StockMixer: A Simple yet Strong MLP-based Architecture for Stock Price Forecasting

Jinyong Fan[1], **Yanyan Shen**[1*]

[1] Shanghai Jiao Tong University
{weizhili, shenyy}@sjtu.edu.cn

Abstract

Stock price forecasting is a fundamental yet challenging task in quantitative investment. Various researchers have developed a combination of neural network models (e.g., RNNs, GNNs, Transformers) for capturing complex indicator, temporal and stock correlations of the stock data. While complex architectures are highly expressive, they are often difficult to optimize and the performances are often compromised by the limited stock data. In this paper, we propose a simple MLP-based architecture named StockMixer which is easy to optimize and enjoys strong predictive performance. StockMixer performs indicator mixing, followed by time mixing, and finally stock mixing. Unlike the standard MLP-based mixing, we devise the time mixing to exchange multi-scale time patch information and realize the stock mixing by exploiting stock-to-market and market-to-stock influences explicitly. Extensive experiments on real stock benchmarks demonstrate our proposed StockMixer outperforms various state-of-the-art forecasting methods with a notable margin while reducing memory usage and runtime cost. Code is available at https://github.com/SJTU-Quant/StockMixer.

Introduction

Stock price forecasting is a fundamental task in the field of quantitative investment. Due to the fact that the movements of different stock prices in a market are not independent with each other, stock price forecasting is practically formulated as a multivariate time series forecasting problem. As the stock market is highly volatile and chaotic, achieving high forecasting accuracy remains an open question.

Numerous efforts have been devoted to improving forecasting performance for profitable stock investment. Early attempts apply basic machine learning methods to uncover complex patterns in stock data, including decision trees (Nugroho, Adji, and Fauziati 2014; Kamble 2017), support vector machines (SVM) (Xie et al. 2013), and k-nearest neighbors (KNN) (Alkhatib et al. 2013). With the advent of deep learning, recent literature focuses on developing neural architectures that are expressive and flexible to exploit various inductive biases based on the intuitive (probably insightful) understandings about the stock market. Generally, there are three kinds of correlations that have been investigated.

- Indicator correlations. Typically, there are several financial indicators serving as raw features for each stock per trading day, e.g., the basic open, high, low, and closing prices. It is desirable to model correlations and dependencies among raw indicators and extract high-level latent features that are informative for future stock trends.
- Temporal correlations. Stock price movements are fundamentally caused by the continuous demand-and-supply balancing. The temporal trends in surrounding trading days are noticeably correlated, e.g., moving in the same or reverse directions. The existence of temporal correlations makes the future trends predictable.
- Stock correlations. As staying in the same market, stocks are correlated. For instance, stocks in the same industry may all have an upstream movement in one trading day due to a bullish event for the industry. Being aware of the stock correlations is thus beneficial for the forecasting.

Existing deep learning methods (Zou et al. 2022) take parts or all of the correlations into account. A typical model architecture consists of a specialized neural module for modeling each individual correlation, followed by a fusion module that combines information from preceding modules for the final prediction. Specifically, Recurrent Neural Networks (RNNs) (Qin et al. 2017; Nelson, Pereira, and De Oliveira 2017; Feng et al. 2018) are used for modeling temporal correlations; Graph Neural Networks (GNNs) (Feng et al. 2019; Li et al. 2021; Sawhney et al. 2021) are good at exchanging stock-wise information; Transformer-based models (Yoo et al. 2021; Wang et al. 2022; Li et al. 2023a) use the attention mechanism to emphasize crucial patterns from correlated subjects. Nevertheless, a hybrid neural architecture increases the model complexity and may further hurt the model's generalization ability. The reasons are three-fold. First, stock price data is of a limited size, i.e., about 250 trading days per year and each day delivers *only one* multivariate time series for training, introducing overfitting risks into the model. Second, a hybrid model involving diverse information exchanging scopes (e.g., local or global) and behaviors (e.g., using gating or attention) is not easy to optimize which may compromise the final performance. Third, some components may learn inaccurate inductive bias that misleads the model training. For instance, GNNs assume smoothness between related stocks but their underlying patterns can be

*corresponding author.

heterogeneous. To these ends, we are interested in *developing a simple neural architecture that is easy to optimize and enjoys strong predictive performance by modeling the above-mentioned correlations effectively*.

Recently, Multi-layer Perception (MLP) architectures have shown promising performance in computer vision tasks than state-of-the-art neural networks that use convolutions and attention mechanisms (Tolstikhin et al. 2021; Touvron et al. 2022; Yu et al. 2022). The architectural simplicity and linear computational efficiency of the MLP architecture inspire us to adapt it to the problem of stock price forecasting. A straightforward way is to perform MLP-based mixing three times for modeling indicator, temporal, and stock correlations, respectively. Specifically, indicator mixing uses matrix multiplication and activation functions to model interactions among indicators within each stock-time pair, while time and stock mixing are performed within the stock-indicator and indicator-time pairs accordingly.

However, the standard MLP-based mixing suffers from poor performance according to our experiments. Through deep analysis, we identify two key technical challenges.

First, due to the high dynamics of the stock market, the time correlations in surrounding trading days are not simply point-wise correlations. At one extreme, the closing prices of a stock within a time window like 5 days are constantly changing, similar to *iid* samples drawn from an underlying distribution. The time mixing that exchanges time-point information is thus insufficient for modeling temo correlations. *Second*, MLP-based mixing over stocks essentially performs information exchange among stocks based on the learned weight matrix. As pointed out in the previous works (Sawhney et al. 2021; Huynh et al. 2023), stock correlations are complex, and the direct stock-to-stock mixing may compromise the model performance. For instance, two stocks in the same industry may randomly have similar trends or divergent trends over time. To sum up, it is time to seek effective time mixing and stock mixing schemes that overcome the above two challenges.

In this paper, we propose a simple yet strong MLP-based architecture named StockMixer for stock price forecasting. Specifically, we design three mixing blocks for modeling the indicator, temporal, and stock correlations effectively. Our insights to tackle the two challenges are the following. For time mixing, the local temporal patterns on surrounding days are correlated to a certain extent. For example, the rising-up and falling-down variation tendencies are not independent and are driven by the latent stock value. Hence, we patch time steps at multiple scales and extract patch tendencies to be mixed. For stock mixing, we recognize that stock correlations are typically influenced by overall stock market conditions or states. For instance, in a bull market, stocks tend to become more correlated and move together as investors become more optimistic. We thus use two MLP structures to learn latent stock states from all stocks and use the states to influence individual stocks. This leads to a more robust modeling of stock correlations, from individual stocks to the whole market, and then back to the stocks. Based on the sophisticated designs for time mixing and stock mixing, our proposed StockMixer enjoys structural complexity as MLP-mixer and achieves more promising predictive performance than state-of-the-art methods.

To summarize, this paper has the following contributions:

- We propose a lightweight and effective MLP-based architecture for stock price forecasting. It consists of indicator mixing, time mixing and stock mixing to capture complex correlations in the stock data.
- We demonstrate the deficiencies of the standard MLP-based mixing. We introduce patch-based multi-scale time mixing and market-aware stock mixing that exploits the characteristics of stock patterns.
- We conduct extensive experiments on three real stock benchmarks NASDAQ, NYSE and S&P500. The experimental results show our proposed StockMixer outperforms state-of-the-art methods in terms of various evaluation metrics.

Related Work

In this section, we review the related work from the literature of stock price forecasting and MLP-based Architectures.

Stock Price Forecasting. Stock price forecasting has undergone a long period of development on top of price-volume indicators from historical data. At the very beginning, conventional mathematical algorithms only focus on numerical features (Piccolo 1990; Wang and Leu 1996; Tseng, Yu, and Tzeng 2002) based on financial technical analysis. With the advances of deep neural network (DNN), parts of the works following previous paradigm employ recurrent neural networks (Nelson, Pereira, and De Oliveira 2017; Qin et al. 2017) or convolutional neural networks (Tsantekidis et al. 2017) to model a single stock price and predict its short-term trend. To enhance the capability of handling fine-grained transition signals, some efforts have explored other techniques such as self-attention mechanism (Li, Shen, and Zhu 2018), adversarial training (Feng et al. 2018), and gated causal convolutions (Wang et al. 2021). Later studies achieve state-of-the-art performance considering the inter-stock relationships. For instance, RSR (Feng et al. 2019) proposes a temporal graph convolution model, which created the compositions of temporal encoder, relation embedding and a prediction layer. LSTM-RGCN (Li et al. 2021) handles both positive and negative correlations among stocks to alleviate the over-smoothing problem when predicting overnight stock price movements. STHAN-R (Sawhney et al. 2021) augments the corporate relevance based on Wiki data and uses hypergraph convolution to propagate higher-order neighbor's information. The latest method ESTIMATE (Huynh et al. 2023) also uses hypergraph to capture the non-pairwise correlations with temporal generative filters for individual patterns per stock.

MLP-based Architectures. Recently, MLP has been reinvestigated in the computer vision domain (Tolstikhin et al. 2021; Liu et al. 2021; Touvron et al. 2022). With linear computation complexity and simpler architectures, MLP-Mixer (Tolstikhin et al. 2021) attains similar performance

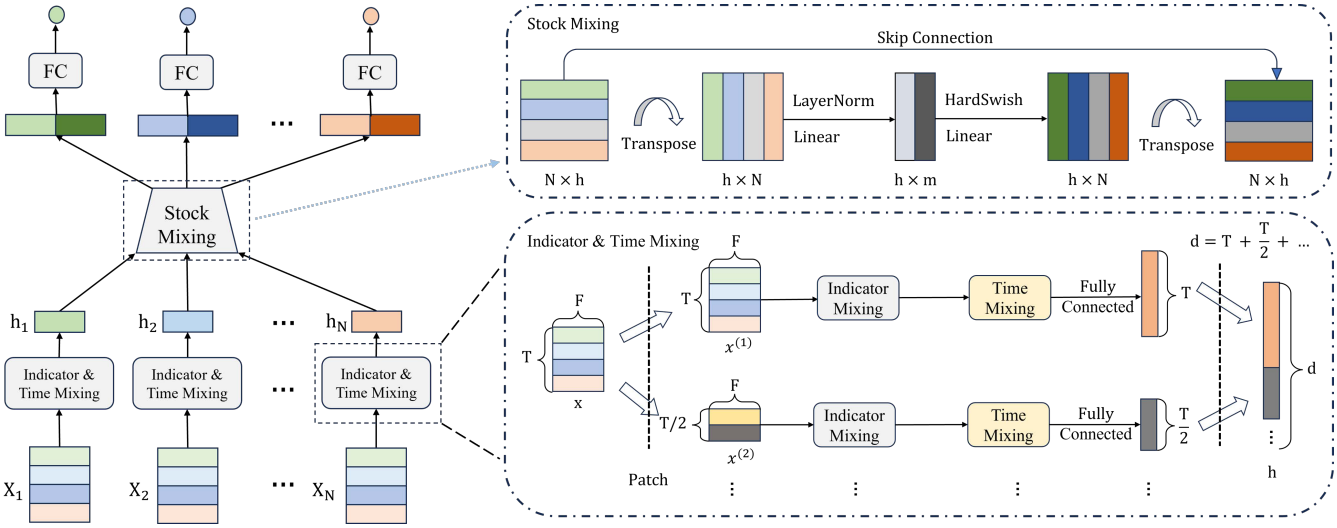


Figure 1: Overview of the proposed StockMixer.

as CNN and Transformer by operating patches and significantly enhancing the amount of inductive bias upon common MLPs. Similar efforts have recently been put in the time series domain. Dlinear (Zeng et al. 2023) and its follow-up works (Das et al. 2023) confirm the feasibility of this simple architecture in temporal prediction tasks. A series of works (Zhang et al. 2022; Li et al. 2023b; Ekambaram et al. 2023) utilize the MLP-Mixer backbone that significantly empowers the learning capability of simple MLP structures to improve the performance of time series forecasting. However, since stock data lacks periodicity and changes dynamically in temporal and stock correlations, the aforementioned MLP-based methods perform even worse than the basic models on stock datasets.

Methodology

Problem Definition

Following the setup of existing works (Feng et al. 2018; Huynh et al. 2023), we input the normalized stock historic patterns with multiple indicators (such as stock open price, close price or 5-day average close price) and output closing price of the next day to calculate the 1-day return ratio. The notations are as follows.

Given all data of the stock market composed of N stocks $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N\}$, each stock $\mathcal{X}_i \in \mathbb{R}^{T \times F}$ contains historical data with lookback window length T, where F denotes the indicator dimension at one time step. Our task is to predict closing price p_i^t on trading day t and calculate the 1-day return ratio $r_i^t = \frac{p_i^t - p_i^{t-1}}{p_i^{t-1}}$. Denoting our model parameters as θ, the process can be expressed as:

$$\mathcal{X} \in \mathbb{R}^{N \times T \times F} \xrightarrow{\theta} p \in \mathbb{R}^{N \times 1} \rightarrow r \in \mathbb{R}^{N \times 1}. \quad (1)$$

Standard MLP-based Mixing

As a lightweight method towards image classification, MLP-Mixer only relies on the linear layers repeatedly implemented in the token or feature channel, residual connection, data scale transformation (such as reshape, transposition) as well as appropriative activation function. Except for a significant enhancement for the computation speed, we value it more in exchanging information between various dimensions. This ability enables close communication between indicators, time, and stocks in the stock market, promoting the expressive power of the model. Residual connection keeps a trade-off between inputs and mixed feature while layer normalization eliminates the impact of data offset to a certain extend. For each original representation $x \in \mathbb{R}^{a \times b}$, we compute a new embedding $y \in \mathbb{R}^{a \times b}$ with mixed features on dimension a as:

$$y = x + \mathbf{W}_2 \sigma(\mathbf{W}_1 \text{LayerNorm}(x)), \quad (2)$$

where x is the inputs feature and y is the output of the block. $\mathbf{W}_1 \in \mathbb{R}^{h \times a}$, $\mathbf{W}_2 \in \mathbb{R}^{a \times h}$ are trainable weights of fully connected layers and h is a tunable hidden dimension always equal to a. σ denotes the non-linear activation function, which makes significant impact on predicted performance. Previous CV models chose GeLU (Howard et al. 2019) that performs better on images, and through experiments we find that ReLU and HardSwish (Avenash and Viswanath 2019) achieve superior performance on temporal data.

The StockMixer Approach

Figure 1 illustrates the overview of **StockMixer**, mainly including two parts: indicator&time mixing and stock mixing. The former extracts the respective representations of each stock, acting as an efficient encoder. The latter gathers the learned representations of all stocks in current market to capture the complex stock correlations. Finally, we combine these two representations to predict close price on trading day. Here, adhering to the arrangement of data dimension, we design the module sequence as indicator mixing, time mixing and stock mixing, deriving our StockMixer.

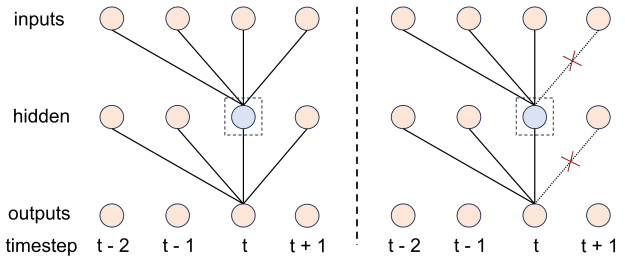


Figure 2: Standard mixing (left) vs time mixing (right).

Indicator Mixing. Historical stock price has shown to be a strong indicator of future stock trends and widely used across financial literature. Previous works fed the sequence of financial indicators into recurrent neural network and ignored the correlations between indicators. For instance, the difference between the open and closing prices of a stock on the same day may imply its future trend, so it is needed to exchange the information of indicators at each time step before calculating the temporal representations. Our indicator mixing is consistent with standard MLP-based mixing. For each stock, we transpose its time and indicator to perform feature mixing on the indicator dimension and formulate the indicator mixing as:

$$\hat{x}^T = x^T + \mathbf{W}_2\sigma(\mathbf{W}_1\text{LayerNorm}(x^T)), \tag{3}$$

where $x^T \in \mathbb{R}^{F\times T}$ denotes transposed original embedding of a single stock and $\hat{x} \in \mathbb{R}^{T\times F}$ is the result of indicator mixing and we take it as the inputs of the following time mixing.

Time Mixing. Unlike standard MLP-based mixing in computer vision emphasizing the equality of patches, information exchange in temporal domain relies more on chronological order. Specifically, information from earlier time steps can influence later time steps, but not vice versa. However, standard MLP-based mixing employs the fully connected structure to exchange information against the characteristic of temporal data. To address this issue, we propose a structural modification on the fully connected hidden layer resembling a self-attention mask in Figure 2. When communicating the temporal information, any t could only see itself and its previous time step's content instead of adopting all equally. This modification ensures that information from later time steps does not leak into the earlier ones, more in line with temporal nature. Replacing the weights with upper triangular matrix realises the process:

$$h = \hat{x} + \mathbf{U}_2\sigma(\mathbf{U}_1\text{LayerNorm}(\hat{x})), \tag{4}$$

where $x \in \mathbb{R}^{T\times F}$ denotes the indicator mixing representation and $U_1 \in \mathbb{R}^{H_t\times T}, U_2 \in \mathbb{R}^{T\times H_t}$ respectively signify the learnable weights of the first and second fully connected layer, that only the upper triangular part of a matrix is trainable to achieve the effect of mask. H_t means the hidden dimension of time and here we set $H_t = T$ as customary.

Although there exist researches (Zeng et al. 2023; Li et al. 2023b; Ekambaram et al. 2023) deploying MLP into Long Time Series Forecasting (LTSF), they rely heavily on stable and periodic datasets (e.x. electricity and transportation), which are easier to learn stable and sufficient representations. Due to the timeliness of the shares market, only recent lookback window is effective for price prediction. It is necessary to utilize the patterns more fully and overcome the sensitivity of linear models to small fluctuations caused by the absence of periodicity. In order to mine information from short sequences as much as possible while enhancing the robustness of the linear layer against time deviation, we segment original time sequence into subsequence-level patches and mix features at different scales. Such segmentation causes dimension extension adverse to mixture, so we map the representations of all time steps in a patch into one overall look. Specifically, we obtain the corresponding single pattern by avgpool or one-dimensional convolution from raw inputs of a stock $x \in \mathbb{R}^{T\times F}$ as:

$$x^{(k)} = \text{Avgpool}(x)_{\text{kernel}=k}, k \in \{\frac{T}{2}, \frac{T}{4}, \ldots, 1\}, \tag{5}$$

and $x^{(k)} \in \mathbb{R}^{\frac{T}{k}\times F}$ represents the compressed sequence when the patch size is k. Then we send the $x^{(k)}$ through indicator mixing and time mixing, obtain its mixing embedding $h^{(k)} \in \mathbb{R}^{\frac{T}{k}\times F}$. After that, we assign a fully connected layer after a concatenation operation to all patch size to aggregate the final temporal representation h. The process is:

$$h^{(k)} = \text{TimeMixing}(\text{IndicatorMixing}(x^{(k)})), \tag{6}$$

$$h = \text{FC}(\text{concat}(h^{(k)})), k \in \{\frac{T}{2}, \frac{T}{4}, \ldots, 1\}. \tag{7}$$

Concretely, in this work, if the length of the input series is 16, we set $k \in \{1,\ 2,\ 4\}$ and thus $h \in \mathbb{R}^{(T+\frac{T}{2}+\frac{T}{4})}$. For the convenience of subsequent narration, we denote the embedding dimension $d = (T + \frac{T}{2} + \frac{T}{4})$. By combining information at different scales, the model can obtain multi-level, rich and diverse feature representations on limited series, and improve its generalization ability on unprecedented data.

Stock Mixing. Based on the previous operations, we obtain the temporal representations of all stocks $H = \{h_1,\ h_2,\ \ldots,\ h_N\} \in \mathbb{R}^{N\times d}$. Next, we describe our Stock Mixing to construct inter stock relation without any prior knowledge like knowledge graphs or industry information. Obviously, it is conceivable that the strong information exchange capability of MLP-Mixer could be applied for relation capture. By setting the hidden dimension of standard mixing $a = N$ in Equation 2, the interaction process can be understood as N market characteristics aggregated by all stocks, in turn, affect these N stocks. This modeling method is no different from message-passing on a fully connected graph which involves an edge between any two stocks. In that case, some insignificant relationships or coincidences will also be considered and the mode of message passing we capture is extremely vulnerable. Due to the small sizes of the stock datasets, the absence of transferability and robustness causes serious overfitting problem.

Drawing insights from that, we hope to preserve the most important and informative market states to improve the performance and interpretability of the model. For one stock,

we do not have to take all other stocks' information into consideration, but decompose the process of direct information exchange among stocks into stock-to-market and market-to-stock similar to hypergraph. To achieve this, we replace the hidden dimension of standard mixing related to stocks with a hyperparameter m to reach the effect of self-learnable hypergraph. The stock mixing can be formulated as:

$$\hat{H} = H + \mathbf{M}_2\sigma(\mathbf{M}_1\text{LayerNorm}(H)), \tag{8}$$

where $\mathbf{M}_1 \in \mathbb{R}^{m\times N}$ compresses N representations into m while the $\mathbf{M}_2 \in \mathbb{R}^{N\times m}$ restores the zoomed information. This process is similar to the process in which node information in a hypergraph is first aggregated onto hyperedges and then the impact of hyperedges on each entity is calculated, except that this process is induced by the model itself. $H \in \mathbb{R}^{N\times h}$ represents the individual embeddings of N stocks in the market and $\hat{H}$ denotes the influence on each stock from extracted relationships.

Finally, we compute the concatenation between stock's own representation H and its corresponding market-influenced representation $\hat{H}$ and then send it to a fully connected layer for dimension reduction to obtain the final prediction.

Loss Function

We use the 1-day return ratio of a stock as the ground-truth rather than the normalized price used in previous work using a combination of a pointwise regression and pairwise ranking-aware loss to minimize the MSE between the predicted and actual return ratios while maintaining the relative order of top ranked stocks with higher expected return for investment as:

$$L = L_{\text{MSE}} + \alpha \sum_{i=1}^{N}\sum_{j=1}^{N} \max(0, -(\hat{r}_i^t - \hat{r}_j^t)(r_i^t - r_j^t)), \tag{9}$$

where $\hat{r}^t$ and r^t are the predicted and actual ranking scores, respectively, and α is a weighting parameter.

Experiments

Experimental Setup

Datasets. We evaluate our approach based on three real-world datasets from the US stock market. The statistics of the datasets are in Table 1. These datasets all contain relatively complete sector-industry relations or Wiki company-based relations, making it easy to compare with other graph-based methods. *NASDAQ* and *NYSE* (Feng et al. 2019) filter transaction records between 01/02/2013 and 12/08/2017 from corresponding markets. Datasets removed abnormal patterns and penny stocks while maintain their representative properties that NASDAQ is more volatile whereas NYSE is more stable. *S&P500* (Huynh et al. 2023) gathers historic price data and the information about industries in the S&P 500 index from the Yahoo Finance database.

	NASDAQ	**NYSE**	**S&P500**
# Stocks	1026	1737	474
Start Time	13-01-02	13-01-02	16-01-04
End Time	17-12-08	17-12-08	22-05-25
Train Days	756	756	1006
Val Days	252	252	253
Test Days	273	273	352

Table 1: Statistics of datasets.

Implementation Details. Our model is implemented with PyTorch. For fair comparison, all samples are generated by moving a 16-day lookback window along trading days. Regarding temporal scale factors, $k \in \{1,\ 2,\ 4\}$ is set for all datasets and only 1 Stock Mixing is employed in the model. We use grid search to find optimal market hyperparameters m, and finalize $m\ =\ 20,\ 25,\ 8$ for **NASDAQ**, **NYSE** and **S&P500**, respectively. For methods that require market information, we construct graphs or hypergraphs according to the preprocessing process in the original paper. The loss factor α is 0.1 and the learning rate is $1e-3$. We conducted all the experiments on a server equipped with Intel(R) Xeon(R) Silver 4110 CPU, 128GB Memory, and a Nvidia GeForce RTX 2080 Ti GPU (12GB Memory). Each experiment was repeated 3 times and the average performance was reported.

Metrics. Previous studies applied distinct pointers, making it troublesome for comprehensive comparison of various methods. To thoroughly evaluate the performance of the techniques, we employ four most frequently used and most stable metrics, among which are two rank-based evaluation metrics, one accuracy-based and the other return-based. *Information Coefficient* (IC) is a coefficient that shows how close the prediction is to the actual result, computed by the average Pearson correlation coefficient. *Rank Information Coefficient* (RIC) is the coefficient based on the ranking of the stocks' short-term profit potential, computed by the average Spearman coefficient. The above two metrics evaluate stock selection ability of model and they are strongly related to rank loss. *Precision@*N evaluates the precision of the top N predictions. For example, when N is 10, and the labels of 4 among these top 10 predictions are positive, then the Precision@10 is 40%. *Sharpe Ratio* (SR) takes into account both return and risk and calculates the average return per unit of volatility in relation to the risk-free rate: $SR = \frac{R_t - R_f}{\theta}$, where R_t represents the return, R_f represents the risk-free rate, and θ represents the standard deviation of the returns.

Baselines. We compare the performance of our architecture with that of several state-of-the-art baselines, as follows: (1) **LSTM** (Hochreiter and Schmidhuber 1997) applies vanilla LSTM on temporal price data for ranking. (2) **ALSTM** (Feng et al. 2018) integrates the adversarial training and stochasticity simulation in an enhanced LSTM to better learn the market dynamics. (3) **RGCN** (Li et al. 2021) adopts Relational Graph Convolutional Networks (RGCN) to model multi-relations. (4) **GAT** (Veličković et al. 2017) utilizes graph attention networks (GAT) to aggregate

	Model	NASDAQ				NYSE				S&P500			
		IC	RIC	prec@N	SR	IC	RIC	prec@N	SR	IC	RIC	prec@N	SR
RNN	**LSTM**	0.032	0.354	0.514	0.892	0.024	0.256	0.512	0.857	0.031	0.186	0.531	1.332
	ALSTM	0.035	0.371	0.522	0.941	0.023	0.276	0.519	0.764	0.029	0.181	0.532	1.298
GNN	**RGCN**	0.034	0.382	0.516	1.054	0.025	0.275	0.517	0.932	0.028	0.175	0.528	1.359
	GAT	0.035	0.377	0.530	1.233	0.025	0.297	0.521	1.070	0.034	0.191	0.541	1.484
	RSR-I	0.038	0.398	0.531	1.238	0.026	0.284	0.519	0.098	0.033	0.200	0.542	1.437
HGNN	**STHAN-SR**	0.039	0.451	0.543	1.416	0.029	0.344	**0.542**	1.228	0.037	0.227	0.549	1.533
	ESTIMATE	0.037	0.444	0.539	1.307	**0.030**	0.327	0.536	1.115	0.035	0.241	**0.553**	1.547
MLP	**Linear**	0.019	0.188	0.505	0.517	0.015	0.163	0.497	0.625	0.016	0.156	0.520	0.674
	StockMixer	**0.043**	**0.501**	**0.545**	**1.465**	0.029	**0.351**	0.539	**1.454**	**0.041**	**0.262**	0.551	**1.586**

Table 2: Comparison results on stock metrics (measured by t-test with p-value < 0.01). The methods for comparison are mainly divided into four types: RNN (Recurrent Neural Network), GNN (Graph Neural Network), HGNN (HyperGraph Neural Network) and MLP (Multi-Layer Perceptron). Bold & underlines show best & second best (SOTA) results, respectively.

stock embeddings encoded by GRU on the stock graph. (5) **RSR** (Feng et al. 2019) combines Temporal Graph Convolution with LSTM to learn the stocks' interaction in a time-sensitive manner. The original proposes two ways, RSR-E using similarity as relation weight as well as RSR-I with neural net for relation weight, and we choose RSR-I with better performance as the baseline. (6) **STHAN-SR** (Sawhney et al. 2021) models the relations with hypergraph attention combined temporal Hawkes attentive LSTM to tailor spatiotemporal network architecture to rank stocks. (7) **ESTIMATE** (Huynh et al. 2023) implements a memory-based mechanism onto an LSTM network in an attempt to learn individual patterns and employs hypergraph attentions to capture the non-pairwise correlations, which pass message by the wavelet basis instead of the Fourier basis. (8) **Linear** only uses simple fully connected layers to predict the final price.

Overall Comparison

Table 2 shows the performance of all the comparison methods. Most of the baselines' results on the benchmarks are reported using their original settings and all of them adopt the same optimization loss in ensuring fair. We have the following key observations: 1) For univariate methods, whether LSTM or enhanced ALSTM performs worse than all other hybrid architectures, which proves the necessity and effectiveness of relationships in the stock market. As it should be, RNN-based encoders are faster the rest due to no additional calculations on relations and perform better with small capacity of stocks(e.g., S&P500) because of the fewer relationships in the market. 2) Hypergraph architectures appreciably have better capability of modeling complicated inter-stock dependencies, since classic graph tends to define pairwise correlations between any two entities. However, tendency of real share prices do not depend on several strongly correlated enterprise but current market attributes. Thus, to some extent, hyperedges gathering the industry information reflect parts of these market attributes. 3) Simple linear model lacks adequate inductive bias and naturally fail while other MLP-based methods for time series perform even worse without considering the characteristics of stock data. Relatively short lookback window overlooks periodicity and time deviation leads to market value varying, which are the main cause of severe overfitting. 4) Balancing the lightweight of MLP models with the excellent performance of hybrid networks, our proposed StockMixer obtains the best results across most metrics and fetches an average relative performance gain of 7.6%, 10.8% and 10.9% in regard of two rank metrics and the risk adjusted returns ($p < 0.01$). Meanwhile, simple but strong design brings parameter quantity second only to RNN and much less computation time compared with graph message passing. In addition, slight performance degradation is observed on NYSE with most stocks (1737), which may indicate that insufficient inductive bias gradually come into force in dealing with larger candidate pools.

Ablation	NASDAQ		NYSE	
Model Component	IC	RIC	IC	RIC
LSTM	0.032	0.354	0.024	0.256
w.o.Indicator Mixing	0.040	0.465	0.027	0.291
w.o.Time Mixing	0.018	0.164	0.016	0.161
w.o.Stock Mixing	0.037	0.376	0.026	0.285
LSTM + Stock Mixing	0.041	0.476	**0.030**	0.307
STHAN-SR	0.039	0.451	0.029	0.344
StockMixer	**0.043**	**0.501**	0.029	**0.351**

Table 3: Ablation study over three components (indicator mixing, time mixing, and stock mixing) on NASDAQ and NYSE.

Ablation Study

Model Component. We attempt to verify the effect of three mixing blocks by removing the one of them respectively and compare with two typical baselines, STHAN-SR and LSTM. We also replace the previous market module with our stock mixing and implement these settlements in NASDAQ and NYSE. The results are shown in Table 3. As shown, different components jointly contribute to the performance. Among three parts, the mixing of time dimension

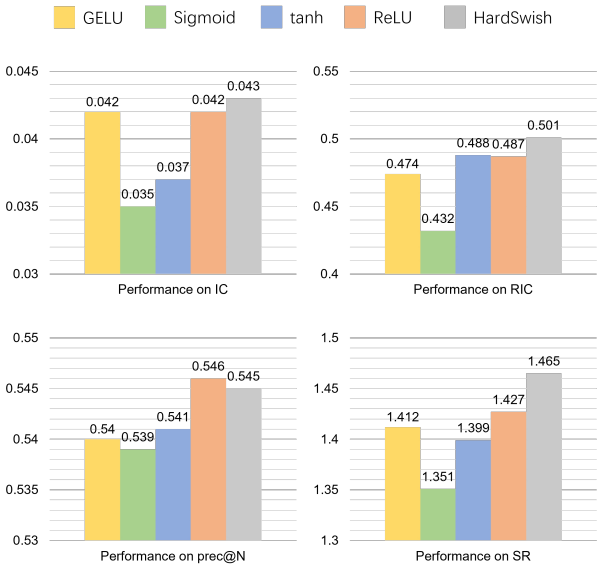


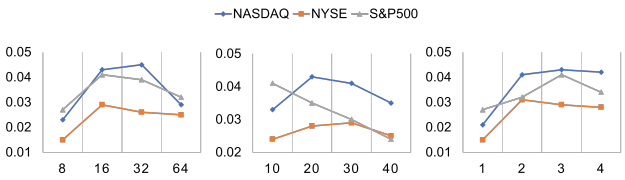
Figure 3: Effects of different activation function on NASDAQ.

matters most, for the poor learning of isolated representation can lead to meaningless relationship modeling. This also explains why StockMixer as well as earlier architecture adopts a framework of time before space. Without incorporating indicator features, MLP-Mixer has slightly worse performance, which confirmed the importance of mixing indicator features into stock movement. From the model performance of ablation experiments, the order of impact on the model is time, stock and indicator. The variant replacing the indicator mixing with vanilla LSTM can draw with the state-of-the-art STHAN-SR that integrates the stock mixing without prior knowledge is quite competent at shares market relationship. We can see that the MLP-based encoder credible alternative to RNN and brings a higher performance gain to LSTM. The most likely reason is that RNN makes the hidden representation lack cross-indicator correlation.

Activation Function. We investigate the impact of different activation functions on model performance. Due to space limit, we depict the results on NASDAQ in Figure 3, similar regularities can be observed on other datasets. In the original module, the GELU function with excellent performance in multiple computer vision, natural language processing, and voice tasks did not achieve the best performance in such sequence prediction tasks as stocks. It can be seen that both Sigmoid and tanh perform mediocrely while ReLU and HardSwish obviously improve model performance over all metrics. This ablation verified the impact of non-linear function on mixing block, and we also conduct the similar experiments to explore the effect of Layer Normalization, where is no significant difference.

Hyperparameter Sensitivity

The results in Figure 4 show the hyperparameter sensitivity. Due to the space limit, we focus on the most important



(a) Window length (b) Market (c) Scale factor

Figure 4: Sensitivity to parameters T, m and k.

hyperparameters and select IC as metric.

Lookback window length T. We analyze the prediction performance of StockMixer when varying the length T of the lookback window in Figure4a. Across all datasets, moderate window length gains the best performance. Too short window length drops quickly due to the lack of information while the overlong sequential patterns also fails as the lack of early information gain and increased learning costs for stocks.

Market dimension m. We consider different m of the hidden dimension of stock mixing in Figure 4b and observe that datasets achieve their best performance at distinct m. As shown, result on S&P500 degrades significantly when w exceeds 10 while that on NYSE does well at around 30. Markets with high capacity prefer larger m that the surge in stock numbers brings more complex market representations

Multi-Scale factor k. We analyze the variance in the profitability depending on the number of scale factors from the ranking in Figure 4c. It is seen that StockMixer performs generally well, while the best results are obtained for $k = 3$.

Conclusion

In this paper, we proposed StockMixer, a simple yet strong architecture with enhanced MLP blocks for stock price forecasting. Instead of using different sub-networks to model indicator, temporal and stock correlations, StockMixer consists of a lightweight combination of indicator, time, and stock mixing blocks. Especially, time mixing takes more scales into consideration which construct preferable temporal encoder and provides improvements for temporal data. In market view, stock mixing decomposes the standard mixing block to into information exchange of stock-to-market and market-to-stock, which is a more robust modeling of stock correlations. Through extensive experiments, we show that StockMixer outperforms all popular benchmarks with an average relative performance gain of 7.6%, 10.8% and 10.9% in regard of three metrics, validating that this architecture offers a powerful alternative to other current methods. In future, we aim to optimize the hyperparameter selection process and adapt StockMixer to more stock markets.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (2022YFE0200500), Shanghai Municipal Science and Technology Major Project

(2021SHZDZX0102), and SJTU Global Strategic Partnership Fund (2021SJTU-HKUST).

References

Alkhatib, K.; Najadat, H.; Hmeidi, I.; and Shatnawi, M. K. A. 2013. Stock price prediction using k-nearest neighbor (kNN) algorithm. *International Journal of Business, Humanities and Technology*, 3(3): 32–44.

Avenash, R.; and Viswanath, P. 2019. Semantic Segmentation of Satellite Images using a Modified CNN with Hard-Swish Activation Function. In *VISIGRAPP (4: VISAPP)*, 413–420.

Das, A.; Kong, W.; Leach, A.; Sen, R.; and Yu, R. 2023. Long-term Forecasting with TiDE: Time-series Dense Encoder. *arXiv preprint arXiv:2304.08424*.

Ekambaram, V.; Jati, A.; Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting. *arXiv preprint arXiv:2306.09364*.

Feng, F.; Chen, H.; He, X.; Ding, J.; Sun, M.; and Chua, T.-S. 2018. Enhancing stock movement prediction with adversarial training. *arXiv preprint arXiv:1810.09936*.

Feng, F.; He, X.; Wang, X.; Luo, C.; Liu, Y.; and Chua, T.-S. 2019. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, 37(2): 1–30.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1314–1324.

Huynh, T. T.; Nguyen, M. H.; Nguyen, T. T.; Nguyen, P. L.; Weidlich, M.; Nguyen, Q. V. H.; and Aberer, K. 2023. Efficient integration of multi-order dynamics and internal dynamics in stock movement prediction. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 850–858.

Kamble, R. A. 2017. Short and long term stock trend prediction using decision tree. In *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, 1371–1375. IEEE.

Li, H.; Shen, Y.; and Zhu, Y. 2018. Stock price prediction using attention-based multi-input LSTM. In *Asian conference on machine learning*, 454–469. PMLR.

Li, L.; Duan, L.; Wang, J.; He, C.; Chen, Z.; Xie, G.; Deng, S.; and Luo, Z. 2023a. Memory-Enhanced Transformer for Representation Learning on Temporal Heterogeneous Graphs. *Data Science and Engineering*, 8(2): 98–111.

Li, W.; Bao, R.; Harimoto, K.; Chen, D.; Xu, J.; and Su, Q. 2021. Modeling the stock relation with graph network for overnight stock movement prediction. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, 4541–4547.

Li, Z.; Rao, Z.; Pan, L.; and Xu, Z. 2023b. Mts-mixers: Multivariate time series forecasting via factorized temporal and channel mixing. *arXiv preprint arXiv:2302.04501*.

Liu, H.; Dai, Z.; So, D.; and Le, Q. V. 2021. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34: 9204–9215.

Nelson, D. M.; Pereira, A. C.; and De Oliveira, R. A. 2017. Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)*, 1419–1426. Ieee.

Nugroho, F. S. D.; Adji, T. B.; and Fauziati, S. 2014. Decision support system for stock trading using multiple indicators decision tree. In *2014 The 1st International Conference on Information Technology, Computer, and Electrical Engineering*, 291–296. IEEE.

Piccolo, D. 1990. A distance measure for classifying ARIMA models. *Journal of time series analysis*, 11(2): 153–164.

Qin, Y.; Song, D.; Chen, H.; Cheng, W.; Jiang, G.; and Cottrell, G. 2017. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*.

Sawhney, R.; Agarwal, S.; Wadhwa, A.; Derr, T.; and Shah, R. R. 2021. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 497–504.

Tolstikhin, I. O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. 2021. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34: 24261–24272.

Touvron, H.; Bojanowski, P.; Caron, M.; Cord, M.; El-Nouby, A.; Grave, E.; Izacard, G.; Joulin, A.; Synnaeve, G.; Verbeek, J.; et al. 2022. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4): 5314–5321.

Tsantekidis, A.; Passalis, N.; Tefas, A.; Kanniainen, J.; Gabbouj, M.; and Iosifidis, A. 2017. Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th conference on business informatics (CBI)*, volume 1, 7–12. IEEE.

Tseng, F.-M.; Yu, H.-C.; and Tzeng, G.-H. 2002. Combining neural network model with seasonal time series ARIMA model. *Technological forecasting and social change*, 69(1): 71–87.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, H.; Li, S.; Wang, T.; and Zheng, J. 2021. Hierarchical Adaptive Temporal-Relational Modeling for Stock Trend Prediction. In *IJCAI*, 3691–3698.

Wang, H.; Wang, T.; Li, S.; Zheng, J.; Guan, S.; and Chen, W. 2022. Adaptive long-short pattern transformer for stock investment selection. In *Proceedings of the Thirty-First*

International Joint Conference on Artificial Intelligence, 3970–3977.

Wang, J.-H.; and Leu, J.-Y. 1996. Stock market trend prediction using ARIMA-based neural networks. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 4, 2160–2165. IEEE.

Xie, B.; Passonneau, R.; Wu, L.; and Creamer, G. G. 2013. Semantic frames to predict stock price movement. In *Proceedings of the 51st annual meeting of the association for computational linguistics*, 873–883.

Yoo, J.; Soun, Y.; Park, Y.-c.; and Kang, U. 2021. 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*, 2037–2045.

Yu, T.; Li, X.; Cai, Y.; Sun, M.; and Li, P. 2022. S2-mlp: Spatial-shift mlp architecture for vision. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 297–306.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 11121–11128.

Zhang, T.; Zhang, Y.; Cao, W.; Bian, J.; Yi, X.; Zheng, S.; and Li, J. 2022. Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. *arXiv preprint arXiv:2207.01186*.

Zou, J.; Zhao, Q.; Jiao, Y.; Cao, H.; Liu, Y.; Yan, Q.; Abbasnejad, E.; Liu, L.; and Shi, J. Q. 2022. Stock Market Prediction via Deep Learning Techniques: A Survey. *arXiv preprint arXiv:2212.12717*.