2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Stock Volatility Forecasting with Transformer Network

Golnaz Sababipour Asl
Department of Computer Science
University of Manitoba
Winnipeg, Canada
sababipg@myumanitoba.ca

Ruppa K. Thulasiram
Department of Computer Science
University of Manitoba
Winnipeg, Canada
tulsi.thulasiram@umanitoba.ca
ORCID ID: 0000-0002-6519-3929

Aerambamoorthy Thavaneswaran
Department of Statistics
University of Manitoba
Winnipeg, Canada
aerambamoorthy.thavaneswaran@umanitoba.ca

***Abstract*—Financial market is in general volatile with so many uncertainties and volatility is one of the main measures of uncertainty in the market among other measures. Hence, forecasting volatility is a critical component in risk management, optimizing portfolios, and in algorithmic trading among other financial problems. There have been few machine learning and artificial intelligence techniques used in the literature for the forecasting problem. Transformer Network (TN) architecture is one of newest such techniques proposed. In this work, we utilized this architecture with multi-head attention mechanism for volatility forecasting. To enhance the performance of the TN, we incorporated different variations of the feed forward layer. The performance of three distinct TN models was evaluated by implementing three different deep learning layers (CNN, LSTM, and a hybrid layer (CNN-LSTM)) in the encoder block of TN as the feed forward layer. The results clearly demonstrate that the TN model with the hybrid layer (CNN-LSTM) outperformed the other models, including a recently proposed data-driven approach.**

***Index Terms*—Volatility, Forecasting, Transformer Network, LSTM, Time2Vec**

I. Introduction

For financial modelers forecasting volatility is one of the biggest challenges due to its complex behaviour. It also presents a high degree of temporal variability. Traditional linear models like the autoregressive (AR) and autoregressive integrated moving average (ARIMA) models have certain limitation on constant variance of price fluctuation assumptions. One of the widely used models in volatility forecasting is generalized autoregressive conditional heteroscedasticity (GARCH) [1] which adds the variance lag term to ARCH model [3].

On account of the high level of complexity and non-linearity of financial markets, deep learning for forecasting has become one of the most attractive methods [21]. The two most popular methods for market forecasting are recurrent neural network (RNN) [16] and long short-term memory (LSTM) [6] models. Each of them has its own limitations. RNNs suffer from the vanishing and exploding gradients problem. LSTM models were introduced to address that problem.

Liu [14] used LSTM network for volatility forecasting and the result of this research demonstrated that LSTM has better performance than classical models. LSTM has a long memory capacity, however, as the input size increases past information could be forgotten [13]. To overcome those limitations, Vaswani et al. [19] presented a novel deep learning model called Transformer Network (TN). Adding positional encoding and attention mechanism made the TN different from traditional convolutional neural network (CNN) and RNN frameworks. Its unique architecture led to remarkable success in natural language processing (NLP) problems [2] and hence considered viable to apply them to forecast time series data as well. Our primary objective is to study TN for volatility forecasting. Major contributions from this study include (i) assessing the viability and performance of TN architecture for volatility forecasting and (ii) compare and contrast the performance of TN architecture with other approaches in general and specifically with a recently proposed data-driven approach [18]. The rest of this manuscript is organized as follows: Following the literature review in section II, the implementation details of TN for the current problem are described in section III. Section IV describes the algorithms for transformer network used in this study. The experiments and results are presented in Section V, discussions in section VI, and conclusion in section VII.

II. Related Work

Using deep learning models for forecasting has become popular due to their performance as compared to traditional models such as GARCH models. Xu et al. [22] adopted a new hierarchical graph neural network (HGNN) to analyse the market state from a hierarchical perspective for stock type forecasting. For the purpose of graph construction, they included both historical sequence patterns and stock relationships in their analysis. The result demonstrated the outperformance of HGNN over other the state-of-the-art solutions including attentive long short-term memory (ALSTM) [4], graph convolutional network (GCN) [11], and graph attention network (GAT) [20] models. Jiang [8] presented a review on the studies that used deep learning models for stock market forecasting during 2017-2019 in order to provide a comprehensive view for researchers interested in this area. Filipovic and Khalilzadeh [5] examined the ability of machine learning methods in forecasting the realized volatility of stock returns.

The study found that machine learning methods have great potential in forecasting risk in the stock market and showed that LSTM architecture provided more accurate forecasting of volatility compared to other machine learning methods. While some researchers used TN in sentiment analysis by analysing comments of users in social networks and financial news [12], [15], [23], others have applied them to time-series forecasting. Sridhar and Sanagavarapu [17] used time-series forecasting as an essential tool for analyzing the fluctuations in the bitcoin and altcoin markets. The performance of the model was evaluated using several metrics, including Root Mean Square Error (RMSE), Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared value, and demonstrated an impressive accuracy comparable with other cryptocurrency price forecasting models. Zhou et al. [24] proposed a novel deep learning model called T2V-TF. They mentioned that existing methods for stock forecasting tend to rely on limited features and importance of multi-source information fusion has been ignored. To address these limitations, they proposed T2V-TF that combines Time2Vec and Transformer network. The research validated the proposed model on a portfolio of stocks from the Chinese stock market and demonstrates that T2V-TF outperformed other traditional models. Wang et al. [21] used daily closing price to forecast stock market indexes in future. They evaluated the performance of TN model from two points of view: forecasting accuracy and net value analysis. The results showed that TN model had the highest Sharpe ratio and total return in comparison to other models.

III. Time Embedding in TN

The historical price of S&P 500 index is utilized to measure the efficiency of the proposed methodology. The data set used in this study consisted of daily data of the S&P 500 index from 1982 to 2023, downloaded from Yahoo! Finance. It comprised of 10,337 rows and 7 columns, namely Date, High, Low, Open, Close, Adj Close, and Volume.

A. *Time Embedding*

To establish the temporal relationships within each input sequence, the TN requires embedded time vectors. Several studies have utilized Time2Vec to produce time-embedded vectors for forecasting models in time series forecasting [17], [24]. Time2Vec, introduced in [9], plays a crucial role in implementing the TN for time-series data. It offers a vector representation of time that encompasses both periodic and non-periodic patterns, ensuring invariance to time rescaling. The findings of the study [9] demonstrated that replacing the concept of time with its Time2Vec representation improves the performance of the final model across various models and problems. Time2Vec denoted as $t2v(\tau)$, is mathematically computed as follows [9]:

$$t2v(\tau) = \begin{cases} \omega_i \tau + \varphi_i & \text{if} \quad i = 0 \\ F(\omega_i \tau + \varphi_i) & \text{if} \quad 1 \leq i \leq K \end{cases}$$

The periodic feature consists of a linear function that is contained within a function F while the non-periodic feature is a linear function with slope ω and intercept φ. By passing the input data sequence through the $t2v(\tau)$ function both periodic and non-periodic features are computed. The features are merged with the input value sequence and are provided to the TN model. In the subsequent step, the computed time features are concatenated with the volatility feature forming a matrix.

B. *Transformer Network*

The TN model offers a unique methodology by avoiding recursion and instead relies extensively on the attention mechanism to capture overall connections between input and output. Embeddings layer, positional encoding, encoder-decoder, multi-head attention, and feed forward network are important components of TN. To enhance the model's flexibility, we made some modifications to the original TN architecture. First, in this study, we have substituted positional encoding with Time2Vec. Second, we modified the decoder to better align with our specific requirements. In the original TN design, both the encoder and decoder blocks consist of two sub-layers: a multi-head attention (MHA) layer and a feed forward network (FFN). In this research, the decoder section has significant alterations, incorporating global average pooling, two dropout layers, and two dense layers, with the final dense layer yielding the output. The encoder unit comprised of MHA and FFN layers. In encoder block, MHA model followed by dropout, layer normalization, LSTM layers, dropout, and layer normalization. The first LSTM layer consisted of 256 units, both LSTM layers utilized the hidden layer activation function (tanh). The input to the encoder involved embedding the input sequence using Time2Vec vectorization. The encoder produced a fixed-length output, before feeding into the decoder. Ultimately, the final dense layer in the decoder is responsible for generating the model's output. Figure 1 demonstrates the TN Model of this research by applying LSTM layer as FFN layer. There are different hyperparameters in TN. The mean squared error is used as the loss function. The Adam optimizer [10] is used to train the model. In order to prevent overfitting, a dropout technique with a rate of 0.1 is utilized for each sub-layer. Models are evaluated with MAE, and MAPE metrics. Batch-size in this research was 32, dimension of key and value for attention layers were 256, and 8 heads were employed for MHA layer. The encoder block is made up stack of 3 layers. All models are fit with 50 epochs.

IV. Transformer Network Algorithms

A clear and concise representation of the algorithm designed for this research problem is presented in Algorithms 1 and 2. S_L refers to the length of the input sequence (S_L = 12) and d_m is the number of data points of input. MHA involves the use of multiple attention functions running in parallel. In MHA, there are h attention layers that run simultaneously (h is equal to 8). Algorithm 1 demonstrates the MHA function.
Processing financial data using an encoder and decoder block for forecasting stock volatility is outlined in Algorithm 2. The first step is to compute the return of a stock. After that the

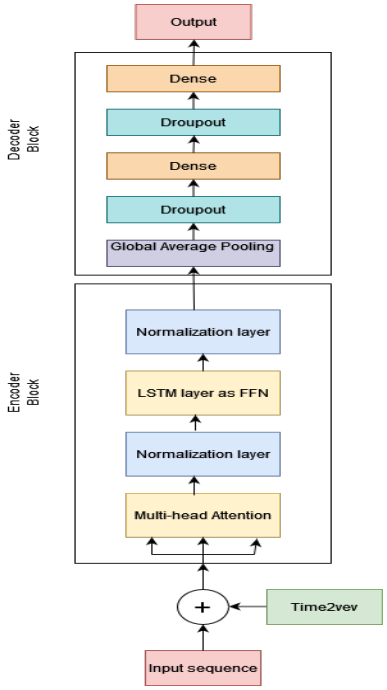

Fig. 1. Implementation of TN Model by applying LSTM layer as feed forward layer

Algorithm 1 Multi-head attention Function

Multi-head Attention Layer (q, k, v)
Input: tensor q,k,v $\in R^{d_m \times S_L}$
output: tensor $y' \in R^{d_m \times S_L}$
$q', k', v' = W_q x + b_q, W_k x + b_k, W_v x + b_v$

$$q' = \begin{bmatrix} \mathrm{q}'_1 \\ \vdots \\ \mathrm{q}'_h \end{bmatrix}, \quad k' = \begin{bmatrix} \mathrm{k}'_1 \\ \vdots \\ \mathrm{k}'_h \end{bmatrix}, \quad v' = \begin{bmatrix} \mathrm{v}'_1 \\ \vdots \\ \mathrm{v}'_h \end{bmatrix}$$

$$y = \begin{bmatrix} Attention(\mathrm{q}_1, k_1, v_1) \\ \vdots \\ Attention(\mathrm{q}_h, k_h, v_h) \end{bmatrix}$$

$y' = W_y y + b_y$
return y'

volatility of the stock is computed. Volatility is computed using the standard deviation, employing a window size of 5. The resulting volatility values are then added into the dataset. Dataset is split into train, validation, and test data. After applying Tme2Vec method, the encoder block processes the input data applying a MHA mechanism to the data, then passing the result through a FFN layer for which three different variations of the FFN, (namely CNN, LSTM, and a hybrid approach combining both (CNN-LSTM)) instead of relying solely on a simple linear layer have been incorporated. The encoder layer count is set at 3, and this process is iterated three times using the EncoderUnit function. The decoder block takes the output of the previous encoder block with global average pooling, two dropout layers, and two dense layers, with the final dense layer yielding the output. The three metrics MSE, MAE, and Mean Absolute Percentage Error (MAPE) are used

Algorithm 2 Transformer Network is used for Stock Volatility Forecasting

Data: Adj Close price of stock, $P_t, t = 0, ..., k, ...T_1$
1. Compute return, volatility

$$r_t \leftarrow \frac{P_t - P_{t-1}}{P_{t-1}},\ t = 1, ..., T_1$$

$vol_t \leftarrow$ Standard Deviation of r_t
2. Normalize data, split data
Use Min-max normalization to normalize vol_t
Split the data vol_t sequentially into 70% train, 15% validation, 15% test
3. Encoder input data, positional encoding
training data are separated into sliding window sequences with a length of 12 days (S_L = 12)
for i in range(seq-len, len(train-data)) **do**
X-train.append(train-data [i - seq-len :i])
Y-train.append(train-data[i])
end for
Apply the Time2Vec method to generate the positional encoding for each X[t], i.e., t2v(X[t]) $\in R^{d_m \times S_L}$.
4. Encoderblock
Input: tensor x $\in R^{d_m \times S_L}$
output: tensor x $\in R^{d_m \times S_L}$
Function EncoderUnit(x):
x' = layerNorm(x + Multi-head Attention (x, x, x))
z' = layerNorm(x' + feedforward(x'))
return z'
for $i = 1, \ldots$ N **do**
x = EncoderUnit(x)
end for
return x
5. Decoderblock
Input: tensor x $\in R^{d_m \times S_L}$
output: tensor $z' \in R^{d_m \times 1}$
Function DecoderUnit(x):
x' = Global Average Pooling(x)
x' = Dropout(x')
x' = Dense(x')
x' = Dropout(x')
z' = Dense(x')
return z'

to measure the accuracy of models from different perspectives.

V. Experiments and Results

In this research, the first approach univariate volatility forecasting involves using only the volatility feature. There are two variations within this approach to compute volatility:

1) Using the Open price.
2) Using the Adj Close price.

The second approach is multivariate volatility forecasting that utilizes all available features. In this approach, two specific features are considered:

1) Using the Adj Close price.
2) Using the Open price.

A. *First approach: Univariate Volatility Forecasting*

1) Using Open Price for computing volatility: By utilizing the random forest model and analyzing the feature importance, we determined that the Open price and Adj Close price are two most significant factors in the provided time

TABLE I

Models	MSE	MAE	MAPE
Transformer Network(LSTM layer)	1.1350e-05	0.0019	30.3417
Transformer Network(CNN layer)	4.6428e-05	0.0049	82.4246
Transformer Network(CNN-LSTM layer)	1.0810e-05	0.0018	29.5690

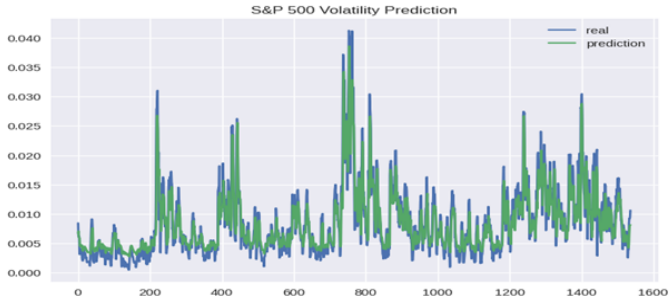

Fig. 2. Forecasting result of volatility of S&P 500

TABLE II

Models	MSE	MAE	MAPE
Transformer Network(LSTM layer)	1.1642e-05	0.0022	28.2502
Transformer Network(CNN layer)	0.0001	0.0051	68.5471
Transformer Network(CNN-LSTM layer)	1.0801e-05	0.0021	26.9638

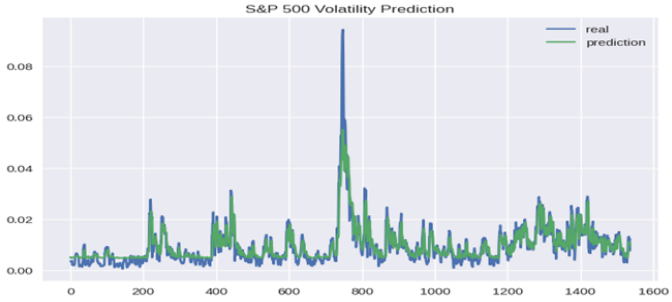

Fig. 4. Forecasting result of volatility of S&P 500

series data. Open price was used as it emerges as the most significant feature according to the feature importance analysis for volatility forecasting. Table I provides the result of test data by employing three distinct FFN layers configurations: CNN, LSTM, and a hybrid approach that combines both (CNN-LSTM). In Figure 2, we present a visual comparison of the actual volatility, with the corresponding forecasting values. This plot aims to provide a comprehensive assessment of the model performance in capturing the true fluctuations in volatility over time. Furthermore, Figure 3 illustrates a plot of the MSE loss over 50 epochs. The smooth and progressively decreasing MSE loss plot indicates a well-optimized training process, where the model effectively adjusts its parameters to minimize forecasting errors. By examining the MSE loss plot in conjunction with the visual comparison of actual and forecasted volatility values, a more comprehensive evaluation of the model's performance can be achieved.

2) Using Adj Close Price for computing volatility: Alternatively, we employed the Adj close price instead of the Open price for volatility forecasting, the same model configuration and evaluation metrics were utilized. The result as presented in Table II, showing the performance of the three FFN layers configurations: CNN, LSTM, and the hybrid CNN-LSTM approach. Figure 4 presents a visual comparison of the actual volatility value with the corresponding forecasted values. This plot aims to offer a thorough evaluation of the model's ability to capture the true fluctuations in volatility over time. Additionally, Figure 5 displays a plot depicting MSE loss over 50 epochs. A smooth and steadily declining MSE loss curve serves as an indicator of a training process characterized by effective parameter optimization, resulting in minimized forecasting errors.

B. Second approach: Multivariate Volatility Forecasting

1) Using Adj close Price for computing volatility: In this step, we used all seven features for S&P 500 volatility forecasting by implementing TN. To accommodate these differing input setups, we made modifications to the input sequence and adjusted the input shape within our TN. To enhance the model's performance, we employed moving average [7] to our dataset to reduce noise and minimize short-term fluctuations that are inherent in the time series data. We then assessed the performance of our three different models by applying CNN, LSTM, and hybrid (CNN-LSTM) instead of the simple feed forward layer. Subsequently, we obtained the results presented in Table III for the test data by employing three

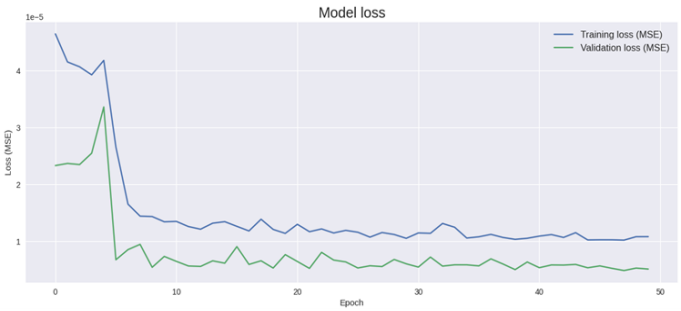

Fig. 3. Model loss by applying TN (CNN-LSTM as FFN layer)

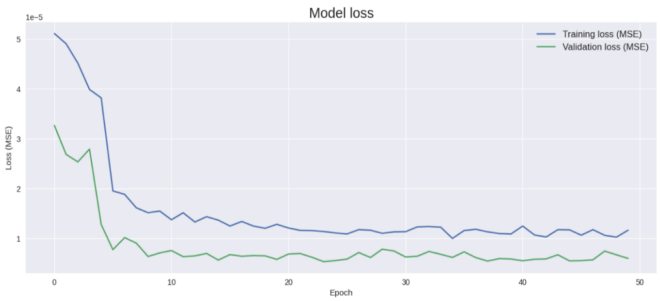

Fig. 5. Model loss by applying TN (LSTM as FFN layer))

TABLE III

Models	MSE	MAE	MAPE
Transformer Network(LSTM layer)	4.7215e-06	0.0018	35.3797
Transformer Network(CNN layer)	2.1032e-05	0.0032	75.5001
Transformer Network(CNN-LSTM layer)	4.0552e-06	0.0016	33.0305

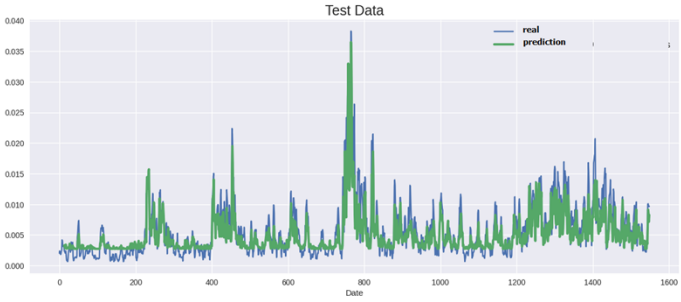

Fig. 6. Forecasting result of volatility of S&P 500

different layer configurations. Figure 6 demonstrates the visual comparison of the actual volatility and corresponding predicted values, utilizing all feature as input for the model. This plot offers a comprehensive evaluation of the model's performance in accurately capturing the volatility fluctuations over time. Figure 7 illustrates the MSE loss plot obtained from training the TN model with all seven features over 50 epochs. A steady and uniform reduction in MSE loss is a testament to the efficacy of the training process, demonstrating the model efficiently adjusts its parameters to minimize prediction errors.

2) Using Open Price for computing volatility: In this step, the Open price is used to calculate volatility. The rest of the steps, including incorporating all seven features for S&P 500 volatility prediction, were replicated as in the previous experiment. We implemented a TN and applied three distinct FFN layers, namely CNN, LSTM, and hybrid (CNN-LSTM). Table IV presents the results of performance comparison of the test data by employing three different layer configurations.

C. Comparision with LSTM and DDEWMA Models

We used Time2Vec with LSTM Network in order to compare its result with TN. By incorporating the time vector

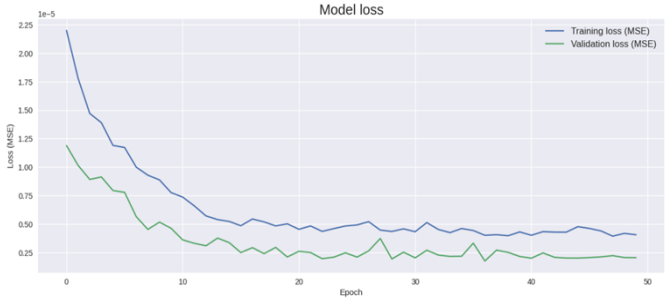

Fig. 7. Model loss by applying Transformer Network using hybrid (CNN-LSTM) as FFN layer

TABLE IV

Models	MSE	MAE	MAPE
Transformer Network(LSTM layer)	2.4181e-06	0.0016	51.9926
Transformer Network(CNN layer)	2.0115e-05	0.0056	92.1289
Transformer Network(CNN-LSTM layer)	2.7377e-06	0.0015	49.9926

TABLE V

Model	MSE	MAE	MAPE
LSTM (using one feature)	1.1678e-05	0.0021	28.0900

through Time2Vec, the LSTM model gains the ability to capture and utilize temporal patterns effectively. We implemented LSTM Network in two steps. Initially, we input only the volatility feature into the Time2Vec with LSTM model, while in the second step, we incorporated all features. To ensure a fair assessment between the LSTM model and the TN model, we employed identical evaluation metrics as those used for the TN model. This methodology enabled us to effectively evaluate and compare the performance of both models. Table V demonstrates the results of implementing Time2Vec with LSTM model. By utilizing all features as input for our Time2Vec+LSTM model, we obtained the results presented in Table VI.

Second model is data-driven exponentially weighted moving average (DDEWMA). The key in the function DDEWMA is to loop through alpha in order to select the alpha that generates the lowest MSE. Alpha represents the optimal smoothing constant, determined by minimizing one-step ahead forecast errors. We used 12-day historical window of data for fitting our optimal alpha. We utilized the Adj Close price to calculate volatility, and solely employed the volatility feature as input for the DDEWMA model. Table VII presents the results of the three metrics, providing an overview of the model's performance.

VI. Discussions

We proposed a TN with a multi-head attention (MHA) mechanism architecture for the task of forecasting stock volatility. To enhance its capabilities in this research, we integrated three different variations of the feed forward layer—namely CNN, LSTM, and a hybrid approach combining both CNN-LSTM —instead of relying solely on a simple linear layer. For initial approach univariate volatility forecasting, we applied two different data for calculating volatility.

TABLE VI

Model	MSE	MAE	MAPE
LSTM (using seven features)	3.3587e-05	0.0041	57.7487

TABLE VII

Model	MSE	MAE	MAPE
DDEWMA (using one feature)	2.69093e-05	0.0037	56.3953

These data included using the Open price and the Adj close price. Their corresponding results presented in Table I and Table II Across both cases, it is evident that using the hybrid (CNN-LSTM) architecture as feed forward networks (FFN) in the TN, outperforms other models. This is supported by its lowest MSE, MAE, and MAPE values. The LSTM-based FFN also demonstrated competitive results, albeit with slightly higher MSE, MAE, and MAPE values. In contrast, the CNN-based FFN variant showed higher MSE, MAE, and MAPE values, indicating relatively lower accuracy in capturing the true fluctuations over time. Based on the result of comparison obtained by using the Open price and Adj Close price led to compute volatility, we can conclude:

1) utilizing the Adj Close price more suitable for capturing volatility patterns.
2) the hybrid model architecture incorporating both CNN and LSTM layers as FFN layer is effective for volatility forecasting.

By closely analysing Figure 2 and Figure 4, it becomes apparent that incorporating the Adj Close price for volatility forecasting adeptly captures the inherent fluctuations and patterns within the domain of volatility forecasting. The predicted values closely correspond to the actual values, exemplifying a remarkable precision in capturing the dynamics of volatility. While it is acknowledged that certain instances exist where the actual and predicted values do not align perfectly, a consistent pattern of upward and downward movements can be observed in both, particularly within the majority of the test data. This observation demonstrates that the model possesses a commendable capacity to capture the broader trends and shifts in volatility, thereby demonstrating its efficacy in forecasting volatility patterns.

The smooth and steadily decreasing MSE loss plots in Figure 3 and Figure 5, indicate a well-behaved training process where the model successfully optimizes its parameters to minimize prediction errors. Analyzing the MSE loss plot alongside the visual comparison of actual and predicted volatility values allows for a more comprehensive assessment of the model's performance. By considering all the results, we can conclude that the hybrid (CNN-LSTM) architecture within the TN outperforms other models in volatility forecasting. For the second approach multivariate volatility forecasting, based on the result of the MSE, MAE, and MAPE presented in Table III and Table IV, it becomes evident that employing the Adj Close price for volatility predictions yields superior results in comparison to Open price. Furthermore, it's worth noting that univariate volatility forecasting outperforms multivariate volatility forecasting.

To conduct a comparison between the Time2Vec+LSTM model and the Time2Vec+TN model, we utilized identical evaluation metrics as employed for the TN model. The results demonstrated the strong performance of the Time2Vec+LSTM model on the dataset, as evidenced by the low values obtained for both MSE and MAE when using only one feature. These metrics indicate that the model's predictions align closely

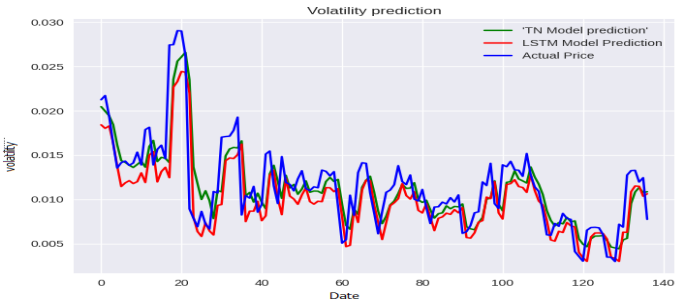

Fig. 8. Comparing the performance of two models in volatility forecasting

with the actual values, highlighting its accuracy. Furthermore, when comparing to the results achieved by the TN model using either LSTM or hybrid (CNN-LSTM) layers as the FFN layer, the LSTM model performs competitively. The results obtained from these models are very close to each other, indicating that the LSTM model is similar in performance to the TN model. Figure 8 provides a comparison between the TN model and the LSTM model in terms of their ability to predict volatility using a limited test dataset (zooming on 140 Data Points). The plot clearly illustrates that both models exhibit promising performance in capturing the ups and downs of volatility. They demonstrate a notable ability to track and predict the fluctuations in volatility, showcasing their effectiveness in capturing the overall trends in the data. Based on the results presented in Table II, Table V, and Table VI, it can be concluded that the LSTM model demonstrates strong performance in univariate volatility forecasting comparable to the performance of the TN model. However, as the complexity of the dataset increases, along with the length of data and the number of features, the LSTM model shows poorer performance. Hence, it can be inferred that the Transformer Network model is better equipped to handle the complex dataset, longer sequences of data, and multiple features, making it more suitable for stock volatility prediction in such cases. The Table VII demonstrates DDEWMA model showed the unsatisfactory results in effectively capturing the volatility projection of the S&P index by displaying higher MAE and MAPE scores. We compared the results achieved by the TN model, the LSTM model, and DDEWMA model in Figure 9 focusing on a limited test dataset (zooming on 140 data points). Table VIII presents the results of the three metrics of three models.

TABLE VIII

Model	MSE	MAE	MAPE
DDEWMA	2.69093e-05	0.0037	56.3953
LSTM	1.1678e-05	0.0021	28.0900
Transformer Network(CNN-LSTM layer)	1.0801e-05	0.0021	26.9638

VII. Conclusions

In this research, we proposed a Transformer Network (TN) with a multi-head attention (MHA) mechanism for forecasting

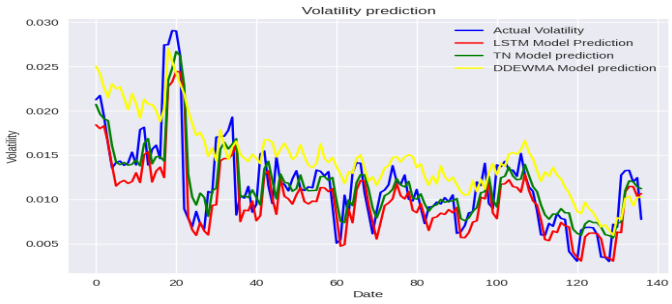

Fig. 9. Comparing the performance of three models in volatility forecasting

stock volatility. To improve the model's performance, we incorporated three variations of the feed forward network (FFN) layer: CNN, LSTM, and a hybrid approach combining both (CNN-LSTM). We conducted experiments using different data for computing volatility, including the Open price and the Adj Close price. The results showed that the TN with the hybrid CNN-LSTM layer as FFN layer outperformed other models. When comparing the result of univariate volatility forecasting versus multivariate volatility forecasting, it was found that employing the Adj Close price alone yielded superior results for forecasting the volatility of the S&P 500 index in both of them. Furthermore, it is important to highlight that the utilization of univariate volatility forecasting surpasses the performance of multivariate volatility forecasting.

In the first comparison approach, we compared three models: DDEWMA, the Time2Vec+LSTM model, and the Time2Vec+TN model. We focused on using only one feature for volatility forecasting. Based on the results, both the LSTM and TN models showed a significant ability to predict volatility. However, the DDEWMA model performed comparatively worse than the others.

In the second comparison approach, we again compared the Time2Vec+LSTM model and the Time2Vec+TN model, but this time we considered all seven features. While the Time2Vec+LSTM model exhibited a remarkable ability to track and predict volatility fluctuations using a single volatility feature, its performance declined when multiple features were involved. On the other hand, the TN model with MHA proved to be more suitable for handling complex datasets and multiple features. Therefore, in scenarios involving stock volatility forecasting with various features, the TN model with MHA is the preferred choice.

ACKNOWLEDGMENT

The first author acknowledges the Research Assistantship from Prof. Thulasiram and Graduate Enhancement of Tri-agency Stipends (GETS) from University of Manitoba. The last two authors acknowledge the Natural Sciences and Engineering Research Council (NSERC) of Canada for Discovery Grants.

REFERENCES

[1] BOLLERSLEV, T. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics 31*, 3 (1986), 307–327.

[2] CHERNYAVSKIY, A., ILVOVSKY, D., AND NAKOV, P. Transformers: "the end of history" for natural language processing? In *Machine Learning and Knowledge Discovery in Databases. Research Track* (Cham, 2021), N. Oliver, F. Pérez-Cruz, S. Kramer, J. Read, and J. A. Lozano, Eds., Springer International Publishing, pp. 677–693.

[3] ENGLE, R. F. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica 50*, 4 (July 1982), 987.

[4] FENG, F., CHEN, H., HE, X., DING, J., SUN, M., AND CHUA, T.-S. Enhancing stock movement prediction with adversarial training. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19* (7 2019), International Joint Conferences on Artificial Intelligence Organization, pp. 5843–5849.

[5] FILIPOVIĆ, D., AND KHALILZADEH, A. Machine learning for predicting stock return volatility. *Swiss Finance Institute Research Paper*, 21-95 (2021).

[6] HOCHREITER, S., AND SCHMIDHUBER, J. Long short-term memory. *Neural computation 9* (12 1997), 1735–80.

[7] HYNDMAN, R. J. *Moving Averages*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 866–869.

[8] JIANG, W. Applications of deep learning in stock market prediction: Recent progress. *Expert Systems with Applications 184* (2021), 115537.

[9] KAZEMI, S. M., GOEL, R., EGHBALI, S., RAMANAN, J., SAHOTA, J., THAKUR, S., WU, S., SMYTH, C., POUPART, P., AND BRUBAKER, M. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321* (2019).

[10] KINGMA, D. P., AND BA, J. ADAM: A method for stochastic optimization, 2014.

[11] KIPF, T., AND WELLING, M. Semi-supervised classification with graph convolutional networks. international conference on learning representations, 2017.

[12] LI, Y., LV, S., LIU, X., AND ZHANG, Q. Incorporating transformers and attention networks for stock movement prediction. *Complexity 2022* (2022).

[13] LIN, H., AND SUN, Q. Financial volatility forecasting: A sparse multi-head attention neural network. *Information 12*, 10 (2021).

[14] LIU, Y. Novel volatility forecasting using deep learning–long short term memory recurrent neural networks. *Expert Systems with Applications 132* (2019), 99–109.

[15] MISHEV, K., GJORGJEVIKJ, A., VODENSKA, I., CHITKUSHEV, L. T., AND TRAJANOV, D. Evaluation of sentiment analysis in finance: From lexicons to transformers. *IEEE Access 8* (2020), 131662–131682.

[16] RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. Learning Representations by Back-propagating Errors. *Nature 323*, 6088 (1986), 533–536.

[17] SRIDHAR, S., AND SANAGAVARAPU, S. Multi-head self-attention transformer for dogecoin price prediction. In *2021 14th International Conference on Human System Interaction (HSI)* (2021), IEEE, pp. 1–6.

[18] THAVANESWARAN, A., PASEKA, A., AND FRANK, J. Generalized value at risk forecasting. *Communications in Statistics-Theory and Methods 49*, 20 (2020), 4988–4995.

[19] VASWANI, A., SHAZEER, N. M., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention is all you need. *ArXiv abs/1706.03762* (2017).

[20] VELICKOVIC, P., CUCURULL, G., CASANOVA, A., ROMERO, A., LIO, P., AND BENGIO, Y. Graph attention networks. *stat 1050*, 20 (2017), 10–48550.

[21] WANG, C., CHEN, Y., ZHANG, S., AND ZHANG, Q. Stock market index prediction using deep transformer model. *Expert Systems with Applications 208* (2022), 118–128.

[22] XU, C., HUANG, H., YING, X., GAO, J., LI, Z., ZHANG, P., XIAO, J., ZHANG, J., AND LUO, J. HGNN: Hierarchical graph neural network for predicting the classification of price-limit-hitting stocks. *Information Sciences 607* (2022), 783–798.

[23] ZHANG, Q., QIN, C., ZHANG, Y., BAO, F., ZHANG, C., AND LIU, P. Transformer-based attention network for stock movement prediction. *Expert Systems with Applications 202* (2022), 117239.

[24] ZHOU, F., ZHANG, Q., ZHU, Y., AND LI, T. T2v_tf: An adaptive timing encoding mechanism based transformer with multi-source heterogeneous information fusion for portfolio management: A case of the chinese a50 stocks. *Expert Systems with Applications 213* (2023), 119020.