Stock Prediction via Fundamental Financial Data using Deep Learning Techniques

Manas Manoj Bedekar
Department of Electrical and Computer Engineering
McGill University
Montreal, QC Canada
manas.bedekar@mail.mcgill.ca

Qingnan Li
Department of Electrical and Computer Engineering
McGill University
Montreal, QC Canada
qingnan.li@mail.mcgill.ca

Sansitha Panchadsaram
Department of Electrical and Computer Engineering
McGill University
Montreal, QC Canada
sansitha.panchadsaram@mail.mcgill.ca

Tony Xu
Department of Electrical and Computer Engineering
McGill University
Montreal, QC Canada
tony.xu@mail.mcgill.ca

Abstract

In today's world of valuable investments, financial analysis has become more challenging. Predicting the stock market is one of the most difficult tasks in the field of quantitative analysis. The prediction is influenced by numerous factors, including company fundamentals, geopolitical events, rational and irrational behavior, investor sentiment, market rumors, and more. These factors contribute to the volatility of stock prices, making them highly challenging to predict with a high degree of accuracy. In this project, we present a generalized deep learning-based model that can predict the price of a stock for the next quarter based on the fundamental financial information of a company. In addition, we provide a comprehensive analysis of various deep learning models' performance using Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) as evaluation metrics. According to our results, the proposed deep learning model based on Convolutional Neural Network (CNN), Bidirectional Long Short-Term Memory (BiLSTM), and attention mechanism outperformed by achieving a RMSE of \$19.54, and a MAPE of 16.8%. Further analysis shows that this model has shown consistent performance for different sectors and managed to generalize the stock predictions and give valuable insights to long-term investors.

1 Introduction

The stock market serves as a pivotal hub where investors come together to trade various financial instruments such as stocks, bonds, and options, to facilitate the flow of capital throughout the economy [1]. At its core, the stock market enables companies to raise funds for their operations, expansion, and innovation by issuing shares of ownership to investors. In return, these investors gain a stake in the company and the potential for financial gains through dividends and capital appreciation [1].

Investors base their decision to invest in a company in 3 main ways: predicting the future cash generated by the business and discounting it back to the present, calculating the liquidation value from selling all the assets in the business, and looking at the prices of comparable companies and recent acquisitions [2]. To select a prospective investment, investors look at companies' financial data, which is published quarterly. However, valuations are a time-consuming and tedious process with basic analysis taking about 4 hours [3]. Considering there are over 55,000 publicly listed companies globally, it is impossible for individual investors to analyze the entire market. Models predicting stock prices would enable investors to make well-informed financial choices.

Prior to the use of machine learning algorithms, researchers used different statistical and econometric methods to design stock price prediction models. However, traditional statistical and econometric models mostly rely on linearity. Since financial data is nonlinear, in order to use the linear models prominent in statistics and econometrics to effectively forecast financial data, nonlinear models must be converted to linear ones [4]. Furthermore, machine learning techniques have had some success predicting short-term stock price movements, but struggle in making accurate longer-term predictions based on companies' financial data [5], [6], [7].

Neural networks are more advantageous than traditional methods because they can learn patterns and relationships directly from the data that they are trained on and they are capable of adjusting their internal parameters and structure based on changes in the input data or the task at hand [4]. Therefore, neural networks possess greater capability to analyze imprecise and noisy data and they are widely employed for predicting time series. More specifically, deep learning models are designed to handle extensive amounts of nonlinear data. These networks are adept at capturing intricate nonlinear connections within financial data, which often contain complex, missing and ambiguous information [4].

Existing deep learning models often perform well in predicting short-term stock prices. Namdari and Li applied three-hidden layer Multilayer Perceptron (MLP) to achieve 66% directional accuracy [8]. Yadav et al. implement a Long Short-Term Memory (LSTM) with four companies resulting in p-values ranging from 0.002 to 0.898 [9]. Lee et al. applied an attention-based Bidirectional Long Short-Term Memory (BiLSTM) to achieve 69% accuracy predicting stock trend [10]. Our goal is to show that sufficiently accurate longer-term predictions can be made based on companies' financial data using deep learning models.

More specifically, the problem formulation revolves around the question: *Can we predict the next-quarter price based on historical fundamental financial data with sufficient accuracy to screen for profitable investment prospects?*

Two constraints were considered based on the application and problem at hand: the performance metric and its margin of error as well as the exploration of the impact of different normalization techniques.

Firstly, the performance metric plays a crucial role in assessing the efficacy of stock price prediction models. To establish a clear margin of error, the model's predicted stock value must have a Mean Absolute Percentage Error (MAPE) of no more than 20% compared to the actual stock value. This threshold of error was defined based on the fact that most deep learning models have been shown to perform within 20% of the true value of technical data [11], [12], [13]. This margin of error was used even though it describes the performance on technical data because most existing deep learning models that predict stock prices using fundamental data do not perform well. Ensuring that the model's predictions remain within this threshold not only instills confidence in the reliability of the model but also facilitates informed decision-making for investors, allowing them to navigate the complexities of the market with greater certainty.

Secondly, normalization of the data allows its patterns to be effectively learned by the model. This project aims to determine the best normalization technique among Min-max and Z-score that maximizes the predictive performance. Moreover, the exploration extends beyond simply assessing the impact of normalization techniques at a singular level; rather, it delves into the nuanced implications of normalization at various denominators, including sector-wise, company-wise, and encompassing the entire stock market. This comprehensive analysis acknowledges the inherent diversity and complexity within financial data, recognizing that different subsets of data may necessitate specific normalization strategies to optimize predictive performance. By shedding light on the relationship between normalization techniques and predictive performance across different levels of denominators, this project aims to provide valuable insights into optimizing model performance in the context of stock price prediction.

Together, these constraints emphasize the multiple challenges inherent in developing robust and reliable predictive models for stock price prediction. By addressing these constraints, this project plans to advance the understanding of deep learning approaches in financial forecasting, paving the way for more accurate and informed investment decision-making. Furthermore, the exploration of normalization techniques not only contributes to methodological advancements in model development but also underscores the importance of meticulous data preprocessing in achieving optimal predictive performance.

2 Dataset Preparation

2.1 Dataset and Resources

In order to design a generalized deep learning model to predict the average stock price for the next quarter, we needed raw fundamental financial data from various companies.

We searched on open-source platforms like Kaggle, GitHub, and DoltHub for a suitable fundamental financial dataset but found none that included consolidated fundamental financial indicators. These indicators are crucial for assessing a company's or financial asset's health and performance. Some common fundamental financial indicators include revenue, earnings, profit margin and debt-to-equity ratio. The majority of open-source datasets focused on technical indicators, containing features such as open prices, close prices, volume, 52-week highs, and 52-week lows. Such technical indicators help predict short-term stock prices but fail to predict long-term investment insights [14]. Previous works [14] [15] have shown that these datasets can create a deep learning model for predicting the price of a specific stock, and do not have an ability to generalize across different stocks. Since we did not find any promising dataset, we started looking into company-wise financial reports.

Ultimately, using a private token API provided by QuickFS [1], we collected the fundamental financial information of 2,418 different companies listed on the NYSE and NASDAQ stock exchanges. The retrieved data had 210 features based on balance sheets, income statements, cash flow, assets, liabilities, returns on investment, market capital, per share revenue, etc. Table 1 provides more information about the data source. Now, to handle and train models with such huge data, we needed a dedicated GPU. For this project, we used NVIDIA RTX 4080, 16GB VRAM GPU to train and evaluate the diverse models.

Table 1: Dataset Details

	Description
Dataset Source	https://quickfs.net/
Stock Exchanges	NYSE and NASDAQ
Total Companies	2,418
Total Sectors	12
Total Features	210
Filtered Companies	1,658
Selected Features	108
Time Span	20 years (2001 - 2021)
GPU	NVIDIA RTX 4080, 16GB VRAM

2.2 Preprocessing

Figure 1 depicts the data preprocessing pipeline. For basic screening, we selected only companies that had historical financial data for more than 20 years, which is equivalent to 80 quarters. Then, we dropped around 100 features, which contained mostly missing values or were highly intercorrelated. This filtered data was divided into three parts: 1. Training Data, spanning 56 Quarters, 2. Validation Data, comprising 14 Quarters, 3. Testing Data, consisting of 10 Quarters.

The scales of the features vary dramatically, which impacts the models' ability to generalize to the entire market. For example, the revenue ranges from $\$-8.63 \times 10^{-10}$ to $\$1.64 \times 10^{11}$ while the payout ratio ranges from -450 to 1880. Consequently, normalization was necessary to ensure the use of a uniform scale, to improve convergence, and to prevent numerical instability. We experimented with applying Min-Max normalization and Z-score normalization on a company-wise, sector-wise, and overall dataset basis.

[1] https://quickfs.net/

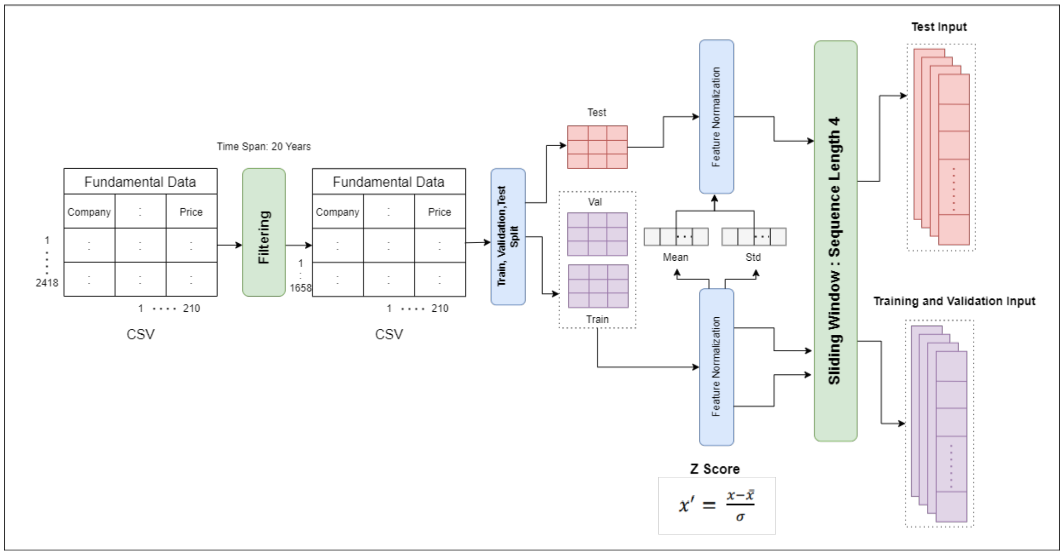


Figure 1: Data Preprocessing Pipeline

While performing normalization, we calculated the means and standard deviations on the training dataset and applied these values to the validation and test data to avoid data leakage.

This normalized data was then passed through a sliding window mechanism with a sequence length of 4 and an overlap setting of 3. This mechanism involves dividing the data into overlapping segments of a fixed size. In this case, the sequence length is set to 4, which means each segment contains 4 time points. The overlap setting determines how much each window overlaps with the previous one. With an overlap setting of 3, each window overlaps with the previous one by 3 time points. The reason behind using this sliding window approach is that this technique can extract meaningful sequential insights, and facilitate more efficient computational processing.

3 Methodology

The task involves evaluating deep learning models' ability to predict next-quarter stock prices using historical fundamental financial data. With the goal of identifying profitable investments, the challenge is to train and assess different deep learning architectures on comprehensive datasets of publicly listed companies' financial records. Our first constraint requires that predictive models achieve a Mean Absolute Percentage Error (MAPE) <= 20%, ensuring a reasonable margin of error in predictions. Additionally, the exploration of different normalization techniques, such as Min-max and z-score normalization, is crucial for enhancing predictive performance. Through rigorous experimentation and evaluation, the goal is to ascertain the viability of deep learning approaches in providing reliable forecasts for guiding investment decisions in financial markets.

3.1 Overview of Models' Components

3.1.1 CNN

CNN is a type of feedforward neural network that has been shown to demonstrate impressive performance in both image processing and natural language processing (NLP) owing to its distinctive architecture and the mechanisms of convolutional and pooling layers [16]. Through local perception and weight sharing, CNN reduces feature dimensions, thereby enhancing training and inference efficiency. Following the convolutional operation in each layer, local features are extracted, and subsequent pooling layers further reduce feature size while selecting the most salient local information [16].

3.1.2 LSTM

LSTM is a network designed to address the issue of vanishing gradient encountered in Recurrent Neural Network (RNN), which is commonly used to capture long-term dependencies between different time steps [17]. The LSTM memory cell comprises three components: the forget gate, the input gate, and the output gate. The forget gate incorporates the cell state C_{t-1} and the output information h_{t-1} from the previous time step by concatenating them and feeding them into a feedforward network [17]. The output value of the forget gate is computed as shown in the formula below:

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

The output value and the input value of the current time are concatenated and fed into the input gate, which calculates the candidate cell state for the current time step. The cell state is then updated by considering both the current cell state, the output of the forget gate for the current time step, and the cell state from the previous time step.

The concatenation of the output from the previous time step and the input from the current time step is further forwarded to the output gate of the memory cell. Here, the output state and the cell state of the current time step are utilized to derive the embedding for the current moment. This embedding is subsequently employed for computations in the next time step, following the same procedure described above [17].

3.1.3 Attention Mechanism

The Attention Mechanism (AM) computes the probability distribution of attention, enabling the selection of significant information from a range of data while disregarding unimportant information. The similarity between the query (output features) and the key (input features) is calculated with a linear transformation using formula:

$$s_t = W_h h_t + b_h$$

Those similarity logits are used to obtain the attention score corresponding to each time step using softmax operation [18]:

$$a_t = \frac{exp\left(s_t\right)}{\sum_t exp\left(s_t\right)}$$

The final embedding $h_{t,final}$ at the predicting time step is computed through a weighted summation of the attention scores multiplied by the embeddings at corresponding time steps, as illustrated in the formula below:

$$h_{t,final} = \sum_t a_t h_t$$

3.2 Baseline model: Multilayer Perceptron

The MLP architecture that we implemented first consists of a layer that flattens the sequential input data into an one-dimensional array. As shown in Figure 5 in Appendix B, this flattened input is then fed into a series of fully connected layers, forming the hidden layers of the network. The hidden layers progressively decrease in size, starting with 256 neurons for the first layer and reducing to 128, 64, 32 and 16 for the subsequent layers. Each fully connected layer applies a linear transformation followed by a Rectified Linear Unit (ReLU) activation function, which introduces non-linearity to capture complex patterns in the sequential data. Dropout layers are added after each hidden layer with a dropout probability of 0.1, which serves as a regularization technique that prevents overfitting by randomly deactivating neurons during training. The last fully connected layer outputs the next-quarter stock price predictions.

MLP was chosen as a baseline model for predicting next-quarter stock prices because even though it lacks the ability to capture temporal dependencies in sequential data, it can still establish a baseline level of performance before exploring more complex architectures.

3.3 LSTM model

As shown in Figure 6 in Appendix B, the LSTM architecture that we implemented consists of an LSTM layer followed by a fully connected layer. The LSTM layer takes input sequences and learns to capture temporal dependencies through its recurrent structure, which includes gates to regulate information flow and prevent the vanishing gradient problem. The hidden states of the LSTM are initialized with zeros and updated through each time step of the input sequence. The final hidden state represents the learning representation of the input sequence, which is passed through a fully connected layer to predict the next-quarter stock prices.

LSTM was chosen as an alternative solution because existing literature show that LSTM networks are well-suited for capturing the temporal dependencies present in sequential data, making them effective for time series prediction tasks, such as stock price forecasting [19], [20], [21].

3.4 LSTM-AM model

The LSTM-AM model extends the previously defined LSTM architecture by incorporating an attention mechanism, which would intuitively improve its ability to capture and emphasize relevant information in the input sequence. As shown in Figure 7 in Appendix B, similar to the LSTM model, the LSTM-AM model consists of an LSTM layer followed by a fully connected layer. However, in addition to the LSTM layer, it includes an attention layer, which applies a linear transformation followed by a softmax activation function. During the forward pass, the LSTM processes the input sequence to generate hidden states. The attention layer computes attention weights for each hidden state, indicating the importance of each time step's information. These attention weights are then used to compute a context vector, which represents a weighted sum of the LSTM outputs, with higher emphasis on the more informative time steps. Finally, the context vector is passed through the fully connected layer to predict the next-quarter stock prices.

3.5 BiLSTM-AM model

The BiLSTM-AM model enhances the LSTM-AM architecture by introducing bidirectional LSTM layers to capture both past and future dependencies in the input sequence. As shown in Figure 8 in Appendix B, the model consists of two birectional LSTM layers, each followed by a dropout layer for regularization. The bidirectional LSTM layers process the input sequence bidirectionally, allowing the model to leverage information from both past and future time steps. The output of the bidirectional LSTM layers is then fed into an attention mechanism, which computes attention weights indicating the importance of each time step's information. Similar to the LSTM-AM model, these attention weights are used to compute a context vector, representing a weighted sum of the LSTM outputs with higher emphasis on more informative time steps. Finally, the context vector is passed through a linear layer to predict next-quarter stock prices.

3.6 CNN-BiLSTM-AM model

The parameter settings are presented in Table 5 (Appendix A.1). The CNN-BiLSTM-AM architecture is designed with a 1D convolutional filter to capture local information from input features at each time step, succeeded by a pooling layer to extract salient features. To mitigate overfitting, a dropout layer is strategically inserted after max pooling. The resulting features are then passed through two BiLSTM layers to capture temporal dependencies, with an additional dropout operation following the second BiLSTM layer. Attention weights for each time step are computed using a linear layer followed by a softmax operation. The final output embedding is calculated as the weighted sum of embeddings at each time step, achieved by multiplying the attention weights with the corresponding embeddings, followed by a fully connected layer [22]. The demonstration of the architecture is shown in Figure 2:

3.7 Evaluation of Models

Due to the inherent challenges of applying k-fold cross-validation to temporally dependent data, we refrain from its use in our evaluation. While the Root Mean Squared Error (RMSE) serves as the primary evaluation metric across all investigated models, its sole use may lack persuasiveness. For instance, an RMSE of \$10 with a price range of \$100 holds different implications compared to the same RMSE with a wider price range. To address this concern, we complement our evaluation with the Mean Absolute Percentage Error (MAPE), which offers insights into the percentage error between the actual and predicted stock price.

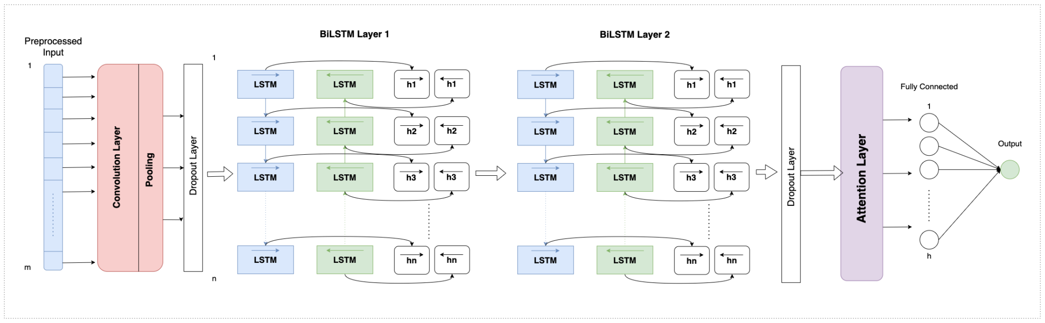


Figure 2: CNN-BiLSTM-AM Model Architecture

To optimize the performance of our deep learning model, we undertake extensive hyperparameter tuning. This process entails systematic variations of hyperparameters such as learning rate, batch size, dropout probability, hidden state dimension, and others. The optimal combination of hyperparameters is determined through a grid search technique, ensuring selection of the configuration that yields the lowest loss on the validation set. Notably, the test set remains unseen by the model during both training and validation phases.

4 Results

To assess the impact of normalization techniques on the models' performance measured using MAPE, we used Min-Max and Z-score normalization techniques. The goal of this assessment is to show whether our design achieves one of the outlined design constraints. During the preprocessing stage, the data was normalized in three different ways: company-wise, sector-wise and over all the stocks. Then, the stock price predictions were evaluated against the actual price predictions using the MAPE evaluation metric. As shown in Table 2, the CNN-BiLSTM-AM model achieved a MAPE of 22% and 16.8% when Min-Max and Z-score normalization were applied company-wise, respectively, representing the lowest MAPE among all compared models. Furthermore, when considering sector-wise normalization, the CNN-BiLSTM-AM model continued to outperform other architectures, achieving a MAPE of 33% and 29%, respectively, for Min-Max and Z-score normalization. Similarly, in the context of the entire market, the CNN-BiLSTM-AM demonstrated robust performance with a MAPE of 44% and 36% for Min-Max and Z-score normalization, respectively. Notably, according to these results, Z-score normalization applied company-wise gave the lowest MAPE as shown by the mean MAPE, highlighting its effectiveness in improving model accuracy in this specific scenario. Thus, Z-score normalization applied company-wise was selected as the normalization technique used during the preprocessing stage for the final CNN-BiLSTM-AM model.

Table 2: Impact of Normalization Techniques on Model Performance in terms of MAPE

	Company-wise		Sector-wise		Entire Market	
	Min-Max	Z-Score	Min-Max	Z-Score	Min-Max	Z-Score
MLP	36%	30%	43%	39%	52%	49%
LSTM	28%	22%	39%	36%	48%	43%
LSTM-AM	26 %	21%	38%	31%	47%	37%
BiLSTM-AM	26 %	17.2%	36%	30%	44%	36%
CNN-BiLSTM-AM	22%	16.8%	33%	29%	44%	36%
Mean MAPE	27%	**21%**	38%	33%	47%	40%

To validate our second design constraint, the RMSE and MAPE performance of all the implemented models were compared. Table 3 shows the obtained RMSE and MAPE results of our different models, which were computed over all the stocks in the dataset. As observed, the CNN-BiLSTM-AM model outperforms the other models, with a RMSE value of $19.54 and a MAPE of 16.8%. By achieving the lowest MAPE and RMSE, the CNN-BiLSTM model indicates better performance since it signifies smaller errors between predicted and actual values. Our results show that most models achieve the constraint that their predicted stock value must have a MAPE of no more than 20% compared to the actual stock value. More specifically, the BiLSTM-AM and CNN-BiLSTM-AM models achieve a MAPE that is less than 20%, indicating their ability to accurately predict the stock prices with a small margin of error. Furthermore, the CNN-BiLSTM-AM model achieved the lowest RMSE value, indicating that its predictions have the smallest average deivation from the actual stock prices among all models. Similarly, the CNN-BiLSTM-AM model also achieves the lowest MAPE of 16.8%, indicating the smallest average percentage difference between its predictions and the actual values, highlighting its superior accuracy compared to the other models.

Figure 3 and Figure 4 show the training and validation loss plots of the implemented models. It is observed that both losses for all models decrease initially as they learn patterns in the data. However, it is evident that the baseline MLP model overfits after only 10 epochs, highlighted by the fact that training loss is much lower than the validation loss. LSTM and LSTM-AM outperformed MLP, yet failed to address the issue of overfitting. Both LSTM and LSTM-AM showed signs of overfitting after only 20 epochs. However, in contrast, the BiLSTM-AM and CNN-BiLSTM-AM models effectively narrowed the margin between training and validation loss by the 50th epoch. This confirmed that the BiLSTM component in the aforementioned models played a major role in facilitating more robust learning and generalization.

Table 3: RMSE and MAPE Performance Comparison of Different Models

Model	RMSE ($)	MAPE (%)
MLP	34.71	30.2
LSTM	22.02	22.0
LSTM-AM	23.73	21.2
BiLSTM-AM	20.89	17.2
CNN-BiLSTM-AM	**19.54**	**16.8**

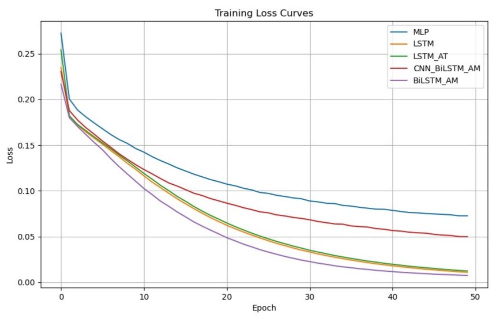


Figure 3: Training loss vs Epochs

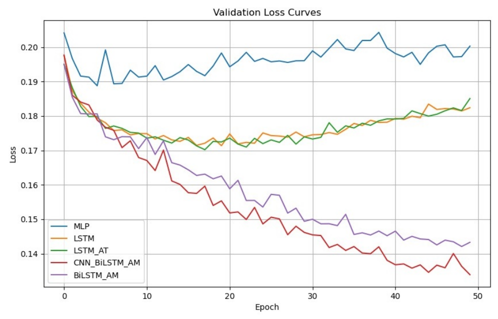


Figure 4: Validation Loss vs Epochs

To evaluate the sector-wise generalization ability of the CNN-BiLSTM-AM model, we computed the sector-wise RMSE values. In this analysis, we also calculated the average price of the stocks within the same sector to gain a better understanding of the implication of the RMSE value in comparison to the average price.

To revert the predicted stock prices back to their original scale, we applied denormalization using the mean and standard deviation calculated company-wise during the standardization process. After denormalization, we computed the root mean square error (RMSE) for each sector to assess the accuracy of the predictions. However, to provide a more interpretable measure of model performance, we also utilized the mean absolute percentage error (MAPE), which expresses the prediction errors as a percentage of the actual stock prices. The rationale behind incorporating MAPE alongside RMSE lies in the fact that RMSE values are inherently relative. For example, an RMSE of $3 may seem significant if compared to an average actual value of $6, but it might appear comparatively less significant when contrasted with an average actual value of $30. Therefore, employing MAPE alongside RMSE offers a clearer understanding of the prediction accuracy across different scales of the actual stock prices.

Table 4 shows the RMSE and MAPE performance of the CNN-BiLSTM-AM model in different sectors. It is observed that the model demonstrates varying performance across different sectors, as evidenced by the RMSE and MAPE values. Sectors, such as Utilities stand out with notably low

Table 4: RMSE and MAPE Performance of CNN-BiLSTM-AM Model in Different Sectors

Sector	Actual Average Price of Stocks ($)	RMSE ($)	MAPE (%)
Basic Material	30.78	6.06	22.88
Consumer Discretionary	120.32	25.63	19.08
Consumer Staples	75.77	13.02	13.32
Energy	45.2	11.59	22.96
Finance	64.29	9.61	12.88
Health Care	92.41	15.07	21.37
Industrials	114.08	17.83	17.08
Real Estate	82.26	11.02	14.16
Technology	79.74	15.87	18.77
Telecommunications	37.25	6.18	14.09
Utilities	**55.49**	**4.75**	**8.00**
Miscellaneous	58.09	11.14	17.11

RMSE ($4.75) and MAPE (%8.00) values, indicating high accuracy in predicting stock prices within this sector. On the other hand, sectors like Consumer Discretionary and Health Care exhibit relatively higher RMSE and MAPE values, suggesting comparatively lower accuracy in predicting stock prices within these sectors.

5 Discussion

As established by one of the design constraints in this project, it is imperative that the MAPE remains within 20% to ensure a sufficient level of accuracy in the predictions to screen for potential investment prospects. In other words, a viable model would need to be able to predict within 20% of the true stock value. The BiLSTM-AM model and CNN-BiLSTM-AM achieve this with a MAPE of 17.2% and 16.8% respectively. From Table 4, we observe that the MAPE of the CNN-BiLSTM-AM model ranges from 8.0% to 22.88%, indicating that nine out of the twelve sectors satisfy the aforementioned constraint. This indicates that, for the majority of the investigated sectors, the implementation of CNN-BiLSTM-AM successfully adheres to this design constraint. This observation highlights the model's ability to generate predictions with a level of accuracy deemed acceptable for practical investment decision-making as defined by our project goal. By consistently satisfying the defined constraint across diverse sectors, the CNN-BiLSTM-AM model demonstrates its reliability and suitability for forecasting stock prices across different industries.

However, the significant difference between the highest and lowest MAPE values in Table 4 emphasizes a limitation in the model's generalization ability. The model performs well in stable industries like Utilities, but struggles in more volatile markets like Energy. Overall, the model has sufficient performance across sectors to be able to be applied to the entire market, but the hypothesis is that it struggles to account for external shocks, such as COVID-19 until these events impact the financial statements.

Normalization had a significant impact on accuracy as illustrated in Table 2. Company-wise Z-score normalization performed the best, because it most successfully mitigated the impact of outliers. There exists a huge range of fundamental indicator values between companies on the magnitude of 10^{21} dollars in the used dataset, which can cause a few companies to significantly skew the data. For example, a single extreme positive value in Min-Max would scale the rest of the data close to 0. Z-score normalization minimizes the impact of extreme values by using mean and standard deviation so the value would be divided by the number of companies, dampening the effect of a few extreme values. Likewise, company-wise outperforms sector-wise which outperforms market-wise normalization, because it normalizes over similar values. For example, a technology company is likely to behave more similarly to other technology companies than companies in other industries. A software technology company such as Google is more likely to behave similarly to its historical self compared to a hardware technology company like Dell. The diverse nature of financial markets is challenging to work with and normalization schemes played a greater role in accuracy than model selection.

Examining the final results, we found that models with LSTM networks have a lower MAPE and RMSE compared to our MLP model architecture, indicating that temporal dependencies within

LSTMs improve the predictive capabilities of models. Furthermore, increasing model complexity by incorporating bidirectional LSTM, CNN, and AM layers leads to further reductions in MAPE and RMSE values. The attention led to a minor 1% improvement in performance suggesting past financial reports are important to predict stock price not just the most recent one. The CNN-BiLSTM-AM model had a MAPE of 0.4% lower than the BiLSTM-AM model, which indicated that the CNN layers did not have a significant impact. The implication is that all of the fundamental financial features were important to the valuation of a company. This makes sense because financial statements provide crucial consolidated metrics.

Therefore, while our CNN-BiLSTM-AM model implementation partially achieve the design constraint on the margin error of MAPE in various sectors, the substantial variance in MAPE across different sectors suggests room for improvement in generalization ability. Potential reasons for this variability include insufficient model complexity and distinct data distributions among companies.

6 Conclusion and Future Work

To conclude, our investigation into the improved predictive performance of deep learning models in predicting next-quarter stock prices based on historical fundamental financial data provided significant insights, with the CNN-BiLSTM-AM model emerging as the best performer. This model's hybrid architectures, combining CNN, BiLSTM and AM, demonstrates better predictive capabilities, surpassing other models, such MLP, LSTM, LSTM-AM and BiLSTM-AM. More specifically, the CNN-BiLSTM-AM model attained the lowest RMSE of $19.54 and the lowest MAPE of 16.8% among all the models evaluated over all the stocks in the preprocessed dataset. Moreover, the exploration of normalization techniques revealed opportunities to improve predictive accuracy by effectively preprocessing the data.

The CNN-BiLSTM-AM model offers several advantages for quarterly stock price prediction. Notably, its generalizability is demonstrated by achieving lower RMSE across various market conditions, indicating its effectiveness in capturing underlying trends. BiLSTM layers capture long-term dependencies in the data, which is crucial for accurate forecasting. Additionally, the incorporation of an attention mechanism (AM) enhances predictive capabilities by focusing on relevant information during prediction. Moreover, the integration of CNN layers aids in feature extraction from sequential data, further improving performance.

However, the CNN-BiLSTM-AM model poses certain challenges. Its computational intensity requires substantial resources during training and inference, potentially limiting scalability. Furthermore, the complex architecture and need to store intermediate computations may result in high memory usage, posing practical constraints. Balancing memory consumption with performance optimization becomes crucial for practical deployment. Additionally, the model's intricate architecture may necessitate meticulous parameter optimization, further increasing computational cost. Nonetheless, with careful management, these challenges can be addressed to leverage the model's predictive capabilities effectively.

In future work, several avenues could be explored to further enhance the effectiveness and applicability of the current best-performing CNN-BiLSTM-AM model. One potential direction is to augment the dataset by including news articles, economic indicators, sentiment data, and legal cases, which would provide insights into potential stock price movements. Additionally, exploring novel methods for incorporating temporal dependencies and contextual information into the modeling framework could improve the accuracy of longer-term predictions. Further research into transformer models could focus on optimizing architecture and training strategies to handle the complexities of financial time series data more effectively. Moreover, investigating ensemble methods that dynamically adapt to changing market conditions and model performance could lead to more robust and adaptive prediction systems.

References

[1] Richard J Teweles and Edward S Bradley. *The stock market*, volume 64. John Wiley & Sons, 1998.

[2] Pablo Fernandez. *Valuation methods and shareholder value creation*. Academic Press, 2002.

[3] Tim Koller, Marc Goedhart, David Wessels, et al. *Valuation: measuring and managing the value of companies*, volume 499. john Wiley and sons, 2010.

[4] Pengfei Yu and Xuesong Yan. Stock price prediction based on deep neural networks. *Neural Computing and Applications*, 32(6):1609–1628, 2020.

[5] Carson Kai-Sang Leung, Richard Kyle MacKinnon, and Yang Wang. A machine learning approach for stock price prediction. In *Proceedings of the 18th international database engineering & applications symposium*, pages 274–277, 2014.

[6] Mehar Vijh, Deeksha Chandola, Vinay Anand Tikkiwal, and Arun Kumar. Stock closing price prediction using machine learning techniques. *Procedia computer science*, 167:599–606, 2020.

[7] Mahla Nikou, Gholamreza Mansourfar, and Jamshid Bagherzadeh. 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, 2019.

[8] Alireza Namdari and Zhaojun Steven Li. Integrating fundamental and technical analysis of stock market through multi-layer perceptron. In *2018 IEEE technology and engineering management conference (TEMSCON)*, pages 1–6. IEEE, 2018.

[9] Anita Yadav, CK Jha, and Aditi Sharan. Optimizing lstm for time series prediction in indian stock market. *Procedia Computer Science*, 167:2091–2100, 2020.

[10] Ming-Che Lee, Jia-Wei Chang, Sheng-Cheng Yeh, Tsorng-Lin Chia, Jie-Shan Liao, and Xu-Ming Chen. Applying attention-based bilstm and technical indicators in the design and performance analysis of stock trading strategies. *Neural computing and applications*, 34(16):13267–13279, 2022.

[11] Ritika Singh and Shashi Srivastava. Stock prediction using deep learning. *Multimedia Tools and Applications*, 76:18569–18584, 2017.

[12] Amadu Fullah Kamara, Enhong Chen, and Zhen Pan. An ensemble of a boosted hybrid of deep learning models and technical analysis for forecasting stock prices. *Information Sciences*, 594:1–19, 2022.

[13] Ming-Che Lee, Jia-Wei Chang, Jason C Hung, and Bae-Ling Chen. Exploring the effectiveness of deep neural networks with technical analysis applied to stock market prediction. *Computer Science and Information Systems*, 18(2):401–418, 2021.

[14] Kaustubh Khare, Omkar Darekar, Prafull Gupta, and VZ Attar. Short term stock price prediction using deep learning. In *2017 2nd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT)*, pages 482–486. IEEE, 2017.

[15] Jingyi Shen and M Omair Shafiq. Short-term stock market price trend prediction using a comprehensive deep learning system. *Journal of big Data*, 7:1–33, 2020.

[16] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

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

[18] Anne Treisman and Garry A. Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12:97–136, 1980.

[19] Luca Di Persio, Oleksandr Honchar, et al. Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International journal of circuits, systems and signal processing*, 10:403–413, 2016.

[20] Ryo Akita, Akira Yoshihara, Takashi Matsubara, and Kuniaki Uehara. Deep learning for stock prediction using numerical and textual information. In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pages 1–6. IEEE, 2016.

[21] Achyut Ghosh, Soumik Bose, Giridhar Maji, Narayan Debnath, and Soumya Sen. Stock price prediction using lstm on indian share market. In *Proceedings of 32nd international conference on*, volume 63, pages 101–110, 2019.

[22] Wenjie Lu, Jiazheng Li, Jingyang Wang, and Lele Qin. A cnn-bilstm-am method for stock price prediction. *Neural Computing and Applications*, 33(10):4741–4753, 2021.

Appendix A Tables

A.1 CNN-BiLSTM-AM Parameters

Table 5: Parameter settings of CNN-BiLSTM-AM model

	Description
Kernel size	1
Number of convolutional filters	256
Number of MaxPooling	1
Number of BiLSTM layers	2
Number of memory cells	4
Drop out layer probability	p = 0.2
Activation function	RELU
Number of hidden units in BiLSTM layer	256

Appendix B Architecture

B.1 MLP

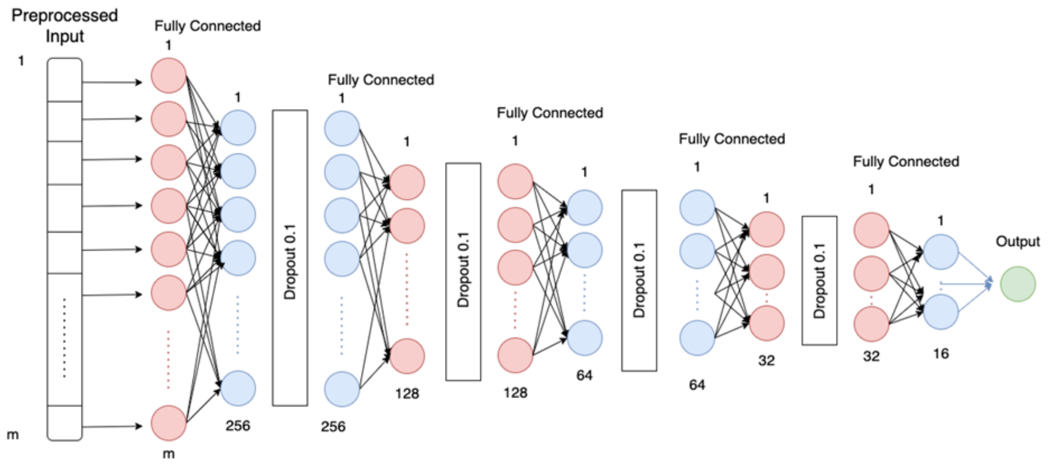
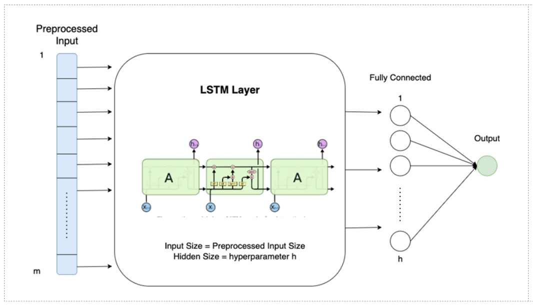


Figure 5: MLP Model Architecture

B.2 LSTM and LSTM-AM


Preprocessed Input
1
m
LSTM Layer
A
A
Input Size = Preprocessed Input Size
Hidden Size = hyperparameter h
Fully Connected
1
h
Output

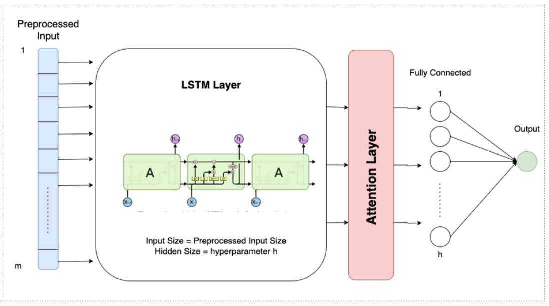


Figure 6: LSTM Model Architecture

Figure 7: LSTM-AM Model Architecture

B.3 BiLSTM-AM

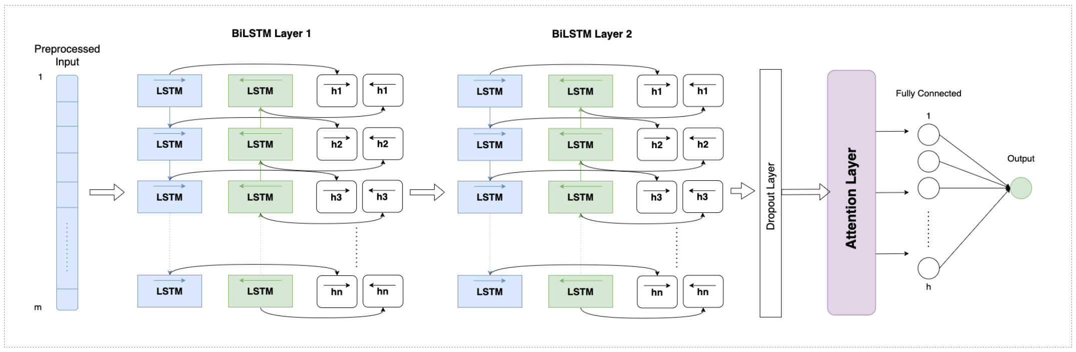


Figure 8: BiLSTM-AM Model Architecture

Appendix C Source Code

The source code for this project can be found on GitHub, kindly visit the below link to run the source code. Click here to be directed to the source code on GitHub