CI-STHPAN: Pre-trained Attention Network for Stock Selection with Channel-Independent Spatio-Temporal Hypergraph

Hongjie Xia, Huijie Ao, Long Li, Yu Liu, Sen Liu, Guangnan Ye*, Hongfeng Chai

School of Computer Science, Fudan University, Shanghai, China
Institute of FinTech, Fudan University, Shanghai, China
hjxia22@m.fudan.edu.cn, yegn@fudan.edu.cn

Abstract

Quantitative stock selection is one of the most challenging FinTech tasks due to the non-stationary dynamics and complex market dependencies. Existing studies rely on channel mixing methods, exacerbating the issue of distribution shift in financial time series. Additionally, complex model structures they build make it difficult to handle very long sequences. Furthermore, most of them are based on predefined stock relationships thus making it difficult to capture the dynamic and highly volatile stock markets. To address the above issues, in this paper, we propose **C**hannel-**I**ndependent based **S**patio-**T**emporal **H**ypergraph **P**re-trained **A**ttention **N**etworks (CI-STHPAN), *a two-stage framework for stock selection, involving Transformer and HGAT based stock time series self-supervised pre-training and stock-ranking based downstream task fine-tuning*. We calculate the similarity of stock time series of different channel in dynamic intervals based on Dynamic Time Warping (DTW), and further construct channel-independent stock dynamic hypergraph based on the similarity. Experiments with NASDAQ and NYSE markets data over five years show that our framework outperforms SOTA approaches in terms of investment return ratio (IRR) and Sharpe ratio (SR). Additionally, we find that even without introducing graph information, self-supervised learning based on the vanilla Transformer Encoder also surpasses SOTA results. Notable improvements are gained on the NYSE market. It is mainly attributed to the improvement of fine-tuning approach on Information Coefficient (IC) and Information Ratio based IC (ICIR), indicating that the fine-tuning method enhances the accuracy and stability of the model prediction.

Introduction

Stock markets play a crucial role in shaping the development of the global economy (Zou et al. 2022), with global market capitalization returning to $100 trillion by June 2023 and investors re-embracing risky assets after the COVID-19. Stock markets are characterized by uncertainty and volatility, and traditional machine learning techniques have been applied to stock price prediction to overcome these difficulties while improving prediction accuracy. However, traditional machine learning requires the manual construction of a large number of features with financial and economic implications, which requires a high level of expertise. Moreover, traditional machine learning techniques have difficulty capturing the interactions among stocks.

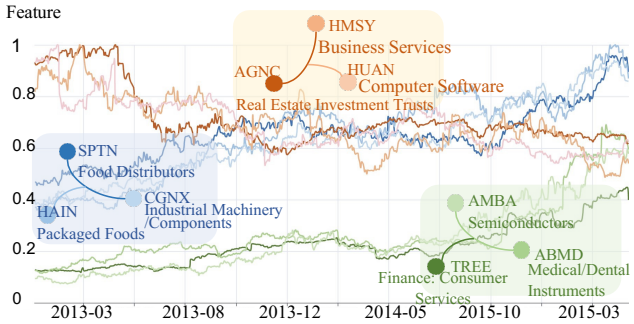


Figure 1: Correlation of stock movements among different sector-industries. Dynamic hyperedges are constructed based on similarity between the normalized time series.

In recent years, deep learning techniques have achieved great success in areas such as natural language processing(NLP), computer vision (CV) and automatic speech recognition (ASR), especially with the release of ChatGPT[1] at the end of 2022, which ushered in the era of large models. Accordingly, researchers in the quantitative investment field also actively tried to use deep learning for stock price prediction to enhance investment returns. Many studies have showed that deep learning-based stock price prediction models outperform traditional machine learning (Sezer, Gudelek, and Ozbayoglu 2020). Examples included LSTM (Feng et al. 2018), GRU (Xu and Cohen 2018) and Transformer (Muhammad et al. 2023), which were based on recurrent neural networks to improve the accuracy. In addition, many studies focused on correlations among stocks, such as holding relationships (Chen, Wei, and Huang 2018), sector relationships (Feng et al. 2019) and momentum spillover effects (Cheng and Li 2021).They applied various graph neural networks (Zhang, Cui, and Zhu 2020) such as GCN (Chen, Wei, and Huang 2018), GAT (Hsu, Tsai, and Li 2021) and HGCN (Sawhney et al. 2020) to fuse stock temporal features and spatial features to improve stock prediction accuracy.

However, all of those methods were end-to-end models.

*Corresponding Author.

[1]OpenAI (2023): https://openai.com/research/gpt-4

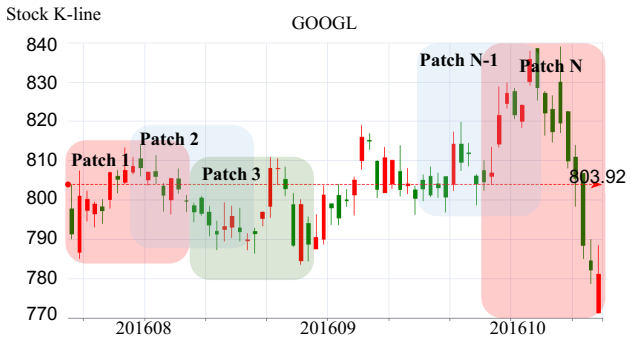


Figure 2: Stock time series patching. Patches are generated in the form of sliding windows over the entire stock time series data. Each patch contains a fixed length P and the interval S between each patch can be set as needed.

For instance, the stock price time series were first extracted by CNN or LSTM for temporal features, and then the spatial features were fused by GNN as node features in the stock relationship graph, and finally the prediction was performed by linear layer, which was regarded as a time series regression problem (Saha, Gao, and Gerlach 2022). This end-to-end approach had complex time complexity and did not take full advantage of the powerful feature representation capability of deep neural networks. Moreoever, most of the above graph-based approaches were based on predefined static relationships in the financial domain, whereas the stock market is highly volatile. In addition, the predefined relationships also greatly limited the types of relationships between stocks where there were also fairly consistent movements between stocks in different industries (as shown in Figure 1).

In this paper, we propose CI-STHPAN for stock selection. To the best of our knowledge, it is the first work to pre-train on stock time-series and fine-tune on a quantitative stock selection downstream task. In addition, we go beyond the construction of predefined stock financial relationships to construct hypergraphs based on long-term stock time series similarities to explore more potential stock relationships. Our main contributions are summarized as follows:

- We propose a two-stage framework for stock selection: Transformer and HGAT based stock time series self-supervised pre-training and stock-ranking based downstream task fine-tuning, which takes full advantage of the powerful spatio-temporal feature representation capability of deep neural networks.
- We apply a range of financial multivariate time series processing techniques, including Patching which allows the model to have a longer lookback window, channel independence and reversible instance normalization (RevIN) which allows the model to overcome the distribution shift properties of financial time series.
- We have conducted extensive experiments on both NASDAQ and NYSE stock markets to validate the superior performance of CI-STHPAN for quantitative stock selection.

Related Work

Transformer Based Time Series Forecasting

In recent years, Transformer-based deep learning models have achieved great success in NLP (Kalyan, Rajasekharan, and Sangeetha 2021), CV (Khan et al. 2022), ASR (Karita et al. 2019) and other fields. The core idea was to use the attention mechanism to learn the association of elements in a sequence autonomously, and thus many researchers applied it to time series forecasting, such as LogTrans (Li et al. 2019), Informer (Zhou et al. 2021), Autoformer (Wu et al. 2021), FEDformer (Zhou et al. 2022), Pyraformer (Liu et al. 2021), Non-stationary Transformers (Liu et al. 2022) and so on[2]. Inspired by the success obtained by pre-train techniques in NLP, such as BERT and GPT, many studies begun to explore pre-train on time-series data. (Zerveas et al. 2021) first proposed a Transformer-based temporal self-supervised learning framework and experimentally verified that self-supervised learning outperformed supervised learning. (Nie et al. 2022) further improved it while responding to (Zeng et al. 2023) question on Transformer-based time series forecasting models by proposing PatchTST, a patch time series prediction Transformer based on channel independent, which enhanced the time series representation learning ability while reducing the training time, and achieved SOTA results in many time series prediction tasks.

Spatio-temporal Based Stock Selection

Stock price movements are influenced by both historical trends and the movements of stocks associated with them. By such observations, ST-GNN[3] integrated both temporal and spatial features in the field of quantitative stock selection, and thus received much research attention in recent years. However, most of them constructed relationships among stocks based on predefined relationships such as stockholders (Chen, Wei, and Huang 2018), industry-sector (Feng et al. 2019; Kim et al. 2019; Sawhney et al. 2020), and concepts (Xu et al. 2021b). The predefined relationships greatly limited the types of relationships among stocks markets which are highly volatile, as shown in Figure 1. Therefore, many studies in recent years started to construct dynamic hidden stock relationships to adapt to highly volatile market, such as hidden conceptual relationships (Xu et al. 2021a), potential relationships among stocks and sectors (Hsu, Tsai, and Li 2021), similarity based dynamic hidden relationships (Wang et al. 2022), relevance implicit hypergraph relations (Huynh et al. 2023), etc.

Design of CI-STHPAN

Problem Formulation

In order to perform stock selection and maximize investment returns, following the work (Feng et al. 2019; Sawhney et al. 2021; Wang et al. 2022), we formulate stock price forecasting as a stock ranking problem. Given a list of stocks $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$, at each trading day t for each stock $s_i \in$

[2]More technical details about the Transformer-based time series forecasting models can be referred to (Zeng et al. 2023).

[3]ST-GNN:Spatio-Temporal Graph Neural Networks

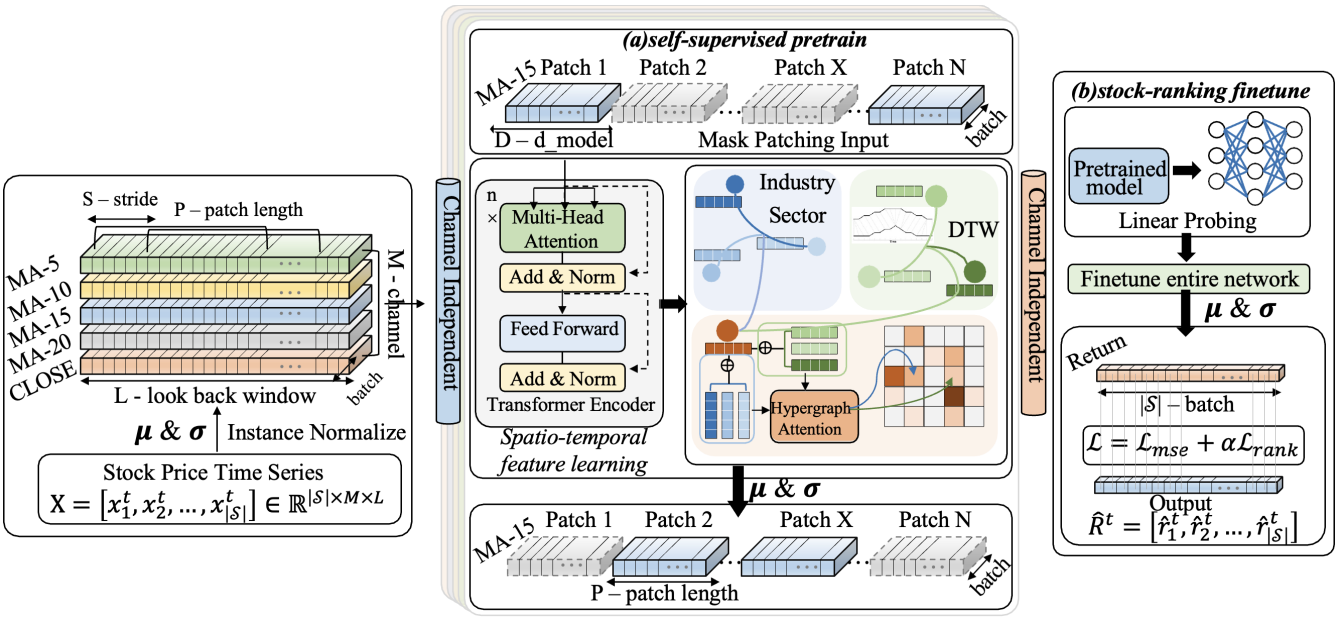


Figure 3: Overview of the proposed CI-STHPAN. A two-stage framework for stock selection, involving Transformer and HGAT based stock time series self-supervised pre-training and stock-ranking based downstream task fine-tuning.

$\mathcal{S}$, according to the trading data $\boldsymbol{X}_i = [\boldsymbol{x}_{t-L}, \ldots, \boldsymbol{x}_{t-1}] \in \mathbb{R}^{M \times L}$ and stock relational hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we construct the model $\mathcal{F}_\theta(\boldsymbol{X}_{[1:|\mathcal{S}|]}, \mathcal{G})$ to predict the return ratio $\hat{r}_i^t = \frac{\hat{c}_i^t - c_i^{t-1}}{c_i^{t-1}}$, where M is the feature dimension and L is the size of look back window. $\mathcal{V} = \{v_1, v_2, \ldots, v_{|s|}\}$ and $\varepsilon = \{e_1, e_2, \ldots, e_R\}$ are the set of nodes and hyperedges, respectively. c_i^{t-1} and $\hat{c}_i^t$ are the ground-truth and predict close price of stock s_i, respectively. Subsequently, based on the predicted return $\hat{\boldsymbol{R}}^t = [\hat{r}_1^t, \hat{r}_2^t, \ldots, \hat{r}_{|\mathcal{S}|}^t]$ we can rank all stocks $y^t = \left\{ y_1^t > y_2^t > \cdots > y_{|\mathcal{S}|}^t \right\}$ such that any pair of stocks $s_i, s_j \in \mathcal{S}$, when $y_i^t > y_j^t, \hat{r}_i^t > \hat{r}_j^t$, and finally we choose the top-ranked k stocks to invest on trading day t.

Data Preprocessing

Following (Feng et al. 2019), we first standardize the raw stock price data. Specifically, we calculate the 5-day, 10-day, 20-day, and 30-day moving averages based on the close prices, in order to represent the stock price time series trend from short to long, thus we get the input of the model with a time series feature dimension M of 5. We further standardize the full sample data based on the maximum value of each channel m on the training set, $\boldsymbol{x}_i^m = \frac{x_i^m}{max(\boldsymbol{x}_{i,train}^m)}$, and followed by patching and hypergraph constrution, respectively.

Stock Time Series Patching Patching empowers the model with a longer time series dependence to improve the prediction accuracy while reducing the complexity of model, the training time and GPU memory usage. As shown in Figure 2, patches are generated in the form of sliding windows over the entire stock time series, and each patch contains a fixed length P. The interval S between each patch affects the number of generated patches $N = \left\lfloor \frac{L-P}{S} \right\rfloor + 2$. The interval should not be too small, otherwise it may fail to capture the local temporal information and if it equals the length of the patch, it means there is no overlap between patches. Therefore, for each dimension of each stock with time series data $\boldsymbol{x}_i^m = [x_{t-L}^m, \ldots, x_{t-1}^m] \in \mathbb{R}^L$, after Patching, it would generate $\boldsymbol{X}_{pi}^m = [\boldsymbol{x}_1^m, \ldots, \boldsymbol{x}_N^m] \in \mathbb{R}^{P \times N}$. It can be considered that the original length L is changed to N.

Stock Hypergraph Construction We construct hypergraph among stocks, which more adequately represents the collective higher-order relationships in the stock market and have been verified by (Sawhney et al. 2021) to work better than pair-wise graphs. Specifically, we construct channel independent hypergraphs among stocks with similar stock price trends based on DTW (Salvador and Chan 2007). In each dimension, for each stock, we calculate the distance matrix $\boldsymbol{\mathcal{D}}^m$ between each point of the pair-wise sequence, and find a path to minimize the sum of elements based on the matrix as:

$$Similarity(s_i^m, s_j^m) = \mathcal{DTW}\left(\{\mathcal{D}_{pq}^m\}_{\Delta T \times \Delta T}^m\right), \quad (1)$$

$$\mathcal{D}_{pq}^m = \left(x_i^{m,p} - x_j^{m,q}\right)^2, p, q \in \{1, \ldots, \Delta T\}. \quad (2)$$

The calculated DTW value represents the similarity between two stocks and subsequently the most similar K stocks are connected by a hyperedge, which finally generates a hypergraph $\mathcal{G}_{DTW}^m = (\mathcal{V}, \mathcal{E}_{DTW})$ on channel m. To avoid using future information for model training, we construct the hypergraphs $\mathcal{G}_{DTW,train}^m = (\mathcal{V}, \mathcal{E}_{DTW,train})$, $\mathcal{G}_{DTW,valid}^m = (\mathcal{V}, \mathcal{E}_{DTW,valid})$ on the training and validation sets, respectively, and use $\mathcal{G}_{DTW,valid}^m$ in the testing phase. In addition, we also construct the domain knowledge

predefined graphs following (Sawhney et al. 2021), including the industry hypergraph $\mathcal{G}_{Industry} = (\mathcal{V}, \mathcal{E}_{Industry})$ and the wikidata association hypergraph $\mathcal{G}_{Wiki} = (\mathcal{V}, \mathcal{E}_{Wiki})$, noting that these two hypergraphs do not change dynamically nor are they channel independent, and thus are shared across the full sample and channels. Based on the different relationship hypergraphs constructed, we can generate different hypergraph adjacency matrices $\boldsymbol{H} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$, $\mathcal{E} = \mathcal{E}_{DTW}\left(\bigcup \mathcal{E}_{Industry}(\cup \mathcal{E}_{Wiki})\right)$ in random combinations.

Model Structure

Transformer Encoder CI-STHPAN extracts the stock temporal features using the vanilla Transformer Encoder (Vaswani et al. 2017). Firstly, the input feature $\boldsymbol{X}_{pi}^m$ is mapped by a linear projection and added with a learnable position encoding so that the input feature dimension is changed from P to Encoder dimension D, $\boldsymbol{X}_{ei}^m = \boldsymbol{W}_e \boldsymbol{X}_{pi}^m + \boldsymbol{W}_{pos}$, where $\boldsymbol{X}_{pi}^m \in \mathbb{R}^{P\times N}$, $\boldsymbol{X}_{ei}^m \in \mathbb{R}^{D\times N}$, $\boldsymbol{W}_e \in \mathbb{R}^{D\times P}$, $\boldsymbol{W}_{pos} \in \mathbb{R}^{D\times N}$. Subsequently, in the multi-head self-attention, each head $h = 1, \ldots, H$ generates query $\boldsymbol{Q}_h^m = \left(\boldsymbol{X}_{ei}^m\right)^T \boldsymbol{W}_h^Q$, key $\boldsymbol{K}_h^m = \left(\boldsymbol{X}_{ei}^m\right)^T \boldsymbol{W}_h^K$, value $\boldsymbol{V}_h^m = \left(\boldsymbol{X}_{ei}^m\right)^T \boldsymbol{W}_h^V$ based on $\boldsymbol{X}_{ei}^m$, where $\boldsymbol{W}_h^Q, \boldsymbol{W}_h^K \in \mathbb{R}^{D\times d_k}$, $\boldsymbol{W}_h^V \in \mathbb{R}^{D\times D}$, $d_k = \frac{D}{H}$. Finally, the output $\boldsymbol{X}_{attention,i}^m \in \mathbb{R}^{D\times N}$ is obtained from a multi-head self-attention layer as:

$$\boldsymbol{X}_{attention,i}^m = ||_{h=1}^{h=H}\, Softmax\left(\frac{\boldsymbol{Q}_h^m \boldsymbol{K}_h^m}{\sqrt{d_k}}\right)\boldsymbol{V}_h^m. \quad (3)$$

The multi-head attention also includes Batch Normalization, which has been verified to be more applicable in temporal data by (Zerveas et al. 2021). It is worth noting that different channels share all parameters as they pass through the various components of the Encoder. The Encoder eventually generates the temporal features $\boldsymbol{X}_{temporal,i}^m \in \mathbb{R}^{D\times N}$ by stacking E layers of multi-head self-attention layers through residual connections.

Hypergraph Attention Network CI-STHPAN extracts the relational features among stocks using a hypergraph attention network. Firstly, the global temporal feature $\boldsymbol{X}_{v,i}^m = \boldsymbol{X}_{temporal,i}^m \boldsymbol{W}_1 + \boldsymbol{b}_1$ is extracted from the individual stock temporal feature $\boldsymbol{X}_{temporal,i}^m \in \mathbb{R}^{D\times N}$, where $\boldsymbol{W}_1 \in \mathbb{R}^{N\times 1}$,$\boldsymbol{b}_1 \in \mathbb{R}^D$, and it is used as the initial embedding of the hypergraph attention network $\boldsymbol{X}_G^{m(0)} = \left[\boldsymbol{x}_{v,0}^m, \ldots, \boldsymbol{x}_{v,|\mathcal{V}|}^m\right] \in \mathbb{R}^{|\mathcal{V}|\times D}$, and subsequently updating each node feature with the adjacent hyperedges based on the hypergraph attention adjacency matrix $\boldsymbol{H}^m \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$ using the multi-head architecture as follows[4]:

$$\boldsymbol{x}_{v,i}^{m(l+1)} = ||_{h=1}^{H} f\left(\boldsymbol{D}_v^{-\frac{1}{2}} \boldsymbol{H}_k^m \boldsymbol{W} \boldsymbol{D}_e^{-1} \boldsymbol{H}_k^{mT} \boldsymbol{D}_v^{-\frac{1}{2}} \boldsymbol{x}_{v,i}^{m(l)} \boldsymbol{P}_k\right), \quad (4)$$

[4]The detailed derivation formula of the hypergraph convolutional network can be referred to (Sawhney et al. 2021).

where $\boldsymbol{D}_v$ and $\boldsymbol{D}_e$ are the degree matrix of each vertex and each hyperedge, respectively, $\boldsymbol{W}$ is the diagonal hyperedge weight matrix, $\boldsymbol{P}_k$ is learnable matrix, f represents the *LeakyReLU* activation function, and each element of $\boldsymbol{H}^m$ is α_{ij}^m representing attention coefficient, which indicates the importance of hyperedge j to node i, and is calculated as:

$$\alpha_{ij}^m = \frac{\exp\left(f\left(\vec{a}^T\left[\boldsymbol{W}_2 \boldsymbol{x}_{v,i}^m \parallel \boldsymbol{W}_2 \boldsymbol{e}_j^m\right]\right)\right)}{\sum_{k\in\mathcal{N}_i} \exp\left(f\left(\vec{a}^T\left[\boldsymbol{W}_2 \boldsymbol{x}_{v,i}^m \parallel \boldsymbol{W}_2 \boldsymbol{e}_k^m\right]\right)\right)}, \quad (5)$$

where $\vec{a}$ represents the feed forward network and the feature of the hyperedge $\boldsymbol{e}_j^m$ is expressed as equal-weighted averages of the features of all the nodes $\mathcal{E}_j$ connected to it:

$$\boldsymbol{e}_j^m = \frac{1}{|\mathcal{E}_j|}\sum_{k\in\mathcal{E}_j} \boldsymbol{x}_{v,k}^m. \quad (6)$$

Inspired by the Transformer, we also introduce Dropout layers and residual connections between each hypergraph attention layer, and perform Batch Normalization after each layer to maintain the robustness of the model. The final $\boldsymbol{X}_{spatio,i}^m \in \mathbb{R}^{N\times D}$ can be obtained by fusing the temporal and spatial features as:

$$\boldsymbol{X}_{spatio,i}^m = \text{MLP}(\boldsymbol{X}_{temporal,i}^m + \text{HGAT}(\boldsymbol{X}_G^{m(0)}, \mathbf{H}^m)). \quad (7)$$

Channel Independence Following (Nie et al. 2022; Han, Ye, and Zhan 2023), we only need to convert the input tensor after Patching from $[|\mathcal{S}| \times M \times D \times N]$ to $[(|\mathcal{S}| * M) \times D \times N]$, and we can consider it as a tensor of length N feature dimension D with batch size $(|\mathcal{S}| * M)$ to realize channel independence without changing any structure of the model. In addition, in order to maintain channel independence in graph feature extraction, hypergraph construction as well as node feature updating are performed channel by channel.

Instance Normalization Financial time-series are highly distribution-shifted in nature, i.e., the distribution of training set and testing set are not consistent. Following (Nie et al. 2022; Kim et al. 2021), we simply need to normalize the data using the mean and variance before Patching and re-add after the output layer to achieve Instance Normalization.

Pre-training

Self-supervised pretrain has been widely used in NLP (Devlin et al. 2018) and CV (Dosovitskiy et al. 2020) in recent years. Research on pretraining on time-series data also gradually developed in recent years, such as (Zerveas et al. 2021; Nie et al. 2022). In this paper, for the first time, a pre-trained model is constructed on stock time-series data and applied to the downstream task of stock selection. Specifically, based on the technique of masked autoencoder, the data $\boldsymbol{X}_{pi}^m \in \mathbb{R}^{P\times N}$ after Patching is randomly masked on patches with a certain proportion to generate masked data $\boldsymbol{X}_{mask,i}^m = [x_1^m, \ldots, x_{maski}^m, \ldots, x_N^m] \in \mathbb{R}^{P\times N}$, as shown in Figure 3. After the fused temporal and spatial feature $\boldsymbol{X}_{spatio,i}^m$ generated by the model introduced in Section 3.3, the regenerated temporal data $\hat{\boldsymbol{X}}_{mask,i}^m = [\hat{\boldsymbol{x}}_1^m, \ldots, \hat{\boldsymbol{x}}_{mask}^m, \ldots, \hat{\boldsymbol{x}}_N^m] \in$

	Model	Methods	NASDAQ		NYSE	
			IRR	SR	IRR	SR
CLF	**ARIMA** (Wang and Leu 1996)	RNN with ARIMA Features	0.10	0.55	0.10	0.33
	HGCluster (Luo et al. 2014)	Stock trend to hypergraph clustering	0.10	0.06	0.11	0.10
	Adv-LSTM (Feng et al. 2018)	Adversarial training	0.23	0.97	0.14	0.81
	HATS (Kim et al. 2019)	Diverse stock relationships	0.15	0.80	0.12	0.73
	HMG-TF (Ding et al. 2020)	Multiscale Gaussian prior	0.19	0.83	0.13	0.75
	LSTM-RCGN (Li et al. 2021)	Using news to predict overnight stock	0.13	0.75	0.10	0.70
	HATR (Wang et al. 2021)	Multi-scale local combinations	0.31	0.92	0.14	0.76
REG	**SFM** (Zhang, Aggarwal, and Qi 2017)	State-frequency memory	0.09	0.16	0.11	0.19
	DA-RNN (Qin et al. 2017)	Two-stage Attetion-RNN	0.14	0.71	0.13	0.66
RL	**DQN** (Carta et al. 2021)	Maximise the gain function	0.20	0.93	0.12	0.72
	iRDPG (Liu et al. 2020)	Intelligent trading agents	0.28	1.32	0.18	0.85
	RAT (Xu et al. 2021a)	Relation-aware transformers	0.40	1.37	0.22	1.03
RAN	**SAE-LSTM** (Bao, Yue, and Rao 2017)	Wavelet Transform & Stacked Autoencoder	0.22	0.95	0.12	0.79
	RSR-I (Feng et al. 2019)	Temporal graph convolution	0.39	1.34	0.21	0.95
	STHAN-SR (Sawhney et al. 2021)	Spatio-temporal attention hypergraph	0.44	1.42	0.33	1.12
	ALSP-TF (Wang et al. 2022)	Adaptive long-short pattern Transformer	<u>0.53</u>	<u>1.55</u>	<u>0.41</u>	<u>1.24</u>
	CI-STHPAN (Ours)	Spatial-temporal pre-training	**0.66**	**2.01**	**0.79**	**2.14**

Table 1: Profitability comparison with Classification (CLF), Regression (REG), Reinforcement Learning (RL), and Ranking (RAN) baselines. Bold & underline depict the best & second-best results($p<0.01$).

$\mathbb{R}^{P\times N}$ is recovered by a linear layer, and finally we train the self-supervised pretrain model by the MSE on the masked patch channel-by-channel:

$$\mathcal{L}_{\text{pretrain}} = \frac{1}{M\times|\mathcal{S}|\times N\times P}\sum_{m=1}^{M}\sum_{s=1}^{|\mathcal{S}|}\sum_{i=1}^{N}\sum_{p=1}^{P}\left(\hat{\boldsymbol{x}}_{\text{mask},i}^{m,p}-\boldsymbol{x}_{\text{mask},i}^{m,p}\right)^2, \tag{8}$$

where M is the number of channels, $|\mathcal{S}|$ is the number of stocks, N is the number of patches, and P is the length of patches.

Finetune

After pre-training on a particular stock market, we fine-tune it on the downstream task of stock ranking. Specifically, the feature representation $\boldsymbol{X}_{spatio,i}^{m}$ generated by the model introduced in Section 3.3, which fused both temporal and spatial feature, is passed through a flatten layer with linear head to predict the close price $\hat{c}_i^t$ for each stock of the next trading day, and further translated into a return ratio $\hat{r}_i^t = \frac{\hat{c}_i^t - c_i^{t-1}}{c_i^{t-1}}$, where c_i^{t-1} is the true close price of stock s_i on the previous trading day, and finally, following (Feng et al. 2019; Sawhney et al. 2021), the model is fine-tuned by combining point-wise regression loss and pair-wise ranking loss as:

$$\mathcal{L}_{finetune} = \|\hat{\boldsymbol{r}}_i^t - \boldsymbol{r}^t{}_i\|^2 + \alpha\sum_{i=1}^{|\mathcal{S}|}\sum_{j=1}^{|\mathcal{S}|} max\left(0, -\left(\hat{\boldsymbol{r}}_i^t - \hat{\boldsymbol{r}}_j^t\right)\left(\boldsymbol{r}_i^t - \boldsymbol{r}_j^t\right)\right), \tag{9}$$

where α is a hyperparameter to balance the two loss terms.

Experiments

Datasets The datasets used in our experiments are consistent with Ref.(Feng et al. 2019) and contain historical stock trading data between 2013 and 2017 for two markets, i.e., NASDAQ and NYSE, the latter being more stable than the former and containing 1026 and 1737 stocks respectively, as well as inter-stock sector-industry relation and wiki relation data obtained from Wikidata. To avoid the data leakage problem, we strictly follow the sequential order to split training/validation/testing sets by 3-year/1-year/1-year. The detailed statistics of the datasets is shown in Table 2.

Martket	NASDAQ	NYSE
Stocks(Nodes)	1,026	1,737
Training Period	01/02/2013 - 12/31/2015	
Validation Period	01/04/2016 - 12/30/2016	
Testing Period	01/03/2017 - 12/08/2017	
Days(Train:Valid:Test)	756 : 252 : 237	

Table 2: Statistics of datasets.

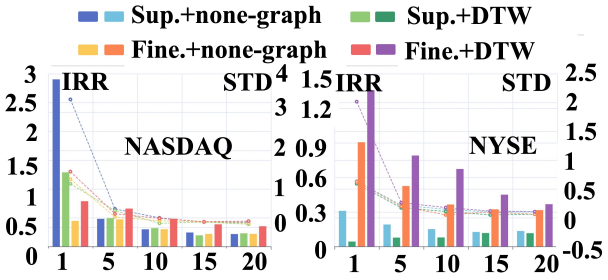


Figure 4: Profitability of the selected top k stocks. STD denotes the standard deviation of 5 independent runs.

Graph	Method	Component	NASDAQ IRR	NASDAQ SR	NASDAQ IC	NASDAQ ICIR	NYSE IRR	NYSE SR	NYSE IC	NYSE ICIR
×	Sup.	*w.o. both*	0.275	0.994	0.019	0.227	0.208	0.711	0.021	0.224
		w. CI	0.351	1.564$^{*\#}$	0.008	0.122	0.262	0.794	0.011	0.123
		w. RevIN	0.530$^{*\#}$	1.544$^{\#}$	-0.002	-0.033	0.198	0.738	-0.027	-0.261
		w. both	0.479$^{\#}$	1.334	0.029	0.294	0.190	0.711	0.030	0.303
	Fine.	*w.o. both*	0.181	0.796	0.023	0.228	0.171	0.614	0.019	0.185
		w. CI	**0.787**	**2.117**$^{*\#}$	0.005	0.057	0.621$^{*\#}$	1.536$^{*\#}$	0.001	0.000
		w. RevIN	0.470$^{\#}$	1.415	0.059	0.494	0.363$^{\#}$	1.101	0.049	0.482
		w. both	0.469$^{\#}$	1.452$^{\#}$	0.058	0.428	0.524$^{*\#}$	1.474$^{*\#}$	**0.078**	**0.658**
✓	Sup.	*w.o. both*	0.197	0.834	0.012	0.145	0.175	0.641	0.028	0.251
		w. CI	0.310	1.617$^{*\#}$	0.023	0.302	0.239	1.024	0.018	0.224
		w. RevIN	0.380	1.340	-0.001	-0.008	0.240	0.841	-0.011	-0.108
		w. both	0.491$^{\#}$	1.591$^{*\#}$	-0.005	-0.061	0.077	0.428	0.001	0.004
	Fine.	*w.o. both*	0.293	1.025	0.031	0.341	0.089	0.381	0.017	0.162
		w. CI	0.164	0.685	0.011	0.121	0.224	0.824	0.019	0.236
		w. RevIN	0.334	1.067	**0.091**	**0.581**	0.488$^{*\#}$	1.408$^{*\#}$	0.052	0.565
		w. both	0.657*	2.013$^{*\#}$	0.043	0.360	**0.789**$^{*\#}$	**2.136**$^{*\#}$	0.053	0.569

Table 3: Ablation study over CI-STHPAN's components. *Sup.* and *Fine.* represents supervised learning and fine-tuning, respectively. *CI* and *RevIN* represents Channel Independence and Revisable Instance Normalization, respectively. The specific implementation of the method *w.o. CI* is to convert the input tensor after Patching from $[|\mathcal{S}| \times M \times D \times N]$ to $[|\mathcal{S}| \times (M * D) \times N]$ and construct only one hypergraph. * and $^{\#}$ indicate the improvement over the ALSP-TF and STHAN-SR, respectively.

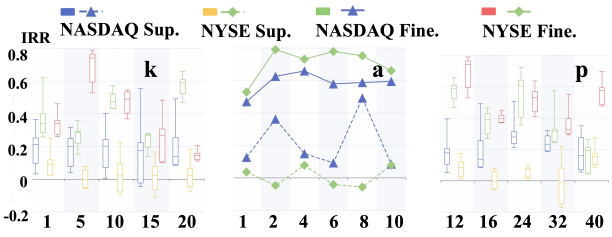


Figure 5: Influence of (a)graph sparsity k, (b)loss factor α, and (c)patch length P. The hypergraph constructed on NASDAQ and NYSE are DTW-20 and DTW-5, respectively.

Implementation Details Our experiments are implemented with PyTorch and PyG, and the results reported are run with the fixed random seed 2023. The look back window in supervised and self-supervised model are 336 and 512, respectively. The layers of the Encoder is 3, dimension D is 128, attention heads H is 16, the hidden units of MLP with *GELU* is 256 and the attention heads in HGAT is 4. We use grid search to find optimal hyperparameters based on the loss of validation set, including loss factor $\alpha \in \{1, 2, 4, 6, 8, 10\}$, DTW top $K \in \{1, 5, 10, 15, 20\}$ and patch length $P \in \{12, 16, 24, 32, 40\}$. All of the experiments are done on a GeForce RTX 3090 Ti GPU by Adam optimizer. The self-supervised models are first pretrained, then finetuned the head and the entire network. The learning rate is adjusted using the OneCycleLR with a maximum value of $1e-4$, and the batch size equals $|S|$.

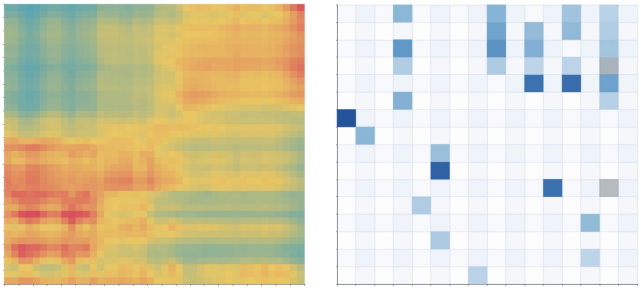
Metrics Following (Feng et al. 2019; Sawhney et al. 2021), we perform a daily buy-hold-sell trading strategy, i.e., the model predicts the return of all individual stocks of the next trading day t+1 at trading day t and we rank and hold the top k stocks S^k and then sell them on trading day t+1. The return per trading day is $R^t = \sum_{i \in S^k} \frac{c_i^t - c_i^{t-1}}{c_i^{t-1}}$. We backtest on the entire testing set and calculate the **IRR** and the Sharpe ratio: $\boldsymbol{SR} = \frac{E[\boldsymbol{R} - R_f]}{std[\boldsymbol{R} - R_f]}$, where R_f is the risk-free rate. In addition, in order to measure the accuracy and stability of the model prediction, we further introduce four metrics, Information Coefficient (**IC**) which computed by the average Pearson correlation coefficient, Information ratio based IC (**ICIR**), calculated by $mean(IC)/std(IC)$.

Overall Performance

In Table 1, we compare CI-STHPAN with four types of baseline methods in terms of profitability. First, CI-STHPAN achieves higher risk-adjusted returns $(p < 0.01)$ than all the baseline models in both NASDAQ and NYSE markets. Second, we find that Transformer-based methods (e.g., HMG-TF, RAT, ALSP-TF) outperform RNN-based methods (e.g., SFM, DARNN, SAE-LSTM), which is attributed to the long-term time-series modeling capability of the self-attention mechanism. We also observe that methods constructing implicit dynamic relations (e.g., ALSP-TF, CI-STHPAN) outperform methods based on predefined relations (e.g., Rank_LSTM, HATS, STHAN-SR), not to mention those that only consider price data (e.g., Adv-LSTM, DQN, HATR). These observations collectively validate the utility of CI-STHPAN as a dynamic spatiotemporal attention based learning to rank stock selection model.

Method	Sparsity	Supervised IRR	Supervised SR	Supervised IC	Supervised ICIR	Fine-tuning IRR	Fine-tuning SR	Fine-tuning IC	Fine-tuning ICIR
None-graph	-	0.479$^{\#}$	1.334	**0.029**	**0.294**	0.469$^{\#}$	1.452$^{\#}$	0.058	0.428
All	*1.09%*	0.311	1.032	-0.006	-0.084	0.386	1.381	0.042	0.301
Industry+Wiki	*0.26%*	0.358	1.045	0.003	0.030	0.440$^{\#}$	1.674$^{*\#}$	0.032	0.284
Industry+DTW	*1.96%*	0.272	0.912	-0.005	-0.046	0.396	1.511$^{\#}$	0.041	0.310
Wiki+DTW	*1.09%*	0.452$^{\#}$	1.428$^{\#}$	0.001	0.011	0.475$^{\#}$	1.691$^{*\#}$	0.059	0.430
Industry	*1.02%*	0.380	1.200	-0.009	-0.116	0.447$^{\#}$	1.352	0.045	0.370
Wiki	*0.19%*	0.263	1.009	-0.001	-0.023	0.591$^{*\#}$	1.856$^{*\#}$	0.062	0.442
DTW_K=20	*2.05%*	0.491$^{\#}$	1.591$^{*\#}$	-0.005	-0.061	**0.657**$^{*\#}$	**2.013**$^{*\#}$	0.043	0.360
DTW_K=15	*1.56%*	**0.553**$^{*\#}$	**1.626**$^{*\#}$	-0.006	-0.078	0.281	1.092	0.048	0.397
DTW_K=10	*1.07%*	0.402	1.411	-0.006	-0.073	0.573$^{*\#}$	1.811$^{*\#}$	**0.068**	**0.461**
DTW_K=5	*0.58%*	0.317	1.027	0.006	0.064	0.359	1.382	0.053	0.409
DTW_K=1	*0.19%*	0.363	1.221	-0.005	-0.061	0.620$^{*\#}$	1.926$^{*\#}$	**0.068**	0.460
RSR-I	*0.19%*	0.390	1.340	0.020	0.217		-		
STHAN-SR	*0.26%*	0.440	1.420	0.015	0.256		-		

Table 4: Influence of different graph information on NASDAQ. *Industry* and *Wiki* relation represent pre-defined domain knowledge. *DTW_K* represents channel-independent stock dynamic relation based on the similarity.



(a) Encoder attention (b) Hypergraph attention

Figure 6: Attention visualization

Fine. on	Pre. on	IRR	SR	IC	ICIR
NASDAQ	NASDAQ	0.657	2.013	0.043	0.360
	NYSE	0.273	1.030	0.066	0.457
NYSE	NASDAQ	0.694	1.928	0.062	0.585
	NYSE	0.789	2.136	0.053	0.569

Table 5: Transfer learning

Ablation Study

In Table 3, firstly, we can observe that *Fine.* improves the model's metrics compared to *Sup.*, and it is particularly notable on the NYSE market. Secondly, the performance of the model improves with the addition of CI and RevIN, more significantly in the case of the introduction of the graph module. In addition, Comparing *w.o. both* vs. *w. CI* and *w. RevIN* vs. *w. both* respectively, we can find that the CI significantly improves the model's IRR and SR metrics.

In-depth Analysis

Graph Information In Table 4, we construct various hypergraphs based on different relation among stocks. We can find that the construction of dynamic implicit stock relation based on DTW performs better compared to predefined relation which is mainly due to the improvement of IC and ICIR, indicating that *Fine.* improves the accuracy and stability of the model. Due to space limitations only NASDAQ results are shown, NYSE results are consistent.

Hyperparameter Figure 4 demonstrates the returns and risks associated with holding different top k stocks, and we can find that holding the top 5 stocks can obtain satisfactory returns and risks. Figure 5 shows the effect of three hyperparameters, loss factor α, graph sparsity K and patch length P, on the model performance in turn. It can be observed that the maximum return can be realized when α=2, P=12, K=20 on NASDAQ and K=5 on NYSE in fine-tuning method.

Attention Visualization Figure 6 illustrates the self-attention weights of temporal features in one of the heads in Encoder and part of stock nodes with their connected hyper edges, respectively, which suggests that CI-STHPAN can provide insights about the influence of lookback window and relevance between stock.

Transfer Learning Table 5 shows the results of pre-training on one market followed by fine-tuning on the other. It can be observed that the pre-trained model on NASDAQ also performs better on the NYSE.

Conclusion

In this paper, we propose CI-STHPAN, a two-stage framework for stock selection, involving Transformer and HGAT based stock time series self-supervised pre-training and stock-ranking based downstream task fine-tuning. In addition, we also construct channel-independent stock dynamic hypergraph based on DTW. Through quantitative and qualitative analysis of the NASDAQ and NYSE markets, we explore this capability of CI-SHPAN to extract features from time series and relational graphs and validate its applicability to quantitative stock selection.

Acknowledgments

This research is supported by the National Key Research and Development Program of China (2023YFC3304800), the Chinese Academy of Engineering strategic research and consulting project "Strategic Research on Financial Risk Monitoring and Early Warning System under the Background of Digital Transformation" (2023-XY-43), and the Natural Science Foundation of Shanghai (23ZR1404900).

References

Bao, W.; Yue, J.; and Rao, Y. 2017. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7): e0180944.

Carta, S.; Ferreira, A.; Podda, A. S.; Recupero, D. R.; and Sanna, A. 2021. Multi-DQN: An ensemble of Deep Q-learning agents for stock market forecasting. *Expert systems with applications*, 164: 113820.

Chen, Y.; Wei, Z.; and Huang, X. 2018. Incorporating corporation relationship via graph convolutional neural networks for stock price prediction. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1655–1658.

Cheng, R.; and Li, Q. 2021. Modeling the momentum spillover effect for stock prediction via attribute-driven graph attention networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 55–62.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ding, Q.; Wu, S.; Sun, H.; Guo, J.; and Guo, J. 2020. Hierarchical Multi-Scale Gaussian Transformer for Stock Movement Prediction. In *IJCAI*, 4640–4646.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

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.

Han, L.; Ye, H.-J.; and Zhan, D.-C. 2023. The Capacity and Robustness Trade-off: Revisiting the Channel Independent Strategy for Multivariate Time Series Forecasting. *arXiv preprint arXiv:2304.05206*.

Hsu, Y.-L.; Tsai, Y.-C.; and Li, C.-T. 2021. FinGAT: Financial Graph Attention Networks for Recommending Top-K K Profitable Stocks. *IEEE Transactions on Knowledge and Data Engineering*, 35(1): 469–481.

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.

Kalyan, K. S.; Rajasekharan, A.; and Sangeetha, S. 2021. Ammus: A survey of transformer-based pretrained models in natural language processing. *arXiv preprint arXiv:2108.05542*.

Karita, S.; Chen, N.; Hayashi, T.; Hori, T.; Inaguma, H.; Jiang, Z.; Someki, M.; Soplin, N. E. Y.; Yamamoto, R.; Wang, X.; et al. 2019. A comparative study on transformer vs rnn in speech applications. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 449–456. IEEE.

Khan, S.; Naseer, M.; Hayat, M.; Zamir, S. W.; Khan, F. S.; and Shah, M. 2022. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s): 1–41.

Kim, R.; So, C. H.; Jeong, M.; Lee, S.; Kim, J.; and Kang, J. 2019. Hats: A hierarchical graph attention network for stock movement prediction. *arXiv preprint arXiv:1908.07999*.

Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; and Choo, J. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32.

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.

Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A. X.; and Dustdar, S. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.

Liu, Y.; Liu, Q.; Zhao, H.; Pan, Z.; and Liu, C. 2020. Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 2128–2135.

Liu, Y.; Wu, H.; Wang, J.; and Long, M. 2022. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35: 9881–9893.

Luo, Y.; Hu, J.; Wei, X.; Fang, D.; and Shao, H. 2014. Stock trends prediction based on hypergraph modeling clustering algorithm. In *2014 IEEE International Conference on Progress in Informatics and Computing*, 27–31. IEEE.

Muhammad, T.; Aftab, A. B.; Ibrahim, M.; Ahsan, M. M.; Muhu, M. M.; Khan, S. I.; and Alam, M. S. 2023. Transformer-based deep learning model for stock price prediction: A case study on Bangladesh stock market. *International Journal of Computational Intelligence and Applications*, 2350013.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*.

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*.

Saha, S.; Gao, J.; and Gerlach, R. 2022. A survey of the application of graph-based approaches in stock market analysis and prediction. *International Journal of Data Science and Analytics*, 14(1): 1–15.

Salvador, S.; and Chan, P. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5): 561–580.

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.

Sawhney, R.; Agarwal, S.; Wadhwa, A.; and Shah, R. R. 2020. Spatiotemporal hypergraph convolution network for stock movement forecasting. In *2020 IEEE International Conference on Data Mining (ICDM)*, 482–491. IEEE.

Sezer, O. B.; Gudelek, M. U.; and Ozbayoglu, A. M. 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90: 106181.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34: 22419–22430.

Xu, K.; Zhang, Y.; Ye, D.; Zhao, P.; and Tan, M. 2021a. Relation-aware transformer for portfolio policy learning. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, 4647–4653.

Xu, W.; Liu, W.; Wang, L.; Xia, Y.; Bian, J.; Yin, J.; and Liu, T.-Y. 2021b. Hist: A graph-based framework for stock trend forecasting via mining concept-oriented shared information. *arXiv preprint arXiv:2110.13716*.

Xu, Y.; and Cohen, S. B. 2018. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1970–1979.

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.

Zerveas, G.; Jayaraman, S.; Patel, D.; Bhamidipaty, A.; and Eickhoff, C. 2021. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2114–2124.

Zhang, L.; Aggarwal, C.; and Qi, G.-J. 2017. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2141–2149.

Zhang, Z.; Cui, P.; and Zhu, W. 2020. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1): 249–270.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*, 27268–27286. PMLR.

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*.