Stock price prediction using deep neural networks

Daehyuk Bue [* 1] **Seokju Kang** [* 1] **Sunhwa Lee** [2]

Abstract

This study evaluates the performance of various Deep Neural Network (DNN) architectures—MLP, RNN, LSTM, GRU, and TCN—in predicting stock prices using historical data from the SP 500, KOSPI 200, and Nikkei 225 indices. The results demonstrate that LSTM models, due to their ability to capture long-term dependencies through sophisticated gating mechanisms, outperform other architectures. Optimizers like Adam and AdamW, along with activation functions such as GELU, ReLU, ELU, and SELU, significantly enhance model performance. The optimal LSTM configuration found includes 32 nodes per layer and a single hidden layer, avoiding overfitting and ensuring high predictive accuracy. This study underscores the superiority of LSTM for stock price prediction and provides insights into the practical application of deep learning in stock price forecasting.

1. Introduction

Predicting stock prices is a critical and challenging task in financial markets, where accurate forecasts can lead to significant economic gains and effective risk mitigation. This study explores the application of various deep neural network (DNN) architectures for stock price prediction, focusing on their ability to handle temporal dependencies and overall forecasting performance.

Traditional machine learning models often struggle to capture the intricate and non-linear relationships present in stock market data. However, deep learning models such as Multi-layer Perceptrons (MLP), Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), Gated Recurrent Units (GRU), and Temporal Convolutional Networks (TCN) offer new avenues for improving prediction accuracy. Each of these models has unique characteristics that make them suitable for different aspects of sequence modeling. For instance, while MLPs are powerful, they lack the temporal modeling capabilities inherent in RNN-based models. Conversely, RNNs and their variants like LSTMs and GRUs are specifically designed to handle time series data by retaining long-term dependencies through their recurrent connections.

This study utilizes historical stock market data from three major indices: the SP 500, the KOSPI 200, and the Nikkei 225. These indices were chosen due to their high accessibility, stable market performance, and analytical significance. The experimental setup includes preprocessing steps such as normalization and the selection of optimal hyperparameters through grid search methods. Various configurations of the LSTM model, including the number of hidden layers, the number of nodes per layer, and different activation functions, were tested to identify the most effective architecture.

The results indicate that LSTM outperforms other models due to its ability to effectively capture long-term dependencies through its sophisticated gating mechanisms. GRUs, while simpler, also demonstrate robust performance, making them suitable alternatives where computational efficiency is a concern. TCNs, leveraging convolutional layers, offer a different approach to sequence modeling but do not achieve the same level of accuracy as LSTM or GRU. MLPs, lacking temporal modeling capabilities, exhibit the lowest performance in stock price prediction tasks.

This paper is organized as follows: Section 2 provides a detailed overview of the methodology, including the theoretical underpinnings of each DNN model. Section 3 describes the experimental setup and data used for model training and evaluation. Section 4 presents the performance analysis and comparison of the models. Finally, Section 5 concludes with a summary of findings and potential directions for future research.

Through this comprehensive analysis, this study aims to contribute to the ongoing efforts in improving stock price prediction models and provide insights into the practical applications of deep learning techniques in financial forecasting.

2. Methodology

2.1. DNN-based stock price prediction method

2.1.1. Multi-layer Perceptron (MLP)

MLP(Rumelhart et al., 1986) consists of a network of densely connected neurons between adjoining layers. One of the peculiarities of Feed Forward Neural Networks is that output of one layer is never fed back to the previous layers. The input which goes to every neuron is the weighted sum of all the outputs from the previous layer of the neural network. The conversion of this input into the output is performed by a continuous and differentiable activation function. The output of one pass is produced after the signals propagate from input to the output layer. The error for the pass is calculated, for regression it is usually root mean squared error or mean squared error. The learning algorithm, generally a kind of gradient descent algorithm, adjusts the weights of the neurons necessary to reduce the error. The data is passed to the model several times to adjust the weights to reduce the errors until the preset number of epochs are reached.

2.1.2. Recurrent Neural Network (RNN)

RNNs(Elman, 1990) achieve explicit modelling of time through self-connection of the hidden layers and record long-term information by improving the nodes in the hidden layers. RNNs have achieved marked results in natural language processing and audio frequency analysis. In conventional RNNs, there are links between the nodes in the hidden layers. Owing to these recurrent feedback links, network models have a memory ability. Thus, RNNs can model information on a time scale. The duration of information transfer can be treated as the model depth. However, earlier RNNs are unable to model information with a long time span and can lead to a vanishing gradient problem when used to build large time scale models. By optimizing the nodes, deep RNNs are able to efficiently model on a time scale and prevent the occurrence of a vanishing gradient problem.

2.1.3. Long Short-Term Memory (LSTM)

LSTMs(Hochreiter & Schmidhuber, 1997) are a kind of RNNs which effectively capture long term dependencies in time series prediction problems. It is due to these dependencies that the order of input plays a significant role prediction. The steps followed in the overall algorithm of LSTM is the same as MLP except for the processing of input inside every neuron. Unlike normal neurons, the output of every LSTM cell is a result of a multistep process. LSTMs have an additional memory, called cell state, which stores relevant past information toaid in prediction. The information stored in the cell state is modified by structures, called gates, in the following steps. Initially, the forget gate decides whether to eliminate any available information. The input gate and tanh layer then decide which new information is to be stored. Further, the information gets added and deleted according to the previous gates. Finally, the activation function is applied to the data and the output is produced.

2.1.4. Gated Recurrent Unit (GRU)

GRUs(Cho et al., 2014) are an advancement in recurrent neural networks (RNNs) designed to enhance the modeling of temporal data by addressing the vanishing gradient problem inherent in conventional RNNs. GRUs achieve this by incorporating gating mechanisms that regulate the flow of information within the network. These mechanisms enable GRUs to capture long-term dependencies and maintain efficient performance over extended time scales. The architecture of GRUs has demonstrated significant improvements in various tasks, including natural language processing and time-series analysis, by effectively mitigating the vanishing gradient problem. Unlike traditional RNNs, which struggle with long-term dependencies due to exponential decay of gradients, GRUs maintain stable gradients through their gating mechanisms. This allows GRUs to build deeper models and capture complex temporal patterns more effectively. By optimizing the information flow through their gates, GRUs can model sequences over long time spans without the degradation of learning signals. This makes them particularly suitable for applications requiring the retention of information over prolonged periods, such as speech recognition, text generation, and financial forecasting.

2.1.5. Temporal Convolutional Network (TCN)

TCNs(Bai et al., 2018) represent an advanced approach in sequence modeling by leveraging convolutional layers to handle temporal data. Unlike recurrent neural networks (RNNs), which model time dependencies through recurrent connections, TCNs utilize dilated causal convolutions to capture long-term dependencies efficiently and effectively. The key advantage of TCNs lies in their ability to process sequences in parallel, leading to faster training times compared to RNNs, which process sequences sequentially. Additionally, TCNs offer a more stable training process since they do not suffer from the same gradient issues that plague traditional RNNs. TCNs have demonstrated superior performance in various domains, achieving state-of-the-art results in areas such as natural language processing and sequence prediction tasks. By optimizing the depth and dilation of convolutional layers, TCNs can model complex temporal patterns with high accuracy and efficiency. The design of TCNs ensures that they can effectively capture hierarchical features from sequential data, providing a robust alternative to traditional RNN-based methods.

2.2. Evaluation metrics for prediction results

We used MAE, RMSE and R^2 as measures to evaluate the performance of the prediction methods. MAE measures error without considering the directions of the predicted values. RMSE measures the average magnitude of estimation error in predicted values. MAE and RMSE are measures

Table 1. Statistics of closing prices of each stock index selected for the experiment.

INDEX	S&P500	KOSPI 200	NIKKEI 225
BEGIN DATE	20220103	20220104	20220104
END DATE	20231229	20231228	20231229
MEAN	4190.94	330.47	28994.23
MEDIAN	4158.24	328.80	27925.42
MAX	4796.56	395.40	33753.33
MIN	3577.03	281.36	24717.53
STD.	278.53	21.56	2509.43
SAMPLES	501.0	489.0	490.0

of closeness which evaluates the accuracy of the predicted value to the actual price. R^2 measures the linear correlation between two variables and eliminates the influence of dimension on different regression problems. We hope the model has low MAE and RMSE and a high R^2 . The three metrics are defined as follows:

$$\mathrm{MAE}=\frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i| \tag{1}$$

$$\mathrm{RMSE}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2} \tag{2}$$

$$R^2=1-\frac{\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(\bar{y}-y_i\right)^2} \tag{3}$$

where y_i represents the true value of the target variable of sample i, $\hat{y}_i$ represents the predicted value of the target variable of sample i, $\bar{y}$ represents the mean of the target variable's true value of all samples and m represents the total number of samples.

3. Experimental results and Analysis

3.1. Description of the experimental data

Stock market index data were selected as experimental data based on the following several factors: high accessibility, stable market performance and relatively high analytical significance. Data for three stock indices for various market environments, namely the S&P 500, the KOSPI 200, the Nikkei 225, were obtained from Yahoo Finance (finance.yahoo.com), The analytical data are the closing prices for each trading day. Table 1 presents a description of the selected data.

3.2. Performance analysis of the prediction model

To determine the effectiveness of the established model in predicting financial data, multiple prediction models were used to predict the data for the three stock indices for various market environments. Table 2 summarizes the description of the comparison experimental models. To facilitate analysis and evaluation, the experimental results are presented in Figure 1, 2, 3 and Tables 3.

Table 2. Description of the prediction models selected for comparison

MODEL	PREPROC	HYPER PARAMETERS	OPTIMIZER
MLP	NORM	HIDDEN LAYER: 6	ADAM
		NODE OF LAYER:	
		[64, 64, 32, 32, 16, 16]	
RNN	NORM	HIDDEN LAYER: 4	ADAM
		NODE OF LAYER: 32	
LSTM	NORM	HIDDEN LAYER: 2	ADAM
		NODE OF LAYER: 32	
GRU	NORM	HIDDEN LAYER: 4	ADAM
		NODE OF LAYER: 32	
TCN	NORM	HIDDEN LAYER: 2	ADAM
		KERNEL SIZE: 7	

The varied performance of different models in stock price forecasting can be attributed to their distinct architectures and their efficacy in handling temporal dependencies. Despite the RNN model showing the best performance in terms of metrics, it generally tends to underperform compared to LSTM in practice due to its susceptibility to the vanishing gradient problem, which hampers its ability to learn long-term dependencies effectively. LSTM, with its sophisticated gating mechanisms, consistently manages these dependencies better, preventing the gradient from vanishing and thereby providing more stable and reliable performance across various stock indices.

GRU follows closely, leveraging a similar gated architecture but with fewer parameters, making it less powerful than LSTM but still robust. TCN, which uses convolutional layers to capture temporal patterns, performs moderately well but not as effectively as LSTM or RNN due to its different approach to sequence modeling. MLP, a feedforward neural network, shows the lowest performance as it lacks the inherent ability to model temporal dependencies, which are crucial for time series prediction.

LSTM was chosen as the final model due to its consistently high performance across various stock indices. Its architecture, comprising forget, input, and output gates, allows it to effectively capture and utilize long-term dependencies, making it particularly suitable for stock price prediction where historical information significantly influences future values. The robustness of LSTM in capturing complex temporal patterns ensures more reliable and accurate forecasts, thereby justifying its selection as the optimal model for this task.

Table 3. Comparison of the results produced by the prediction models for the S&P 500

MODEL NAME	MAE	RMSE	R^2
MLP	88.2656	103.901	0.5901
RNN	31.2993	39.4952	0.9408
LSTM	46.1378	60.0706	0.8630
GRU	49.3733	57.2571	0.8755
TCN	58.8399	70.7416	0.8099

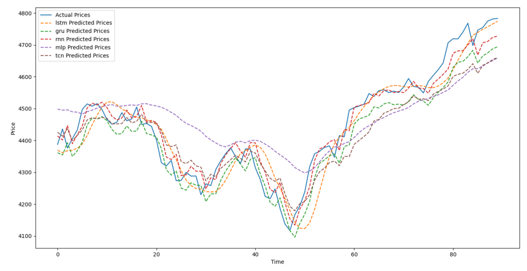

Figure 1. Comparison of the results produced by various prediction models for S&P 500 Index

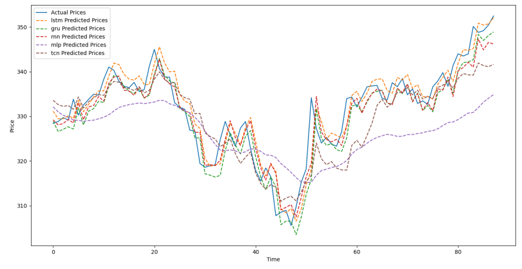

Figure 2. Comparison of the results produced by various prediction models for KOSPI 200 Index

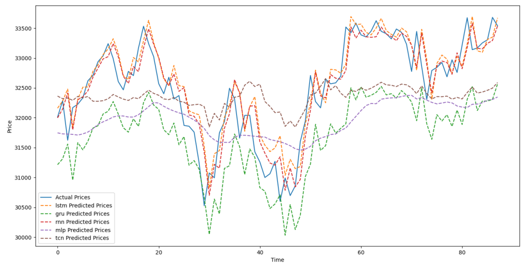

Figure 3. Comparison of the results produced by various prediction models for NIKKEI 225 Index

3.3. Analysis of the LSTM structure

The three main parameters of the established DNN-based prediction model, namely the number of hidden layers, the number of LSTM nodes in the hidden layers and the activation function of the LSTM output, were determined through an experiment. Other hyperparameters were similarly set based on the grid searching method and the relevant literature(Song et al., 2019; Minh et al., 2018) as follows: time step, 1; initial learning rate, 0.001; the initial weights follow the normal distribution pattern; probability for preservation of nodes, 0.2; and number of iterations, 2500. To facilitate better observation and comparison of the experimental results, normalization was first performed in the model parameter experiment. In addition, relatively fine experimental data were selected: 500 sets of data for the S&P 500 for a nearly 2-year (January 3, 2020 to December 29, 2023) period.

Number of Nodes First, the number of LSTM nodes in the hidden layers was analysed through an experiment under the following conditions: number of hidden layers, 2, and activation function of the output, Rectified Linear Unit (ReLU) function. Table 4 and Figure 4 present the experimental results for the test set.

The experimental results demonstrate that there was no significant difference between the graphs plotted by the established single-LSTM hidden layer models with various numbers of nodes. The data in Table 4 indicate that the model prediction accuracy changed with the number of nodes in the hidden layers. Relatively excellent results were obtained when the number of nodes in the hidden layers was set to 32, 56.

Number of Layers After the number of LSTM nodes in the hidden layers was determined, the number of hidden layers was determined through an experiment. Models established in most of the relevant studies (including those on RNNs) do not contain too many hidden layers. Thus, the experimental range for the number of hidden layers was set to [1, 8]. Table 5 and Figure 5 present the experimental results.

The LSTM model's performance declines as the number of layers increases. A single-layer LSTM performs best, indicating that simpler models are more effective for this task. As the number of layers grows, the model becomes increasingly complex, which leads to overfitting. This overfitting means that the model learns the noise and fluctuations in the training data rather than the underlying patterns, resulting in poorer generalization to new data. Consequently, the predictive accuracy drops, and the model's ability to explain

Table 4. Results of the comparison experiment on the number of nodes in the hidden layers of the model

Number of node	MAE	RMSE	R^2
8	32.9854	40.7184	0.9371
16	32.7917	41.0661	0.9360
24	33.0875	39.5494	0.9406
32	27.5483	35.5835	0.9519
40	29.0268	36.4776	0.9495
48	31.1683	37.3405	0.9471
56	27.6562	35.1318	0.9531
64	28.5440	37.4215	0.9468

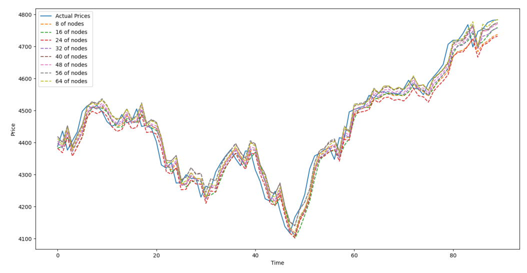

Figure 4. Results of the comparison experiment on the number of nodes in the hidden layers of the model

the variance in the data decreases. Thus, we can find that the simpler one-layer model outperforms deeper models by avoiding unnecessary complexity and overfitting.

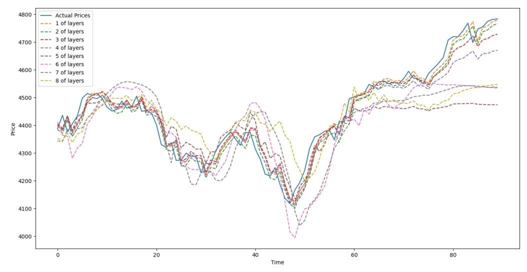

Figure 5. Results of the comparison experiment on the number of hidden layers

Activation function Table 6 and Figure 6 present the experimental results of activation function experiments. Regarding that, the LSTM model performed best with the GELU activation function, while ReLU(He et al., 2015), ELU(Clevert et al., 2016), and SELU(Klambauer et al., 2017) also showed similar strong performances. In contrast, Sigmoid(Rumelhart et al., 1986) exhibited the lowest performance. These results stem from the necessity for the LSTM model to effectively learn complex patterns and non-linearities in stock price prediction. ReLU and its variants (ELU, SELU, GELU) provide strong non-linear capabilities and mitigate the vanishing gradient problem, allowing the model to capture long-term dependencies well. Specifically, GELU(Hendrycks & Gimpel, 2016) offers more flexible non-linearity by reflecting the probabilistic nature of inputs, thereby enhancing predictive performance. On the other hand, Sigmoid restricts output values between 0 and 1, making it prone to the vanishing gradient problem, which hampers its ability to learn the intricate patterns in time series data, resulting in lower performance.

Table 5. Results of the comparison experiment on the number of hidden layers

Number of layer	MAE	RMSE	R^2
1	27.4987	34.7529	0.9541
2	31.3871	38.2601	0.9444
3	31.0988	38.4353	0.9439
4	39.5045	50.1326	0.9045
5	82.7246	114.6833	0.5006
6	93.6487	114.2244	0.5046
7	100.7699	118.7916	0.4642
8	93.8319	115.2337	0.4958

Table 6. Experimental results obtained when different activation functions of the output gate were selected

Activate function	MAE	RMSE	R^2
ReLU	27.5599	34.9128	0.9537
Sigmoid	110.9188	138.014	0.2768
Tanh	46.9835	60.1055	0.8628
LeakyReLU	37.524	44.1356	0.9260
ELU	27.7744	35.7294	0.9515
SELU	27.7625	36.0371	0.9506
Softplus	65.1998	81.9638	0.7449
GELU	27.3778	34.9853	0.9535

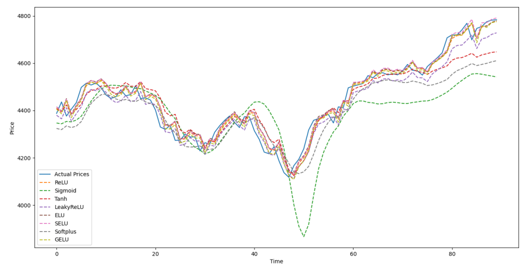

Figure 6. Experimental results obtained when different activation functions of the output gate were selected

Table 7. Results of the comparison experiment on the type of optimizers

TYPE OF OPTIMIZER	MAE	RMSE	R^2
ADAM	29.0663	35.9513	0.9509
RMSPROP	36.8911	48.7467	0.9097
ADAMW	28.9726	35.9927	0.9508
ADAGRAD	106.4595	123.2675	0.4230
ADAMAX	31.4249	38.4774	0.9437
RPROP	89.2555	105.5276	0.5771

3.4. Analysis of the LSTM Optimizer

The LSTM model's performance varies significantly with different optimizers. Table 7 and Figure 7 present the experimental results. Adam(Kingma & Ba, 2014) and AdamW(Loshchilov & Hutter, 2017) optimizers show the best performance with low MAE, and RMSE values, and high R^2 scores around 0.951, indicating strong predictive accuracy and model fit. RMSprop also performs reasonably well but is slightly less effective than Adam and AdamW, with higher error metrics and a lower R^2 value of approximately 0.910. Adamax(Kingma & Ba, 2014) shows moderate performance, with acceptable error metrics but lower predictive accuracy compared to Adam and AdamW. Adagrad(Duchi et al., 2011) and Rprop(Riedmiller & Braun, 1993) exhibit the lowest performance, with significantly higher MAE, and RMSE values, and much lower R^2 scores of 0.423 and 0.577, respectively. These results suggest that while Adam and its variant AdamW are well-suited for this stock price prediction task due to their adaptive learning rate capabilities and efficient handling of sparse gradients, Adagrad and Rprop may struggle with the dynamic nature of stock price data, leading to poorer predictive performance.

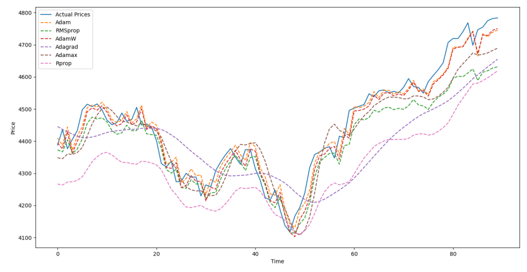

Figure 7. Results of the comparison experiment on the type of optimizers

4. Conclusion

This study investigates the application of various Deep Neural Network (DNN) architectures for stock price prediction, focusing on their ability to handle temporal dependencies and overall forecasting performance. The analysis covers models such as Multi-layer Perceptrons (MLP), Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), Gated Recurrent Units (GRU), and Temporal Convolutional Networks (TCN). The primary objective was to evaluate and compare these models using historical stock market data from three major indices: the SP 500, the KOSPI 200, and the Nikkei 225. The performance was assessed using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2).

Our results highlight several key findings. LSTM models outperform other DNN architectures in predicting stock prices due to their ability to effectively capture long-term dependencies through sophisticated gating mechanisms. GRUs also show strong performance, closely following LSTM, and offer a robust alternative where computational efficiency is a priority. While TCNs leverage convolutional layers to capture temporal patterns and provide faster training times compared to RNNs, they do not achieve the same level of accuracy as LSTM or GRU.

Additionally, the choice of optimizer and activation function significantly impacts model performance. Adam and AdamW optimizers provide the best performance, suggesting their suitability for dynamic stock price data. Activation functions like GELU, ReLU, ELU, and SELU show strong performance, while Sigmoid is less effective due to its propensity for the vanishing gradient problem. The number of LSTM nodes and hidden layers is also crucial for model performance. Optimal configurations found in this study include 32 nodes per layer and a single hidden layer, balancing complexity and performance. Overly complex models with more layers tend to overfit, resulting in lower predictive accuracy.

The results of this study provide valuable insights into the practical applications of deep learning techniques in financial forecasting. LSTM models, in particular, are highlighted as the most effective for stock price prediction due to their superior handling of temporal dependencies. Future research can build on these findings by exploring hybrid models that combine the strengths of different DNN architectures or integrating additional market indicators to further enhance prediction accuracy.

In conclusion, this research demonstrates that deep learning models, especially LSTM, offer significant improvements over traditional machine learning models for stock price prediction. By effectively capturing the temporal dependencies in financial data, these models contribute to more accurate and reliable forecasting, aiding in economic decision-making and risk management in financial markets.

References

Bai, S., Kolter, J. Z., and Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)*, 2016.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Elman, J. L. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pp. 971–980, 2017.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Minh, D. L., Sadeghi-Niaraki, A., Huy, H. D., Min, K., and Moon, H. Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. *Ieee Access*, 6:55392–55404, 2018.

Riedmiller, M. and Braun, H. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE international conference on neural networks*, pp. 586–591. IEEE, 1993.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Song, Y., Lee, J. W., and Lee, J. A study on novel filtering and relationship between input-features and target-vectors in a deep learning model for stock price prediction. *Applied Intelligence*, 49:897–911, 2019.