

Hypergraph Neural Networks to Predict Stock Movements By Exploring Higher-order Relationships

Tuna Alaygut
Artificial Intelligence and Data Engineering Department
Ozyegin University
Istanbul, Select State, Turkiye
tuna.alaygut@ozyegin.edu.tr

Emre Sefer*
Artificial Intelligence and Data Engineering Department
Ozyegin University
Istanbul, Select State, Turkiye
emre.sefer@ozyegin.edu.tr

Abstract

Predicting stock price movements can be framed as a classification task, where the goal is to anticipate whether a stock will increase, decrease, or remain stable. Most existing approaches rely solely on the movement patterns of individual stocks or stock pairs, overlooking the more complex, higher-order connections that exist among groups of stocks. In practice, stocks are often interrelated in higher orders, for example, by belonging to the same industry sector or being jointly held within the same investment fund. To address this, we compare 4 hypergraph neural network-based approaches to make spatio-temporal predictions for stock movement prediction, which explicitly leverages these higher-order dependencies. We use two heterogeneous hypergraphs, where one hypergraph represents sector-based associations and the other one represents fund-holding relationships among stocks. In general, we found the hierarchical hypergraph attention mechanism and temporal attention to be effective in achieving better performance. A hierarchical hypergraph attention mechanism models these relationships by weighting the contributions of stock nodes, hyperedges, and even the hypergraphs themselves. Temporal attention captures time-dependent dynamics of both stock and sector sequences, effectively accounting for the influence of past states. Experiments on real-world datasets demonstrate that the methods specializing in hypergraph integration achieve superior performance compared to existing methods, both in terms of predictive accuracy and profitability.

CCS Concepts

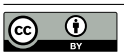
• **Computing methodologies** → **Artificial intelligence**; **Machine learning**; **Neural networks**; • **Applied computing** → **Economics**.

Keywords

Hypergraph Neural Network, Spatiotemporal Prediction, Stock Movement Prediction

ACM Reference Format:
Tuna Alaygut and Emre Sefer. 2025. Hypergraph Neural Networks to Predict Stock Movements By Exploring Higher-order Relationships. In *6th ACM International Conference on AI in Finance (ICAIF '25), November 15–18, 2025, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3768292.3770389

*Corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License.
ICAIF '25, Singapore, Singapore
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2220-2/25/11
https://doi.org/10.1145/3768292.3770389

1 Introduction

For many years, the stock market has remained one of the most significant avenues for both individual and institutional investors seeking to build wealth. Recent reports show that by the first quarter of 2021, the combined market capitalization of major global stock exchanges had surpassed 100 trillion U.S. dollars. While the well-known Efficient Market Hypothesis [11] argues pessimistically that future stock prices cannot be predicted from currently available information, decades of continued research [44] on stock price forecasting have achieved remarkable progress and provided compelling evidence supporting the predictability of stock markets.

The stock market has long been a focal point of interdisciplinary research at the intersection of artificial intelligence and finance because of its vast profit potential [43]. Stock movement prediction aims to forecast the future direction of stock prices and is typically formulated as a time series forecasting problem [25]. In this case, machine learning and deep learning approaches become promising as they produce good results on a distinct set of domains [5, 23, 24, 28–33, 37]. Traditional approaches often rely on classical machine learning techniques such as Logistic Regression [45] and Support Vector Machines [47], while others incorporate sentiment analysis [35, 40] by leveraging textual data to anticipate market trends. However, the stock market is characterized by strong non-stationarity, high volatility, and rapidly growing data volumes. These challenges have motivated the adoption of deep learning models, which excel at capturing the temporal evolution of historical trading patterns. Among them, graph-based methods model stocks as nodes and their pairwise relationships as edges, enabling representation learning via graph convolution before predicting price movements through fully connected layers [20]. To better reflect the dynamic nature of stock relationships, time-aware graph models have been introduced [14, 39].

Yet, stock interactions often extend beyond simple pairwise links; many are shaped by higher-order relationships involving entire groups of stocks. For instance, some higher-order relationship examples are companies within the same industry sector or securities jointly held by a single investment fund. In a conventional graph, where each edge connects only two nodes, representing such group-level dependencies requires forming fully connected subgraphs among all related stocks. This results in extremely dense graph structures that hinder the efficient propagation of meaningful stock price movement signals [10, 41].

A stock sector refers to a group of companies that operate within the same industry, such as healthcare, energy, or real estate [38]. Stocks within the same sector tend to share comparable attributes and often display correlated price movements [16]. Economic activity typically follows recognizable cycles, during which industries and their leading companies experience periods of growth or decline depending on the stage of the cycle. These shifts are mirrored in the performance of popular investment sectors, offering investors insight into potential opportunities and early signals of changing market trends [17]. Consequently, analyzing and leveraging the sector rotation effect, the cyclical movement of capital between sectors, represents a valuable approach to enhancing stock movement prediction. In a similar way, an investment fund can be viewed as a financial vehicle composed of stocks with shared attributes [19]. Stocks held within the same fund, therefore, demonstrate higher-order relationships, as their inclusion reflects collective investment strategies and shared market characteristics.

In summary, stock movement prediction faces two main challenges:

(1) Modeling higher-order relationships among multiple stocks: Traditional graph-based approaches are limited because each edge connects only two nodes. When representing a group of related stocks, this leads to constructing a fully connected subgraph, which is inefficient for propagating information. Hypergraphs, on the other hand, extend the concept of graphs by allowing hyperedges that can directly link multiple stocks simultaneously, providing a more natural and scalable way to represent group-level dependencies.
(2) Leveraging the sector rotation effect to enhance prediction: A sector index is derived from the weighted prices of its constituent stocks and strongly correlates with their performance. However, sector indexes themselves exhibit distinct price dynamics because each sector has a unique composition of companies. Exploiting these differences offers an opportunity: by predicting sector rankings and identifying capital flows between sectors, one can better capture the overall market direction and refine individual stock movement predictions, ultimately improving forecast accuracy.

In this paper, we compare 4 hypergraph neural networks for stock movement prediction. Among these methods, [9] proposes Hypergraph Tri-Attention Network (HGTAN) that enhances hypergraph convolutional networks by organizing attention into three hierarchical levels: intra-hyperedge, inter-hyperedge, and inter-hypergraph modules. This structure enables HGTAN to dynamically assess the relative significance of nodes, hyperedges, and entire hypergraphs during information flow among stocks, thereby fully leveraging potential correlations and cooperative effects in stock price movements. [27] introduces STHAN-SR, a neural hypergraph model designed for stock selection, which casts the task of stock prediction into a learning-to-rank framework. Its central innovation lies in capturing the intricate interrelationships among stocks by representing them as a hypergraph and integrating a temporal Hawkes–based attention. [42] introduces HGTS-Former, which divides the data into patches, with each patch embedded as a token. A multi-head self-attention mechanism is then applied to strengthen the temporal representations of these tokens. To capture complex structure, they build hierarchical hypergraphs that integrate temporal patterns within individual channels and fine-grained relationships across different variables. [36] introduces a Multi-Scale Hypergraph Transformer (MSHyper), which reinterprets hyperedges as nodes and forms a hyperedge graph that strengthens and enriches the overall hypergraph modeling capability. We compare the performance of these hypergraph neural networks with a comprehensive set of baselines. We conduct experiments on the real-world dataset to demonstrate the effectiveness of the methods in terms of accuracy and profitability.

2 Related Work

2.1 Stock Prediction

Conventional time series forecasting techniques, such as Kalman Filters [3], rely on past technical indicators of stocks to estimate future price movements. Machine learning models like Random Forest [26] and XGBoost [46] have achieved notable gains in predictive accuracy and remain widely applied in investment analysis. However, their effectiveness is constrained by the need for manually engineered features, which depend heavily on domain expertise and often fail to capture the complex, non-linear dependencies within temporal data. With the advancement of deep learning, models such as Convolutional Neural Networks (CNNs) [8] and Long Short-Term Memory (LSTM) [22] have become popular alternatives for stock price prediction [1]. These models have been further enhanced by integrating a wider spectrum of external information sources, such as financial news, social media signals, and company profiles, to boost prediction accuracy [21].

More recently, graph neural networks (GNNs) have demonstrated strong potential for stock forecasting by explicitly modeling relationships among companies, for example, corporate hierarchies and industry connections [2, 15]. Yet, most graph-based methods still treat these relationships as pairwise links. Stocks, however, are often tied together as part of larger collective groups, and reducing such group-wise structures to simple pairs can result in the loss of critical higher-order relational information.

2.2 Hypergraph Learning

A hypergraph is a more general relational structure than a conventional graph. Instead of edges that link only two nodes, a hyperedge in a hypergraph can simultaneously connect any number of nodes, enabling richer representations of complex relationships [4]. Hypergraph theory has been successfully applied in diverse fields, including image classification, object segmentation, and multimodal learning. In the financial domain, Hypergraph Convolutional Neural Networks (HGCNs) have been employed to learn stock representations by leveraging group-level relationship information [12]. However, a key limitation of HGCNs is that they treat all neighboring stocks within a hyperedge as equally important, which can lead to inaccurate information propagation to the target node [7].

To address this, various enhanced models have been proposed. For example, [27] introduced a Spatio-Temporal Hypergraph Attention Network, capturing inter-stock relations of different types and strengths to support stock ranking. Similarly, [9] developed

Temporal-Relational Hypergraph Tri-Attention Networks to better model group-wise stock dependencies.

3 Problem Definition

This section provides the necessary data terminology and formulates the research problem examined throughout the paper.

Relative Price Strength (RPS): RPS_m^t denotes the relative strength of the rate of change in a sector's index over the past m days at day t. A larger value of RPS_m^t indicates a stronger upward trend in the corresponding sector index. Let x_z^t represent the index value of sector z on day t. Then, the m-day index change rate of sector z at day t, denoted as $c(z)_m^t$, is defined by:

$$c(z)_m^t = \frac{x_z^t - x_z^{t-m}}{x_z^{t-m}}.$$

After computing the index change rates for all sectors on day t, they are ranked in descending order. The rank of sector z at day t is expressed as $r(z)_m^t$.

Stock Hypergraphs: $\mathcal{G}_z = (\mathcal{S}, \mathcal{E}_z, \boldsymbol{w}_z)$ and $\mathcal{G}_f = (\mathcal{S}, \mathcal{E}_f, \boldsymbol{w}_f)$ are two hypergraphs representing the sector-belonging and fund-holding relationships of stocks. $\mathcal{E}_z$ and $\mathcal{E}_f$ are sets of hyperedges, where each hyperedge contains stocks that belong to the same sector or are held by the same fund. $\boldsymbol{w}_z$ and $\boldsymbol{w}_f$ are hyperedge weight vectors, where the element $w_{z,e}$ is the weight of hyperedge $e \in \mathcal{E}_z$.

Stock Movement Prediction: Given a set $\mathcal{Z}$ consisting of N sectors, the sequence data of sector z are denoted by $\boldsymbol{R}_z = \left[\boldsymbol{r}_z^1, \ldots, \boldsymbol{r}_z^w\right], z \in \mathcal{Z}$, where $\boldsymbol{r}_z^t \in \mathbb{R}^k$ is the multi-scale RPS data of z at day $t \in [1, w]$, and $[\cdot, \cdot]$ is the concatenation operation. k is the dimension of the sector RPS data. w is the lookback window length for sequence data. For a set $\mathcal{S}$ containing M stocks, the sequence data of stock s over w days are denoted as $\boldsymbol{X}_s = \left[\boldsymbol{x}_s^1, \ldots, \boldsymbol{x}_s^w\right], s \in \mathcal{S}$, where $\boldsymbol{x}_s^t \in \mathbb{R}^d$ denotes the various technical indicators of s at day $t \in [1, w]$ and d is the dimension of the stock features.

4 Methods

4.1 HGTAN: Temporal-Relational Hypergraph Tri-Attention Networks

To model relations, HGTAN [9] constructs heterogeneous hypergraphs capturing group relationships among stocks, such as industry membership and fund holdings. Let there be K types of hypergraphs. For each hypergraph type k, we make the following definitions: 1- A node set $V = \{1, 2, \ldots, N\}$ representing stocks, 2- A hyperedge set $\mathcal{E}^{(k)} = \{e_1^{(k)}, e_2^{(k)}, \ldots\}$, where each hyperedge $e_j^{(k)} \subseteq V$ is a group (e.g. stocks sharing the same industry or held by the same fund), 3- A hypergraph incidence matrix $\mathbf{H}^{(k)} \in \{0,1\}^{N \times |\mathcal{E}^{(k)}|}$, where

$$H_{i,j}^{(k)} = \begin{cases} 1 & \text{if stock } i \in e_j^{(k)}, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, each hypergraph encodes groupwise relationships beyond pairwise edges.

4.1.1 Temporal Encoding via Attention. Before relational modeling, the model encodes per-stock temporal features. Let $\mathbf{h}_i^{(t)}$ be a hidden state for stock i at time t. HGTAN uses an attention-based recurrent layer (e.g. an attention-augmented RNN) to get final temporal embeddings

$$\mathbf{h}_i = \text{TemporalEncoder}\big(\{\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(T)}\}\big).$$

So each stock i is associated with a summary embedding $\mathbf{h}_i \in \mathbb{R}^d$. To integrate relational information, HGTAN introduces three levels of attention to guide information flow:

(1) Intra-hyperedge attention: determines the relative importance of nodes within the same hyperedge (i.e., which stocks in a hyperedge should contribute more to the hyperedge embedding).
(2) Inter-hyperedge attention: determines how much each hyperedge (out of many) contributes when reconstructing or propagating back to a node.
(3) Inter-hypergraph attention: integrates across multiple hypergraph types (e.g. industry vs fund hypergraphs), weighting hypergraphs differently.

Hyperedge embedding via intra-hyperedge attention. For the hypergraph type k, for each hyperedge $e_j^{(k)}$, HGTAN computes a hyperedge-level embedding $\mathbf{u}_j^{(k)}$ as a weighted aggregation of member node embeddings:

$$\mathbf{u}_j^{(k)} = \sum_{i \in e_j^{(k)}} \alpha_{i \to j}^{(k)} \, \mathbf{h}_i$$

where the intra-hyperedge attention weights $\alpha_{i \to j}^{(k)}$ are computed via an attention mechanism such as softmax of compatibility scores:

$$\alpha_{i \to j}^{(k)} = \frac{\exp\big(a_{\text{intra}}^{(k)}(\mathbf{h}_i, e_j^{(k)})\big)}{\sum_{i' \in e_j^{(k)}} \exp\big(a_{\text{intra}}^{(k)}(\mathbf{h}_{i'}, e_j^{(k)})\big)}.$$

Here $a_{\text{intra}}^{(k)}(\cdot, \cdot)$ is a learnable attention scoring function such as a feedforward network.

Node-level update via inter-hyperedge attention. Then, for each node i, HGTAN aggregates messages from hyperedges to update a relational embedding $\mathbf{h}_i^{(k)\prime}$ (for hypergraph k):

$$\mathbf{h}_i^{(k)\prime} = \sum_{j : i \in e_j^{(k)}} \beta_{j \to i}^{(k)} \, \mathbf{u}_j^{(k)}$$

where

$$\beta_{j \to i}^{(k)} = \frac{\exp\big(a_{\text{edge}}^{(k)}(\mathbf{u}_j^{(k)}, \mathbf{h}_i)\big)}{\sum_{j' : i \in e_{j'}^{(k)}} \exp\big(a_{\text{edge}}^{(k)}(\mathbf{u}_{j'}^{(k)}, \mathbf{h}_i)\big)}.$$

Again $a_{\text{edge}}^{(k)}$ is a learnable attention scoring function. Thus, node i receives relational aggregated features from hyperedges it belongs to.

Cross-hypergraph fusion via inter-hypergraph attention. If there are multiple hypergraph types $k = 1, \ldots, K$, each gives an updated embedding $\mathbf{h}_i^{(k)\prime}$. The model fuses them to produce a final relational embedding $\mathbf{h}_i^{\text{rel}}$:

$$\mathbf{h}_i^{\text{rel}} = \sum_{k=1}^{K} \gamma_k \, \mathbf{h}_i^{(k)\prime}$$

with attention weights

$$\gamma_k = \frac{\exp\big(a_{\text{graph}}(\mathbf{h}_i^{(k)\prime})\big)}{\sum_{k'} \exp\big(a_{\text{graph}}(\mathbf{h}_i^{(k')\prime})\big)},$$

Final embedding and prediction. The final representation for node i is formed by combining its temporal embedding $\mathbf{h}_i$ and relational embedding $\mathbf{h}_i^{\text{rel}}$, e.g.:

$$\mathbf{z}_i = \text{Combine}(\mathbf{h}_i, \mathbf{h}_i^{\text{rel}}),$$

Then a prediction head (such as an MLP) outputs a trend prediction:

$$\hat{y}_i = \text{softmax}\big(W_{\text{out}} \, \mathbf{z}_i + b_{\text{out}}\big).$$

4.2 STHAN-SR: Spatiotemporal Hypergraph Attention Network for Stock Ranking

4.2.1 Temporal Feature Encoding. For each stock s, on each day τ, STHAN-SR [27] computes a vector of temporal features (e.g. 1-day return, moving averages over windows 5, 10, 20, 30 days), which is denoted by

$$\mathbf{q}_\tau \in \mathbb{R}^d, \quad \tau = t - T, \ldots, t - 1.$$

STHAN-SR feeds $\{\mathbf{q}_\tau\}$ into an LSTM to obtain hidden states

$$\mathbf{h}_\tau = \text{LSTM}(\mathbf{q}_\tau, \mathbf{h}_{\tau-1}), \quad \tau = t - T, \ldots, t - 1.$$

STHAN-SR defines a temporal attention mechanism $\zeta(\cdot)$ to aggregate the hidden states

$$h = \zeta\big(\{\mathbf{h}_\tau\}\big) = \sum_\tau \beta_\tau \, \mathbf{h}_\tau,$$

where

$$\beta_\tau = \frac{\exp\big(\mathbf{h}_\tau^\top W \, \mathbf{h}\big)}{\sum_{\tau'} \exp\big(\mathbf{h}_{\tau'}^\top W \, \mathbf{h}\big)}.$$

To better reflect excitations over time (events influencing future returns), STHAN-SR incorporates a Hawkes attention mechanism:

$$\mathbf{z} = \sum_\tau \Big(\lambda_\tau + \varepsilon \cdot \max(\lambda_\tau, 0) \, e^{-\gamma \Delta t_\tau}\Big),$$

where Δt_τ is time difference between current day t and day τ, ε is an excitation parameter, γ is a decay parameter, and $\lambda_\tau = \beta_\tau \, \mathbf{h}_\tau$. Hence $\mathbf{z} \in \mathbb{R}^d$ is the final temporal embedding of each stock.

4.2.2 Spatial Modeling via Hypergraph. STHAN-SR defines a hypergraph Laplacian operator:

$$\Delta = I - D_v^{-1/2} H \, W \, D_e^{-1} H^\top D_v^{-1/2}.$$

Then the hypergraph convolution (one layer) updates node features $X^{(l)} \in \mathbb{R}^{N \times F^{(l)}}$ to

$$X^{(l+1)} = \text{ELU}\big(D_v^{-1/2} H W D_e^{-1} H^\top D_v^{-1/2} \, X^{(l)} \, P\big),$$

where $P \in \mathbb{R}^{F^{(l)} \times F^{(l+1)}}$ is a learnable weight matrix. To adaptively weight each hyperedge per node, STHAN-SR defines attention coefficients

$$\alpha_{ie} = \text{softmax}_{e \in \mathcal{N}(i)}\big(\text{LeakyReLU}(\mathbf{a}^\top [P\mathbf{x}_i \parallel P\mathbf{x}_e])\big),$$

where $\parallel$ denotes concatenation, $\mathcal{N}(i)$ are hyperedges incident to node i, and $\mathbf{x}_e$ is some aggregate embedding of hyperedge e.

4.2.3 Joint Spatiotemporal Fusion and Ranking Loss. After processing, each stock i has: 1- Temporal embedding $\mathbf{z}_i$ (from Hawkes attention), 2- Spatial embedding $\mathbf{u}_i$ (after hypergraph convolutions). STHAN-SR combines them by concatenation (or other fusion):

$$\mathbf{h}_i^{\text{final}} = \text{Concat}(\mathbf{z}_i, \mathbf{u}_i).$$

Given predicted scores $\hat{y}_i$ for each stock, the model is trained using a ranking loss (e.g. pairwise or listwise). For instance, a pairwise hinge-like loss:

$$\mathcal{L}_{\text{rank}} = \sum_{i,j} \max\big(0, \; m - (\hat{y}_i - \hat{y}_j)\big) \cdot \mathbb{I}[r_i > r_j],$$

where m is a margin, and $\mathbb{I}[r_i > r_j]$ indicates that stock i truly outperformed j on that day.

4.3 HGTS-Former: Hierarchical HyperGraph Transformer for Multivariate Time Series

4.3.1 Patchification and Token Embedding. HGTS-Former [42] forms patches along time with window length P and stride S. Let $\mathbf{X} \in \mathbb{R}^{C \times T}$ denote a multivariate time series with C channels (variables) and T time steps. For channel c, it defines the n-th patch

$$\mathbf{p}_{c,n} \; = \; \mathbf{X}_{[c,\, t_n : t_n + P - 1]} \in \mathbb{R}^P, \quad t_n = 1 + (n-1)S, \;\; n = 1, \ldots, N,$$

where $N = \left\lfloor \frac{T-P}{S} \right\rfloor + 1$. Each patch is linearly embedded into d-dimensional tokens:

$$\mathbf{z}_{c,n} \; = \; \mathbf{E} \, \mathbf{p}_{c,n} + \mathbf{b}_e \in \mathbb{R}^d, \qquad \mathbf{E} \in \mathbb{R}^{d \times P}.$$

HGTS-Former stacks per-channel tokens as $\mathbf{Z}_c = [\mathbf{z}_{c,1}, \ldots, \mathbf{z}_{c,N}]^\top \in \mathbb{R}^{N \times d}$ and optionally adds learnable temporal positions $\mathbf{P} \in \mathbb{R}^{N \times d}$: $\widetilde{\mathbf{Z}}_c = \mathbf{Z}_c + \mathbf{P}$.

4.3.2 Temporal Multi-Head Self-Attention (MHSA). For each channel c, HGTS-Former refines temporal representations via MHSA. Let $\widetilde{\mathbf{Z}}_c \in \mathbb{R}^{N \times d}$ be inputs. For head $h = 1, \ldots, H_a$,

$$\mathbf{Q}^{(h)} = \widetilde{\mathbf{Z}}_c \mathbf{W}_Q^{(h)}, \quad \mathbf{K}^{(h)} = \widetilde{\mathbf{Z}}_c \mathbf{W}_K^{(h)}, \quad \mathbf{V}^{(h)} = \widetilde{\mathbf{Z}}_c \mathbf{W}_V^{(h)}, \quad \mathbf{W}_\star^{(h)} \in \mathbb{R}^{d \times d_h},$$

with $d_h = d/H_a$. Scaled dot-product attention per head:

$$\text{Attn}^{(h)}(\widetilde{\mathbf{Z}}_c) = \text{softmax}\left(\frac{\mathbf{Q}^{(h)} \mathbf{K}^{(h)\top}}{\sqrt{d_h}}\right) \mathbf{V}^{(h)} \in \mathbb{R}^{N \times d_h}.$$

4.3.3 Hierarchical Hypergraphs: Intra- and Inter-Layers. HGTS-Former builds a two-level (hierarchical) hypergraph to 1- aggregate intra-channel temporal patterns and 2- capture inter-channel fine-grained relations.

Vertices at the intra level. For each channel c, treat its N patch tokens as vertices $\mathcal{V}_c^{\text{intra}} = \{v_{c,1}, \dots, v_{c,N}\}$ with features $\mathbf{H}_c$ (rows are $\mathbf{h}_{c,n}^\top$).

Intra-hyperedge construction. For each anchor patch $v_{c,n}$, build a hyperedge $e_{c,n}^{\text{intra}}$ by k-NN in the feature space of $\mathbf{H}_c$:

$$e_{c,n}^{\text{intra}} \;=\; \big\{\, v_{c,m} \;:\; v_{c,m} \in \text{kNN}(\mathbf{h}_{c,n}) \,\big\}.$$

This yields an incidence matrix $\mathbf{H}_c^{\text{intra}} \in \{0,1\}^{N \times M_c}$ (often $M_c \approx N$).

Intra-hypergraph attention aggregation. Let $\mathbf{X}_c^{\text{intra}} = \mathbf{H}_c \in \mathbb{R}^{N \times d}$ be vertex features. HGTS-Former produces a hyperedge feature for e via attention-weighted pooling of member vertices:

$$\alpha_{v \to e} = \frac{\exp\big(\mathbf{a}_e^\top \tanh(\mathbf{W}_e \mathbf{x}_v)\big)}{\sum_{u \in e} \exp\big(\mathbf{a}_e^\top \tanh(\mathbf{W}_e \mathbf{x}_u)\big)}, \quad v \in e, \tag{1}$$

$$\mathbf{y}_e = \sum_{v \in e} \alpha_{v \to e}\, \mathbf{W}_v \mathbf{x}_v, \qquad \mathbf{W}_e, \mathbf{W}_v \in \mathbb{R}^{d \times d},\;\; \mathbf{a}_e \in \mathbb{R}^d. \tag{2}$$

Stacking $\mathbf{y}_e$ gives $\mathbf{Y}_c^{\text{intra}} \in \mathbb{R}^{M_c \times d}$.

Edge-to-Node (E2N) projection (intra). HGTS-Former projects hyperedge features back to vertices with normalized incidence:

$$\mathbf{Z}_c^{\text{intra}} \;=\; \sigma\big(\mathbf{D}_v^{-1}\, \mathbf{H}_c^{\text{intra}}\, \mathbf{W}^{\text{intra}}\, \mathbf{D}_e^{-1}\, \mathbf{Y}_c^{\text{intra}}\, \mathbf{W}_{\text{intra}}^{\text{proj}}\big) \;\in\; \mathbb{R}^{N \times d}, \tag{3}$$

with trainable $\mathbf{W}^{\text{intra}}, \mathbf{W}_{\text{intra}}^{\text{proj}} \in \mathbb{R}^{d \times d}$ and nonlinearity σ. Finally, it fuses with residual:

$$\widetilde{\mathbf{H}}_c \;=\; \mathbf{H}_c \;+\; \mathbf{Z}_c^{\text{intra}}.$$

Vertices at the inter level. HGTS-Former now summarizes each channel c by a *channel token* obtained via attentive pooling over its refined patch tokens $\widetilde{\mathbf{H}}_c$:

$$\beta_n^{(c)} = \frac{\exp\big(\mathbf{a}_c^\top \tanh(\mathbf{W}_c \widetilde{\mathbf{h}}_{c,n})\big)}{\sum_{m=1}^{N} \exp\big(\mathbf{a}_c^\top \tanh(\mathbf{W}_c \widetilde{\mathbf{h}}_{c,m})\big)}, \tag{4}$$

$$\mathbf{s}_c = \sum_{n=1}^{N} \beta_n^{(c)}\, \mathbf{W}_p\, \widetilde{\mathbf{h}}_{c,n} \in \mathbb{R}^d. \tag{5}$$

It stacks channel tokens into $\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_C]^\top \in \mathbb{R}^{C \times d}$.

Inter-hyperedge construction. Build hyperedges among channels to capture fine-grained cross-variable relations. For anchor channel c, it forms

$$e_c^{\text{inter}} \;=\; \big\{\, u \in \{1, \dots, C\} \;:\; u \in \text{kNN}(\mathbf{s}_c) \,\big\},$$

yielding $\mathbf{H}^{\text{inter}} \in \{0,1\}^{C \times M}$.

Inter-hypergraph attention aggregation. Analogous to Equations 1-2, itn computes hyperedge features from member channels $u \in e$:

$$\alpha_{u \to e} = \frac{\exp\big(\mathbf{a}_e^\intercal \tanh(\mathbf{W}_e \mathbf{s}_u)\big)}{\sum_{v \in e} \exp\big(\mathbf{a}_e^\intercal \tanh(\mathbf{W}_e \mathbf{s}_v)\big)}, \tag{6}$$

$$\mathbf{y}_e = \sum_{u \in e} \alpha_{u \to e}\, \mathbf{W}_u \mathbf{s}_u, \qquad \Rightarrow\; \mathbf{Y}^{\text{inter}} \in \mathbb{R}^{M \times d}. \tag{7}$$

4.3.4 Feed-Forward Network and Readout. We broadcast the inter-level refinement back to patches if needed (e.g., via cross-attention) or proceed with a per-channel FFN to obtain task features:

$$\mathbf{G} \;=\; \text{FFN}\big(\widetilde{\mathbf{S}}\big) \;\in\; \mathbb{R}^{C \times d}.$$

For forecasting, produce $\widehat{\mathbf{Y}} = \mathbf{G}\mathbf{W}_{\text{out}} + \mathbf{b}_{\text{out}} \in \mathbb{R}^{C \times H}$. For imputation, map to $C \times T$ with a decoder $\mathcal{D}_\phi$ (e.g., transposed patching): $\widehat{\mathbf{X}} = \mathcal{D}_\phi(\mathbf{G})$.

4.4 MSHyper: Multi-Scale Hypergraph Transformer for Long-Range Forecasting

4.4.1 Input Embedding and Positional Encoding. MSHyper [36] partitions the input block into non-overlapping or possibly overlapping patches across the temporal dimension. Let the patch size be P, so the number of patches is

$$N = \left\lfloor \frac{L}{P} \right\rfloor.$$

It defines the following for each channel c and patch index $n = 1, \dots, N$:

$$\mathbf{p}_{c,n} = \mathbf{X}_{c,\,(n-1)P+1:nP} \in \mathbb{R}^P.$$

MSHyper embeds each patch via a linear projection into a d-dimensional space:

$$\mathbf{z}_{c,n} = \mathbf{E}_p\, \mathbf{p}_{c,n} + \mathbf{b}_p \in \mathbb{R}^d,$$

with $\mathbf{E}_p \in \mathbb{R}^{d \times P}$.

MSHyper stacks per-channel patch embeddings:

$$\mathbf{Z}_c = [\mathbf{z}_{c,1}, \dots, \mathbf{z}_{c,N}]^\top \in \mathbb{R}^{N \times d}.$$

MSHyper also adds positional encodings to inject temporal order:

$$\widetilde{\mathbf{Z}}_c = \mathbf{Z}_c + \mathbf{P}_{\text{pos}} \in \mathbb{R}^{N \times d},$$

where $\mathbf{P}_{\text{pos}} \in \mathbb{R}^{N \times d}$ are learned or fixed positional vectors. Then, MSHyper concatenates across channels:

$$\mathbf{Z} = \text{Concat}(\widetilde{\mathbf{Z}}_1, \dots, \widetilde{\mathbf{Z}}_C) \in \mathbb{R}^{(CN) \times d}.$$

4.4.2 Multi-Scale Hypergraph Construction. The novelty of MSHyper lies in constructing multi-scale hypergraphs to capture both short- and long-term dependencies over the token graph. Let there be S scales indexed by $s = 1, \dots, S$. For scale s, MSHyper builds a hypergraph

$$\mathcal{G}^{(s)} = (\mathcal{V}, \mathcal{E}^{(s)}, \mathbf{W}^{(s)}, \mathbf{H}^{(s)}),$$

where the vertex set $\mathcal{V}$ is the set of all patch tokens (size CN), $\mathcal{E}^{(s)}$ is the set of hyperedges at scale s, $\mathbf{W}^{(s)}$ is the diagonal hyperedge-weight matrix, and $\mathbf{H}^{(s)} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{E}^{(s)}|}$ is the incidence matrix.

The process to build $\mathcal{E}^{(s)}$ is as follows:

(1) Anchor selection: MSHyper selects a subset of tokens (e.g. every k_s-th token, or via clustering) as anchor vertices for hyperedges at scale s.
(2) Hyperedge membership: For each anchor token v, MSHyper collects its top-k neighbors in feature space to form a hyperedge $e^{(s)}$.
(3) Incidence & weights: MSHyper assigns incidence $H_{v,e}^{(s)} = 1$ if $v \in e$, else 0.

As a result, we have S hypergraphs capturing relations at multiple scales, from local to global. MSHyper defines vertex and hyperedge degrees per scale as follows:

$$\mathbf{D}_v^{(s)} = \operatorname{diag}\left(\mathbf{H}^{(s)}\,\mathbf{W}^{(s)}\,\mathbf{1}\right), \quad \mathbf{D}_e^{(s)} = \operatorname{diag}\left(\mathbf{H}^{(s)\top}\,\mathbf{1}\right).$$

4.4.3 Hypergraph Transformer Layer. Each layer of the model fuses token features via hypergraph attention at multiple scales, along with standard transformer-style cross-scale mixing and residual connections. Suppose the input features at layer ℓ are

$$\mathbf{X}^{(\ell)} \in \mathbb{R}^{(CN)\times d}.$$

For scale s, MSHyper defines hyperedge aggregation followed by Edge-to-Node (E2N) projection:

(1) Vertex-to-Hyperedge (V2E) aggregation with attention: For each hyperedge e, we compute:

$$\alpha_{v\to e}^{(s)} = \frac{\exp\left(\mathbf{a}^{(s)\top}\,\tanh\left(\mathbf{W}_q^{(s)}\,\mathbf{x}_v^{(\ell)}\right)\right)}{\sum_{u\in e}\exp\left(\mathbf{a}^{(s)\top}\,\tanh\left(\mathbf{W}_q^{(s)}\,\mathbf{x}_u^{(\ell)}\right)\right)}, \quad v \in e.$$

Then hyperedge embedding:

$$\mathbf{y}_e^{(s)} = \sum_{v\in e} \alpha_{v\to e}^{(s)}\,\mathbf{W}_v^{(s)}\,\mathbf{x}_v^{(\ell)}.$$

Stack all hyperedges to get $\mathbf{Y}^{(s)} \in \mathbb{R}^{|\mathcal{E}^{(s)}|\times d}$.

(2) Edge-to-Node propagation: Project hyperedge embeddings back to vertices:

$$\mathbf{Z}^{(s)} = \sigma\left((\mathbf{D}_v^{(s)})^{-1}\,\mathbf{H}^{(s)}\,\mathbf{W}^{(s)}\,(\mathbf{D}_e^{(s)})^{-1}\,\mathbf{Y}^{(s)}\,\mathbf{W}_{\text{E2N}}^{(s)}\right) \quad \in \mathbb{R}^{(CN)\times d},$$

with trainable $\mathbf{W}_{\text{E2N}}^{(s)} \in \mathbb{R}^{d\times d}$ and nonlinearity $\sigma(\cdot)$.

Then, MSHyper fuses across all S scales (plus optionally direct self-attention interaction) via weighted sum:

$$\widetilde{\mathbf{X}}^{(\ell)} = \mathbf{X}^{(\ell)} + \sum_{s=1}^{S} \lambda^{(s)}\,\mathbf{Z}^{(s)},$$

where $\lambda^{(s)}$ are learnable or fixed scale weights. Next, as in transformer architectures, MSHyper applies a feed-forward network (FFN) with residual:

$$\mathbf{X}^{(\ell+1)} = \widetilde{\mathbf{X}}^{(\ell)} + \text{FFN}\left(\text{LayerNorm}(\widetilde{\mathbf{X}}^{(\ell)})\right).$$

MSHyper stacks L such layers to obtain the final representation $\mathbf{X}^{(L)}$.

4.4.4 Long-Range Forecasting Head. After L transformer layers, MSHyper obtains token-level features:

$$\mathbf{X}^{(L)} \in \mathbb{R}^{(CN)\times d}.$$

In this case, MSHyper can perform de-patching or aggregation to convert back to temporal predictions. For example: 1- One can decode per channel, such as splitting $\mathbf{X}^{(L)}$ into channel blocks $\mathbf{X}_c^{(L)} \in \mathbb{R}^{N\times d}$, 2- One can apply a linear decoder (or transposed convolution) to map patch features back to H time-step predictions:

$$\widehat{\mathbf{Y}}_c = \mathbf{X}_c^{(L)}\,\mathbf{W}_{\text{dec}} \in \mathbb{R}^{N\times H},$$

then concatenate across channels to produce $\widehat{\mathbf{Y}} \in \mathbb{R}^{C\times H}$.

5 Experimental Setup

5.1 Dataset

We compile a real-world dataset for S&P 500 stocks by using yfinance. The dataset is shown in Table 1. The dataset consists of three key components:

(1) **Sector Sequence Data:** We select a total of 109 categories(10 sectors, 25 industry groups, and 74 industries) based on the GICS Industry classification. For each sector, we retrieve its index over a 3-year period. Using the definition provided in Section 3, we compute the multi-scale Relative Price Strength (RPS) sequences for these sectors.

(2) **Stock Sequence Data:** For all S&P 500 stocks, we collect daily trading indicators, including opening price, highest price, lowest price, closing price, trading volume, and trading amount. Each indicator for each stock is normalized by dividing by its maximum value over the entire observed trading period to ensure scale consistency.

(3) **Stock Relationship Data:** We represent sector-level relationships using the sector hypergraph $\mathcal{G}_z = (\mathcal{S}, \mathcal{E}_z, \boldsymbol{w}_z)$, described by the incidence matrix $\boldsymbol{H}_z \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{E}_z|}$, where $|\mathcal{S}| = 467$ (stocks) and $|\mathcal{E}_z| = 109$ (sectors). An entry $h(v, e) = 1$ indicates that stock v belongs to the sector represented by hyperedge e.

Property	Data
Train Period	2021/11/01-2023/04/25
Validation Period	2023/04/26-2023/10/24
Test Period	2023/10/25-2024/04/25
#Days (Train, Validation, Test)	362, 121, 121
#Stocks (Nodes)	497
#Sectors (Hyperedges in $\mathcal{G}_z$)	109
#Falling Days/Rising/Steady Days	35%:37%:28%

Table 1: The dataset statistics present both the temporal characteristics and the structural details of stock hypergraphs.

5.2 Baselines

We evaluate the effectiveness of the methods by benchmarking it against a range of baseline approaches, organized into two main categories:

(1) Price-based models: CNN [34] leverages convolutional layers to extract spatial features from stock price sequences. LSTM [22] employs a two-layer long short-term memory architecture to capture the temporal dynamics of sequential stock data. Adv-ALSTM [12], an enhanced LSTM variant, integrates a temporal attention aggregation module to emphasize important time steps.

(2) Graph-oriented models: GCN [6] represents the spatial dependencies among stocks by modeling them as graph-structured data. TGC [13] introduces temporal graph convolutions to capture both relational and sequential patterns across companies. HATS [18] applies hierarchical attention to selectively aggregate heterogeneous relationship information.

5.3 Evaluation Metrics

The criteria used to label stocks as declining ((-1)), increasing ((1)), or remaining stable ((0)) on day (t+1) are specified as follows:

$$y = \begin{cases} -1, & \frac{(p_{t+1}-p_t)}{p_t} \leq \eta_{\text{falling}} \\ 1, & \frac{(p_{t+1}-p_t)}{p_t} \geq \eta_{\text{rising}} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where p_t denotes the closing price on the t-th trading day, and the thresholds η_{falling} and η_{rising} are fixed at -0.50% and 0.55%, respectively. The effectiveness of the classification is assessed using the metrics Accuracy, Precision, Recall, and F1 score, which are defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{10}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{11}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{12}$$

where TP, TN, FP, and FN means true positive, true negative, false positive, and false negative sample counts, respectively.

6 Results

6.1 Direction Prediction Performance

As presented in Table 2, hypergraph neural network-based HGTS-Former attains the highest performance across all evaluation metrics, achieving 62.58%, 59.46%, 60.14%, and 59.79% in accuracy, precision, recall, and F1-score, respectively. These results surpass those of the second-best approach HGTAN, which is also a hierarchical hypergraph neural network-based approach. This superior performance arises from the model's ability to represent stock interrelations through a hypergraph structure and to exploit the sector rotation phenomenon in the stock market. Specifically, both approaches first capture the temporal dynamics of individual stocks, then model them at the group (sector) level using a hypergraph hierarchical attention mechanism. This mechanism adaptively learns optimal pathways for information propagation by assessing the relative importance of stock nodes, hyperedges, and hypergraphs, thereby uncovering higher-order dependencies among stocks. Among the baselines, HATS outperforms the rest of the baselines, as it is also a hypergraph-based approach. The hierarchical attention appears to be important in achieving better performance. These findings provide compelling evidence that explicitly capturing and leveraging the sector rotation effect substantially enhances the accuracy and robustness of stock movement prediction.

As illustrated in Table 2, the results indicate that, on average, graph-based methods outperform models that rely exclusively on stock sequence data (e.g., CNN, LSTM, Adv-ALSTM), thereby providing empirical evidence for the advantage of integrating stock relationship information in stock movement prediction. However, the inclusion of graph learning does not guarantee superior performance across all evaluation metrics. For instance, Adv-ALSTM attains a precision of 52.02%, which is 0.5% higher than that of GCN. This can be attributed to the fact that GCN models stock relations only at the pairwise level, resulting in an excessively large and dense graph structure that introduces noise. Consequently, its predictive capacity becomes less effective compared to recurrent neural networks, which are inherently better suited for modeling sequential data.

The compared hypergraph neural networks surpass all baseline models, demonstrating the superiority of representing stock relationships through hypergraphs rather than traditional graphs. When a sector or a fund includes n stocks, a conventional graph requires C_n^2 edges to capture all pairwise relationships, whereas a hypergraph can represent the same configuration with a single hyperedge. This eliminates the need to construct edges between every pair of stock nodes, thereby enabling a more efficient propagation of stock price movement signals across the hypergraph.

6.2 Financial Evaluation

During the backtesting process, we compute 4 financial metrics: 1- Annual Return (AR): AR calculates the overall portfolio's value change between years $y-1$ to y. 2- Volatility: Volatility is the standard deviation of portfolio returns over the test period. 3- Sharpe Ratio (SR): SR measures portfolio excess returns relative to the risk-free rate divided by the volatility, which measures the portfolio's risk. 4- Maximum Drawdown (MDD): MDD calculates the loss of a portfolio for a given time period $[t, T], t < T$. In our study, MDD is measured relative to the portfolio's peak value. An equal amount of capital is allocated to each stock predicted to increase in price the following day, with purchases made at the closing price. If any currently held stock is forecasted to decline, it is sold at the closing price; otherwise, it remains in the portfolio. At the conclusion of the backtesting period, all remaining positions are liquidated, and the final IRR is determined. To highlight the benefits of employing neural network-based stock movement prediction for trading decisions, we compare the results against a buy-and-hold baseline strategy. In this benchmark approach, all stocks are purchased at the start of the backtesting period and held without adjustment until the end, at which point they are sold to assess overall performance.

Table 3 illustrates the comparative profitability of various methods, revealing that all evaluated models surpasses the buy-and-hold benchmark. This confirms the capability of deep neural networks to produce effective trading signals for stock prediction. Among the hypergraph neural network-based approaches, similar to direction prediction, HGTS-Former and HGTAN outperform the rest of the methods. For instance, HGTAN achieves an annualized return rate of 12.55%, markedly exceeding that of competing approaches. In particular, it outperforms the second-best method by 1.30%, yielding returns more than four times higher than those from the buy-and-hold strategy. These findings highlight the value of incorporating sector knowledge to enhance stock movement prediction performance. Furthermore, models such as CNN and LSTM deliver comparable yet inferior results. Notably, methods employing hypergraph-based spatial modeling consistently outperform those using conventional graph structures, indicating that hypergraphs are more effective in capturing higher-order relationships among multiple stocks in real-world markets and facilitate better representation learning of stock nodes.

Method	Accuracy	Precision	Recall	F1 Score
CNN	46.86 ± 0.25	45.47 ± 0.45	48.96 ± 0.45	47.15 ± 1.83
LSTM	48.75 ± 0.51	47.32 ± 0.92	47.13 ± 0.72	47.22 ± 1.83
Adv-ALSTM	53.76 ± 0.31	53.02 ± 0.20	48.90 ± 0.53	50.87 ± 0.74
GCN	54.34 ± 0.73	52.52 ± 0.35	50.86 ± 0.94	51.67 ± 0.85
TGC	54.84 ± 0.21	52.74 ± 0.67	51.03 ± 0.63	51.87 ± 1.33
HATS	55.10 ± 0.84	53.71 ± 0.71	51.21 ± 0.93	52.43 ± 0.89
HGTAN	59.48 ± 0.36	57.26 ± 0.64	54.34 ± 0.26	55.76 ± 1.73
STHAN-SR	55.21 ± 0.53	52.27 ± 0.81	53.44 ± 0.19	53.00 ± 0.98
HGTS-Former	**62.58 ± 0.39**	**59.46 ± 0.55**	**60.14 ± 0.33**	**59.79 ± 1.29**
MSHyper	57.99 ± 0.50	56.55 ± 0.65	55.50 ± 0.38	56.07 ± 1.11

Table 2: Performance of hypergraph neural networks for stock prediction against baseline techniques. Boldface formatting indicates the best results. This notation is consistently applied in all subsequent tables.

Method	Annualized Return	Annualized Volatility	Sharpe Ratio	Max Drawdown
CNN	8.71%	6.14%	1.41	**−29.17%**
LSTM	8.44%	7.01%	1.20	−34.12%
Adv-ALSTM	8.02%	7.15%	1.12	−33.33%
GCN	8.88%	7.14%	1.24	−31.09%
TGC	10.10%	7.81%	1.29	−35.55%
HATS	10.69%	7.40%	1.44	−33.10%
HGTAN	**12.55%**	6.55%	**1.91**	−30.21%
STHAN-SR	10.98%	7.61%	1.44	−32.45%
HGTS-Former	11.25%	**6.44%**	1.74	−29.98%
MSHyper	10.68%	7.51%	1.42	−34.31%

Table 3: Financial performance of hypergraph neural networks for stock prediction against baseline techniques. Boldface formatting indicates the best results.

6.3 Ablation Study

We also analyze the impact of running approaches without sector data and fund holdings data. For all methods, using hypergraphs made up of both sector and fund holdings data outperformed the scenarios without any of the data. The findings highlight that integrating both sector-level and individual stock trends significantly enhances the accuracy of stock movement prediction. The outcomes further reveal that employing hypergraphs enables the capture of higher-order dependencies among stocks, which encode more comprehensive information than traditional graph structures and consequently improve predictive performance.

7 Conclusion

In this study, we introduced and systematically evaluated hypergraph-based neural network frameworks for stock movement prediction, emphasizing the importance of modeling higher-order dependencies among stocks and incorporating the sector rotation effect. By representing stock relationships as heterogeneous hypergraphs that capture both sector-level and fund-holding associations, our approach provides a more expressive and compact relational representation than traditional graph structures. Experimental results on real-world S&P 500 data demonstrate that HGTS-Former and HGTAN methods, which utilize hypergraph hierarchical attention, consistently outperform state-of-the-art baselines across all evaluation metrics, achieving substantial gains in accuracy, precision, recall, and F1-score. Moreover, the simulated trading analysis confirmed that hypergraph neural network approaches exceed both conventional graph-based methods and price-only models, and significantly outperform the benchmark buy-and-hold strategy. These results collectively verify that hypergraph hierarchical attention enables efficient propagation of stock price movement signals while effectively leveraging multi-stock and sector-level interactions. In future work, new approaches can be developed to incorporate dynamic hypergraph learning, allowing relationships among stocks to evolve over time, and integrate multi-modal signals such as financial news or sentiment embeddings.

Acknowledgments

This conference publication and conference attendance are supported by TUBITAK (Scientific and Technological Research Council of Turkey) 2224-A program.

References

[1] Khalid Alkhatib, Huthaifa Khazaleh, Hamzah Ali Alkhazaleh, Anas Ratib Alsoud, and Laith Abualigah. 2022. A new stock price forecasting method using active deep learning approach. *Journal of Open Innovation: Technology, Market, and Complexity* 8, 2 (2022), 96.

[2] Gary Ang and Ee-Peng Lim. 2024. Temporal Implicit Multimodal Networks for Investment and Risk Management. *ACM Transactions on Intelligent Systems and*

Technology 15, 2 (2024), 1–25.

[3] Ranjeeta Bisoi and Pradipta K Dash. 2014. A hybrid evolutionary dynamic neural network for stock market trend analysis and prediction using unscented Kalman filter. *Applied Soft Computing* 19 (2014), 41–56.

[4] Stefano Boccaletti, Pietro De Lellis, CI Del Genio, Karin Alfaro-Bittner, Regino Criado, Sarika Jalan, and Miguel Romance. 2023. The structure and dynamics of networks with higher order interactions. *Physics Reports* 1018 (2023), 1–64.

[5] Selinay Cetin and Emre Sefer. 2025. A Graphlet-based Explanation Generator for Graph Neural Networks Over Biological Datasets. *Current Bioinformatics* 20, 9 (2025), 840–851.

[6] Yingmei Chen, Zhongyu Wei, and Xuanjing Huang. 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.

[7] Zirui Chen, Xin Wang, Chenxu Wang, and Jianxin Li. 2022. Explainable link prediction in knowledge hypergraphs. In *Proceedings of the 31st ACM international conference on information & knowledge management*. 262–271.

[8] Hyejung Chung and Kyung-shik Shin. 2020. Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction. *Neural Computing and Applications* 32, 12 (2020), 7897–7914.

[9] Chaoran Cui, Xiaojie Li, Chunyun Zhang, Weili Guan, and Meng Wang. 2023. Temporal-relational hypergraph tri-attention networks for stock trend prediction. *Pattern recognition* 143 (2023), 109759.

[10] Zehra Erva Ergun and Emre Sefer. 2025. FinSentiment: Predicting Financial Sentiment Through Transfer Learning. *Intelligent Systems in Accounting, Finance and Management* 32, 3 (2025), e70015.

[11] Eugene F Fama. 1965. The behavior of stock-market prices. *The journal of Business* 38, 1 (1965), 34–105.

[12] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. 2019. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 1–30.

[13] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3558–3565.

[14] Jianliang Gao, Xiaoting Ying, Cong Xu, Jianxin Wang, Shichao Zhang, and Zhao Li. 2021. Graph-based stock recommendation by time-aware relational attention network. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16, 1 (2021), 1–21.

[15] Qiang Gao, Xinzhu Zhou, Li Huang, Kunpeng Zhang, Siyuan Liu, and Fan Zhou. 2024. Relational Fusion-based Stock Selection with Neural Recursive Ordinary Differential Equation Networks. *Information Fusion* 110 (2024), 102468.

[16] Yi-Ling Hsu, Yu-Che Tsai, and Cheng-Te Li. 2021. Fingat: Financial graph attention networks for recommending top-k k profitable stocks. *IEEE transactions on knowledge and data engineering* 35, 1 (2021), 469–481.

[17] Zhenni Jin and Kun Guo. 2021. The dynamic relationship between stock market and macroeconomy at sectoral level: evidence from Chinese and US stock market. *Complexity* 2021, 1 (2021), 6645570.

[18] Raehyun Kim, Chan Ho So, Minbyul Jeong, Sanghoon Lee, Jinkyu Kim, and Jaewoo Kang. 2019. Hats: A hierarchical graph attention network for stock movement prediction. *arXiv preprint arXiv:1908.07999* (2019).

[19] Mahinda Mailagaha Kumbure, Christoph Lohrmann, Pasi Luukka, and Jari Porras. 2022. Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems with Applications* 197 (2022), 116659.

[20] Zengyu Lei, Caiming Zhang, Yunyang Xu, and Xuemei Li. 2024. DR-GAT: Dynamic routing graph attention network for stock recommendation. *Information Sciences* 654 (2024), 119833.

[21] Youru Li, Zhenfeng Zhu, Xiaobo Guo, Shaoshuai Li, Yuchen Yang, and Yao Zhao. 2023. HGV4Risk: Hierarchical global view-guided sequence representation learning for risk prediction. *ACM Transactions on Knowledge Discovery from Data* 18, 1 (2023), 1–21.

[22] David MQ Nelson, Adriano CM Pereira, and Renato A De Oliveira. 2017. Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)*. Ieee, 1419–1426.

[23] Robert Patro, Geet Duggal, Emre Sefer, Hao Wang, Darya Filippova, and Carl Kingsford. 2012. The missing models: a data-driven approach for learning how networks grow. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Beijing, China) *(KDD '12)*. Association for Computing Machinery, New York, NY, USA, 42–50. doi:10.1145/2339530.2339541

[24] Rob Patro, Emre Sefer, Justin Malin, Guillaume Marçais, Saket Navlakha, and Carl Kingsford. 2011. Parsimonious Reconstruction of Network Evolution. In *Algorithms in Bioinformatics*, Teresa M. Przytycka and Marie-France Sagot (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 237–249.

[25] Hao Qian, Hongting Zhou, Qian Zhao, Hao Chen, Hongxiang Yao, Jingwei Wang, Ziqi Liu, Fei Yu, Zhiqiang Zhang, and Jun Zhou. 2024. Mdgnn: Multi-relational dynamic graph neural network for comprehensive and dynamic stock investment prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 14642–14650.

[26] Perry Sadorsky. 2021. A random forests approach to predicting clean energy stock prices. *Journal of Risk and Financial Management* 14, 2 (2021), 48.

[27] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, Tyler Derr, and Rajiv Ratn Shah. 2021. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 497–504.

[28] Emre Sefer. 2021. Hi–C interaction graph analysis reveals the impact of histone modifications in chromatin shape. *Applied Network Science* 6, 1 (2021), 54.

[29] Emre Sefer. 2022. BioCode: A Data-Driven Procedure to Learn the Growth of Biological Networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 19, 6 (2022), 3103–3113.

[30] Emre Sefer. 2022. ProbC: joint modeling of epigenome and transcriptome effects in 3D genome. *BMC genomics* 23, 1 (2022), 287.

[31] Emre Sefer. 2025. Anomaly Detection via Graph Contrastive Learning. In *Proceedings of the 2025 SIAM International Conference on Data Mining (SDM)*. SIAM, 51–60.

[32] Emre Sefer and Carl Kingsford. 2011. Metric Labeling and Semi-metric Embedding for Protein Annotation Prediction. In *Research in Computational Molecular Biology*, Vineet Bafna and S. Cenk Sahinalp (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 392–407.

[33] Emre Sefer and Carl Kingsford. 2021. Metric Labeling and Semimetric Embedding for Protein Annotation Prediction. *Journal of Computational Biology* 28, 5 (2021), 514–525. doi:10.1089/cmb.2020.0425 arXiv:https://doi.org/10.1089/cmb.2020.0425 PMID: 33370163.

[34] Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. 2017. Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*. IEEE, 1643–1647.

[35] Betul Seyhan and Emre Sefer. 2023. NFT primary sale price and secondary sale prediction via deep learning. In *Proceedings of the Fourth ACM International Conference on AI in Finance*. 116–123.

[36] Zongjiang Shang and Ling Chen. 2024. Mshyper: Multi-scale hypergraph transformer for long-range time series forecasting. *arXiv preprint arXiv:2401.09261* (2024).

[37] Necla N Soylu and Emre Sefer. 2024. DeepPTM: protein post-translational modification prediction from protein sequences by combining deep protein language model with vision transformers. *Current Bioinformatics* 19, 9 (2024), 810–824.

[38] Heni Sulastri, Sheila Maulida Intani, and Rianto Rianto. 2023. Application of bagging and particle swarm optimisation techniques to predict technology sector stock prices in the era of the COVID-19 pandemic using the support vector regression method. *International Journal of Computational Science and Engineering* 26, 3 (2023), 255–267.

[39] Yasin Uygun and Emre Sefer. 2025. Financial asset price prediction with graph neural network-based temporal deep learning models. *Neural Computing and Applications* (2025), 1–27.

[40] Jiawei Wang and Zhen Chen. 2024. SPCM: a machine learning approach for sentiment-based stock recommendation system. *IEEE Access* 12 (2024), 14116–14129.

[41] Jujie Wang and Shuzhou Zhu. 2023. A multi-factor two-stage deep integration model for stock price prediction based on intelligent optimization and feature clustering. *Artificial Intelligence Review* 56, 7 (2023), 7237–7262.

[42] Xiao Wang, Hao Si, Fan Zhang, Xiaoya Zhou, Dengdi Sun, Wanli Lyu, Qingquan Yang, and Jin Tang. 2025. HGTS-Former: Hierarchical HyperGraph Transformer for Multivariate Time Series Analysis. *arXiv preprint arXiv:2508.02411* (2025).

[43] Quanzhou Xiang, Zhan Chen, Qi Sun, and Rujun Jiang. 2024. RSAP-DFM: regime-shifting adaptive posterior dynamic factor model for stock returns prediction. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*. 6116–6124.

[44] Pei-Yi Yang, Zhao-Rong Lai, Xiaotian Wu, and Liangda Fang. 2018. Trend representation based log-density regularization system for portfolio optimization. *Pattern Recognition* 76 (2018), 14–24.

[45] Yanlin Yang, Xuemei Hu, and Huifeng Jiang. 2022. Group penalized logistic regressions predict up and down trends for stock prices. *The North American Journal of Economics and Finance* 59 (2022), 101564.

[46] Kyung Keun Yun, Sang Won Yoon, and Daehan Won. 2021. Prediction of stock price direction using a hybrid GA-XGBoost algorithm with a three-stage feature engineering process. *Expert Systems with Applications* 186 (2021), 115716.

[47] Borong Zou, Hong Wang, Hui Li, Ling Li, and Yuhan Zhao. 2022. Predicting stock index movement using twin support vector machine as an integral part of enterprise system. *Systems Research and Behavioral Science* 39, 3 (2022), 428–439.