Investigating the use of three recurrent neural networks (RNN) – Vanilla RNN, LSTM and GRU to predict Amazon's (AMZN) stock price for the next 30-days based on Year-to-Date (YTD) stock information.

a1772276

Abstract

This report outlines the investigation undertaken to view the impact of architectural differences in Recurrent Neural Networks, specifically, Vanilla RNNS, GRU and LSTM on their efficiency in predicting stock based on past data. It was found that LSTM performed the best, as hypothesised. Since a vanilla RNN loses information overtime due to its vanishing gradient problem, this is catered to by an LSTM. Therefore, using Year-to-Date data from December 2024 for Amazon's stock information, the LSTM, with minor mean-squared error was able to predict the stock's price for the next give days.

1. Introduction

Recurrent Neural Networks (RNNs) are a specialized type of artificial neural network designed to process sequential data, making them highly effective for tasks involving time series, natural language, and other ordered inputs [1][2]. Conventional neural networks have independent inputs and outputs. However, in an RNN, its unique architecture involving a specialized memory mechanism enables it to consider past information when generating predictions [3]. For every iteration, the output of the previous iteration is used as input to consider it as important information for the next calculation steps, thus allowing the neural network to consider all previous information, unlike a generic neural network where each output is distinct.

This feature of RNNs is described as hidden states, which function as memory units [6]. This state is continuously updated as the network processes each element in a sequence, allowing it to capture and utilize information from previous inputs [3]. This architecture makes RNNs particularly well-suited for tasks where context is crucial, such as language modelling or sentiment analysis [5]. To support this mechanism, the same set of weights is applied across all elements in a sequence [4][7]. This parameter sharing enables RNNs to handle sequences of varying lengths while keeping the number of parameters lower compared to traditional feedforward networks [3][6].

This special RNN architecture makes them suitable for Natural Language Processing (NLP), Speech recognition, Machine translation, Time series analysis etc.

This paper will introduce traditional RNNs and compare the architectures of three-different types of RNNs, including a Vanilla RNN, Long-Short Term Memory (LSTM) and Gates Recurrent Unit (GRU) architectures.

2. Method Description

2.1.Dataset

To conduct this investigation, a dataset of time-series nature was used. The daily historical stock data from December 2023-December 2024 for Amazon (AMZN) was used as the primary dataset for this investigation. It included the features – Date, Close/Last, Open, High, Low and Volume.

This dataset was cleaned to remove the '$' symbols for all price describing columns. All features were then normalised using Sci-kit-Learn's MinMax Scaler.

This dataset was split into train, validation and test sets and used for training three different types of architectures – Vanilla RNN, LSTM and GRU.

2.2. Vanilla Recurrent Neural Network (Vanilla- RNN)

Vanilla RNN are the simplest form of an RNN. RNNs involve a hidden state that store and provide the output of th previous iteration as an input to the next, allowing the neural network to take-in all information sequentially and synthesise a pattern between the inputs overtime. However, with increasing number of iterations, an RNN may eventually drop or forget the older input provided to it. This occurs due to a phenomenon called the "vanishing gradient problem". This indicates that the gradient or weight term for the older inputs *vanishes* as it gets closer to 0, with increasing iterations. Naturally, this hampers an RNN's ability to learn long-term dependencies effectively [8].

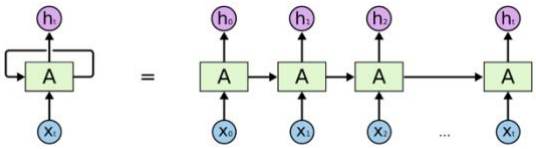
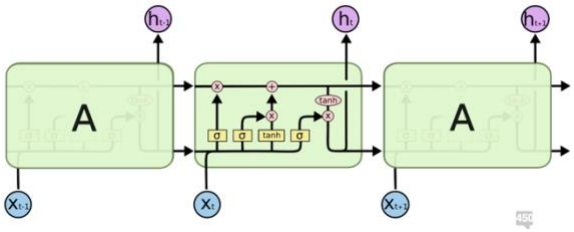


Figure 1: Demonstration of the RNN Architecture

2.3.Long-Short term Memory (LSTM)

LSTM networks are a specialized type of RNN that address the limitations of standard RNNs – i.e., the loss of memory due to the vanishing gradient. They include three elements in addition to the hidden state – namely, the input gate, forget gate, and output gate. These gates orchestrate the flow of information, allowing LSTMs to retain relevant information over longer sequences while mitigating the vanishing gradient problem. This architecture enables LSTMs to excel in tasks requiring long-term memory, such as language modelling and time series prediction [9].

Figure 2: Demonstration of LSTM Architecture



2.4. Gates Recurrent Unit (GRU)

Gated Recurrent Units (GRUs) are another variant of RNNs that simplify the LSTM architecture by combining the forget and input gates into a single update gate. GRUs also utilize a reset gate to determine how balance the information in memory, i.e., they help in determining which information to forget. This streamlined design reduces

computational complexity while still effectively capturing dependencies in sequential data. GRUs have been shown to perform comparably to LSTMs on various tasks, particularly in scenarios with less complex data [10].

2.5.Comparison of the Architectures

While RNNs serve as foundational models for processing sequential data, they often struggle with long-term dependencies due to their simple architecture. In contrast, both LSTMs and GRUs are designed to overcome these limitations by incorporating gating mechanisms that control the flow of information. LSTMs excel in handling high-complexity sequences due to their more intricate structure but may require more computational resources. GRUs offer a simpler alternative that can outperform LSTMs on low-complexity sequences while being computationally efficient. Ultimately, the choice between these architectures depends on the specific characteristics of the task at hand and the complexity of the data being processed [9][10].

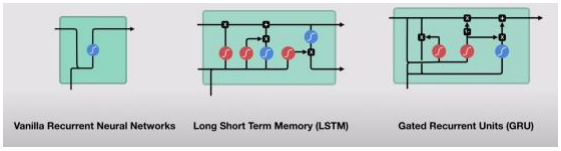


Figure 3: Comparative Architectures of RNN, LSTM and GRU. The red states demonstrate the three additional gates - forget, input and output in LSTM and GRU Architectures

3. Method Implementation

- **Link to GitHub Repo**

4. Experiment and Analysis

This report presents an analysis based on four experiments conducted to evaluate the performance of Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) architectures. The aim of these experiments was to understand the hyperparameters each RNN architecture needs to perform its best.

4.1.Hyperparameter Tuning

The objective of this experiment was to investigate the impact of hyperparameter settings on model performance. Particularly, learning rate and dropout rates were the chosen parameters to be varied in this experiment. Learning Rates was varied between 0.001, 0.01 and 0.1; while drop-out rate values were varied between 0.2 and 0.5.

The LSTM consistently achieved the highest accuracy across all learning rates, demonstrating its robustness to hyperparameter changes. 0.01 Learning rate performed the best amongst others. GRU performed well at lower learning rates but struggled with higher rates, leading to overfitting. RNN exhibited significant sensitivity to the learning-rate changes, resulting in lower overall performance compared to LSTM and GRU. However, consistently across all three architectures, 0.01 performed the best, while 0.001 performed moderately, with 0.1, performing most erratically.

4.2. Sequence Length Variation

The objective of this experiment was to analyse how different input sequence lengths affect model accuracy. Input

sequences were tested at lengths of 10-, 50-, and 100-time steps.
Initially, smaller time-steps were used, including 10, 30 and 60. LSTM significantly outperformed both GRU and RNN on longer sequences (100 time-steps), effectively capturing long-term dependencies due to its gating mechanisms. GRU showed decent performance on shorter sequences but could not match LSTM's accuracy on longer ones. RNN struggled with some short and all long sequences, losing context of older data over time.

In conclusion, the experiments clearly demonstrated that LSTM is the superior architecture for handling time-series data when compared to RNN and GRU. Its ability to capture long-term dependencies while maintaining robustness against overfitting makes it the preferred choice for complex tasks. However, within this experiment LSTM still experienced overfitting, potentially due to limited data samples.

The overall comparison of the strengths and weaknesses of the architecture are given in Appendix 2.

5. Reflection

As expected, the investigation results demonstrate the superiority of LSTM in capturing both short-term and long-term dependencies in stock price data. The GRU model's performance was close to that of LSTM, suggesting it could be a viable alternative in scenarios where computational efficiency is a priority. The traditional RNN, while still providing reasonable predictions has major variations between the actual and predicted stock price values.

As it can be seen in Appendix 1 – LSTM has the most similar predicted stock prices for the next 5 days as compared to a standard RNN and a GRU. Reflecting the same, Figure 4 demonstrates the performance metrics of the three architectures.

	MSE	RMSE	Accuracy
RNN	32.854390	5.731875	84.210526
LSTM	24.481864	4.947915	94.736842
GRU	36.373306	6.031029	78.947368

Figure 4: Performance Metrics for RNN, GRU, and LSTM

However, the dataset used was quite steadily increasing in nature and only included 365 samples. In future experiments, a much larger number of samples will be used. Various versions of the model can be trained including with historical data of the past 6 months, 3 years, or 5 years.

Similarly, a stock that is unstable will also be used for experimentation to view how the architectures will behave with such an unstable pattern. Other aspects of improvements could include:

1. Feature Engineering: Incorporate additional relevant features such as market sentiment, macroeconomic and political indicators.
2. Ensemble Methods: Combine predictions from multiple models or use hybrid-models to potentially improve overall accuracy and robustness.

3. Hyperparameter Tuning: Conduct a more extensive search for optimal hyperparameters using techniques like Grid-Search. A key hyperparameter that can be used for experimentation is variation in time-steps.
4. Longer Forecast Horizon: When using a dataset with a larger number of samples, it will allow for the investigation of the models' performance on longer time-periods (e.g., 60 or 90 days) to assess their long-term forecasting capabilities.

In conclusion, while the LSTM model demonstrated the best performance in this study, there is still room for improvement. Future work will focus on enhancing the model's ability to capture market dynamics and incorporating a wider range of relevant information to improve prediction accuracy.

References

1. GeeksforGeeks (2024) 'Introduction to Recurrent Neural Networks', GeeksforGeeks, 15 November. Available at: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/ (Accessed: 5 December 2024).

2. CAIS++ (no date) 'Recurrent Neural Networks', CAIS++. Available at: https://caisplusplus.usc.edu/curriculum/neural-network-flavors/recurrent-neural-networks (Accessed: 4 December 2024).

3. AlmaBetter (no date) 'Recurrent Neural Network', AlmaBetter. Available at: https://www.almabetter.com/bytes/articles/recurrent-neural-network (Accessed: 5 December 2024).

4. Datafloq (no date) 'Enterprise Applications of Recurrent Neural Networks', Datafloq. Available at: https://datafloq.com/read/enterprise-applications-recurrent-neural-networks/ (Accessed: 5 December 2024).

5. Yao, J., et al. (2017) 'Visual Analysis of Recurrent Neural Networks', Hong Kong University of Science and Technology. Available at: https://cse.hkust.edu.hk/~huamin/vast_yao_rnnvis_2017.pdf (Accessed: 5 December 2024).

6. AWS (no date) 'What is a Recurrent Neural Network?', Amazon Web Services. Available at: https://aws.amazon.com/what-is/recurrent-neural-network/ (Accessed: 4 December 2024).

7. Analytics Steps (no date) 'Recurrent Neural Network (RNN): Types and Applications', Analytics Steps. Available at: https://www.analyticssteps.com/blogs/recurrent-neural-network-rnn-types-and-applications (Accessed: 5 December 2024).

8. Asquith, N., 2020. Understanding RNNs, LSTMs and GRUs. Towards Data Science. Available at: https://towardsdatascience.com/understanding-rnns-lstms-and-grus-ed62eb584d90?gi=0196e8b0c27d [Accessed 4 December 2024].

9. Chen, X., Zhang, H. and Liu, Y., 2023. A comparison of LSTM and GRU networks for learning symbolic sequences. *Intelligent Computing*, pp.771-785 1.

10. Raza, M.R., Hussain, W. and Merigó, J.M., 2021. Cloud Sentiment Accuracy Comparison using RNN, LSTM and GRU. In: Innovations in Intelligent Systems and Applications Conference (ASYU). IEEE.

Appendix

1. Actual vs Predicted Stock Prices for the next 5 Days:

RNN Predictions:

Day 1: Actual 155.20, Predicted 163.96

Day 2: Actual 159.00, Predicted 174.19

Day 3: Actual 161.26, Predicted 178.08

Day 4: Actual 159.12, Predicted 174.18

Day 5: Actual 157.75, Predicted 169.21

LSTM Predictions:

Day 1: Actual 155.20, Predicted 158.57

Day 2: Actual 159.00, Predicted 160.91

Day 3: Actual 161.26, Predicted 163.37

Day 4: Actual 159.12, Predicted 165.64

Day 5: Actual 157.75, Predicted 167.61

GRU Predictions:

Day 1: Actual 155.20, Predicted 160.10

Day 2: Actual 159.00, Predicted 164.85

Day 3: Actual 161.26, Predicted 167.82

Day 4: Actual 159.12, Predicted 170.38

Day 5: Actual 157.75, Predicted 172.65

2. Strengths and Weaknesses of each RNN Model:

Model	Strength	Weakness
Vanilla RNN	Simple architecture; faster training	Poor performance on long sequences; high sensitivity to hyperparameters

LSTM	Excellent for long-term dependencies; robust against overfitting; superior accuracy	More complex architecture; requires more computational resources
GRU	Efficient with fewer parameters; performs well on moderate-length sequences	Inferior performance compared to LSTM on longer sequences