# Multivariate Realized Volatility Forecasting with Graph Neural Network

Qinkai Chen
Ecole Polytechnique
Exoduspoint Capital Management
Palaiseau, France
qinkai.chen@polytechnique.edu

Christian-Yann Robert
ENSAE Paris
Palaiseau, France
christian-yann.robert@ensae.fr

## ABSTRACT

Financial economics and econometrics literature demonstrate that the limit order book data is useful in predicting short-term volatility in stock markets. In this paper, we are interested in forecasting short-term realized volatility in a multivariate approach based on limit order book data and relational stock market networks. To achieve this goal, we introduce Graph Transformer Network for Volatility Forecasting. The model allows combining limit order book features and a large number of temporal and cross-sectional relations from different sources. Through experiments based on about 500 stocks from S&P 500 index, we find a better performance for our model than for other benchmarks.

## CCS CONCEPTS

• **Computing methodologies** → **Semantic networks**; **Modeling methodologies**; Neural networks.

## KEYWORDS

realized volatility prediction, graph neural networks, multivariate modeling, options pricing

**ACM Reference Format:**
Qinkai Chen and Christian-Yann Robert. 2022. Multivariate Realized Volatility Forecasting with Graph Neural Network. In *3rd ACM International Conference on AI in Finance (ICAIF '22), November 2–4, 2022, New York, NY, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3533271.3561663

## 1 INTRODUCTION

Volatility is an important quantity in finance that evaluates the price fluctuation and represents the risk level of an asset. It is one of the most important indicators used in risk management and equity derivatives pricing. It is noteworthy that the volatility is not observable, but Andersen and Bollerslev [1] show that the realized volatility is a good estimator of this statistical measure of the dispersion of returns. Forecasting realized volatility has therefore attracted the attention of various researchers.

Brailsford and Faff [9] use GARCH (Generalized AutoRegressive Conditional Heteroskedasticity) models to forecast realized volatilities based on daily prices. Gatheral and Oomen [16] introduce several simple volatility estimators based on Limit Order Book (LOB) data[1] , showing that the use of LOB data can lead to better predicting results. More recently, Rahimikia and Poon [32] and Zhang and Rosenbaum [34] use machine learning techniques such as Recurrent Neural Network (RNN) to improve such predictions.

The aforementioned literatures adopt a univariate approach in this task, which means that the models only consider one outcome for one stock at the same time, instead of jointly considering all stocks, although the asset returns on the financial markets can be highly correlated [12]. Andersen et al. [2] propose linear multivariate volatility forecasting methods based on daily price data to take into account this correlation, while Bucci [11] further uses neural networks to forecast realized volatility covariance matrix non-linearly. Bollerslev et al. [8] first propose a parametric multivariate model based on LOB data using covolatility and covariance matrices.

Compared with the univariate approach, multivariate models can capture the relations among observations. Recently, Graph Neural Networks (GNN) [10] were proposed to integrate such relationship into the commonly used non-linear neural networks. This approach achieves significant success in multiple applications, such as traffic flow prediction [26], recommender systems [6] and stock movement prediction [13, 36]. To the best of our knowledge, no graph-based structure for volatility forecasting has been proposed in the literature.

Hence, to further improve the volatility forecasting performance, inspired by previous researches (**Sec. 2**) and our real-life use cases, we build a multivariate volatility forecasting model (**Sec. 3**) based on Graph Neural Network: Graph Transformer Network for Volatility Forecasting (GTN-VF). This model predicts the short-term volatilities from LOB data and both cross-sectional and temporal relationships from different sources (**Sec. 4**). With various experiments on about 500 stocks from the S&P 500 index, we demonstrate that GTN-VF outperforms other baseline models with a significant margin on different forecasting horizons (**Sec. 5**).

## 2 RELATED WORK

### 2.1 Volatility Forecasting

As introduced in Section 1, there are two types of volatility forecasting models: univariate models and multivariate models.

In univariate approaches, researchers can design intuitive estimators [43, 44], without considering relations among observations.

[1] A limit order book is the list of orders that a trading venue uses to record the interest of buyers and sellers in a particular financial instrument. We introduce our LOB data and aggregation methods in Section 5.1.


*ICAIF '22, November 2–4, 2022, New York, NY, USA*

ACM ISBN 978-1-4503-9376-8/22/11…$15.00
https://doi.org/10.1145/3533271.3561663

More commonly, researchers adopt time-series methods to model the time dependence while ignoring the cross-sectional relations and calibrating one set of parameters for each asset. For example, Brailsford and Faff [9] propose GARCH models, Sirignano and Cont [38] use RNN models with Long Short-Term Memory (LSTM) and Ramos-Perez et al. [33] adopt a Transformer structure [40].

Multivariate models add cross-sectional relationships and achieve better results. For example, Kwan et al. [25] introduce multivariate threshold GARCH model and Bollerslev et al. [8] propose a multivariate statistical estimator based on co-volatility matrices. It is worth noting that all models above only use asset covariances as the source to build the relationship while ignoring the intrinsic relations between the companies that issue the stocks.

## 2.2 Graph Neural Network

A graph is composed of nodes and edges, where a node represents an instance in the network and an edge denotes the relationship between two instances. It is an intuitive structure to describe the relational information. Recently, many researches focus on generalizing neural networks on graph structure to capture non-linear interactions among the nodes.

Bruna et al. [10] first generalize the Convolutional Neural Network (CNN) on graph-based data. Kipf and Welling [24] propose Graph Convolutional Network (GCN) and Defferrard et al. [15] introduce ChebNet, both of which have reduced network complexity and better predictive accuracy.

However, the aforementioned models need to load all the graph data into the memory at the same time, making training larger relational networks impossible. Gilmer et al. [17] state that Graph Neural Networks are essentially message passing algorithms. It means that the model makes decision not only based on one node's observation, but also the information passed from all other related nodes defined in the format of a graph. Based on this generalization, Hamilton et al. [20] propose GraphSAGE which allows batch training on graph data.

Shi et al. [37] further show that using a Transformer-like operator to aggregate node features and the neighbor nodes' features gives a better performance than a simple average such as GraphSAGE or an attention mechanism such as Graph Attention Network [41].

To close the gap in the researches, we propose GTN-VF, which adopts the state-of-the-art Graph Neural Networks to model the relationships. In addition, GTN-VF allows to integrate a large number of relational information, including both widely used covariance and other rarely used relations in previous researches such as sector and supply chain.

# 3 PROBLEM FORMULATION

We formulate this multivariate volatility forecasting problem as a regression task. The goal is to predict the realized volatility vector over the next $\Delta T$ seconds at a given time $t$ with all previously available data.

We first define the return of stock $s$ at time $t$ as

$$r_{s,t} = log\left(\frac{P_{s,t}}{P_{s,t-1}}\right) \tag{1}$$

where $P_{s,t}$ is the last trade price of $s$ at $t$.

We then use $RV_{s,t,\Delta T}$ to denote the realized volatility for stock $s$ between $t$ and $t + \Delta T$, it is defined as:

$$RV_{s,t,\Delta T} = \sqrt{\sum_{i=t}^{t+\Delta T} r_{s,i}^2} \tag{2}$$

Previous researches [29, 31, 32] usually calibrate one model for each stock. This prediction model $f_s$ for stock $s$ can be written as:

$$\widehat{RV_{s,t,\Delta T}} = f_s\left([D_{s,t_1}, ..., D_{s,t_m}], \theta\right) \tag{3}$$

where $D_{s,t}$ denotes the LOB data related to stock $s$ between $t$ and $t - \Delta T'$, and $t_1 < ... < t_m < t$. $\Delta T'$ is a parameter denoting the backward window used to build features for $t$, while $\theta$ represents the model parameters.

However, as stated in Section 1, the realized volatilities of stocks are related through their LOBs. We also consider this effect in our model, and we use $\mathcal{G}_s$ to denote the relationship of stock $s$ with all other stocks. In additional to the relationship among stocks, the relationship among the timestamps is also taken into account. For example, at time $t$, we check whether the behaviors of the stocks are similar to their behaviors at previous timestamps. We use $\mathcal{G}_t$ to denote this temporal relationship. Our prediction model $g$ is finally written as:

$$\widehat{RV_{s,t,\Delta T}} = g\left(\begin{bmatrix} D_{s_1,t_1} & ... & D_{s_1,t_m} \\ ... & ... & ... \\ D_{s_n,t_1} & ... & D_{s_n,t_m} \end{bmatrix}, \mathcal{G}_s, \mathcal{G}_t, \theta\right) \tag{4}$$

Our goal is to minimize the Root Mean Square Percentage Error (RMSPE) defined as

$$RMSPE = \sqrt{\frac{1}{N}\sum_{s,t}\left(\frac{\widehat{RV_{s,t,\Delta T}} - RV_{s,t,\Delta T}}{RV_{s,t,\Delta T} + \epsilon}\right)^2} \tag{5}$$

where $N$ is the total number of observations and $\epsilon$ is a small constant to avoid overflow.

We use RMSPE instead of the standard Mean Square Error (MSE) because the volatilities of different stocks are intrinsically different. The RMSPE loss function helps normalize the difference among stocks to make sure that the model has a similar effect on all stocks.

# 4 GRAPH TRANSFORMER NETWORK FOR REALIZED VOLATILITY FORECASTING

Our model consists of two main components: a LOB data encoder and a Graph Transformer Network. The LOB data encoder transforms numerical LOB data into multiple features. It also transforms categorical information, such as the stock ticker, into a fixed-dimension embedding. It finally concatenates the numerical features and the embedding for categorical features as the node feature.

The Graph Transformer Network then takes all the node features and the pre-defined relationship information as input. After training, it will give each node a new meaningful embedding which contains information from both LOB data and relational data. With a fully connected layer, we can get the final prediction of the realized volatility for this node.

An illustration of the whole structure is shown in Figure 1.

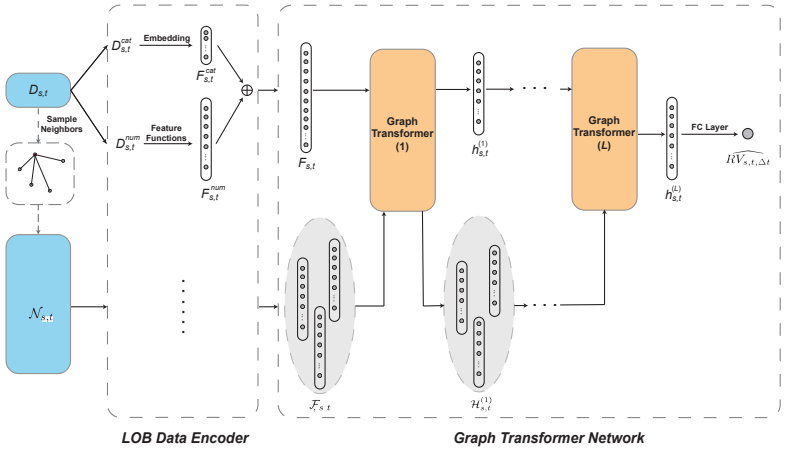


**Figure 1: Structure of our Graph Transformer Network for Volatility Forecasting. This illustration shows the prediction process for one node $node_{s,t}$. The LOB data encoder first transforms its LOB data into a fixed-dimension node feature $F_{s,t}$. GTN then takes this node feature and all the features from all other nodes connected with this node ($\mathcal{F}_{s,t}$). After $L$ Graph Transformer operations, we get the node embedding $h_{s,t}^{(L)}$ as output. We then use a fully connected layer to transform this node embedding into our final prediction $\widehat{RV_{s,t,\Delta T}}$.**

## 4.1 LOB data encoder

We first divide our LOB data $D_{s,t}$ into two parts: numerical data $D_{s,t}^{num}$ and categorical data $D_{s,t}^{cat}$. For numerical data, we can define different functions to aggregate them into numerical values. Suppose that we have $k_{num}$ such functions $e_1, ..., e_{k_{num}}$, we get $k_{num}$ features, and we put them into a single vector $F_{s,t}^{num}$ with

$$F_{s,t}^{num} = [f_{s,t}^1, ..., f_{s,t}^{k_{num}}]^\top \tag{6}$$

where $f_{s,t}^i = e_i(D_{s,t}^{num})$. $F_{s,t}^{num}$ is therefore a vector of size $k_{num} \times 1$.

Following Kercheval and Zhang [22], Bissoondoyal-Bheenick et al. [7] and Makinen et al. [28], we define a similar set of numerical features. We also add some other features which are suitable for our dataset. We show the detailed list of 73 features we use in our experiments in Appendix A.

For other categorical features, such as stock ticker, we simply adopt an embedding layer to transform them into a fixed-dimension vector $F_{s,t}^{cat}$. This vector is of size $k_{cat} \times 1$ where $k_{cat}$ is the embedding dimension we can choose.

We then concatenate these two vectors into one vector $F_{s,t} \in \mathbb{R}^{k \times 1}$, where $k = k_{num} + k_{cat}$. This operation is written as

$$F_{s,t} = F_{s,t}^{num} \oplus F_{s,t}^{cat} \tag{7}$$

where $\oplus$ denotes the concatenation operation.

## 4.2 Graph Transformer Network

As stated in Section 2.2, Graph Transformer operator shows a better performance compared with other structures. That is why we use it to build our network in this study.

We first build a graph with $m \times n$ nodes where $m$ is the number of timestamps and $n$ is the number of stocks. Each node $node_{s,t}$ represents the situation of stock $s$ at time $t$. Its initial node feature is $F_{s,t}$ (Eq. (7)) encoded by LOB data encoder. From the relationship $\mathcal{G}_s$ and $\mathcal{G}_t$, we can find all other nodes connected with $node_{s,t}$. We use $\mathcal{N}_{s,t}$ to denote all the connected nodes. For each $node_{s,t}$, the model takes both its own LOB features ($F_{s,t}$) and the LOB features from neighbors ($\mathcal{F}_{s,t}$) into account. It then forecasts the realized volatility based on both self node features and neighbor node features, instead of the traditional approach which considers only self node features.

The Graph Transformer operator for the $l$-th layer with $C$ heads is written as:

$$\widehat{h}_{s,t,c}^{(l+1)} = W_{1,c} h_{s,t}^{(l)} + \sum_{node_{i,j} \in \mathcal{N}_{s,t}} \alpha_{i,j,c} W_{2,c} h_{i,j}^{(l)} \tag{8}$$

$$h_{s,t}^{(l+1)} = \sigma(\oplus_{c=1}^{C} \widehat{h}_{s,t,c}^{(l+1)}) \tag{9}$$

Eq. (8) first calculates the output vector $\widehat{h}_{s,t,c}^{(l+1)}$ for one single head $c$, in which $h_{i,j}^{(l)} \in \mathbb{R}^{d_l \times 1}$ is the $l$-th layer hidden node embedding for $node_{i,j}$, $W_{1,c}, W_{2,c} \in (\mathbb{R}^{\widehat{d}_{l+1} \times d_l})^2$ are trainable parameters. $\alpha_{i,j,c}$ are attention coefficients associated with $node_{i,j}$ for head $c$. It is calculated via dot product attention [4] by

$$\alpha_{i,j,c} = \text{softmax}\left( \frac{(W_{3,c} h_{s,t}^{(l)})^\top (W_{4,c} h_{i,j}^{(l)})}{\sqrt{d_l}} \right) \tag{10}$$

where $W_{3,c}$ and $W_{4,c}$ are both trainable parameters of size $\widehat{d}_{l+1} \times d_l$.

We then use Eq. (9) to aggregate the output from all heads into a final output vector $h_{s,t}^{(l+1)}$ for the $l$-th layer. It is then used as the input for the $(l+1)$-th layer. In this equation, $\sigma$ is an activation function such as ReLU [19]. We show the structure of this operator in Figure 2.

We can then accumulate multiple layers of this structure to better retreive information. Suppose that our Graph Transformer Network (GTN) has $L$ layers in total, for each node, its initial node features $h_{s,t}^{(0)} = F_{s,t}$ will be transformed into a node embedding

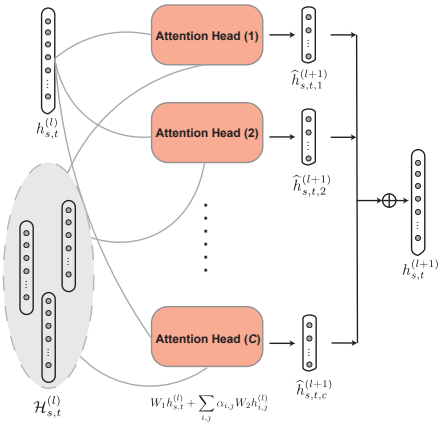


**Figure 2: Illustration of Graph Transformer operator. $\mathcal{H}^{l}_{s,t}$ represents the $l$-th layer hidden vectors for all the nodes in $\mathcal{N}_{s,t}$. We can then accumulate multiple layers of this structure to build a Graph Transformer Network.**

$h^{(L)}_{s,t} \in \mathbb{R}^{d_L \times 1}$. We then use a fully-connected layer to get the final predictions of realized volatility from the node embeddings. This operation is written as

$$\widehat{RV_{s,t,\Delta T}} = \sigma(W_0^\top h^{(L)}_{s,t}) \tag{11}$$

where $W_0 \in \mathbb{R}^{d_L \times 1}$ are trainable parameters in the fully connected layer.

We finally use RMSPE (Eq. (5)) as our loss function to evaluate the model and propagate back into the model.

# 5 EXPERIMENTS

## 5.1 LOB data

We use NYSE daily TAQ data[2] as our limit order book data. The data contain all the quotes (only first limit, i.e., best bid and best ask) and trades happening in the US stock exchanges. We select the entries concerning all the stocks included in the S&P 500 index[3] as our universe. Compared with other researches (5 stocks in Makinen et al. [28], 23 stocks in Rahimikia and Poon [32]), this large selection of around 500 stocks also covers some less liquid stocks. We will show that it is more difficult to have a good prediction on less liquid stocks in Section 5.5.1.

*5.1.1 Data Sampling.* Following Barndorff-Nielsen et al. [5] and Rahimikia and Poon [31], we first sample the data with a fixed frequency $T_f$. Since we focus on short-term volatility forecasting in this paper, we use a *one second* sampling frequency instead of 5 minutes used by Rahimikia and Poon [31] to forecast daily volatility. Our sampling strategy is as follows:

- For quote data, we snapshot the best ask price ($P^1_a$), best bid price ($P^1_b$), best ask size ($V^1_a$) and best bid size ($V^1_b$) for each stock at the end of each second.
- For trade data, we aggregate all the trades for each stock during each second. We record the number of trades ($N_t$), the total number of shares traded ($V_t$) and the volume weighted average price[4] ($P_t$) of all trades.

[2] https://www.nyse.com/market-data/historical/daily-taq
[3] https://www.spglobal.com/spdji/en/indices/equity/sp-500/#overview

**Table 1: An example of sampled quote and trade data for stock A on Jan. 3rd, 2017.**

**(a) sampled quote data**

| Date | Symbol | seconds | $P^1_b$ | $V^1_b$ | $P^1_a$ | $V^1_a$ |
|---|---|---|---|---|---|---|
| 1/3/2017 | A | 0 | 45.92 | 1 | 46.09 | 1 |
| 1/3/2017 | A | 1 | 45.92 | 3 | 46 | 2 |
| 1/3/2017 | A | 2 | 45.92 | 2 | 46 | 2 |
| 1/3/2017 | A | 5 | 45.92 | 2 | 46 | 1 |
| 1/3/2017 | A | 6 | 45.94 | 1 | 46.05 | 3 |

**(b) sampled trade data**

| Date | Symbol | seconds | $N_t$ | $V_t$ | $P_t$ |
|---|---|---|---|---|---|
| 1/3/2017 | A | 0 | 8 | 1500 | 45.802 |
| 1/3/2017 | A | 1 | 7 | 24959 | 45.94963 |
| 1/3/2017 | A | 2 | 1 | 300 | 45.96 |
| 1/3/2017 | A | 3 | 1 | 100 | 45.9544 |
| 1/3/2017 | A | 5 | 7 | 916 | 45.97817 |

An example of the sampled quote and trade data is shown in Table 1. In addition to the previously defined LOB fields, we also have *Date*, *Symbol* and *seconds*. The *seconds* field signifies the number of seconds after the market open. We note that the *seconds* are not continuous. This is because there are seconds for which there is no update in the LOB. For quote data it implies that the quote is the same as the last second, for trade data it implies that there is no trade in that second.

*5.1.2 Data Bucket.* As introduced in Section 3, our goal is to forecast the realized volatility $\Delta T$ seconds after a given timestamp $t$ based on the features built from a backward window of $\Delta T'$ seconds.

Hence, we need to build buckets which have the length of $\Delta T + \Delta T'$ seconds between $t - \Delta T'$ and $t + \Delta T$. In each bucket, we only use the returns between $t$ and $t + \Delta T$ to calculate our target $RV_{s,t,\Delta T}$, we use both returns and other information from LOBs between $t - \Delta T'$ and $t$ to build a set of features with LOB data encoder.

In our experiments on US stocks, we create 6 buckets for each stock each day. We select 10:00, 11:00, 12:00, 13:00, 14:00 and 15:00 EST as 6 different $t$. For the sake of simplicity, we use $\Delta T = \Delta T'$. We use three different $\Delta T$ of 600 seconds, 1200 seconds and 1800 seconds to show that our model is robust to this choice and is capable of forecasting volatility on different horizons. This choice also ensures that there is no overlap between buckets to avoid information leakage. The detailed experiment results are shown in Section 5.4.

*5.1.3 Data Split.* We split our LOB data into 3 parts: train, validation and test. We ensure that the validation set and the test set are no earlier than the training set to avoid forward looking. We also remove the buckets where there are no quotes or trades during $\Delta T$. Detailed statistics of our dataset are shown in Table 2.

## 5.2 Graph Building

As stated in Section 3, we consider both temporal ($\mathcal{G}_t$) and cross-sectional ($\mathcal{G}_s$) relationships among buckets. In this subsection, we

[4] $P_t = \frac{\sum_i^{N_t} P_i V_i}{\sum_i^{N_t} V_i}$ where $P_i$ is the price for trade $i$ and $V_i$ is the number of shares traded. $N_t$ is the total number of trades recorded during second $t$.

**Table 2: The statistics of the LOB data with $\Delta T = 600$.**

| | train | val | test |
|---|---|---|---|
| start | Jan-17 | Jan-20 | Jan-21 |
| end | Dec-19 | Dec-20 | Oct-21 |
| # time | 4,464 | 1,505 | 1,236 |
| # stock | 494 | 494 | 494 |
| # bucket | 2,141,108 | 743,068 | 607,770 |
| proportion | 61% | 21% | 18% |

introduce the relationships we build in our experiments, including one temporal relationship and three cross-sectional relationships.

*5.2.1 Temporal Relationship.* To construct the temporal relationship among the nodes, we only use LOB data. The intuition behind this temporal relationship is that at a new moment $t$, we check if there are moments in the history which are similar.

As introduced in Eq. (6), for each $node_{s,t}$, we can calculate $k_{num}$ features where $f^i_{s,t}$ is the $i$-th feature for this node. For each time $t$, we build time feature vector $Q^i_t \in \mathbb{R}^{n\times 1}$ with

$$Q^i_t = ([f^i_{1,t}, ..., f^i_{n,t}])^\top. \tag{12}$$

Given a time $t_0$, we calculate the RMSPE (Eq. (5)) of $Q^i$ between $t_0$ and all other $t < t_0$. We then choose the $K$-smallest RMSPE to form $K$ pairs of times, represented by $\mathcal{G}_{t_0} = [(t_0, t_1), ..., (t_0, t_K)]$. Then for each such pair $(t_i, t_j)$, we connect the two nodes where $s$ is the same and the time is $t_i$ and $t_j$ respectively. This can be written as:

$$\begin{aligned} &\forall (t_i, t_j) \in \mathcal{G}_{t_0}, \forall k \in [1, n], \\ &\text{connect } node_{s_k,t_i} \text{ and } node_{s_k,t_j} \end{aligned} \tag{13}$$

After this operation, we get $K \times n$ single-directed edges in the graph for $t_0$. We then repeat the same process for all $t$, and get $K \times n \times m$ edges in total.

In our study, we set $K = 2$. We use two features to get $2\times2\times n\times m$ edges in the graph, namely, the average quote WAP[5] for the first 100 seconds and the average quote WAP for the last 100 seconds in the bucket.

*5.2.2 Cross-sectional Relationship.* Unlike times, there are intrinsic relations among stocks since each stock represents a company in the real life. Hence, in addition to building relationship based on LOB features, we can also build the cross-sectional graph with external data. We build the following three graphs with feature correlation, stock sector and supply chain data.

*Stock Feature Correlation.* We use the same idea to build stock feature correlation as the time feature correlation introduced in 5.2.1.

We first build stock feature vector $Q'^i_s \in \mathbb{R}^{m\times 1}$ with

$$Q'^i_s = ([f^i_{s,1}, ..., f^i_{s,m}])^\top. \tag{14}$$

We then build the edges for $s_0$ with

$$\begin{aligned} &\forall (s_i, s_j) \in \mathcal{G}_{s_0}, \forall k \in [1, m], \\ &\text{connect } node_{s_i,t_k} \text{ and } node_{s_j,t_k} \end{aligned} \tag{15}$$

with $\mathcal{G}_{s_0} = [(s_0, s_1), ..., (s_0, s_{K'})]$ denoting the stock pairs which are among the $K'$-smallest feature RMSPE for stock $s_0$.

In this study, we set $K'$ to 2 and repeat the same process for all stocks with the same features as in 5.2.1 to build other $2\times2\times n\times m$ edges.

*Stock Sector.* In finance, each company is classified into a specific sector with Global Industry Classification Standard[6] (GICS). It is shown that the performances of stocks in the same sector are often correlated [39]. Hence, we simply connect a pair of stocks if they belong to the same sector. This is written as:

$$\begin{aligned} &\forall (s_i, s_j) \in \mathcal{G}_{sector}, \forall k \in [1, m] \\ &\text{connect } node_{s_i,t_k} \text{ and } node_{s_j,t_k} \end{aligned} \tag{16}$$

where $\mathcal{G}_{sector}$ is the ensemble of all the stock pairs in the same sector.

There are four granularities in GICS sector data: Sector, Industry Group, Industry, Sub-Industry. We can therefore construct four different types of edges with these GICS sector data. In our experiments, we use the Industry granularity as it gives a good performance with a reasonable number of edges.

*Supply Chain.* Supply chain describes the supplier-customer relation between companies and it is proved to be useful in multiple financial tasks such as risk management [42] and performance prediction [13]. We use the supply chain data from Factset[7] to build this graph. We connect two companies if they have a supplier-customer relationship in the training period. This is described as:

$$\begin{aligned} &\forall (s_i, s_j) \in \mathcal{G}_{supply}, \forall k \in [1, m] \\ &\text{connect } node_{s_i,t_k} \text{ and } node_{s_j,t_k} \end{aligned} \tag{17}$$

where $\mathcal{G}_{supply}$ is the ensemble of all the supplier-customer relations among the stocks.

We provide detailed statistics of each type of relationship we built in Table 3 with $\Delta T = 600$. In addition to using these relations separately, we can also join these relations by simply putting all the edges together in the same graph. We note that it is not exactly the sum of all individual edge counts since there are duplicated edges. We will show that combining the edges can help improve the result in Section 5.4. It is also worth noting that although there are many edges included in our experiments, we do not need extra memory as we can train them in suitable batches. The pseudocode used to build all four relationships is included in a supplemental document.

**Table 3: The number of edges in each relationship in the training set.**

| Type | Relation | **Number of edges** |
|---|---|---|
| Temporal | Feature Correlation | 8.36M |
| Cross-sectional | Feature Correlation | 8.21M |
| | Sector | 47.86M |
| | Supply Chain | 22.22M |
| Total | | 76.33M |

[5]Weighted Average Price, defined as $\frac{P^1_b V^1_a + P^1_a V^1_b}{V^1_a + V^1_b}$

[6]https://www.msci.com/our-solutions/indexes/gics

[7]https://www.factset.com/marketplace/catalog/product/factset-supply-chain-relationships

## 5.3 Experiment Setup

In order to prove the effectiveness of our model structure, we also include the performance of some other widely used models as our benchmarks.

- `Naïve Guess`: We simply use the realized volatility between $t - \Delta T'$ and $t$ to predict the target. It is written as $\widehat{RV_{s,t,\Delta T}} = RV_{s,t-\Delta T',\Delta T'}$.
- `HAR-RV`: Heterogeneous AutoRegressive model of Realized Volatility. A simple but effective realized volatility prediction model proposed by Corsi [14].
- `LightGBM`: A gradient boosting decision tree model introduced by Ke et al. [21]. It is proven to be highly effective on tabular data.
- `MLP`: Multi-Layer Perception network [35]. We build a fully connected neural network with three layers, which have 128, 64, 32 hidden units respectively.
- `TabNet`: A neural network proposed by Arik and Pfister [3] which specializes in dealing with tabular data.
- `Vanilla GTN-VF`: Our Graph Transformer Network for Volatility Forecasting trained without any relationship information.

In addition, we compare the model performance with different relations individually to demonstrate how different types of relational data help improve the result compared with Vanilla GTN-VF and other benchmarks. These variants include Cross-sectional Feature Correlation (GTN-VF Cross FC), Temporal Feature Correlation (GTN-VF Temp FC), Cross-sectional Sector relationship (GTN-VF Sector) and Cross-sectional Supply Chain relationship (GTN-VF Cross Supply Chain). The full GTN-VF includes all four types of relations.

MLP, TabNet and all the variants of our GTN-VF model are implemented in PyTorch [30] with Adam optimizer [23]. For our GTN-VF, we use a 3-layer ($L = 3$) Graph Transformer Network with 8 heads ($C = 8$). All three layers have 128 channels ($d_1 = d_2 = d_3 = 128$). We embed our numerical features into a 73-dimension vector ($k_{num} = 73$, Section 4.1) and categorical features into a 32-dimension vector ($k_{cat} = 32$). Other baseline models are implemented without deep learning framework.

In order to provide a fair comparison and guard against hyperparameter hacking, we sweep over the same set of hyperparameters for all GTN-VF variants and choose the best setting for each variant according to the performance on the validation set. We then fix these parameters for all experiments on the test set. The appendix contains further implementation details on both baseline models and GTN-VF models.

## 5.4 Experiment Results

The detailed results of our experiments are shown in Table 4. We can see that our full GTN-VF model with all four types of relational information outperforms all baseline models and each type of relational information individually on all prediction horizons. The improvement is significant. In average, we gain 6% in RMSPE compared with Naïve Guess and 2% compared with the best baseline model TabNet on test set. It proves that our GTN model structure and relation building methods are effective.

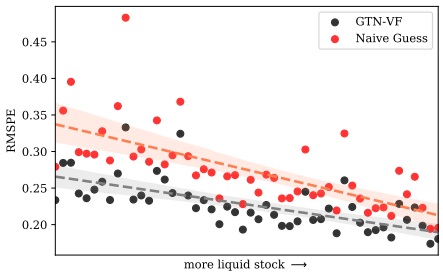


**Figure 3: The relationship between stock liquidity and prediction RMSPE. The dots denote the RMSPE for the buckets and the dashed lines are the trend lines calculated with linear regression. This figure is based on test result and $\Delta T = 600$.**

In terms of networks, all relations are useful since they all show improvement compared with the vanilla model. The temporal feature correlation shows the most predicting power, contributing 1.2% RMSPE gain, while the Sector relation only contributes 0.7% improvement when the window is set to 600 seconds, although it has the largest number of edges. This can be explained by the 'noise' included in this type of information since we need to connect every two stocks in the same sector although not all of them have significant connection. However, feature correlation only selects the two most related stocks or times, making it more discriminational when building edges. This suggests that if we have the constraint on the number of edges in the graph, quality is more important than quantity. On the other hand, these relations are complementary to each other, adding more relations on top of existing relations can help improve the prediction if we have enough computing power.

It is also worth noting that when we forecast the realized volatility with a longer forward looking and backward looking window, the result is better. This is simply because the same unpredictable volatility jump causes more volatility changes in shorter forecasting horizon [27].

## 5.5 Ablation Studies

*5.5.1 Prediction accuracy and stock liquidity.* In general, it is more difficult to have a good prediction on less liquid stocks because there are fewer market participants for them. One sudden change in quote or trade can cause a significant volatility jump, which is difficult to foresee. We analyze the result to understand where the improvement comes from.

We first split the stocks into 50 buckets according to their average daily turnover, which represents the liquidity of a stock. We calculate the RMSPE in each bucket for both Naïve Guess and GTN-VF. The result is illustrated in Figure 3.

First, we notice that more liquid stocks have smaller RMSPE for both models, which is intuitive. The prediction for the most liquid stocks is 5% better than the least liquid stocks in terms of RMSPE, which is a large margin in realized volatility forecasting. We can also see that our graph based model has more improvement on the less liquid stocks (around 8% for the least liquid stocks and 2% for the most liquid stocks), although it is more effective than Naïve Guess on all scenarios.

**Table 4: RMSPE values of all the models on both validation set and test set.**

| | $\Delta T = 600$ | | $\Delta T = 1200$ | | $\Delta T = 1800$ | |
|---|---|---|---|---|---|---|
| Model | val | test | val | test | val | test |
| Naïve Guess | 0.2911 | 0.2834 | 0.2650 | 0.2628 | 0.2296 | 0.2364 |
| HAR-RV | 0.2684 | 0.2612 | 0.2149 | 0.2061 | 0.1968 | 0.1939 |
| LightGBM | 0.2583 | 0.2492 | 0.2414 | 0.2035 | 0.2349 | 0.1963 |
| MLP | 0.2431 | 0.2514 | 0.2200 | 0.2308 | 0.2270 | 0.1999 |
| TabNet | 0.2517 | 0.2478 | 0.2212 | 0.1996 | 0.2204 | 0.2019 |
| Vanilla GTN-VF | 0.2457 | 0.2498 | 0.2229 | 0.2251 | 0.2092 | 0.2160 |
| GTN-VF Cross FC | 0.2414 | 0.2382 | 0.2162 | 0.2196 | 0.2066 | 0.2046 |
| GTN-VF Temp FC | 0.2326 | 0.2358 | 0.1974 | 0.1921 | 0.1896 | 0.1853 |
| GTN-VF Cross Sector | 0.2406 | 0.2422 | 0.2067 | 0.2248 | 0.2071 | 0.2091 |
| GTN-VF Cross Supply Chain | 0.2430 | 0.2411 | 0.2105 | 0.2244 | 0.2057 | 0.2000 |
| GTN-VF Cross FC + Temp FC | 0.2326 | 0.2306 | 0.1936 | 0.1917 | 0.1848 | 0.1802 |
| GTN-VF | **0.2314** | **0.2287** | **0.1916** | **0.1892** | **0.1809** | **0.1798** |

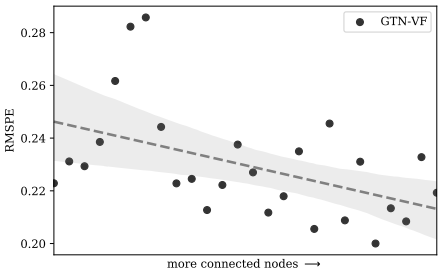


**Figure 4: The relationship between node connection and RMSPE. This figure is based on test result and $\Delta T = 600$.**

*5.5.2 Prediction accuracy and node connection.* We also investigate how our model performs on different nodes. We use the same approach as in Section 5.5.1 by splitting nodes into buckets according to the number of edges connected to each node. We can see from Figure 4 that more connected nodes usually have better RMSPE result, with a 2% difference between the most connected and the least connected. This can be explained by the fact that more connected nodes make decision based on more information from their neighbors, while the nodes with fewer or no connections can only rely on the information from themselves. This phenomenon proves again the effectiveness of our graph based method.

## 6 CONCLUSION

We forecast the short-term realized volatility in a multivariate approach. We design a graph based neural network: Graph Transformer Network for Volatility Forecasting which incorporates both features from LOB data and relationship among stocks from different sources. Through extensive experiments on around 500 stocks, we prove the that our method outperforms other baseline models, both univariate and multivariate. In addition, the model structure allows to combine a large number of relations, the study of the effectiveness of other relational data is open for future researches.

## ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support of the Chaire *Machine Learning & Systematic Methods in Finance* of Ecole Polytechnique. The authors would like to thank Mathieu Rosenbaum and Jianfei Zhang for their valuable advice during this work. A supplemental document is included with this paper with more details regarding the implementation of our GTN-VF.

## REFERENCES

[1] Torben G Andersen and Tim Bollerslev. 1998. Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International economic review* (1998), 885–905.
[2] Torben G Andersen, Tim Bollerslev, Peter Christoffersen, and Francis X Diebold. 2005. Volatility forecasting.
[3] Sercan O Arık and Tomas Pfister. 2020. Tabnet: Attentive interpretable tabular learning. *arXiv* (2020).
[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[5] Ole E Barndorff-Nielsen, P Reinhard Hansen, Asger Lunde, and Neil Shephard. 2009. Realized kernels in practice: Trades and quotes.
[6] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).
[7] Emawtee Bissoondoyal-Bheenick, Robert Brooks, and Hung Xuan Do. 2019. Asymmetric relationship between order imbalance and realized volatility: Evidence from the Australian market. *International Review of Economics & Finance* 62 (2019), 309–320.
[8] Tim Bollerslev, Nour Meddahi, and Serge Nyawa. 2019. High-dimensional multivariate realized volatility estimation. *Journal of Econometrics* 212, 1 (2019), 116–136.
[9] Timothy J Brailsford and Robert W Faff. 1996. An evaluation of volatility forecasting techniques. *Journal of Banking & Finance* 20, 3 (1996), 419–438.
[10] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).
[11] Andrea Bucci. 2020. Cholesky–ANN models for predicting multivariate realized volatility. *Journal of Forecasting* 39, 6 (2020), 865–876.
[12] John Y Campbell, Sanford J Grossman, and Jiang Wang. 1993. Trading volume and serial correlation in stock returns. *The Quarterly Journal of Economics* 108, 4 (1993), 905–939.
[13] Qinkai Chen and Christian-Yann Robert. 2021. Graph-Based Learning for Stock Movement Prediction with Textual and Relational Data. *arXiv preprint arXiv:2107.10941* (2021).
[14] Fulvio Corsi. 2009. A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics* 7, 2 (2009), 174–196.
[15] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016), 3844–3852.

[16] Jim Gatheral and Roel CA Oomen. 2010. Zero-intelligence realized variance estimation. *Finance and Stochastics* 14, 2 (2010), 249–283.
[17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.
[18] Corrado Gini. 1921. Measurement of inequality of incomes. *The economic journal* 31, 121 (1921), 124–126.
[19] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 315–323.
[20] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.
[21] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017), 3146–3154.
[22] Alec N Kercheval and Yuan Zhang. 2015. Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance* 15, 8 (2015), 1315–1329.
[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[24] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[25] CK Kwan, WK Li, and K Ng. 2005. A multivariate threshold GARCH model with time-varying correlations. *Econometric reviews* 24 (2005).
[26] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).
[27] Feng Ma, Yin Liao, Yaojie Zhang, and Yang Cao. 2019. Harnessing jump component for crude oil volatility forecasting in the presence of extreme shocks. *Journal of Empirical Finance* 52 (2019), 40–55.
[28] Ymir Mäkinen, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. 2019. Forecasting jump arrivals in stock prices: new attention-based network architecture using limit order book data. *Quantitative Finance* 19, 12 (2019), 2033–2050.
[29] Peter Malec. 2016. A Semiparametric Intraday GARCH Model. (2016).
[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
[31] Eghbal Rahimikia and Ser-Huang Poon. 2020. Big data approach to realised volatility forecasting using HAR model augmented with limit order book and news. *Available at SSRN 3684040* (2020).
[32] Eghbal Rahimikia and Ser-Huang Poon. 2020. Machine learning for realised volatility forecasting. *Available at SSRN 3707796* (2020).
[33] Eduardo Ramos-Pérez, Pablo J Alonso-González, and José Javier Núñez-Velázquez. 2021. Multi-Transformer: A New Neural Network-Based Architecture for Forecasting S&P Volatility. *Mathematics* 9, 15 (2021), 1794.
[34] Mathieu Rosenbaum and Jianfei Zhang. 2022. On the universality of the volatility formation process: when machine learning and rough volatility agree. *arXiv preprint arXiv:2206.14114* (2022).
[35] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. *Learning internal representations by error propagation.* Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.
[36] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, and Rajiv Shah. 2020. Deep Attentive Learning for Stock Movement Prediction from Social Media Text and Company Correlations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 8415–8426.
[37] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509* (2020).
[38] Justin Sirignano and Rama Cont. 2019. Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance* 19, 9 (2019), 1449–1459.
[39] Raman Vardharaj and Frank J Fabozzi. 2007. Sector, style, region: Explaining stock allocation performance. *Financial Analysts Journal* 63, 3 (2007), 59–70.
[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).
[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[42] Shuo Yang, Zhiqiang Zhang, Jun Zhou, Yang Wang, Wang Sun, Xingyu Zhong, Yanming Fang, Quan Yu, and Yuan Qi. 2020. Financial Risk Analysis for SMEs with Graph-based Supply Chain Mining.. In *IJCAI*. 4661–4667.
[43] Lan Zhang. 2006. Efficient estimation of stochastic volatility using noisy observations: A multi-scale approach. *Bernoulli* 12, 6 (2006), 1019–1043.
[44] Bin Zhou. 1996. High-frequency data and volatility in foreign-exchange rates. *Journal of Business & Economic Statistics* 14, 1 (1996), 45–52.

# A LIST OF NODE FEATURES

We introduce the features we use in our experiments.

In each bucket, we have $\Delta T'$ lines as training data. We first calculate an indicator for each line, we then aggregate these indicators in the same bucket with an aggregation function (aggregator). In such way, we have one value per indicator per aggregator as one feature for the bucket. The full list of indicators and aggregators are listed in Table 5.

For some important indicators, we also calculate their progressive features. It means that instead of applying an aggregator on all the lines, we apply it on the lines between 0 and $\Delta T'/6$, $\Delta T'/3$, $\Delta T'/2$, $2\Delta T'/3$, $5\Delta T'/6$, $\Delta T'$. In such way, we have 6 features per indicator per aggregator for a progressive feature. This is shown in the column Progressive in Table 5.

We define our aggregation functions as follows. We use $a_i$ to denote the $i$-th line in the bucket and $N$ represents the total number of lines.

- percentage difference
$$\frac{\sum_i^N \mathbf{1}_{a_i \neq a_{i-1}}}{N}$$
- median deviation
$$median(|a_i - \overline{a}|)$$
- energy
$$\frac{1}{N}\sum_i^N a_i^2$$
- InterQuartile Range (IQR)
$$Q_{75}(a) - Q_{25}(a)$$
where $Q_i(a)$ denotes the $i$-th percentile value for the series $a$.

# B DETAILS ON EXPERIMENT SETUP

*In all baseline models, except for Naïve guess and HAR-RV that do not need features, the input features ($F_{s,t}$) are the same as GTN-VF, including 73 numerical features and 1 categorical feature. The categorical feature embedding is of size 32 whenever applicable. All loss functions are set to RMSPE. The hyperparameters are chosen based on the validation set performance, and the results presented are based on the best set of hyperparameters.*

## B.1 Model Details

*B.1.1 HAR-RV.* The model is written as

$$\widehat{RV_{s,t+1d,\Delta T}} = c + \beta^{(d)} RV_{s,t}^{(d)} + \beta^{(w)} RV_{s,t}^{(w)} + \beta^{(m)} RV_{s,t}^{(m)} \quad (18)$$

where $RV_{s,t}^{(d)}$, $RV_{s,t}^{(w)}$ and $RV_{s,t}^{(m)}$ are respectively the daily, weekly and monthly average realized volatilities before $t$. $\beta^{(d)}$, $\beta^{(w)}$, $\beta^{(m)}$ and $c$ are coefficients determined by linear regression. The

**Table 5: The list of features built from LOB data.**

| Type | Notation | Description | Aggregators | Progressive | Count |
|---|---|---|---|---|---|
| Quote | $\frac{P_b^1 V_a^1 + P_a^1 V_b^1}{V_a^1 + V_b^1}$ | WAP | mean, std, gini[18]<br>mean of first 100, mean of last 100 | N | 5 |
| | $P_a^1$ | Ask Price | % difference | N | 1 |
| | $P_b^1$ | Bid Price | % difference | N | 1 |
| | $\frac{P_a^1 - P_b^1}{P_a^1 + P_b^1}$ | Price Relative Spread | mean, std, gini | N | 3 |
| | $WAP - P_b^1$ | WAP bid difference | mean, std, gini | N | 3 |
| | $log(\frac{WAP_i}{WAP_{i-1}})$ | Return | realized volatility | Y | 6 |
| | $log(\frac{WAP_i}{WAP_{i-1}})^2$ | Squared Return | std, gini | N | 2 |
| | $\frac{V_a^1 - V_b^1}{V_a^1 + V_b^1}$ | Size Relative Spread | mean, std, gini | N | 3 |
| | $V_a^1$ | Ask Size | % difference | N | 1 |
| | $V_b^1$ | Bid Size | % difference | N | 1 |
| | $V_a^1 / V_a^1$ | Normalized Ask Size | mean, std, gini | N | 3 |
| | $V_a^1 + V_b^1$ | Total Size | sum, max | N | 2 |
| | $\lvert V_a^1 - V_b^1 \rvert$ | Size Imbalance | sum, max | N | 2 |
| Trade | $P_t$ | Price | % greater than mean, % less than mean<br>median diviation, energy, IQR | N | 5 |
| | $log(\frac{P_{t,i}}{P_{t,i-1}})$ | Return | realized volatility | Y | 6 |
| | | | % greater than 0, % less than 0 | N | 2 |
| | $log(\frac{P_{t,i}}{P_{t,i-1}})^2$ | Squared Return | std, gini | N | 2 |
| | $V_t$ | Size | sum | Y | 6 |
| | | | max, median diviation, energy, IQR | N | 4 |
| | $t$ | Seconds | count | Y | 6 |
| | $N_t$ | Order Count | sum | Y | 6 |
| | | | max | N | 1 |
| | $P_t \times V_t$ | Amount | sum, max | N | 2 |
| | Total | | | | 73 |

weekly and monthly average realized volatilities are calculated with $RV_{s,t}^{(w)} = \frac{1}{5}(RV_{s,t} + ... + RV_{s,t-4d})$ and $RV_{s,t}^{(w)} = \frac{1}{21}(RV_{s,t} + ... + RV_{s,t-20d})$.

In our implementation, we calibrate one HAR-RV model for each sampling time since we find a better result than mixing all the sampling times and calibrate only one model. Therefore, we have 6 different HAR-RV models for 10:00, 11:00, 12:00, 13:00, 14:00 and 15:00.

*B.1.2 LightGBM.* We use the LightGBM package[8] to implement this baseline model. We use gradient boosting decision tree (GBDT) algorithm and set its learning to 0.1. The model is trained for a maximum of 1,000 iterations and the training process is stopped earlier if there is no improvement on the validation set for 50 iterations.

*B.1.3 MLP.* There are three hidden layers in this fully-connected neural network. Their sizes are 128, 64 and 32 respectively. The learning rate of Adam optimizer is set to 0.01, and the batch size is set to 2048. The model is trained for a maximum of 200 iterations and the training process is stopped earlier if there is no improvement on the validation set for 20 iterations.

It is worth noting that we normalize our features to values between -1 and 1 before training. This is to avoid overflow in the computation process.

[8]https://github.com/microsoft/LightGBM

*B.1.4 TabNet.* We set the width of both prediction layer and attention embedding to 16. All other settings, including learning rate, batch size, training epochs, early stopping and feature normalization are the same as the MLP model.

*B.1.5 GTN-VF and its variants.* In our experiments, the GTN-VF and its variants share the same set of hyperparameters and model dimension. The model dimension is already introduced in Section 5.3.

During training, we set the batch size to 2048 and the initial learning rate to 0.001. The model is trained for a maximum of 100 iterations and the training process is stopped earlier if there is no improvement on the validation set for 20 iterations. We also reduce the learning rate by half if there is no improvement for 5 epochs.

In the neighbor sampling process, we sample all the connected neighbors without a maximum limit.

## B.2 Hardware

Except for Naïve Guess, HAR-RV and LightGBM which can be quickly trained without GPU, we ran the experiments on a single machine with one NVIDIA Tesla V100 GPU (16GB RAM and 32GB/s bandwidth), 8 cores of Intel Xeon CPU (Broadwell E5-2686 v4) and 61GB of RAM. In average, a GTN-VF with all four relationships takes one hour to train. For comparison, the training time for MLP is 20 minutes and 3 hours for TabNet.