LiteFormer: An Encoder-Only Multi-Head Attention Transformer for Financial Time Series Forecasting

Nguyen Quoc Anh [1], Ha Xuan Son[1*], 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 Artificial Intelligence 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 LiteFormer, 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 mean-squared variants of evaluation metrics, the proposed LiteFormer consistently surpasses renowned recurrent network-based forecasting models, averaging a reduction in forecasting errors by over 50%. Being trained across 20-year daily stock datasets across multiple industries, LiteFormer accurately captures flash crashes, cyclical or seasonal patterns, and long-term dependencies in technology, finance, and pharmaceutical stocks. Hence, 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.

Keywords: Lightweight Transformer, Positional Encoding, Time Series, Stock Markets, 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 Transformers 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]. The post-pandemic economic rebound also 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 abnormalities [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 reflect the latest market condition.

This research 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 visualising 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.

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 [12–15].

2.1 Related Work

Gui & Wu [12] 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. [13] 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 [14], 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 [15] 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 [16,17].

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 [18]. 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. [19] stated that LSTM architectures, consisting of memory cells, input, output, and forget gates, could easily outdo linear or polynomial regression. Jialin & Qiao [20] then leveraged the financial predicting field by introducing a hybrid architecture of CNN for high-dimensional feature 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 bottlenecks in real-time applications [21].

2.2 Transformer Architectures

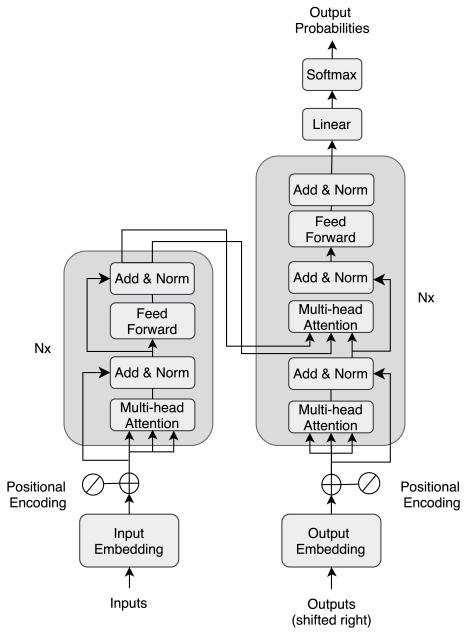

Figure 1: The original Transformer architecture [22]

Transformer model was first introduced by Vaswani et al. [22] 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 [23–26].

Lin [24] studied that traditional Transformer, despite its capability of dodging patterns 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. [25] 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 [26] 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 employ mainly attention heads. While there are emerging studies on Transformer-based multi-modal architectures [27–29], their potential for prompt time series forecasting thus ensuring sustainable overheads in stock price prediction remains a largely unexplored research area. This motivates the authors to conduct a lightweight Transformer-based model for scalping and day trading with smooth deployment on lower mid-range devices.

3 Methodology

3.1 Data Collection and Evaluation Metrics

Table 1: Testing portfolio

Category	Ticker	Full Name
Technology	AMZN	Amazon.com Inc.
	CSCO	Cisco Systems Inc.
	INTC	Intel Corporation
Finance	C	Citigroup Inc.
	GS	Goldman Sachs Group Inc.
	JPM	JPMorgan Chase & Co
Pharmaceutical	ABBV	AbbVie Inc.
	JNJ	Johnson & Johnson
	NVS	Novartis AG

This research has gathered and analyzed raw Open-High-Low-Close-Volume (OHLCV) data from nine publicly-listed enterprises across different industries

within the S&P 500 Index. Data was fetched from Yahoo Finance's API through *yfinance* library. Each open-access dataset encompasses 20 years, from July 28, 2003, to July 28, 2023, consisting of 5,036 daily observations, laying a suitable foundation for ML-based 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. Furthermore, collecting stocks from various industries allows researchers to test the model's adaptability and robustness across varying market dynamics and behaviors. In particular, the technology sector tend to be highly volatile due to rapid innovation cycles, shifting consumer demands, and geopolitical tensions, causing sudden and unpredictable changes like flash crashes or instant uptrends in stock prices. In contrast, the finance industry is driven by macroeconomic indicators, interest rates, and regulatory changes, resulting in cyclical patterns that require the model to capture external dependencies. Meanwhile, the pharmaceutical industry is influenced by regulatory approvals, clinical trial outcomes, and product patents, introducing binary and event-driven volatility that further complicates prediction efforts for the model. Therefore, excelling these environments showcases a model's applicability in real-time operation.

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. Likewise, Mean Squared Error (MSE) is used as the loss function because it penalizes larger errors more severely. 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.

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

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i| \tag{2}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{3}$$

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 LiteFormer

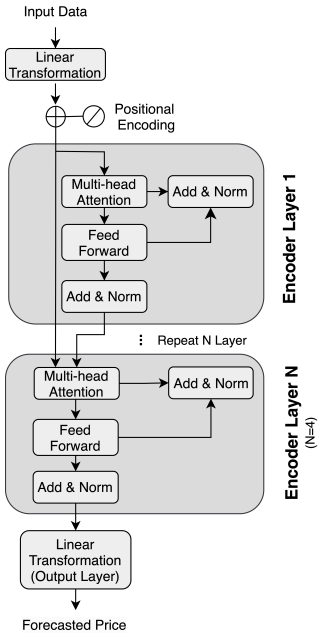

Figure 2: The proposed Transformer-based architecture

This study proposes a lightweight Transformer model (Fig. 2), named LiteFormer, 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 (PE) to imbue the sequence with temporal information, compensating for the Transformer's inherent lack of sequential processing. Consequently, the positional encoding for position *pos* and dimension i is calculated using sine and cosine functions:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{4}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{5}$$

Where:

pos is the position in the sequence.

i is the dimension.

$d_{\mathbf{model}}$ is the model dimension, which in this model is set to 128.

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. Specifically, LiteFormer utilises 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. The multi-head attention layer enables the model to focus on different parts of the input sequence simultaneously. It is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

Furthermore, this computation is extended across multiple heads, where each head is defined as follows for our model construction. For our implementation with $d_{\text{model}} = 128$ and $h = 8$ heads, each head processes the input with a key dimension of $d_k = 16$, which is presented as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O \tag{7}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{8}$$

Where:

Q, K, V are the query, key, and value matrices derived from the input.

W_i^Q, W_i^K, W_i^V are the parameter matrices for the i-th head.

W^O is the output parameter matrix.

h is the number of heads.

d_k is the dimensionality of the key vectors.

After the attention mechanism, each layer in the Transformer contains a position-wise feed-forward network, which applies a linear transformation to each position separately and identically. The weight matrices W_1 and W_2 have dimensions of $d_{\text{model}} \times d_{\text{ff}}$ and $d_{\text{ff}} \times d_{\text{model}}$ respectively, where d_{ff} is the dimension of the feed-forward layer's inner layer. In LiteFormer, d_{ff} is set to 512.

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{9}$$

x is the input to the feed-forward layer.

W_1, b_1 are the weights and biases for the first linear transformation.

W_2, b_2 are the weights and biases for the second linear transformation.

Dropout [30] 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 [30], which are applied within each sub-layer of the encoder to mitigate overfitting

and promote stability in the learning process. Here, $\mathrm{E}[x]$ and $\mathrm{Var}[x]$ are the mean and variance computed over the feature dimension, γ and β are learnable parameters of the same dimensionality as x, and ϵ is a small constant added for numerical stability, typically on the order of $1e-5$ to $1e-6$. The process is mathematically explained as:

$$\mathrm{LayerNorm}(x) = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} \odot \gamma + \beta \tag{10}$$

x is the input vector to the layer normalization.

μ is the mean of the features.

σ is the standard deviation of the features.

γ is a scale parameter, learned during training.

β is a shift parameter, learned during training.

The processed information from the final encoder layer is then transformed by a linear layer, producing the forecasted price as the output. An early stopping mechanism complements our training methodology, ceasing further epochs when validation loss fails to improve, thereby conserving computational resources while preventing over-training [31].

Table 2: LiteFormer model summary

Layer	Output Shape	Param #
input_layer	(None, 1)	0
positional_encoding	(1, 128)	0
transformer_encoder_layer_1	(1, 128)	196,608
transformer_encoder_layer_2	(1, 128)	196,608
transformer_encoder_layer_3	(1, 128)	196,608
transformer_encoder_layer_4	(1, 128)	196,608
linear	(1, 1)	129
Total Trainable Parameters		786,561
Total Non-Trainable Parameters		0
Total Parameters		786,561

Table 2 outlines the overall technical architecture of our LiteFormer. LiteFormer is structured to process sequential data effectively by leveraging four stacked transformer encoder layers, each with an output shape of (1, 128) and containing 196,608 parameters. The inclusion of positional encoding addresses the lack of inherent order in transformers, allowing the model to capture temporal dependencies in the input data. Despite its seemingly moderate total of 786,561 trainable parameters, the model's architecture emphasizes depth over width, allowing for a more nuanced representation of complex patterns.

4 Experiments and Results

4.1 Experimental Process

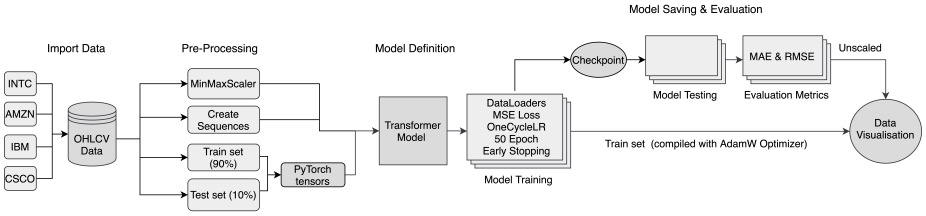

Figure 3: Full experimental process

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 [32], 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. Specifically, the MinMaxScaler scales each feature x to a range of $[-1, 1]$ using the formula:

$$x_{\text{scaled}} = 2 \times \frac{x - x_{\min}}{x_{\max} - x_{\min}} - 1 \tag{11}$$

Where:

x is the original value of the feature.

$x_{\min}$ and $[x_{\max}]$ is the minimum value of the feature in the training set.

$x_{\max}$ is the maximum value of the feature in the training set.

The training dataset was normalized using the above equations, 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 ensures 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. These sequences were converted into PyTorch tensors and DataLoader

instances with a batch size of 64, preparing the datasets for the Transformer model to accurately capture underlying time series patterns.

The model architecture defined (Section 3.2) included a Mean Squared Error (MSE) loss function for quantifying prediction errors. During the training phase, we used the AdamW optimizer [33] with an initial learning rate of 0.001. The AdamW optimization algorithm updates the model parameters θ_t using the following update rule:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{12}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{13}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{14}$$

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \lambda \theta_{t-1} \tag{15}$$

Where:

m_t and $[v_t]$ are the first and second moment estimates at time step t.

β_1, β_2 are hyperparameters that control the exponential decay rates.

g_t is the gradient of the loss with respect to the parameters at time step t.

η is the learning rate.

λ is the weight decay coefficient.

ϵ is a small constant to prevent division by zero.

In addition, a OneCycleLR scheduler [34] was initiated to vary the learning rate cyclically, promoting better convergence. This scheduler adjusts the learning rate η_t according to a cosine annealing schedule:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{t}{T}\pi\right)\right) \tag{16}$$

Where:

$\eta_{\min}$ and $[\eta_{\max}]$ are the minimum and maximum learning rates.

t is the current iteration.

T is the total number of iterations.

Finally, an early stopping mechanism [31] 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: 1. The gradients were reset. 2. The model's predictions $\hat{y}$ were computed. 3. The loss L was calculated using the MSE loss function. 4. Backpropagation was performed to update the model's weights. 5. The learning rate scheduler's step was invoked to adjust the learning rate. After each training epoch, the model was evaluated on 10% of the training set to compute the validation loss, which was used as a criterion for the early stopping check. The training and validation losses were logged for monitoring the model's performance over time. L is denoted as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{17}$$

Accordingly, y_i and $\hat{y}_i$ are the true and predicted values, respectively, and N is the batch size. 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 optimal version 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 insights into the model's predictive accuracy.

Table 3: Hyperparameter Tuning Summary

Parameter	Tested Values	Optimal Value
`n_features`	1, 2, 3	1
`d_model`	32, 64, 128, 256, 512	128
`n_heads`	2, 4, 8, 16, 32	8
`n_hidden`	128, 256, 512, 1024, 2048	512
`n_layers`	1, 2, 3, 4, 5, 6	4
`dropout`	0.05, 0.1, 0.15, 0.2, 0.25	0.1
`batch_size`	32, 64, 128, 256	64
`learning rate`	0.0001, 0.001, 0.01, 0.1	0.001
`optimizer`	Adam, AdamW, SGD	AdamW

Table 3 shows the optimal value for `d_model` was found to be 128, balancing the model's capacity to capture intricate patterns without excessively increasing computational complexity. A multi-head attention mechanism with `n_heads`

set to 8 was selected, indicating a preference for moderate parallel processing to enhance the model's attention capabilities. The model performs best with a hidden layer size (`n_hidden`) of 512 and a depth of 4 layers (`n_layers`), suggesting that a deeper network effectively learns the nuanced patterns in sequential stock data. Notably, a dropout rate of 0.1 was chosen to mitigate overfitting, while the `AdamW` optimizer with a learning rate of 0.001 and a batch size of 64 strikes a balance between training stability and efficiency.

4.2 Performance Comparison

Table 4: Performance metrics for Tech industry stocks

	CSCO		AMZN		INTC	
Model	MAE	RMSE	MAE	RMSE	MAE	RMSE
SVR [35]	0.145	0.176	0.151	0.183	0.130	0.158
LSTM [19]	0.128	0.205	0.119	0.201	0.087	0.103
CNN-LSTM [20]	0.082	0.106	0.087	0.108	0.082	0.105
CNN-BiLSTM-ECA [10]	0.083	0.106	0.085	0.107	0.077	0.099
LiteFormer	0.043	0.059	0.044	0.055	0.026	0.035

Table 5: Performance metrics for Finance industry stocks

	C		GS		JPM	
Model	MAE	RMSE	MAE	RMSE	MAE	RMSE
SVR [35]	0.163	0.208	0.174	0.215	0.143	0.196
LSTM [19]	0.095	0.137	0.129	0.146	0.082	0.123
CNN-LSTM [20]	0.084	0.130	0.077	0.115	0.091	0.122
CNN-BiLSTM-ECA [10]	0.078	0.103	0.086	0.109	0.069	0.111
LiteFormer	0.024	0.031	0.032	0.041	0.027	0.036

Table 6: Performance metrics for Pharma industry stocks

	ABBV		JNJ		NVS	
Model	MAE	RMSE	MAE	RMSE	MAE	RMSE
SVR [35]	0.152	0.193	0.169	0.201	0.211	0.185
LSTM [19]	0.104	0.143	0.097	0.130	0.087	0.118
CNN-LSTM [20]	0.082	0.111	0.089	0.121	0.077	0.104
CNN-BiLSTM-ECA [10]	0.079	0.110	0.081	0.112	0.073	0.102
LiteFormer	0.026	0.034	0.029	0.038	0.032	0.041

To clarify the effectiveness of LiteFormer in stock price forecasting, four established models were used as baselines for comparison (Tables 4-6). Each sector exhibits unique characteristics that pose different challenges for deep learning models. Technology stocks, such as CSCO and AMZN, are marked by rapid growth and high volatility, driven by innovation. Finance stocks, like C and GS, generally offer more stability, being influenced by economic cycles and interest rates. Pharma stocks, including ABBV and JNJ, carry specific risks associated with regulatory approvals and patent expirations, often resulting in high volatility following product launch outcomes.

In analyzing the technology sector (Table 4), LiteFormer clearly outperforms other models. For CSCO, LiteFormer reduces the MAE and RMSE by approximately 47.56% and 44.34%, respectively, compared to the nearest competitor, CNN-LSTM. Similarly, for AMZN, LiteFormer diminishes the MAE and RMSE by around 48.28% and 48.60% relative to the second-best model, CNN-BiLSTM-ECA. For INTC, the performance gains are even more pronounced; LiteFormer reduces the MAE by 66.23% and the RMSE by 64.65% compared to CNN-BiLSTM-ECA. This translates into an aggregate improvement of approximately 52.94% in forecasting accuracy across the technology sector, highlighting LiteFormer's capacity to handle high volatility and rapid market changes more effectively than traditional models.

Concerning the finance sector (Table 5), LiteFormer demonstrates consistent superiority over the baseline models. For C, it achieves reductions in MAE and RMSE of 69.23% and 69.90% against CNN-BiLSTM-ECA, the closest contender. In the case of GS, the improvements are similarly substantial, with LiteFormer lowering the MAE by 58.44% and the RMSE by 64.16%. For JPM, LiteFormer again achieves remarkable accuracy, decreasing the MAE by 60.87% and the RMSE by 67.57%. These results culminate in an average error reduction of 65.83% in the finance sector, showcasing LiteFormer's ability to capture stable yet cyclical financial market trends more reliably than others.

Regarding the pharmaceutical sector (Table 6), LiteFormer continues to outperform all baseline models. For ABBV, LiteFormer reduces the MAE and RMSE by 67.09% and 67.92%, respectively, compared to CNN-BiLSTM-ECA. In forecasting JNJ, LiteFormer decreases the MAE by 64.20% and the RMSE by 66.07%. For NVS, LiteFormer achieves reductions of 68.49% in MAE and 68.63% in RMSE. On average, this results in a forecast accuracy improvement of 66.93% across pharmaceutical stocks. Despite the unique challenges posed by the sector, including regulatory hurdles and patent uncertainties, LiteFormer demonstrates its robust adaptability and predictive power.

Contrastingly, models like SVR and LSTM underperform significantly, deviating by over 50% in MAE and RMSE across sectors. These results emphasize the potential of transformer-based models like LiteFormer in time series forecasting. The considerable error reduction and increased reliability of predictions illustrate LiteFormer's superiority in short-term forecasting across diverse market conditions. Further analyses will explore its multi-step forecasting capabilities, further establishing its optimal prediction horizons and enhancing stock data analysis from a machine learning perspective.

Table 7: Performance metrics across runs on stock data (Takes 1 to 5)

	Take				
	1	2	3	4	5
CSCO	18.26	18.51	18.19	18.58	19.02
AMZN	19.60	19.87	19.82	19.37	19.72
INTC	16.17	15.72	15.90	16.91	16.23
C	21.45	20.89	21.30	21.77	21.12
GS	22.80	22.35	22.60	23.05	22.47
JPM	19.95	19.60	20.20	19.85	20.05
ABBV	17.65	17.89	17.73	17.55	17.92
JNJ	20.15	20.30	20.05	20.22	20.40
NVS	18.70	18.95	18.83	18.65	18.90

Table 8: Performance metrics across runs on stock data (Takes 6 to 10)

	Take				
	6	7	8	9	10
CSCO	18.14	18.73	19.05	18.64	18.03
AMZN	18.64	18.74	19.32	19.09	19.76
INTC	16.20	15.80	16.74	16.83	16.44
C	21.50	21.34	20.78	21.66	21.03
GS	22.83	22.19	22.75	22.92	22.61
JPM	19.75	19.92	20.15	19.87	19.78
ABBV	17.85	17.68	17.95	17.88	17.76
JNJ	20.11	20.18	20.36	20.08	20.25
NVS	18.72	18.87	18.80	18.92	18.76

The empirical results presented in Tables 7 and 8 further highlight the LiteFormer model's efficiency 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 use of external GPUs. This setup is considered lower mid-range compared to the current baseline standards in computing devices, suitable for resource-constraint compatibility test [36].

After 10 iterations across different stock datasets, the model consistently demonstrated low running times. Particularly, in the technology sector, the average runtime for CSCO remained under 19.02 seconds, with slight fluctuations across takes, while for AMZN and INTC, the average runtimes were approximately 19.50 seconds and 16.46 seconds, respectively. Similarly, in the finance sector, the runtimes for C, GS, and JPM ranged between 20.89 and 23.05 seconds. In the pharmaceutical sector, runtimes were notably efficient, averaging around 17.77 seconds for ABBV, 20.22 seconds for JNJ, and 18.83 seconds for NVS. These results indicate that LiteFormer can achieve high precision in forecasting while maintaining short computation times, averaging less than 19.40 seconds across all stocks. This demonstrates the model's feasible applicability in environments with limited computational resources, making it a practical choice for real-world deployment on standard personal computing devices.

Table 9: Multi-step forecasting results on stock data (steps 1-5)

Model	Metrics	1-step	2-step	3-step	4-step	5-step
LiteFormer on CSCO	MAE	0.043	0.047	0.052	0.056	0.050
	RMSE	0.059	0.065	0.070	0.074	0.069
LiteFormer on AMZN	MAE	0.044	0.048	0.053	0.061	0.066
	RMSE	0.055	0.060	0.066	0.074	0.080
LiteFormer on INTC	MAE	0.026	0.030	0.035	0.038	0.042
	RMSE	0.035	0.041	0.046	0.051	0.056
LiteFormer on C	MAE	0.024	0.028	0.032	0.036	0.040
	RMSE	0.031	0.037	0.043	0.048	0.053
LiteFormer on GS	MAE	0.028	0.033	0.037	0.041	0.045
	RMSE	0.037	0.043	0.048	0.054	0.059
LiteFormer on JPM	MAE	0.021	0.025	0.029	0.034	0.037
	RMSE	0.028	0.033	0.038	0.043	0.048
LiteFormer on ABBV	MAE	0.026	0.031	0.035	0.038	0.041
	RMSE	0.034	0.039	0.044	0.049	0.054
LiteFormer on JNJ	MAE	0.025	0.029	0.034	0.037	0.041
	RMSE	0.034	0.038	0.044	0.048	0.053
LiteFormer on NVS	MAE	0.023	0.027	0.031	0.035	0.039
	RMSE	0.032	0.037	0.042	0.046	0.051

Table 10: Multi-step forecasting results on stock data (steps 6-10)

Model	Metrics	6-step	7-step	8-step	9-step	10-step
LiteFormer on CSCO	MAE	0.055	0.058	0.062	0.065	0.060
	RMSE	0.075	0.079	0.083	0.088	0.084
LiteFormer on AMZN	MAE	0.070	0.074	0.079	0.084	0.089
	RMSE	0.085	0.090	0.095	0.101	0.107
LiteFormer on INTC	MAE	0.045	0.048	0.052	0.056	0.058
	RMSE	0.060	0.065	0.069	0.074	0.078
LiteFormer on C	MAE	0.043	0.047	0.051	0.056	0.060
	RMSE	0.055	0.060	0.066	0.072	0.077
LiteFormer on GS	MAE	0.049	0.053	0.058	0.062	0.067
	RMSE	0.063	0.068	0.074	0.080	0.086
LiteFormer on JPM	MAE	0.040	0.044	0.048	0.052	0.056
	RMSE	0.050	0.055	0.060	0.066	0.071
LiteFormer on ABBV	MAE	0.046	0.051	0.055	0.060	0.064
	RMSE	0.057	0.063	0.069	0.075	0.080
LiteFormer on JNJ	MAE	0.045	0.049	0.054	0.059	0.063
	RMSE	0.056	0.061	0.067	0.072	0.078
LiteFormer on NVS	MAE	0.042	0.046	0.050	0.055	0.059
	RMSE	0.053	0.058	0.063	0.069	0.074

Tables 9 - 10 employ LiteFormer to examine its stability in multi-step forecasting analysis, where the model predicts multiple time steps into the future. In a multi-step forecasting scenario, the model is tasked with predicting future values $y_{t+1}, y_{t+2}, \ldots, y_{t+h}$ for a given time horizon h. This can be achieved through either a recursive strategy (using previous predictions as inputs for subsequent forecasts) or a direct strategy (predicting all future values in one go). For this analysis, the recursive strategy was used, where the predicted value at each time step becomes the input for the next step:

$$\hat{y}_{t+i} = f(\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots, \hat{y}_{t+i-1}) \tag{18}$$

Where:

$\hat{y}_{t+i}$ is the predicted value at step i.

f is the model function incorporating past predictions to forecast future values.

In each step i, the input consists of the previous forecasted values, which inherently compounds prediction errors as i increases, thus potentially leading to greater divergence from the true future values. The stability and accuracy of the model across multiple time steps are then evaluated using MAE and RMSE at each forecast horizon. The tables confirm the model's forecast horizon where 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 10-step forecast, showcasing its proficiency in making highly accurate short-term predictions. This indicates that the model's accuracy diminishes with each subsequent step due to the compounding nature of errors, a common characteristic of multi-step forecasting.

5 Results Visualization

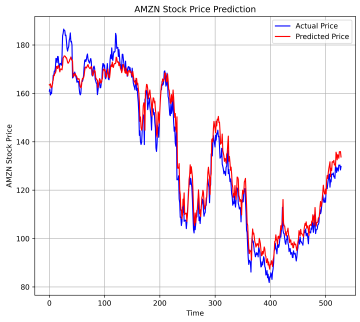

Figure 4: AMZN forecasting results

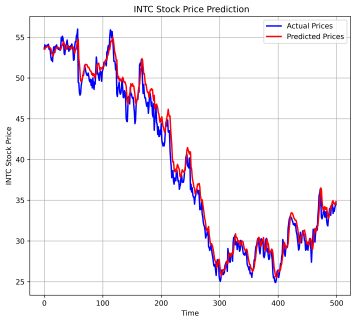

Figure 5: INTC forecasting results

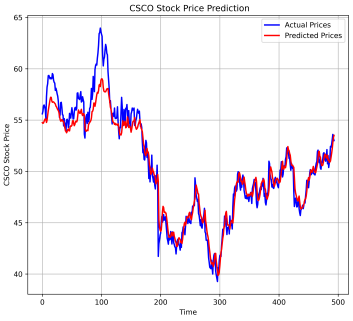

Figure 6: CSCO forecasting results

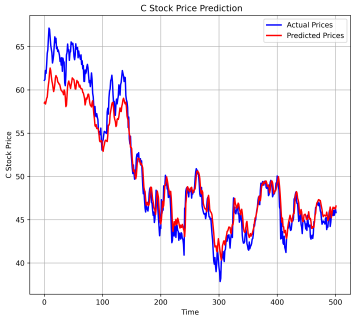

Figure 7: C forecasting results

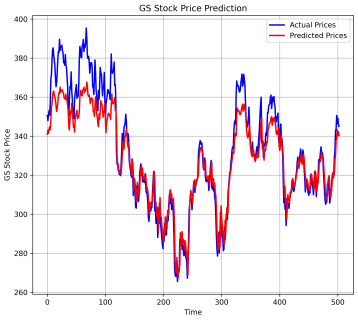

Figure 8: GS forecasting results

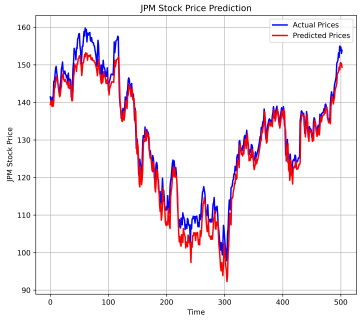

Figure 9: JPM forecasting results

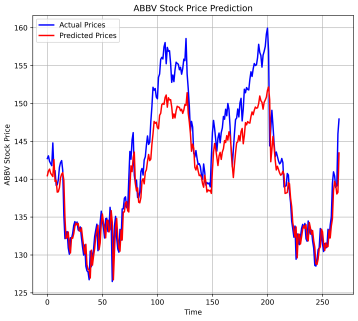

Figure 10: ABBV forecasting results

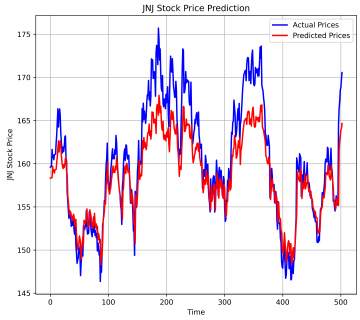

Figure 11: JNJ forecasting results

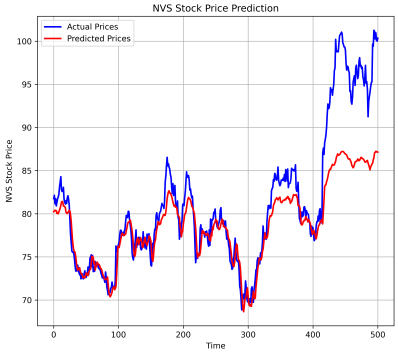

Figure 12: NVS forecasting results

The graphical representations (Fig. 4-12) illustrate the correlations between actual and predicted stock prices across 500-time steps in the test sets. The real stock prices are shown by the blue line, while the forecasted prices are represented by the red line. A decrease in the gap between these lines indicates lower price variation, which reflects higher forecasting precision.

Concerning AMZN (Fig. 4), the model's predictive accuracy in the short-term range (0-150 time steps) struggles to capture abnormal price fluctuations. However, in the mid-term (150-350 time steps), following an abrupt decline in stock value, the model exhibits improved proficiency in identifying both upward and downward price trends, as evidenced by the predicted (red) line closely tracking the actual (blue) line. Despite this improvement, challenges arise in the long-term phase (350-500 time steps), particularly in scaling the predicted values to align with the real stock prices during the emerging uptrend. These difficulties stem from AMZN's highly volatile patterns. For INTC (Fig. 5), although LiteFormer demonstrates weaknesses in short-term predictions, it adeptly captures the immediate decline in stock prices in the mid-term. The model accurately reflects the cyclical downtrend observed from point 150 to 300. Additionally, it successfully tracks INTC's establishment of a 100-step seasonality pattern and its subsequent breakthrough of the resistance zone post time step 400, showcasing LiteFormer's robust capability in responding to instant upward movements. CSCO (Fig. 6) then presents a similar pattern to the aforementioned cases, where LiteFormer struggles with short-term forecasting due to unpredictable market behaviors and unexpected price shocks. Nevertheless, from time step 200 onwards, the model optimally captures the equity recovery, accurately foreseeing the next flash crash and subsequent rebound.

In the case of C (Fig. 7), significant challenges arise in the short-term (0-150 time steps), where the model fails to adequately capture rapid fluctuations and downward movements. However, during the mid-term range (150-350 time steps), the model demonstrates noticeable improvement, successfully identifying the stock's cyclical downtrend and periodic fluctuations. In the long-term phase (350-500 time steps), it continues to follow the overall trend of the stock, though occasional lags in response to sudden market reversals highlight its limitations in adapting to abrupt changes. For GS (Fig. 8), short-term predictions exhibit moderate accuracy, with some difficulty in capturing rapid fluctuations, resulting in a discernible gap between the actual and predicted lines. In the mid-term range (150-350 time steps), the model better captures the downward trends and cyclical nature of GS's stock movements. However, in the long-term phase (350-500 time steps), the model consistently underestimates the stock's actual price during sharp upward movements, indicating challenges in forecasting large-scale market changes over extended periods. Regarding JPM (Fig. 9), the model's predictive accuracy improves over time. Short-term performance (0-150 time steps) struggles with sudden market changes, leading to noticeable discrepancies. However, during the mid-term phase (150-350 time steps), the model aligns more closely with the general downward trend and subsequent recovery, effectively tracking cyclical patterns. In the long-term phase (350-500 time steps), while the model successfully follows the overall upward trend, it

occasionally delays in response to steep price increases, indicating slight limitations in adapting to rapid market movements.

The analysis of ABBV (Fig. 10) demonstrates strong short-term forecasting (0-150 time steps), where the predicted line closely mirrors actual stock movements. During the mid-term phase (150-250 time steps), the model continues to capture general trends, though some discrepancies arise during periods of extreme volatility, such as the sharp peak around the 200th time step. In the long-term (250-500 time steps), the model reflects the overall price direction but struggles to scale with sudden upswings, leading to underestimations during high market volatility. For JNJ (Fig. 11), the model shows relatively consistent performance in both short- and mid-term ranges. In the short-term (0-150 time steps), it captures initial trends with reasonable accuracy. Throughout the mid-term (150-350 time steps), the model aligns with the stock's cyclical behaviors, though it occasionally underestimates certain peaks and valleys, particularly around the 300th time step. In the long-term phase (350-500 time steps), the predictions become less precise, especially during periods of heightened volatility, indicating difficulties in adapting to abrupt market shifts. NVS case (Fig. 12) finally reveals high accuracy in both the short- and mid-term ranges (0-350 time steps), effectively capturing most price fluctuations. However, during the long-term phase (350-500 time steps), the model struggles to keep up with a steep upward trend, resulting in an increasing gap between predicted and actual values. This divergence highlights the model's limitations in managing sudden, large-scale market changes over extended periods.

In summary, LiteFormer demonstrates great potential in mid-term forecasting, particularly in capturing cyclical trends and moderate fluctuations across various stocks. However, its short-term accuracy is frequently hindered by abrupt market fluctuations, leading to discrepancies in predicting sudden changes. Long-term forecasts reveal additional challenges, particularly in scaling to match the magnitude of sharp price movements, indicating a need for improvement in handling high market volatility. Notably, LiteFormer performs relatively better in the finance sector, as evidenced by its improved tracking of cyclical patterns and resistance zones in stocks such as INTC and JPM. In contrast, its performance in the tech sector is less consistent, struggling with the high volatility in stocks like AMZN and CSCO. In the pharmaceutical sector, the model shows strong short-term predictions, as seen with ABBV and NVS, but still encounters difficulties with sudden market shifts over extended periods. This sector-based evaluation suggests LiteFormer's strengths lie in finance stocks while highlighting areas for further enhancement in both tech and pharma forecasting, driven by their abrupt influence.

6 Discussion

6.1 Summary of Findings and Contributions

In comparison to the original vanilla Transformer [22] and our previous studies [37–39], this research offers a reconstructed LiteFormer 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. LiteFormer outperforms renowned ML and DL-based architectures in terms of MAE and RMSE. Operating on a local machine with basic configurations, the model requires only 19.36 seconds to generate forecasting results across stock datasets, enabling promising deployment on weak devices.

Moreover, the proposed LiteFormer showcased its strengths 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 [24, 26, 28] about the Transformer's time series applicability due to its inefficacy in tracing long-term dependencies and consuming large computational 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, LiteFormer 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 LiteFormer 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, limiting analyses across varying temporal windows. The model's runtime is also established on an average local machine, which may result in longer processing times.

As a result, utilising up-to-date high-end devices for model training might result in a twofold increase in processing speed, particularly if researchers in-

tend 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 [37]. 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 signaling [39].

7 Conclusion

This research introduces a lightweight Transformer architecture (LiteFormer) 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 their effectiveness even with 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 to time-sensitive financial analysis tasks, promising a significant impact on the digital economy's predictive analytics landscape.

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] Jiyuan Gui and Xiaoyun. Forecasting the stock price of vaccine manufacturers in china using machine learning and econometrics model. In *Other Conferences*, 2022.

[13] 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.

[14] Yuxin Zhao. Comparison of stock price prediction in context of arima and random forest models. *BCP Business & Management*, 2023.

[15] 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.

[16] Arash Gharehbaghi. Deep learning in time series analysis. 2023.

[17] Angelo Casolaro, Vincenzo Capone, Gennaro Iannuzzo, and Camastra. Deep learning for time series forecasting: Advances and open problems. *Inf.*, 14:598, 2023.

[18] 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.

[19] Qiaoyu Wang, Kai Kang, and Zhihan Zhang. Application of lstm and conv1d lstm network in stock forecasting model. *Artificial Intelligence Advances*, 2021.

[20] 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.

[21] Xia Hu, Lingyang Chu, and Jian Pei. Model complexity of deep learning: a survey. *Knowledge and Information Systems*, 63:2585 – 2619, 2021.

[22] 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.

[23] Edmond Lezmi and Jiali Xu. Time series forecasting with transformer models and application to asset management. *SSRN Electronic Journal*, 2023.

[24] 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.

[25] Ailing Zeng, Mu-Hwa Chen, L. Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *AAAI Conference on Artificial Intelligence*, 2022.

[26] 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.

[27] Tong Li and Zhaoyang Liu. Master: Market-guided stock transformer for stock price forecasting. *ArXiv*, abs/2312.15235, 2023.

[28] 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.

[29] 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.

[30] Nitish Srivastava et al. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[31] Lutz Prechelt. Early stopping-but when? In *Neural Networks*, 1996.

[32] Lucas et al. The choice of scaling technique matters for classification performance. *Appl. Soft Comput.*, 133:109924, 2022.

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

[34] Leslie N. Smith and Nicholay Topin. Super-convergence: very fast training of neural networks using large learning rates. In *Defense + Commercial Sensing*, 2018.

[35] 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.

[36] Simon Leitner. The best laptops spring 2023 - 95 reviewed notebooks, 2023.

[37] Anh Nguyen, Son Ha, and Phien Nguyen. Phase space reconstructed neural ordinary differential equations model for stock price forecasting. *PACIS 2024 Proceedings*, 2023.

[38] 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.

[39] Anh Nguyen, Son Ha, and Phien Nguyen. Transforming stock price forecasting: Deep learning architectures and strategic feature engineering. *MDAI 2024 Proceedings*, 2024.

[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.

[42] Donn S. Fishbein. Neural networks and genetic algorithms : Another tool for the technical analysis of financial markets. 2002.