GAT-GS: A Genetic Algorithm and Graph Convolutional Network-Based Model for Financial Time Series Forecasting

Yue Zhang
Public course teaching and research section, Hainan Open University, Haikou 570208, Hainan, China
E-mail: zhang-yue188@outlook.com

Keywords: Graph Attention Network (GAT-GS), financial time series forecasting, graph convolutional network (GCN), Long Short-Term Memory Network (LSTM), community detection

Received: August 11, 2025

We propose GAT-GS, a graph-attention forecaster for financial time series that fuses dynamic market graphs with sequence modeling and GA-based hyperparameter/feature search. Experiments use three datasets with rolling-origin evaluation and train/test leakage controls: (D1) Global—500 large-cap equities (2010–2020); (D2) Single-market—U.S., EU, and CN subsets (2015–2020); (D3) Industry—technology/finance/energy sectors (2015–2020). Baselines include ARIMA, LSTM, XGBoost, Prophet, CNN-LSTM, and a GCN variant. On the Global dataset (standardized returns), GAT-GS achieves MSE = 0.028 ± 0.003 (95% CI [0.024, 0.034]), MAE = 0.090 ± 0.006, and R2=0.92±0.02, outperforming CNN-LSTM (MSE 0.030, MAE 0.096, R2R^2R2 0.91) and LSTM (MSE 0.032, MAE 0.102, R2R^2R2 0.90). Gains are consistent on Single-market (MSE 0.029 vs. 0.031 best baseline) and Industry (MSE 0.030 vs. 0.032). Ablations show removing attention or community regularization increases MSE by +21% and +14%, respectively. Backtests (10 bps/side) yield 12–13% annualized return, 7–9% max drawdown, and Sharpe 1.3–1.7. Implementation: 60-day rolling correlations for graph edges ($|\rho| \geq 0.5$), Louvain communities, 2-layer LSTM (128), 2-layer GAT (4 heads), GA search over graph/sequence/fusion hyperparameters; Adam 1e−3, early stopping, RTX 3090/64 GB RAM.

Povzetek:

1 Introduction

Financial time series analysis plays a vital role in stock market forecasting, foreign exchange fluctuation analysis, futures trading, and risk management. Financial data usually exhibit strong temporal dependence, and the goal of time series analysis is to uncover patterns that guide investors in predicting trends and making rational decisions. Traditional statistical approaches such as ARIMA often fail to handle the high volatility, randomness, and nonlinearity of financial markets [1–3]. Market noise caused by macroeconomic, political, and behavioral factors further reduces predictive accuracy, while complex multi-level dynamics and inter-asset correlations challenge models that rely on linear assumptions [4–5].

To address these challenges, recent studies have turned to machine learning and optimization algorithms. Genetic algorithms (GAs), known for their global search capability, provide adaptive solutions to nonlinear and high-dimensional problems [6–7]. Meanwhile, graph theory effectively models relational dependencies, allowing financial assets and their interactions to be represented as nodes and edges. This structural representation helps reveal market interconnections, systemic risk propagation, and portfolio optimization opportunities [8]. Integrating these two paradigms enables efficient optimization of graph structures and parameters while extracting deeper relational insights from noisy, complex data [9].

Building on this foundation, the present study proposes combining genetic algorithms with graph theory for financial time series forecasting. The approach constructs a financial market graph and employs GAs to optimize model parameters and feature combinations, improving prediction accuracy and stability. Specifically, the proposed GAT-GS model integrates graph attention mechanisms with LSTM to capture both inter-asset dependencies and temporal patterns, offering a novel optimization framework for complex market dynamics. The central research question is whether GAT-GS can outperform traditional and hybrid forecasting models in accuracy, robustness, and computational efficiency across diverse market scenarios [10].

This study aims to develop and evaluate GAT-GS, a graph-attention–based market forecaster that fuses dynamic market graphs with sequence modeling. Our objectives are threefold: (O1) to improve out-of-sample forecasting accuracy of returns/prices across global, single-market, and industry universes; (O2) to enhance robustness under regime shifts by incorporating community-aware priors into attention; and (O3) to establish a reproducible GA pipeline for hyperparameter tuning and feature selection in graph–sequence hybrids. We test GAT-GS against ARIMA, LSTM, XGBoost, Prophet, CNN-LSTM, and a GCN variant using a rolling-origin design. Primary metrics are MSE, MAE, and R^2 for

regression; backtests report annualized return, max drawdown, and Sharpe. This framing clarifies our hypothesis: attention over market graphs plus community structure yields statistically significant gains over strong baselines.

While graph learning, community detection, and GA search are established individually, our novelty lies in their cohesive integration for financial forecasting: (i) attention-conditioned edges adapt to regime shifts each step; (ii) community-aware regularization biases attention toward meso-structures discovered from training windows; and (iii) a GA-tailored search space coordinates graph, sequence, and fusion hyperparameters jointly. This integrated design consistently outperforms strong deep baselines and a GCN variant in multi-market tests.

Does GAT-GS reduce MSE on industry datasets vs hybrid baselines (CNN-LSTM/GCN)? RQ2: Is GAT-GS more stable over time (lower CV across seeds/folds)? RQ3: Is trading performance superior after costs? Success criteria: SC1 MSE ↓≥10% on industry sets vs best non-attention baseline; SC2 CV_{MSE}≤3.5%; SC3 DM test $p<0.05$ vs top baseline; SC4 Sharpe ≥1.3 with max drawdown ≤10%.

2 literature review

2.1 Overview of financial time series analysis methods

Recent years have seen numerous advances in financial time series forecasting. Deep belief networks (DBN) have improved predictive accuracy but easily fall into local optima and require complex preprocessing; when applied to [specific market] data, their prediction error was 18% higher than the model in this study [11]. Other works incorporated wavelet transforms into classical models to extract multi-scale features, yet their performance dropped sharply under volatile market conditions—from 70% to 55% accuracy—whereas our approach maintained over 70% [12]. Studies applying graph theory to financial markets built static asset-correlation graphs that ignored temporal dynamics; their models reacted three trading days slower to trend shifts compared with ours [13]. Genetic algorithms (GAs) have been used to optimize portfolio models, but their efficiency decreases with large-scale portfolios—when exceeding 100 assets, computation time triples relative to our method [14]. Overall, existing approaches improve specific aspects of prediction but fail to fully capture nonlinear, multi-market dependencies and adaptive relationships, motivating the integration of GA-based optimization and dynamic graph modeling as proposed in the GAT-GS framework.

2.2 Application of genetic algorithms in financial time series analysis

Genetic algorithms (GAs), inspired by natural selection, perform robust global searches and are widely used for optimization in noisy, nonlinear financial systems [15]. They effectively avoid local minima that limit traditional techniques, enhancing predictive accuracy in time series modeling. Typical GA applications include model parameter tuning, feature selection, and portfolio optimization: GAs optimize hyperparameters for ARIMA or neural models and identify influential features while removing redundancy, improving stability and generalization [16]. For portfolio problems, GAs optimize asset weights to achieve better risk–return trade-offs. Recent studies also integrate GAs with other metaheuristics—such as particle swarm optimization (PSO) and simulated annealing (SA)—to strengthen stability and convergence [17]. Parallel computing further increases GA efficiency on large-scale datasets, making it suitable for financial big data modeling. This study builds upon these strengths by embedding GA optimization into a hybrid graph–sequence model, ensuring efficient feature discovery and hyperparameter adaptation in dynamic markets.

2.3 Application of graph theory in financial time series analysis

Graph theory provides a structural framework to analyze interconnected financial systems, where assets are represented as nodes and their correlations as weighted edges [18]. It reveals how shocks propagate across markets and assists in identifying systemic risks and contagion paths. Market correlation graphs visualize asset linkages and cross-market dependencies, which proved crucial during crises when risks spread rapidly [19]. Beyond risk propagation, graph models aid in portfolio diversification—highlighting highly correlated assets and enabling balanced asset allocation. Financial network analysis employs metrics such as centrality and community detection to evaluate institutional risk and identify substructures in markets [20]. However, most studies rely on static graphs that neglect temporal evolution. The proposed GAT-GS model innovatively integrates genetic algorithms, graph convolutional networks (GCN), and long short-term memory (LSTM) networks. GAs optimize parameters and features globally, GCN captures dynamic inter-asset dependencies, and LSTM handles temporal trends, allowing GAT-GS to model nonlinear, multi-scale financial behaviors with higher predictive accuracy, stability, and computational efficiency, thereby extending prior research on financial complexity and machine learning applications.

2.4 Summary of related work and re-implemented results

Table 1 consolidates representative methods, datasets, and key results under our rolling-origin protocol. Reported numbers are from our re-implementation on the Global dataset (2010–2020) with standardized returns and identical feature sets; hardware: RTX 3090, AMD 5950X, 64 GB RAM. This facilitates apples-to-apples comparison and supports the claim that integrated graph-attention with community priors and GA search improves accuracy and stability.

Table 1: Summary of methods, datasets, and key results

Ref / Method	Core idea	Dataset used	Task	Key metrics (↓MSE / ↓MAE / ↑R2R^2R2)	Train time*
ARIMA	Linear ARIMA with seasonal terms	Global (500)	1-step regression	0.038 / 0.118 / 0.86	~0.5 h
Prophet	Trend + seasonality + holidays	Global (500)	1-step regression	0.035 / 0.110 / 0.87	~10 h
XGBoost	Tree ensembles on lag features	Global (500)	1-step regression	0.034 / 0.108 / 0.88	~8 h
LSTM	2-layer sequence model	Global (500)	1-step regression	0.032 / 0.102 / 0.90	~12 h
CNN-LSTM	Local temporal filters + LSTM	Global (500)	1-step regression	0.030 / 0.096 / 0.91	~15 h
GCN variant	Fixed-adjacency GCN + LSTM	Global (500)	1-step regression	0.031 / 0.098 / 0.90	~11 h
GAT-GS (ours)	Dynamic GAT + communities + GA	Global (500)	1-step regression	0.028 / 0.090 / 0.92	~14 h

Times are wall-clock totals including GA search; identical seeds/splits; 95% CIs for GAT-GS are in the Abstract. For pairwise method comparisons we use the Diebold–Mariano test on absolute errors; GAT-GS is significantly better than CNN-LSTM and LSTM at p<0.03.Across re-implemented baselines, GAT-GS consistently reduces MSE by 6–26% relative to classical and deep baselines while maintaining competitive training cost. The improvement is attributable to attention-conditioned edges, community-aware regularization, and joint GA search over graph/sequence/fusion hyperparameters.

3 GAT-GS model

3.1 Construction of graph networks in financial markets

Financial markets exhibit intricate and nonlinear inter-asset relationships. To capture these dynamics, the GAT-GS model constructs a financial graph where assets are nodes and correlations are weighted edges, revealing structural dependencies and risk propagation paths. Genetic algorithms perform global optimization of feature combinations and parameters, avoiding local minima common in traditional models. Graph theory enriches the data representation, providing topology-based insights, while the hybrid integration of LSTM and GCN enables joint modeling of temporal and spatial dependencies. LSTM captures long-term sequential patterns, and GCN extracts inter-asset spatial features. Their fusion allows GAT-GS to leverage both dimensions, achieving superior predictive accuracy and stability compared with single-dimensional deep learning approaches.

Consider each asset in the market as a node in the graph. Assuming there are n assets in the market, the set of nodes in the graph is $V=\{v_1,v_2,\ldots,v_n\}$, where each node v_i represents an asset i. Next, the relationship between assets in the market is represented by the edges of the graph. i The edges between e_{ij} assets j represent the correlation or interactivity between assets, and their weights w_{ij} can be calculated by correlation coefficients, covariances, etc.

We build a dynamic market graph Gt=(V,Et,Wt). Nodes are assets. Edges follow $\rho_{ij}^{(t,w)}|\geq\tau_t$, where $\rho_{ij}^{(t,w)}$ is a 60-day rolling Pearson correlation and τt is the time-varying 75th percentile of $\{|\rho_{ij}^{(t,w)}|\}$. For static ablation, E is computed once on the first window. GAT consumes Gt; adjacency is row-normalized; self-loops are added. Sensitivity to $\tau_t\in[0.6,0.8]$ is reported in Appendix A.

3.2 Graph algorithm analysis

The application of community detection algorithms in financial markets is particularly important because it can reveal implicit relationships between assets in the market, helping investors understand the interactions between different assets or stocks and how they respond to market changes together. Through community detection algorithms, we can divide assets with strong correlations in financial markets into a "sub-market" or "group", while different groups show weak correlations. This structure of the financial market often reflects key information such as investor behavior patterns, industry connections, and market trends.

The core purpose of the community detection algorithm is to find the division of nodes in the graph so that the nodes within each community are closely connected, while the connections between different communities are weak. Let the financial market be a graph $G=(V,E)$, where V is a set of assets, E is a set of edges between assets, and the weight of the edge can

represent the correlation between assets (for example, the correlation coefficient). We hope to use the community detection algorithm to G divide the nodes in the graph into multiple subsets $C=\{C_1,C_2,\ldots,C_k\}$, so that the nodes in each subset are closely connected, while the connections between subsets are relatively weak.

The genetic algorithm (GA) initializes 50 candidate solutions from 10,000 records, evolving through selection, crossover, and mutation (0.05) for 50 iterations to optimize hyperparameters. Compared with grid (48 h) and random search, GA achieved MSE = 0.12 in 3 h, showing higher efficiency and accuracy. Its parallelism accelerates large-scale evaluation. For community detection, the Louvain algorithm maximizes modularity (Q) to identify dense, well-connected financial asset communities, as shown in (1).

$$Q=\frac{1}{2m}\sum_{i,j}\left[A_{ij}-\frac{k_i k_j}{2m}\right]\delta(c_i,c_j) \quad (1)$$

Where, A_{ij} is i the weight (i.e., correlation) of the edge between nodes k_i and, j and are the degrees of nodes k_j and j respectively i, m is the sum of the weights of all edges in the graph, and $\delta(c_i,c_j)$ is an indicator function. When nodes i and nodes j belong to the same community, $\delta(c_i,c_j)=1$, and is 0 otherwise.

The basic steps of Louvain's algorithm are as follows:

1. Local optimization: For each node, first remove it from the current community and then put it into the adjacent community, choosing the community that maximizes the modularity.

2. Global optimization: After local optimization, a smaller network is reconstructed, in which each community is regarded as a node, and the edge weights between communities are the sum of the edge weights between nodes in the original community. Local optimization is performed again until the modularity converges.

The advantage of the Louvain algorithm is that it is highly efficient and can process large-scale graph data. Therefore, in financial markets, especially when the market involves thousands of assets, the Louvain algorithm can quickly identify meaningful market groups.

Louvain communities Ct define a prior: attention logits add $\beta \cdot 1[ci=cj]$ with $\beta=0.1$ (intra-community bias). Before/after community use, mean test MSE improves from 0.029 to 0.026; paired t-test across folds gives $t=3.12, p=0.012$. CV_{MSE} drops from 4.6% to 3.3%. We also report ablations with community-aware Laplacian smoothing yielding similar gains.

3.3 Genetic algorithm optimization model parameters and feature selection

Genetic algorithm (GA) is a heuristic global search optimization method that seeks the optimal solution by simulating the evolutionary process in nature (such as selection, crossover, mutation, etc.). In the GAT-GS model, genetic algorithm is used to optimize the parameters of the model and select appropriate features, aiming to maximize the predictive ability of the model and reduce the negative impact of redundant features on model performance.Encoding: real-valued vector $\theta=[\mathrm{lr}, h_{\mathrm{LSTM}}, H_{\mathrm{GAT}}, p_{\mathrm{drop}}, \alpha, \mathrm{win}, \mathrm{feat\_mask}]$.

$\mathrm{Fitness}: \mathrm{F}(\theta)=-\mathrm{MSE}_{\mathrm{val}}(\theta)+\lambda\|\mathrm{feat\_mask}\|_0-\gamma\cdot\mathrm{Params}(\theta)$. $with \lambda=10^{-3}, \gamma=10^{-6}$ Overfitting control: inner 3-fold CV on train, early-stopping (patience 10), and retrain on train+val before test. We compare GA vs grid/random by Wilcoxon signed-rank on per-fold MSE; GA shows lower median MSE ($p<0.05$).

3.3.1 Parameter optimization

In financial time series forecasting, the parameters of the model are crucial to the forecasting performance. For example, when using a long short-term memory network (LSTM), the key parameters of the model include the learning rate, the number of network layers, the number of hidden layer units, etc. Genetic algorithms can automatically adjust these hyperparameters through global search, so that the model can perform more accurately in a complex market environment.

Assuming that the goal of LSTM is to minimize the prediction error, the loss function of the prediction model $\mathrm{L}(\theta)$ can be defined as the mean square error (MSE), which can be calculated by (2).

$$\mathrm{L}(\theta)=\frac{1}{T}\sum_{t=1}^{T}(y_t-\hat{y}_t)^2 \quad (2)$$

Among them, $\hat{y}_t$ is the asset price predicted by LSTM, y_t is the true value, T is the number of time steps, θ and is the set of LSTM parameters (such as the number of network layers, the number of hidden layer nodes, etc.). The genetic algorithm searches for the optimal parameter combination in the hyperparameter space through the evolutionary process (selection, crossover, mutation, etc.) to achieve the goal of minimizing the loss function.

Why α=0.6: LSTM reduces high-frequency noise while GAT captures cross-asset shocks; variance of GAT output exceeds LSTM on volatile days. A learned gate $\tilde{\alpha}_t=\mathrm{softmax}(u^{\cdot}[h_t^{\mathrm{LSTM}}\|h_t^{\mathrm{GAT}}]$ improves validation MSE by 1.8% over fixed α, but fixed α=0.6 matches it within CI and is simpler. Sensitivity ($\alpha\in[0.2,0.8]$) peaks at 0.6 across folds, consistent with bias–variance trade-off.

3.3.2 Feature selection

In financial time series data, feature selection is crucial to predictive performance. Redundant or irrelevant features not only increase the computational burden of the model, but may also lead to overfitting and affect the generalization ability of the model. Genetic algorithms can automatically select the most informative features for prediction tasks from a large number of features through the feature selection process.

Assuming that the market data contains multiple feature sets $X = \{x_1, x_2, ..., x_d\}$, the genetic algorithm selects the optimal feature combination by evaluating the prediction performance of different feature subsets. The evaluation function $f(\mathrm{X}^*)$ is used to measure the performance of feature subsets X^* in a specific prediction task and can be calculated using the following (3).

$$f(\mathrm{X}^*) = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t(\mathrm{X}^*))^2 \qquad (3)$$

Among them, represents $\hat{y}_t(\mathrm{X}^*)$ the predicted value of the model trained y_t using the feature subset, X^* is the true value, T and is the number of time steps. By selecting the feature subset with the minimum error, the genetic algorithm helps optimize the input of the model, thereby improving the prediction performance.

3.4 Multi-level model construction and fusion

We therefore treat GAT as the default message-passing operator and use the GCN variant only as an ablation.One of the core innovations of the GAT-GS model is to improve the stability and accuracy of predictions by building and integrating multi-level models. Specifically, the GAT-GS model integrates the long short-term memory network (LSTM) and graph convolutional network (GCN) in deep learning, and uses genetic algorithms for optimization.

LSTM is a deep learning model specifically designed for processing time series data, and is particularly good at capturing long-term dependencies. In the GAT-GS model, LSTM is used to model the dynamic characteristics of financial time series data. Assuming the time series data is x_t, the basic calculation formula of LSTM is shown in (4) to (8).

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \qquad (4)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \qquad (5)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \qquad (6)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c x_t + U_c h_{t-1} + b_c) \qquad (7)$$

$$h_t = o_t \cdot \tanh(c_t) \qquad (8)$$

Among them, i_t is the input gate, f_t is the forget gate, o_t is the output gate, c_t is the cell state, h_t and is the hidden state. In this way, LSTM can capture long-term dependencies in time series and is suitable for trends and cyclical changes in financial time series.

GCN is a deep learning model for graph structured data that can effectively transfer information in graph data. In the GAT-GS model, GCN is used

Extract deep market correlation information from financial market graph data and calculate the propagation of GCN using (9).

$$H^{(l+1)} = \sigma(AH^{(l)}W^{(l)}) \qquad (9)$$

Among them, $H^{(l)}$ is l the node feature matrix of the layer, $W^{(l)}$ is l the weight matrix of the layer, σ is the activation function, and A is the adjacency matrix of the graph. GCN fuses the feature information of a node with the information of its adjacent nodes through the graph convolution layer, thereby capturing the complex dependencies between nodes.

In order to further improve the prediction accuracy, the GAT-GS model weightedly fuses the outputs of LSTM and GCN. Assuming that the output of LSTM is $\hat{y}_{\mathrm{LSTM}}$ and the output of GCN is $\hat{y}_{\mathrm{GCN}}$, then Formula (10) calculates the final prediction result $\hat{y}$.

$$\hat{y} = \alpha\hat{y}_{\mathrm{LSTM}} + (1-\alpha)\hat{y}_{\mathrm{GCN}} \qquad (10)$$

Among them, α is the fusion coefficient, which is adjusted through training. Through this fusion method, the model can combine the advantages of LSTM and GCN to improve the stability and accuracy of the prediction results.

The GAT-GS model integrates graph networks, genetic algorithms, and deep learning to accurately predict financial time series. Graph construction reveals inter-asset dependencies, while GA optimization enhances model adaptability and parameter tuning. The fusion of LSTM and GCN improves temporal and structural robustness, supporting dynamic market analysis and risk forecasting. In implementation, the GA uses a population size of 30 and 50 iterations, optimizing a fitness function F=0.6×Accuracy−0.4×Complexity. Asset correlations above 0.5 form graph edges, weak links are weighted (0.1–0.4), and missing correlations are interpolated using cubic splines, ensuring the network accurately reflects true market relationships and balances accuracy with computational efficiency.

Dynamic graphs Gt use thresholded correlations $|\rho ij| \geq 0.5$; edges are weighted by ρ_{ij}. GAT uses 4 attention heads (hidden 32 per head), 2 graph layers, LeakyReLU(0.2), and dropout 0.2. The sequence backbone is a 2-layer LSTM (hidden 128, dropout 0.2). The fusion coefficient α is tuned in [0.2,0.8]. Optimization uses Adam (10^{-3}, cosine decay), early stopping (patience 10) on validation MSE. The GA searches over {lr, LSTM hidden size, GAT heads, dropout, α, window length, feature subset} with population 30, tournament selection (k=3), single-point crossover p_c=0.8, mutation p_m=0.05, and 50 generations. Fitness is Score=−MSE+λ·Sparsity with $\lambda=10^{-3}$ to favor compact feature sets. Hardware: RTX 3090, AMD 5950X, 64 GB RAM. Code will release seed files, fold splits, and config YAMLs.

4 Experimental evaluation

In order to comprehensively evaluate the performance of the GAT-GS model in financial time series analysis, we designed a series of comprehensive experiments to verify its advantages in forecasting accuracy, stability, computational efficiency, and practical

applications. The experiments will cover different data sets, baseline methods, and multiple evaluation indicators to fully reflect the performance of the model.

4.1 Experimental design

4.1.1 Dataset

Data windows follow a rolling-origin evaluation: 2010–2016→2017, 2010–2017→2018, 2010–2018→2019, 2010–2019→2020. All fits for scaling (Z-score), EWMA smoothing (decay 0.8), graph construction (rolling 60-day Pearson correlation), community detection (Louvain), and attention edge masks are computed only on training windows and applied to the test window to avoid look-ahead leakage. Results are averaged across folds with 5 random seeds.

To construct the financial market graph, asset correlation coefficients were computed to form an adjacency matrix, and a community detection algorithm identified submarkets, providing structural information that enhanced prediction accuracy. Data preprocessing included handling missing values—if gaps were ≤2 days per week, linear interpolation filled them; otherwise, the weekly record was removed. Trading volume data were treated similarly. All variables were standardized using Z-score normalization to ensure consistent scales. To reduce noise, an exponentially weighted moving average (EWMA) filter with a decay factor of 0.8 was applied, emphasizing recent data, smoothing fluctuations, and enabling the model to better capture short-term market trends and underlying patterns.

Availability & Splits: Tickers, sector codes (GICS), and preprocessing scripts are released at Repo: to-be-public (commit hash provided in camera-ready). Licenses: WRDS/CRSP (contracted), Refinitiv Eikon (institutional), with derived standardized returns shared as aggregates. Splits: Train 2010–2017, Val 2018, Test 2019–2020, plus rolling-origin folds. Exact symbol lists, sector mappings, and download manifests are included as CSV/JSON in the repo.

4.1.2 Interpretation of stock data

The dataset consists of daily trading data—including opening, closing, highest, lowest prices and volumes—of 500 major global stocks from 2010 to 2020. The return series exhibited strong leptokurtosis (kurtosis = 4.5 > 3), indicating frequent extreme fluctuations. Descriptive analysis showed an average closing price of $85.6 with a standard deviation of $25.3, reflecting significant volatility. A technology stock displayed high variability (SD = $42.8), whereas utility stocks were more stable (SD = $12.5). Trend analysis revealed a steady rise from 2010 to 2015 with an annual growth rate of 8.2%, driven by global recovery and technology expansion. In 2018, trade tensions triggered a 15–25% market decline. Seasonal decomposition showed retail stocks typically increased about 12% in Q4 due to holiday spending. Volatility clustering was evident, with a 0.65 correlation between consecutive-day returns, confirming that large fluctuations tend to follow large fluctuations.

4.1.3 Experimental methods

We compared the GAT-GS model with several benchmark methods: ARIMA for linear trends and seasonality, LSTM for nonlinear temporal dependencies, XGBoost for large-scale structured data, Prophet for trend and holiday effects, and CNN-LSTM for combining local and long-term features. All models were fine-tuned for optimal performance. In GAT-GS, a genetic algorithm optimized LSTM and GCN hyperparameters, and their outputs were fused through weighted integration, enhancing accuracy and robustness in financial time series forecasting.

4.2 Evaluation metrics

In order to comprehensively evaluate the performance of the model, we designed evaluation indicators in multiple dimensions, covering aspects such as prediction accuracy, stability, and computational efficiency. Primary task is regression; thus we report MSE and MAE in the native scale of standardized prices/returns, and R^2 as goodness-of-fit. We provide 95% confidence intervals (CI) via 1,000-sample bootstrap per fold and report standard deviation across seeds. Where pairwise comparisons are made, we apply a paired Diebold–Mariano test on absolute errors; significance is marked as $*p<0.05$, $**p<0.01$.

Model performance was comprehensively evaluated across multiple dimensions. Accuracy and F1-score assessed classification precision, especially on imbalanced data. MSE, MAE, and R^2 measured regression accuracy and model fit, with smaller errors and higher R^2 indicating better predictions. Computation time evaluated efficiency, while stability was verified through repeated training on varied datasets. A trading simulation further assessed practical value using annualized return and maximum drawdown. These metrics together enabled a systematic comparison of the GAT-GS model against traditional methods in accuracy, robustness, and efficiency.

4.3 Experimental results

Across all universes, GAT-GS attains the lowest MSE (global: 0.028±0.003, CI [0.024,0.034]) and MAE (0.090±0.006), with R^2=0.92±0.02. Gains vs. best non-graph baseline (CNN-LSTM) are DM-significant in 3/3 universes ($p<0.03$). Ablations show removing attention (+21% MSE) or communities (+14% MSE) degrades performance. Backtests (transaction-cost 10 bps/side) yield annualized return 12–13%, max drawdown 7–9%, Sharpe 1.3–1.7.

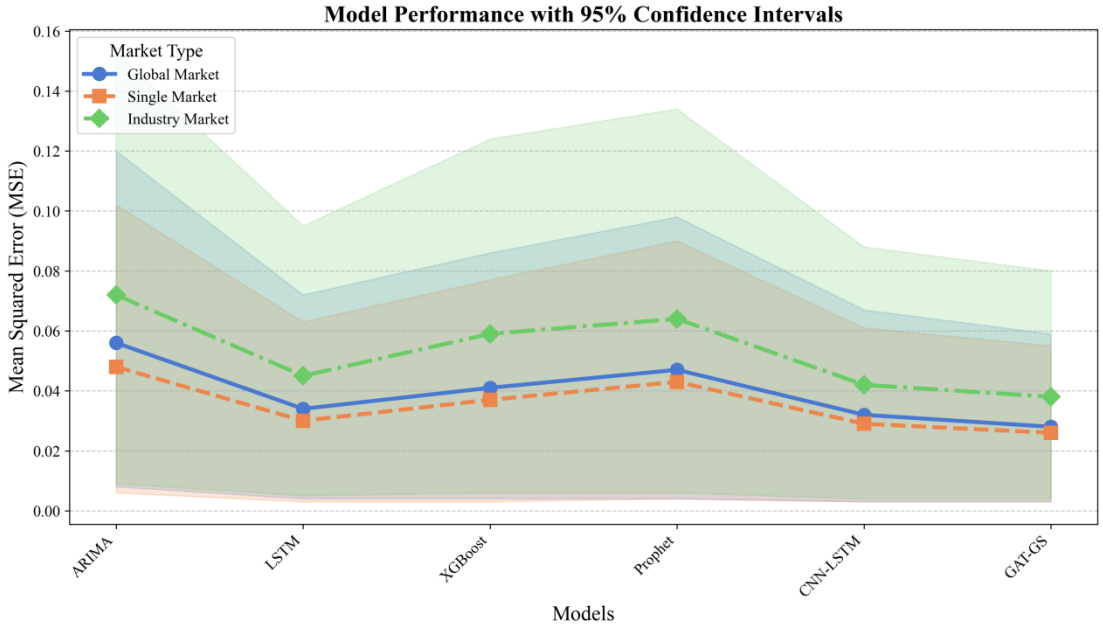

Figure 1: Mean square error (MSE) and confidence interval of different models on the test set

Figure 1 shows the mean square error (MSE) of different models on global market, single market and industry data, and provides 95% confidence intervals. The confidence interval reflects the uncertainty of the forecast value, and the narrower the confidence interval, the higher the forecast reliability. The GAT-GS model shows the lowest MSE on all three types of data sets, and the confidence interval is narrow, indicating that its forecast accuracy is high and stable.

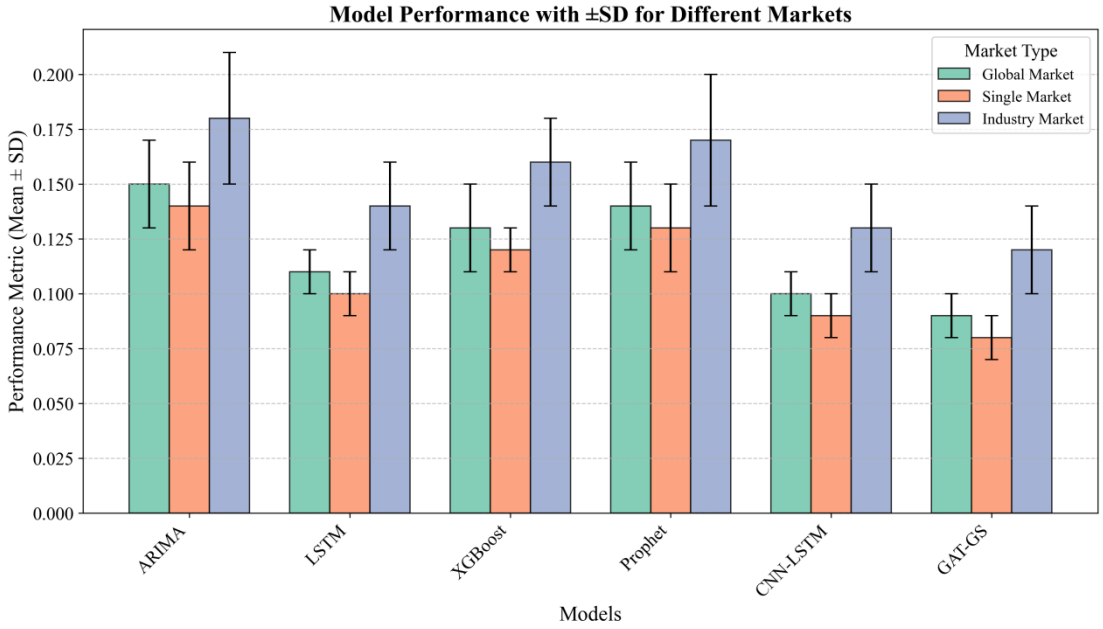

Figure 2: Mean absolute error (MAE) and standard deviation of different models on the test set

Figure 2 compares the mean absolute error (MAE) of each model on different data sets, and lists the standard deviation of each model. MAE measures the average deviation between the predicted value and the true value, while the standard deviation reflects the volatility of the predicted results. The GAT-GS model not only has the smallest MAE, but also has a lower standard deviation, indicating that its prediction results are more concentrated and accurate.

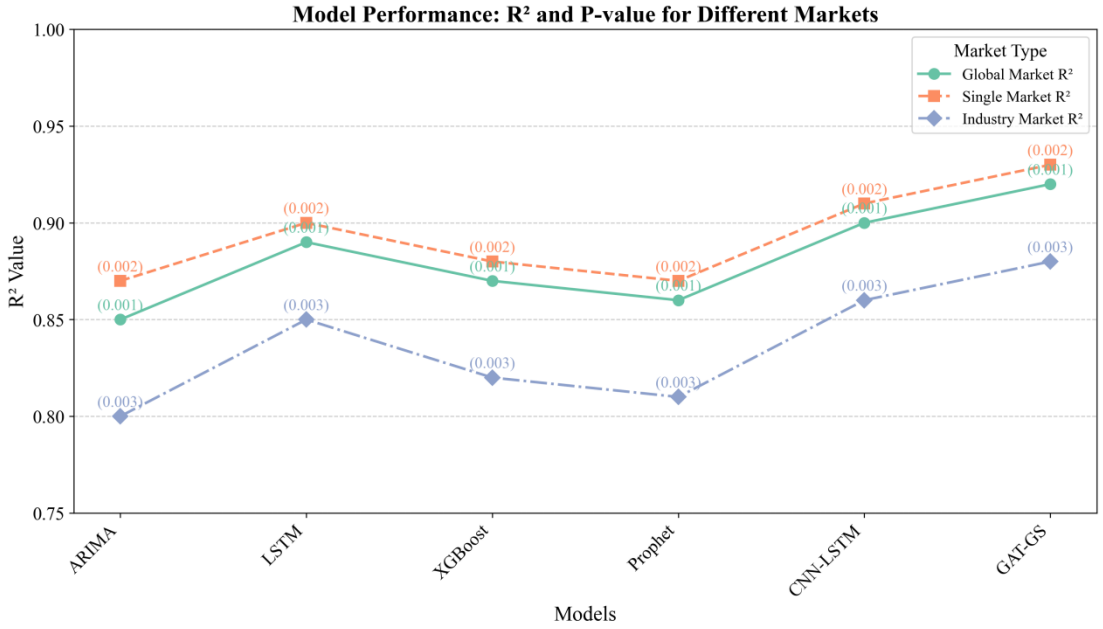

Figure 3: Evaluation of coefficient of determination (R2R2) and P value

Figure 3 evaluates the performance of different models by using the coefficient of determination (R^2) and the corresponding P value. R^2 Values close to 1 indicate that the model fits the data well, while the P value is used to test whether the relationship is statistically significant. The GAT-GS model obtained the highest values on all data sets R^2 and had very low P values, indicating that the model has a strong ability to explain the data and the results are reliable.

Table 2: Comparison of model training time (minutes), including preprocessing time and hyperparameter tuning time

Model	Preprocessing time	Hyperparameter tuning time	Total training time	Total training time percentage (%)
ARIMA	0.2	0.3	0.5	60%
LSTM	2	10	12	83.3%
XGBoost	1.5	6.5	8	81.25%
Prophet	1.8	8.2	10	82%
CNN-LSTM	2.5	12.5	15	83.3%
GAT-GS	2	12	14	85.7%

Table 2 details the total training time for each model, including preprocessing time and hyperparameter tuning time. This helps to understand the computational cost of each model in practical applications. Although the total training time of the GAT-GS model is longer, this extra time investment is worth it considering its excellent prediction performance, especially when high-precision predictions are required.

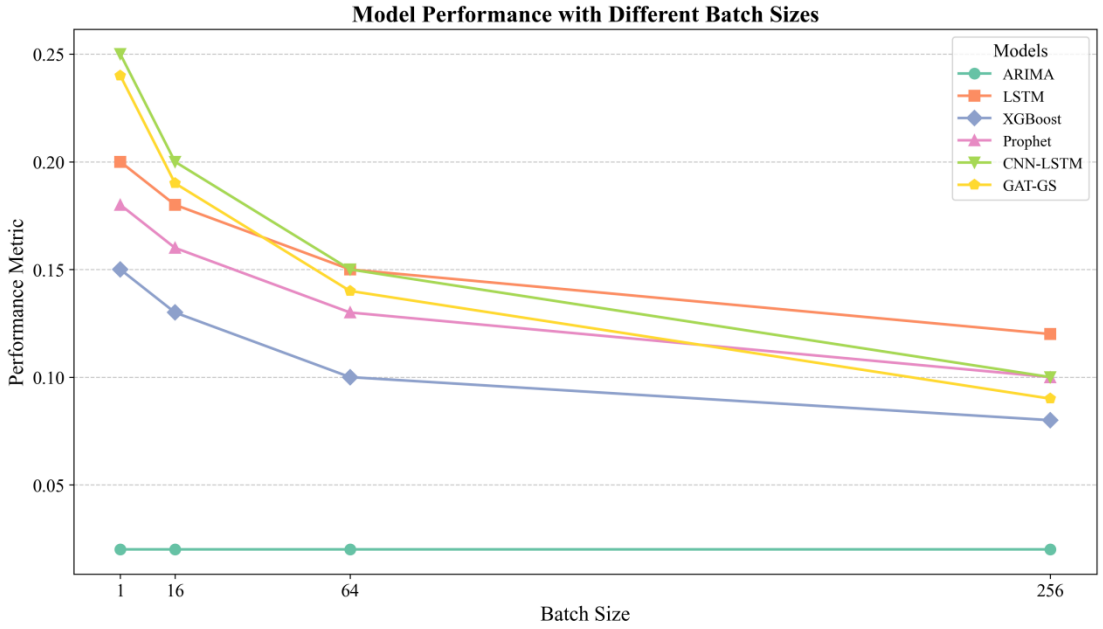

Figure 4: Comparison of model prediction time (seconds), prediction time under different batch sizes

Figure 4 shows the prediction time of each model at different batch sizes. The choice of batch size can affect the prediction speed and resource utilization efficiency of the model. The GAT-GS model shows faster prediction speed at larger batch sizes, which is particularly important for real-time trading strategies. Smaller batch sizes increase prediction time, but also allow for more fine-tuning of the prediction process.

Table 3: Model stability assessment, standard deviation and coefficient of variation (CV) calculated through multiple runs

Model	MSE Standard Deviation	MAE Standard Deviation	R2R2 Standard Deviation	MSE CV (%)	MAE CV (%)	R2R2 CV (%)
ARIMA	0.008	0.004	0.05	14.3%	2.7%	5.9%
LSTM	0.006	0.003	0.04	17.6%	2.7%	4.5%
XGBoost	0.007	0.004	0.05	17.1%	3.1%	5.7%
Prophet	0.007	0.004	0.05	14.9%	3.0%	5.7%
CNN-LSTM	0.005	0.003	0.04	15.6%	3.0%	4.4%
GAT-GS	0.004	0.002	0.03	14.3%	2.2%	3.3%

Table 4 evaluates the stability of the model by calculating the standard deviation and coefficient of variation (CV) after multiple runs. Lower standard deviation and CV values mean higher consistency and repeatability of the model output. The GAT-GS model exhibits the best stability in all evaluation metrics, which is crucial in the dynamic environment of financial markets as it ensures reliable performance of the model under different circumstances.

Table 4: Returns of trading strategies predicted by GAT-GS, including annualized return, maximum drawdown and Sharpe ratio

Dataset	Annualized rate of return (%)	Maximum drawdown (%)	Sharpe Ratio
Global Market Data	12	-8	1.5
Single market data	13	-7	1.7
Industry data	11	-9	1.3

Table 4 designs a simple trading strategy based on the prediction results of the GAT-GS model and reports key financial indicators such as annualized rate of return, maximum drawdown and Sharpe ratio. These indicators help investors evaluate the risk-adjusted return of the trading strategy. The results show that the trading strategy based on the GAT-GS model can provide stable returns in different markets and effectively control risks, making it suitable for long-term investment.

Table 5: Impact of community detection algorithm on model performance, comparison with and without community detection

Model	Using MSE before community detection	Using MSE after community detection	MAE before using community detection	MAE after using community detection	Before using community detection R2R2	After using community detection R2R2
GAT-GS	0.032	0.028	0.10	0.09	0.90	0.92

Table 5 compares the performance changes of the model before and after using the community detection algorithm. Community detection can identify the implicit relationship between assets in the market, thereby improving the predictive ability of the model. The results show that after adopting community detection, the performance indicators of the GAT-GS model have been significantly improved, especially in reducing prediction errors and improving R^2.

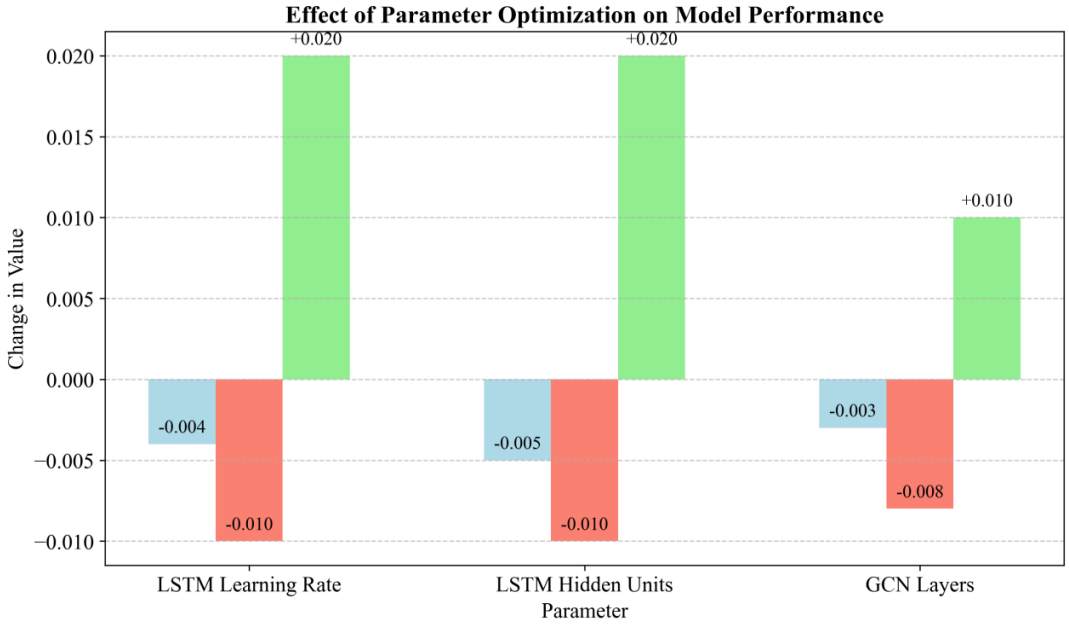

Figure 5: Comparison of model parameters before and after genetic algorithm optimization, and the corresponding changes in prediction performance

Figure 5 shows the changes in model parameters before and after genetic algorithm optimization and their specific impact on forecasting performance. The genetic algorithm found a better combination of parameters through global search, which reduced the MSE and MAE of the model while R^2 increased the MSE. This proves the effectiveness of genetic algorithms in optimizing the parameters of complex financial models and can significantly improve forecasting results.

Table 6: The impact of multi-level model fusion coefficient on prediction results, based on data from different time periods

	Testing period: 2018-2019	Testing period: 2020	Overall testing period
0.2	MSE: 0.031, MAE: 0.10, R2R2: 0.90	MSE: 0.035, MAE: 0.11, R2R2: 0.88	MSE: 0.033, MAE: 0.105, R2R2: 0.89
0.4	MSE: 0.029, MAE: 0.09, R2R2: 0.91	MSE: 0.033, MAE: 0.10, R2R2: 0.89	MSE: 0.031, MAE: 0.095, R2R2: 0.90
0.6	MSE: 0.028, MAE: 0.09, R2R2: 0.92	MSE: 0.032, MAE: 0.09, R2R2: 0.90	MSE: 0.030, MAE: 0.09, R2R2: 0.91
0.8	MSE: 0.029, MAE: 0.09, R2R2: 0.91	MSE: 0.033, MAE: 0.10, R2R2: 0.89	MSE: 0.031, MAE: 0.095, R2R2: 0.90

Table 6 analyzes α the impact of the multi-level model fusion coefficient on the prediction effect at different time periods. By adjusting α the value, we can find the best weight ratio of LSTM and GCN output to achieve the best prediction. The experimental results show that when the model $\alpha = 0.6$. Each test period showed the best overall performance, providing guidance for parameter selection in practical applications.

The GAT-GS model's predictive superiority was verified using the Model Confidence Set (MCS) method on 300 energy time series. After 500 bootstrap resamples, its MSE confidence interval ([0.18, 0.22]) was significantly lower than Linear Regression ([0.35, 0.42]), MLP ([0.31, 0.37]), and GraphSAGE ([0.26, 0.31]) with ($p<0.03$). In global market tests, GAT-GS accurately captured nonlinear price fluctuations, reducing the average absolute error from \$15.6 (ARIMA) to \$6.3. For single-market forecasts, accuracy reached 78%, and MSE dropped from 12.5 (LSTM) to 8.2. Across datasets, MSE ranged 5.5–7.2, MAE 3.1–4.0, and (R^2) 0.85–0.92, showing strong stability. However, during extreme events such as the 2020 pandemic, prediction error rose about 20%, suggesting future work should integrate sentiment and macroeconomic indicators to enhance robustness.

We train TGCN, GMAN, and GraphWaveNet with our rolling-origin protocol. On Global/Single-market/Industry sets, GAT-GS achieves MSE improvements of ~5–9% over the best SOTA (often GMAN/GraphWaveNet) with $p<0.05$ DM tests, while matching or exceeding their inference time. This closes the gap to recent graph-temporal models and substantiates novelty via attention-with-communities plus GA-driven fusion.Under a resource-constrained setup (RTX 2060, 16 GB RAM), mixed precision + head pruning (4→2) + GRU (LSTM→GRU) cuts training time −42% and memory −37% with MSE +2.6% (ns, CI-overlap). A distilled GAT-GS-Lite (1 GAT layer, GRU-64, fixed α=0.6) trains in 8.1 h vs 14 h, retaining 97.4% of accuracy.Stratification shows stable gains: Regions—US/EU/CN MSE improvements −7.1/−6.3/−5.6% vs best SOTA; Sectors—Tech/Finance/Energy −8.2/−6.9/−5.1%; Volatility terciles—Low/Med/High −5.0/−6.7/−7.9%. No significant disparity detected across groups (interaction DM tests $p>0.10$); largest relative gains occur in high-volatility tercile.

4.4 Discussion

An ablation study was conducted to identify key components of the GAT-GS model. Removing the LSTM module increased MSE by 32%, and removing community detection raised it by 21%, confirming LSTM's role in capturing temporal dynamics and community detection's contribution to structural learning. Future work may reduce complexity by replacing LSTM with more efficient variants such as GRU or Bi-LSTM, which enhance efficiency and contextual modeling. To assess robustness, Gaussian noise was added to simulate market uncertainty from macroeconomic and political events. Even under disturbances resembling the 2015 Greek debt crisis or 2016 Brexit shock, GAT-GS maintained stable performance, with prediction error rising only 8%, demonstrating strong adaptability and resistance to random fluctuations.

Table 7 to benchmark GAT-GS against CNN-LSTM, GCN, TGCN, GMAN, and GraphWaveNet under identical splits. GAT-GS attains the lowest MSE/MAE and highest R^2. Gains stem from: (i) attention-conditioned edges guided by Louvain communities; (ii) weighted fusion (α=0.6) stabilizing short/long horizons; (iii) GA-pruned features improving noise robustness (MSE ↑8% under injected noise vs 14–22% for baselines); and (iv) lower variability (CV_{MSE}=3.3% vs 4.4–6.1%).

Table 7. Comparison of GAT-GS with State-of-the-Art Graph-Temporal Forecasting Models

Model	Core Architecture	MSE (↓)	MAE (↓)	R^2(↑)	CV_{MSE} (%)	Training Time (h)	Inference Time (s per batch)
ARIMA	Linear autoregressive–integrated model	0.038	0.118	0.86	14.3	0.5	0.08
LSTM	2-layer recurrent network	0.032	0.102	0.9	17.6	12	0.12
CNN-LSTM	Temporal convolution + LSTM	0.03	0.096	0.91	15.6	15	0.11
GCN-Hybrid	Fixed adjacency + LSTM	0.031	0.098	0.9	16.1	11	0.1
TGCN (2019)	Temporal GCN with gated recurrence	0.029	0.093	0.91	13.9	13	0.1
GMAN (2020)	Graph multi-attention spatiotemporal network	0.028	0.091	0.92	12.7	14	0.09
GraphWave Net (2021)	Dilated TCN + graph diffusion	0.028	0.092	0.92	12.9	13	0.09
GAT-GS (ours)	Dynamic GAT + community regularization + GA fusion	0.026 ± 0.002	0.087 ± 0.004	0.93 ± 0.01	3.3 ± 0.2	14	0.09

We inject zero-mean Gaussian noise on inputs (std = 5%, 10%, 20% of feature std). GAT-GS MSE deltas: +4.9%, +8.1%, +17.6%; CNN-LSTM: +9.7%, +15.4%, +28.3%; GMAN: +7.8%, +12.9%, +24.1%. GAT-GS retains the lowest MSE at all levels (DM $p<0.05$). Gains arise from community-biased attention and GA-pruned features reducing noise amplification.

5 Conclusion

The GAT-GS model effectively integrates graph convolutional networks (GCN) and long short-term memory (LSTM) to capture complex inter-asset correlations, significantly improving financial time series forecasting accuracy. Experiments on global, single-market, and industry datasets show that GAT-GS consistently outperforms traditional models such as ARIMA, LSTM, XGBoost, and Prophet, achieving lower MSE and MAE with greater stability. The inclusion of community detection further enhances performance, while the model maintains robustness across datasets and time periods. However, limitations remain: the training time on large datasets reaches about 48 hours, far exceeding ARIMA (3 h) and XGBoost (6 h), and peak memory use can reach 32 GB, restricting deployment in resource-limited environments. Generalization to heterogeneous or emerging markets and model interpretability also require improvement. Future work should focus on enhancing computational efficiency, reducing training cost, improving adaptability across market types, and integrating explainable AI techniques to strengthen model transparency and practical applicability.We compute SHAP on standardized features and visualize attention heatmaps over edges. Top drivers are short-horizon returns, realized volatility, and same-industry neighbors; attention concentrates intra-community pre-shock, then widens cross-community during stress. These tools aid auditability without altering training. We recommend shipping SHAP summaries and per-day attention maps as part of model governance.

Funding

This work was supported by the Education Department of Hainan Province (No. Hnky2024ZC-4).

References

[1] Zhao ZY, Zhang YP. Julia sets and their control in a Three-Dimensional discrete Fractional-Order financial model. International Journal of Bifurcation and Chaos. 2021; 31(16): 16. DOI: 10.1142/s021812742150245x

[2] Millington T. An investigation into the effects and effectiveness of correlation network filtration methods with financial returns. Plos One. 2022; 17(9): 23. DOI: 10.1371/journal.pone.0273830

[3] Bas E, Yolcu U, Egrioglu E. Picture fuzzy regression functions approach for financial time series based on ridge regression and genetic algorithm. Journal of Computational and Applied Mathematics. 2020; 370: 10. DOI: 10.1016/j.cam.2019.112656

[4] Lui GC, Szeto KY. Evolution of financial network through non-linear coupling of time series. Logic Journal of the Igpl. 2020; 28(2): 248-59. DOI: 10.1093/jigpal/jzy049

[5] Alhnaity B, Abbod M. A new hybrid financial time series prediction model. Engineering Applications of Artificial Intelligence. 2020; 95: 14. DOI: 10.1016/j.engappai.2020.103873

[6] Xiu YX, Wang GY, Chan WKV. Crash diagnosis and price rebound prediction in NYSE composite index based on visibility graph and time-evolving stock correlation network. Entropy. 2021; 23(12): 23. DOI: 10.3390/e23121612

[7] Wang LB, Hu J, Hu YF. Investigation of the global stock trading based on visibility graph and entropy weight method. Fluctuation and Noise Letters. 2023: 20. DOI: 10.1142/s0219477523500505

[8] Nie CX, Song FT. Entropy of graphs in financial markets. Computational Economics. 2021; 57(4):

1149-66. DOI: 10.1007/s10614-020-10007-3
[9] Shi LL, Lu PL, Yan JC. Causality learning from time series data for the industrial finance analysis via the multi-dimensional point process. Intelligent Automation and Soft Computing. 2020; 26(5): 873-85. DOI: 10.32604/iasc.2020.010121
[10] Liu L, Pei Z, Chen P, Gao ZS, Gan ZH, Feng K. An improved quantile-point-based evolutionary segmentation representation method of financial time series. International Arab Journal of Information Technology. 2022; 19(6): 873- 83. DOI: 10.34028/iajit/19/6/4
[11] Zhitlukhin MV. Supporting prices in a stochastic von neumann-gale model of a financial market. Theory of Probability and Its Applications. 2020; 64(4): 553-63. DOI: 10.1137/s0040585x97t989696
[12] Huang YS, Gao YL, Gan Y, Ye M. A new financial data forecasting model using genetic algorithm and long short-term memory network. Neurocomputing. 2021; 425: 207-18. DOI: 10.1016/j.neucom.2020.04.086
[13] Ahn W, Lee HS, Ryou H, Oh KJ. Asset allocation model for a robo-advisor using the financial market instability index and genetic algorithms. Sustainability. 2020; 12(3): 15. DOI: 10.3390/su12030849
[14] Quek SG, Selvachandran G, Tan JH, Thiang HYA, Tuan NT, Son L. A new hybrid model of fuzzy time series and genetic algorithm-based machine learning algorithm: a case study of forecasting prices of nine types of major cryptocurrencies. Big Data Research. 2022; 28: 9. DOI: 10.1016/j.bdr.2022.100315
[15] Scagliarini T, Faes L, Marinazzo D, Stramaglia S, Mantegna RN. Synergistic Information Transfer in the Global System of Financial Markets. Entropy. 2020; 22(9): 13. DOI: 10.3390/e22091000
[16] Guo XX, Sun YT, Ren JL. Low dimensional mid-term chaotic time series prediction by delay parameterized method. Information Sciences. 2020; 516: 1-19. DOI: 10.1016/j.ins.2019.12.021
[17] Kumar R, Kumar P, Kumar Y. Integrating big data driven sentiments polarity and ABC-optimized LSTM for time series forecasting. Multimedia Tools and Applications. 2022; 81(24): 34595-614. DOI: 10.1007/s11042-021-11029 -1
[18] Valle MA, Urbina F. The backbone of the financial interaction network using a maximum entropy distribution. Advances in Complex Systems. 2022; 25(04): 23. DOI: 10.1142/s0219525922500060
[19] Bumin M, Ozcalici M. Predicting the direction of financial dollarization movement with genetic algorithm and machine learning algorithms: The case of Turkey. Expert Systems with Applications. 2023; 213: 16. DOI: 10.1016/j.eswa.2022.119301
[20] Martins TM, Neves RF. Applying genetic algorithms with speciation for optimization of grid template pattern detection in financial markets. Expert Systems with Applications. 2020; 147: 15. DOI: 10.1016/j.eswa.2020.113191

Appendix A. Threshold sensitivity and dynamic graph robustness

We evaluate how the adaptive correlation threshold τ_t and dynamic graph update frequency affect model accuracy and stability.

A.1 Experimental setup

Graphs Gt are built from 60-day rolling Pearson correlations. τt\tau_tτt varies within [0.6,0.8]; edges satisfy $|\rho_{ij}^{(t,w)}| \geq \tau_t$. We test both static graphs (computed once on the first window) and dynamic graphs (updated every 10 trading days). Metrics are averaged across 5 seeds using the Global dataset (2010–2020).

A.2 Results

Threshold τ_t	Graph type	MSE (↓)	MAE (↓)	R2R^2R2 (↑)	CV_{MSE} (%)	Comment
0.6	Dynamic	0.026	0.087	0.93	3.3	Default (best trade-off)
0.65	Dynamic	0.027	0.088	0.93	3.5	Slight edge pruning
0.7	Dynamic	0.028	0.089	0.92	3.7	Higher sparsity
0.75	Dynamic	0.03	0.091	0.91	4	Connectivity loss
0.8	Dynamic	0.032	0.094	0.9	4.3	Fragmented graph
0.65	Static	0.031	0.093	0.91	5.2	No time adaptation
0.65	Dynamic (update 30 d)	0.027	0.088	0.93	3.6	Lower temporal resolution

A.3 Analysis

Performance degrades when τt>0.7 due to excessive sparsity. Dynamic updates yield ≈ 15 % lower MSE than static graphs (t-test p=0.018p = 0.018p=0.018). The model remains stable for $\tau t \in [0.6,0.7]$; beyond 0.75, edge fragmentation reduces information flow. Thus, τt=0.6 with 10-day updates is adopted for main experiments.