

# Predicting Stock Exchange Index: A Transformer-based Model Approach

*Albert Matthew Alejo<sup>1</sup>, Jaime Angelo Nery<sup>1</sup>,  
Zielle Frances Realda<sup>1</sup>, Jeric Briones<sup>1,2,\*</sup>*  
\*jbriones@ateneo.edu

<sup>1</sup>Department of Mathematics

<sup>2</sup>Department of Finance and Accounting  
Ateneo de Manila University  
Quezon City, Philippines

## Abstract

*Growing interest in machine learning and artificial intelligence, as well as advancements in technology, have inspired works which explore the feasibility of using these models on financial time series. Building up on those works, this work explores the potential of utilizing a Transformer-based architecture to forecast the Philippine Stock Exchange index (PSEi). To check its feasibility, data from 2010 to 2019 was analyzed and used for back-testing, with the performance of the model compared to other neural network architectures. Furthermore, to find potential limitations of the model, data from the stock indices of Indonesia, Malaysia, and Thailand, and data from the 2008 global financial crisis and the COVID-19 pandemic were also considered. A comparative analysis among the architectures considered highlighted the superior predictive capability of Transformer models. The additional tests conducted on regional stock index data, however, revealed worsened performance. Additionally, evaluating the models against stock index data from bear market also revealed worsened performance in terms of error metrics. The Transformer thus failed to keep its performance advantage, suggesting that it is not robust with respect to varying market conditions. Regardless, these results still indicate that the Transformer model is a viable option for forecasting financial time series such as stock index prices and returns. Moreover, these findings also highlight how advances in technology can contribute in advancing computational methods for financial math and quantitative finance.*

## 1 Introduction

A facet of financial math revolves around using mathematical and statistical modeling to predict and analyze various key financial metrics such as bond yields and stock prices. Among these metrics, stock prices have shown to be difficult to predict due to their seemingly random nature. Despite this, there have been many attempts to develop quantitative methods for forecasting these stochastic financial time series. These include technical analysis models, time series methods, and econometric multivariate models, among others [11, 13]. These approaches,

however, rely heavily on historical price and volume data to predict future movements. This reliance assumes that past market behavior will continue into the future, which may not always hold true, especially when the market encounters unforeseen events.

With the advancement in technology, there is growing interest in using more computational models such as machine learning and artificial intelligence for quantitative finance and financial math [2, 7, 12]. For example, a popular architecture used in stock price prediction is the recurrent neural networks (RNN). These RNN models capture dependencies by sequentially processing data, allowing them to leverage temporal dependencies across previous time steps. In [10], RNNs were applied to minute-wise stock price data from three National Stock Exchange (NSE) listed companies. Their results indicated that RNN had competitive predictive capabilities for two stocks, albeit with a marginally higher error rate observed for the third stock. However, the study highlighted the model's limitation in capturing dynamic shifts accurately within rapidly changing systems like the stock market [10]. Additionally, RNNs struggle to capture long-term dependencies due to their sequential processing, which can result in unstable predictions. To address these challenges, long short-term memory (LSTM) models were developed by incorporating the concept of cell states into the RNN architecture. These cell states serve as memory units that retain information across time steps. Another model that has proven successful in stock price prediction is the convolutional neural networks (CNN). Originally designed for image processing and computer vision tasks, they have been adapted to handle 1-dimensional vector inputs, allowing them to process sequential data (like stock prices) through analogous transformations and convolutional processes. These convolutional filters extract features and patterns in the data. In stock price prediction for instance, a filter may learn to detect the pattern directly after an increase in the price or return [1, 9].

The Transformer model [14] is another emerging architecture worth considering because, similar to the RNN, it also processes sequential data. While this model was made with natural language processing in mind, various studies have shown success with using the model to forecast or predict stock prices [8, 15]. Despite these, the Transformer model still faces challenges when applied to long-term sequential data such as time series data. As the Transformer model was originally designed for relatively short sequences, the model's key mechanism, while powerful, has limitations and challenges when dealing with longer sequences of data. Nonetheless, while the adaptation of the Transformer model is not as widespread as compared to RNN for the prediction task, previous studies [3, 5, 6, 15] have shown success in the prediction of major stock indices. These then underscore the importance of this work to determine the viability of these models with emerging market indices such as the Philippine Stock Exchange index.

Having said these, this work endeavors to investigate the applicability and effectiveness of the Transformer model in forecasting tasks, specifically focusing on stock market index prediction for emerging markets. To be specific, stock index log returns will be predicted using the aforementioned model, with stock index prices computed using the predicted log returns. The work thus aims to achieve the following objectives: (1) utilize the Transformer model to forecast 1-day stock returns using data from the Philippine Stock Exchange Index (PSEi) from 2010 to 2019; (2) evaluate the performance of the model by computing selected performance metrics, and comparing the backtested results against a rudimentary RNN, LSTM, and CNN; and (3) further test the approach by including data from other regional markets (IDX Composite, FTSE Bursa Malaysia KLCI, and SET Index), the 2008 financial crisis, and the COVID-19 pandemic as inputs.

As this study wishes to explore the feasibility of using a Transformer model to predict stock index returns in the Philippine setting, fine-tuning the hyperparameters of the model was not fully explored. Only one hyperparameter (dropout) was tuned using a simple grid search method. Moreover, unlike other implementations, this study did not make use of a separate validation set for hyperparameter tuning. Other hyperparameter values were simply taken from established practices from related literature. Finally, results from traditional forecasting approaches are not included as this work wishes to focus on verifying whether the aforementioned machine learning models can perform the prediction task at hand.

## 2 The Transformer Model

To better understand the Transformer model, this section provides an overview of key concepts essential for understanding and contextualizing the contents of the study. It also explains how each component layer contributes to the ability of the Transformer to efficiently process sequential data and capture patterns within the data.

### 2.1 Positional Encoding

In order to make use of the dates in the inputs, the model positionally encodes them using `Time2Vec` [4]. This method gives a vector representation of time, making it useful in time series analysis and forecasting tasks. `Time2Vec` takes in a parameter  $\tau$  as its input, where  $\tau$  represents the periodicity or the characteristic time scale of the time series data. In this work, the parameter  $\tau$  is calculated based on the Unix time of the date at midnight, which is then scaled by a factor of  $\frac{1}{86,400}$  (1 day = 86,400 seconds). Note that Unix time is simply the number of seconds that have elapsed since January 1, 1970 00:00:00 UTC. That is,

$$\tau = \frac{\text{Unix time of the date at midnight}}{86400}. \quad (1)$$

This way,  $\tau$  represents the number of days elapsed since the Unix epoch (January 1, 1970) at midnight, providing a way to represent time information suitable for `Time2Vec`.

### 2.2 Model Components

These pre-processed inputs are then fed into the Transformer model based on [14] (see Figure 1). The Transformer model is divided into two main components: the encoder and the decoder. The encoder is responsible for processing the input sequence and extracting contextual representations of the data, which will then be used by the decoder for the final output. On the other hand, the decoder is responsible for generating the output sequence based on the representations learned by the encoder. Both of these components consist of  $M$  layers that each process the data in parallel. At the heart of each of those layers is the *multi-headed self-attention*. Aside from the multi-headed self-attention layer, there are also intermediate layers like the *residual connection layer* and a *layer normalization* that enhance the performance of the model.

Multi-headed self-attention is simply a self-attention mechanism that is done in parallel using different heads whose outputs are then concatenated together afterwards. Basically, this

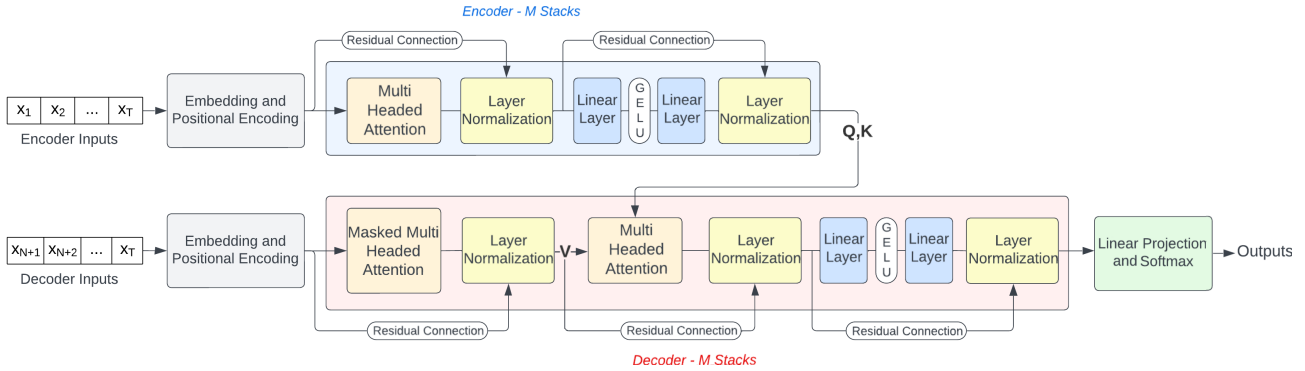


Figure 1: Illustration of the Transformer model as adapted in this work. The top block represents the *encoder* while the bottom block represents the *decoder*. Here, the output of the former is fed as input of the latter.

mechanism calculates the importance of each element in the sequence relative to the other elements, allowing it to capture dependencies in the data. A single self-attention mechanism makes use of learnable representations of the input data called *queries*, *keys*, and *values*. Let  $\mathbf{X}$  be the  $T \times d$  positionally-encoded matrix with embeddings of size  $d$ . Define the Query ( $\mathbf{Q}$ ), Key ( $\mathbf{K}$ ), and Value ( $\mathbf{V}$ ) matrices as  $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$ ,  $\mathbf{K} = \mathbf{X}\mathbf{W}_K$ , and  $\mathbf{V} = \mathbf{X}\mathbf{W}_V$ , respectively. Here,  $\mathbf{W}_Q \in \mathbb{R}^{d \times d_k}$ ,  $\mathbf{W}_K \in \mathbb{R}^{d \times d_k}$  and  $\mathbf{W}_V \in \mathbb{R}^{d \times d_v}$ ,  $d_k$  refers to the dimensionality of  $\mathbf{Q}$  and  $\mathbf{K}$ , while  $d_v$  refers to the dimensionality of  $\mathbf{V}$ . Thus, the matrices  $\mathbf{Q}$  and  $\mathbf{K}$  have dimensions  $T \times d_k$  while  $\mathbf{V}$  has dimensions  $T \times d_v$ .

These components are used to compute attention scores among different positions in the input sequence. These attention scores are defined by  $\mathbf{Q}\mathbf{K}^\top$ , where  $\mathbf{K}^\top$  denotes the transpose of  $\mathbf{K}$ . This operation measures the similarity among each query vector and key vector, highlighting what is most important for prediction-making. These attention scores are then scaled by the square root of the dimensionality of the key vectors ( $\sqrt{d_k}$ ). This helps stabilize the gradients during training. From here, the scaled attention scores are passed into the softmax function, converting them into a valid probability distribution. This is then multiplied to the value matrix  $\mathbf{V}$  to obtain the final output of the attention mechanism. Vaswani et. al refers to this set of operations as the *scaled dot-product attention* (see Figure 2). Let  $\mathbf{Q}$ ,  $\mathbf{K}$  and  $\mathbf{V}$  be the matrices defined earlier. The output of the scaled dot-product attention is given by:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}. \quad (2)$$

As mentioned above, the resulting matrices from each of the self-attention heads are concatenated together to form one large matrix. Not only does this allow the Transformer to process information in parallel, it also allows the model to attend to different parts of the input sequence. This then allows it to better detect different patterns and sequences in the data.

### 2.3 Architecture Summary

In the decoder architecture, one of the multi-headed attention layers undergoes an additional process called *masking* (see Figure 1). The masking mechanism restricts the model from attending to future positions in the sequence during self-attention. Within masked multi-head

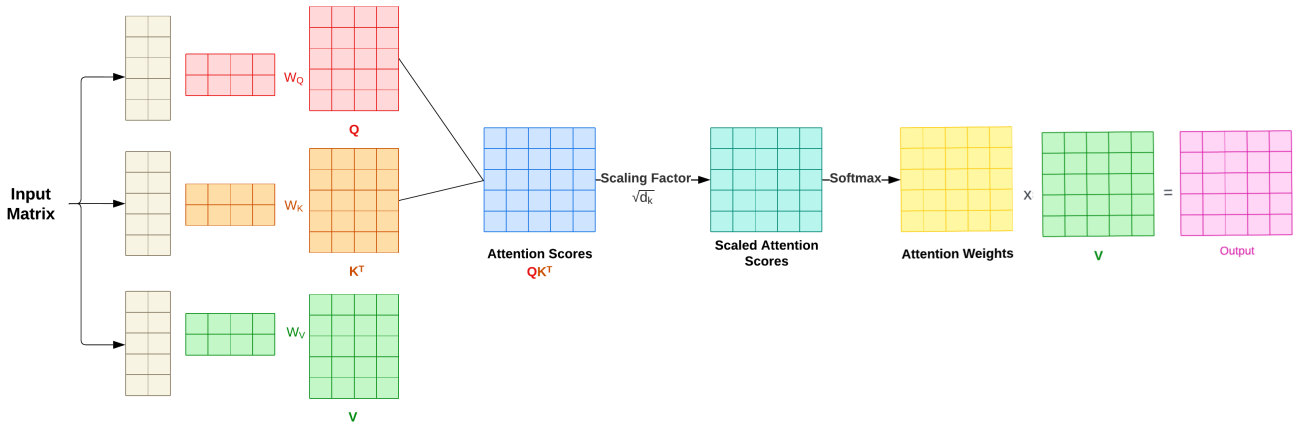


Figure 2: Illustration of self-attention mechanism as adapted in this work. Here, the input matrix is partitioned into sub-matrices, where each sub-matrix is processed separately before being concatenated.

attention, a masking matrix is introduced to the scaled dot-product attention mechanism prior to the softmax function computation. This masking matrix typically assumes the form of an upper triangular matrix, wherein entries corresponding to future inputs are assigned a negative infinity value or a significantly large negative value. This masking operation effectively nullifies their influence, resulting in attention scores of zero following the softmax function application.

Aside from the self-attention mechanism, the model also employs some intermediate layers that make data-processing more efficient and accurate. Residual connection is employed between each sub-layer so that the inputs have a path that connects directly to the output. Secondly, layer normalization is applied on the output of a residual connection block. This process normalizes inputs across the features of each individual training case. This means that normalization is not affected by the size of the mini-batch, making it ideal for online learning tasks or large distributed models with small mini-batches. Note that the residual connection and layer normalization generally make the calculations of the Transformer more efficient. Dropout was also utilized in the training process in order to prevent overfitting. This is a form of regularization where randomly selected neurons are (temporarily) ignored, with probability  $\delta$  (called the dropout rate), during training.

Similarly, a feedforward neural network is introduced in order to reduce dimensionality and further relate the attention scores with the corresponding inputs. Here the data is simply processed from the input layer, then through the hidden layers, and to the output layer (a softmax layer) without any feedback or loops in the data.

### 3 Methodology

To assess the feasibility of using the Transformer model for the stock index prediction task, the said model is applied to data from the PSEi under normal market conditions. Specifically, the study sourced data from Yahoo! Finance, comprising the closing prices of the relevant indices spanning from 2008 to 2022. In order to test the robustness of the model, stock index data from varying market conditions would be considered. In particular, three different tests would be conducted on the Transformer model. First, the historical PSEi data from 2018 to 2019 would

be used as input to the Transformer model. Next, stock index data in the same historical window from select neighboring countries would also be considered as input to the Transformer model. Finally, to test the performance of the model in bear market conditions, PSEi data from the 2008 financial crisis and the 2020 COVID-19 Pandemic would be utilized. Furthermore, in order to compare the Transformer model against the RNN, LSTM, and CNN, the same tests would be conducted for the other models. These models were implemented in Python 3 and used PyTorch 2 as the deep learning library. All relevant codes and programs can be found in this Github repository.

### 3.1 Model Implementation Overview

Implementation of the Transformer model was carried out as follows. First, the data was reindexed to incorporate missing weekend data. To address these gaps, the *forward fill* method was employed, propagating the last known valid observation to fill missing values. After gap filling, one-day logarithmic returns of the PSEi were calculated, serving as the final inputs for the model. The model was then trained using stock index data from 2010 to 2019. This process was repeated five different times (using five different seeds) for replicability and robustness. After implementing the Transformer model, the RNN, LSTM, and CNN models were implemented to allow benchmarking of results. Note that some of the hyperparameter values used for all four models were adapted from existing literature, such as [15]. Finally, metrics such as mean squared error (MSE), mean absolute error (MAE), root mean squared error (RMSE), and mean absolute percentage error (MAPE) were used for performance evaluation.

### 3.2 Testing with Regional Data and Stress Testing

Expanding the analysis to include varying market conditions would provide additional insights into the model's robustness. First, stock index data from Indonesia, Malaysia, and Thailand would be applied to the model. That is, historical data from 2018 to 2019 from the Jakarta Stock Exchange Composite Index (JKSE), FTSE Bursa Malaysia KLCI (FBM KLCI), and the Stock Exchange of Thailand (SET) index would be fed as inputs to the model. These three regional markets—Indonesia, Thailand, and Malaysia—were selected due to their geographic proximity to the Philippines, the market capitalization of their companies, and their classification as emerging markets. Second, PSEi data from the global financial crisis and the COVID-19 pandemic would also be used as inputs. Including in the analyses periods marked by significant economic events, such as the global financial crisis and the COVID-19 pandemic, could provide additional insights as to how the model behave in stress conditions. Furthermore, stress testing the model using these two bear market conditions could show the effects of non-normal market conditions to the limitations and robustness of the model.

## 4 Results and Discussion

Results of the four models under normal economic conditions can be seen in Table 1 and Figure 3. For the PSEi data from 2018 to 2019, the Transformer model outperformed the other three models based on the metrics considered. As depicted in Table 1, the Transformer model consistently outperformed the other three models across all evaluation metrics for stock

Table 1: Results of test data for Transformer, RNN, LSTM, and CNN models under normal conditions for the PSEi. In boldface are the better values for each metric.

Metric	Transformer	RNN	LSTM	CNN
MSE	<b>0.0031</b>	0.9774	0.9718	0.9664
RMSE	<b>0.0556</b>	0.9886	0.9858	0.9831
MAE	<b>0.0459</b>	0.7162	0.6711	0.6807
MAPE	<b>0.1830</b>	2.1076	1.3518	1.7747

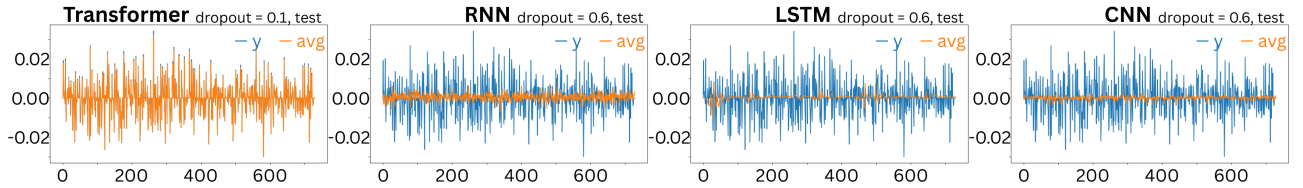


Figure 3: Results (from left to right) for Transformer, RNN, LSTM, and CNN on the test dataset. Actual log return values are in blue while predicted log return values are in orange.

index prediction. These findings underscore the superior accuracy and effectiveness of the Transformer architecture for this specific task. Furthermore, the Transformer had the lowest standard deviation for the MSE metric, indicating consistent performance across the different seeds.

#### 4.1 Comparison with Regional Market and Bear Market Data

Figure 4 presents the MSE bars of the four models when applied to stock index data from neighboring countries and different market conditions. For the regional market test, it was observed that the performance of the models have decreased across the board with error metrics that are higher overall. In all the scenarios, it was either the CNN or LSTM that had the best performance, depending on the evaluation metric. However, for the Malaysian stock index, the LSTM outperformed all other metrics across the board. Of note is the performance of the Transformer model across the different scenarios which showed the worst performance among the models. Notably, the Transformer has a relatively inconsistent MSE value for the different seeds provided when compared to the other models. Furthermore, the MSE of the LSTM seem to be the most consistent among the different models.

On the other hand, for the bear market test, results reveal notable variations in the performance of the models compared to their performance under normal conditions. In both crises and for all the models, all performance metrics worsened, as expected. Observing the standard deviations for MSEs, the Transformer also had the highest variations with this error metric. Meanwhile, the CNN generally had the best overall performance across the two scenarios with the RNN maintaining a performance advantage for some metrics. Furthermore, the standard deviation of the LSTM was the lowest, and consequently the most consistent among the models for the stress test.



Figure 4: MSE bars for the models used for regional indices (top) and bear market conditions (bottom) considered. Notice that the Transformer has inconsistent error values.

## 4.2 Discussion

It is worth noting that the results from Table 1 are in line with the expected performance from previous literature [15]. The Transformer model was able to better capture the patterns in the stock index data when compared to the other models. These results suggest that forecasting stochastic financial time series is a promising application of the Transformer model. The improved performance of the Transformer model over the traditional models may be attributed mostly to its multi-headed attention mechanism. This allows the Transformer to capture key characteristics within the data. This study suggests that the underlying mechanism of the Transformer is better at extracting relevant features over the mechanisms of the RNN, LSTM, and CNN which facilitates the better overall performance in forecasting [15].

Results of the regional market test indicate that in spite of the selection criteria, the markets of the different countries still fundamentally behaved differently. It was noted that the Transformer had the worst performance among the four models. This may be a sign of overfitting, as the Transformer model was not trained on data from other countries. Regardless, the high variance exhibited by the Transformer for these scenarios indicate that it is not robust to changes in market behavior. Similar results were also observed across both crises considered for the bear market test. In both instances, the Transformer exhibited the poorest performance among the four models, while the CNN and RNN demonstrated the strongest results under these stress conditions. These differing conditions may have influenced the Transformer’s performance on crisis data. However, these findings should be interpreted with caution, as they may represent outlier data. Future research could explore these results in greater depth.

## 5 Conclusion

In summary, this study demonstrated the feasibility of using the Transformer model to predict the stock index in the Philippines, despite its primary design for natural language processing. Although promising, these results are only indicative of the feasibility of using Transformer models for forecasting stocks in the Philippines. The tests conducted on neighboring markets reveal that this application of the Transformer model may not be robust with respect to varying market conditions; perhaps due to the data used or possibly overfitting issues. Similarly, when

subjected to data from bear market conditions, the Transformer model fails to maintain its performance advantage over the other models. Though, more research needs to be done in this area to better understand the limitations of the model with regards to stress testing.

Given its feasibility for forecasting financial time series, more advanced machine learning models can be explored to have better predictive performance for prediction tasks. A more robust hyperparameter selection process may also be conducted to potentially improve the performance of the Transformer. Nonetheless, while the primary objective of the model is to forecast stock indices, the framework considered for the Transformer model may be extended to individual or a portfolio of stocks. More importantly, this model may be used in conjunction with trading algorithms that base their decisions on this model. Given the advances in technology, applications of these state-of-the-art forecasting models and trading algorithms can be the start of a more robust quantitative finance industry in the Philippines.

In conclusion, this work highlights how advances in technology can also advance the computational methods for financial math and quantitative finance. Specifically, this paper has highlighted the Transformer model's capability to forecast financial time series, particularly in the domain of stock price prediction. By extension, this paper has also highlighted the feasibility of using machine learning models for forecasting financial time series from emerging markets. While traditional approaches rely on explicit probabilistic components to encapsulate uncertainty, the Transformer model, despite being deterministic in nature, demonstrates its proficiency in capturing uncertainty through its probabilistic outputs. This underscores the versatility of the Transformer architecture in effectively modeling complex temporal dependencies and capturing the inherent randomness present in financial data. By leveraging its ability to learn intricate patterns from sequential data, the Transformer model offers a promising avenue for advancing predictive modeling in finance, especially in emerging markets like the Philippines, contributing to more robust and accurate forecasting methodologies.

**Acknowledgements** The authors would like to thank the Ateneo de Manila University Department of Mathematics, and Juan Carlo Mallari, Aldo Zelig Tong, J. Lemuel Martin, and Alfonso Miguel Sevidal for their constructive feedback on this work.

## References

- [1] Sheng Chen and Hongxiang He, *Stock Prediction Using Convolutional Neural Network*, IOP Conference Series: Materials Science and Engineering, vol. 435, 2018, p. 012026.
- [2] Erkam Guresen, Gulgun Kayakutlu, and Tugrul Daim, *Using Artificial Neural Network Models in Stock Market Index Prediction*, Expert Systems with Applications **38** (2011), no. 8, 10389–10397.
- [3] Israt Jahan and Sayeed Sajal, *Stock Price Prediction Using Recurrent Neural Network (RNN) Algorithm on Time-Series Data*, Proceedings of Midwest Instruction and Computing Symposium, 2018.
- [4] Seyed Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker, *Time2Vec: Learning a Vector Representation of Time*, 2019.

- [5] Nadeem Malibari, Iyad Katib, and Rashid Mehmood, *Predicting Stock Closing Prices in Emerging Markets With Transformer Neural Networks: The Saudi Stock Exchange Case*, International Journal of Advanced Computer Science and Applications **12** (2021), no. 12, 876–886.
- [6] Tashreef Muhammad, Anika Aftab, Muhammad Ibrahim, Md Mainul Ahsan, Maishameem Muhi, Shahidul Khan, and Mohammad Alam, *Transformer-Based Deep Learning Model for Stock Price Prediction: A Case Study on Bangladesh Stock Market*, International Journal of Computational Intelligence and Applications **22** (2023), no. 03, 2350013.
- [7] Jigar Patel, Sahil Shah, Priyank Thakkar, and Ketan Kotecha, *Predicting Stock Market Index Using Fusion of Machine Learning Techniques*, Expert Systems with Applications **42** (2015), no. 4, 2162–2172.
- [8] Akshay Sachdeva, Geet Jethwani, Chinthakunta Manjunath, M Balamurugan, and Adapalli Krishna, *An Effective Time Series Analysis for Equity Market Prediction Using Deep Learning Model*, 2019 International Conference on Data Science and Communication, 2019, pp. 1–5.
- [9] Lounnapha Sayavong, Zhongdong Wu, and Sookasame Chalita, *Research on Stock Price Prediction Method Based on Convolutional Neural Network*, 2019 International Conference on Virtual Reality and Intelligent Systems, 2019, pp. 173–176.
- [10] Sreelekshmy Selvin, R Vinayakumar, E Gopalakrishnan, Vijay Menon, and K Soman, *Stock Price Prediction Using LSTM, RNN and CNN-sliding Window Model*, 2017 International Conference on Advances in Computing, Communications and Informatics, 2017, pp. 1643–1647.
- [11] Dev Shah, Haruna Isah, and Farhana Zulkernine, *Stock Market Analysis: A Review and Taxonomy of Prediction Techniques*, International Journal of Financial Studies **7** (2019), no. 2, 26.
- [12] Alaa Sheta, Sara Ahmed, and Hossam Faris, *A Comparison between Regression, Artificial Neural Networks and Support Vector Machines for Predicting Stock Market Index*, International Journal of Advanced Research in Artificial Intelligence **4** (2015), no. 7, 55 – 63.
- [13] J Urrutia, J Diaz, and E Baccay, *Forecasting Philippine Daily Stock Exchange Index*, Journal of Fundamental and Applied Sciences **9** (2017), no. 7S, 202–218.
- [14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin, *Attention is All You Need*, Advances in Neural Information Processing Systems, vol. 30, 2017.
- [15] Chaojie Wang, Yuanyuan Chen, Shuqi Zhang, and Qiuhui Zhang, *Stock Market Index Prediction Using Deep Transformer Model*, Expert Systems with Applications **208** (2022), 118128.