# Stock market price forecasting using evolving graph neural network*

Mykola Korablyov[1,†], Oleksandr Fomichov[1,†] , Igor Kobzev[2,†], Danylo Antonov[1,*,†], and Oleksandr Tkachuk[1,†]

[1] *Kharkiv National University of Radio Electronics, Kharkiv 61166, Ukraine*
[2] *Simon Kuznets Kharkiv National University of Economics, Kharkiv 61166, Ukraine*

**Abstract**

Predicting stock prices is essential to inform investment decisions in the financial market. Analyzing financial market movements and stock price behavior is extremely complex due to the dynamic, nonlinear, non-stationary, non-parametric, and chaotic markets. Various approaches are used to analyze stocks for financial market forecasting purposes. Traditional methods based on time series information for one company's stocks do not consider the relationships between stocks of other companies, which can improve the efficiency of stock price forecasting. The use of graph neural networks for these purposes, in which the relationships of time series are represented as a relationship graph structure, and the variables are defined as graph nodes, significantly improves forecasting accuracy. Existing forecasting methods usually assume that the structure of the relationships graph, which is described by the relationships matrix and determines the aggregation method of the graph neural network, is fixed by definition. Therefore, they cannot effectively consider dynamic changes in relationship graphs. In this paper, an evolving graph neural network is proposed for forecasting stock prices in the stock market. To extract dynamic correlations between price movements in financial time series, a relationships graph is constructed in the form of clusters, the generation and the evolution of the structure and parameters of which are implemented using a dendritic artificial immune network (DaiNet). For each generated cluster of the relationships graph, the price encoding is performed using transformers to determine the price information. Then, the messages from the relational graph structure and the input time sequences are aggregated based on the use of the attention layer of the time graph. At the last GNN layer, the final prediction of the future price movement of each stock is performed using a multilayer perceptron to integrate the components.

**Keywords**

stock, financial market, forecasting, profit, interaction, relationships graph, evolution, graph neural network, artificial immune network [1]

## 1. Introduction

Multivariate time series (MTS) modeling plays an important role in modern intelligent systems. By modeling the evolution of states or events in the future, forecasting enables decision-making and plays an important role in many practical areas, such as finance, transportation, healthcare, etc. However, accurate forecasting of MTS is still a challenging task due to the possible presence of hidden changing correlations both within and between time series.

The ability to predict stock prices is essential to inform investment decisions in the financial market. Financial markets determine the interactions between companies and investors and have a significant impact on many areas of human activity, such as business, education, technology, etc. Financial market analysis indicates that stock prices are inherently volatile, which makes it difficult to predict their movements. At the same time, analyzing financial market movements and stock price

*International Workshop on Computational Intelligence, co-located with the IV International Scientific Symposium "Intelligent Solutions" (IntSol-2025), May 01-05, 2025, Kyiv-Uzhhorod, Ukraine
1,* Corresponding author.
mykola.korablyov@nure.ua (M. Korablyov); oleksandr.fomichov@nure.ua (O. Fomichov); ikobzev12@gmail.com (I. Kobzev); danylo.antonov@nure.ua (D. Antonov); oleksandr.tkachuk@nure.ua (O. Tkachuk)
0009-0005-2540-7741 (M. Korablyov); 0000-0001-9273-9862 (O. Fomichov); 0000-0002-7182-5814 (I. Kobzev); 0009-0000-2079-3413 (D. Antonov); 0009-0006-2943-9887 (O. Tkachuk)

behavior is a complex task due to the dynamic, nonlinear, non-stationary, non-parametric, and chaotic nature of the markets.

Various approaches are used to solve the problem of forecasting the financial market: statistical methods, pattern recognition, graphs, machine learning, etc. Traditional learning methods consider time series of stock changes as independent and equally distributed relative to each other, which does not coincide with the real situation in the financial market. For example, two stocks in the same sector may have a higher correlation than stocks in different areas. The effect of linkage in the financial market, where stock prices are affected by the prices of related stocks, requires the use of more complete data. Taking into account the relationships between stocks can improve the efficiency of stock price forecasting.

In the classical statistical field, the autoregressive model (AR) and its variants are the most popular forecasting methods due to their efficiency and ideal mathematical properties. However, they are mainly applied to univariate forecasting problems and assume a linear relationship between variables. With the rapid growth of data volume, AR models perform poorly in forecasting in more complex settings due to their relatively low expressiveness.

Multivariate time series forecasting examines the correlation between variables. Deep learning methods are used to handle nonlinear dependencies in this area. Some of the first deep learning-based multivariate time series forecasting models were LSTNet [1] and TPA-LSTM [2], which combine a convolutional neural network (CNN) and a recurrent neural network (RNN) to extract both intra- and intertemporal dependencies.

Recently, graph neural networks have been effectively used for multivariate time series forecasting, in which time series interactions are represented as a relationship graph structure, and variables are represented as graph nodes [3, 4, 5]. Existing forecasting methods usually assume that the relationship graph structure, which is described by the adjacency matrix and determines the aggregation method of the graph neural network, is fixed (static) by definition and is either built manually by an expert, or by natural language processing (NLP), or is self-trained.

Creating entity relationship graphs is a challenging task because they are ambiguous and dynamically changing. For example, financial knowledge graphs mainly include the basic business and investment relations of entities that are labeled by domain experts or extracted from unstructured texts. However, different experts have different knowledge, which can lead to different entity relationship graphs, which significantly affects the accuracy of prediction.

Using NLP methods for forecasting faces significant challenges in extracting relations with high accuracy. That is, the relationships may be distorted by either one-sided text news or inaccurate extraction models. In addition, these relationships may change dynamically over time. Existing stock price forecasting methods based on graph learning are suboptimal under dynamically changing situations. Thus, stock price forecasting methods that assume a fixed structure of the relationship graph have low forecasting accuracy and are resource-intensive.

In real applications, the interactions of variables are dynamic and evolutionary. In addition, the interactions of time series at different time scales may also be different. Therefore, forecasting using a static relationship graph may lead to significant biases, since correlations change over time in real MTS data.

As the graph structure changes over time, it changes across different observation scales. For example, the correlation between variables in the short term may differ from the long term. For example, in finance, two stocks may rise and fall simultaneously in the short term due to changing policies, but they may diverge in the long term if one company is thriving while the other is on the verge of bankruptcy. A fixed relationship matrix cannot handle these changes.

To make the graph neural network a flexible and practical structure, it is necessary to model evolutionary and multi-scale interactions of time series in it. In this way, it is possible to simultaneously capture pairwise correlations and time dependence.

Therefore, developing a stock market price forecasting model based on an evolving graph neural network that considers changing relationships between stocks is an urgent task that can improve the accuracy of stock price forecasting.

## 2. Analysis of existing research

Methods for forecasting multivariate time series can be divided into two categories: a) methods based on implicit dependence; and b) methods based on structural dependence. In the first category of methods, one of the representative methods of which is LSTNet [1], the dependence of variables is fixed using a convolution over variables. In the second category of methods, the correlation of variables is represented as relationship graphs (RG), and forecasting is implemented using graph neural networks (GNN).

As is known, various financial market factors significantly impact changes in stock prices [6]. In existing studies, the most common approach is the manual construction of various factors as input functions [7, 8]. Thus, in [9], market data is integrated with fundamental and technical stock indicators for decision-making. In [10], a connection is established between news information and related objects for forecasting stock price movements.

To learn the sequential hidden features of historical information and then apply them to the forecasting problem [11], most existing methods use recurrent neural networks and their variants, such as LSTM and GRU [12]. However, the market data for each stock is processed independently, which does not take into account the internal relationships between stocks, leading to low forecast accuracy and low performance. In [13], the correlation information of stocks is used as input to the forecasting model, but the dynamic changes in the relationships between stocks are not automatically recorded.

In [14], it is noted that changes in stock prices are associated not only with their historical prices but also with related stocks of other companies, and knowledge graphs are used to store and represent these relationships. For efficient learning on graphically structured data, [15] proposes to use a graph neural network (GNN), which has shown high performance in various areas. In [16], company relationships are modeled in knowledge graphs, and graph convolution networks are used to predict stock price changes. In [17], a graph attention neural network is used to learn company correlations. It should be noted that GNNs built using these methods are limited to fixed, predefined company relationships that are maintained by manual editing or natural language processing methods [18].

In [19], Graph Wavenet (GWN) is used to learn the correlation of time series variables using a static representation of nodes and capture the temporal pattern using convolutional neural networks (CNN). In [19], the static correlation of time series variables is also learned, but a new graph convolution module is proposed for MTS prediction. The dynamic relation between variables in [20] is modeled using self-attention mechanisms, but the constructed GNNs are very sensitive to the input data, which leads to significant variance in prediction.

In [21], a hybrid model of stock analysis is proposed that uses a combination of different intelligent technologies: recurrent neural networks (RNN) to obtain stock price characteristics, artificial immune systems (AIS) to obtain information about the relationship between stocks, and graph neural networks (GNN) to estimate stock returns. But the structure of the generated relationship graph is also fixed.

Some papers study graph structures that change over time [22] However, these graph structures do not directly follow the forecasting problem, which may lead to biased results. Papers [23] use transformers to forecast MTS, but they do not fully consider the issues of the dynamic dependence of variables. However, in real conditions, corporate relations change over time, and the relationship graphs that describe them may be heterogeneous, that is, several types of relations may exist between companies.

Thus, existing methods do not allow full use of all the information from real graphs of company relationships. This work aims to construct an evolving graph neural model for analyzing financial market shares that would integrate information about changes in stock prices over time and the relationships between them, and would improve the accuracy of their forecasting.

## 3. Architecture of the proposed temporal graph neural network

Let us consider the architecture of the proposed temporal graph neural network and its components, presented in Figure 1, which is used to predict stock prices in the financial market.

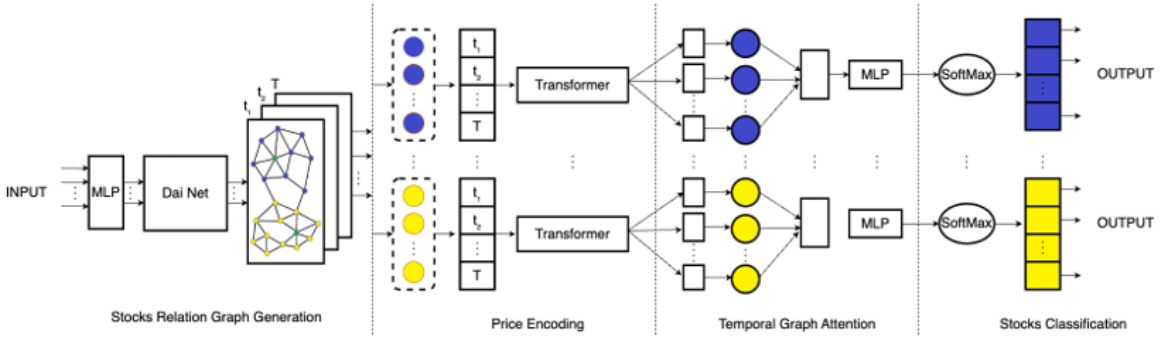


**Figure 1:** Architecture of a Temporal Graph Neural Network for Stock Price Forecasting.

First, based on the incoming input time series of stock trading characteristics, a graph of relationships between company stocks is generated in clusters, which determines the dynamic relations in the market for each trading day. Then, prices are encoded using transformers that select the central time nodes of clusters and their neighboring nodes to determine price information for each generated cluster of the relationship graph. Next, aggregation of messages from the relationship graph structure and input time sequences is performed based on the use of a time graph attention layer that adaptively calculates the importance of neighbors and aggregates information by the importance of neighbors.

In the last layer of GNN, a final prediction of the future price movement of each stock is made using a multilayer perceptron to integrate the above components. Using fully connected layers allows the model to aggregate the extracted features of the time series, which simplifies the prediction process. A final layer with a SoftMax activation function is used to output the probabilities of predicting the future movements of each stock.

Let us consider the implementation of these stages in more detail.

### 3.1. Generation and evolution of the relationships graph between the company's shares

We will design a GNN to extract dynamic correlations between variables from time series to solve the problem of stock price forecasting. Time series in financial market analysis are chronological sets of observations, such as daily sales results and stock prices of companies. We will use the sets of observations to generate time graphs of company relations for each trading day.

The inputs of the MLP are time series of trading characteristics of stocks $X=\left(X^{(1)}, X^{(2)}, \ldots, X^{(t)}, \ldots, X^{(T)}\right)$, $X \in R^{N\,xTxn}$, where $X^{(t)}=\left(x_1^{(t)}, x_2^{(t)}, \ldots, x_n^{(t)}\right)$ – time series of characteristics of one stock on trading day $t$, $N$ – number of stocks, $T$ – length of time series, $n$ – stock dimension. At the outputs of the MLP, we obtain characteristics (representations) of the stock prices of each time series $P_t^i=\left(p_1^i, p_2^i, \ldots, p_t^i, \ldots, p_T^i\right)$, $i=\overline{1,N}$ for the trading day $t$

$$P_t^i=\Theta_t\left(X_t^i\right), \tag{1}$$

where $\Theta_t$ – transformation function.

The MLP output $P_t^i$ is then used to determine the relations between stocks for each trading day $t$ in the form of time graphs, which are represented by relationship matrices

$$A_t^i = £_t(P_t^i), \tag{2}$$

Where $£_t$ – a function of forming relationship matrices.

Thus, the time graphs for each trading day are represented by a set of interconnected nodes and edges $G=\{V, E\}$ with timestamps, where $V$ and $E$ indicate the set of nodes and edges, respectively. An edge $e_{ij} \in E$ of the graph can be represented by an ordered tuple $\{v_i, v_j\}$, which indicates the edge points from node $v_i$ to node $v_{j,}$ and $M_i$ indicates the number of neighboring nodes connected to node $v_j$ that are included in the relationships graph on trading day *t*.

The company relationships graphs are viewed as sets of time graphs that are automatically generated based on historical price sequences, and in which nodes $V=\{V_{t_1}, V_{t_2}, \ldots, V_T\}$ denote companies and edges $E=\{E_{t_1}, E_{t_2}, \ldots, E_T\}$ represent their relations. Each node $V_{t_i}$ on different trading days $t_i$ can be connected by relations with several edges $E_{t_i}$ with timestamps. In this case, the appearance of nodes $V_{t_i}$ and edges $E_{t_i}$ on different trading days $t_i$ can be different.

As noted, the relationships between graph variables not only evolve over time but also change across time scales, making it difficult to describe such correlations using a fixed relationship matrix. In addition, the evolving patterns of graph structure are also not the same across time scales. Thus, there are problems to be solved:

1. The structure of the relationships graph changes over time. Most existing works use a fixed and static adjacency matrix from start to finish, which obviously cannot cope with such a condition.
2. The structure of the relationships graph changes across different observation scales. The correlation between variables in the short term may differ from the long term. For example, in the short term, two stocks may rise and fall simultaneously when financial policy changes, but in the long term, they may diverge if one company is thriving while the other is about to go bankrupt. Using a fixed relationship matrix will not cope with this condition either.

To record correlations between variables for certain time intervals, we will generate the structure of an evolving relationships graph. The task of determining the relationships between company shares can be considered a clustering task, which provides a timely and adequate way to obtain correlations between shares for each trading day. Clustering methods that use biological principles of computing organization have become widespread, among which artificial immune systems can be distinguished.

To solve the problem of stock clustering, it is proposed to use the dendritic artificial immune network (DaiNet) model [24], which allows forming a *K*-connected graph. The graph vertices are antibodies describing the characteristics of the company's stock prices, and the edges are affinity, determining the degree of connectivity between them. Formation of daiNet is a multi-stage optimization process aimed at reducing the number of connections between antibodies and using the values of affinities and avidities for this [24]. The result of clustering will be a network of antibodies with certain clusters.

Initially, the antibody network is formed as a graph, where each node is connected to all other nodes of this graph. The threshold affinity value *NAT* (Natural Affinity Threshold) is used as a criterion regulating the number of affinity bonds between antibodies, which is the average affinity between all antibodies in the population

$$NAT(AB) = \frac{\sum_{i=1}^{n} aff(ab_i, ab_j)}{n(n-1)}, \tag{3}$$

where $n$ is the number of antibodies in the population; $aff(ab_i, ab_j)$ is the affinity value between the $i$-th and $j$-th antibodies

$$aff_{ij} = (1 + d_{ij})^{-1}, \tag{4}$$

where $d_{ij}$ is the Euclidean or Manhattan distance between the features of the $i$-th and $j$-th immune objects.

According to this, the links between antibodies are removed if their affinities do not exceed the $NAT$ value. Then, the stimulation levels $s_i$ of each antibody are calculated based on its affinities with other immune objects that form the network

$$s_i = \frac{1}{K} \sum_{j=1}^{K} aff(ab_i, ab_j), \tag{5}$$

based on which the centers of the graph clusters are determined.

After the distribution of cluster centers, the process of determining the belonging of antibodies of the immune network to them occurs using avidity, which is based on the relation and affinity between immune objects

$$av_i = \sum_{j=1}^{m} aff(ab_i, ab_j), \tag{6}$$

where $av_i$ is the avidity value of the $i$-th antibody with other antibodies of the cluster; $m$ is the number of antibodies in the cluster; $aff(ab_i, ab_j)$ is the affinity value between the $i$-th and $j$-th antibodies of the same cluster by (4).

Thus, DaiNet allows for automating and simplifying the process of forming a relationship graph between company shares. The result of clustering a dendritic artificial immune network will be a network of antibodies with certain clusters, which for trading day t are represented by an evolving learning graph (EGL), described by the relation matrix $A^{(t)}$. The parameters of the relation matrix $A^{(t+1)}$ for the trading day ($t$+1) depend on the changes in the trading characteristics of the stocks $X^{(t)}$ and the parameters of the matrix $A^{(t)}$ for the trading day $t$

$$A^{(t+1)} = Q(A^{(t)}, X^{(t)}), \tag{7}$$

where $Q$ is a function that determines the evolution of GNN.

It should be noted that correlations between multivariate time series in practical problems change over time, therefore, the characteristics of nodes and arcs of the graph of company stock relations are subject to changes when moving from one trading day to another. Therefore, for GNN to have evolutionary properties that take into account changing correlations both within and between time series of company stocks, it is necessary to generate corresponding relationship matrices A^((t)) for a sequence of trading days, which will improve the accuracy of forecasting.

### 3.2. Stock price encoding

From each generated cluster of the relationship graph G, the cluster centers and their neighboring nodes are selected, and the price movement information is encoded into time representations using a converter-encoder. The input characteristic of the movement of price sequences of shares on a trading day $t$ is defined as $X = \left(X^{(1)}, X^{(2)}, \ldots, X^{(t)}, \ldots, X^{(T)}\right)$, $X \in R^{N x T x n}$, where $N$ is the number of shares, $n$ is the dimension of the share, $T$ is the number of trading days. To obtain the input tensor $H^{(t)} \in R^{NxTxn}$ of shares, we use a linear transformation for trading characteristics

$$H_i^{(t)} = F_i X^{(t)} + a_i, \tag{8}$$

where $F_i \in R^{Nxn}$ and $a_i \in R^N$ are the learning parameters.

Thus, when encoding prices in the relationship graph, the time centers of clusters and their neighboring nodes are selected for encoding price information.

### 3.3. Using the time graph attention mechanism

Given the output sequences of encoder nodes $V_i^{(t)}$ and the time graph of relations $G^{(t)}=\{V^{(t)}, E^{(t)}\}$ for each trading day $t$, the graph attention mechanism can be used to aggregate messages on successive inputs. To aggregate messages from relation graph structures and input time sequences, embedding smoothing of all nodes is performed, and a two-stage temporal attention mechanism is used. The two-stage temporal attention layers adaptively calculate the importance of neighbors, aggregate messages according to their importance, and simultaneously aggregate messages from all relation graph clusters. Message aggregation is implemented as follows

$$S_r^{(t)} = AL_r \cdot P_{0,r}, \tag{9}$$

where $S_r^{(t)} \in R^{nxN}$ is the output of the attention layer of the time graph on the trading day $t$, $P_{0,r}$ is the output projection matrix, $AL_r$ is the vertex of the attention layer of the time graph, which is described as follows

$$AL_r = \sum_j \sigma \cdot \left( \sum_i \left( p_{i,j} \cdot h_i \right) \right), \tag{10}$$

where σ is the sigmoid activation function, $h_i$ is the i-th row of the price embedding, $p_{i,j}$ is the importance of the time boundary (i,j) at the i-th vertex of the graph, which is determined using the ReLU activation function.

Thus, using a graph attention layer allows adaptive computation of the importance of neighbors in a relationship graph and aggregating information according to their importance.

### 3.4. Probabilistic stock price forecasting

The final layer of the GNN, using multilayer perception with a SoftMax output activation function, allows for a probabilistic forecast of the future price movement of each stock. The SoftMax activation function is the key element of the prediction layer. The SoftMax function is defined as follows: each output variable value is an exponential function of the input variable value divided by the sum of the exponentials of all input variable values. This transformation allows the model to express the output variable values as a probability distribution between different variables. Applying this function in a fully connected layer allows the model to predict output values that are interpreted as a probability, where each value represents the model's confidence that the output value belongs to a particular stock category. Moreover, using cross-entropy as a loss function in such problems allows the model to optimize its weights in such a way as to maximize the predicted probabilities of the actual labels of the resulting classes. This provides more accurate and reliable forecasting of stock prices in the financial market using deep learning and graph neural networks.

Nodes in a fully connected layer are connected to all nodes in the previous layer, which allows for efficient combination and classification of features extracted from previous processing stages. Using fully connected layers allows the model to aggregate the extracted features of the time series, which simplifies the forecasting process.

## 4. Experimental studies

This section presents the practical implementation of an evolving graph neural network for stock price forecasting. The experiments are designed to demonstrate how dynamic graph construction and adaptive clustering can improve forecasting accuracy in a real-world financial setting.

### 4.1. Data acquisition and preprocessing

Historical data on stocks was obtained, including daily closing prices, trading volumes, and selected technical indicators covering T trading days. The data were preprocessed as follows:

- Normalization: All features were normalized using z-score normalization to ensure a comparable scale.
- Sliding Window Generation: A sliding window of 20 trading days was formed for the input time series. This approach provides each sample with sufficient historical context for temporal graph construction.
- Feature Engineering: In addition to raw price data, technical indicators (e.g., moving averages, RSI) were computed and incorporated into the feature set.

These steps yield a multivariate time series dataset that serves as the basis for constructing temporal graphs. Figure 2 shows graphs of changes in average monthly stock prices for 16 IT companies, and Figure 3 shows a heat map of normalized prices for the first 20-day sliding window.

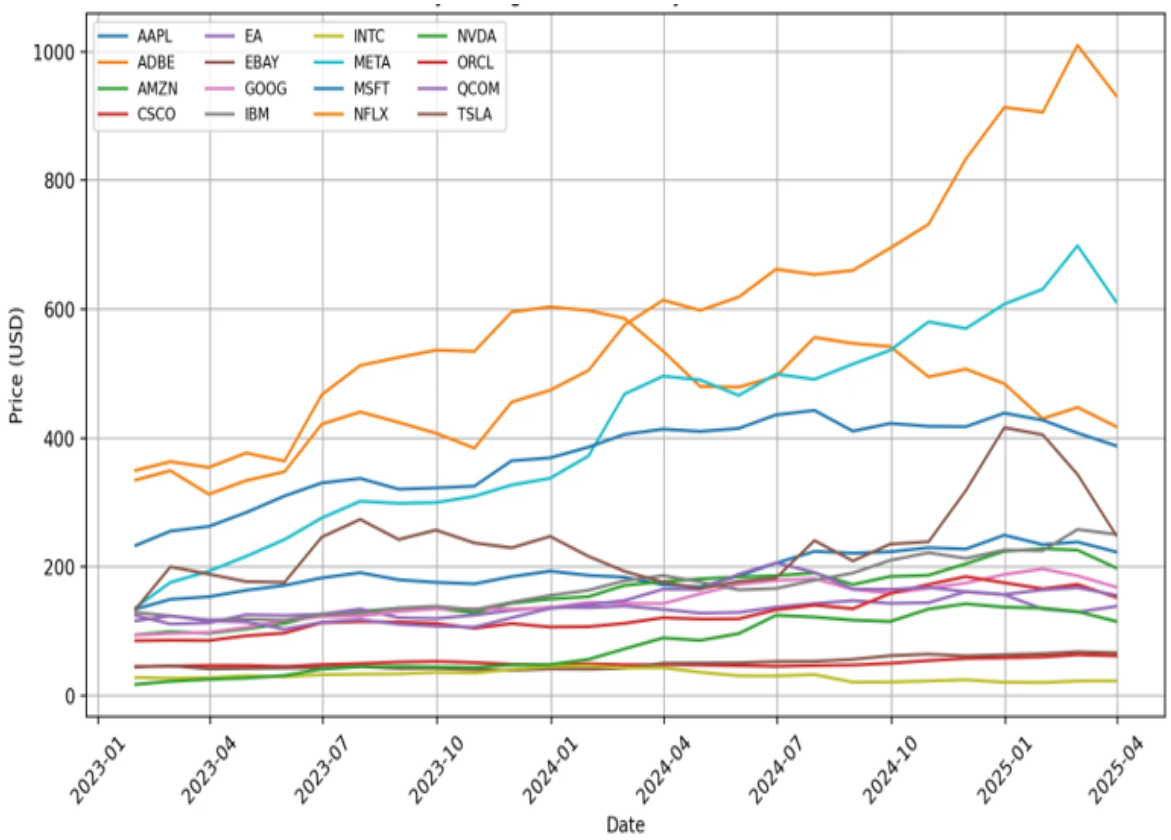


**Figure 2**: Changes in average monthly prices of company shares (Jan 2023 - Mar 2025)

Global z-score normalization is applied across the entire data range, meaning each stock's mean and standard deviation are computed from all available trading days. If the first 20 days happen to have prices consistently below the overall mean, their z-scores will naturally be negative.

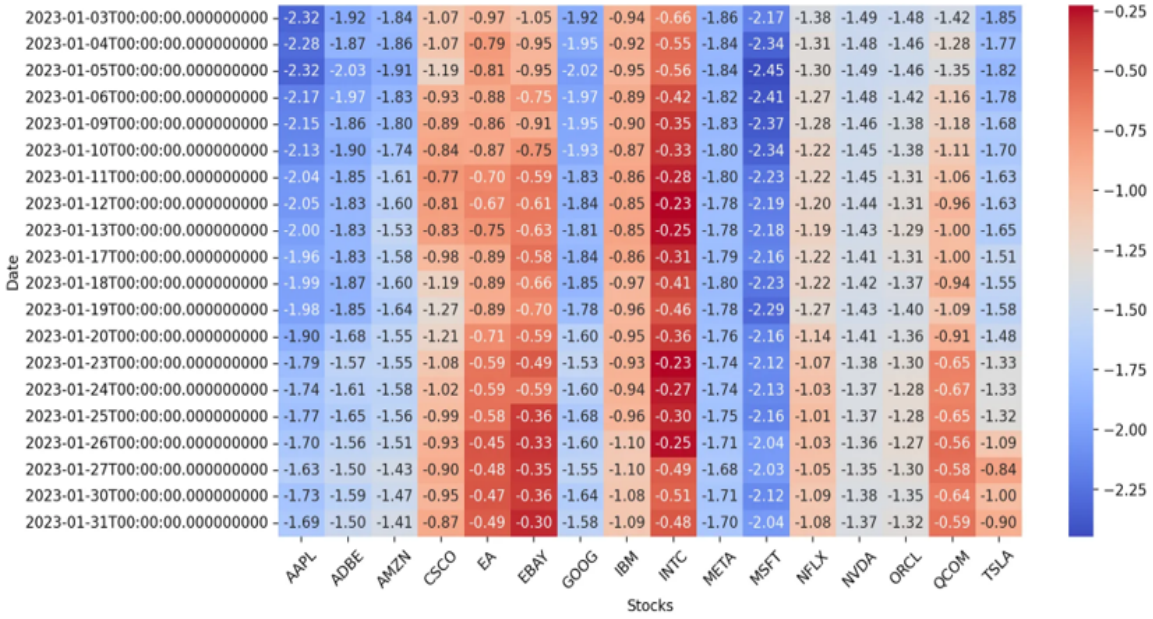


**Figure 3:** Heatmap of normalized prices for the first 20-day sliding window

### 4.2. Temporal graph construction

For each trading day, we construct a graph where:

- Each node represents an individual stock.
- The boundaries are set by calculating the affinities between historical stock price movements (in a sliding window). A threshold is applied to the affinity values to create a relation matrix. This process captures significant interdependencies between stocks, resulting in a temporal graph that evolves with market conditions.

This method ensures that the constructed graph reflects the changing relationships among stocks on a day-by-day basis.

### 4.3. Clustering using the dendritic artificial immune network

To generate the relation graph and extract meaningful substructures, we apply the daiNet algorithm to each daily graph:

- Affinity Computation: For every pair of nodes, the affinity is computed using the Manhattan distance between their feature vectors. The natural affinity threshold (NAT) is determined as the average affinity across all node pairs.
- Edge Pruning: Edges with affinity values below the NAT are pruned from the graph. This reduces noise and emphasizes stronger, more significant relationships.
- Stimulation Level and Cluster Center Identification: For each node, a stimulation level is calculated based on its remaining affinities. Nodes exhibiting high stimulation levels are designated as cluster centers.
- Avidity-Based Cluster Assignment: The remaining nodes are assigned to clusters by evaluating their avidity — quantifying the strength of association with each identified center. This iterative process results in evolving clusters that dynamically reflect the underlying inter-stock relationships.

The daiNet-based clustering effectively transforms the raw dynamic graph into a set of coherent subgraphs, each representing a cluster of closely related stocks.

### 4.4. Implementation and experimental results

The implementation was carried out using Python. Key libraries include:

- Pandas and NumPy: For data manipulation and preprocessing.
- Scikit-learn: For preliminary clustering experiments and evaluation of similarity measures.
- PyTorch Geometric: For implementing the graph neural network components.
- NetworkX and Matplotlib: For visualizing the evolving graph structures.

For each trading day, the temporal graph was constructed and clustered using the steps described above. The resulting clusters served as input for a transformer-based encoder that captured temporal price features. These features, aggregated via a temporal graph attention layer, were then passed to a multilayer perceptron (MLP) for probabilistic forecasting.

As a result, we will focus on three key metrics of the experiments over 50 epochs: learning loss (Figure 4), prediction root mean square error (RMSE) (Figure 5), and prediction accuracy (Figure 6).

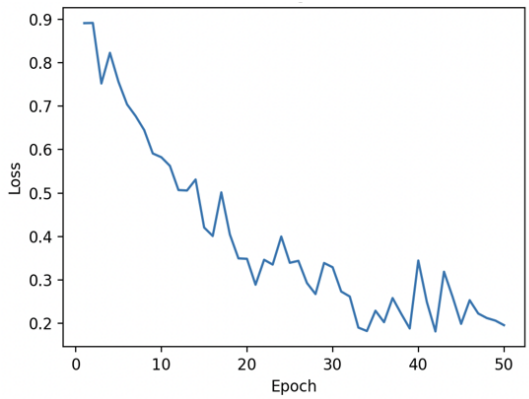


**Figure 4.** Training Lose

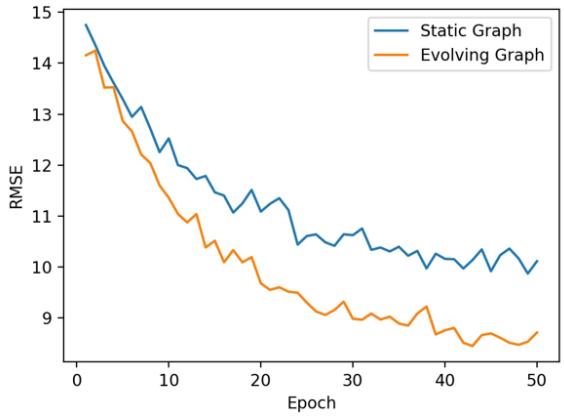


**Figure 5.** Forecast RMSE

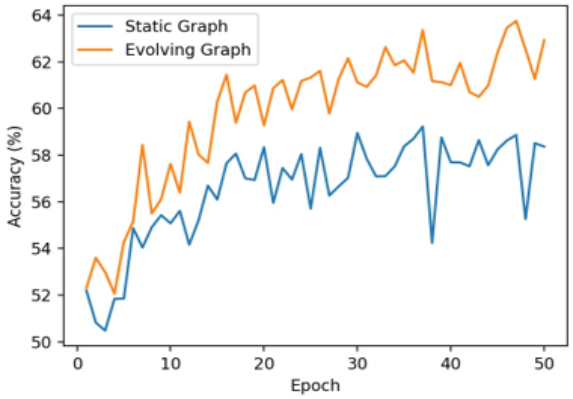


**Figure 6.** Forecast Accuracy

The evolving graph model's training loss decreases steadily from around 1.0 to approximately 0.3, with minor fluctuations indicative of normal training noise. This trend demonstrates that the model converges effectively over time.

The forecast RMSE of the evolving graph model remains below that of the static graph model throughout training, dropping from roughly 15 to around 9. The consistently lower RMSE highlights the benefit of capturing dynamic relationships via evolving graph structures.

The forecast accuracy for predicting directional movement shows that both models improve from near 50% (random guessing) to well above 55%. The evolving graph model, however, outperforms the static approach by a margin of about 3–5 percentage points at most epochs, peaking around 62%. Overall, these results underscore the advantage of incorporating dynamic clustering and time-aware graph attention for more accurate and robust stock price forecasting.

## 5. Conclusion

Today, the financial market uses combinations of different technologies, the interaction of which allows for more informed decisions. Various approaches are used to analyze stocks and forecast the financial market, but they usually do not take into account the relationships between stocks of different companies. The use of graph neural networks (GNN), in which the relationships of time series are represented as a relationship graph, significantly increases the accuracy of forecasting. However, in existing methods, the structure of the relationship graph is fixed by definition.

In this paper, an evolving graph neural network is used to forecast stock prices in the financial market, in which the generation and evolution of the structure and parameters of the graph, presented in the form of clusters, is implemented using a dendritic artificial immune network (DaiNet). The graph vertices are antibodies describing the characteristics of the company's stock prices, and the edges are affinity, determining the degree of connectivity between them. Formation of daiNet is a multi-stage optimization process aimed at reducing the number of connections between antibodies and using the values of affinities and avidities for this [24]. The result of clustering will be a network of antibodies with certain clusters.

To determine the price information for each generated cluster of the relationship graph, the prices are encoded using transformers. Based on the use of the attention layer of the time graph, information is aggregated by the importance of neighbors. The last layer of the GNN, using multilayer perception with the output activation function SoftMax, allows you to get a probabilistic forecast of the future price movement of each stock.

The conducted experimental studies showed that the use of dynamic graph construction and stock clustering based on DaiNet not only tracks the temporal nature of the relationships between stocks but also improves forecasting efficiency. To improve the characteristics of stock analysis and financial market forecasting, further research is planned on the use of evolving time-inhomogeneous GNNs.

## Declaration on Generative AI

The authors have not employed any Generative AI tools.

## References


[1] G. Lai, W.-C. Chang, Y. Yang, and H. Liu. Modeling long- and short-term temporal patterns with deep neural networks. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (2018), 95–104.

[2] S.-Y. Shih, F.-K. Sun, and H.-Y. Lee. Temporal pattern attention for multivariate time series forecasting. Machine Learning 108, 8 (2019), 1421–1441.

[3] M. Patel, K. Jariwala, and C. Chattopadhyay. A Systematic Review on GNN-based Methods for Stock Market Forecasting. ACM Computing Surveys, Vol. 57, Iss. 2, Art. No 34 (2024), 1-38.

[4] J. Wang, S. Zhang, Y. Xiao, and R. Song. A Review on Graph Neural Network Methods in Financial Applications. Journal of Data Science 20 (2), (2022), 111–134.
[5] D. Cheng, F. Yang, S. Xiang, and J. Liu. Financial time series forecasting with multi-modality graph neural network. Pattern Recognition, Vol. 121, (2022), 108218.
[6] F. Feng, H. Chen, X. He, J. Ding, M. Sun, and T.-S. Chua. Enhancing Stock Movement Prediction with Adversarial Training. In IJCAI, arXiv:1810.09936. (2019).
[7] D. Cheng, F. Yang, S. Xiang, and J. Liu. Financial time series forecasting with multi-modality graph neural network. Pattern Recognition 121 (2022), 108218.
[8] X.-Y. Liu, J. Rui, J. Gao, L. Yang, H. Yang, Z. Wang, C. D. Wang, and G. Jian. FinRL-Meta: Data-Driven Deep Reinforcement Learning in Quantitative Finance. Data-Centric AI Workshop, NeurIPS (2021).
[9] M. Ballings, D. V. Poel, N. Hespeels, and R. Gryp. Evaluating multiple classifiers for stock price direction prediction. Expert Syst. Appl. 42 (2015), 7046–7056.
[10] W. Li, R. Bao, K. Harimoto, D. Chen, J. Xu, and Q. Su. Modeling the Stock Relation with Graph Network for Overnight Stock Movement Prediction. In IJCAI (2020), 4541-4547.
[11] Z. Jin, Y.-C. Yang, and Y. Liu. Stock closing price prediction based on sentiment analysis and LSTM. Neural Computing and Applications 32 (2019), 9713–9729.
[12] K. Cho, B. V. Merrienboer, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. ArXiv abs/1406.1078 (2014).
[13] D. Cheng, F. Yang, S. Xiang, and J. Liu. Financial time series forecasting with multi-modality GNN. Pattern Recognition 121 (2022), 108218.
[14] W. Liu, Y. Zhang, J. Wang, Y. He, J. Caverlee, P. Chan, D.S. Yeung, and P.-A. Heng. Item Relationship Graph Neural Networks for E-Commerce. IEEE Transactions on Neural Networks and Learning Systems (2021), 1-15.
[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S Yu Philip. A comprehensive survey on GNN. IEEE transactions on neural networks and learning systems 32, 1 (2020), 4–24.
[16] Y. Chen, Z. Wei, and X. Huang. 2018. Incorporating Corporation Relationship via Graph Convolutional Neural Networks for Stock Price Prediction. Proceedings of the 27th ACM International Conference on Information and Knowledge Management (2018), 1655-1658.
[17] R. Sawhney, S. Agarwal, A. Wadhwa, and R. Shah. Deep Attentive Learning for Stock Movement Prediction from Social Media Text and Company Correlations. In Proceedings of the Conference on Empirical Methods in NLP (2020), 8415–8426.
[18] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems (2021).
[19] Z. Wu, S. Pan, G. Long, J. Jang, and C. Zhang. Graph wavenet for deep spatial-temporal graph modeling. arXiv preprint arXiv:1906.00121 (2019).
[20] Q. Zhang, J. Chang, G. Meng, S. Xiang, and C. Pan. Spatio-temporal graph structure learning for traffic forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34 (2020), 1177–1185.
[21] M. Korablyov, O. Fomichov, D. Antonov, S. Dykyi, I. Ivanisenko, and S. Lutskyy. A hybrid stock analysis model for financial market forecasting. IEEE Proceedings of the International Conference on Computer Science and Information Technologies (CSIT 2023), (2023), 1-4.
[22] A. Natali, A., E. Isufi, E., M. Coutiño, M., and G. Leus. Learning time-varying graphs from online data. IEEE Open Journal of Signal Processing 3 (2022), 212–228.
[23] M. Xu, W. Dai, C. Liu, X. Gao, W. Lin, G. J. Qi, and H. Xiong. Spatial-temporal transformer networks for traffic flow forecasting. ArXiv abs/2001.02908 (2020).
[24] M. Korablyov, O. Fomichov, M. Ushakov, and M Khudolei. Dendritic Artificial Immune Network Model for Computing. Proceedings of the 7th International Conference on Computational Linguistics and Intelligent Systems (CoLInS 2023). (2023), 206-217.