LaTeX Author Guidelines for CVPR Proceedings

Anonymous CVPR submission

Paper ID *****

Abstract

This paper explores the application of recurrent neural network that RNN focusing on stock price prediction using Google stock price data. Here three deep learning architectures are used that is Long Short-term Memory (LSTM), Vanilla RNN and Gated Recurrent Unit (GRU) that are implemented, evaluated and compared in terms of their ability to predict future stock prices. Each model was trained and tuned using a grid search approach to optimize hyper parameters such as the number of units, dropout and rate, batch size and epochs. The dataset was preprocessed using MinMaxScaler to normalize the features and sliding windows were used to create sequences of data for training , validation and testing. Performance metrics including Mean Absolute Error (MAE) and Mean Squared Error (MSE) were also used to evaluate each model. Experimental results demonstrate that all three models capture the temporal dependencies of the data with GRU model showing the most performnace then the others.

1. Introduction

Time-series forecasting plays a crucial role in various industries such as finance, healthcare and supply chain management where accurate predictions can drive informed decision-making. Also Stock price prediction has gained significant attention due to its potential for financial gains and market analysis. Recent advancements in deep learning have introduced powerful methods for modeling time-series data. Recurrent Neural Networks (RNN) that is designed to capture sequential dependencies is used as a robust solution for forecasting tasks. Variants of RNN such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) address issues of vanishing gradients and enable the modeling of long-term dependencies. In this study we aim to compare the performance of three recurrent neural network architectures—Vanilla RNN, LSTM, and GRU for stock price prediction using Google Stock Price data. The dataset is preprocessed and divided into training, validation and testing sets with a sliding window approach applied to create sequences of past observations for forecasting future stock prices. Each model is trained and evaluated using hyperparameter tuning to identify optimal configurations and their performance is compared using metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE). This paper is about evaluation of RNN-based architectures and their suitability for financial time-series data. The results offer insights into the strengths and weaknesses of each model, aiding practitioners in selecting the most appropriate architecture for similar tasks

1.1. Methodology

The methodology for this study involves the implementation, training and evaluation of three neural network architectures that is Vanilla RNN, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) for the task of stock price prediction. The approach consists of several sequential steps that are below:

1.1.1 Data Preparation

The dataset used in this study is the Google Stock Price dataset that contains several features such as Open, High, Low, and Close prices. The following preprocessing steps are performed on the dataset:

1. Feature Selection: Only relevant features such as Open, High, Low and Close were used and missing data such as Volume was excluded.

2. Data Cleaning: Non-numeric values were converted and any rows containing NaN values were removed.

3. Normalization: MinMaxScaler was applied to scale the data into the range 0, 1 to ensuring faster convergence during training.

4. Sliding Window Technique: A sliding window of 60 timesteps was used to create input-output pairs for the models. Each input consists of 60 timesteps and the output predicts the Open price for the next timestep. The dataset was then split into training, validation, and testing sets.

1.1.2 Models

1. **Long Short-Term Memory(LSTM) model:**

LSTM networks are a specialized form of neural networks (RNN) that effectively address the problem of vanishing gradients and making them suitable for capturing long-term dependencies in sequential data. In this study the LSTM architecture was implemented to predict the next-day Open price of Google stock based on the previous 60 days of stock data.

a. **Model Architecture :**

* The model consists of two LSTM layers and the first layer outputs sequential data with a specified number of units that is 50 or 100 and followed by a dropout layer to reduce overfitting.

* The second LSTM layer processes the outputs from the first layer and is also followed by a dropout layer.

* A fully connected Dense layer with one unit is used to predict the next-day Open price.

Hyperparameter Tuning :

* A grid search was performed by the following hyperparameters:

*Number of LSTM units: 50, 100.

*Dropout rate: 0.2, 0.3.

*Batch size: 16, 32.

*Epochs: 50.

*Validation loss (Mean Squared Error) was used to identify the best configuration.

Results and Analysis :

Parameter	**Value**
Units	100
Dropout Rate	0.2
Batch Size	16
Epochs	50
Mean Absolute Error (MAE)	9.14
Mean Squared Error (MSE)	134.65

Table 1. Best Configuration and Performance Metrics for LSTM Model

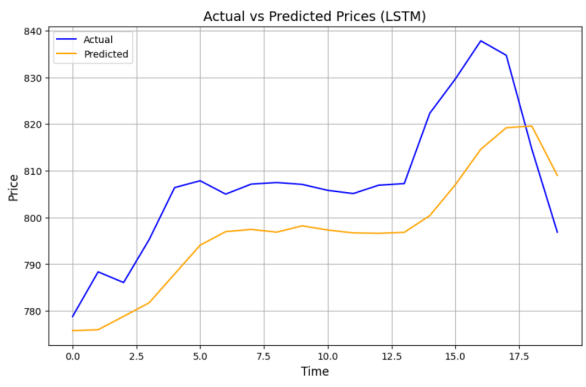

Figure 1. Actual vs Predicted Prices for the LSTM Model

The graph is the the Actual vs Predicted Prices for the LSTM model on the test dataset. The blue line represents the actual stock prices and the orange line is the predicted prices generated by the model. The LSTM model effectively captures the overall trend of the stock prices while demonstrating its ability to learn temporal patterns from the data. However, the model shows slight deviations during sharp peaks and troughs that indicating sensitivity to sudden fluctuations. Despite these issues the LSTM predictions closely follow the actual prices and also highlighting its suitability for time-series forecasting tasks.

2. **Vanilla RNN model:**

The Vanilla RNN model is a fundamental recurrent architecture used for time-series forecasting. It employs sequential data processing to capture temporal dependencies. This model includes SimpleRNN layers, dropout for regularization and a dense layer for final predictions. Despite being simpler than advanced architectures like LSTM it demonstrated the ability to learn patterns effectively.

a. **Model Architecture :**

* Recurrent Layers: Two SimpleRNN layers with units as the hyperparameter.

*Dropout Layers: To prevent overfitting dropout was applied after each recurrent layer.

*Dense Layer: A single dense layer was added to predict the next value in the sequence.

*Hyperparameter Tuning: Grid search was used to optimize the units, dropout rate, batch size, and number of epochs based on validation loss.

c. **Result and Analysis :**

Configuration	**Value**
Units	100
Dropout Rate	0.2
Batch Size	16
Epochs	50
Performance Metrics	
Mean Absolute Error (MAE)	4.94
Mean Squared Error (MSE)	44.87

Table 2. Configuration and Performance Metrics of the Best-Performing Vanilla RNN Model

The figure 2 illustrates the Actual vs Predicted Prices for the Vanilla RNN model on the test dataset. The blue line represents the actual stock prices and the orange line depicts the predicted prices. The Vanilla RNN effectively captures the temporal trends in the data. Compared to the LSTM the Vanilla RNN demonstrates slightly higher sensitivity to sudden fluctuations in prices as seen in the deviations near sharp peaks. However it performs well overall with low error metrics which highlighting its capability for simpler time-series tasks.

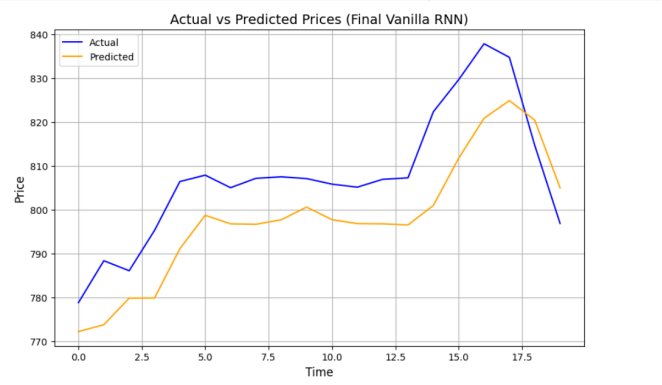

Figure 2. Actual vs Predicted Prices for the vanilla RNN Model

3. **GRU Model :**

The Gated Recurrent Unit (GRU) model was employed to capture sequential dependencies in the dataset while minimizing computational complexity. It used a stacked architecture with two GRU layers followed by dropout layers to prevent overfitting and a dense layer for regression output. The hyperparameter tuning was carried out for units, dropout rate, batch size, and epochs. The model demonstrated competitive performance in predicting stock prices.

a. **Model Architecture :**

* Input Layer: The input shape was designed to handle sequences of time-series data with features from the dataset.

* GRU Layer 1: A GRU layer with 100 units to capture sequential dependencies and temporal patterns from the input data. It was configured to return sequences for the next layer.

* Dropout Layer 1: A dropout layer with a rate of 0.3 to reduce overfitting by randomly deactivating some neurons during training. GRU Layer 2: A second GRU layer with 100 units for deeper learning of sequential features.

* Dropout Layer 2: Another dropout layer with a rate of 0.3, ensuring regularization at this layer.

* Dense Output Layer: A dense layer with a single neuron for the regression task, predicting the stock price for the next day.

Result and Analysis :

Hyperparameter	**Value**
Units	100
Dropout Rate	0.3
Batch Size	16
Epochs	50
Performance Metrics	
Mean Absolute Error (MAE)	7.89
Mean Squared Error (MSE)	85.19

Table 3. Hyperparameters and Performance Metrics for the GRU Model

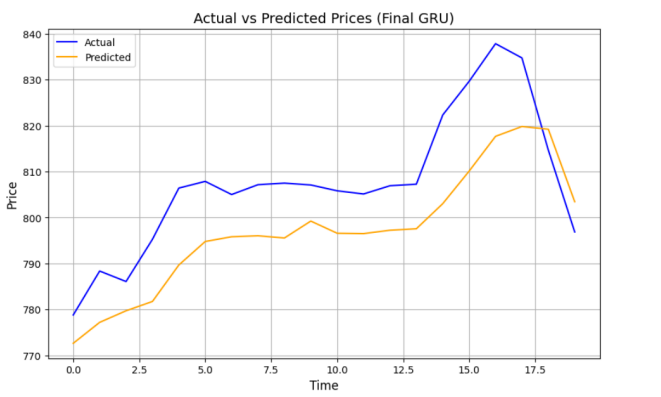

Figure 3. Actual vs Predicted Prices for the GRU Model

The graph above illustrates the Actual vs. Predicted Prices for the GRU model. The blue line represents the actual stock prices and the orange line denotes the predicted prices. The model closely follows the trends and patterns of the actual prices and showing its effectiveness in capturing temporal dependencies. Minor deviations in the prediction highlight areas for further improvement.

hyperref

References

References

[1] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997. DOI: 10.1162/neco.1997.9.8.1735

[2] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, *Empirical evaluation of gated recurrent neural networks on sequence modeling*, arXiv preprint arXiv:1412.3555, 2014. Available online: arxiv.org/abs/1412.3555

[3] I. Sutskever, J. Martens, and G. E. Hinton, *Generating text with recurrent neural networks*, Proceedings of the 28th International Conference on Machine Learning (ICML), pp. 1017–1024, 2011. Available online: proceedings.mlr.press/v15/sutskever11a.html

[4] Dr. Xinyu Wang, *Resources for Advanced Machine Learning*, The University of Adelaide. Available online: myuni.adelaide.edu.au/courses/94658/pages/resources

Link to Assignment : `https://github.com/Anusha0113/Deep-Learning-Fundamentals`