# A Lightweight Multi-Head Attention Transformer for Stock Price Forecasting

Anh Q. Nguyen[1], Son X. Ha[1], and Nguyen Ngoc Phien[2,3]

[1] Department of Economics and Finance, RMIT University Vietnam, Ho Chi Minh City, Vietnam
s3926339@rmit.edu.vn, ha.son@rmit.edu.vn
[2] Center for Applied Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[3] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam
nguyenngocphien@tdtu.edu.vn

**Abstract.** Despite the growth and implementation of AI in live trading machines in the financial industry, predictive models face challenges in dissecting and capturing the chaotic nature of stock prices. Hence, this research proposes a distinctive lightweight Transformer model with a sustainable architecture consisting mainly of positional encoding and advanced training techniques to mitigate model overfitting, hence offering prompt forecasting results through a univariate approach to the closing price of stocks. Employing MSE for loss alongside MAE and RMSE as core evaluation metrics, the proposed Transformer consistently surpasses renowned forecasting models such as LSTM, SVR, CNN-LSTM, and CNN-BiLSTM-ECA, averaging a reduction in forecasting errors by over 50%. Being trained across AMZN, INTC, CSCO, and IBM 20-year daily stock datasets, the Transformer demonstrates a high degree of accuracy in capturing flash crashes, cyclical or seasonal patterns, and long-term dependencies inherent in tech stocks. Moreover, it only takes the model 19.36 seconds to generate forecasting results on a non-high-end local machine, fitting into the 1-minute trading window. To our knowledge, this lightweight approach is highly unparalleled in stock price forecasting.

**Keywords:** Lightweight Transformer · Positional Encoding · Univariate Approach · Time Series · Flash Crashes

## 1 Introduction

In contemporary industrialization marked by swift advancements in technology, the application of Artificial Intelligence (AI) across sectors is notable. AI leverages neural network architectures such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTM), or Transformer to improve Machine Learning (ML) processes in computer vision tasks [1], employing Deep Learning (DL) mechanisms for cancer detection [2] and weather prediction [3]. Likewise, the finance sector, revitalized

by ML in the digital economy, now draws significant interest in trading financial assets with algorithms [4]. Moreover, the post-pandemic economic rebound presents fresh opportunities in corporate stock investments, prompting investors to leverage predictive modeling for optimal profitability in financial trading.

Accordingly, traditional models such as ARIMA (Autoregressive Integrated Moving Average), SARIMA (Seasonal ARIMA), or Linear Regression often rely on the premise that past behavior and trends can predict future stock prices [5,6]. Nevertheless, the inherent non-linear characteristics of stocks introduce significant biases within these singular predictive models due to the assumption of fixed data variance or data linearity, overlooking decisive factors that influence stock prices longitudinally (e.g., macroeconomic indicators, market sentiment, herd behavior) [7]. Nonetheless, the advent of ML and DL models handles the aforementioned limitations. Techniques such as Support Vector Machines (SVM), Decision Trees, or Random Forest Regression (RFR) are able to incorporate multiple features thus capturing non-linear relationships through complex network architectures, enabling them to learn larger sets of data alongside increasing input features [8]. DL, as part of ML, later leverages the forecasting field with its deep neural architectures in CNN, RNNs, or LSTM, applying hierarchical end-to-end learning to solve unstructured data at greater scalability and performance in accordance with increasing size of time series data points [9]. Moreover, hybrid architectures between conventional statistical, ML, and DL models also renovate the forecasting field with dynamic multi-modal structures, improving feature extraction and drawing significant insights from data abnormality [10]. However, while ML/DL models might improve forecasting accuracy, they introduced challenges related to overfitting, interpretability, and the need for extensive data preprocessing. Additionally, RNNs-based variants, despite their capability of handling sequence data, often struggled with long-term dependencies due to issues like vanishing gradient [11]. Hence, such complex frameworks demand high computational resources, raising overheads for per-minute forecasting, and potentially limiting their long-term viability in real-time trading systems.

Therefore, this paper proudly introduces a unique lightweight multi-head Transformer model, equipping positional encoding, an optimized encoder architecture that includes multi-head self-attention mechanisms, and feed-forward neural networks, particularly designed for univariate time series forecasting and analysis. This eliminates the need for conventional Transformer components (e.g., token embeddings, decoder mechanisms, masked self-attention, etc.) seen in NLP tasks thus refining the model for continuous numerical predicting and reducing the complexity of architectures to provide instant financial decisions. We believe that the proposed Transformer model could capture market shocks hence reducing market bias and mitigating forecasting errors by efficiently learning only the closing prices of stocks, which wholly reflect the latest market condition.

This paper is constructed as follows: Section 2 reviews previous work in the financial forecasting field; Section 3 introduces data collection and evaluation processes alongside the proposed Transformer architecture; Section 4 conducts models' comparison, performance, and results before visualizing and interpreting

forecasting accuracy in Section 5; Section 6 draws insights and limitations from the research thereby giving solutions for real-time enhancement in Section 7. This empirical research stems from the foundation laid by our previous studies [12–14], and represents our most accurate findings to date in the algorithmic trading field.

## 2 Literature Review

The stock market is a multifaceted landscape, influenced by uncorrelated factors ranging from psychology to economics. Statistical models, due to their straightforward architectures and assumption of data linearity, may fall short in capturing the interrelationships of stocks. Therefore, Machine Learning (ML) in statistics emerges, offering adaptive learning from multivariate data patterns [15–18].

### 2.1 Related Work

Gui & Wu [15] applied the Back Propagation Neural Network (BP-NN) Model to forecast the 1-year stock price of Chinese vaccine manufacturers, resulting in BP-NN surpassing ARIMA and RFR in Root Mean Square Error (RMSE) and R-Square metrics by 30% due to BP-NN's deep layered architecture and nonlinear activation functions, actively adjusting weights via backpropagation. Kumar et al. [16] then combined Seasonal ARIMA and Extreme Gradient Boosting, termed SARIMA-XGBoost, to predict the Indian Stock Index, which returned an accuracy of 89.48% and a Mean Absolute Error (MAE) of 0.016, showing the crucial role of SARIMA in handling data seasonality that could later benefit ML models. Likewise, Zhao [17], after comparing Long Short-Term Memory (LSTM), RFR, and ARIMA algorithms in forecasting sets of Chinese stock samples, concluded that while ML frameworks offer greater forecasting accuracy as they tackled data non-linearity better, their architectures are highly complex with extravagant overheads in long-term processing, whilst, ARIMA provides simpler and quicker predictions but at lower precision. Nevertheless, Singh [18] pointed out that conventional ML models, with a limited number of layers and their heavy reliance on manual feature engineering, often struggle with capturing complex patterns in large datasets that offer high dimensionality. This research gap is later filled by Deep Learning (DL) integration, as a subset of ML [19,20].

Deep Learning, with its hierarchical structures and ability to learn feature representations using deep neural networks, offers a powerful solution for handling and extracting meaningful insights from chaotic historical data patterns [11]. Accordingly, LSTM and CNN architectures were found to perfectly capture the long-term stock volatility, whilst LightGBM worked better in the short term [21]. This is due to LSTM memory cells and CNN's convolutional layers that allow them to recognize long-term dependencies, while LightGBM's decision tree boosting swiftly adapts to short-term market fluctuations. Wang et al. [22] stated that LSTM architectures, consisting of memory cells, input, output, and forget gates, could easily outdo linear or polynomial regression. Jialin & Qiao [23] then leveraged the financial predicting field by introducing a hybrid architecture

of CNN for high-dimensional features extraction and LSTM for data synthesis at gates, termed CNN-LSTM, which improved the forecasting accuracy by 36.4%. However, the complexity of DL-based models, requiring extensive computational resources for training and inference, and hyperparameter tuning can hamper their deployment in environments where rapid processing is critical like trading, leading to potential bottlenecks in real-time financial applications [24].

## 2.2 Transformer Architectures

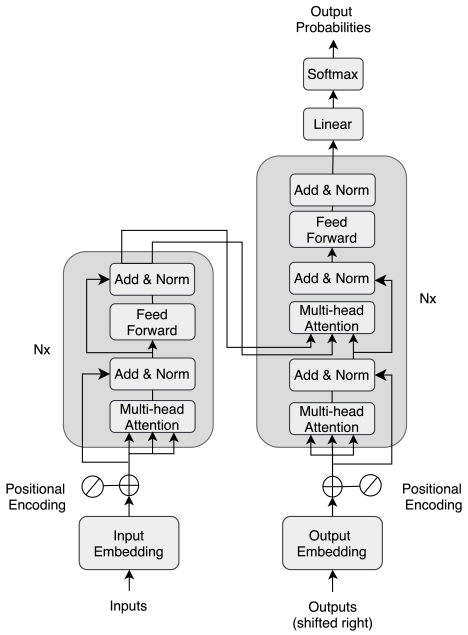


**Fig. 1.** The original Transformer architecture [25]

Transformer model was first introduced by Vaswani et al. [25] in 2017, representing a groundbreaking shift in the approach to sequence-to-sequence tasks in natural language processing (NLP) as it solely relied on attention mechanisms over recurrent and convolutional layers in traditional ML/DL frameworks, resulting in state-of-the-art performance on the English-to-German and English-to-French machine translation tasks. Vanilla Transformer (Fig. 1) composed of a stack of encoders and decoders, where each layer employs multi-head self-attention mechanisms and position-wise fully connected feed-forward networks, facilitating direct dependencies between all input and output positions. This challenges researchers in applying Transformer for time series analysis [26–29].

Lin [27] studied that traditional Transformer, despite its capability of dodging pattern of autocorrelation in the long term, performed weaker than normal LSTM in both minute and daily frequency trading data of A-share stocks, providing nearly five times higher MAE and RMSE. Similarly, questioning the effectiveness of the vanilla Transformer, Zeng et al. [28] applied it to nine popular datasets for long-term time series forecasting (LTSF) problems. The results were disappointing as Transformer was highly complex but ineffective in understanding LTSF, hence providing larger MAE and MSE in comparison to a basic linear model. Hu [29] then developed a state-of-the-art Temporal Fusion Transformer (TFT), which successfully dominated SVR and LSTM due to TFT's integration of Gated Linear Units (GLUs) mechanisms and variable selection networks, deepening its ability to capture relevant temporal dynamics unlike traditional Transformers, which employs mainly attention heads. While there are emerging studies on Transformer-based multi-modal architectures [30–32], their potential for prompt time series forecasting with less complicated networks thus ensuring sustainable overheads in stock price prediction remains a largely unexplored research area. This motivates the authors to conduct a lightweight Transformer for scalping and day trading with smooth deployment on lower mid-range devices.

# 3 Methodology

## 3.1 Data Collection and Evaluation Metrics

This research has gathered and analyzed raw Open-High-Low-Close-Volume (OHLCV) data from four prominent technology enterprises: Amazon (AMZN), CISCO (CSCO), International Business Machines Corporation (IBM), and Intel Inc. (INTC). Data was collected from Yahoo Finance's open-access website. Each dataset encompasses a 20-year period, ranging from July 28, 2003, to July 28, 2023, consisting of 5,036 daily observations, which laid a suitable foundation for Transformer application. Accordingly, this study adopts a univariate approach, in which "Close" price would be the target feature, allowing the training model to isolate and examine the intricate patterns in stocks' closing prices.

The Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were chosen as evaluation metrics for their effectiveness in identifying predictive errors in univariate analysis. MAE represents the average magnitude of errors in forecasts, offering a direct measure of overall prediction accuracy, which is crucial in the volatile field of stock market values. RMSE, known for intensively magnifying larger errors, underscores significant discrepancies that can indeed have serious, profound financial implications. The combined use of these metrics provides a comprehensive evaluation of the model's forecasting accuracy in the complex and high-stakes environment of stock price prediction. Likewise, higher error values indicate greater divergence between actual and predicted values.

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

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{2}$$

Where:

$n$ is the total number of data points or observations.
$Y_i$ represents the actual (true) value of the stock price.
$\hat{Y}_i$ represents the predicted value of the stock price.

## 3.2 Lightweight Transformer

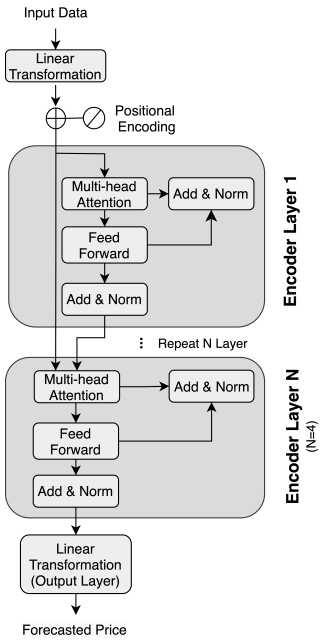


**Fig. 2.** The proposed Transformer architecture

This study proposes a lightweight Transformer model (Fig. 2) that is tailored for immediate sequential time series analysis. Input data undergoes an initial linear transformation, enhancing its representation before being processed by the sequential layers of the model. This is followed by the addition of positional encodings to imbue the sequence with temporal information, compensating for the

Transformer's inherent lack of sequential processing. The core of the model comprises multiple encoder layers, each consisting of a multi-head attention mechanism that simultaneously processes the input data in varied representational spaces, enabling the capture of complex dependencies. These attention outputs are then normalized and passed through a feed-forward network, further distilled, and normalized again. The processed information from the final encoder layer is then transformed by a linear layer, producing the forecasted price as the output. Likewise, the model employs a multi-head self-attention mechanism, capable of distilling dependencies across different positions in the input sequence. Specifically, it utilizes a model dimensionality of 128, 8 parallel attention heads, a hidden layer size of 512, and a total of 4 layers (N=4) in the encoder stack.

Dropout [33] was applied at a rate of 0.1 to prevent overfitting during training. Positional encodings are injected to provide the model with temporal context, vital for sequences where order impacts meaning. We enhance the robustness of the model with dropout regularization and layer normalization [33], which are applied within each sub-layer of the encoder to mitigate overfitting and promote stability in the learning process. Consequently, the architecture results in a linear output layer, which maps the encoded temporal features to the predicted future stock price, reflecting the model's final synthesis of the input data's underlying trends. In addition, an early stopping mechanism complements our training methodology, ceasing further epochs when validation loss fails to improve, thus conserving computational resources and preventing model over-training [34].

# 4 Experiments and Results

## 4.1 Experimental Process

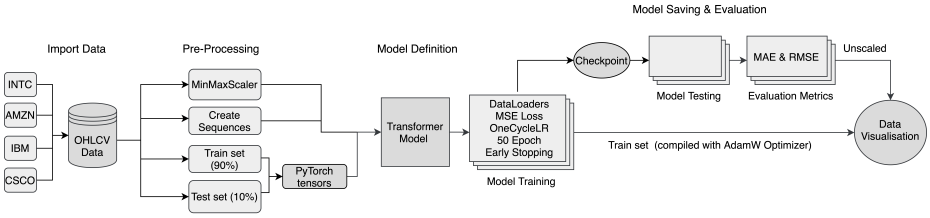


**Fig. 3.** Full experimental process

Our experimental process commenced by initializing the random number generators in both NumPy and PyTorch libraries with a seed value of 42 to ensure reproducibility across experiments. Following the loading of OHLCV data (Fig. 3), we immediately divided the dataset into training (90%) and testing (10%) sets to prevent data leakage. This ensures that the normalization process, using

MinMaxScaler [35], did not incorporate information from the testing set, thus maintaining the integrity of our experimental setup. The normalization was applied to achieve uniformity in value ranges (-1 to 1), addressing issues of varying magnitudes within financial data. The training dataset was normalized, and the same scaler was used to transform the test set, containing approximately 503 observations, to avoid introducing any bias. This approach prevents overfitting and ensuring that our model's performance is evaluated on truly unseen data.

Subsequent to normalization, sequences from the "Close" price data were crafted, each with a length of 1 day, to predict the subsequent day's closing price, encapsulating short-term temporal dependencies. This sequence creation, resulting in 4,531 input-output pairs, was performed exclusively on the normalized training data, ensuring that the model's exposure to future data points was strictly regulated. Converted into PyTorch tensors and DataLoader instances with a batch size of 64, the datasets prepared the Transformer model for accurately capturing underlying time series patterns. The model architecture defined (Section 3.2) included a Mean Squared Error (MSE) loss function for quantifying prediction errors and an AdamW optimizer [36] with an initial learning rate of 0.001. A OneCycleLR scheduler [37] was initiated to vary the learning rate cyclically, promoting better convergence. An early stopping mechanism [34] was implemented, terminating training if the validation loss did not improve after five epochs, preventing overfitting and ensuring the model generalized well to unseen data. This careful separation of training, validation, and testing sets, combined with strategic data normalization and sequence creation, safeguards against data leakage, ensuring the integrity of our experimental process.

During the training phase, the model was set to train mode, and the dataset was iterated batch-wise. For each batch, the gradients were first reset, the model's predictions were computed, the loss was calculated, and backpropagation was performed to update the model's weights. The learning rate scheduler's step was invoked at the end of each batch processing to adjust the learning rate. After each training epoch, the model was evaluated on 10% of the training set to compute the validation loss. This evaluation loss was then used as a criterion for the early stopping check. The training and validation losses were logged for monitoring the model's performance over time. Once the early stopping condition was met, the training process was concluded, and the model's state with the lowest validation loss was restored from the saved checkpoint. This model state is considered the best model as it yielded the least error on the testing set during the training process. The best model was then used to perform a final evaluation on the testing set. We employed the model in inference mode to ensure that operations like dropout were not applied during this phase. Lastly, MAE and RMSE metrics were computed to provide comprehensive insights into the model's predictive accuracy. Finally, the normalized predictions were unscaled and visualized.

## 4.2 Performance Comparison

**Table 1.** Optimized model performance on stock datasets

| | IBM | | CSCO | | AMZN | | INTC | |
|---|---|---|---|---|---|---|---|---|
| Model | MAE | RMSE | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| LSTM [22] | 0.086 | 0.102 | 0.128 | 0.205 | 0.119 | 0.201 | 0.087 | 0.103 |
| SVR [38] | 0.125 | 0.152 | 0.145 | 0.176 | 0.151 | 0.183 | 0.130 | 0.158 |
| CNN-LSTM [23] | 0.075 | 0.095 | 0.082 | 0.106 | 0.087 | 0.108 | 0.082 | 0.105 |
| CNN-BiLSTM-ECA [10] | 0.071 | 0.090 | 0.083 | 0.106 | 0.085 | 0.107 | 0.077 | 0.099 |
| Transformer | 0.020 | 0.027 | 0.043 | 0.059 | 0.044 | 0.055 | 0.026 | 0.035 |

According to Table 1, our single-step Transformer model demonstrates a consistently strong performance across stock datasets. For IBM, it achieves an RMSE that is 70.0% lower than its closest competitor, the CNN-BiLSTM-ECA. Regarding CSCO, the Transformer has a smaller MAE compared to CNN-LSTM by approximately 42.7%, and it surpasses CNN-LSTM's RMSE by 37.9%, thus indicating superior performance in both metrics. Likewise, the lightweight Transformer is more pronounced with AMZN, where it outperforms CNN-BiLSTM-ECA's metrics by 49.4% and 46.6%, respectively, suggesting a strong handling of both average and large errors. With INTC, the Transformer improves upon its nearest competitor's evaluation metrics by 66.2% and 63.9%. This significant improvement highlights the Transformer's efficiency in learning patterns. However, LSTM and SVR's simplicity and sensitivity to hyperparameters result in the highest errors among others, discouraging traditional ML-based frameworks. Therefore, the Transformer has surpassed common architectures in sequential time series analysis, averagely improving the MAE and RMSE by 57.5% and 54.6% across stock datasets, showcasing its exceptional predictive accuracy.

**Table 2.** Performance metrics across runs on stock data

| | Duration (second) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Take | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| IBM | 23.05 | 23.30 | 22.90 | 23.10 | 23.80 | 23.10 | 23.50 | 23.00 | 23.60 | 23.20 |
| CSCO | 18.26 | 18.51 | 18.19 | 18.58 | 19.02 | 18.14 | 18.73 | 19.05 | 18.64 | 18.03 |
| AMZN | 19.60 | 19.87 | 19.82 | 19.37 | 19.72 | 18.64 | 18.74 | 19.32 | 19.09 | 19.76 |
| INTC | 16.17 | 15.72 | 15.90 | 16.91 | 16.23 | 16.20 | 15.80 | 16.74 | 16.83 | 16.44 |

The empirical results presented in Table 2 nominate the proposed Transformer model as a lightweight solution suitable for deployment on modest hardware configurations. The experiments were conducted on the author's MacBook

Pro 2016 base version, equipped with a 2.9 GHz Dual-Core Intel Core i5 processor, 8 GB 2133 MHz LPDDR3 RAM, and Intel Iris Graphics 550 with 1536 MB of VRAM, without the aid of external GPUs. This local machine configuration is lower mid-range in comparison to the baseline standards of current computing devices [39]. After 10 iterations of running across stock data, the model took less than 19.40 seconds on average to generate forecasting results, maintaining stable evaluation metrics at high precision. Hence, this demonstrates the feasible applicability of our lightweight Transformer in limited computational environments.

Table 3 - 4 then employs Transformer and examines its stability in multi-step forecasting analysis, confirming its forecast horizon where the model's predictions are most accurate at low MAE and RMSE. Assigning the model to operate within this optimal step ensures that the forecasting leverages the model's strengths, thereby reducing the likelihood of significant predictive errors and enhancing the reliability of the stock data analysis. Consequently, the single-step Transformer (1-step) still excels in forecasting with an average improvement of 21.6% in MAE and 24.5% in RMSE compared to the next 10-step forecast, showcasing its proficiency in making highly accurate short-term predictions.

**Table 3.** Multi-step forecasting results on stock data (steps 1-5)

| Model | Evaluation Metrics | 1-step | 2-step | 3-step | 4-step | 5-step |
|---|---|---|---|---|---|---|
| Transformer on IBM | MAE | 0.020 | 0.024 | 0.028 | 0.035 | 0.037 |
| | RMSE | 0.027 | 0.032 | 0.038 | 0.045 | 0.048 |
| Transformer on CSCO | MAE | 0.043 | 0.044 | 0.059 | 0.059 | 0.045 |
| | RMSE | 0.059 | 0.064 | 0.079 | 0.078 | 0.062 |
| Transformer on AMZN | MAE | 0.044 | 0.055 | 0.052 | 0.069 | 0.073 |
| | RMSE | 0.055 | 0.073 | 0.065 | 0.087 | 0.092 |
| Transformer on INTC | MAE | 0.026 | 0.030 | 0.040 | 0.043 | 0.047 |
| | RMSE | 0.035 | 0.039 | 0.050 | 0.054 | 0.060 |

**Table 4.** Multi-step forecasting results on stock data (steps 6-10)

| Model | Evaluation Metrics | 6-step | 7-step | 8-step | 9-step | 10-step |
|---|---|---|---|---|---|---|
| Transformer on IBM | MAE | 0.042 | 0.045 | 0.045 | 0.051 | 0.051 |
| | RMSE | 0.054 | 0.057 | 0.057 | 0.065 | 0.065 |
| Transformer on CSCO | MAE | 0.059 | 0.052 | 0.072 | 0.069 | 0.061 |
| | RMSE | 0.078 | 0.071 | 0.090 | 0.090 | 0.080 |
| Transformer on AMZN | MAE | 0.067 | 0.756 | 0.068 | 0.075 | 0.087 |
| | RMSE | 0.086 | 0.097 | 0.090 | 0.097 | 0.108 |
| Transformer on INTC | MAE | 0.048 | 0.050 | 0.053 | 0.055 | 0.055 |
| | RMSE | 0.061 | 0.065 | 0.068 | 0.072 | 0.071 |

# 5 Data Visualiation

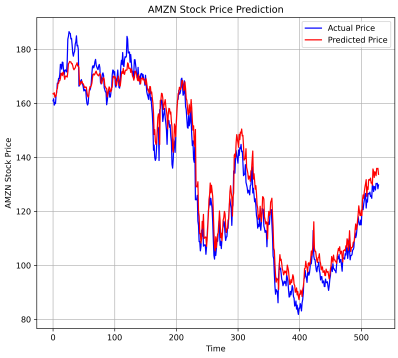


**Fig. 4.** AMZN forecasting results

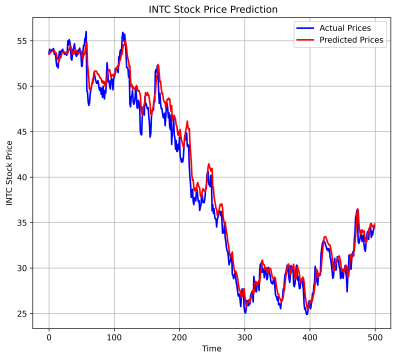


**Fig. 5.** INTC forecasting results

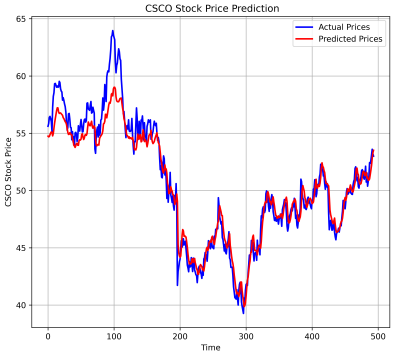


**Fig. 6.** CSCO forecasting results

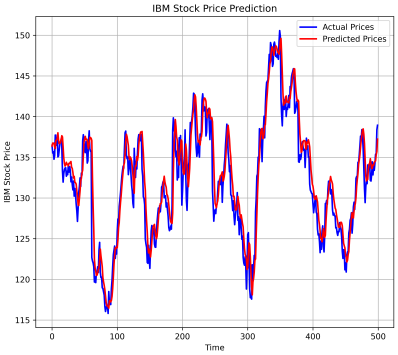


**Fig. 7.** IBM forecasting results

The graphical representations (Fig. 4-7) showcase the correlations between actual and predicted stock prices across 500 discrete time steps. The real stock prices are depicted through a blue line, whereas the forecasted prices are illustrated by a red line. Therefore, a decrease in the gap between these lines indicates a lower variation in price thus informing greater forecasting precision.

Regarding the analysis of AMZN (Fig. 4), the predictive accuracy in the short-term range (0-150 time steps) fails to capture the abnormal price fluctuations adequately. However, in the mid-term forecasting phase (150-350 time steps), following an abrupt decline in stock values, the model demonstrates an

improved proficiency in identifying both upward and downward price trends across various periods, evidenced by the predominance of the predicted (red) line over the actual (blue) line. Nonetheless, the model encounters challenges in accurately scaling the predicted values to align with the real stock prices during the long-term phase (350-500 time steps), particularly with the emerging uptrend due to AMZN's highly volatile patterns. Concerning INTC (Fig. 5), despite Transformer being weak in short term, our model adeptly captures the immediate decline in INTC's stock prices in the mid term and accurately reflects the cyclical yet overall downtrend observed from point 150 to 300. Next, the model follows INTC's establishment of a 100-step seasonality pattern and its subsequent breakthrough of the resistance zone after time step 400, indicating the Transformer's robust tracking capability against instant upward movement.

CSCO (Fig. 6) presents similar patterns to the aforementioned cases, in which Transformer struggles with short-term forecasting due to strongly abnormal market behaviors, consisting of unexpected price shocks. However, from time step 200 onwards, the Transformer captures the equity recovery optimally, foreseeing the next flash crash and rebound of value. Contrastingly, IBM analysis (Fig. 7) illustrates a distinct scenario where the forecasted values converge with the actual stock prices from short to mid term before failing to cover the long-term peaks, thus underestimating most minor fluctuations, similar to the CSCO case.

# 6 Discussion

## 6.1 Summary of Findings and Contributions

In comparison to the original vanilla Transformer [25] and our previous studies [12–14], this research offers a reconstructed one that is finer-tuned for time series forecasting tasks as it features tailored sequence processing designed for numerical data, efficient positional encoding for capturing chronological dependencies, and a streamlined design with fewer parameters and layers for enhanced computational efficiency. These modifications not only reduce training complexity but also reinforce model applicability and performance in real-time time series analysis. Our lightweight Transformer outperforms renowned ML and DL-based architectures (e.g., LSTM, SVR, CNN-LSTM, CNN-BiLSTM-ECA) in terms of MAE and RMSE. Operating on an earlier-generation local machine with basic configurations, the model requires only 19.36 seconds to generate forecasting results across stock datasets, enabling promising deployment on lower-tier devices.

Moreover, the proposed Transformer showcased its profound capabilities in discerning cyclical and seasonal trends in stock values, as well as its agility in responding to immediate market downturns, which is a remarkable finding in pattern recognition within the stock markets, as flash crashes caused by market manipulation or human herding behavior tend to deteriorate algorithmic-trading models performance [40–42]. All have resolved the uncertainties questioned by previous research [27, 29, 31] about the Transformer's time series applicability due to its inefficacy in tracing long-term dependencies and consuming large com-

putational resources because of the power-consuming architectures of the self-attention mechanism, which quadratically scales with the sequence length, in comparison to basic linear or ML-based models. By simplifying the Transformer and applying various training algorithms simultaneously to ideal series of data at different operating stages, the lightweight Transformer mitigates common problems of non-stationarity, overfitting, underfitting, and long-term computational overheads. The empirical findings contribute to innovative algorithmic trading.

### 6.2 Limitations and Recommendations

The proposed Transformer relies solely on the closing prices of stocks for prediction, as this study isolates the "Close" data for rigorous examination, ignoring influential features such as OHLCV data or macroeconomic indicators, which may limit the model's ability to fully comprehend the complexities of the stock market. However, this does not diminish the quality of the paper, as our focus is to leverage insights from closing prices alone, while also providing opportunities for researchers to explore multivariate analysis. Therefore, using a fixed sequence length for single-step forecasting may not be optimal for capturing longer-term dependencies or patterns that span multiple time steps, thus limiting analyses across varying temporal windows. Thus, the model's runtime is established on an average local machine, which may result in longer processing times.

As a result, utilizing up-to-date high-end devices for model training might result in a twofold increase in processing speed, particularly if researchers intend to extend time series sequences and conduct multivariate analysis alongside multi-step forecasting. We recommend employing Phase Space Reconstruction technique to preprocess input time series stock data to enhance its data dimensionality, thereby revealing underlying patterns [12]. Feature engineering is also proposed, as technical indicators (e.g., RSI, SMA, and MACD) derived from the target feature, and macroeconomic indicators (e.g., S&P 500, CSI, and GNI) gathered from open-access sources could offer the model a more holistic view of the digital economy hence enriching its decision-making in real-time trading [14].

## 7 Conclusion

This research introduces a lightweight Transformer architecture tailored for univariate stock price forecasting, demonstrating superior performance over popular Machine Learning models at minimal runtime alongside stable evaluation metrics across stock datasets. Therefore, this encourages researchers to de-complex and transform the vanilla architecture rather than overcomplicate it within time series tasks. Our findings also challenge previous assumptions about the applicability of Transformer models in time series analysis and forecasting, proving its effectiveness even on low computational resources. However, recognizing the limitations of relying solely on closing prices for prediction, we propose explorations into multivariate analysis and the inclusion of macroeconomic indicators alongside a larger computational workforce to enhance pattern recognition. This study

not only improves algorithmic strategies but also opens avenues for applying lighter Transformer models in time-sensitive financial analysis tasks, promising a significant impact on the digital economy's predictive analytics landscape.

## 8 Availability of Data and Materials

Stock datasets were publicly sourced from Yahoo Finance and stored in the "data" folder under author's open-access Github repository; Source code of the lightweight Transformer and models for comparison is stored in the "src" folder; Graphs for the data visualization part could be found in the "graphs" folder; Runtime records are kept in the "rec" folder. All can be distributed under the GNU License. Mathematical foundation is attached in the supplementary file.

## References

1. Arun Kumar, Dimpal Sharma, Girraj Khandelwal, and Gajanand Sharma. Computer vision, machine learning based monocular biomechanical and security analysis. *Journal of Discrete Mathematical Sciences & Cryptography*, 2023.
2. Nusrat Mohi et al. Breast cancer detection using deep learning: Datasets, methods, and challenges ahead. *Computers in biology and medicine*, 149:106073, 2022.
3. Jonathan A. Weyn, Dale R. Durran, Rich Caruana, and Nathaniel Cresswell-Clay. Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models. *Journal of Advances in Modeling Earth Systems*, 13, 2021.
4. Peipei Liu, Yunfeng Zhang, Fangxun Bao, Xunxiang Yao, and Caiming Zhang. Multi-type data fusion framework based on deep reinforcement learning for algorithmic trading. *Applied Intelligence*, 53:1683–1706, 2022.
5. An Xiao Xuan et al. A comprehensive evaluation of statistical, machine learning and deep learning models for time series prediction. *2022 7th International Conference on Data Science and Machine Learning (CDMA)*, pages 55–60, 2022.
6. Hanlin Mo. Comparative analysis of linear regression, polynomial regression, and arima model for short-term stock price forecasting. *Advances in Economics, Management and Political Sciences*, 2023.
7. Lucia Inglada-Perez. Uncovering non-linearity and chaos in financial markets: Empirical evidence for four major stock market indices. *Entropy*, 22, 2020.
8. Gaurang Sonkavde et al. Forecasting stock market prices using machine learning and deep learning models: A systematic review, performance analysis and discussion of implications. *International Journal of Financial Studies*, 2023.
9. Farhad Shiri, Thinagaran Perumal, Norwati Mustapha, and Raihani Mohamed. A comprehensive overview and comparative analysis on deep learning models: Cnn, rnn, lstm, gru. *ArXiv*, abs/2305.17473, 2023.
10. Yu Chen et al. Stock price forecast based on cnn-bilstm-eca model. *Sci. Program.*, 2021:2446543:1–2446543:20, 2021.
11. Laith Alzubaidi et al. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 8, 2021.
12. Anh Nguyen, Son Ha, and Phien Nguyen. Cnn-bilstm-gru and phase space reconstruction in retail stock price forecasting. *SSRN Electronic Journal*, 2023. DOI: 10.2139/ssrn.4729759.

13. Anh Nguyen, Son Ha, and Phien Nguyen. Cnn-bilstm and time delay embedding: A single-step hybrid deep learning model for stock price forecasting. *SSRN Electronic Journal*, 2023. DOI: 10.2139/ssrn.4729187.
14. Anh Nguyen, Son Ha, and Phien Nguyen. Transforming stock price forecasting: Deep learning architectures and strategic feature engineering. *SSRN Electronic Journal*, 2023. DOI: 10.2139/ssrn.4729146.
15. Jiyuan Gui and Xiaoyun. Forecasting the stock price of vaccine manufacturers in china using machine learning and econometrics model. In *Other Conferences*, 2022.
16. Deepak Kumar et al. Analysis and prediction of stock price using hybridization of sarima and xgboost. *2022 International Conference on Communication, Computing and Internet of Things (IC3IoT)*, pages 1–4, 2022.
17. Yuxin Zhao. Comparison of stock price prediction in context of arima and random forest models. *BCP Business & Management*, 2023.
18. Umesh Pratap Singh. A critical review of the effectiveness of machine learning & deep learning approaches in forecasting stock market trends. *International Journal on Recent and Innovation Trends in Computing and Communication*, 2023.
19. Arash Gharehbaghi. Deep learning in time series analysis. 2023.
20. Angelo Casolaro, Vincenzo Capone, Gennaro Iannuzzo, and Camastra. Deep learning for time series forecasting: Advances and open problems. *Inf.*, 14:598, 2023.
21. Jiabao Li. The comparison of lstm, lgbm, and cnn in stock volatility prediction. *Proceedings of the 2022 7th International Conference on Financial Innovation and Economic Development (ICFIED 2022)*, 2022.
22. Qiaoyu Wang, Kai Kang, and Zhihan Zhang. Application of lstm and conv1d lstm network in stock forecasting model. *Artificial Intelligence Advances*, 2021.
23. Liu Jialin et al. Cnn-lstm model stock forecasting based on an integrated attention mechanism. *2022 3rd International Conference on Pattern Recognition and Machine Learning (PRML)*, pages 403–408, 2022.
24. Xia Hu, Lingyang Chu, and Jian Pei. Model complexity of deep learning: a survey. *Knowledge and Information Systems*, 63:2585 – 2619, 2021.
25. Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.
26. Edmond Lezmi and Jiali Xu. Time series forecasting with transformer models and application to asset management. *SSRN Electronic Journal*, 2023.
27. Zhuoran Lin. Comparative study of lstm and transformer for a-share stock price prediction. In *Proceedings of the 2023 2nd International Conference on Artificial Intelligence, Internet and Digital Economy (ICAID 2023)*, Atlantis Highlights in Intelligent Systems, pages 72–82. Atlantis Press, 2023.
28. Ailing Zeng, Mu-Hwa Chen, L. Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI Conference on Artificial Intelligence*, 2022.
29. Xiao-Ping Hu. Stock price prediction based on temporal fusion transformer. *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 60–66, 2021.
30. Tong Li and Zhaoyang Liu. Master: Market-guided stock transformer for stock price forecasting. *ArXiv*, abs/2312.15235, 2023.
31. Agus Tri Haryono, Riyanarto Sarno, and Kelly Rossa Sungkono. Transformer-gated recurrent unit method for predicting stock price based on news sentiments and technical indicators. *IEEE Access*, 11:77132–77146, 2023.
32. Vaishnavi Tevare and P. S. Revankar. Forecasting stock prices with stack transformer. *2023 International Conference on Circuit Power and Computing Technologies (ICCPCT)*, pages 1262–1269, 2023.

33. Nitish Srivastava et al. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.
34. Lutz Prechelt. Early stopping-but when? In *Neural Networks*, 1996.
35. Lucas et al. The choice of scaling technique matters for classification performance. *Appl. Soft Comput.*, 133:109924, 2022.
36. Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.
37. Leslie N. Smith and Nicholay Topin. Super-convergence: very fast training of neural networks using large learning rates. In *Defense + Commercial Sensing*, 2018.
38. Yanhui Guo, Siming Han, Chuanhe Shen, Y. Li, Xijie Yin, and Yu Bai. An adaptive svr for high-frequency stock price forecasting. *IEEE Access*, 6:11397–11404, 2018.
39. Simon Leitner. The best laptops spring 2023 - 95 reviewed notebooks, 2023.
40. Alex Quinn. Algorithm trading in & for the foreign exchange. *Quantitative Computerized Trading EURO PACIFIC QUANT RESEARCH GROUP*, 2014.
41. John Fry and Jean-Philippe Serbera. Modelling and mitigation of flash crashes. *Munich Personal RePEc Archive*, 2017. MPRA Paper No. 82457.
42. Donn S. Fishbein. Neural networks and genetic algorithms : Another tool for the technical analysis of financial markets. 2002.