Temporal Bipartite Graph Neural Networks for Bond Prediction

Dan Zhou
dz239@njit.edu
New Jersey Institute of technology
Newark, New Jersey, USA

Ajim Uddin
au76@njit.edu
New Jersey Institute of technology
Newark, New Jersey, USA

Zuofeng Shang
zshang@njit.edu
New Jersey Institute of technology
Newark, New Jersey, USA

Xinyuan Tao
xinyuan.tao@njit.edu
New Jersey Institute of technology
Newark, New Jersey, USA

Dantong Yu
dtyu@njit.edu
New Jersey Institute of technology
Newark, New Jersey, USA

ABSTRACT

Understanding bond (debt) valuation and predicting future prices are of great importance in finance. Bonds are a major source of long-term capital in U.S. financial markets along with stocks. However, compared with stocks, bonds are understudied. One main reason is the infrequent trading in the secondary market, which results in irregular intervals and missing observations. This paper attempts to overcome this challenge by leveraging network information from bond-fund holding data and proposes a novel method to predict bond prices (yields). We design the temporal bipartite graph neural networks (TBGNN) with self-supervision regularization that entails multiple components: the bipartite graph representation module of learning node embeddings from the bond and fund interactions and their associated factors; the recurrent neural network module to model the temporal interactions; and the self-supervised objective to regularize the unlabeled node representation with graph structure. The model adopts a minibatch training process (Minibatch Stochastic Gradient Descent) in a deep learning platform to alleviate the model complexity and computation cost in optimizing different modules and objectives. Results show that our TBGNN model provides a more accurate prediction of bond price and yield. It outperforms multiple existing graph neural networks and multivariate time series methods: improving R^2 by 6%-51% in bond price prediction and 5%-70% in bond yield prediction.

CCS CONCEPTS

• **Computing methodologies** → *Regularization*; *Feature selection*;
• **Applied computing** → *Economics*; *Forecasting*.

KEYWORDS

FinTech; Graph Neural Networks; Bipartite Network; Bond Prediction

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

ACM ISBN 978-1-4503-9376-8/22/11…$15.00
https://doi.org/10.1145/3533271.3561751

ACM Reference Format:
Dan Zhou, Ajim Uddin, Zuofeng Shang, Xinyuan Tao, and Dantong Yu. 2022. Temporal Bipartite Graph Neural Networks for Bond Prediction. In *proceedings of ACM International Conference on AI in Finance (ICAIF'22).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3533271.3561751

1 INTRODUCTION

Financial time-series prediction gets increasing attention with considerable success from the machine learning community. While most studies focus on stock price prediction [3, 4, 16, 29], bond price prediction is significantly understudied. In the U.S., the bond market is one of the primary sources of long-term capital for companies and has an equivalent size to the stock market. By 2022 Q1, the total value of outstanding securities in the U.S. bond market is $48 trillion, and that in the stock market is $49 trillion.[1] U.S. bond market is therefore too large to neglect. In addition, stocks and corporate bonds have the same underlying firms, but the annual issuance of corporate bonds is far more than the stock issuance (IPO). In 2022 Q1, the year-to-date new issuance of corporate bonds is $522 million while only $28.9 million in stocks. Therefore, understanding bond valuation and predicting future prices have great potentials for firms to make financing decisions, for investors to make investments, and for regulators to monitor financial markets.

The less research on bond valuation and price prediction than on the stock market can be attributed to the limited availability of historical data. While stock trading data go back to the middle of the 1900s, corporate bond trading data became easy to access in July 2002 after the introduction of TRACE.[2] Moreover, corporate bonds are not traded regularly, and many investors of corporate bonds buy and hold until maturity. Numerous bonds are traded only a couple of times in the secondary market during their lifetime. For instance, in our sample data from 2002 to 2016, more than 90% of corporate bonds have no more than two trades during the outstanding periods. As a result, the temporal data for bond historical prices often have irregular time intervals between observations (with trades in some months and without trades in other months). During non-traded months, no price is observed, preventing the bond pricing discovery. Given the nature of bond data, how to extract useful information without regularly observed trades is the key.

[1] See information from SIFMA at https://www.sifma.org/resources/research/research-quarterly-fixed-income-issuance-and-trading/ and https://www.sifma.org/resources/research/research-quarterly-equities/
[2] See information about the TRACE database at https://www.finra.org/investors/learn-to-invest/types-investments/bonds/types-of-bonds/corporate-bonds

This paper proposes a novel Temporal Bipartite Graph Neural Networks (TBGNN) for predicting bond prices and yields by tackling the challenge of irregular data intervals. The rationale behind the bipartite graph neural network is the interconnected network among financial instruments. The price of financial instruments often moves together due to the interconnected idiosyncratic factors, market participants, and global network conditions. Recent studies show that graph neural network offers a better approach for Spatio-temporal analysis[7, 9, 17, 22], learns node representations among network participants [20], and builds better predictive models[2, 34]. In addition, studies show that graph frameworks allow better information retrieval from the interconnected financial network and improve stock return predictability[11, 19, 21, 28]. Inspired by their success, we propose a bipartite temporal network to model Spatio-temporal interconnected relations among bonds by propagating information through their common bondholders (fund managers). They often possess information leveraging their skills, training, and network connections. Managers holding the same bonds likely have a common information set or similar perception of the bonds.

We compensate for the missing trading information by integrating complete information from the spatial domain (quarterly bond holding data) using the bipartite graph representation framework. For each quarter, fund managers and their holding bonds are represented efficiently with a bipartite network, shown in Figure 1(a), where each fund in the node set V is connected to the bonds in the node set U through their co-investment in the same bond. Subsequently, each bond in node-set U is connected to other bonds in the node set U indirectly through their co-ownership. The bipartite affinity matrix consists of the investment of a given bond by a given fund, which represents the edge weights. Over time, the temporal information in each bipartite network captures the information of a bond's prospective price movement by market participants.

The proposed TBGNN model consists of three modules. The first module attempts to learn bipartite graph embedding for each set of nodes, i.e., bonds and fund managers for each time point, and helps to overcome the problem associated with intermittent pricing and yield data by extracting information from its neighbors and network properties. The second module uses a Long Short Temporal Memory (LSTM) neural network to learn temporal dependency between each time point. Finally, in the third module, spatial and temporal dependency from the first two modules are combined to predict the future target variables.

We evaluate the proposed model using 21,747,966 monthly observations by 15,411 corporate bonds from July 2002 to December 2016 and show that TBGNN is superior to other recurrent and graph neural network-based models in both bond price and yield prediction. In the bond price prediction, our algorithm outperforms recurrent neural networks LSTM and GRU by 51.49% and 32.87% and graph neural network GCN and GraphSage by 38.12% and 6.89%, respectively. We list the major contributions as follows:

- Network plays an important role in propagating information among financial instruments. The outperformance in prediction confirms that the bipartite graph neural networks capture pricing information even without observed trades.
- TBGNN is highly efficient when incorporating new bonds into the existing bipartite graph. We use the fund nodes as a bridge to build the interdependence among bonds and avoid quadratic pair-wise relation calculation among bonds. We design sparse network technology to scale our algorithm to large networks with ~thirty-thousand nodes and millions of interactions.
- Our approach resolves the challenge of the lack of pre-existing networks and offers a novel network construction and subsequent Graph Neural Networks among relevant entities from ubiquitous tabular investment data.
- The self-supervised regularization overcomes the challenge of highly sparse financial data. The self-supervision framework uses existing graph structures themselves, in addition to the scarce target variables, to learn the representation of unlabeled nodes.
- Our proposed TBGNN is an effective model for bond price prediction as it consistently outperforms the compared benchmarks by impressive margins.

2 PROBLEM DEFINITION

Corporate bond price reflects the present value of all future cash flows, usually including semi-annual coupon payments and the face value ($1,000 in most cases) at the maturity. The bond yield, commonly referred to as yield to maturity (YTM), is the expected average return from now to the end of maturity. To predict bond prices and yields, we represent the time series bond data as a third-order bond factor tensor $\mathcal{X} \in \mathbb{R}^{n \times k \times T}$ with n unique bonds, each of which has k factors over T time intervals. For each target variable, let $Y \in \mathbb{R}^{n \times T}$ be the targets or the response variable of n bonds at T time intervals that depend on the bond factor tensor. Figure 1-c shows the frontal slice (matrix) $\mathcal{X}_{:,:,t} \in \mathbb{R}^{n \times k}$ of the bond factor tensor and the associated targets $Y_{:,t} \in \mathbb{R}^{n}$ at time t.

This paper utilizes quarterly bond-fund investment data to extract price information without the presence of observed trades. The bond-fund investment dataset shares the same bond entities and provides complementary information to minimize the noise in the irregular bond trading data. It can be represented as a bipartite graph time series that consists of T dynamic bipartite graphs $\mathcal{G}^{1:T} = \{\mathcal{G}^1, \mathcal{G}^2, \cdots, \mathcal{G}^T\}$. At each time point, the graph $\mathcal{G}^t = (U, V, \mathcal{E}^t, \mathcal{A}_{:,:,t}))$ has two disjoint node sets: $U = \{u_1, \cdots, u_n\}$ representing n bonds and $V = \{v_1, \cdots, v_m\}$ denoting m institutional funds invested on these bonds (shown in Figure 1-b). The affinity matrix $\mathcal{A}_{:,:,t}$ in Figure 1-a uses investment values to represent the edge weights of $\mathcal{G}^t$. The edge set $\mathcal{E}^t$ in each graph G^t consists of edges only between two different types of nodes and represents non-zero investment values on bonds by funds. We combine the information properties of the two aforementioned datasets to build an efficient end-to-end graph neural network model as shown in Figure 1-d.

Given the temporal bipartite graph $\mathcal{G}^{1:T}$, the input attributes $\mathcal{X} \in \mathbb{R}^{n \times k \times T}$ for node set U, and the target variables $Y \in \mathbb{R}^{n \times T}$ of U for T time steps, we define a node-level prediction task at each time step t on the temporal bipartite graph as follows:
Input: lag-p historical snapshots $\mathcal{G}^{t-p+1:t}$ and the node features $\mathcal{X}_{:,:,t-p+1:t} \in \mathbb{R}^{n \times k \times p}$ in node set U in lag-p steps.

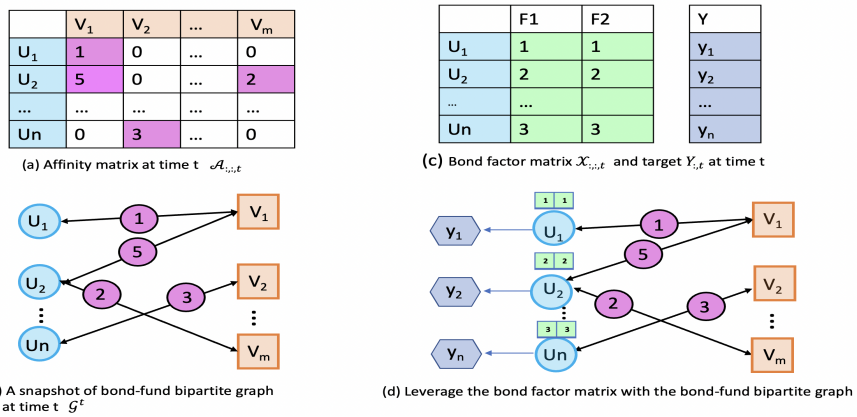


Figure 1: We view any tabular matrix as a bipartite graph $\mathcal{G}^t$ that encodes the interactions (edges) between the rows and columns with the attribute values in each row corresponding to the edge weights $\mathcal{A}_{:,:,t}$ (Figure 1-a). When the number of features is fixed, the matrix size is linear to the number of rows, which makes the corresponding bipartite network $\mathcal{G}^t$ in Figure 1-b highly efficient and eliminates the quadratic complexity. When we use sparse matrix representation and only keep non-zero values, we further reduce the network size. Combined with the side information from other data sources, for example $\mathcal{X}_{:,:,t}$ in Figure 1-c, we formulate a graph neural network that contains n nodes $U_1, U_2, \ldots, U_n$ with feature $\mathcal{X}_{:,:,t}$ and network $\mathcal{G}^t$.

Predict: $Y_{:,t+1:t+I} \in \mathbb{R}^{n \times I}$, the targets for up to I time steps in future. The objective is to minimize the mean squared error (MSE) of:

$$\mathcal{L}_{TP} = \sum_{i=1}^{I} \frac{1}{\sum_j (w_{t+i})_j} \left\| w_{t+i} \odot (\hat{Y}_{:,t+i} - Y_{:,t+i}) \right\|_F^2 \qquad (1)$$

where $w_{t+i} \in \{0,1\}^n$ is the indicate vector of the available targets among n bonds at time $t+i$. In this paper, we focus on the next time step prediction.

3 METHODOLOGY

Figure 2 presents the overview of the proposed TBGNN framework. We map the spatiotemporal observations of predictors (X) and targets (Y) onto the nodes in temporal bipartite graphs and attempt to learn the node representations in the lag-p graph snapshots $\mathcal{G}_{t'}$, where $t - p + 1 \le t' \le t$. Afterward, we input these node representations into the recurrent neural networks to predict the target variables from the learned lag-p node embeddings. Due to the scarcity of target variables, our framework incorporates the self-learning capability to compensate for the supervision starvation problem. The framework outlined in Alg. 1 contains four modules: 1) Graph representation learning module (Section 3.2, colored in green in Figure 2, and Algorithm 1 lines 6-7) for extracting node embedding from each of p snapshots $\mathcal{G}_{t'} \in \mathcal{G}^{t-p+1:t}$ and producing two node embedding matrices: Bond matrix $H^t_{G,U} \in \mathbb{R}^{n \times d}$ and Fund matrix $H^t_{G,V} \in \mathbb{R}^{m \times d}$, for two types of nodes in each bipartite graph (where d is the node dimension); 2) Temporal learning module (Section 3.3, colored in orange in Figure 2, and Algorithm 1 line 8) that employs a Long Short Temporal Memory (LSTM) to scan through p node embedding matrices and extract temporal patterns; 3) the target prediction module Section 3.4, colored in blue in Figure 2, and Algorithm a lines 9) that uses the hidden state of $H^t_{R,U}$ to predict the target variable at time step $t+1$; and 4) Self-supervised graph topology reconstruction module in (Section 3.5, colored in yellow in Figure 2, and Algorithm 1 lines 11) that takes the output matrix from the second module and reconstructs the bipartite graph at the concurrent time steps.

3.1 Initialize Node Features in Bipartite Graph

The bipartite graph representation learning module considers each individual bipartite graph $\mathcal{G}^{t'}$ in the lag-p snapshots on investment networks $\{\mathcal{G}^{t-p+1}, \cdots, \mathcal{G}^t\}$ and aims to learn the embedding $h^{t'}_{G,u} \in \mathbb{R}^{1 \times d}$ and $h^{t'}_{G,v} \in \mathbb{R}^{1 \times d}$ for each node in two different node groups U and V, respectively. Given a graph neural network $G^{t'}$ at each time t', the inputs associated with bond set U consist of the third frontal slice of the original feature tensor $\mathcal{X}_{::t'}$.

Our dataset does not include the input features for the node set V. Three types of embedding initialization are often used for each node: 1) one-hot vector, 2) all one vector, and 3) random initialization for each fund node. The first type works well for a small number of nodes and increases the memory footprint quadratically with the number of nodes. The bond-fund holding data contains $m = 13059$ monthly funds over $T = 173$ months, and using the one-hot coding strategy is not feasible. Instead, we choose the third type and use the random initialization to generate a q-dimensional node embedding for each fund node ($q \ll m$). We pull these node embeddings into a trainable embedding matrix $Z_V \in \mathbb{R}^{m \times q}$ and use it as the input node features for V in graph neural networks. Note that the embedding matrix is shared across all $T = 173$ months and contains the time-invariant factors for each fund. On the other hand, the temporal bipartite networks involve dynamic interactions (investment weights) that take in time-variant factors from bonds, including bond ratings, accrual interests, and time-to-mature each month, and drive the structural changes in the temporal bipartite graphs over T time intervals. Because Z_V is shared across all T iterations of graph learning and prediction, the number of parameters in the initialized fund features is much smaller than the one-hot features and enables fast training.

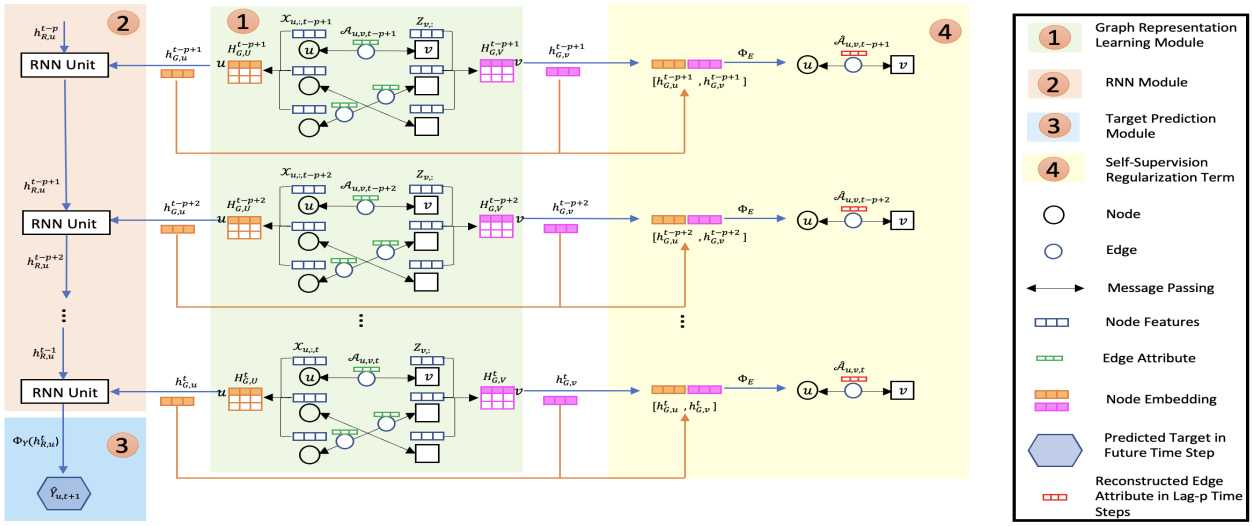


Figure 2: TBGNN Model Architecture

3.2 Bipartite Graph Representation Learning

A general bipartite graph neural network model is essentially a function f_G parameterized by θ_G as follows:

$$h^{t'}_{G,u}, h^{t'}_{G,v} = f_G(X_{:,:,t'}, \mathcal{G}^{t'}, \theta_G) \tag{2}$$

Note that we do not include Z_V as input because it is trainable. The entire graph neural network of f_G is colored in green in Figure 2. We adopt a similar network architecture in [33] and design a message passing function to learn the node embedding on the bipartite graph. Our network architecture has different functionalities from the original one in [33], allows the nodes in any side of bipartite graph to have original features, and serves as embedding learning layers for subsequent temporal inference. The bipartite graph allows message passing alternately from one side to the other side to aggregate information from the nodes from the opposite group through the existing edges. To aggregate information flows from each node v in V to each node u in U in $\mathcal{G}_{t'}$, the l layer GNN $(1 < l \leq L)$ defines a message passing function that takes the concatenation of the node embedding $h^{t'}_{G,u}(l-1) \in \mathbb{R}^{1\times d}$ and edge embedding $e^{t'}_{uv}(l-1) \in \mathbb{R}^{1\times d}$ at layer $(l-1)$ as the input:

$$n^{t'}_u(l) = AGG_l\left(\sigma(\mathbf{P}^{(l)} \cdot [h^{t'}_{G,u}(l-1), e^{t'}_{uv}(l-1)] \mid \forall v \in \mathcal{N}(u, \mathcal{E}^{t'}_{drop}))\right) \tag{3}$$

where $n^{t'}_u(l)$ is the aggregated information for node u learned from $\mathcal{G}^{t'}$ in the l-th layer message passing; AGG_l is the aggregation function; σ is the non-linear activation function; $\mathbf{P}^{(l)}$ is the trainable weight matrix; $[\cdot,\cdot]$ is the concatenation function; $\mathcal{N}$ represents the neighborhood function on a node; $\mathcal{E}^{t'}_{drop}$ is the sampled edge set by applying edge dropout on $\mathcal{G}^{t'}$. Subsequently, we update the node embedding $h^{t'}_{G,u}(l)$ as follows:

$$h^{t'}_{G,u}(l) = \sigma(\mathbf{Q}^{(l)} \cdot [h^{t'}_{G,u}(l-1), n^{t'}_u(l)]) \tag{4}$$

where $\mathbf{Q}^{(l)}$ is the trainable weight matrix. Similarly, the information aggregates onto node v (in this example, institutional investment fund) from all invested nodes $u \in U$. The edge embedding $e^{t'}_{uv}(l-1)$ is then updated as follows:

$$e^{t'}_{uv}(l) = \sigma(\mathbf{O}^{(l)} \cdot [e^{t'}_{uv}(l-1), h^{t'}_{G,u}(l), h^{t'}_{G,v}(l)]) \tag{5}$$

where $\mathbf{O}^{(l)}$ is the trainable weight matrix.

As shown in Figure 2, we initialize node embedding on the first layer GNN $(l = 1)$ with the node features $\mathcal{X}$ and the edge embedding with the original edge weights. The initial input edge weights are the investment amount on bond u by fund v and indicate the fund manager's interest in a bond. For each snapshot $\mathcal{G}^{t'}$ in the lag-p bipartite graphs, the node embedding of U and V and edge embedding in the first layer GNN are processed by the feed-forward MLP (multi-layer perceptron networks) networks as follows:

$$h^{t'}_{G,u}(1) = MLP_{\Pi_U}(\mathcal{X}_{u,:,t'}) \tag{6}$$

$$h^{t'}_{G,v}(1) = MLP_{\Pi_V}(Z_{v,:}) \tag{7}$$

$$e^{t'}_{uv}(1) = MLP_{\Pi_E}(\mathcal{A}_{u,v,t'}) \tag{8}$$

where the MLP_{Π_U}, MLP_{Π_V}, and MLP_{Π_E} are the full-connected networks with the weight matrices $\Pi_U \in \mathbb{R}^{k\times d}$, $\Pi_V \in \mathbb{R}^{q\times d}$, and $\Pi_E \in \mathbb{R}^{1\times d}$, respectively.

Figure 2 shows that all lag-p graphs share the same bipartite graph neural network stack. Here $\mathcal{X}_{:,:,t-p+1:t} \in \mathbb{R}^{n\times k\times p}$ represents the node features of U and $\mathcal{G}^{t-p+1:t}$ are the lag-p graphs. For node u in U or node v in V, the bipartite graph representation learning module (BGNN) generates two series of outputs:

$$\begin{aligned} &BGNN(\mathcal{X}_{:,:,t-p+1:t}, \mathcal{G}^{t-p+1:t}) \\ &= \{H^{t-p+1}_{G,U}, \cdots, H^{t}_{G,U}\}, \{H^{t-p+1}_{G,V}, \cdots, H^{t}_{G,V}\} \\ &= \{h^{t-p+1}_{G,u}, \cdots, h^{t}_{G,u} | u \in U\}, \{h^{t-p+1}_{G,v}, \cdots, h^{t}_{G,v} | v \in V\} \end{aligned} \tag{9}$$

where the $h^{t}_{G,u}$ represents the node embedding of node u yielded by the last layer of the bipartite graph $\mathcal{G}^{t}$. We concatenate all lag-p node embeddings of u and form the time series that will be processed by the temporal module in the following section.

3.3 Modeling of Node-level Temporal Dynamics

Given the sequence of the lag-p node embeddings from the graph representation learning model, the next step is to apply Long short-term memory (LSTM) [14] to estimate the temporal dynamics. LSTM loads $h_{G,u}^{t'} \in \{h_{G,u}^{t-p+1}, h_{G,u}^{t-p+2}, \cdots, h_{G,u}^{t}\}$ at each time t' and the hidden state $h_{R,u}^{t'-1}$ from prior times and generates the hidden state at time t':

$$h_{R,u}^{t'} = LSTM_{W_g^\star, W_r^\star}(h_{G,u}^{t'}, h_{R,u}^{t'-1}) \tag{10}$$

$h_{\mathbf{R},u}^{t'} \in \mathbb{R}^{1\times r}$ denotes the hidden state vector at time t' learned from a sequence of node embeddings; $W_g^\star \in \mathbb{R}^{d\times r}$ and $W_r^\star \in \mathbb{R}^{r\times r}$ (here $\star = \{f, i, c, o\}$) represents the weight matrices of gate functions applied to the node embedding $h_{G,u}^{t'}$ and the hidden state $h_{R,u}^{t'-1}$.

Given the node embedding $\{h_{G,u}^{t-p+1}, h_{G,u}^{t-p+2}, \cdots, h_{G,u}^{t}\}$ of node u, LSTM module generates the sequence of hidden states $\{h_{R,u}^{t-p+1}, \cdots, h_{R,u}^{t}\}$. The LSTM module attempts to enhance the hidden state vector $h_{R,u}^{t'}$ with both the spatial information from the bipartite graph at time t' and the temporal information from the lag-p graph series. The learned temporal enriched hidden vectors serve as the powerful representations for the subsequent multi-task learning for predicting the future target values and reconstructing investment profiles on bonds.

3.4 Target Prediction Module

The hidden state vector $h_{R,u}^{t} \in \mathbb{R}^{1\times r}$ of each node $u \in U$ at time t plays a central role in predicting and regression. During training, the LSTM cell directly takes the output of the BGNN module in Section 3.2 and its hidden state vector $h_{R,u}^{t-1}$ of node u and generate the new hidden state $h_{R,u}^{t}$. Subsequently, we regress the two targets $\hat{Y}_{u,t+1}$ (bond yields and price) onto the new hidden state (shown in blue in Figure 2) as follows:

$$\hat{Y}_{u,t+1} = MLP_{\Phi_Y}(h_{R,u}^{t}) \tag{11}$$

The objective function for the target prediction task is to minimize the mean squared loss between the ground truth targets $Y_{:,t+1}$ and the predicted values in $\hat{Y}_{:,t+1}$. We back-propagate the gradient of the objective function $\mathcal{L}_{TP}$ defined in Equation 1 to the graph representation module, LSTM module and the target prediction module and update all affected parameters.

The LSTM and target prediction modules are a parameterized function $f_{W_r^\star, W_g^\star, \Phi_Y} =: (h_{G,u}^{t-p+1}, h_{G,u}^{t-p+2}, \cdots, h_{G,u}^{t}) \rightarrow \hat{Y}_{u,t+1}$ and implemented as a composite network stack of $MLP_{\Phi_Y} \circ LSTM_{W_g^\star, W_r^\star}(\cdot)$.

3.5 Self-supervision to regularize prediction loss

Predictive models for financial data are prone to supervision starvation that often occurs with the insufficient number of training samples. To overcome this problem, we define a self-supervision module (the fourth module in Figure 2) that uses the graph structure itself instead of target variables to supervise the representation learning on all nodes and regularize the learned node embeddings to reconstruct the entire graph topology and edge weights. Similar to the assumption in [10], we define the self-supervised learning module based on the hypothesis that the node embedding suitable for recovering graph structure is also appropriate for predicting node target variables. The static graph structure prediction consists of edge-level predictions and is defined by the mapping function: $\hat{\mathcal{A}}_{u,v,:} = f_{uv}(h_{G,u}, h_{G,v})$, where $\hat{\mathcal{A}}_{u,v,:}$ denotes the predicted attributes of the edge between vertex u and v and $h_{G,u}$ and $h_{G,v}$ represent the node embedding learned from bipartite graph neural networks. The self-supervision task is inspired by the famous collaborative filter example of movie recommendation based on the user-movie bipartite graph comprising edge attributes for movie ratings. In our bond price prediction case, if the investment amounts from funds are predictive of the bond price, the node embedding that leverages the investment relationship between the bonds and funds may improve the bond target regression and prediction tasks. We extend the idea into dynamic graphs to uncover the temporal enriched node embeddings and model the individual investment fund/bonds interaction patterns and the global trend of the investment market.

Algorithm 1: TBGNN Algorithm

Input : Temporal bipartite graph $\mathcal{G}^{1:T}$
Input attributes $\mathcal{X} \in \mathbb{R}^{n\times k\times T}$ for nodes U
Random initialized node features $Z_V \in \mathbb{R}^{m\times q}$ for nodes V
$\Omega_{U^{1:T}}$ denotes index set of node U has targets
Output : for $t = 1, \cdots, T$
the predicted targets $Y_{:,t+1:t+I} \in \mathbb{R}^{n\times I}$ for up to I time steps in the future.

repeat
  for $t = p+1, \cdots, T$, *given the lag-p historical snapshots* $G^{t-p+1:t}$ **do**
    for $\mathcal{B}_k \subset \Omega_{U^{t+1}}$ `// minibatch of nodes in U with target variables` **do**
      for $\mathcal{G}^{t'} \in G^{t-p+1:t}$ **do**
        Calculate node embedding $H_{G,U}^{t'}$; `// equation. 4`
        Calculate node embedding $H_{G,V}^{t'}$
      Calculate LSTM hidden states $h_{R,u}^{t} = LSTM_{W_g,W_r}(h_{G,u}^{t-p+1:t}), \forall u \in \mathcal{B}_k$; `// equation. 10`
      Calculate predicted targets $\hat{Y}_{u,t+1}, \forall u \in \mathcal{B}_k$; `// equation. 11`
      Calculate target prediction loss $\mathcal{L}_{TP,\mathcal{B}_k} = \sum_{u\in\mathcal{B}_k} \left\|\hat{Y}_{u,t+1} - Y_{u,t+1}\right\|_F^2$; `// eqn. 1`
      Calculate regularization loss $\mathcal{L}_S$; `// eqn. 13`
      Calculate loss $\mathcal{L}_{\mathcal{B}_k} = \mathcal{L}_{TP,\mathcal{B}_k} + \lambda_S \mathcal{L}_S$ `// eqn. 14`
      Back-propagate $\partial\mathcal{L}_{\mathcal{B}_k}$ on $\theta_G, W_g^\star, W_r^\star, \Phi_Y, \Phi_E$; `// Back propagate on target prediction & reconstruction loss`
      Back propagate $\partial\mathcal{L}_{\mathcal{B}_k}$ on Z_V ; `// Continue back-propagate to learn fund embedding`
until *maximum number of epochs or early stopping*;

At time $t' \in [t-p+1, \cdots, t]$, we predict the bond investment network $\mathcal{A}_{:,:,t'}$ as follows:

$$\hat{\mathcal{A}}_{u,v,t'} = MLP_{\Phi_E}([h^{t'}_{G,u}, h^{t'}_{G,v}]) \tag{12}$$

where MLP_{Φ_E} is a multilayer perceptron network with the weight matrix $\Phi_E \in \mathbb{R}^{2d\times 1}$ and $h^{t'}_{G,u}, h^{t'}_{G,v}$ are the output of the graph learning module in Equation 2.

The self-supervision module colored yellow in Figure 2 attempts to recover all lag-p graph that comprises the investment profiles of all funds on all bonds. The main idea of the self-supervised frameworks is to use the graph topology information embedded in the edge attributes to regularize the node representation learned from the GNN module. The objective function of the self-supervision regularization term is to minimizegraph reconstruction error:

$$\mathcal{L}_S = \sum_{t'=t-p+1}^{t} \sum_{e_{uv}\in\mathcal{E}^{t'}} \left\| \mathcal{A}_{u,v,t'} - \hat{\mathcal{A}}_{u,v,t'} \right\|_F^2 \tag{13}$$

Finally, the objective function sums up the errors of the target prediction (Eqn. 1) and network reconstruction (Eqn. 13). To ensure the graph representation learning and recurrent neural networks work properly, we use λ_S to trade-off between these two errors (Algorithm 1 line 12) as follows:

$$\mathcal{L} = \mathcal{L}_{LP} + \lambda_S \mathcal{L}_S. \tag{14}$$

We summarize the entire framework description in Algorithm 1.

4 EXPERIMENTAL DETAILS

4.1 Data

The quarterly bond-fund holding data is collected from eMAXX that provides each fund's end-of-quarter holding position for each bond at a given quarter. The two disjoint node sets represent the bond and fund; each edge captures the investment relationship between a given bond and a given fund. Monthly trading data is constructed from the enhanced TRACE database that contains intra-day trading information of corporate bonds. For each bond, we extract the end-of-month trading prices (percent of $1,000) and yields in percent. We also construct the bond characteristics, including the four well-known factors in bond pricing [8]: monthly ratings, time to maturity, coupon payment, and accrual interests from Mergent Fixed Income Securities Database (FISD). [3] We replicate the quarterly bond-fund holding data to align with the monthly trading data and ensure the months within a quarter share the same bond-fund investment graph. Our sample period begins from August 2002 to September 2016, a total of 173 months.

To clean the data, we remove the bonds with zero holding position, without any position changes, and without any trade. We also require each bond to have at least two trading observations over two quarters and remove bonds with a maturity of less than one year by following the literature [1]. Our final sample contains the data of 15,411 bonds with investments from 13,059 funds over 173 months. There are 89, 766, 167 transactions, each of which corresponds to an edge in the bond-fund bipartite graphs. All edge attribute values are normalized to the range [0, 1]. We choose the price and yield as the target variables with the missing ratio 35.43% and 33.59%, respectively.

[3]We consider ratings from Moody's, S&P, and Fitch and then compute monthly mean, median, the lowest, and most recent ratings within three rating agencies. Also, there are 22 categories of ratings from AAA, AA+, AA, to D. The common practice in finance is to convert each rating into a numerical value. Because the credit rating difference between any two adjacent rating categories is not identical, the conversion to numerical variables makes an incorrect assumption that the credit rating difference between any two adjacent categories is identical. Rather than simply converting to numerical variables, we adopt the best practice in ML, i.e., converting each category as one dummy variable, and gain impressive improvement over the numerical conversion.

Table 1: Hyper-parameters for TBGNN

	train/val/test (months)	node dimension	hidden state dimension	λ_S	batch size
Price	120/20/33	200	200	0.001	512
Yield	120/20/33	200	200	0.01	512

4.2 Baseline methods

In the experiments, we evaluate the performance of our proposed methods, TBGNN and its variant TBGNN-self regularized by self-supervision. We choose five state-of-the-art baseline models designed for high-dimensional time series prediction (three of them are graph neural network-based methods and the other two are multivariate time series methods) and summarize them as follows:

GCN-LSTM [6] is graph convolutional neural network (GCN) and co-trained with a Long Short-term memory (LSTM) on node embeddings.

GraphSage-LSTM [18] is a general inductive framework for graph node representation learning that generates embeddings by sampling and aggregating features from a node's local neighborhood. An LSTM is applied on the node embeddings to capture the temporal dynamics.

GAT-LSTM [30] is a graph node embedding learning approach with an attention layer for (implicitly) assigning different importances to different nodes within a node's neighborhood. This graph neural network is co-trained with LSTM on the node embeddings.

Long Short-term memory (LSTM) [15] and **Gated Recurrent Unit (GRU)** [5] are commonly used recurrent neural networks. We apply the same model for each bond and only use the bonds' features to predict the targets of price and yield. These two methods can not utilize the network structure in the temporal bipartite graph and only offer independent multivariate time series predictions.

4.3 Hyper-parameter settings

Train/validation/test sets are split along the temporal dimension. For all experiments, we use the Adam optimizer with the default configuration and fix the learning rate $lr = 1e-4$ and the batch size = 512. We use MSEs (mean squared errors) as the loss function and set early stopping on the validation loss with $patience = 20$. We tune the hyper-parameters for TBGNN and select the proper dimensions of node embedding and LSTM hidden states. We use the grid search to choose the coefficient λ_S of the self-supervised regularization term within $[1e^{-5}, 1e^{-4}, 1e^{-3}, 0.01, 0.1, 0.5, 1, 5, 10]$. Table 1 shows the hyper-parameters. For the baseline models, we tune the hyper-parameters separately for each dataset and keep them consistent among methods for all experiments.

Table 2: Results for predicting bond price

Model	RMSE	MAE	MAPE	R^2
LSTM	13.0268	8.8412	9.2599	0.2450
GRU	12.1886	7.9721	8.3993	0.3390
GCN-LSTM	12.4308	8.2581	8.8154	0.3125
GraphSage-LSTM	10.8844	6.8728	7.9540	0.4702
GAT-LSTM	13.8322	9.0223	9.9963	0.1488
TBGNN	11.0027	7.1699	8.1717	0.4714
TBGNN-self	**10.5132**	**6.4900**	**7.4952**	**0.5050**
TBGNN-self (partial features)	11.7732	8.2650	9.0998	0.3833

Table 3: Results: Predicting Bond Yields

Model	RMSE	MAE	MAPE	R^2
LSTM	3.0510	2.0375	70.9421	0.2439
GRU	3.1374	2.1512	74.3205	0.2005
GCN-LSTM	2.5569	1.3840	43.0853	0.4690
GraphSage-LSTM	2.1530	1.1747	32.0331	0.6434
GAT-LSTM	2.9240	1.8374	65.8508	0.3056
TBGNN	2.0742	1.0407	29.6957	0.6506
TBGNN-self	**1.9911**	**1.0217**	**28.7886**	**0.6736**
TBGNN-self (partial features)	2.0320	1.1346	32.4293	0.6646

4.4 Time Series Prediction Results

To show the generalization capability of TBGNN in the financial time-series prediction, we conduct two tasks to predict next month's bond prices and yields. There are four out-of-sample measurements to assess the model prediction accuracy: RMSE (square root of mean squared errors), MAE (mean absolute errors), MAPE (mean absolute percent errors), and R^2. The smaller RMSE, MAE, and MAPE values indicate smaller prediction errors and higher prediction accuracy. The larger R^2, the better prediction.

The Bond price prediction results in Table 2 demonstrate that TBGNN-self exceeds far above the five baselines modes of LSTM, GRU, GCN-LSTM, GraphSage-LSTM, GAT-LSTM, and TBGNN. TBGNN-self improves the R^2 by 51.49%, 32.87%, 38.12%, 6.89%, 70.55%, and 6.66%. Our algorithm outperforms two multi-variate time series prediction algorithms, LSTM and GRU, which only use the bonds' features to predict the price and ignore the graph structures in the bond-fund holding networks. The experimental results show that LSTM and GRU do not have sufficient information to capture the temporal patterns of the bond prices with irregular intervals between observations and generate inferior prediction results. This observation confirms that the network structure is indispensable in modeling bond price fluctuations and contributing to the decision-making in the bond market. The graph-based baseline GraphSage-LSTM shows a similar performance to TBGNN but cannot outperform TBGNN-self. Incorporating self-supervision regularization in the optimization processing, TBGNN-self can capture the intrinsic relationship between the bond and fund with insufficient training samples. The results in Table 2 indicate that self-supervision resolves the supervision starvation situation and plays a critical role in further improving our best model TBGNN-self over the simple version of TBGNN by 6.66%.

The Bond yield prediction results are shown in Table 3. The proposed TBGNN-self retains its superiority to all other baselines. TBGNN and TBGNN-self consistently outperform the multi-variate time series prediction baselines, LSTM and GRU, and all graph baselines in predicting the bond yields. The outperformance reaffirms the importance of finance networks in the information propagation and the self-supervised regularization term in the financial data with high sparsity and irregular time intervals.

4.5 Evaluate Input Feature Importance in Prediction

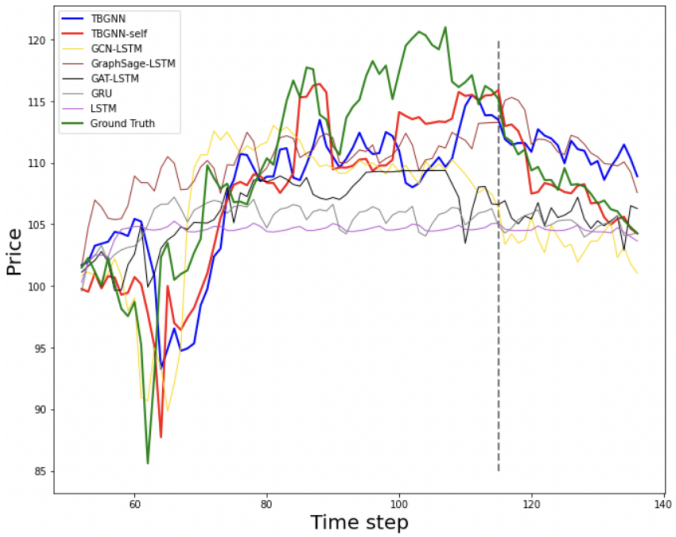
To improve the model interpretability and assess the influence of independent variables on prediction, we design the experiments that include/exclude bond features when predicting bond prices and yields. In the table 2 and 3, the TBGNN-self includes all bond features while TBGNN-self (partial features) excludes semi-annual coupon payment (cp) and accrual interests (ai). In both cases, we observe performance decline once cp and ai are removed from bond features. R^2 decreases by 24.10% for bond price and by 1.34% for bond yields, respectively. The experiments show that coupon payment and accrual interests impact bond price more than yield. The impact on yields is insignificant and tends to be confused with noise. Ceteris paribus, the bond price is an increasing function of coupon payment. If investors get more semi-annual coupon payments for some bonds, these bonds are more attractive to investors, resulting in a high valuation and price. In contrast, bond yields closely reassemble returns (i.e., price percent changes), and a high valuation does not necessarily mean a large increase or decrease in bond yields. These experiments show that our TBGNN effectively uses input features for prediction, attributes the targets to appropriate input features, and is consistent with the well-established understanding of the bond market behavior.

4.6 Visualization

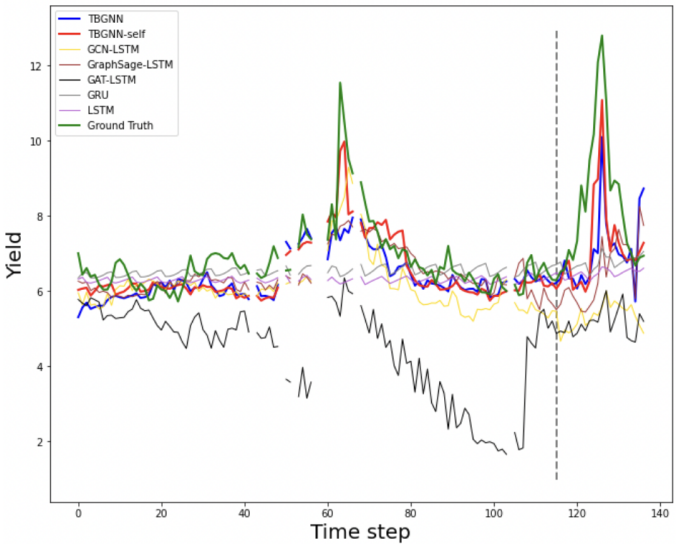
We plot the future value prediction results versus the ground truth on the bond price and bond yield in Figure 3. The green line is the original ground truth. The curve before the dashed line is generated from training data and in-sample tests, while the curve after the dashed line is the prediction result of the testing set. The blue and red curves are the temporal trends extracted from the TBGNN and TBGNN-self prediction results. Compared with all baselines, our model matches closely with the ground truth. In the in-sample and out-of-sample tests data, our model extracts the time patterns and captures the general trends and the prominent peaks in the ground-truth patterns. Our strategy of leveraging network information from the bond-fund bipartite graph and applying recurrent neural networks to extract the temporal patterns demonstrate impressive performance in the prediction task.

5 RELATED WORK

Our work is first related to the recent works in graph neural network models. In finance, Graph neural network based spatiotemporal modeling shows impressive performance in stock return prediction [3, 4, 12, 28] and credit risk prediction [31]. A majority of these works focus primarily on unipartite node set and a static network with predefined topologies [4, 21]. In addition, models for stock pricing usually are based on a complete dataset and cannot be



(a) Bond Price



(b) Bond Yield

Figure 3: Visualization of future value prediction. Note that many gaps appear in in-sample yield predictions due to the lack of the observed yield training targets.

directly applied to bond return prediction with irregular intervals and missing observations.

Our work is also associated with research to model data with irregular intervals [25, 32]. Previous works mainly deal with irregular data by imputing missing values through matrix factorization [27], tensor factorization [23, 26], self-attention [24], and deep recurrent neural network [13]. There are only a few efforts to deal with irregular and missing data in finance. For example, coupled matrix [27] and coupled tensor factorization [26] first impute missing values in financial analysts' earnings forecast data and then predict firms' earnings. However, the prediction performance of coupled matrix or tensor factorization largely depends on sufficient inputs. To ensure reliable prediction, many entities with insufficient information have to be removed. Differently, our bipartite graph neural networks are designed to handle sparse data and accommodate any entity even with a small number of observations, suggesting a broad application to all relevant entities and preventing information loss due to filtering. Moreover, we bypass the missing value imputation and learn node embedding directly from the existing data for bond price and yield prediction.

6 CONCLUSION

This paper presents a bipartite graph neural network-based model for bond price/yield prediction. Even without any observed trade, our proposed TBGNN can still extract information from the bond-fund network and facilitate future price/yield prediction. Our model outperforms multiple advanced machine learning and unipartite graph neural network-based models. In the future price and yield prediction, our model improves R^2 by 6%-51% and 5%-70%, respectively over other state-of-art baselines. This work complements the current literature on asset price prediction and helps financial researchers better understand the bond markets. More importantly, our work offers a novel and general solution to one critical problem in finance, i.e., the scarce data observations with irregular intervals. This paper demonstrates that a carefully designed model will overcome the lack of observations in individual entities and yield impressive performance by leveraging networks embedded in data.

REFERENCES

[1] Jennie Bai, Turan G Bali, and Quan Wen. 2019. Common risk factors in the cross-section of corporate bond returns. *Journal of Financial Economics* 131, 3 (2019).
[2] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. 2020. Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting. *Advances in Neural Information Processing Systems* 33 (2020).
[3] Wei Chen, Manrui Jiang, Wei-Guo Zhang, and Zhensong Chen. 2021. A novel graph convolutional feature based convolutional neural network for stock trend prediction. *Information Sciences* 556 (2021), 67–94.
[4] Dawei Cheng, F.Z. Yang, Sheng Xiang, and Jin Liu. 2022. Financial time series forecasting with multi-modality graph neural network. *Pattern Recognition* 121 (2022), 108218.
[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *CoRR* abs/1606.09375 (2016). arXiv:1606.09375 http://arxiv.org/abs/1606.09375
[7] Zulong Diao, Xin Wang, Dafang Zhang, Yingru Liu, Kun Xie, and Shaoyao He. 2019. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 890–897.
[8] Eugene F Fama and Kenneth R French. 1993. Common risk factors in the returns on stocks and bonds. *Journal of financial economics* 33, 1 (1993), 3–56.
[9] Shen Fang, Qi Zhang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2019. GSTNet: Global Spatial-Temporal Network for Traffic Flow Prediction.. In *IJCAI*. 2286–2293.
[10] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. 2021. SLAPS: Self-Supervision Improves Structure Learning for Graph Neural Networks. *CoRR* abs/2102.05034 (2021). arXiv:2102.05034 https://arxiv.org/abs/2102.05034
[11] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. 2019. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 1–30.
[12] Shibo Feng, Chen Xu, Yu Zuo, Guo Chen, Fan Lin, and Jianbing XiaHou. 2022. Relation-aware dynamic attributed graph attention network for stocks recommendation. *Pattern Recognition* 121 (2022), 108119.
[13] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. 2020. Gp-vae: Deep probabilistic time series imputation. In *International conference on*

artificial intelligence and statistics. PMLR, 1651–1661.

[14] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2003. Learning Precise Timing with Lstm Recurrent Networks. *J. Mach. Learn. Res.* 3, null (mar 2003), 115–143. https://doi.org/10.1162/153244303768966139

[15] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. 2003. Learning Precise Timing with Lstm Recurrent Networks. *J. Mach. Learn. Res.* 3, null (mar 2003), 115–143. https://doi.org/10.1162/153244303768966139

[16] Shihao Gu, Bryan Kelly, and Dacheng Xiu. 2020. Empirical asset pricing via machine learning. *The Review of Financial Studies* 33, 5 (2020), 2223–2273.

[17] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 922–929.

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* (2017).

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

[20] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[21] Wei Li, Ruihan Bao, Keiko Harimoto, Deli Chen, Jingjing Xu, and Qi Su. 2020. Modeling the Stock Relation with Graph Network for Overnight Stock Movement Prediction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, Christian Bessiere (Ed.). 4541–4547. https://doi.org/10.24963/ijcai.2020/626

[22] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*.

[23] Hanpeng Liu, Yaguang Li, Michael Tsang, and Yan Liu. 2019. Costco: A neural tensor completion model for sparse tensors. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[24] Jiawei Ma, Zheng Shou, Alireza Zareian, Hassan Mansour, Anthony Vetro, and Shih-Fu Chang. 2019. CDSA: cross-dimensional self-attention for multivariate, geo-tagged time series imputation. *arXiv preprint arXiv:1905.09904* (2019).

[25] Qingxiong Tan, Mang Ye, Baoyao Yang, Siqi Liu, Andy Jinhua Ma, Terry Cheuk-Fung Yip, Grace Lai-Hung Wong, and PongChi Yuen. 2020. Data-gru: Dual-attention time-aware gated recurrent unit for irregular multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 930–937.

[26] Ajim Uddin, Xinyuan Tao, Chia-Ching Chou, and Dantong Yu. 2020. Nonlinear Tensor Completion Using Domain Knowledge: An Application in Analysts' Earnings Forecast. In *2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 377–384.

[27] Ajim Uddin, Xinyuan Tao, Chia-Ching Chou, and Dantong Yu. 2022. Are missing values important for earnings forecasts? A machine learning perspective. *Quantitative Finance* 22, 6 (2022), 1113–1132.

[28] Ajim Uddin, Xinyuan Tao, and Dantong Yu. 2021. Attention Based Dynamic Graph Learning Framework for Asset Pricing. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1844–1853.

[29] Ajim Uddin and Dantong Yu. 2020. Latent factor model for asset pricing. *Journal of Behavioral and Experimental Finance* 27 (2020), 100353.

[30] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph Attention Networks. https://doi.org/10.48550/ARXIV.1710.10903

[31] Daixin Wang, Zhiqiang Zhang, Jun Zhou, Peng Cui, Jingli Fang, Quanhui Jia, Yanming Fang, and Yuan Qi. 2021. Temporal-aware graph neural network for credit risk prediction. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 702–710.

[32] Qing Ye, Wai Yuen Szeto, and Sze Chun Wong. 2012. Short-term traffic speed forecasting based on data recorded at irregular intervals. *IEEE Transactions on Intelligent Transportation Systems* 13, 4 (2012), 1727–1737.

[33] Jiaxuan You, Xiaobai Ma, Daisy Yi Ding, Mykel J. Kochenderfer, and Jure Leskovec. 2020. Handling Missing Data with Graph Representation Learning. *CoRR* abs/2010.16418 (2020). arXiv:2010.16418 https://arxiv.org/abs/2010.16418

[34] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3634–3640.