Stock Price Forecasting Using a Transformer with Time-Interval-Based Attention and Trainable Stock Selection

Hongkai Jiang, Baiting Wu, Xiaolin Hu*
Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China.
*Corresponding Author. Email: xlhu@mail.tsinghua.edu.cn

***Abstract*—Recently, there has been growing interest in predicting stock price movements by exploring the relationships between stocks. However, existing methods focus on the correlation between time steps and aggregate features from all stocks, regardless of the number of stocks, which leads to several limitations. First, in time-series data, especially stock data, the correlation between time intervals is more indicative of future trends than correlations at single time steps. Second, not all features contribute equally to stock price movements. Finally, not all stocks are correlated, and aggregating information from unrelated stocks introduces noise into the prediction. To address these challenges, we introduce TITAN, a Transformer with Time-Interval-Based Attention and Trainable Stock Selection, enabling the model to capture cross-time correlations, focus on the most informative features, and dynamically select the most relevant stocks. TITAN simulates complex stock correlations by aggregating information across time, feature, and stock dimensions. While validated on stock forecasting, the proposed network advances neural network architectures for multivariate time-series modeling more broadly, particularly in domains with noisy, high-dimensional, or relational data. Experiments on stock market indices demonstrate the effectiveness of TITAN compared to current state-of-the-art methods and highlight the importance of each module through ablation studies.**

***Index Terms*—Deep learning, stock movement prediction, attention mechanism, gumbel-softmax.**

I. Introduction

Stock price prediction is a critical area of research in financial forecasting, with the potential to significantly enhance investment strategies. As financial markets are influenced by a complex set of factors, predicting future stock prices requires methods that can effectively capture and analyze the interactions within stock data. Recent research has focused on leveraging advanced machine learning techniques—particularly attention mechanisms from the vanilla Transformer [1]—to model these relationships and improve prediction accuracy. However, despite the progress made, several issues persist in current methods, limiting their effectiveness in providing robust predictions.

One of the main problems with existing approaches is that they typically emphasize relationships between individual time steps rather than between time intervals [2]–[4]. In time-series prediction, especially for stock prices, time intervals—such as specific trading periods or market cycles—carry meaningful patterns that can offer deeper insights into future trends. As shown in Figure 1, the similarity between trading data on individual days, such as those highlighted in the red boxes, may not directly correlate with similar future performance. However, a consistent trend in the data, as indicated by the green boxes, is often a stronger signal that the trend is likely to continue. In other words, patterns observed over time can provide valuable insights into potential future movements, as trends tend to persist rather than being solely driven by isolated daily fluctuations. Therefore, by focusing primarily on individual time steps, these methods may overlook crucial interval-level dependencies, limiting their ability to model long-term market behaviors effectively. To address this issue, we propose a new architecture that computes attention across different lags in a time sequence, aggregating information from similar time intervals.

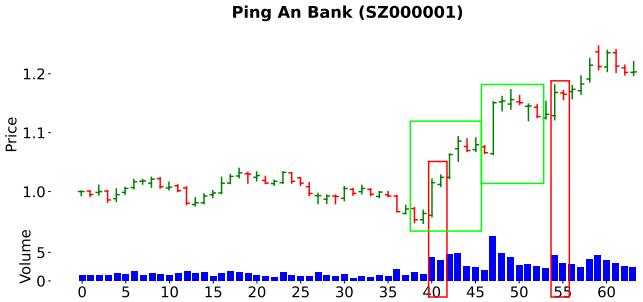


Fig. 1. Price chart of the past 64 trading days for Ping An Bank (SZ000001), listed on China's SZSE Component Index, as of January 10, 2013.

The second issue with existing methods is their lack of mechanisms for identifying indicative features. In stock prediction, datasets are often high-dimensional, containing numerous features related to trading price, volume, and other factors. However, a key insight from real-world investors is that the effectiveness of features typically changes dynamically rather than remaining consistent over time [5]–[7]. In traditional finance, investors often rely on statistical methods to identify features with strong signals [8]–[10], which is extremely labor-intensive. A prior work, MASTER [4], introduces a gating mechanism that performs feature selection by multiplying individual stock features with market-level features. While this approach provides a form of soft feature weighting, it applies the same gate across all stocks and lacks the capacity to adaptively select features based on stock-specific relevance.

Inspired by the cross-dimension attention in Crossformer [11], we introduce an attention mechanism over the feature dimension, enabling the model to more effectively focus on correlations.

The third problem is that existing methods are designed to aggregate features from different stocks, but none are explicitly built to select the most relevant ones. Stock data is characterized by a high level of noise and relatively weak signals. While aggregating information from other stocks can enhance predictive power, it can also introduce additional noise if the stocks are not strongly correlated. As the stock pool grows larger, aggregating from all other stocks often leads to reduced effectiveness, which we will demonstrate in Section IV-C. Previous methods have attempted to restrict stock aggregation using pre-defined domain knowledge [2], [12] or by applying thresholds on correlation scores between stocks [3]. However, these approaches are typically too rigid and cannot be integrated into the training process due to their non-differentiable nature. In this work, we apply Gumbel-Softmax [13] to enable differentiable stock selection, allowing the model to dynamically focus on the most relevant stocks during training.

In particular, we propose TITAN (**T**ransformer with Time-**I**nterval-Based Attention and **T**rain**a**ble Stock Selectio**n**), a method that explores the relationships between time intervals and features, and selects related stocks in a trainable manner. Although our study is grounded in financial forecasting, the methodological contributions of TITAN are not finance-specific. The proposed time-interval-based attention mechanism provides a general solution for modeling temporal dependencies across different time intervals. Similarly, the trainable entity selection module based on Gumbel-Softmax introduces a general neural mechanism for differentiable subset selection, which is broadly applicable to problems involving noisy or weakly correlated entities. Thus, our work contributes not only to financial forecasting but also to the design of more general neural architectures for complex sequential data. Our method achieves superior prediction accuracy compared to state-of-the-art approaches. The contributions of this work are summarized as follows:

- We propose a novel architecture for stock price movement forecasting, including an attention mechanism that learns correlations between time intervals and features.
- We introduce a new approach for selecting correlated stocks in large stock pools. To the best of our knowledge, we are the first to apply such a method for stock selection.
- We conduct comprehensive experiments that demonstrate the superiority of our model over existing methods.

II. Related Work

In the last decade, machine learning has become a powerful tool for forecasting stock price movements, often outperforming traditional linear regression methods. These models can be broadly categorized into two main types: (i) methods based on historical data and (ii) methods leveraging correlations between stocks. Type (i) approaches, including LSTM [14], DA-RNN [15], and AdaRNN [16], use RNNs and attention mechanisms to capture temporal patterns, with Transformer-based architectures [11], [17]–[19] now achieving state-of-the-art performance. Type (ii) approaches model inter-stock relationships, often combining RNNs or attention mechanisms for feature embedding with GNNs for graph-based interactions [2], [3], [12], [20]. While GNNs require a fixed stock pool, attention-based methods [4], [21], [22] flexibly aggregate historical and cross-sectional data, making them more practical in dynamic markets. Building on this, we propose an improved attention-based method that achieves superior performance.

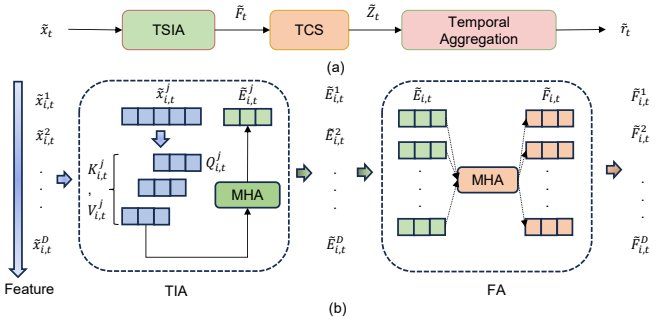


Fig. 2. Overview of the TITAN framework. (a) The overall pipeline of TITAN. (b) The TSIA module.

III. Problem Definition

Stock price forecast is commonly approached as a regression problem. Let us denote the set of N stocks in the market as $S = \{s_1, s_2, \ldots, s_N\}$. For each stock s_i, $i \in N$, a group of predictive factors are collected at each trading day $t \in [1, T]$ to form its feature $x_{i,t} \in \mathbb{R}^D$ where D is the dimensionality of the features. The objective is to predict the future return ratio r_i for each stock based on its historical features.

Following the typical setting in existing stock price forecast works [2]–[4], [12], [23], we forecast return ratios instead of the absolute value of prices. The return ratio r_i over a predetermined interval d days is calculated as:

$$r_i = \frac{p_{i,t+d} - p_{i,t+1}}{p_{i,t+1}}, \tag{1}$$

where $p_{i,t}$ is the closing price of stock s_i at time t, and d represents the prediction interval. This return ratio captures the relative price change of the stock over the prediction horizon.

IV. Model Architecture

A. Overview

The general pipeline of TITAN is shown in Figure 2a, consisting of three steps: (1) Two-Stage Intra-Stock Aggregation (TSIA): For each stock's feature sequence, we divide the sequence into overlapping time intervals of fixed length. Then, for each feature dimension, we aggregate information from other time intervals to learn the relationships between intervals rather than individual time steps. Self-attention is then applied

across features to capture dependencies along the feature dimension, allowing us to identify the most contributive signals. (2) Trainable Cross-Stock Selection (TCS): We gather relevant stock features using an attention mechanism, where each stock aggregates the feature embeddings of other stocks based on the learned attention weights. (3) Temporal Aggregation: The feature embedding from the last time step is used as a query to aggregate all previous information, which is then used to make the final prediction. In the following sections, we elaborate on the details of each step of our method.

B. TSIA

We use two stages of aggregation: one in the time dimension (Time-Interval Aggregation) and the other in the feature dimension (Feature Aggregation), which are shown in Figure 2b. These stages are designed to learn the relationships between time intervals within each individual stock's data, enabling more effective learning of temporal and feature-based dependencies.

1) Time-Interval Aggregation (TIA): To leverage historical data, for each trading day t, we use features from the past M days, denoted as $\mathbf{x}_{i,t-M+1:t}$, for each stock s_i to make predictions. For simplicity, we refer to this sequence of features as $\tilde{\mathbf{x}}_{i,t} \in \mathbb{R}^{M\times D}$. Unlike previous methods that either aggregate all past trading days into a single representation at the latest trading day [2], [21] or apply self-attention across the entire sequence [4], we embed different lags within the sequence as individual time steps. As illustrated in Figure 2, we apply a sliding window of size m (interval length) with stride 1 over each M-day sequence to generate overlapping sub-sequences. We illustrate use the latest m days (in the figure, $m = 3, M = 5$) of $\tilde{\mathbf{x}}_{i,t}$ as the query $Q_{i,t}$, and apply Multi-head Attention (MHA) [1] to capture its relationships with all preceding intervals.

Note that combining all lags of $Q_{i,t}$ creates a 3D matrix, which requires dimensionality reduction for attention calculation. For this reason, we perform MHA across time intervals within each feature dimension. For the sequence of each feature $\tilde{\mathbf{x}}^j_{i,t}$, $j \in D$, with query $Q^j_{i,t}$ and all its lags as key and value $K^j_{i,t}$, $V^j_{i,t}$, the attention mechanism is performed as follows:

$$\tilde{\mathbf{E}}^j_{i,t} = \text{MHA}(\mathbf{Q}^j_{i,t}, \mathbf{K}^j_{i,t}, \mathbf{V}^j_{i,t}). \tag{2}$$

Then, $\tilde{\mathbf{E}}_{i,t} \in \mathbb{R}^{m\times D}$ is the concatenation of sequences from each feature's intervals, where each time step aggregates information from different lags of $Q_{i,t}$ and embeds correlations between time intervals of each feature sequence. As a result, when calculating correlations between stocks, the attention mechanism captures trends across different stocks by attending to similar lags in their sequences.

2) Feature Aggregation (FA): After the cross-time aggregation, we perform aggregation in the feature dimension. Applying the attention mechanism in the feature dimension offers two advantages. First, it enables the automatic identification of more influential features, as the attention weights within the aggregation of other features are adjusted during the training process. Second, it facilitates the exploration of cross-feature correlations between stocks during the cross-stock aggregation step. Previous methods [4], [21] capture the cross-time correlation between stocks, which represent the correlation between the same features of two stocks at different time steps, meaning the correlation is always between the same features. By applying the attention mechanism in the feature dimension, we can capture relationships between different features across multiple stocks.

Specifically, after obtaining $\tilde{E}_{i,t} \in \mathbb{R}^{m\times D}$ from time-interval aggregation, we apply self-attention over the feature dimension to compute correlations among the D feature sequences of length m. We transpose $\tilde{E}_{i,t}$ to $\tilde{E}'_{i,t} \in \mathbb{R}^{D\times m}$, set $Q^E_{i,t}, K^E_{i,t}, V^E_{i,t} = \tilde{E}'_{i,t}$, and perform multi-head attention as follows:

$$\tilde{F}'_{i,t} = \text{FFN}(\text{MHA}(Q^E_{i,t}, K^E_{i,t}, V^E_{i,t}) + \tilde{E}'_{i,t}). \tag{3}$$

Here, FFN denotes the feed-forward layers in the residual connection. The result $\tilde{F}'_{i,t}$ is then transposed back to $\tilde{F}_{i,t} \in \mathbb{R}^{m\times D}$, producing the feature embedding for each stock at time t. This embedding aggregates information from both time and feature dimensions, enabling the model to capture complex relationships across time steps and features in subsequent steps.

C. TCS

In this module, we use an attention mechanism to aggregate information from correlated stocks by modeling their relationships within the stock dimension. While attention mechanism is effective for calculating stock correlations, it has limitations. In an index with hundreds of stocks, not all stocks are correlated. When aggregating information using attention, the mechanism incorporates data from all stocks based on attention weights, introducing noise due to the high-noise, weak-signal nature of stock data. This issue is less significant with smaller stock pools, but as the stock pool increases, the noise can overwhelm the signal. We conducted a controlled experiment by randomly sampling different numbers of stocks from China's CSI800 index and comparing prediction accuracy with and without cross-stock aggregation using the current SOTA method [4]. The difference between two methods, shown in Figure 3[1], reveal that the benefit of cross-stock aggregation diminishes as the stock pool grows and disappears entirely when all 800 stocks are included. For clarity, the metrics for the "800" group is the whole CSI800 index since the maximum number of stocks we can get from CSI800 are 800 stocks. All other experiments used randomly sampled stocks.

As a result, a method that limits the number of stocks for aggregation is needed. Several approaches exist for stock selection, such as setting a threshold on attention weights [3] or choosing stocks with the highest attention weights. However, discrete selection is non-differentiable and cannot be trained via gradient descent. Moreover, directly selecting

[1]The explanation of the metrics can be found in section V-D.

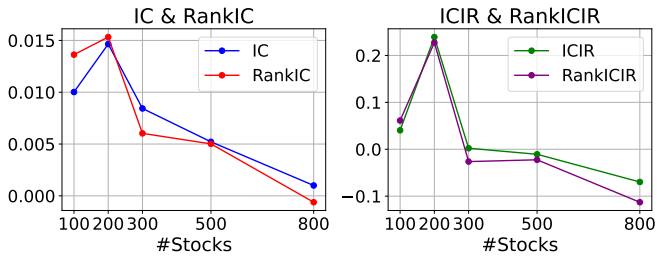


Fig. 3. Comparison of prediction accuracy metrics with different number of stocks. The values in the figures represent the difference in performance between models with and without cross-stock aggregation. In each panel we show the average value over 5 randomly sampled groups of stocks except the last point which corresponds to the results of all 800 stocks from CSI800.

stocks can be too rigid, potentially overlooking subtle relationships. To address this, we apply the Gumbel-Softmax trick [13], which approximates discrete selection as a differentiable process. By treating each stock as a category with a probability proportional to its attention weight, selecting the most relevant stocks becomes equivalent to sampling from a categorical distribution. The Gumbel-Softmax function provides a continuous, differentiable approximation to the $\arg\max$ function [13], as shown below:

$$y_i = \frac{\exp\left((\log(\pi_i) + g_i)/\tau\right)}{\sum_{j=1}^{N} \exp\left((\log(\pi_j) + g_j)/\tau\right)} \quad \text{for } i = 1, ..., N. \tag{4}$$

Here, π_i represents the probability (logit) of the discrete category, and g_i are i.i.d. samples drawn from a Gumbel distribution on $[0, 1]$. The parameter τ, called the softmax temperature, controls the smoothness of the selection process. A low temperature (close to 0) makes the selection more deterministic, approximating a one-hot vector where one category has probability near 1 and others near 0. As τ increases (typically between 1.0 and 10.0), the distribution flattens, spreading probabilities more evenly across categories and allowing multiple categories to be selected.

In our case, after performing intra-stock aggregation and obtaining the embedded features $\tilde{F}_t \in \mathbb{R}^{N\times m\times D}$, we transpose the stock and time dimensions to obtain $\tilde{F}'_t \in \mathbb{R}^{m\times N\times D}$. Then, at each time step t within the time window m, we compute attention between features from N stocks, denoted as $F'_t \in \mathbb{R}^{N\times D}$, which serve as the query, key, and value matrices: Q_t^F, K_t^F, and V_t^F, respectively. Let Z_t^a represent the attention weights, computed as the dot product between Q_t^F and K_t^F. Before multiplying the attention weights with the value V_t^F, we interpret Z_t^a as the category probabilities π_i in Equation 4. We then apply the Gumbel-Softmax function to Z_t^a to obtain the final prediction:

$$Z'_t = \text{FFN}\left(\text{Gumbel-Softmax}(Z_t^a) \cdot V_t^F + F'_t\right). \tag{5}$$

Finally, we transpose the stock and time dimensions back to obtain $\tilde{Z}_t \in \mathbb{R}^{N\times m\times D}$. This enables the model to learn to select more relevant stocks in a differentiable manner, thereby reducing noise when aggregating features from other stocks.

D. Temporal Aggregation

After embedding the features through time, stock, and feature aggregation, we apply a simple attention mechanism for each stock. At the latest time step t, the feature $Z_{i,t}$ serves as the query, and the entire sequence $\tilde{Z}_{i,t}$ as both key and value. A fully connected layer (FC) then generates the final prediction $\hat{r}_i$ as follows:

$$\hat{r}_i = \text{FC}(\text{MHA}(Z_{i,t}, \tilde{Z}_{i,t}, \tilde{Z}_{i,t})). \tag{6}$$

The prediction accuracy is optimized using an MSE loss function with respect to the ground truth return r_i:

$$loss = \sum_{i\in N} \text{MSE}(r_i, \hat{r}_i). \tag{7}$$

V. Experiments

A. Datasets

We used the datasets from previous work [4], which included daily trading data for the CSI300 and CSI800 indices from China's A-share market. Basic information about the training, validation, and test datasets is provided in Table I. The datasets consisted of two feature groups: the first group included 158 Alpha indicators for individual stocks, inherited from Microsoft Qlib [24], while the second group contained 63 features derived from the CSI300, CSI500, and CSI800[2] indices. These features primarily captured variations in trading price and volume over different time spans, including 1, 5, 10, 20, 30, and 60 days. The prediction labels were defined as the stock return over the subsequent 5 trading days. Number of stocks in CSI300 and CSI800 datasets are shown in Table I.

TABLE I
Basic Information of Datasets

Dataset	Periods	#Stocks
Train	Q1 2008-Q1 2020	668 & 1543
Valid	Q2 2020	319 & 843
Test	Q3 2020-Q4 2022	420 & 996

B. Baseline Methods

We compared our method against a diverse set of baselines, including XGBoost [25], LSTM, TCN [26], Transformer [1], GAT [20], DTML [21], MASTER [4]: the current SOTA method for stock prediction using correlation between stocks.

We did not include GNN-based methods [2], [3], [12] that require a customized dataset with fixed stock pool, as the constituents of the market indices are regularly updated. Note that GAT [20] is also a GNN method but is included because it does not require a fixed stock pool.

[2]CSI800 is the combination of CSI300 and CSI500.

TABLE II
PERFORMANCE ON CSI300 AND CSI800

Dataset	Model	Metrics			
		IC	ICIR	RankIC	RankICIR
CSI300	XGBoost	0.051 ± 0.001	0.37 ± 0.01	0.050 ± 0.001	0.36 ± 0.01
	LSTM	0.049 ± 0.001	0.41 ± 0.01	0.051 ± 0.002	0.41 ± 0.03
	GRU	0.052 ± 0.004	0.35 ± 0.04	0.052 ± 0.005	0.34 ± 0.04
	TCN	0.050 ± 0.002	0.33 ± 0.04	0.049 ± 0.002	0.31 ± 0.04
	Transformer	0.047 ± 0.007	0.39 ± 0.04	0.051 ± 0.002	0.42 ± 0.04
	GAT	0.054 ± 0.002	0.36 ± 0.02	0.041 ± 0.002	0.25 ± 0.02
	DTML	0.049 ± 0.006	0.33 ± 0.04	0.052 ± 0.005	0.33 ± 0.04
	MASTER	0.064 ± 0.006	0.42 ± 0.04	**0.076 ± 0.005**	0.49 ± 0.04
	TITAN	**0.074* ± 0.006**	**0.57* ± 0.03**	**0.076 ± 0.004**	**0.58* ± 0.04**
CSI800	XGBoost	0.040 ± 0.000	0.37 ± 0.01	0.047 ± 0.000	0.42 ± 0.01
	LSTM	0.028 ± 0.002	0.32 ± 0.02	0.039 ± 0.002	0.41 ± 0.03
	GRU	0.039 ± 0.002	0.36 ± 0.05	0.044 ± 0.003	0.39 ± 0.07
	TCN	0.038 ± 0.002	0.33 ± 0.04	0.045 ± 0.002	0.38 ± 0.05
	Transformer	0.040 ± 0.003	0.43 ± 0.03	0.048 ± 0.003	0.51 ± 0.05
	GAT	0.043 ± 0.002	0.39 ± 0.02	0.042 ± 0.002	0.35 ± 0.02
	DTML	0.039 ± 0.004	0.29 ± 0.03	0.053 ± 0.008	0.37 ± 0.06
	MASTER	0.052 ± 0.006	0.40 ± 0.06	0.066 ± 0.007	0.48 ± 0.06
	TITAN	**0.068* ± 0.005**	**0.67* ± 0.03**	**0.069* ± 0.007**	**0.66* ± 0.06**

The best results for each comparison group are highlighted with **bold** and marked with * for statistical significance (by t-test at $p = 0.01$ level).

C. Training Settings

All models were trained for five runs with random initialization, and results were averaged. We used a sequence length of $M = 16$ and interval length $m = 8$ across all datasets. Models were implemented in PyTorch and trained on a single NVIDIA RTX 3090 GPU. The model dimension d_{model} was selected from $\{128, 256, 512\}$, with one attention layer and matching feed-forward dimensions. Training used the Adam optimizer [27] with a learning rate of $1\text{e}{-5}$. The Gumbel–Softmax temperature τ was set to 5.0 (CSI300), 0.1 (CSI800).

D. Evaluation

To evaluate prediction accuracy, we adopt four standard metrics: Information Coefficient (IC), IC Information Ratio (ICIR), Rank Information Coefficient (RankIC), and RankIC Information Ratio (RankICIR). IC measures the Pearson correlation between predicted and actual returns, while RankIC uses the Spearman correlation to assess ranking accuracy. Both metrics are averaged across all trading days in the test period and take values in $[-1, 1]$, with higher values indicating better performance. ICIR and RankICIR further normalize IC and RankIC by their respective standard deviations to provide standardized performance measures. These metrics are widely used in prior stock prediction studies [4], [16], [24].

E. Main Results

The results on the CSI300 and CSI800 datasets, summarized in Table II, show that our method consistently outperforms the baselines across nearly all prediction metrics. Specifically, it achieved a 15.6% and 23% increase in IC over the second-best methods on CSI300 and CSI800, respectively, highlighting superior prediction accuracy.

F. Effect of TSIA

To further examine the effect of modules within out network, we compared TSIA to TSA, which stands for the time-step intra-stock aggregation, on CSI800 index. The results are shown in Table III. We observed a significant improvement

TABLE III
PERFORMANCE OF TSA AN TSIA ON CSI800

Module	Metrics			
	IC	ICIR	RankIC	RankICIR
TSA	0.052 ± 0.004	0.51 ± 0.05	0.055 ± 0.004	0.53 ± 0.04
TSIA	**0.063* ± 0.003**	**0.55* ± 0.08**	**0.064* ± 0.006**	**0.53 ± 0.08**

The best results for each comparison group are highlighted with **bold** and marked with * for statistical significance (by t-test at $p = 0.01$ level).

when using the TSIA module with attention aggregation between time steps on CSI800 index, whose IC had about 20% increase. This highlighted the advantage of incorporating correlations between time intervals and features, which enhanced the model's ability to capture temporal dependencies and improve predictive accuracy.

Based on our observations, TSA tended to focus primarily on the most recent day (i.e., the current day). On the contrary, while TSIA also mainly focused on the current day, it was capable of capturing patterns from previous days. This behavior is reflected by the average attention patterns observed across all test samples, presented in Figure 4a. To illustrate this difference, we examined the attention map for Shanghai Pudong Development Bank during a significant event on April 18th, 2022, when certain banks in China failed to repay principal amounts to their customers. As shown in Figure 4b, despite the impact of this event, the attention heatmap from TSA predominantly concentrated on the current day. In contrast, TSIA effectively captured the event's influence across the following week, reflecting the lingering effect of past events on the stock's performance. This comparison highlighted the advantage of using TSIA, which enables the model to better capture long-term trends and temporal dependencies, thereby enhancing its ability to detect patterns from prior days.

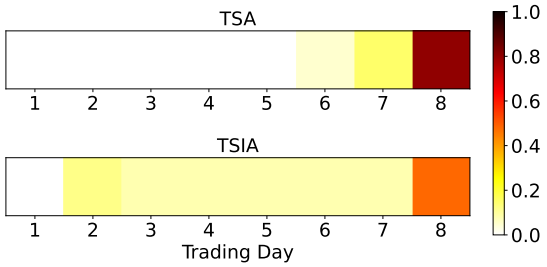
TABLE IV
PERFORMANCE OF CS AND GS ON CSI800

Module	Metrics			
	IC	ICIR	RankIC	RankICIR
CS	0.061 ± 0.005	0.56 ± 0.04	0.063 ± 0.006	0.55 ± 0.05
GS	**0.068* ± 0.005**	**0.68* ± 0.03**	**0.069* ± 0.007**	**0.66* ± 0.06**

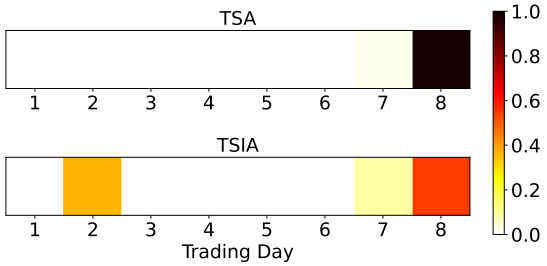
The best results for each comparison group are highlighted with **bold** and marked with * for statistical significance (by t-test at $p = 0.01$ level).

G. Effect of TCS

To further examine the effect of TCS, we compared CS, which refers to standard self-attention-based cross-stock aggregation, with GS, which incorporates the Gumbel-Softmax trick for stock selection during cross-stock aggregation. Both modules were combined with TSIA, and the results are presented in Table IV. As shown, adding CS did not improve accuracy on the CSI800 dataset and even degraded some metrics. This supports our hypothesis that aggregating too many stocks introduces noise, especially in large indices like CSI800 with many weakly correlated or irrelevant stocks. In contrast, GS significantly improved performance, demonstrating the effectiveness of using a trainable selection mechanism to retain only the most relevant stocks for aggregation.



(a)



(b)

Fig. 4. (a) Average attention weights from past trading days to the current day across the test dataset. (b) The attention from last trading day to the 8 days in the sequence using TSA and TSIA. This is the attention map of Shanghai Pudong Development Bank one week after the event on April 18, 2022.

VI. Conclusion

We propose a novel framework for stock price movement forecasting that jointly aggregates information across time, feature, and stock dimensions. By modeling temporal correlations, adaptively aggregating high-dimensional features, and learning to select relevant stocks, the method achieves consistent improvements over state-of-the-art approaches in prediction accuracy. Future work will explore scaling the framework to larger stock universes.

VII. Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 62576187.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[2] R. Sawhney, S. Agarwal, A. Wadhwa, T. Derr, and R. R. Shah, "Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, pp. 497–504, May 2021.

[3] H. Wang, T. Wang, S. Li, J. Zheng, S. Guan, and W. Chen, "Adaptive long-short pattern transformer for stock investment selection," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, L. D. Raedt, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2022, pp. 3970–3977, main Track.

[4] T. Li, Z. Liu, Y. Shen, X. Wang, H. Chen, and S. Huang, "Master: Market-guided stock transformer for stock price forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, pp. 162–170, Mar. 2024.

[5] J. H. Cochrane, "Presidential address: Discount rates," *The Journal of Finance*, vol. 66, no. 4, pp. 1047–1108, 2011.

[6] K. Hou, C. Xue, and L. Zhang, "Replicating Anomalies," *The Review of Financial Studies*, vol. 33, no. 5, pp. 2019–2133, 12 2018.

[7] S. Gu, B. Kelly, and D. Xiu, "Empirical Asset Pricing via Machine Learning," *The Review of Financial Studies*, vol. 33, no. 5, pp. 2223–2273, 02 2020.

[8] E. F. Fama and K. R. French, "The cross-section of expected stock returns," *The Journal of Finance*, vol. 47, no. 2, pp. 427–465, 1992.

[9] I. Welch and A. Goyal, "A comprehensive look at the empirical performance of equity premium prediction," *The Review of Financial Studies*, vol. 21, no. 4, pp. 1455–1508, 03 2007.

[10] E. F. Fama and K. R. French, "A five-factor asset pricing model," *Journal of Financial Economics*, vol. 116, no. 1, pp. 1–22, 2015.

[11] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *The Eleventh International Conference on Learning Representations*, 2023.

[12] H. Xia, H. Ao, L. Li, Y. Liu, S. Liu, G. Ye, and H. Chai, "Ci-sthpan: Pre-trained attention network for stock selection with channel-independent spatio-temporal hypergraph," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 8, pp. 9187–9195, Mar. 2024.

[13] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2017.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, nov 1997.

[15] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2627–2633.

[16] Y. Du, J. Wang, W. Feng, S. Pan, T. Qin, R. Xu, and C. Wang, "Adarnn: Adaptive learning and forecasting of time series," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ser. CIKM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 402–411.

[17] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 11 106–11 115, May 2021.

[18] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[19] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162, 17–23 Jul 2022, pp. 27 268–27 286.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[21] J. Yoo, Y. Soun, Y.-c. Park, and U. Kang, "Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2037–2045.

[22] K. Choi, G. Shin, J. Yang, and H. Kim, "Dycor: Capturing hidden stock relationships for stock trend prediction," in *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, ser. CIKM '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 458–467. [Online]. Available: https://doi.org/10.1145/3746252.3761413

[23] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, and T.-S. Chua, "Temporal relational ranking for stock prediction," *ACM Trans. Inf. Syst.*, vol. 37, no. 2, mar 2019.

[24] X. Yang, W. Liu, D. Zhou, J. Bian, and T.-Y. Liu, "Qlib: An ai-oriented quantitative investment platform," 2020.

[25] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 785–794.

[26] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.