Multivariate Stock Time-Series Forecasting Using an Improved Transformer

Ru Zheng *

School of Mathematics and Science, Huaqiao University, quanzhou 362021, China

* Corresponding Author Email: 2321111051@stu.hqu.edu.cn

Abstract. As financial markets become increasingly digital and intelligent, stock-price forecasting has emerged as an important topic in financial engineering, artificial intelligence, and data mining. Stock prices are shaped by firm fundamentals, macroeconomic conditions, investor sentiment, and market policies, while also exhibiting strong nonlinearity, nonstationarity, and noise. Extracting useful information from historical financial data and developing accurate and stable forecasting models therefore remain challenging. This study proposes an improved Transformer for multivariate stock time-series forecasting that combines feature-gated embedding with a residual prediction head. Daily data for Apple (AAPL), Microsoft (MSFT), and Google (GOOG) were obtained from the public YFinance interface for the period from January 1, 2000, to January 1, 2020. The model uses eight features—Open, High, Low, Close, Volume, RSI, ATR, and MACD—from the previous 30 trading days to predict the closing price five trading days ahead. A strictly chronological split, training-set-only normalization, and separate sliding-window construction are adopted to prevent look-ahead leakage. The proposed model is compared with Persistence, residual XGBoost, LSTM, a vanilla Transformer, and a standard feature-gated Transformer through the main experiment, cross-stock generalization tests, an ablation study, and a supplementary return-prediction experiment. On AAPL, the model achieves an RMSE of 1.2712, an MAE of 0.9242, and a MAPE of 2.5268%, slightly outperforming Persistence and residual XGBoost. It performs nearly identically to Persistence on GOOG and is slightly weaker than Persistence and residual XGBoost on MSFT, while consistently outperforming the LSTM and Transformer-based deep-learning baselines. These results demonstrate the accuracy and stability of the proposed model.

Keywords: Financial time series; stock-price forecasting; Transformer; feature gating; multivariate forecasting.

1. Introduction

Stock-price forecasting has always been an important problem in the field of finance research as well as artificial intelligence applications. As capital-market trading has expanded and data access has improved, investors and researchers have gained access to large volumes of historical trading data and have used mathematical models and intelligent algorithms to identify price patterns. In recent years, machine-learning and deep-learning methods have been widely applied to financial-market forecasting, asset pricing, and limit-order-book modeling [5–7]. Accurate forecasts can support investment decisions, risk management, portfolio optimization, and quantitative trading. Nevertheless, stock prices are influenced by firm fundamentals, macroeconomic conditions, investor sentiment, and market policies, and their movements are highly uncertain, nonlinear, nonstationary, and noisy. Stock forecasting therefore remains a difficult task.

Stock-price series generally have non-linear, non-stationary nature and lots of noises. Traditional time-series approaches are generally based on relatively limiting statistical assumptions. Take the ARIMA model as an example, it is a conventional statistical prediction method which uses autoregressive term and moving average term [1]; whereas the GARCH model is usually used to portray conditional heteroskedasticity and volatility clustering in finance markets [2]. But these models can be understood, however forecasting ability limited if you use them on nonlinear financial data with multiple inputs.

With advances in machine learning and deep learning, data-driven approaches have increasingly been applied to financial forecasting. Compared with traditional statistical models, machine-learning

methods can capture nonlinear relationships to some extent. XGBoost, in particular, provides stable performance on structured-data prediction tasks through gradient-boosted trees [3]. Among deep-learning methods, LSTM mitigates vanishing gradients in long-term dependency modeling through gating mechanisms [4] and has been used in financial-market forecasting [5]. More recently, the Transformer has become a major sequence-modeling architecture because of its self-attention mechanism and parallel computation [8]. Informer, Autoformer, FEDformer, PatchTST, TimesNet, and iTransformer have further extended Transformer-based time-series forecasting [9–18]. Transformer models have also been applied to stock-index forecasting, stock-movement prediction, and asset allocation [19–23]. These studies provide the methodological basis for the improved Transformer developed here.

Given this, we present an improved Transformer for multivariate stock time-series prediction. Rather than predicting the next day's closing price, it forecasts five trading days into the future which is more difficult and less affected by basic price persistence. A feature-gated embedding layer was also added to the input in order to adaptively weight the price-volume variable and the technical indicators. This relates also to variable selection mechanisms found within interpretable multivariate timeseries models as well as channel recalibration [10,24]. At output stage, we learn a residual prediction head that predicts the change from now till then over the current close. It conforms with good short term continuity and small variations of stock-price series.

The major contributions include the following three aspects. Firstly, a multi-variable framework is developed to predict the close price 5 days ahead. Combining fundamental OHLCV information and RSI, ATR, MACD to represent prices-volumes conditions, momentum, volatility and trends. The second is an improved Transformer with a feature-gated embedding and a residual prediction head is proposed for better modeling of feature contributions and short-term price deviations. Thirdly experiments are carried out on AAPL, MSFT and GOOG using Persistence, ResidualXGBoost, LSTM vanillaTransformer as benchmark. Also did some ablation tests and gate important features analysis, added up return predicting experiment.

2. Related Work

2.1. Traditional Statistical Models

Traditional statistics models were among the first methods used for financial time-series forecasting, with ARIMA and GARCH being typical examples. The ARIMA model is a time series model made up of autoregressive, differencing and moving-average components, which is suitable for relatively stationary time series [1]. GARCH is mainly applied in volatility clustering and depicts market volatility persistence by conditional variance [2]: Stock price series are highly nonstationary and nonlinear, so purely linear statistical structures can not fully describe their complicated dynamics. They can only be seen as simple reference materials or volatility modeling ones.

2.2. Machine-Learning Models

With computing capacity improved, machine-learning approaches like support vector machines, random forests and XGBoost have been utilized in predicting stock. Those methods can model nonlinear relationships and can benefit from feature engineering. Combining many regression trees to make up for this XGBoost improves the stability of the model and the fitting ability on non-linear data, it serves as a strong baseline on structured data [3]. Empirical financial studies also show that machine learning can capture nonlinear feature interactions in asset pricing and return prediction that conventional linear models may overlook [6]. But XGBoost lacks an inherent memory of time and usually needs its historical windows to be made flat prior to modeling. And thus, the window construction and features' quality greatly determine its performance.

2.3. Deep-Learning Models

Deep-learning models can automatically extract complicated representations, so it has been broadly adopted for forecasting financial time-series. RNNs and LSTM transmit information between different times using hidden states and gates; this makes them appropriate for sequences [4]. Also, in the case of LSTM extracting some useful patterns from the past price series on predicting financial markets[5]; Models like DeepLOB that are created for high-frequency limit-order books prove further that deep neural networks have potential when it comes to dealing with complex financial-sequences [7] However, LSTMs handle observations sequentially, they can't be entirely parallel. In the case of long sequence, the earlier informations would also lose in the propagation process.

2.4. Transformer-Based Models

Transformer models sequences with self-attention, which can measure relations between any two positions and aggregate information according to attention weights [8]: Unlike recurrent models such as LSTM it does not process the input recursively and computes dependencies at all times steps in parallel. Informer, Autoformer, FEDformer, PatchTST, TimesNet and iTransformer improve on long-sequence efficiency, trend decomposition, frequency domain modeling, pattern based representation and variable-wise modeling [11–18]. Reviews find that the Transformer is a major branch of current research into time-series prediction. [15] In finance, Transformer model have been applied for stock-index forecasting, stock-movement prediction, Chinese banking-stocks forecast, asset allocation and federated stock prediction [19–23]. Based on the vanilla Transformer, this paper introduces features gate and residual output structure so make it better fit multivariate financial series.

3. Dataset and Preprocessing

3.1. Data Sources

The following 3 American tech stocks are looked at; Apple (AAPL), Microsoft (MSFT), Google (GOOG). Daily data were collected via public YFinance interface for the common period of January 1, 2000 to January 1, 2020. Raw field has open high low close price and trading volume These stock was selected because of the three following factors: Firstly these are all top technology firms that actively trade and have fairly complete records Secondly firm from same sector can be compared so we could test model on new data Third AAPL is main experiment stock, MSFT & GOOG give cross-stock validation samples.

3.2. Feature Selection

The last input consists of 8 characteristics: five basic price-volume variables, namely Open, High, Low, Close, Volume, and three technical indicators, RSI, ATR, MACD. Use a 14-day RSI to show overbought or oversold and momentum. Use a 14-day ATR to indicate the actual range of price swings. MACD uses 12, 26 and 9 settings and retain the DIF value as trend feature All technical indicator values are calculated based on recursive computation from the raw market data, no external sources involved.

3.3. Data Cleaning and Normalization

During preprocessing, observations with missing values, suspension day of trading, and incomplete technical indicator information are removed, only the complete feature days of trading remain. To prevent data from leaking from training to testing set I first split the data based on time and then normalize it First 70% is taken as train Set rest 30% for Test without shuffling each feature’s min, max was calculated only for training The above mentioned statistics obtained from the training set will be used in order to standardize all both the training set and test set utilizing Min-Max Normalization. The input range [0, 1] will be achieved after this process. Normalization is like this:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Where x denotes the original feature value, x_{min} and x_{max} are the minimum and maximum of that feature in the training set, and x_{norm} is the normalized value. The same training-set minimum and maximum are applied to both the training and test sets, preventing test information from entering model training. Normalization reduces scale differences among features and improves convergence and training stability.

3.4. Sliding-Window Sample Construction

The supervised samples use a window with 30 trading days. So every input is made up of 30 consecutive observations each with eight features, which can be described as 30×8. Main target is the closing price on the fifth following trading day: The information from [t−29, t] is utilized to forecast the close at t+5. The window advances by one and separately creates samples in training set and test set. All models' predictions are inverse-transformed back into actual U.S. dollar prices before being evaluated.

4. Methodology

4.1. Overall Architecture of the Improved Transformer

As shown in Figure 1, the proposed stock-forecasting model consists of a feature-gated embedding layer, positional encoding, a Transformer encoder, attention pooling, and a residual output layer. The input is a 30 × 8 multivariate time-series matrix, and the output is the normalized closing-price forecast for the fifth trading day ahead. First, the feature-gating layer adaptively reweights the eight input variables. A linear projection and layer normalization then map the inputs into a hidden space with a model dimension of 64. Sinusoidal positional encoding is added, after which two Transformer encoder layers extract temporal dependencies [8]. Finally, attention pooling and the residual prediction head produce the future closing-price estimate.

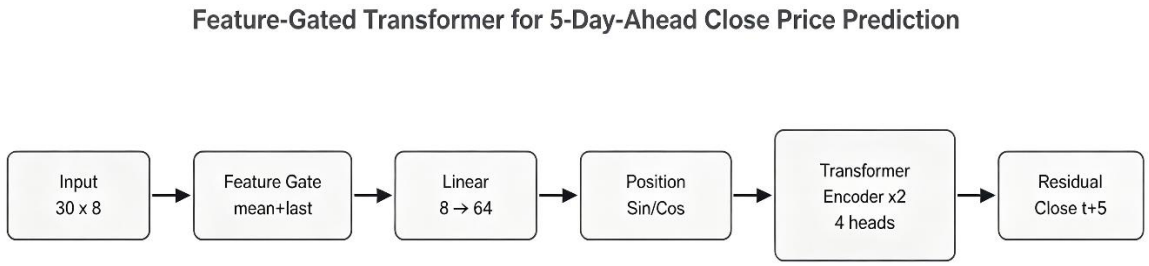


Figure 1. Overall architecture of the improved Transformer.

4.2. Feature-Gated Embedding Layer

The multivariate stock features vary in terms of scale and predictive strength. A feature-gated embedding layer is applied on input side for modeling the feature importance adaptively. Layer first perform mean pool over time then concatenates it with Feature Vector at Final Time Step for Global Representation This vector gets passed through a fully connected layer, ReLU activation, Dropout, another FC layer, and finally Sigmoid to get 8 gate weights. these weights will be multiplied element by element to original inputs after which it will pass through linear layer and go to transformer hidden space. Inspired from variable selection networks and channel attention/recalibration [10, 24], we design such that more emphasis is given to those variables having information content from multivariate input.

4.3. Positional Encoding and Self-Attention

Because the Transformer contains no recurrent structure, it cannot inherently perceive the temporal order of the input as an RNN or LSTM can [8]. Positional encoding is therefore added to represent the order of trading days. It assigns a distinct positional representation to each time step, allowing the model to distinguish observations from different days.

The original Transformer uses sine and cosine functions to construct positional encodings:

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

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{3}$$

Where pos denotes the time-step position, i is the feature-dimension index, and the final term denotes the model hidden dimension. Positional encoding provides sequence-order information without recurrence. It is added elementwise to the feature embeddings before the sequence enters the Transformer encoder.

Standard sinusoidal positional encoding is used to provide each trading day with positional information. Self-attention then measures relationships across time steps through the query matrix Q, key matrix K, and value matrix V:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{4}$$

Where d_k is the dimensionality of the key matrix K and scales the dot product to prevent excessively large values from saturating the softmax gradient. The mechanism assigns different attention weights to historical trading days, identifies time points and price patterns that contribute most to the forecast, and captures long-range dependencies in financial sequences.

4.4. Multi-Head Attention and Feed-Forward Network

A single attention operation learns temporal dependencies in only one representation space. Multi-head attention projects the input into several subspaces and computes attention separately in each, allowing the model to capture different temporal patterns. It is defined as follows:

$$MultiHead(Q,K,V) = Concat(head_1, head_2, \ldots, head_h)W^O \tag{5}$$

Where each head_i is an independent attention head and W^O is the output projection matrix. Different heads may focus on short-term price fluctuations, longer-term trends, or volume effects. Multi-head attention therefore enables the Transformer to interpret stock time series from multiple perspectives.

Each Transformer encoder layer also includes a feed-forward neural network in addition to multi-head attention. The feed-forward network usually contains two linear layers separated by a nonlinear activation function and further transforms and abstracts the attention output. It is expressed as:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{6}$$

The Transformer encoder contains two layers, each with four attention heads, a model dimension of 64, and a feed-forward hidden dimension of 128. Multi-head attention learns temporal dependencies in different subspaces, including short-term fluctuations, intermediate trends, and changes in trading volume. Each encoder layer uses residual connections, LayerNorm, and Dropout to stabilize training and reduce overfitting.

4.5. Residual Output Layer

Short-term stock prices have great continuity, and the future is very close to the present. So it's going to be a big deviation if we directly predict an absolute price in the future. Introduce a residual

prediction head that learns how much change there will be between the closing price five trading days ahead of time and the current closing price. To be precise, Transformer encoder creates a residual item, adding it to the closing price at the end of the input window to get forecast. This structure matches the Persistence benchmark, letting the model also learn extra price movement after very strong price stickiness.

4.6. Loss Function and Training Procedure

The model uses SmoothL1 loss and the AdamW optimizer, with a learning rate of 0.0005 and a batch size of 32 The final 10% of the training period is set aside chronologically for validation purposes Early stopping stops the training if the loss on validation does not decrease after certain number of epochs passed. After being trained, it will generate rolling-window predictions for test data and convert them back to actual U.S.dollar prices Training procedure has five steps Firstly, raw stocks date cleaning, normarlizing. Secondly, sliding windows to build inputs and targets Thirdly use the training samples to fit Transformer Fourthly, forecast for testing Lastly, RMSE, MAE and MAPE are calculated for comparison in Persistence, residual XGBoost, LSTM, vanilla Transformer, standard feature-gated Transformer and proposed model.

5. Experimental Design

5.1. Parameter Settings

All deep-learning models use the same training/test split, window length, and evaluation metrics. Table 1 summarizes the main experimental settings.

Table 1. Summary of experimental hyperparameters.

Parameter	Value	Parameter	Value
Sliding-window length	30.0000	Forecast horizon	5.0000
Input-feature dimension	8.0000	Attention heads	4.0000
Training/test ratio	7:3	Model dimension	64.0000
Learning rate	0.0005	Batch size	32.0000
Optimizer	AdamW	Loss function	SmoothL1
Epochs	80.0000	Early-stopping patience	10.0000

5.2. Benchmark Models

We use the following benchmarks to test our model: persistence, residual XGBoost, LSTM, vanilla Transformer, standard feature-gated Transformer, and improved Transformer. Persistence is simply taking current close as a forecast for 5 days ahead which forms a very solid no-learning baseline on price prediction in short horizon. Residual XGBoost flattens the historical window of size 30×8 into 240 inputs, then employs gradient-boosted trees to learn the residuals between the closing price at t+5 and the final close in the input window [3]. LSTM represents the recurrent-neural-network baseline [4,5]. Vanilla transformer is tested for forecasting by using just self – attention [8]. The standard feature-gated Transformer evaluates feature gating without residual output. proposed Transformer incorporates feature-gated embedding with attention pooling, together with a residual prediction head.

5.3. Evaluation Metrics and Interpretability Analysis

RMSE, MAE, and MAPE are taken as the main metrics for evaluation. RMSE imposes greater punishment on large errors, MAE measures average absolute deviation from expected value, and MAPE calculates relative forecasting error. all three metrics after the predicted values are inverse transformed to actual prices. As there is an explicit feature-gating layer in the proposed model, we also report the average gate weight of each feature over the AAPL test set so understand how much emphasis the model gives relatively to different input variables compared with others This analysis

follows the general approach of measuring variable contributions in interpretable multivariate time-series models [10, 24].

6. Results and Discussion

6.1. Main Results for AAPL

Table 2 gives the five-day-ahead closing price forecast for AAPL Residual XGBoost has an RMSE of 1.3704, MAE of 1.0345, and MAPE of 2.8116%, which is significantly better than a tree model that predicts absolute prices directly. The target transformed into a residual relative to current close generates good baseline by machine learning Comparatively LSTM achieves an RMSE value of 8.7437 with a MAPE of 12.9145%:;the standard Transformer gives an RMSE of 10.7798 along with a MAPE of 17.4697%; while the typical feature-gated Transformer yields an RMSE score of 12.2998 alongside a MAPE of 20.4662%. Under this condition, deep learning models without residuals outputs have large drifts in prices and cannot catch up with the closing price in the next five days continuously.

As shown in Figure 2 and 3, the proposed model is best on AAPL with RMSE 1.2712, MAE 0.9242, and MAPE 2.5268%. It has some improvement over Persistence, whose RMSE, MAE, and MAPE are 1.2824, 0.9366, and 2.5546%, respectively. And it beats residual XGBoost, which gets an RMSE of 1.3704, MAE of 1.0345, and MAPE of 2.8116%. Due to Persistence as well as the residual XGBoost are good short-term price forecasters, the added error decrease for AAPL shows that feature gating along with residual prediction assist in getting local movement learned without destroying continuity.

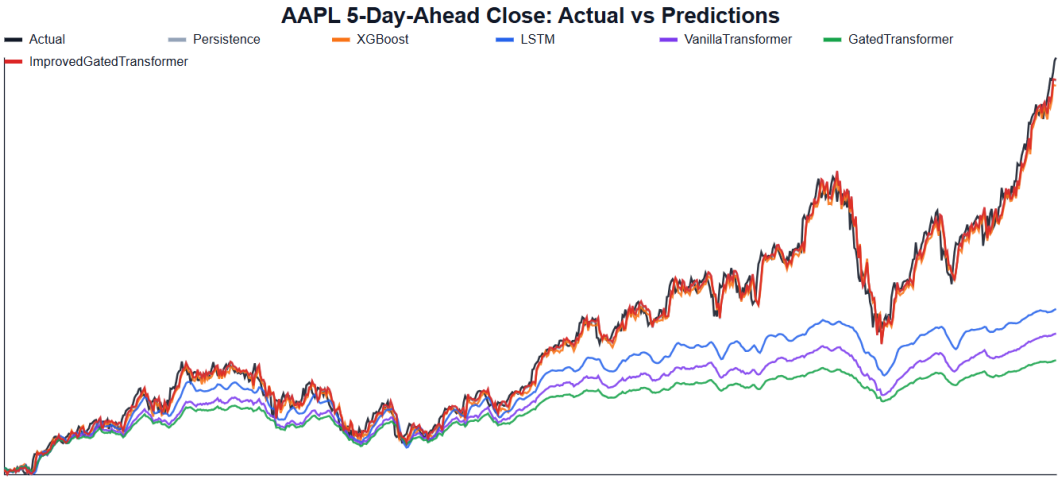


Figure 2. Actual and predicted AAPL closing prices on the test set.

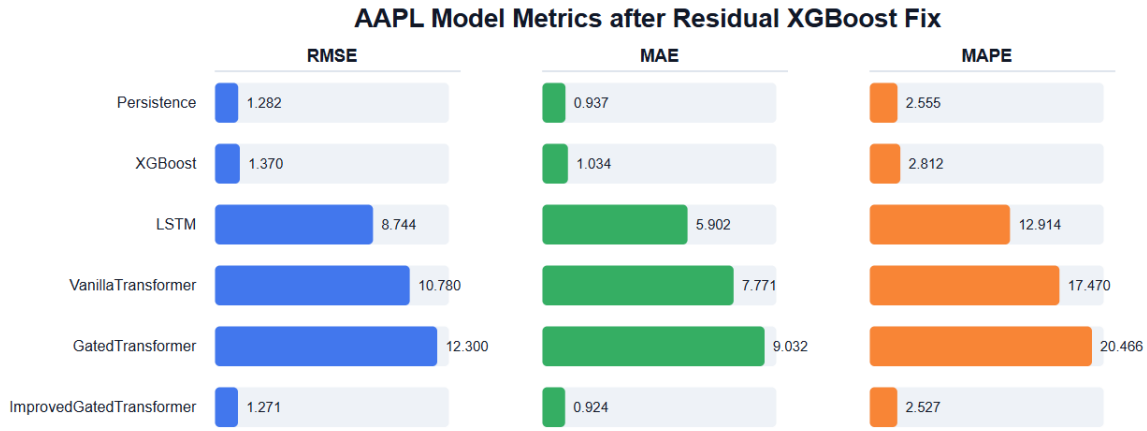


Figure 3. Comparison of model error metrics on AAPL.

Table 2. Performance comparison on the AAPL dataset.

Model	RMSE	MAE	MAPE (%)
Persistence	1.2824	0.9366	2.5546
Residual XGBoost	1.3704	1.0345	2.8116
LSTM	8.7437	5.9015	12.9145
Vanilla Transformer	10.7798	7.7714	17.4697
Gated Transformer	12.2998	9.0318	20.4662
Improved Transformer	1.2712	0.9242	2.5268

6.2. Cross-Stock Generalization

To evaluate applicability for different stocks, we repeat the experiment on MSFT and GOOG and report results on all three stocks in Table 3. As shown in Figure 4, the improved model greatly exceeds LSTM, plain Transformer, and regular feature-gated Transformer in all three stocks. For the stock GOOG they had RMSE of 13.0102, 7.3535, 8.6631 compared to our proposed model achieving an RMSE score of 1.5776 For the stock MSFT they achieved RMSE scores of 24.2564, 27.4112, 27.1484 against our proposed model achieving a RMSE score of 2.5862. Residual XGBoost is also steady across these three stocks with RMSEs of 1.3704, 1.6265 and 2.5426 for AAPL, GOOG and MSFT respectively. This shows that residual targets really help make trees more suitable when testing for upward trends.

Persistence is very competitive on both GOOG and MSFT. In terms of RMSE and MAPE, on the basis of Persistence model in GOOG has an RMSE of 1.5746 and a MAPE of 2.4427%, whereas the proposed model has an RMSE of 1.5776 and a MAPE of 2.4510%; these are nearly identical, and the proposed model marginally outperforms residual XGBoost whose RMSE and MAPE are 1.6265 and 2.6462% respectively. For MSFT: persistence attains rmse of 2.2431, mape of 2.1499%, residual xgboost reaches rmse of 2.5426, mape of 2.4329%, and our proposed method gains 2.5862 as rmse and 2.5575 as mape, respectively. Though slightly worse than those two good baselines, it still greatly beats LSTM and those two non-residual Transformers. These results confirm price's strength for five-day ahead forecast. The main advantage of the proposed model lies in its relatively stable error control compared to common deep learning baselines and good competitiveness against strong benchmarks for AAPL and GOOG.

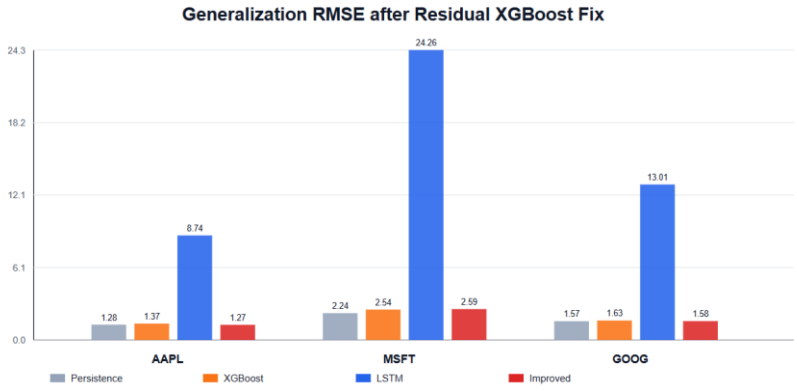


Figure 4. RMSE comparison in the cross-stock generalization experiment.

Table 3. Cross-stock generalization results.

Stock	Metric	Persistence	Residual XGBoost	LSTM	Vanilla Transformer	Gated Transformer	Improved Transformer
AAPL	RMSE	1.2824	1.3704	8.7437	10.7798	12.2998	1.2712
AAPL	MAPE	2.5546	2.8116	12.9145	17.4697	20.4662	2.5268
GOOG	RMSE	1.5746	1.6265	13.0102	7.3535	8.6631	1.5776
GOOG	MAPE	2.4427	2.6462	20.9786	11.0516	13.2919	2.4510
MSFT	RMSE	2.2431	2.5426	24.2564	27.4112	27.1484	2.5862
MSFT	MAPE	2.1499	2.4329	15.3852	17.3379	16.8177	2.5575

6.3. Ablation Study

An ablation study on AAPL to test the effect of each structural part as presented in Table 4: The raw Transformer has poor performance with RMSE equal to 17.3182 and MAPE of 32.4154%. Adding a normal linear embedding leads to a reduction in RMSE to 10.7798 and MAPE to 17.4697%, which shows that suitable features can enable the Transformer to process multivariate financial sequences. Adding feature-gated embedding gets an RMSE of 12.2998 and a MAPE of 20.4662 %, which is not improved compared with adding only linear embedding. Feature gate alone is insufficient for solving drift problems related to direct short term price prediction.

When the residual prediction head is added to the feature gated setup, RMSE tumbles down drastically to 1.2712 and MAPE plummets way down to 2.5268%. so, the improvement of the final model does not come only from picking features or designing attention but also lies in matching how predictions are made against what financial price series really look like Residual form changes it into trying to learn for a future change over current price instead of making an absolute forecast, this brings down learning difficulty as well as those big errors by doing direct price forecasts.

Table 4. Results of the ablation study.

Model	RMSE	MAE	MAPE (%)
Raw Transformer	17.3182	13.7014	32.4154
+ Linear Embedding	10.7798	7.7714	17.4697
+ Feature-Gated Embedding	12.2998	9.0318	20.4662
+ Residual Prediction Head (Improved)	1.2712	0.9242	2.5268

6.4. Feature-Gate Importance

In order to test the model for its focus on particular inputs, we calculate the average feature-gate output values on AAPL. As shown in Figure 5 and Table 5, Open, Volume and MACD get higher weight averagely, it shows that when predicting 5 day forward close price model pays attention to opening-price info, market activity & change in trend. High, Low and ATR can tell us extra detail about how much the prices are ranging and also how risky it is due to fluctuation RSI and Close have lesser weights as intense short term continuity of price is already captured directly through residual outputs. Gate Weights are not strict causal effects but instead indicate comparative responses from gates given different input features into our model.

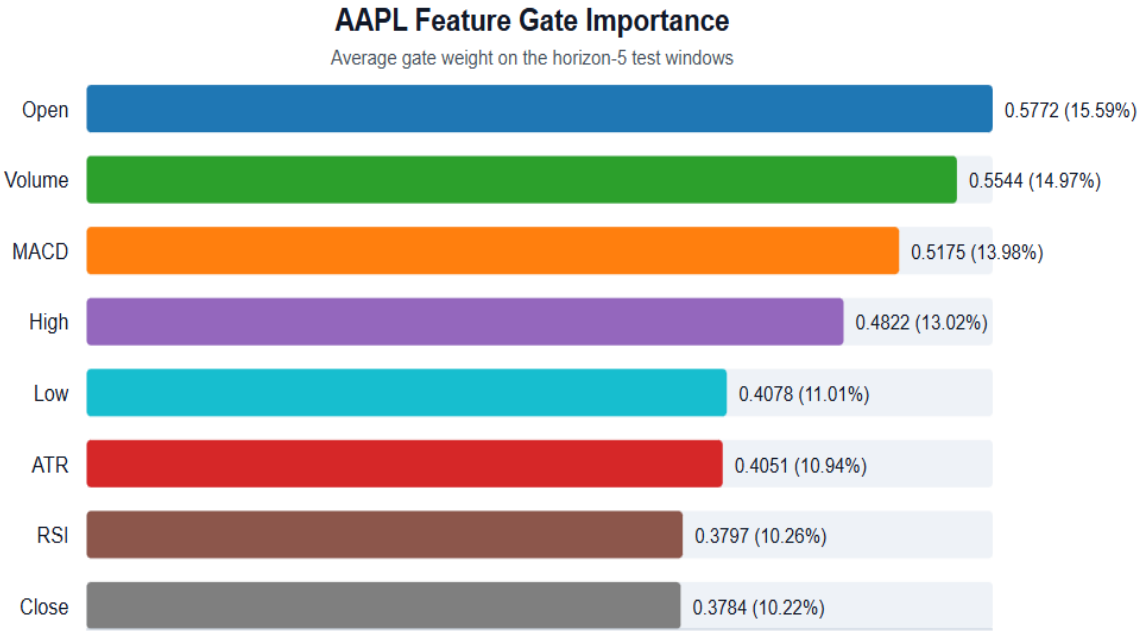


Figure 5. Feature-gate importance on the AAPL test set.

Table 5. Feature-gate importance for AAPL.

Feature	Average Gate Weight	Normalized Contribution
Open	0.5772	15.59%
Volume	0.5544	14.97%
MACD	0.5175	13.98%
High	0.4822	13.02%
Low	0.4078	11.01%
ATR	0.4051	10.94%
RSI	0.3797	10.26%
Close	0.3784	10.22%

6.5. Supplementary Five-Day Return Experiment

In addition to predicting the closing price five trading days ahead, a supplementary experiment forecasts the corresponding five-day return, defined as $r_{t,5} = Close_{t+5}/Close_t - 1$. The model first predicts the five-day return and then reconstructs the future price from the current close as predicted $Close_{t+5} = Close_t(1 + \text{predicted } r_{t,5})$. This experiment adds returns, moving averages, and historical volatility calculated from the original OHLCV data and requires no external data source.

As shown in Figure 6 and Table 6, the supplementary AAPL results show that return forecasting is more difficult and that price-error differences across models are small. The Zero Return baseline has a Price RMSE of 1.2837 and a Price MAPE of 2.5592%, while the proposed model has a Price RMSE of 1.2820 and a Price MAPE of 2.5592%; the two are nearly identical. However, directional accuracy rises from 41.25% for ZeroReturn to 52.08% for the proposed model. Thus, although the model does not substantially reduce the numerical error of return forecasts, it shows some ability to identify the direction of future returns. Five-day-ahead closing-price forecasting is therefore retained as the main experiment, while return forecasting is used as a robustness and extension analysis.

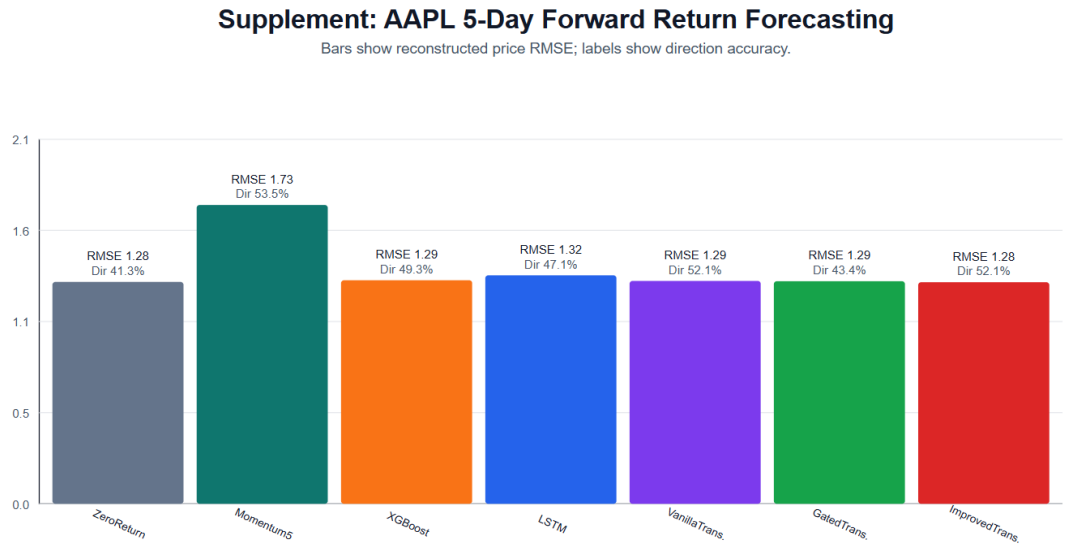


Figure 6. Metrics for the supplementary five-day return experiment on AAPL.

Table 6. Supplementary five-day return-forecasting results for AAPL.

Model	Price RMSE	Price MAPE (%)	Directional Accuracy (%)
Zero Return	1.2837	2.5592	41.2526
Five-Day Momentum	1.7290	3.5022	53.5058
XGBoost	1.2935	2.5720	49.2852
LSTM	1.3220	2.6109	47.1069
Vanilla Transformer	1.2894	2.5576	52.1443
Gated Transformer	1.2878	2.5733	43.3628
Improved Transformer	1.2820	2.5592	52.0762

7. Conclusion

This paper proposes an improved Transformer model that incorporates feature-gated embedding along with a residual prediction head for multivariate stock time series forecasting. The daily data on AAPL, MSFT, and GOOG are represented by eight features: Open, High, Low, Close, Volume, RSI, ATR, and MACD. Previous 30 days of trading will be used to forecast the closing price five days ahead. in order to prevent leakage of data, it uses chronological split and normalization only on training set as well as separate window creation in test set.

The proposed model obtains the best results on the main AAPL experiment, slightly beating the strong persistence and residual XGBoost benchmarks, and significantly beating LSTM, vanilla Transformer, standard feature-gated Transformer. Cross stock tests: It’ s almost the same as Persistence on GOOG, but better than residual XGBoost. On MSFT it's a little worse than persistence and residual XGBOOST but still much better than the other deep learning baselines. From above we could see that price persistence together with its residual machine-learning benchmark is also quite strong in the area of short-term stock prediction, and from more strict comparison it can also be observed that residual Transformer has certain robustness. From the ablation test we know that the residual prediction head gives the main benefit because it turns the task into figuring out how much prices will deviate from where they are now in the future.

Future research could expand this study in the following four directions. Firstly, Macro-economy Indicators, Sector Indices, Corporate Finance Data and News Sentiment might be able to make the information more complete. The second is more work on return forecasts and directional forecasts will improve the usefulness of making trading decisions. The third way is that longer sample periods as well as a broader selection of stocks would allow stronger testing of the generalization capability. Fourthly, time-series-specific Transformer architectures or multiscale attention mechanisms [11–18] can better represent the complex dynamics of financial markets

References

[1] Box G E P, Jenkins G M, Reinsel G C, Ljung G M. Time Series Analysis: Forecasting and Control[M]. 5th ed. Hoboken: Wiley, 2015.

[2] Bollerslev T. Generalized autoregressive conditional heteroskedasticity[J]. Journal of Econometrics, 1986, 31(3): 307-327.

[3] Chen T, Guestrin C. XGBoost: A scalable tree boosting system[C] //Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016: 785-794.

[4] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[5] Fischer T, Krauss C. Deep learning with long short-term memory networks for financial market predictions[J]. European Journal of Operational Research, 2018, 270(2): 654-669.

[6] Gu S, Kelly B, Xiu D. Empirical asset pricing via machine learning[J]. The Review of Financial Studies, 2020, 33(5): 2223-2273.

[7] Zhang Z, Zohren S, Roberts S. DeepLOB: Deep convolutional neural networks for limit order books[J]. IEEE Transactions on Signal Processing, 2019, 67(11): 3001-3012.

[8] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C] //Advances in Neural Information Processing Systems. 2017, 30: 5998-6008.

[9] Qin Y, Song D, Chen H, et al. A dual-stage attention-based recurrent neural network for time series prediction[C]//Proceedings of the 26th International Joint Conference on Artificial Intelligence. 2017: 2627-2633.

[10] Lim B, Arik S O, Loeff N, Pfister T. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting[J]. International Journal of Forecasting, 2021, 37(4): 1748-1764.

[11] Zhou H, Zhang S, Peng J, et al. Informer: Beyond efficient Transformer for long sequence time-series forecasting[C] //Proceedings of the AAAI Conference on Artificial Intelligence. 2021, 35(12): 11106-11115.

[12] Wu H, Xu J, Wang J, Long M. Autoformer: Decomposition Transformers with auto-correlation for long-term series forecasting[C]//Advances in Neural Information Processing Systems. 2021, 34: 22419-22430.

[13] Zhou T, Ma Z, Wen Q, et al. FEDformer: Frequency enhanced decomposed Transformer for long-term series forecasting[C] //Proceedings of the 39th International Conference on Machine Learning. 2022: 27268-27286.

[14] Zeng A, Chen M, Zhang L, Xu Q. Are Transformers effective for time series forecasting?[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2023, 37(9): 11121-11128.

[15] [15] Wen Q, Zhou T, Zhang C, et al. Transformers in time series: A survey[C]//Proceedings of the 32nd International Joint Conference on Artificial Intelligence. 2023: 6778-6786.

[16] Nie Y, Nguyen N H, Sinthong P, Kalagnanam J. A time series is worth 64 words: Long-term forecasting with Transformers[C]//International Conference on Learning Representations. 2023.

[17] Wu H, Hu T, Liu Y, et al. TimesNet: Temporal 2D-variation modeling for general time series analysis[C] //International Conference on Learning Representations. 2023.

[18] Liu Y, Hu T, Zhang H, et al. iTransformer: Inverted Transformers are effective for time series forecasting[C] //International Conference on Learning Representations. 2024.

[19] Wang C, Chen Y, Zhang S, Zhang Q. Stock market index prediction using deep Transformer model[J]. Expert Systems with Applications, 2022, 208: 118128.

[20] Zhang Q, Qin C, Zhang Y, et al. Transformer-based attention network for stock movement prediction[J]. Expert Systems with Applications, 2022, 202: 117239.

[21] Peng Z Y, Guo P C. A data organization method for LSTM and Transformer when predicting Chinese banking stock prices[J]. Discrete Dynamics in Nature and Society, 2022: 7119678.

[22] Ma T, Wang W, Chen Y. Attention is all you need: An interpretable Transformer-based asset allocation approach[J]. International Review of Financial Analysis, 2023, 90: 102876.

[23] Thwal C M, Tun Y L, Kim K, Park S B, Hong C S. Transformers with attentive federated aggregation for time series stock forecasting[C] //37th International Conference on Information Networking. IEEE, 2023: 499-504.

[24] Hu J, Shen L, Sun G. Squeeze-and-excitation networks[C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 7132-7141.