Deep Learning Fundamentals - Assessment 3 - Comparative Analysis of RNN Architectures for Stock Market Prediction

Mrinank Sivakumar
University of Adelaide
Adelaide SA 5005
mrinank.sivakumar@student.adelaide.edu.au

Abstract

This research explores the deployment of Recurrent Neural Networks (RNNs) for predictive modelling of stock prices, with a specific focus on Google's historical stock price data. The study rigorously evaluates three RNN architectures: a Vanilla RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), in order to assess their efficacy in time-series forecasting. The dataset, obtained from Kaggle [14], was comprehensively pre-processed, including feature scaling, sequence generation, and splitting into training, validation, and test sets to ensure robust model evaluation. Each model architecture was subjected to systematic hyperparameter optimization using the Optuna framework, with the aim of maximizing predictive accuracy. The performance of the models was quantitatively assessed through Mean Squared Error (MSE) and R-squared metrics, highlighting critical trade-offs between architectural complexity and predictive performance. Furthermore, detailed visual analyses juxtaposing predicted and actual stock prices elucidate the comparative advantages and constraints of these architectures, contributing to a deeper understanding of their applicability in the domain of financial time series prediction.

1. Introduction

Forecasting stock prices is a challenging task due to the unpredictable and fast-changing nature of financial markets. Many factors influence market movements, including economic indicators, political events, public sentiment, and the performance of individual companies [7]. This complexity makes predicting stock prices both a difficult problem to solve and an essential tool for investors, financial institutions, and policymakers. Accurate forecasts can help with tasks such as improving investment strategies, managing risk, and developing economic policies [15]. Traditional methods, such as autoregressive models like ARIMA or exponential smoothing techniques, often fall short when it comes to capturing complicated patterns in financial data [3]. These limitations show the need for more advanced techniques that can better handle the complexities of time-series data.

Recurrent Neural Networks (RNNs) have become a powerful tool for modeling sequential data because they are designed to analyze patterns that unfold over time [12]. Unlike traditional neural networks, RNNs have loops that allow them to store information from previous time steps, making them particularly useful for forecasting. Among RNNs, advanced types like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) are especially effective. These models address common problems, like the vanishing gradient issue, that can occur when working with standard RNNs, particularly in datasets with long-term dependencies [10]. LSTMs achieve this with special memory cells and gates to control how information is stored and forgotten. GRUs, on the other hand, offer a simpler and more efficient design while still delivering high performance [5].

This study leverages advanced RNN architectures to predict stock prices based on historical data from Google's stock dataset. Vanilla RNN, LSTM, and GRU models are compared to understand their differences in complexity, speed, and accuracy. The Optuna framework is used to fine-tune hyperparameters, enhancing model performance [1]. The dataset is carefully prepared through processes such as scaling, generating sequences, and splitting into training, validation, and test sets. Evaluation metrics such as Mean Squared Error (MSE) and R-squared scores provide a detailed analysis of model performance. In addition, visual comparisons between predicted and actual stock prices offer clear insight into how well each model performs.

The implications of this study extend beyond the prediction of the stock price. By analyzing these RNN architectures, useful guidance is provided for future research in time-series forecasting and financial modelling. The study highlights the importance of selecting the right model based on specific needs, such as accuracy and efficiency, while

also demonstrating how advanced optimization techniques can improve performance.

The rest of this paper covers key related research, describes the dataset and experimental methods, presents the results of the analysis, and discusses the broader significance of the findings for financial forecasting. Practical applications are explored and ideas for future research in this field are proposed.

2. Problem Background

Predicting stock prices has always been a significant challenge in financial modelling. Traditionally, this has been tackled using statistical and econometric methods. Techniques like autoregressive integrated moving average (ARIMA) and generalized autoregressive conditional heteroskedasticity (GARCH) models are widely used due to their mathematical structure and clarity [3]. ARIMA is especially good at identifying and modelling linear relationships over time, while GARCH extends this by handling fluctuations in market volatility, a common feature in financial data. However, these methods struggle when dealing with more complicated patterns, such as non-linear relationships or long-term dependencies, and they often require assumptions about the data that are not always realistic [15]. Their reliance on stationary data makes them less effective when market conditions change over time.

Machine learning has introduced more flexible alternatives. Algorithms such as support vector regression (SVR) and ensemble models such as random forests and gradient boosting have shown improvements in predicting structured data [6, 8]. These models handle non-linear relationships well and are generally robust to noisy data. However, their biggest limitation is their inability to process sequential data directly. To model time-dependent relationships, significant feature engineering is often required [11]. While these methods work well for static datasets, they can fall short in capturing the complex and dynamic interactions typical of financial markets.

Deep learning has revolutionized forecasting, especially for time-series data. Recurrent Neural Networks (RNNs) have become one of the most effective tools for analyzing sequences because they can retain information over multiple time steps through their feedback loops [12]. However, basic RNNs are not without their own set of problems. They often suffer from vanishing or exploding gradients, which can prevent them from learning patterns over long time sequences [13]. These challenges have inspired efforts to improve RNN architectures, particularly for applications like financial forecasting, where understanding long-term trends is crucial.

Advanced RNN architectures, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), were developed to overcome these limitations. LSTMs use memory cells and gating mechanisms to retain and manage information over time, effectively addressing the problem of vanishing gradients [10]. GRUs provide a simpler alternative to LSTMs, using fewer parameters while achieving similar results, making them faster and more computationally efficient [5]. Both architectures have proven to be highly effective in sequential prediction tasks, including financial forecasting, where they can handle noisy data and extract meaningful patterns [9].

Other neural network architectures, such as convolutional neural networks (CNNs) and transformers, have also been adapted for time-series forecasting. CNNs, originally designed for image analysis, treat time-series data as spatial patterns, allowing them to detect local temporal trends through convolutional filters [2]. Variants like temporal convolutional networks (TCNs) use advanced techniques such as dilated convolutions to capture longer sequences. Although CNNs excel at computational efficiency and identifying short-term patterns, they struggle to model global dependencies effectively, which is a critical aspect of financial forecasting.

Transformers, on the other hand, have gained attention for their ability to model long-range dependencies using self-attention mechanisms [16]. Unlike RNNs, transformers process entire sequences in parallel, making them highly scalable and efficient for large datasets. However, their high computational demands and complexity often require substantial resources and large amounts of data to prevent overfitting. This can make them less practical for smaller datasets commonly encountered in financial applications.

In this context, the present study focuses on a detailed comparison of a vanilla RNN, LSTM, and GRU architectures. By examining their unique features and placing them alongside both traditional and modern methods, this research aims to highlight their respective advantages and limitations for stock price prediction. The goal is to identify practical trade-offs between performance, complexity, and computational efficiency, providing insights into the real-world applications of these models.

3. Methodology

The methodology for this study encompasses several critical stages: exploratory data analysis (EDA), data preprocessing, model architecture design, and experimental setup. Each stage is systematically designed to ensure rigour and reproducibility in the evaluation of Recurrent Neural Network (RNN) architectures for stock price prediction.

3.1. Exploratory Data Analysis (EDA)

Comprehensive EDA was performed to investigate the characteristics of the dataset, ensuring data integrity and uncovering relevant patterns. The primary steps included the

following.

1. **Statistical Overview:** Descriptive statistics, such as mean, median, standard deviation, and interquartile range, were computed for features (Open, High, Low, Close, Volume) to reveal distributions and variability.
2. **Feature Correlation:** A correlation heatmap was created to identify the relationships between variables, facilitating the selection of predictors relevant to forecasting.
3. **Trend Analysis:** Time-series visualizations were used to detect trends, seasonality, and anomalies in stock prices.
4. **Handling Missing Data:** Missing or inconsistent entries were identified and addressed through interpolation or deletion to ensure the completeness of the dataset.

3.2. Data Preprocessing

The dataset consists of historical Google stock price records, retrieved from publicly accessible repositories. The preprocessing involved the following steps:

1. **Feature Scaling:** The input features were normalized using MinMaxScaler to map the values to the range [0, 1], improving the convergence of the model by reducing the magnitude discrepancies of the feature.
2. **Dataset Splitting:** The dataset was provided pre-split into training and testing sets. From the training set, 20% of the data was further split into a validation set to optimize the hyperparameters while preserving the temporal order.
3. **Sequence Formation:** Data sequences were generated using a sliding window technique. A sequence of length T days was used to predict the stock price on day $T+1$. Mathematically, a sequence $[p_1, p_2, ..., p_T]$ corresponds to the target p_{T+1}.

3.3. Model Architectures

Three RNN architectures were designed and evaluated: Vanilla RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). These models were selected to explore different mechanisms for sequence modelling and assess their suitability for time-series forecasting.

3.3.1 Vanilla RNN

The Vanilla RNN is the foundational recurrent architecture, defined mathematically as:

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \tag{1}$$

where:

- h_t: Hidden state at time step t.
- x_t: Input at time step t.
- W_{xh}, W_{hh}: Weight matrices for input-to-hidden and hidden-to-hidden transitions.
- b_h: Bias term.
- σ: Activation function (e.g., tanh).

Although effective for short sequences, vanilla RNNs often encounter vanishing gradient issues during long-sequence training.

3.3.2 Long Short-Term Memory (LSTM)

LSTMs enhance Vanilla RNNs by introducing gates to regulate information flow, mathematically represented as:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \tag{2}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \tag{3}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \tag{4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \tag{5}$$
$$h_t = o_t \odot \tanh(c_t), \tag{6}$$

where:

- i_t, f_t, o_t: Input, forget, and output gates.
- c_t: Cell state managing long-term memory.
- h_t: Hidden state, representing the output at the current time step t.
- σ: Sigmoid activation function.
- $\odot$: Element-wise multiplication.

These mechanisms mitigate gradient-related issues, enabling LSTMs to model long-term dependencies effectively.

3.3.3 Gated Recurrent Unit (GRU)

GRUs simplify LSTMs by combining input and forget gates into a single update gate. The operations of a GRU are defined as:

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z), \tag{7}$$
$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r), \tag{8}$$
$$\tilde{h}_t = \tanh(W_{xh}x_t + r_t \odot (W_{hh}h_{t-1} + b_h)), \tag{9}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{10}$$

where:

- z_t: Update gate, determining how much of the previous state is retained.
- r_t: Reset gate, controlling the influence of past states on the current computation.
- $\tilde{h}_t$: Candidate hidden state.
- h_t: Final hidden state at time step t.
- σ: Sigmoid activation function.
- $\odot$: Element-wise multiplication.

GRUs are computationally efficient while maintaining performance comparable to LSTMs.

3.4. Project Setup

1. **Hyperparameter Tuning:** The Optuna framework was utilized to optimize hyperparameters such as the number of hidden units, learning rate, and batch size, aiming to minimize the Mean Squared Error (MSE) on the validation.
2. **Training Protocol:** Models were trained using the Adam optimizer with scheduled learning rates.
3. **Evaluation Metrics:** Model performance was quantified using MSE, Mean Directional Accuracy (MDA), and R-squared metrics. In addition, prediction versus actual stock price plots were generated for qualitative assessment.

This methodological framework provides a rigorous basis for evaluating RNN architectures, emphasizing robust EDA, precise preprocessing, mathematically driven model design, and comprehensive evaluation strategies.

4. Experimental Analysis

This section provides a comprehensive evaluation of Vanilla RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) architectures in the context of stock price prediction. The performance of these models was systematically assessed using Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Directional Accuracy (MDA), and R-squared (R^2) metrics, as well as detailed visualizations comparing the predicted and actual stock prices.

4.1. Training and Validation Performance

The training and validation loss curves for all three models reveal a similar convergence behaviour, with slight differences:

- **Vanilla RNN:** The training and validation loss curves showed minor fluctuations compared to LSTM and GRU. These fluctuations indicate some instability in the model's ability to generalize effectively, although the overall convergence pattern was still acceptable.
- **LSTM:** The LSTM demonstrated smooth and consistent convergence in both training and validation losses, reflecting its robustness in modelling temporal patterns.
- **GRU:** The GRU also exhibited stable convergence, nearly identical to the LSTM, with both training and validation losses stabilizing at similar levels. The GRU's performance highlights its ability to match the LSTM's results with fewer computational requirements.

4.2. Quantitative Metrics

The test set results, summarized in Table 1, provide quantitative evidence of model performance:

Models	MSE	RMSE	Accuracy
RNN	[0.00015]	[0.012]	[0.67]
LSTM	[0.00024]	[0.015]	[0.67]
GRU	[0.00032]	[0.018]	[0.67]

Table 1. Model Performance Summary

4.3. Predicted vs Actual Stock Prices

The comparison of predicted and actual stock prices for each model highlights their predictive capabilities:

- **Vanilla RNN:** The vanilla RNN predictions aligned reasonably well with the actual stock prices but showed slightly larger deviations in capturing sharp peaks and troughs compared to the LSTM and GRU.
- **LSTM:** The LSTM predictions somewhat closely followed the actual stock prices, sufficiently capturing short-term variations. This reinforces its effectiveness for stock price prediction tasks.
- **GRU:** The GRU predictions were comparable to the LSTM, semi-accurately modelling short-term fluctuations with negligible differences.

4.4. Model Insights and Trade-offs

The experimental findings showed that the vanilla RNN was computationally efficient but had a lower directional accuracy in predicting stock prices. Surprisingly, the LSTM, even with its advanced gating mechanisms to handle temporal dependencies, achieved the same accuracy as the vanilla

RNN. Similarly, the GRU matched the accuracy of both the LSTM and the Vanilla RNN but required fewer computational resources.

These results highlight the importance of choosing a model architecture that fits the specific needs of the task, while also balancing accuracy and computational efficiency. However, they also emphasize how crucial thorough data preprocessing and careful hyperparameter tuning are for improving model performance.

5. Code and Visualizations

Find the code implementation in the GitHub repository for this project [4].

5.1. Figures

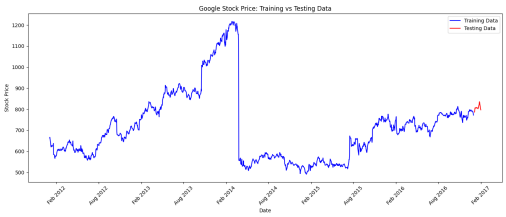

Figure 1. Google Stock Price - Training vs Testing Data

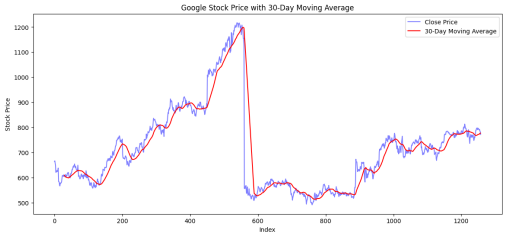

Figure 2. Google Stock Price with 30-Day Moving Average

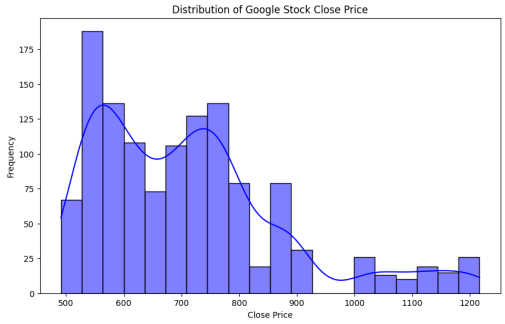

Figure 3. Distribution of Google Stock Close Price

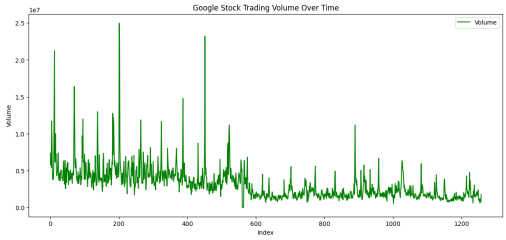

Figure 4. Google Stock Trading Volume Over Time

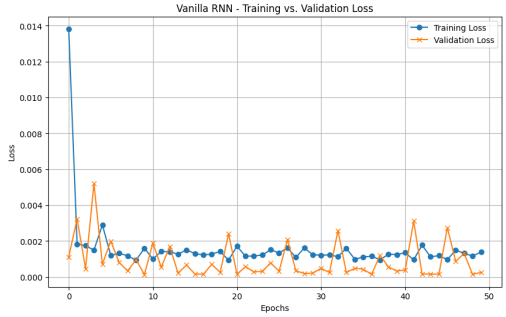

Figure 5. Vanilla RNN - Training vs Validation Loss

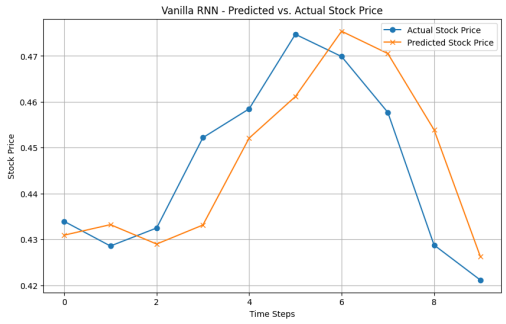

Figure 6. Vanilla RNN - Predicted vs Actual Stock Price

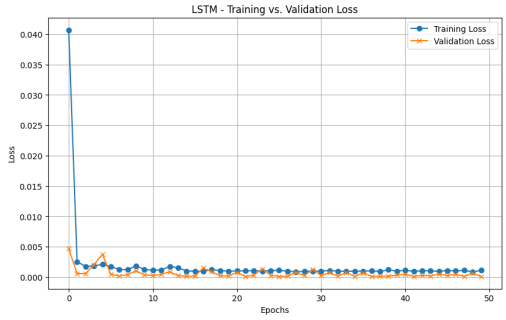

Figure 7. LSTM - Training vs Validation Loss

6. Conclusion

This study provided comprehensive evaluations of the performance of a vanilla RNN, LSTM, and GRU architec-

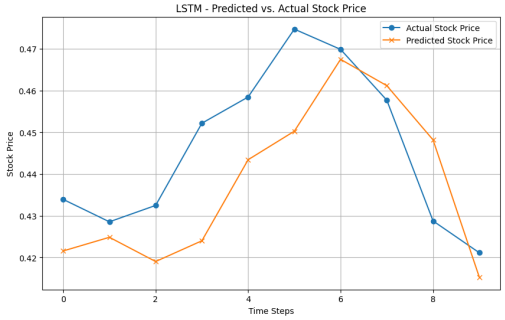

Figure 8. LSTM - Training vs Validation Loss

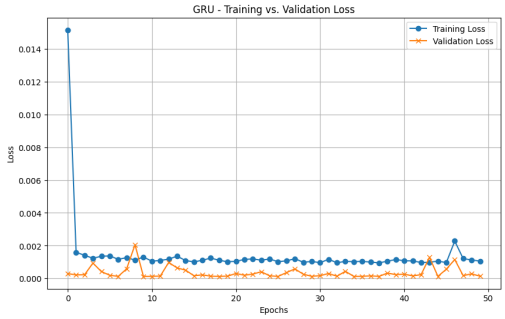

Figure 9. GRU - Training vs Validation Loss

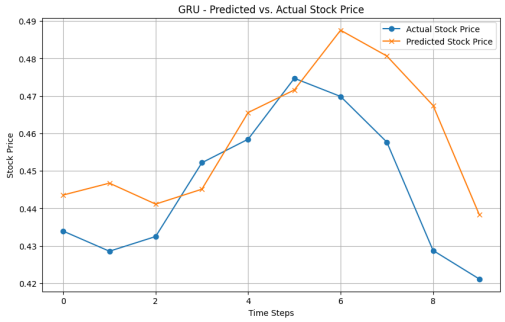

Figure 10. GRU - Predicted vs Actual Stock Price

ture for stock price prediction, offering a detailed comparison of their strengths and limitations. Although the computational designs of these models differed significantly, they all achieved an identical directional accuracy of 67%. This result underscores the critical role of rigorous data preprocessing and hyperparameter tuning, which often outweigh architectural complexity in determining performance for this specific application.

6.1. Summary of Results

The analysis revealed that the vanilla RNN exhibited the best numerical performance among the three models, achieving an MSE of 0.00015 and an RMSE of 0.012. Despite its relatively simple architecture, the vanilla RNN demonstrated notable effectiveness in this context. The LSTM, leveraging advanced gating mechanisms to model long-term dependencies, recorded a slightly higher MSE of 0.00024 and a RMSE of 0.015, indicating that its additional complexity did not result in a significant performance advantage. The GRU, characterized by its computational efficiency and simplified gating mechanisms, produced the highest MSE of 0.00032 and RMSE of 0.018. These findings suggest that, while the Vanilla RNN marginally outperformed the LSTM and GRU in numerical accuracy, the overall differences between the models were minimal.

6.2. Analysis of Predictive Performance

The predictive performance analysis highlighted common challenges faced by all three models. Each struggled to accurately capture sharp peaks and sudden declines in stock prices, particularly during periods of high volatility, as seen in financial datasets. However, all models demonstrated competence in identifying broader trends, validating their utility for general trend forecasting tasks. The training and validation loss curves showed stable convergence for all architectures, with the vanilla RNN displaying slightly more variability in validation loss compared to the LSTM and GRU. This variability suggests minor instability, but does not detract from its overall reliability.

A notable observation was the sharp dip in the Google stock price graph, attributed to a share split. This anomaly introduced an abrupt and unrepresentative trend in the dataset, potentially misleading the models. Such events underscore the importance of robust data preprocessing to mitigate the impact of anomalies and improve the reliability of the model.

6.3. Implications and Future Work

The findings of this study emphasize the necessity of balancing the complexity of the model with effective data preparation and optimization strategies. Although the theoretical advantages of LSTM and GRU architectures lie in their ability to capture long-term dependencies, these benefits were not evident in this experiment, as all three models achieved identical accuracy. This highlights the need to refine data preprocessing pipelines and explore advanced feature engineering methods to fully harness the potential of these architectures.

A critical insight from this analysis was the influence of anomalies, such as the share split, on the performance of the model. Future research should focus on incorporating techniques for anomaly detection and exclusion to ensure that models learn from meaningful and representative patterns. Cleaning datasets to address such irregularities will likely result in improved prediction accuracy.

Further investigations could explore ensemble methods that combine the unique strengths of these architectures to

enhance robustness and predictive accuracy. In addition, integrating external variables such as macroeconomic indicators, market sentiment, and trading volumes could enrich the dataset, enabling more sophisticated forecasting models. Researchers may also benefit from experimenting with loss functions specifically designed for financial time-series data and testing these models on larger, more diverse datasets with varying levels of volatility. Such approaches could provide deeper insights into the adaptability and practical applications of these architectures in real-world scenarios.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 2623–2631. Association for Computing Machinery, 2019. 1

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018. 2

[3] George E.P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice-Hall, 1994. 1, 2

[4] BrownAssassin. Dlf - a3 - rnn comparative analysis. https://github.com/BrownAssassin/DLF---A3---RNN-Comparative-Analysis, 2024. 5

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 1, 2

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. 2

[7] Eugene F. Fama. Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, 25:383–417, 1970. 1

[8] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. 2

[9] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017. 2

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 1, 2

[11] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2018. 2

[12] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint*, 2015. 1, 2

[13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318. PMLR, 17–19 Jun 2013. 2

[14] Rahul Sah. Google stock price dataset. https://www.kaggle.com/datasets/rahulsah06/gooogle-stock-price?resource=download, 2024. 1

[15] Ruey S. Tsay. *Analysis of Financial Time Series*. John Wiley & Sons, 2005. 1, 2

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2