Available Online: 31 July 2026 DOI: 10.54254/2977-3903/2026.35794

STSGAT: a Spatio-Temporal Sentiment Graph Attention Network for stock volatility forecasting

Ziheng Chen

University of Wariwick, Coventry, UK

yingbaiaaa@outlook.com

Abstract. Stock volatility forecasting is important in financial markets, as it supports risk management, portfolio allocation, derivative pricing, and investment decision-making. However, traditional statistical and time series models often struggle to capture nonlinear dynamics, cross-asset relationships, and external information influencing market behaviour. Although machine learning and deep learning methods have improved prediction performance, many existing models still rely mainly on historical market data and insufficiently incorporate financial news and investor sentiment. To address these limitations, this study proposes a hybrid Spatio-Temporal Sentiment Graph Attention Network (STSGAT), combining Long Short-Term Memory (LSTM) and Graph Neural Networks (GNNs). LSTM captures temporal dependencies within each stock, while GNN models structural relationships and information propagation across stocks. VADER-based sentiment scores from financial news are incorporated to account for both market structure and news-driven emotional dynamics. Overall, this study develops an integrated STSGAT framework combining temporal learning, graph learning, and sentiment analysis. Empirical results show that the proposed model outperforms baseline models, providing more accurate guidance for financial investment strategies and risk-related decision-making.

Keywords: stock volatility forecasting, Graph Neural Networks, Long Short-Term Memory, financial news sentiment, Spatio-Temporal Learning

1. Introduction

Forecasting stock market volatility has always been a central focus in financial research and practice. Volatility serves as a barometer of market uncertainty and plays a decisive role in risk management, asset pricing and portfolio construction. Both academics and market professionals rely on accurate volatility forecasts to guide their decision-making. Whilst traditional market data provides a foundation, it often fails to fully reflect market dynamics. For example, financial news frequently contains timely insights into corporate events, macroeconomic changes and investor expectations, information that may not be reflected in historical price movements [1, 2]. Integrating such external information is increasingly recognised as key to enhancing our understanding of volatility and improving forecasting capabilities. Against this backdrop, this study proposes a modelling approach that combines market data with news signals to more effectively reflect the complexity of contemporary financial markets.

In the field of financial forecasting, a variety of methods have been applied, ranging from traditional statistical models to modern machine learning and deep learning techniques. Classical models such as ARIMA and GARCH are favoured for their interpretability and computational efficiency; however, their reliance on assumptions of linearity and stationarity limits their effectiveness in capturing the complex dynamics typical of financial markets [1, 2]. Although machine learning models such as Support Vector Machines (SVMs) and random forests are better able to model non-linear relationships, they still require extensive feature engineering and often neglect temporal dependencies [3]. In recent years, deep learning methods—particularly those employing recurrent architectures such as LSTMs—have demonstrated greater capability in learning sequence patterns and have achieved encouraging results in stock price forecasting tasks [4, 5]. Despite these advances, traditional models tend to focus narrowly on the temporal information of individual stocks, whilst overlooking the broader structural relationships between assets in real markets. Stocks are interrelated through sector affiliation, market trends and information spillovers, whilst financial news further influences volatility by shaping market sentiment and expectations. Although Graph Neural Networks (GNNs) and sentiment analysis techniques have been employed separately to address these issues, their integration remains limited in the existing literature [2, 6, 7].

Stock volatility is influenced by both interactions among related assets and information released through financial news. These factors are often overlooked when forecasting models rely primarily on historical price series. Previous studies have demonstrated that graph-based approaches are effective in modelling inter-stock dependencies, while sentiment extracted from financial news provides valuable external information about corporate events, market expectations, and investor behaviour [1, 2, 6, 7]. To incorporate these complementary sources of information, this study develops a hybrid forecasting framework that combines LSTM networks with graph neural networks and financial news sentiment. Within the proposed framework, LSTM is responsible for learning the temporal dynamics of individual stocks, whereas the graph neural network captures structural relationships across the market. News sentiment is introduced as an additional source of information to reflect the influence of market events on volatility. The proposed framework is evaluated to determine whether integrating temporal information, inter-stock dependencies, and news sentiment leads to more accurate volatility forecasts and provides stronger support for investment-related decision making.

This study makes several technical and practical contributions: Technically, the proposed STSGAT model integrates LSTM, GNN, and sentiment scores into a unified forecasting framework. LSTM contributes by capturing temporal dependencies and volatility patterns within individual stocks. GNN contributes by modelling structural relationships and information spillovers between different stocks. Sentiment scores extracted from financial news provide external market information and help the model reflect news-driven investor expectations. Practically, this framework can support financial institutions in improving risk management and portfolio allocation, helping companies better understand how market information affects their stock volatility, and providing individual investors with more accurate forecasting signals for investment decision-making.

2. Related work

2.1. Statistical methods

Numerous statistical models, such as ARIMA, GARCH, and VAR [8], are widely used in time series analysis. These models have achieved considerable success in financial forecasting.

Among statistical forecasting methods, ARIMA, GARCH, and VAR remain some of the most widely adopted approaches for financial time series analysis. ARIMA models describe temporal dependence through

autoregressive and moving-average terms and have demonstrated satisfactory performance in relatively stable forecasting tasks, particularly for short-term prediction in economics and epidemiology [9, 10]. Their predictive accuracy, however, often decreases as the forecasting horizon becomes longer. GARCH models extend conventional time series analysis by explicitly modelling conditional variance, making them well suited for describing volatility dynamics and financial risk. VAR models provide a framework for analysing multiple time series simultaneously and are capable of representing interactions among different variables. Nevertheless, when applied to highly interconnected financial systems characterised by nonlinear relationships and structural changes, VAR models may become computationally demanding and their forecasting performance can deteriorate. These limitations have encouraged increasing interest in machine learning and deep learning methods, which offer greater flexibility for modelling complex financial data.

2.2. Machine learning

Machine learning models such as Support Vector Machines (SVM) and Random Forests [3] have been applied to financial forecasting tasks, offering enhanced flexibility in capturing non-linear patterns. For instance, Random Forests leverage ensemble learning to handle high-dimensional feature spaces and identify complex interactions among variables, demonstrating robust performance in stock market prediction [1]. SVM excels at mapping data into higher-dimensional spaces through kernel functions, enabling it to separate non-linear patterns effectively and has been successfully applied to integrate technical indicators with sentiment analysis for stock price prediction [2]. Compared with the traditional statistical models discussed above, machine learning methods demonstrate stronger capability in modelling non-linear relationships, while also offering flexibility in feature selection and a certain degree of model interpretability. However, these methods remain constrained by their dependence on feature engineering and their relatively limited capacity to automatically capture temporal dependencies in sequential data. Such limitations have encouraged the emergence of deep learning architectures designed to learn temporal representations directly from raw data.

2.3. Deep learning

Deep learning models, including LSTM, GNN, and Transformers [5, 11], have emerged as effective approaches for time series forecasting. These models can capture long-term dependencies and complex nonlinear temporal structures that traditional forecasting algorithms often fail to represent. Building upon prior work, Recurrent Neural Networks (RNN) have been widely adopted for modelling sequential dependencies in time series data. RNN architectures enable the capture of temporal patterns by maintaining hidden states that encode historical information, making them particularly suitable for financial forecasting tasks [4, 12]. However, standard RNNs suffer from the vanishing gradient problem during training, limiting their ability to learn long-term dependencies. To address this limitation, advanced recurrent architectures such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) have been developed. These variants introduce gating mechanisms that selectively retain or discard information, effectively mitigating gradient vanishing and enabling the learning of complex sequential patterns in financial data [4].

Over recent years, published work has applied Graph Neural Networks (GNNs), offering a robust approach to modelling dependencies within time-series data. In simple terms, each variable (stock, sensor, asset, etc.) is conceptualised as a node, with relationships represented as edges. Whilst classical deep learning techniques process sequences in isolation, GNNs simultaneously capture temporal and structural dependencies, thereby enabling the design of models for diverse correlations within dynamic systems. Within finance, Graph Convolutional Networks (GCNs) and hybrid GNN-LSTM schemes have demonstrated strong performance in stock forecasting and volatility simulation based on inter-asset relational information [6, 13]. Model variants

like Model-GNNs further enhance stability and generalization capabilities by incorporating temporal learning and mathematical priors like matrix inversion and Taylor series equivalence [14]. These studies confirm that GNN-based architectures often provide more interpretable approaches for complex time series forecasting, thereby justifying their application in this paper.

In summary, although these models have shown strong predictive capability, they often fail to adequately account for external influencing factors, which can substantially affect stock market behaviour. This limitation highlights the importance of exploring and integrating complementary approaches that are better suited to incorporating exogenous information into forecasting frameworks. Such integration may improve predictive accuracy and, in turn, provide more robust and reliable analytical foundations for asset allocation and investment decisions.

2.4. Financial news impact on the stock market

Financial news plays an important role in stock market forecasting because it conveys information about firm performance, investor expectations, macroeconomic developments, and market sentiment that is not fully reflected in historical price series alone. As financial markets are highly sensitive to new information, news reports can influence both price movements and volatility shifts by shaping investors' reactions and market psychology.

Sentiment analysis provides a practical way to convert unstructured financial news into quantitative information that can be incorporated into forecasting models. Instead of treating news reports as raw text, the approach assigns sentiment scores that reflect the polarity of the information conveyed in each article. These scores can then be combined with conventional market variables, allowing textual information and numerical indicators to be analysed within the same forecasting framework. Such integration offers a broader description of market conditions than relying on historical trading data alone.

In financial forecasting, the integration of sentiment analysis with deep learning has attracted increasing attention, as deep learning models are better able to capture temporal patterns and nonlinear interactions than conventional machine learning approaches. Recent literature reviews show that deep learning methods such as LSTM and CNN have achieved promising results in stock market prediction and often outperform traditional statistical models like ARIMA as well as some conventional machine learning methods when complex and heterogeneous data sources are involved [1]. More specifically, research on the Hong Kong stock market demonstrates that combining stock price indicators with news sentiment features in an LSTM-based framework produces higher predictive accuracy than SVM- and MKL-based alternatives, highlighting the importance of jointly modelling temporal dynamics and sentiment information [2].

Building on this line of research, recent studies have further explored the use of graph-based deep learning models in finance. Survey evidence suggests that hybrid and computational intelligence approaches are increasingly valued for their ability to handle the nonlinear and interconnected nature of stock markets [15]. In particular, GNN-based models have been shown to capture structural dependencies among related assets more effectively than models relying solely on sequential inputs. For example, one study combines GNNs with LSTM and rolling window analysis to model relationships among stocks in the Nikkei 225 market and reports superior investment performance relative to standard LSTM benchmarks [6]. Another study applies GNNs to financial volatility modelling and Value-at-Risk estimation, showing that graph-based representations improve prediction performance by exploiting interdependencies between assets [7]. Motivated by these findings, the present study integrates sentiment analysis with a GNN-based stock volatility forecasting framework, incorporating VADER-derived sentiment scores from financial news so that the model can capture not only structural relationships among assets but also the emotional dynamics conveyed by market news.

3. Methodology

3.1. Data description

This paper utilizes two primary datasets, covering 1,008 trading days, from 01/01/2019, to 12/31/2022.

• Stock prices: This dataset is collected from Yahoo Finance and comprises the daily closing prices of 18 stocks. The list of selected companies and their corresponding tickers is shown in Table 1 [16].

• Financial news: This dataset is collected from GDELT and contains daily news articles corresponding to each stock [17].

Table 1. Company names and tickers

Company Name	Ticker
Apple Inc.	AAPL
Amazon.com Inc.	AMZN
Bank of America Corporation	BAC
The Walt Disney Company	DIS
Alphabet Inc. (Class A)	GOOGL
The Goldman Sachs Group, Inc.	GS
Intel Corporation	INTC
Johnson & Johnson	JNJ
JPMorgan Chase & Co.	JPM
McDonald's Corporation	MCD
Meta Platforms, Inc.	META
Microsoft Corporation	MSFT
Oracle Corporation	ORCL
Pfizer Inc.	PFE
Starbucks Corporation	SBUX
Tesla, Inc.	TSLA
Visa Inc.	V
Walmart Inc.	WMT

Closing prices are subsequently converted to logarithmic returns, which is defined as $r_{t,i} = ln(\frac{P_{t,i}}{P_{t-1,i}})$, where $P_{t,i}$ denotes the closing price of stock i at time t. Volatility sequences are then calculated using the 5-day rolling standard deviation, given by

$$\sigma_{t,i} = \sqrt{\frac{1}{5-1} \sum_{k=0}^{4} (r_{t-k,i} - \overline{r_{t,i}^{(5)}})^2} \tag{1}$$

$$\overline{r_{t,i}^{(5)}} = \frac{1}{5} \sum_{k=0}^{4} r_{t-k,i} \tag{2}$$

Due to forward missing values caused by the rolling window, the final volatility matrix measures 1,003×18. To mitigate the impact of differing volatility scales across stocks on model training, each stock's volatility sequence undergoes columnar z-score standardization (mean 0, variance 1) in the experiments. This is denoted as $X \in R^{T \times N}$, where T = 1,003, N = 18.

Table 2 describes statistics for each stock, including minimum, maximum, mean, standard deviation, to assess overall distribution patterns and volatility behaviour. There is considerable heterogeneity across stocks in terms of both price levels and volatility. For example, technology stocks such as AAPL and MSFT exhibit relatively higher standard deviations, indicating greater price variability, whereas more stable firms like WMT display lower volatility. The wide range between minimum and maximum values across assets reflects substantial market fluctuations during the sample period, including periods of market stress and recovery. These findings highlight the presence of volatility clustering and cross-asset differences, which motivate the need for models capable of capturing both temporal dependencies and inter-stock relationships.

Table 2. Stock description

Ticker	Count	Max	Min	Mean	Std
AAPL	1008.0	182.009995	35.547501	110.75	43.22
AMZN	1008.0	186.570496	75.014000	129.21	33.40
BAC	1008.0	49.380001	18.080000	33.36	7.00
DIS	1008.0	201.910004	84.169998	136.88	28.51
GOOGL	1008.0	149.838501	51.273499	93.08	30.07
GS	1008.0	423.850006	134.970001	277.28	77.85
INTC	1008.0	68.470001	25.040001	50.19	9.36
JNJ	1008.0	186.009995	111.139999	154.66	16.05
JPM	1008.0	171.779999	79.029999	125.99	22.77
MCD	1008.0	278.399994	137.100006	221.85	27.65
META	1008.0	382.179993	88.910004	229.44	69.46
MSFT	1008.0	343.109985	97.400002	216.99	65.24
ORCL	1008.0	103.650002	39.799999	66.60	14.04
PFE	1008.0	61.250000	27.030361	41.19	6.91
SBUX	1008.0	126.059998	56.330002	90.99	15.17
TSLA	1008.0	409.970001	11.931333	159.33	116.34
V	1008.0	250.929993	128.130005	197.47	24.64
WMT	1008.0	53.290001	30.953333	43.15	5.32

To incorporate financial news into the forecasting framework, we apply sentiment analysis to extract polarity score of each news article. Specifically, every news item related to stock i on day t is processed to obtain a sentiment value reflecting its emotional polarity. Since multiple news articles may correspond to the same stock on the same day, these article-level sentiment scores are further aggregated by averaging them. In this way, a daily average sentiment score is constructed for each stock, which serves as the stock-specific sentiment representation for that day and is used as an additional input feature in the subsequent model.

3.2. Evaluation metrics

We employ a set of common time series forecasting metrics to evaluate model performance. Prediction accuracy is assessed using Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

MAE measures the average absolute deviation between predicted and actual values, defined as:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3}$$

RMSE, which penalizes larger errors more heavily due to the squaring operation, is computed as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4}$$

where y_i represents the actual volatility value, $\hat{y}_i$ denotes the predicted value, and n is the number of observations. These metrics measure the average deviation between estimated and actual volatility values, with lower values indicating superior predictive performance.

3.3. Sentiment analysis

To quantify the emotional content embedded in financial news, we employ the Valence Aware Dictionary and Sentiment Reasoner (VADER), a lexicon- and rule-based sentiment analysis tool widely used for short-form text analysis and financial sentiment extraction. VADER is suitable for financial text analysis due to its efficiency, interpretability, and ability to capture key linguistic features in news text, making it effective for real-time sentiment extraction in our GNN-based framework.

VADER operates by combining a sentiment lexicon—a dictionary of lexical features labelled according to their semantic orientation—with a set of grammatical rules that capture sentiment-relevant linguistic phenomena. Each word is assigned a sentiment intensity score ranging from −4 to +4, and these scores are aggregated while accounting for key heuristics, including punctuation amplification, capitalization emphasis, degree modifiers, contrastive conjunctions, and negation handling. The final output is a compound sentiment score normalized to the range [-1, 1], where values closer to −1 indicate negative sentiment, values closer to +1 indicate positive sentiment, and values near 0 indicate neutral sentiment.

Formally, for each news headline or article d, VADER produces a compound score:

$$Compound(d) = \frac{\sum_{i=1}^{m} s_i}{\sqrt{\sum_{i=1}^{m} s_i + \alpha}} \tag{5}$$

where s_i represents the valence score of the i^{th} sentiment-bearing word after applying grammatical adjustments, m is the number of sentiment-bearing tokens, and α is a normalization parameter.

For each stock i on day t, multiple news articles may be published. Therefore, sentiment scores are aggregated at the daily level to construct a stock-specific sentiment representation. Specifically, let $N_{i,t}$ denote the number of news articles associated with stock i on day t , and $s_{i,t}{}^{(k)}$ denote the compound sentiment score of the k^{th} article. The daily sentiment score is computed as:

$$S_{i,t} = \frac{1}{N_{i,t}} \sum_{k=1}^{N_{i,t}} s_{i,t}{}^{(k)} \tag{6}$$

This aggregation ensures that sentiment information from multiple news sources is consolidated into a single daily feature for each stock.

To capture short-term sentiment dynamics and reduce noise, a rolling window of length T is applied to construct sentiment sequences:

$$S_{i,t} = [S_{i,t-T+1}, \ldots, S_{i,t}] \tag{7}$$

These sentiment sequences are subsequently used to measure similarity between stocks and form the basis for dynamic graph construction in the proposed framework, where inter-stock relationships evolve according to sentiment patterns extracted from financial news.

3.4. STSGAT model

To overcome the limitations of traditional statistical models and sequence-based neural networks, this study proposes a Spatio-Temporal Sentiment Graph Attention Network (STSGAT), which jointly models temporal dynamics and cross-asset dependencies driven by financial news sentiment.

Unlike conventional approaches that rely on static relationships (e.g., industry classification or correlation matrices), this study constructs a time-varying graph structure based on sentiment similarity extracted from financial news.

For each stock i at time t, a sentiment sequence $S_{i,t}$ is obtained using the method described in Section 4.3. The similarity between stock i and stock j is computed using cosine similarity:

$$A_{i,j}^{(t)} = \frac{S_{i,t} \cdot S_{j,t}}{\|S_{i,t}\| \cdot \|S_{j,t}\|} \tag{8}$$

where $A^{(t)}$ denotes the dynamic adjacency matrix at time t.

This formulation enables the graph structure to evolve over time, reflecting changes in market sentiment and information flow across assets.

For each stock, historical input features—including volatility and sentiment—are processed using an LSTM network to capture temporal dependencies. The LSTM generates a hidden representation:

$$h_{i,t} = LSTM(X_{i,t-T+1}, \ldots, X_{i,t}) \tag{9}$$

These embeddings summarize the recent temporal behaviour of each stock and serve as node features in the graph.

To model cross-asset dependencies, a Graph Attention Network (GAT) is applied to the dynamically constructed graph. For each pair of connected nodes i and j, an attention coefficient is computed as:

$$e_{ij} = LeakyReLU(a^T [Wh_i \| Wh_j]) \tag{10}$$

In practice, the neighbourhood set $N(i)$ is constructed by selecting stocks with non-zero or top-K similarity scores in the dynamic adjacency matrix, ensuring a sparse and informative graph structure.

These coefficients are normalized across the neighbourhood of node i using the softmax function:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k \in N(i)} exp(e_{ik})} \tag{11}$$

The updated node representation is then obtained by aggregating information from neighbouring nodes:

$$h_i\prime = \sigma(\sum_{j \in N(i)} \alpha_{ij} W h_j) \tag{12}$$

Through this attention mechanism, the model learns the relative importance of neighbouring stocks and captures heterogeneous and time-varying inter-stock dependencies.

Finally, the updated node representations are passed through a Multilayer Perceptron (MLP) to generate the prediction:

$$\widehat{y_{i,t}} = f(h_i\prime) \tag{13}$$

where $f(\cdot)$ denotes a nonlinear mapping. The model is trained end-to-end by minimizing the Mean Squared Error (MSE):

$$L = \frac{1}{N} \sum_{i=1}^{N} (y_{i,t} - \widehat{y_{i,t}})^2 \tag{14}$$

Optimization is performed using the Adam optimizer, and early stopping based on validation performance is applied to prevent overfitting.

3.5. Comparison methods

3.5.1. AR model

The AutoRegressive (AR) model is a fundamental time series forecasting technique that predicts future values based on a linear combination of past observations. The AR model assumes that the current value of a time

series can be expressed as a weighted sum of its previous values plus a stochastic error term. This approach is particularly suitable for capturing temporal dependencies in stationary or weakly stationary time series data.

An AR model of order p, denoted as $AR(p)$, is formally defined as:

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t \tag{15}$$

where:

y_t represents the value of the time series at time t;

c is a constant term (intercept);

ϕ_i denotes the autoregressive coefficient for lag i, capturing the influence of the observation i periods in the past;

p is the order of the model, indicating the number of lagged values included;

y_{t-i} represents the lagged value of the time series at time $t - i$;

ϵ_t is the white noise error term at time t, assumed to be independently and identically distributed with mean zero and constant variance σ^2.

The optimal lag order p is determined through model selection criteria such as the Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC), which balance model fit against complexity to prevent overfitting.

For volatility forecasting, the AR model captures the autocorrelation structure inherent in financial time series, where current volatility levels tend to be influenced by recent historical volatility. In contrast, the AR model provides a simple and interpretable baseline, it assumes linear relationships and may not adequately capture the complex nonlinear dynamics and regime-switching behaviour often observed in financial market volatility. This limitation motivates the comparison with more sophisticated models, including our proposed GNN-based approach, which can capture heterogeneous relationships across multiple assets.

3.5.2. ARIMA model

The AutoRegressive Integrated Moving Average (ARIMA) model extends the AR framework by incorporating differencing operations to handle non-stationary time series and adding moving average components to capture the influence of past forecast errors. ARIMA models are widely employed in econometric forecasting due to their flexibility in modelling various temporal patterns, including trends, seasonality, and autocorrelation structures commonly observed in financial data.

The ARIMA(p, d, q) model is a combination of AutoRegressive (AR), Integrated (I - differencing), and Moving Average (MA) components. The general form of the ARIMA model is:

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t \tag{16}$$

where:

Y_t = Actual value at time t ;

c = Constant term;

ϕ_i = Coefficients for the AutoRegressive (AR) part, measuring the impact of past values;

Y_{t-i} = Past values (lags) used for prediction;

θ_j = Coefficients for the Moving Average (MA) part, measuring the impact of past forecast errors;

ϵ_t = Error term (random noise) at time t ;

ϵ_{t-j} = Past forecast errors;

p = Order of AR (number of lags of the differenced series);

q = Order of MA (number of past errors);

d = Order of differencing (to make data stationary).

Model identification involves determining the appropriate orders (p, d, q) through a combination of diagnostic tools:

• Augmented Dickey-Fuller (ADF) test for assessing stationarity and determining the differencing order d ;

• Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots for identifying appropriate values of p and q ;

• Information criteria (AIC/BIC) for final model selection, balancing goodness-of-fit against model complexity.

In the context of volatility forecasting, ARIMA models capture both the persistence of volatility shocks (through the AR component) and the short-term impact of unexpected market events (through the MA component). The differencing operation addresses non-stationarity commonly observed in financial time series, such as trending behaviour in volatility levels.

However, ARIMA assumes linear relationships and homoscedastic errors, which may not adequately represent the volatility clustering, asymmetric responses to positive and negative shocks, and complex interdependencies across multiple assets observed in financial markets. These limitations underscore the need for more advanced approaches, such as our proposed GNN-based framework, which can capture nonlinear cross-asset dependencies and heterogeneous market dynamics without requiring strong parametric assumptions.

3.5.3. LSTM model

Long Short-Term Memory (LSTM) networks are employed as a nonlinear benchmark model to capture temporal dependencies in stock volatility. As a variant of Recurrent Neural Networks (RNNs), LSTM is specifically designed to address the vanishing gradient problem, enabling the model to effectively learn long-range dependencies in sequential data [18].

For a given time step t:

1. Forget Gate (decides what to remove):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{17}$$

2. Input Gate (decides what new information to add):

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{18}$$

$$\widetilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{19}$$

3. Update Cell State:

$$C_t = f_t \circ C_{t-1} + i_t \circ \widetilde{C}_t \tag{20}$$

4. Output Gate (decides what to output):

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{21}$$

$$h_t = o_t \circ tanh(C_t) \tag{22}$$

where:

x_t = input at time t;

h_{t-1} = Hidden state from previous time step;

C_t = Cell state at time t;

W_f, W_i, W_C, W_o = Weight matrices for forget, input, cell, and output gates respectively;

b_f, b_i, b_C, b_o = Bias terms for each gate;

σ = Sigmoid activation function;

$tanh$ = Hyperbolic tangent activation function;

$\circ$ = Element-wise multiplication;

$[h_{t-1}, x_t]$ = Concatenation of previous hidden state and current input;

$\widetilde{C}_t$ = Candidate cell state.

In the context of this study, the LSTM is primarily used as a temporal feature extractor that encodes sequential patterns in each stock independently. It captures key characteristics of financial time series, including volatility persistence, clustering effects, and nonlinear temporal dynamics.

However, LSTMs face critical limitations for multi-asset forecasting as they process each asset independently, ignoring cross-asset dependencies and volatility spillovers in real markets. Even multivariate extensions assume static relationships through fixed weight matrices, failing to represent dynamic correlations that surge during crises. Scalability becomes prohibitive with large parameters, and LSTMs lack explicit representation of network structure like industry sectors or supply chains. These limitations underscore the need for GNN-based frameworks that explicitly model financial network topology, capture time-varying dependencies through message-passing, and scale efficiently to large portfolios.

3.5.4. GNN model

Graph Neural Networks (GNNs) are a class of deep learning models designed for graph-structured data. Their core idea is to learn node representations by aggregating information from neighbouring nodes through a message-passing mechanism. This makes GNNs particularly effective for modelling complex relationships and dependencies among data points, and therefore suitable for applications such as social networks, recommendation systems, fraud detection, and financial market analysis.

Formally, a graph is represented as $G = (V, E)$, where V denotes the set of nodes and E denotes the set of edges. Each node is associated with a feature vector, and the representation of a node is iteratively updated by combining its own features with information aggregated from its neighbors.

A general message-passing framework in GNNs can be written as

$$h_v^{(l+1)} = \text{AGGREGATE}\left(h_v^{(l)}, \left\{h_u^{(l)} \mid u \in N(v)\right\}\right) \tag{23}$$

where $h_v^{(l)}$ denotes the representation of node v at layer l, and $N(v)$ is the set of neighboring nodes of v.

In the case of Graph Convolutional Networks (GCNs), the aggregation is often implemented as mean pooling followed by a learnable linear transformation and a nonlinear activation function:

$$h_v^{(l+1)} = \sigma(W \sum_{u \in N(v)} \frac{h_u^{(l)}}{|N(v)|}) \tag{24}$$

where W is a trainable weight matrix, $|N(v)|$ is the number of neighbors of node v, and $\sigma(\cdot)$ is a nonlinear activation function.

A graph typically consists of nodes, edges, node features, and edge features. In financial market analysis, nodes may represent individual stocks, edges may encode relationships such as return correlation or sector co-membership, node features may include historical returns and trading volume, and edge features may capture measures such as correlation strength or volatility spillover intensity.

4. Experiment

This chapter evaluates the effectiveness of the proposed framework for stock volatility prediction by comparing it with traditional statistical models and deep learning baselines. The experimental workflow encompasses dataset construction and preprocessing, train/validation/test partitioning, hyperparameter tuning, and performance assessment using MAE and RMSE metrics.

4.1. Experiment setup

Each stock sequence is divided into training, validation, and test sets at a ratio of 80%/10%/10%. Early stopping is applied using the validation set, with final evaluation conducted on the test set. The historical time window length is lags = 10 days. Samples are generated from t = 10 days to $t = (T - 1)$ days.

To ensure a fair comparison, the core time window length for all models was uniformly set to lags = 10. The key parameters for each model are as follows.

1. AR and ARIMA Models: The AutoRegressive (AR) model is implemented with a lag order of p = 10, meaning that the previous 10 days of observations are used to predict the current volatility. The ARIMA model is specified as ARIMA (p = 10, d = 0, q = 0), where p = 10 represents the autoregressive lag order, d = 0 indicates that no differencing is applied, and q = 0 means that no moving-average term is included. Both models are trained using the combined training and validation dataset, corresponding to the first 90% of the time series.

2. LSTM Model: The LSTM model uses a sequence length of 10, meaning that the previous 10 days of data are used as input for prediction. The model is built with 3 LSTM layers, a hidden state dimension of 64, and a dropout rate of 0.2 to reduce overfitting. It is trained using the Adam optimizer with a learning rate of 5×10^{-4}, with a maximum of 100 training epochs and early stopping patience of 15. The loss function used during training is MSELoss, while MAE and RMSE are used for testing evaluation.

3. GNN Model: The GNN model uses a hybrid architecture that combines graph convolutional layers with a temporal GRU module. The graph topology is constructed using the pairwise correlation coefficient matrix between stocks, where the top 30% of absolute correlation coefficients are retained as adjacency relationships, with self-connections added. The adjacency matrix is then normalized using $A_{norm} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. For spatial modelling, a 3-layer graph convolutional network is used, consisting of Linear, LayerNorm, and ReLU operations, with a hidden dimension of 64 and a dropout rate of 0.3. For temporal modelling, a 3-layer GRU is applied to the sequential graph representations from each time step. The model is trained with a batch size of 32, a learning rate of 5×10^{-4}, a maximum of 150 epochs, early stopping patience of 20, and gradient clipping with $max_{norm} = 1.0$. MSELoss is used as the training loss, while MAE and RMSE are used for testing evaluation.

4. STSGAT Model: The proposed STSGAT model integrates the LSTM and GNN components described above into a unified forecasting framework. The LSTM component follows the same setting as the standalone LSTM model, using a sequence length of 10, 3 LSTM layers, a hidden dimension of 64, and a dropout rate of 0.2 to capture temporal dependencies within each stock. The GNN component follows the same graph construction and spatial modelling settings as the standalone GNN model, where inter-stock relationships are built from the top 30% of absolute pairwise correlation coefficients and normalised using $A_{norm} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. In addition, sentiment scores extracted from financial news are incorporated as external features to capture news-driven market effects. The model is trained using MSELoss, and MAE and RMSE are used for testing evaluation.

4.2. Experiment result

This section reports the test set performance of three baseline models (AR, ARIMA, LSTM) and provides the same evaluation metrics for the GNN model.

Table 3. Experiment results

Model	RMSE	MAE
AR (10)	0.765661	0.563434
ARIMA (10,0,0)	0.765055	0.562342
LSTM	0.443448	0.291726
GNN	0.679235	0.496468
STSGAT	0.416568	0.270334

Table 3 presents the average MAE and RMSE on the test set (averaged across 18 stocks). From this table, the results show that STSGAT achieves the best forecasting performance, with the lowest RMSE (0.416568) and MAE (0.270334). This suggests that integrating temporal learning, graph-based stock relationships, and news sentiment information can improve volatility prediction. Compared with AR, ARIMA, LSTM, and GNN, STSGAT performs better because it combines the advantages of LSTM and GNN while also incorporating external market information such as financial news sentiment and news volume. Therefore, the proposed model provides a more comprehensive and accurate framework for stock volatility forecasting.

AR is virtually identical to ARIMA: this is because the ARIMA model employed in the experiment utilised ARIMA (10,0,0), whose structure closely approximates that of AR (10) in terms of predictive capability. Consequently, the RMSE and MAE metrics are essentially consistent, indicating that relying solely on the linear autoregressive assumption yields limited fitting capability for this volatility sequence.

5. Conclusion

This study makes both technical and practical contributions by proposing a Spatio-Temporal Sentiment Graph Attention Network (STSGAT) for stock volatility forecasting. From a technical perspective, STSGAT provides an integrated framework that combines temporal learning, graph learning, and sentiment analysis. First, the LSTM component captures time-dependent volatility patterns within individual stocks. Second, the graph-based component models cross-asset structural relationships, allowing the model to learn inter-stock correlations, volatility spillover effects, and sector co-movement patterns. Third, sentiment information extracted from financial news is incorporated as an external signal, enabling the model to reflect news-driven market expectations and investor sentiment.

In addition, this study establishes a unified and reproducible experimental pipeline using 18 stocks from 2019 to 2022, including price data collection, volatility calculation, z-score normalization, a consistent lag window of 10 days, fixed train/validation/test splits, and common evaluation metrics such as MAE and RMSE. The empirical results show that STSGAT outperforms traditional statistical models and baseline deep learning models, demonstrating the value of jointly modelling temporal dynamics, cross-stock dependencies, and financial news sentiment.

From a practical perspective, the proposed framework can support financial institutions in improving risk management and portfolio allocation, help companies better understand how market news and inter-market relationships affect stock volatility, and provide individual investors with more reliable forecasting signals for investment decision-making.

Future work can further improve the proposed framework in three main directions. First, sentiment information can be more deeply integrated into the model by using daily sentiment scores as node features or exogenous variables alongside volatility sequences, allowing the model to capture both structural

dependencies and news-driven external shocks. Different sentiment extraction methods, such as finance-specific lexicons or BERT-based financial text models, could also be compared to evaluate their impact on forecasting performance. Second, future studies can replace the static correlation-based graph with dynamic graph structures, such as sliding-window correlations or attention-based adjacency matrices, to better capture time-varying relationships between stocks under different market conditions. Third, model interpretability should be enhanced through edge-weight visualization, node contribution analysis, or attribution methods, so that the model can better explain which asset relationships and sentiment factors drive volatility predictions.

References

[1] Kumbure, M. M., Lohrmann, C., Luukka, P., & Porras, J. (2022). Machine learning techniques and data for stock market forecasting: A literature review. *Expert Systems With Applications, 197*, Article 116659. https://doi.org/10.1016/j.eswa.2022.116659

[2] Li, X., Wu, P., & Wang, W. (2020). Incorporating stock prices and news sentiments for stock market prediction: A case of Hong Kong. *Information Processing & Management, 57*(5), Article 102212. https://doi.org/10.1016/j.ipm.2020.102212

[3] Teles, G., Rodrigues, J. J. P. C., Rabêlo, R. A. L., & Kozlov, S. A. (2021). Comparative study of support vector machines and random forests machine learning algorithms on credit operation. *Software: Practice and Experience, 51*(12), 2492–2500. https://doi.org/10.1002/spe.2842

[4] Fang, W., Chen, Y., & Xue, Q. (2021). Survey on research of RNN-based spatio-temporal sequence prediction algorithms. *Journal on Big Data, 3*(3), 97–110. https://doi.org/10.32604/jbd.2021.016993

[5] Zhao, Z., Chen, W., Wu, X., Chen, P. C. Y., & Liu, J. (2017). LSTM network: A deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems, 11*(2), 68–75. https://doi.org/10.1049/iet-its.2016.0208

[6] Matsunaga, D., Suzumura, T., & Takahashi, T. (2019). *Exploring graph neural networks for stock market predictions with rolling window analysis*. arXiv. https://arxiv.org/abs/1909.10660

[7] Xu, K., Wu, Y., Xia, H., Sang, N., & Wang, B. (2022). Graph neural networks in financial markets: Modeling volatility and assessing value-at-risk. *Journal of Computer Technology and Software, 1*(2). https://ashpress.org/index.php/jcts/article/view/85

[8] Grachev, O. Y. (2017). *Application of time series models (ARIMA, GARCH, and ARMA-GARCH) for stock market forecasting* [Honors thesis, Northern Illinois University, Huskie Commons]. https://huskiecommons.lib.niu.edu/studentengagement-honorscapstones/177

[9] Ospina, R., Gondim, J. A. M., Leiva, V., & Castro, C. (2023). An overview of forecast analysis with ARIMA models during the COVID-19 pandemic: Methodology and case study in Brazil. *Mathematics, 11*(14), Article 3069. https://doi.org/10.3390/math11143069

[10] Schaffer, A. L., Dobbins, T. A., & Pearson, S.-A. (2021). Interrupted time series analysis using autoregressive integrated moving average (ARIMA) models: A guide for evaluating large-scale health interventions. *BMC Medical Research Methodology, 21*, Article 58. https://doi.org/10.1186/s12874-021-01235-8

[11] Timashov, A. (2025). Graph transformers. *Universum: Technical Sciences, 1*(130). https://doi.org/10.32743/UniTech.2025.130.1.19149

[12] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Association for Computational Linguistics. https://aclanthology.org/D14-1179/

[13] Liu, J., Chen, Y., Huang, X., Li, J., & Min, G. (2023). GNN-based long and short term preference modeling for next-location prediction. *Information Sciences, 629*, 1–14. https://doi.org/10.1016/j.ins.2023.01.131

[14] Zhang, Y., Li, S., Weng, J., & Liao, B. (2024). GNN model for time-varying matrix inversion with robust finite-time convergence. *IEEE Transactions on Neural Networks and Learning Systems, 35*(1), 559–569. https://doi.org/10.1109/TNNLS.2022.3175899

[15] Kumar, G., Jain, S., & Singh, U. P. (2021). Stock market forecasting using computational intelligence: A survey. *Archives of Computational Methods in Engineering, 28*, 1069–1101. https://doi.org/10.1007/s11831-020-09413-5

[16] Yahoo Finance. (n.d.). *Yahoo Finance*. https://finance.yahoo.com

[17] GDELT Project. (n.d.). *The GDELT Project*. https://www.gdeltproject.org

[18] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735