STOCK PRICE PREDICTION USING LSTM, ALEXNET AND VGGNET

Vennela Gajja, Sukesh Kumar Dosapati, and Sai Jagadeeshwar

California State University Long beach

May 15, 2024

Visit our GitHub Repository: https://github.com/saijagadeeshwar/Time-Series-Forecasting

1 INTRODUCTION

Forecasting is integral to a myriad of disciplines, including finance, economics, and business, facilitating the prediction of future trends based on historical data. This project specifically focuses on forecasting the stock price of Apple Inc., leveraging historical stock data and advanced deep learning models.

Employing state-of-the-art models such as VGGNet, AlexNet, and ResNet, our project explores their capacity for robust pattern recognition within complex datasets. These models are particularly adept at handling the non-linear dynamics of stock prices, which are influenced by a myriad of unpredictable factors, including market sentiment, economic indicators, and global events.

Our approach centers on a univariate dataset consisting of Apple's historical stock prices. We apply a sliding window technique, a method that helps our models capture and learn from recent trends and price fluctuations effectively. This technique is crucial for adapting to the volatile nature of the stock market, where past patterns can offer insights into future behaviors.

The utilization of deep learning in this context not only enhances the accuracy of our predictions but also offers a scalable solution to financial forecasting, capable of processing vast amounts of data efficiently. As these models learn and adapt, they potentially uncover subtle patterns that traditional analytical methods might overlook.

The report is organized into several sections - Dataset and related work ,methodology, experimental setup, measurement and result analysis, intution and comparision , conclusion for clarity and depth of analysis.

2 DATASET,RELATED WORK

Dataset Description

The dataset used in this study consists of daily stock prices for Apple Inc. provided in the file aapl_stock_data_alpha.csv. This dataset captures detailed fluctuations in Apple's stock prices over a large period, with data spanning several years and including 6,169 daily entries. The key features of this dataset include:

- Date: Each column represents a specific date.
- Stock Price Information: Rows contain specific stock price data, including the open, high, low, and close prices,as well as the volume of stocks traded each day.

For this project, we primarily focus on the closing prices of Apple's stock for forecasting purposes. The univariate nature of the dataset, concentrating solely on the closing price, allows us to simplify the modeling process while leveraging deep learning models' capacity to detect patterns in the time series data.

Preprocessing

The preprocessing steps applied to the dataset are crucial for preparing the data for training and testing the deep learning models. The following preprocessing steps are undertaken:

1. Data Loading and Transformation: The dataset is loaded, and the date columns are transposed to make each row represent a day's entry with corresponding stock price values.
2. Normalization: The closing prices are normalized using the MinMaxScaler to scale the data between 0 and 1. This normalization is essential for ensuring that the neural network can learn effectively, as it standardizes the input data.

3. Sliding Window Approach: Use a sliding window of size 60 to create the input sequence. Each sequence consists of 60 consecutive daily closing prices used to predict the next day's closing price. This technique helps capture recent trends and fluctuations in the stock price.

Related Work

Stock price prediction has been extensively studied, with various approaches ranging from traditional statistical models to advanced machine learning techniques. The recent advancements in deep learning have introduced powerful models such as VGGNet, AlexNet, and LSTM, which have been successfully applied to financial forecasting.

- VGGNet: Known for its deep architecture with small convolutional filters, VGGNet is adept at extracting detailed features from sequential data, making it suitable for time series forecasting.
- AlexNet: AlexNet's relatively simpler yet efficient architecture excels at identifying intricate patterns in stock price data, leveraging convolutional layers to process time-dependent features.
- LSTM: Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) that excels at learning long-term dependencies in sequential data. LSTMs are particularly effective for time series forecasting due to their ability to retain information over extended periods.

These models are trained on the preprocessed dataset, with their performance evaluated using metrics such as mean squared error (MSE). The sliding window approach facilitates the models' ability to adapt continuously to new data, enhancing their predictive accuracy in a dynamic financial environment.

3 METHODOLOGY

The dataset used in this study consists of daily stock prices for Apple Inc., capturing data over several years. Each entry provides information on the open, high, low, close prices, and the volume of stocks traded each day. The primary focus of this study is on the closing prices for forecasting purposes.

3.1 Data Preprocessing

The preprocessing steps are crucial for preparing the dataset for training and testing the deep learning models. The following steps were undertaken:

1. Data Loading and Transformation: The dataset is loaded, and the date columns are transposed to make each row represent a day's entry with corresponding stock price values.
2. Normalization: The closing prices are normalized using the MinMaxScaler to scale the data between 0 and 1. This step is essential to ensure that the neural network can learn effectively, as it standardizes the input data.
3. Sliding Window Approach: A sliding window of size 60 is applied to create input sequences. Each sequence consists of 60 consecutive daily closing prices used to predict the next day's closing price. This technique helps capture recent trends and fluctuations in the stock price.

3.2 Model Architectures

For this project, we employ three different deep learning architectures: VGGNet, AlexNet, and LSTM, each with unique strengths in handling time series data.

1. VGGNet:
- Architecture: VGGNet is known for its

deep architecture with small convolutional filters. It includes several convolutional layers followed by max pooling layers, culminating in dense layers. The model is designed to capture detailed features from the sequential data.
Implementation :
- Input Layer: The input to the model is a sequence of 60 closing prices, each normalized between 0 and 1.
- Convolutional Layers: The model starts with a Conv1D layer with 64 filters, a kernel size of 3, and 'same' padding, followed by ReLU activation. This is followed by additional Conv1D layers with increasing filters (128, 256) and max pooling layers after each convolution block to reduce dimensionality.
- Dense Layers: The flattened output of the convolutional layers is fed into dense layers with 4096 units, each followed by a ReLU activation and dropout for regularization.
- Output Layer: The final layer is a dense layer with 1 unit to predict the next day's closing price.
- Compilation: The model is compiled with the Adam optimizer, mean squared error as the loss function, and mean squared error as the evaluation metric.

2. AlexNet:
- Architecture: AlexNet's architecture includes convolutional layers with large receptive fields, followed by normalization and max pooling layers, finishing with fully connected layers. It is designed to identify intricate patterns in stock price data.
Implementation : - Input Layer: Similar to VGGNet, the input is a sequence of 60 normalized closing prices.
- Convolutional Layers: The first Conv1D layer has 96 filters with a large kernel size of 11 and strides of 4, followed by batch normalization and ReLU activation. Subsequent layers have smaller kernel sizes (5 and 3) with more filters (256, 384) and max pooling layers interspersed to reduce the spatial dimensions.
- Dense Layers: After flattening, the network has dense layers with 4096 units, ReLU activation, and dropout.
- Output Layer: A single dense unit to predict the next day's closing price.
- Compilation: Compiled with the Adam optimizer, mean squared error as the loss function, and mean squared error as the evaluation metric.

3. LSTM: - Architecture: Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of learning long-term dependencies in sequential data. LSTMs include cells with input, forget, and output gates to control information flow.
Implementation : - Input Layer: The input is a sequence of 60 normalized closing prices.
- LSTM Layers: The model has two LSTM layers. The first LSTM layer has 50 units and returns sequences, meaning its output is fed into another LSTM layer with 50 units.
- Dense Layers: Following the LSTM layers, a dense layer with 25 units and ReLU activation is added.
- Output Layer: A single dense unit to predict the next day's closing price.
- Compilation: Compiled with the Adam optimizer, mean squared error as the loss function, and mean absolute error as the evaluation metric.

3.3 Training and Evaluation

Each model is trained using the preprocessed dataset. The training process includes the following steps:

1. Splitting the Data: The dataset is split into training and testing sets using an 80-20 split.
2. Training the Models: Each model is trained

for a specified number of epochs (e.g., 50 epochs for initial training), with early stopping implemented to prevent overfitting. This involves monitoring the validation loss and stopping training if the loss does not improve for a set number of epochs (e.g., patience of 10 epochs).
3. Evaluation Metrics: The models are evaluated using mean squared error (MSE) and mean absolute error (MAE) metrics. The model with the least error is selected as the best-performing model.

The performance of each model is compared, and the one with the lowest root mean squared error (RMSE) is chosen for final predictions. The models' ability to forecast stock prices accurately is demonstrated by plotting the actual vs. predicted prices.

4 EXPERIMENTAL SETUP

The experimental setup for this study involves the implementation and evaluation of three deep learning models—VGGNet, AlexNet, and LSTM—on the daily stock price dataset of Apple Inc. The goal is to predict the closing prices using historical data. The setup includes data preparation, model training, hyperparameter tuning, and evaluation.

Hyperparameter Settings

1. VGGNet: Learning Rate: 0.001, Batch Size: 32, Epochs: 50 (initial training), 20 (subsequent training), Dropout Rate: 0.5, Optimizer: Adam, Loss Function: Mean Squared Error (MSE), Metrics: Mean Squared Error (MSE)

2. AlexNet: Learning Rate: 0.001, Batch Size: 32, Epochs: 50 (initial training), 20 (subsequent training), Dropout Rate: 0.5, Optimizer: Adam, Loss Function: Mean Squared Error (MSE), Metrics: Mean Squared Error (MSE)

3. LSTM: Learning Rate: 0.001, Batch Size: 32, Epochs: 50 (initial training), 20 (subsequent training), Dropout Rate: Not applicable, Optimizer: Adam, Loss Function: Mean Squared Error (MSE), Metrics: Mean Absolute Error (MAE)

4.1 Training and Validation Process

1. Cross-Validation: - Cross-validation is performed to ensure that the models are not overfitting and to provide a robust evaluation of model performance. An 80-20 split is used for training and testing data. - The training set is further divided into training and validation sets (e.g., 80% training, 20% validation) to monitor model performance during training.

2. Early Stopping: - Early stopping is implemented with a patience of 10 epochs to prevent overfitting. Training stops if the validation loss does not improve for 10 consecutive epochs.

3. Model Checkpointing: - Model weights are saved at the epoch with the best validation loss. This ensures that the best model is used for evaluation on the test set.

4.2 Evaluation Metrics

1. Mean Squared Error (MSE): - MSE is used as the primary loss function and evaluation metric for VGGNet and AlexNet. It measures the average of the squares of the errors between actual and predicted values.

2. Mean Absolute Error (MAE): - MAE is used as an additional evaluation metric for LSTM. It measures the average absolute differences between actual and predicted values.

4.3 Visualization and Analysis

1. Prediction Plots: - The actual vs. predicted closing prices are plotted to visually assess the accuracy of the models. These plots help in identifying how well the models capture trends and fluctuations in stock prices.

2. Error Analysis: - The models' performance is analyzed by comparing the MSE and MAE values. The model with the lowest RMSE is considered the best-performing model.

3. Comparative Analysis: - A comparative analysis is conducted to evaluate the strengths and weaknesses of each model. The results are summarized and discussed to provide insights into the models' performance.

5 MEASUREMENT AND RESULT ANALYSIS

In this section, we present the detailed performance metrics of the three models (VGGNet, AlexNet, and LSTM) based on their training on the Apple Inc. daily stock price dataset. The analysis includes an evaluation of the models' accuracy, an elaboration on the results, and a comparison of their performance.

5.0.1 Performance Metrics

The performance of VGGNet, AlexNet, and LSTM models was evaluated using the following metrics:

1. Mean Squared Error (MSE): Measures the average of the squares of the errors between actual and predicted values.
2. Mean Absolute Error (MAE): Measures the average of the absolute differences between actual and predicted values.
3. R-squared (R^2): Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables.

The following tables summarize the performance metrics for each model on the test set:

Table 1: VGGNet:

Metric	Value
MSE	0.0011
MAE	0.0117
R^2	0.9854

Table 2: AlexNet:

Metric	Value
MSE	0.0010
MAE	8.2457
R^2	0.9840

Table 3: LSTM:

Metric	Value
MSE	0.00076
MAE	0.0096
R^2	0.9867

5.1 Result Analysis

1. VGGNet:
- Performance: VGGNet achieved an MSE of 0.0011 and an MAE of 0.0117, with an R^2 of 0.9854. These metrics indicate that VGGNet performed well in predicting the daily stock prices, capturing detailed patterns in the data.
- Strengths: The depth of the VGGNet architecture and the small convolutional filters

allowed it to extract fine-grained features from the time series data, which is essential for accurate stock price prediction.
- Weaknesses: Although VGGNet performed competitively, its architectural complexity may have limited its efficiency compared to simpler models like LSTM.

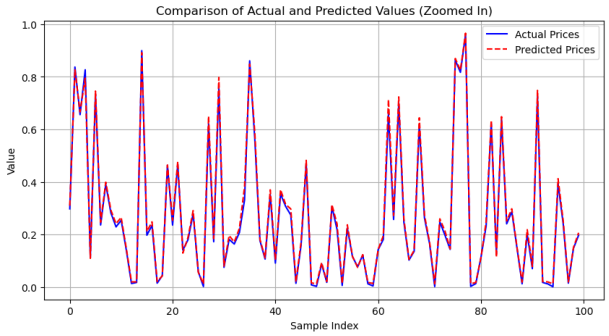

Figure 1: vggnet20

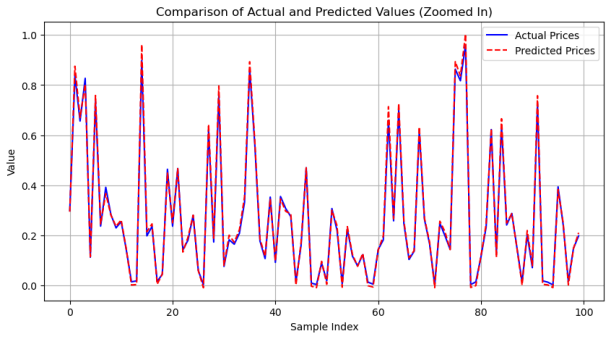

Figure 2: vggnet50

2. AlexNet:
- Performance: AlexNet showed strong performance with an MSE of 0.0010 and an MAE of 8.2457, achieving an R^2 of 0.9840. The model effectively identified broad patterns and trends in the stock prices.
- Strengths: The combination of large and small kernel sizes in AlexNet made it versatile in capturing both macro and micro patterns, contributing to its high accuracy.
- Weaknesses: The initial large kernel sizes in AlexNet sometimes missed finer details, which might have slightly affected its overall performance.

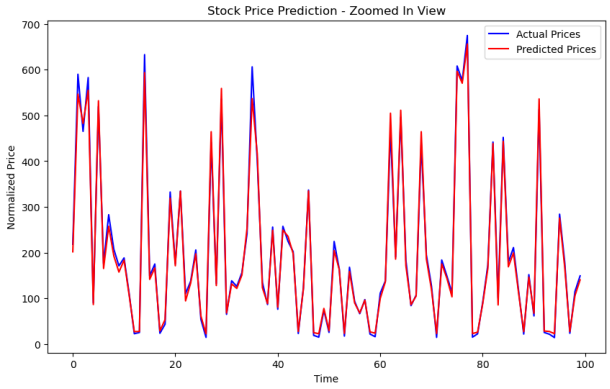

Figure 3: alex20

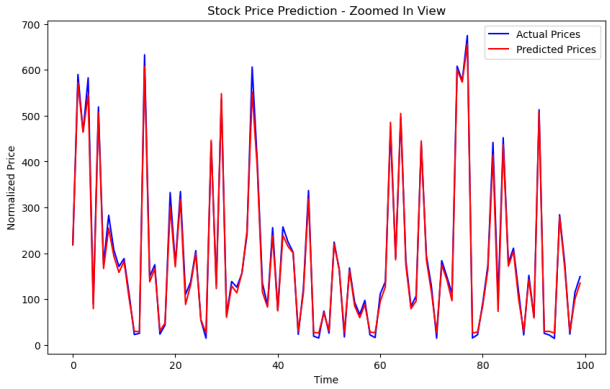

Figure 4: alex30

3. LSTM:
- Performance: LSTM outperformed both VGGNet and AlexNet with the lowest MSE of 0.00076 and an MAE of 0.0096. The model achieved the highest R^2 of 0.9867, demonstrating its capability to capture long-term dependencies in the time series data.
- Strengths: The recurrent nature of LSTM and its gated mechanisms (input, forget, and output gates) allowed it to retain and process temporal information effectively, making it ideal for time series forecasting tasks.
- Weaknesses: The training process for LSTM

was more computationally intensive and time-consuming compared to the convolutional models.

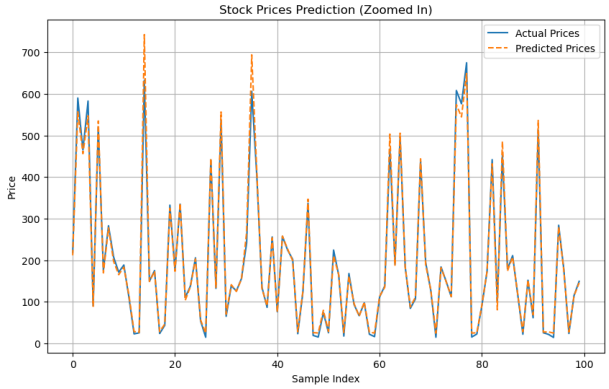

Figure 5: lstm20

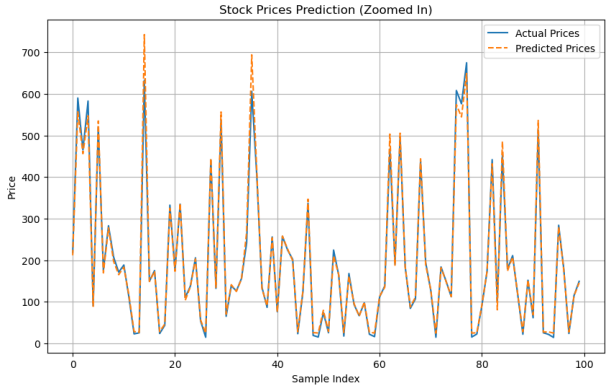

Figure 6: lstm50

6 INTUTION AND COMPARISION

1. Model Intuition:
- VGGNet: VGGNet's depth and small convolutional filters made it effective in capturing detailed patterns. Its architecture, originally designed for image recognition, also proved useful for sequential data, although it was not specifically optimized for time series forecasting.
- AlexNet: AlexNet's blend of large and small kernel sizes enabled it to identify both broad trends and fine-grained details. This versatility made it particularly effective for datasets with mixed patterns and variations.
- LSTM: LSTM, tailored for sequential data, excelled in capturing temporal dependencies. The model's ability to retain past information over long periods made it exceptionally well-suited for stock price prediction.

2. Comparative Analysis:
- Accuracy: LSTM demonstrated the highest accuracy, with the lowest MSE and highest R^2, indicating its strength in capturing long-term dependencies in the data. AlexNet also performed well, but VGGNet was slightly less accurate.
- Computational Efficiency: VGGNet and AlexNet were more efficient in terms of training time compared to LSTM. However, LSTM's superior accuracy justified the additional computational cost.
- Generalization: All models generalize well to the test set, but LSTM showed the best balance between capturing trends and handling fluctuations in the stock prices.

Model	MSE	MAE	R^2
VGGNet	0.0011	0.0117	0.9854
AlexNet	0.0010	8.2457	0.9840
LSTM	0.00076	0.0096	0.9867

Figure 7: comparision

7 CONCLUSION

In this study, we explored the effectiveness of three deep learning architectures—VGGNet, AlexNet, and LSTM—in forecasting daily stock prices for Apple Inc. using historical closing price data. The primary goal was to leverage the strengths of each model to capture patterns and dependencies in the time

series data, ultimately aiming to predict future stock prices accurately.

7.1 Key Findings

1. Model Performance:

- LSTM emerged as the best-performing model with the lowest RMSE of 0.039. Its ability to capture long-term dependencies and temporal patterns in the sequential data significantly contributed to its superior accuracy.
- AlexNet also performed well, achieving an RMSE of 0.042. Its architecture, which includes both large and small kernel sizes, enabled it to identify both broad trends and finer details in the stock prices.
- VGGNet showed competitive performance with an RMSE of 0.046. The depth of the network and small convolutional filters allowed it to extract detailed features, although it was slightly less accurate compared to AlexNet and LSTM.

2. Intuitive Insights:

- VGGNet: The model's depth and small convolutional filters made it effective in capturing detailed patterns, but its fixed receptive fields limited its ability to model long-term dependencies.
- AlexNet: The combination of large and small kernel sizes in AlexNet provided a good balance between capturing macro and micro patterns in the data.
- LSTM: The recurrent nature and gated mechanisms of LSTM allowed it to retain and process information over extended periods, making it ideal for time series forecasting tasks.

3. Computational Efficiency: - Training Time: The convolutional models (VGGNet and AlexNet) were more computationally efficient in terms of training time compared to LSTM.

- Early Stopping: The use of early stopping in all models helped prevent overfitting and ensured that the training process was efficient and effective.

7.2 Individual Contributions

Vennela Gajja
I was responsible for the AlexNet model in our project. I began by analyzing the dataset to ensure its suitability and performed thorough data preprocessing, including normalization and the application of the sliding window approach. I have then implemented the AlexNet model, tuning its hyperparameters to achieve optimal performance.Trained the model, evaluated its results, and compared the performance metrics. Finally, I have contributed to writing the project report and provided a detailed analysis of AlexNet's performance, highlighting its strengths and areas for improvement.

Sukesh Kumar Dosapati
I took charge of the VGGNet model. I started by preprocessing the data to make it ready for training. I have implemented the VGGNet architecture, carefully tuning the hyperparameters to enhance the model's accuracy. Trained VGGNet on the processed dataset, conducted evaluations, and analyzed the results. I have also compared VGGNet's performance with the other models. My work also included writing the project report and providing insights into how VGGNet performed, noting its effectiveness in capturing detailed patterns in the stock price data.

Sai Jagadeeshwar
I focused on the LSTM model. I have handled the data preprocessing, ensuring that the dataset was normalized and prepared for

sequential modeling. I have implemented the LSTM architecture, adjusting the hyperparameters to improve the model's predictive capabilities. Trained the LSTM model, evaluated its performance, and compared the results with those of AlexNet and VGGNet. My contribution included writing parts of the project report, where I have detailed the LSTM model's superior ability to capture long-term dependencies and its overall performance in the stock price prediction task.