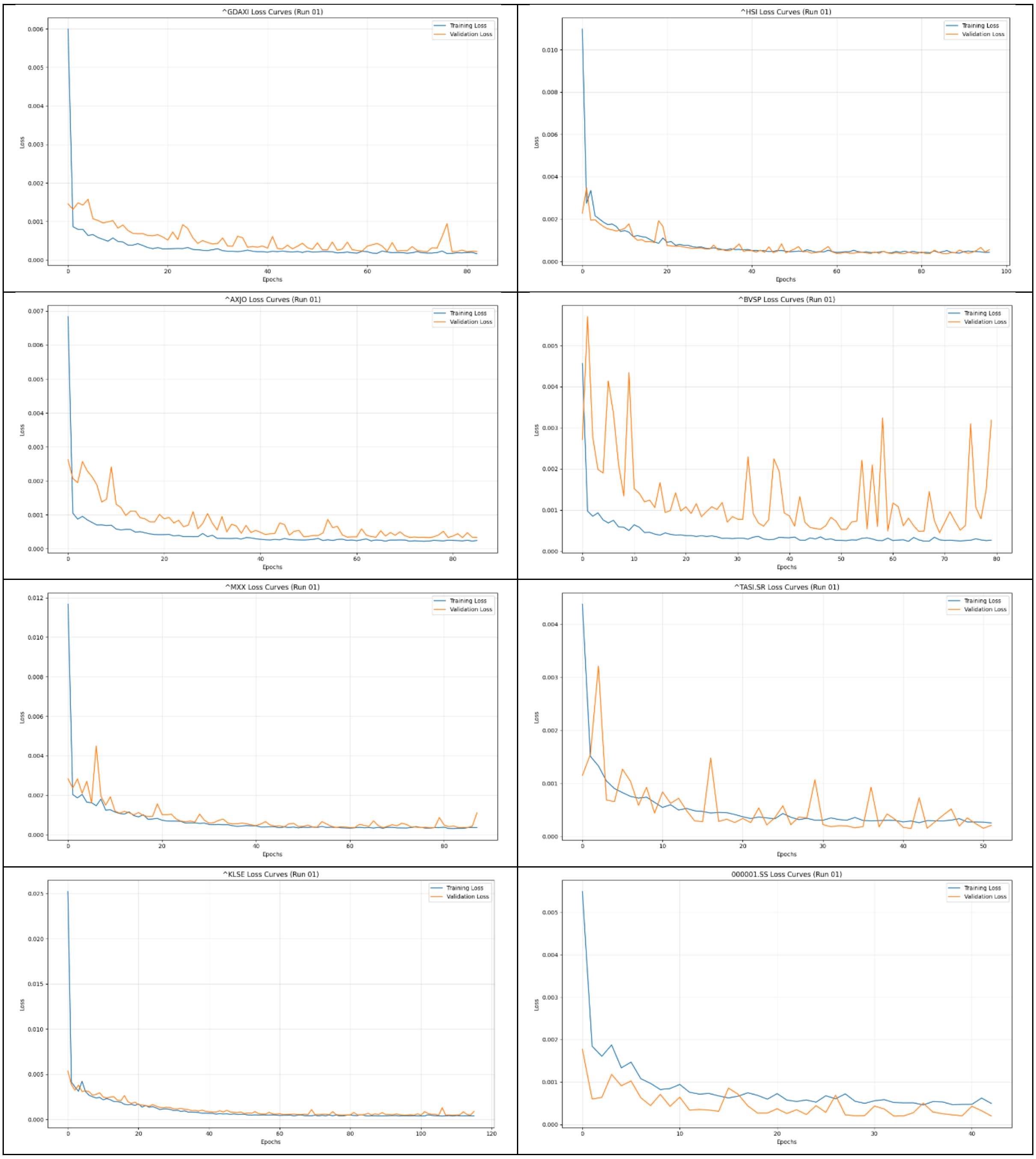
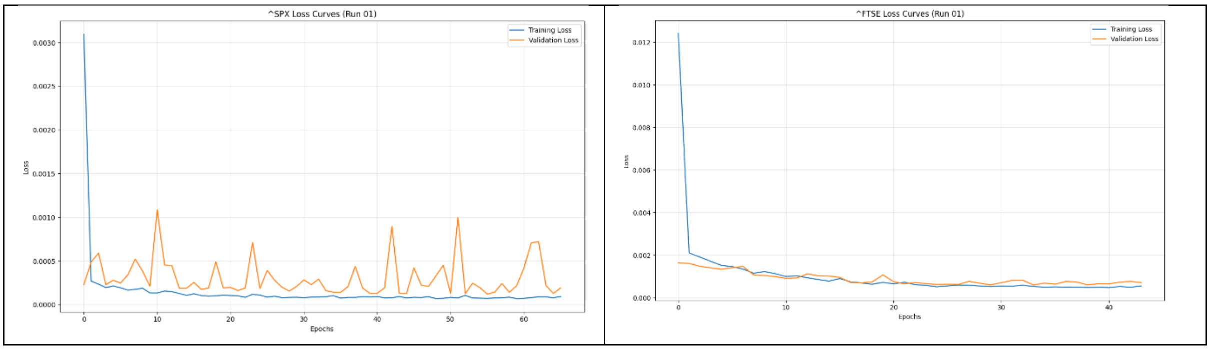
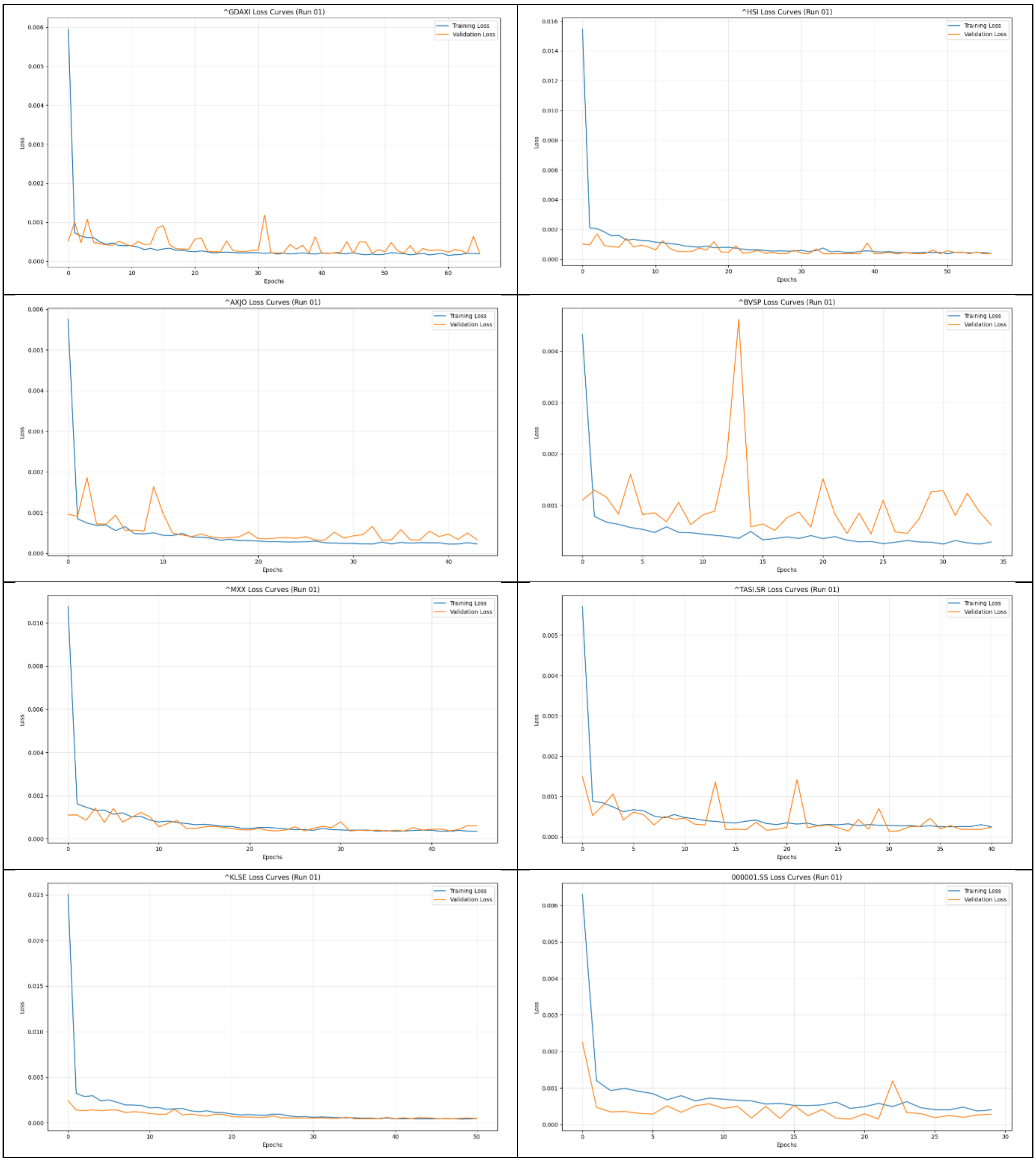


Figure 9: Snapshot of GRU's First Run Training and Validation Loss





[4]: Full Python Script of the Project

```
import yfinance as yf
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from tensorflow.keras.layers import GRU, Dense, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping

# List of tickers
tickers = ["^SPX", "^FTSE", "^GDAXI", "^HSI", "^AXJO","^BVSP", "^MXX", "^TASI.SR",
"^KLSE", "000001.SS"]
```

```
# Hyperparameters
MOVING_WIN_SIZE = 250
PREDICTION_DAYS = 20
DS_SPLIT = 0.8
EPOCHS = 1000

# Store results
metrics_table = []

for ticker in tickers:
    for i in range(1, 6):
        print(f"PROCESSING: {ticker} {i:02d}\n")

        # Load dataset
        df = yf.Ticker(ticker).history(start="2010-01-01", end="2024-12-31")
        df = df.filter(["Close"])

        # Normalize data
        scaler = MinMaxScaler(feature_range=(0, 1))
        scaled_prices = scaler.fit_transform(df.values)

        # Create sequences
        all_x, all_y = [], []
        for j in range(len(scaled_prices) - MOVING_WIN_SIZE):
            x = scaled_prices[j:j + MOVING_WIN_SIZE]
            y = scaled_prices[j + MOVING_WIN_SIZE]
            all_x.append(x)
            all_y.append(y)

        all_x, all_y = np.array(all_x), np.array(all_y)

        # Split dataset
        train_ds_size = round(all_x.shape[0] * DS_SPLIT)
        train_x, train_y = all_x[:train_ds_size], all_y[:train_ds_size]
        test_x, test_y = all_x[train_ds_size:], all_y[train_ds_size:]

        # Build GRU model
        model = Sequential([
            GRU(units=128, return_sequences=True, input_shape=(train_x.shape[1], 1)),
            Dropout(0.2),
            GRU(units=128, return_sequences=True),
            Dropout(0.2),
            GRU(units=128),
            Dropout(0.2),
            Dense(units=8),
            Dense(units=1)
        ])

        model.compile(optimizer="adam", loss="mean_squared_error")
        callback = EarlyStopping(monitor="val_loss", patience=10,
restore_best_weights=True)
        model.fit(train_x, train_y, validation_split=0.2, epochs=EPOCHS,
callbacks=[callback], verbose=0)

        # Make predictions on test set
        preds = model.predict(test_x)
        preds = scaler.inverse_transform(preds)
        test_y_actual = scaler.inverse_transform(test_y)

        # Evaluate model
        rmse = np.sqrt(mean_squared_error(test_y_actual, preds))
```

```
        mae = mean_absolute_error(test_y_actual, preds)
        r_squared = r2_score(test_y_actual, preds)

        # Store in table
        metrics_table.append([ticker, i, rmse, mae, r_squared])

        # Predict the next 20 days (recursive method)
        last_window = df[-MOVING_WIN_SIZE:].values  # Last 250 days
        last_window_scaled = scaler.transform(last_window)
        X_test = np.array([last_window_scaled])
        predicted_prices = []

        for j in range(PREDICTION_DAYS):
            pred_price = model.predict(X_test)
            predicted_prices.append(pred_price[0, 0])
            # Update the input sequence with the new prediction for the next iteration
            pred_price_reshaped = pred_price.reshape(1, 1, 1)
            X_test = np.concatenate((X_test, pred_price_reshaped), axis=1)
            X_test = X_test[:, 1:, :]

        predicted_prices = np.array(predicted_prices).reshape(-1, 1)
        predicted_prices = scaler.inverse_transform(predicted_prices)

        # Plot results - Updated to show last 250 days and 20-day prediction
        # Get the last 250 days of real data
        last_250_days = df.iloc[-MOVING_WIN_SIZE:]

        plt.figure(figsize=(12, 6))
        plt.plot(last_250_days.index, last_250_days["Close"], linewidth=2,
label="Actual Values")

        # Future dates for predictions
        future_dates = pd.date_range(start=df.index[-1],
periods=PREDICTION_DAYS+1)[1:]
        plt.plot(future_dates, predicted_prices, 'r--', label=f"Next {PREDICTION_DAYS}
Days Prediction")

        plt.xlabel("Date")
        plt.ylabel("Price")
        plt.legend()
        plt.title(f"{ticker} Price Movement ({i:02d}) - Last {MOVING_WIN_SIZE} Days +
{PREDICTION_DAYS} Days Forecast")
        plt.grid(True, alpha=0.3)
        plt.tight_layout()
        plt.show()

        # Print metrics for each run
        print(f"\n  RMSE: {rmse:.5f}")
        print(f"  MAE:  {mae:.5f}")
        print(f"  R²:   {r_squared:.5f}\n")

# Display results table
metrics_df = pd.DataFrame(metrics_table, columns=["Ticker", "Run", "RMSE", "MAE", "R²"])
print("\n=== Performance Evaluation ===")
print(metrics_df)
```