Attention-based Graph Neural Networks in Firm CDS Prediction

Jonathan Brogaard* Belinda (Chen) Chen†

February 20, 2025

Abstract

Credit Default Swap (CDS) spreads exhibit network effects driven by firms' default interdependence. This paper employs Graph Neural Networks (GNNs) to predict CDS spreads by modeling firms as nodes and pairwise idiosyncratic volatility spillovers as directed edges. GNNs effectively capture dynamic inter-firm network effects, improving CDS prediction accuracy by over 50% compared to traditional models without edge features. Additionally, we enhance the GNN with node- and edge-attention layers, which highlight key nodes (e.g., manufacturing and intermediary firms) and edges (e.g., connections between intermediary, retail trade, or information firms and other firms) as critical for accurate CDS spread prediction.

Keywords: Graph Neural Networks, CDS Prediction, Contagion Risk, Idiosyncratic Volatility Spillover

JEL Classification: C45, C58, G12, G17

*David Eccles School of Business, University of Utah;brogaardj@eccles.utah.edu
†Gies College of Business, University of Illinois at Urbana-Champaign; chenc16@illinois.edu

1 Introduction

The interdependence of firms' defaults represents a critical area of financial research. Firms are intricately interconnected through a production-based supplier-customer network, where companies purchase intermediate inputs from other firms to produce their own goods, which subsequently serve as inputs for other firms. Consequently, firm-specific default risks can cascade beyond the originating firm, potentially affecting an entire network of interconnected firms.[1] While real defaults remain relatively rare, financial instruments like Credit Default Swaps (CDS)—which depend on default intensities and firm valuations—are widely used in assessing and managing default risk. Notably, researchers have documented that CDS spreads exhibit significant network effects.[2] Therefore, this paper investigates the prediction of firm CDS spreads by leveraging inter-firm network information.

While previous studies have applied machine learning to predict CDS spreads, they have primarily relied on firm-specific characteristics and macroeconomic factors as inputs, due to the limitations of traditional machine learning algorithms in directly incorporating neighboring features. Our paper, however, is the first to integrate network features into a machine learning framework. Specifically, we employ a novel machine learning algorithm, the Graph Neural Network (GNN), which is particularly well-suited for handling complex network structures. This algorithm allows for the incorporation of both firm-level characteristics (node features) and inter-firm linkage characteristics (edge features) as inputs. By embedding information from neighboring nodes, GNNs predict target values while effectively capturing firms' dependencies within the network. The inter-firm linkage characteristics provided to the algorithm are the pairwise idiosyncratic volatility spillover measures,

[1] Acemoglu, Carvalho, Ozdaglar, and Tahbaz-Salehi (2012) demonstrate how an asymmetric supplier-customer network structure can impede risk diversification and amplify idiosyncratic risks, specifically firm-specific productivity shocks, into aggregate risk. Jacobson and Von Schedvin (2015) provides empirical documentation of real default contagion propagating through supplier-customer chains.

[2] See Kitwiwattanachai and Pearson (2015) and Kitwiwattanachai (2015) for comprehensive analyses.

estimated using the method outlined in Chen (2023). These measures enhance the model's ability to account for the interconnectedness of firms, offering a significant improvement over traditional machine learning approaches.

The rationale for using these edge features—specifically, statistical estimates of pairwise idiosyncratic volatility spillovers—is as follows. First, the dynamic structure changes of idiosyncratic volatility spillovers contain additional information that existing firm-level features do not account for. Specifically, Chen (2023) develops a production-based network model to demonstrate the economic and asset pricing implications of idiosyncratic volatility spillovers, supporting these implications with empirical evidence. The author shows that time-varying idiosyncratic volatility spillovers, driven by dynamic supplier-customer chains, predict future aggregate uncertainty and are priced as volatility risk. Thus, these pairwise idiosyncratic volatility spillover measures, which are not typically captured by existing firm-level characteristics, serve as a powerful tool for quantifying comprehensive aggregate risk that affects firms' valuations.

Second, measures of idiosyncratic volatility spillovers are both asymmetric and dynamic. They capture the directional influence between firms (i.e., the impact of firm i on firm j differs from the reverse) as well as the evolving temporal dependencies of network graphs as market fundamentals change.

We utilize monthly CDS spreads for 678 firms from 2005 to 2020. We use 94 accounting variables and 8 macroeconomic variables as node characteristics, following Gu, Kelly, and Xiu (2020), along with measures of idiosyncratic volatility spillovers as edge characteristics, to train GNN models for predicting CDS spreads. Our out-of-sample tests demonstrate that the GNN algorithm significantly outperforms all classic machine learning techniques. Among these, the Convolutional Neural Network (CNN) serves as an ideal benchmark for comparison. While CNN also relies on a neural network structure, it only incorporates node characteristics and does not utilize edge charac-

teristics. Under identical training configurations, GNN reduces the out-of-sample prediction error by more than half compared to CNN, highlighting the pivotal role of network effects in CDS prediction. The prediction accuracy achieved by GNN improves for both investment-grade and high-yield firms, with a more pronounced enhancement observed for investment-grade firms. Intuitively, investment-grade firms, being typically robust and well-established, often participate in numerous input-output contracts, positioning them centrally within financial networks. The inclusion of edge characteristics is particularly effective in predicting the CDS spreads of investment-grade firms, as these firms are more susceptible to contagion risk through complex input-output networks.

Furthermore, this paper reveals the mechanism of GNNs through a novel technique called 'attention'. While GNNs have traditionally operated as a black box due to their nonlinearity and complexity, we introduce node-attention and edge-attention layers into the algorithm, where each node and edge is assigned a trainable attention score. To minimize the training loss, the GNN algorithm selectively focuses on specific nodes and edges. The final attention scores, obtained after training, can be interpreted as importance scores for each node and edge. By outputting these attention scores, we can identify which firms or inter-firm linkages are most significant in CDS prediction.

We examine the temporal node and edge attention scores from a GNN model with a single hidden layer and extrapolate firm-level attention to the sector level for better interpretability. The node attention consistently highlights two sectors as the focus of the GNN algorithm over time: the Finance and Insurance sector and the Manufacturing sector. Intuitively, these sectors occupy central positions in a production-based trade network. Manufacturing and financial firms typically produce general goods and services that serve as common inputs for other sectors. As a result, the health of firms in these sectors has a broader impact on the cross-sectional CDS spreads.

The edge attention analysis reveals temporal shifts in the importance of sectoral connections. In

earlier years, such as 2012, the algorithm predominantly focuses on edges between the Information sector or Retail Trade sector and other sectors, reflecting the rise of new technology and the growing influence of the retail trade industry. In more recent years, such as 2018, it focuses on connections between the Finance and Insurance sector or the Health sector and other sectors, highlighting the growing importance of intermediary sectors in diverse business dealings and an increased focus on health-related issues. During other periods, edge attention becomes more intricate, especially when sectors like Real Estate or complementary sectors such as Utilities and Services experience sector-specific shocks. In such cases, the edges representing volatility spillovers to specific sectors become focal points for the algorithm. This reveals the complex and dynamic nature of financial networks. By identifying and adapting to critical edges at different times, the GNN algorithm enhances the forecasting accuracy of firm CDS spreads.

Overall, this paper enhances CDS pricing models by integrating insightful network features and leveraging advanced machine learning methodologies.

Related Literature

This paper contributes to the literature on CDS and default evaluations. Many studies propose structural models to estimate default probabilities, including the Distance-to-Default (DID) model by Merton (1974) and the 'naive' model by Bharath and Shumway (2008). Jarrow and Yu (2001) models counterparty default risk using a two-sector framework. Building on their work, this paper proposes a generalized structural framework with multiple representative firms to model the interdependence of firms' defaults and their CDS spreads. The network effect of CDS spreads is consistent with the findings of Kitwiwattanachai and Pearson (2015), who demonstrate that CDS spreads can be used to infer asset correlations.

Empirical studies have documented real default contagion. For example, Jacobson and Von Sched-

vin (2015) show that corporate failures can result from counterparty failures through the trade credit channel. Building on this empirical evidence, this paper is the first to investigate the prediction of default-related instruments in a more generalized setting by directly incorporating granular pairwise network linkages into the prediction task.[3]

Our work also contributes to the growing literature on inter-firm networks. Acemoglu et al. (2012) demonstrates that an asymmetric supplier-customer network hinders risk diversification and can translate idiosyncratic risks into significant aggregate fluctuations. Herskovic (2018) reveals that dynamic structural changes in the input-output network have important implications for GDP and asset prices. Furthermore, Chen (2023) provides empirical evidence that the dynamics of idiosyncratic risk spillovers through the network can predict aggregate volatility and influence the cross-section of assets. This paper adopts the same measures of idiosyncratic volatility spillovers as Chen (2023) and, for the first time, applies them as linkage characteristics in a machine learning model to predict CDS spreads. Although these measures are constructed using stock data, we rely on the mechanism outlined by Chen (2023) to explain the price discovery of CDS spreads. Specifically, idiosyncratic volatility spillovers reflect the connections between firms' fundamentals, and the dynamics of these pairwise spillovers capture additional systematic risk.[4]

Lastly, this paper contributes to the growing literature on machine learning in asset pricing. For example, Gu, Kelly, and Xiu (2020) and Gu, Kelly, and Xiu (2021) apply machine learning techniques to predict stock returns, demonstrating their effectiveness in handling high-dimensional

[3]Additional contributions to the measurement of default and default-related instruments include Altman (1968), Almeida and Philippon (2007), Campbell, Hilscher, and Szilagyi (2008), Gouriéroux, Monfort, and Renne (2014), Kitwiwattanachai (2015), and Bao, Hou, and Zhang (2023). Further literature includes Beaver (1966), Zmijewski (1984), Yang, Platt, and Platt (1999), and Galil, Shapir, Amiram, and Ben-Zion (2014).

[4]Additional relevant literature includes Carvalho and Gabaix (2013), Gabaix (2011), Carvalho (2008), Diebold and Yılmaz (2014), Acemoglu, Akcigit, and Kerr (2016), Blasques, Koopman, Lucas, and Schaumburg (2016), Härdle, Wang, and Yu (2016), Acemoglu, Ozdaglar, and Tahbaz-Salehi (2017), Demirer, Diebold, Liu, and Yilmaz (2018), Chen, Härdle, and Okhrin (2019), Liu (2022), Dew-Becker (2023), Engle and Kelly (2012), Herskovic, Kelly, Lustig, and Van Nieuwerburgh (2016), and Herskovic, Kelly, Lustig, and Van Nieuwerburgh (2020).

data. Kelly, Pruitt, and Su (2019) develops a model called Instrumented Principal Component Analysis (IPCA), which incorporates latent factors and time-varying loadings to predict the cross-section of CDS spreads.[5] This paper introduces a new machine learning algorithm, Graph Neural Networks, to overcome the limitations of traditional machine learning algorithms, which cannot directly incorporate linkage characteristics. We are the first to integrate high-frequency, dynamic, and asymmetric network features for asset prediction—features that are largely missing from publicly available datasets. We construct network features based on micro-founded financial intuition and show that they have important implications for asset management.

The paper is structured as follows: Section 2 outlines a conceptual framework for CDS pricing. Section 3 describes the methodology, detailing the architecture of the GNN (and GNN-attention) and the implementation specifics. Section 4 presents the empirical results, and Section 5 concludes the paper.

2 Conceptual Framework

In this section, we develop a structural framework to demonstrate that CDS spreads are influenced by both firm-level characteristics (i.e., node characteristics) and inter-firm network effects (i.e., edge characteristics), which motivates our use of the GNN algorithm.

Our study of Credit Default Swap (CDS) pricing builds upon the works of Das, Hanouna, and Sarin (2009).[6]We assume that the default intensity λ of firms follows a stochastic process. This

[5]Additional relevant literature includes Kelly and Jiang (2014), Kelly, Malamud, and Zhou (2024), Wang, Lin, Cui, Jia, Wang, Fang, Yu, Zhou, Yang, and Qi (2019), Uddin, Tao, and Yu (2021), and Zhang, Pu, Cucuringu, and Dong (2023).

[6]Related literature also includes Duan, Sun, and Wang (2012), Duffie (1999) and Jankowitsch, Pullirsch, and Veža (2008).

allows us to express the survival probability from time 0 to time τ as follows:

$$s_t = \exp(-\int_0^\tau \lambda_t dt), \tag{1}$$

where s_t is the survival probability, capturing the likelihood of a firm surviving without defaulting over a given period, and λ_t is the default intensity at time t. Following Das, Hanouna, and Sarin (2009), we express the default intensity in Equation (1) as a linear function of both macroeconomic variables and firm-specific characteristics. This relationship is expressed as follows:

$$\lambda_i = \beta_0' M + \beta_1' X_i, \tag{2}$$

where λ_i represents the default intensity of firm i, M denotes macro variables, X_i represents firm-level characteristics, and β_0 and β_1 are vectors of coefficients capturing the impact of these variables on the default intensity. Equation (2) provides a basic understanding of the factors influencing default intensity. We then expand upon this framework to incorporate network effects.

The existing literature shows that the default intensity of firm i is influenced not only by its intrinsic characteristics, denoted as X_i, but also by contagion effects from other firms. For instance, Jarrow and Yu (2001) incorporates the concept of counterparty risk and introduces inter-firm edges into the reduced-form model of firm default intensity. By generalizing their framework[7], we express the default intensity as follows:

$$\lambda_i = \beta_0' M + \beta_1' X_i + \sum_{j=1}^{n} \beta_2' a_{ij}, \tag{3}$$

[7]In Jarrow and Yu (2001), the authors examine debt-related edges between firms. They propose a scenario where each of two firms holds the other's debt. In such a case, the default of firm i would precipitate an increase in the default probability of firm j, and vice versa. To capture this contagion risk, they introduce a jump term: $\lambda_t^i = a_i + a_j \mathbf{1}_{\mathbf{t}\geq\tau^\mathbf{j}}$. Their analysis focuses on a two-firm framework due to the computational complexities involved in extending the model to n firms. Building on their work, our study seeks to provide a generalized model of default intensity among n firms.

where a_{ij} represents the contagion effect from firm j to firm i. Equation (3) forms the core of our network-based default intensity model, which we will use to derive our CDS pricing formula.

Under the risk-neutral probability measure, the CDS spread can be derived by equating the expected present value of premium payments with the expected present value of default loss payments. The expression for the CDS spread is given by

$$\text{CDS Spread}_i = \frac{\mathbb{E}\left[\int_0^T \exp\left(-\int_0^\tau r_t dt\right) s_\tau \lambda_\tau \left(1-\phi_\tau\right) d\tau\right]}{\mathbb{E}\left[\int_0^T \exp\left(-\int_0^\tau r_t dt\right) s_\tau d\tau\right]}, \tag{4}$$

where r_t is the risk-free rate, s_τ is the survival probability up to time τ, λ_τ is the default intensity at time τ, ϕ_τ is the recovery rate at time τ, and T is the maturity of the CDS contract. Equation (4) provides the theoretical foundation for CDS pricing in a continuous-time setting. We then examine the discrete version of the CDS spread expression, wherein each small time interval is denoted as δ. In the particular scenario where the default probability remains constant conditional on each state variable and the recovery rate ϕ is also constant, the CDS spread can be expressed as a function of the default intensity λ:

$$\text{CDS Spread}_i = \frac{(1-\phi)(1-\exp(-\lambda_i \delta))}{\delta}, \tag{5}$$

where ϕ is the constant recovery rate and δ is the small time interval. Incorporating the expression for default intensity in Equation (3) into the CDS spread formula in Equation (5), the CDS spread can be expressed as a nonlinear function of both firm-specific characteristics, macro characteristics and the network effects from other interconnected firms:

$$\begin{aligned}\text{CDS Spread}_i &= \frac{(1-\phi)}{\delta}(1-\exp(-\delta(\beta_0' M + \beta_1' X_i + \sum_{j=1}^{n} \beta_2' a_{ij}))) \\ &\equiv f(M, X_i, a_{ij}).\end{aligned} \tag{6}$$

This refined formulation in Equation (6) provides a more nuanced understanding of the factors influencing a firm's CDS spread in a networked financial system. It is important to emphasize here that this structural framework gives us two critical insights. Firstly, it highlights the importance of considering the nonlinearity of characteristics when predicting CDS spreads, which justifies the preference for machine learning techniques over linear regression models. Secondly, it underscores the necessity of integrating edge information, i.e. the network effect into the predictive algorithm. This is the rationale behind the introduction of Graph Neural Networks (GNN), an algorithm adept at handling complex network graph structures. In the subsequent section, we will elaborate on the GNN framework.

3 Methodology

In this section, we elaborate on the algorithms of Graph Neural Networks (GNN) and GNN-attention, along with their respective implementation details.

3.1 Graph Neural Networks

The Graph Neural Network (GNN) algorithm encompasses two primary frameworks: the updating scheme, which relates to intra-layer design, and the message passing scheme, which is inter-layer design. Figure 1 illustrates these schemes, with panel A showing the intra-layer scheme and panel B depicting the inter-layer scheme.

FIGURE 1 ABOUT HERE

The updating scheme is articulated as follows:

$$h_i^{(k+1)} = AGG\left\{ACT\left(DP\left(BN\left(\omega^{(k)}\hat{h}_i^{(k)} + b^{(k)}\right)\right)\right), j \in \mathcal{N}(i)\right\}, \tag{7}$$

where $h_i^{(k+1)}$ is the node embedding for layer $k+1$, $AGG(\cdot)$ is the aggregation function, $ACT(\cdot)$ is the nonlinear activation function, $DP(\cdot)$ is the dropout function, $BN(\cdot)$ is the batch normalization function, $\omega^{(k)}$ is the trainable weight matrix, $b^{(k)}$ is the bias term, $\hat{h}_i^{(k)}$ is the normalized embedding of node i at layer k, and $\mathcal{N}(i)$ denotes the local neighborhood of node i.

Equation (7) describes the rule for updating node embeddings within a GNN comprising a single hidden layer. The embeddings from neighboring nodes are aggregated, and the resultant vector undergoes a series of transformations: batch normalization, dropout, and nonlinear activation, followed by the application of a trainable weight matrix and a bias term. We use Batch Normalization (see (Ioffe and Szegedy, 2015)) to help stabilize and accelerate the training of deep neural networks. We employ dropout to prevent overfitting and use the rectified linear unit (ReLU) as an activation function to introduce nonlinearity. Finally, we aggregate node characteristics by using edge-weighted average node characteristics to reflect the relative importance of neighboring information.

To capture the dynamic network effect, represented by Panel A in Figure 1, we incorporate an additional technique. For each hidden layer, we adopt Long Short-Term Memory (LSTM) to capture temporal graph dependencies. It's important to note that we do not apply LSTM to each input variable; instead, we use it to aggregate the temporal information from the GCN neuron outputs. As the basic GCN framework takes the graph as fixed, adding LSTM between neurons introduces temporal dependencies. To avoid model overcomplexity and misspecification, we fine-tune all parameters for both GCN and LSTM, which is elaborated in Section 3.3.2.

The message passing scheme is described as follows. Each $\hat{h}_i^{(k)}$ in a minibatch covers both the

node i and its associated information, as delineated by the subsequent equation:

$$\hat{h}_i^{(k)} = h_i^{(k)} \bigoplus_{j \in \mathcal{N}(i)} (a_{i,j}), \tag{8}$$

where $\hat{h}_i^{(k)}$ is the normalized embedding of node i at layer k, $h_i^{(k)}$ is the embedding of node i at layer k, $\bigoplus$ represents the concatenation operation and $\mathcal{N}(i)$ is the neighborhood of node i.

In equation (8) , $a_{i,j}$ denotes the edge characteristic (i.e., the directed contagion effect) from node j to node i. The matrix collecting all elements $a_{i,j}$ is defined as A, i.e., $A \equiv [a_{ij}]$. A will also be referred to as the adjacency matrix.

To encode the network effect, we follow the conventional graph convolutional network (GCN) algorithm proposed by Kipf and Welling (2016). Under GCN, the network information is encoded as follows:

$$\hat{H}^{(k)} = \begin{cases} D^{-\frac{1}{2}} \hat{A} D^{-\frac{1}{2}} H^{(k)} W^{(k)} & \text{if } A \text{ is symmetric} \\ D^{-1} \hat{A} H^{(k)} W^{(k)} & \text{if } A \text{ is asymmetric} \end{cases} \tag{9}$$

where $\hat{H}^{(k)}$ is the output of the k-th layer, $\hat{A} = A + I$ with I being the identity matrix, D is the out-degree matrix, $H^{(k)}$ is the input to the k-th layer, and $W^{(k)}$ is the trainable weight matrix for the k-th layer. In equation (9), the out-degree matrix D is a diagonal matrix where the diagonal elements are equal to the column sums of A.

Finally, to address the challenge of an unbalanced panel where each CDS only exists for a certain period throughout the time, we apply a masking scheme to the GNN algorithm. We mask the nodes (i.e., firms) and the corresponding edges if they do not exist.

3.2 GNN-Attention

To elucidate the mechanism within the neuron training, we enhance the basic GNN algorithm with a technique called 'attention'[8].We refer to this enhanced algorithm as GNN-attention. While a basic GNN is often a black box with nonlinearity and complexity, incorporating attention layers allows us to assign different weights to nodes and edges to minimize the training error. These weights, when output, can effectively reveal which nodes or edges are given more 'attention'.

We incorporate node attention into the intra-layer design. We initiate a trainable $n \times 1$ node attention vector, which is updated from layer to layer. This vector is then aggregated with the node features matrix to dynamically weight neighboring nodes based on their importance scores, enabling the GNN to concentrate on more relevant information during the intra-layer update. The algorithm is as follows:

$$\begin{cases} \text{node attention}^{(k+1)} = DP\left(ACT\left(BN\left(\text{node attention}^{(k)}\right)\right)\right) \\ h_i^{(k+1)} = AGG_2\left\{AGG_1\left\{DP\left(ACT\left(BN\left(\omega^{(k)}\hat{h}_i^{(k)} + b^{(k)}\right)\right)\right), \text{Node attention}^{(k)}\right\}, j \in \mathcal{N}(i)\right\} \end{cases} \tag{10}$$

where $\text{node attention}^{(k)}$ is the node attention vector at layer k, $h_i^{(k+1)}$ is the node embedding for node i at layer $k+1$, $DP(\cdot)$ is the dropout function, $ACT(\cdot)$ is the activation function, $BN(\cdot)$ is the batch normalization function, $\omega^{(k)}$ is the trainable weight matrix for layer k, $\hat{h}_i^{(k)}$ is the normalized embedding of node i at layer k, $b^{(k)}$ is the bias term for layer k, $AGG1(\cdot)$ and $AGG_2(\cdot)$ are aggregation functions, and $\mathcal{N}(i)$ is the neighborhood of node i.

The aggregation function $AGG_1(\cdot)$ in equation (10) aggregates the transformed features of

[8]The related literature includes Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, and Polosukhin (2017), Veličković, Cucurull, Casanova, Romero, Lio, and Bengio (2017) and Thekumparampil, Wang, Oh, and Li (2018)

neighboring nodes with these attentions, using an element-wise product. This weighted aggregation emphasizes the contributions of neighboring nodes based on their respective attentions, thereby prioritizing nodes deemed more important. The second aggregation function, $AGG_2(\cdot)$, is the same as the $AGG(\cdot)$ in Equation (7), and it combines information from both the node itself and its neighboring effects.

For edge attention, we incorporate it through the inter-layer scheme. Specifically, we initialize a trainable $n \times 1$ edge attention vector and integrate it with the existing node representation through element-wise multiplication:

$$\hat{H}^{(k)} = \begin{cases} \left(\left(D^{-\frac{1}{2}}\hat{A}D^{-\frac{1}{2}}\right) \odot \text{edge attention}^{(k)}\right) H^{(k)}W^{(k)} & \text{if } A \text{ is symmetric} \\ \left(\left(D^{-1}\hat{A}\right) \odot \text{edge attention}^{(k)}\right) H^{(k)}W^{(k)} & \text{if } A \text{ is asymmetric} \end{cases} \tag{11}$$

where $\hat{H}^{(k)}$ is the output of the k-th layer, D is the degree matrix, $\hat{A}$ is the modified adjacency matrix, edge attention$^{(k)}$ is the edge attention vector at layer k, $H^{(k)}$ is the input to the k-th layer, $W^{(k)}$ is the trainable weight matrix for the k-th layer, and $\odot$ denotes element-wise multiplication. The edge attention is updated following a standard intra-layer scheme:

$$\text{edge attention}^{(k+1)} = DP\left(ACT\left(BN\left(\text{edge attention}^{(k)}\right)\right)\right), \tag{12}$$

where edge attention$^{(k+1)}$ is the edge attention vector at layer $k+1$, and $DP(\cdot)$, $ACT(\cdot)$, and $BN(\cdot)$ are as defined previously. Under this specification in Equations (12) and (11), each node representation incorporates information from its neighboring nodes with edge attention assigned.

The integration of node and edge attention forces the neurons to put more weight on important nodes and edges to minimize the error. By outputting the weights from node and edge attention, we can open the black box of GNN to reveal which firm and which inter-firm edge is more important

in the CDS prediction.

3.3 Implementation

3.3.1 Estimate of Inter-firm edge Characteristics

Here we elaborate on the edge characteristics, which are derived from a systemic risk measurement. Specifically, we estimate the idiosyncratic volatility spillover among sectors and extrapolate to the firm level. The measure of idiosyncratic volatility spillover is used as an input for the inter-firm edge characteristics in GNN. We use 2-digit NAICS codes for industry classification.

We estimate sector-level idiosyncratic volatility spillover following the procedures outlined by Diebold and Yılmaz (2014), Demirer et al. (2018) and Chen (2023). The process involves several steps: We calculate daily industry stock returns as value-weighted average of firm returns using all firms in CRSP dataset. Next, we estimate the daily industry idiosyncratic returns by regressing these industry returns against the Fama-French 3 factors for each calendar month and take the residual return as the idiosyncratic return.[9]Following this, we determine the idiosyncratic volatility by calculating the monthly standard deviation of these daily idiosyncratic returns.

Subsequently, we estimate pairwise idiosyncratic volatility spillovers. For each rolling window of 90 months, we estimate a LASSO Vector Autoregression (VAR) model of the volatility panel and perform a Generalized Variance Decomposition (GVD) for the 6-month-ahead forecasting error.[10]

[9]The details of the industry classification based on NAICS 2-digit codes are presented in Internet Appendix A. For robustness checks, we also present results using alternative industry classifications, including Fama-French 48 sectors and Fama-French 12 sectors. Additionally, we provide results using idiosyncratic returns after removing only the CAPM factor, or PC 5 factors, or without removing any common factors. All robustness results are available in Internet Appendix E.

[10]The estimation results are robust to rolling window lengths ranging from 80 to 100 months. Furthermore, the findings remain consistent across forecasting horizons ranging from 6 to 10 months.

Specifically, for n sectors, the VAR model is specified as follows:

$$\begin{bmatrix} logvol_{1,t} \\ logvol_{2,t} \\ logvol_{3,t} \\ \dots \\ logvol_{n,t} \end{bmatrix} = \begin{bmatrix} \phi^0_{1,t} \\ \phi^0_{2,t} \\ \phi^0_{3,t} \\ \dots \\ \phi^0_{n,t} \end{bmatrix} + \begin{bmatrix} \phi_{1,1,t} & \dots & \phi_{1,n,t} \\ \phi_{2,1,t} & \dots & \phi_{2,n,t} \\ \phi_{3,1,t} & \dots & \phi_{3,n,t} \\ & \dots & \\ \phi_{n,1,t} & \dots & \phi_{n,n,t} \end{bmatrix} \begin{bmatrix} logvol_{1,t-1} \\ logvol_{2,t-1} \\ logvol_{3,t-1} \\ \dots \\ logvol_{n,t-1} \end{bmatrix} + \begin{bmatrix} \varepsilon^{VAR}_{1,t} \\ \varepsilon^{VAR}_{2,t} \\ \varepsilon^{VAR}_{3,t} \\ \dots \\ \varepsilon^{VAR}_{n,t} \end{bmatrix}, \tag{13}$$

where $logvol_{i,t}$ is the log idiosyncratic volatility of sector i at time t, $\phi_{i,j,t}$ are the VAR coefficients, and $\varepsilon^{VAR}_{i,t}$ are the error terms.

This VAR model is estimated in a high-dimensional setting. Following Demirer et al. (2018), we employ LASSO to estimate the parameters. LASSO imposes sparsity on the coefficients automatically and serves as a computationally efficient method for both shrinkage and variable selection. The regularization parameter is selected for each equation using 10-fold cross-validation.

We then perform a Generalized Variance Decomposition (GVD) on the 6-month-ahead forecasting error ε^{VAR}_t of each VAR model in Equation (13). The element a_{ij} is derived as

$$a_{ij} = \frac{\sigma_{jj}^{-1} \sum_{l=0}^{L-1} \left(e_i' \Theta_l \Sigma e_j\right)^2}{\sum_{l=0}^{L-1} \left(e_i' \Theta_l \Sigma \Theta_l' e_i\right)}, \tag{14}$$

where $L = 6$, e_i is the unit vector with the ith element as 1 and all others zero, Σ is the variance-covariance matrix of the residual $\varepsilon^{VAR}_{i,t}$, σ_{jj} is the jth diagonal element of Σ, and Θ_l represents the coefficient matrix Φ multiplied by the l-lagged shock vector in the infinite MA representation.[11]

[11] It's worth noting that GVD is particularly advantageous compared to the standard variance decomposition, which is typically based on Cholesky-factor orthogonalization. The primary reason for this preference is that GVD is invariant to ordering while a Cholesky-factor orthogonalization is sensitive to variable ordering.

We collect the a_{ij} in Equation (14) through each rolling window into time-varying adjacency matrices $A_t = [a_{ij}]_t$.[12] For each matrix, we normalize each row to sum to 1, resulting in the following structure:

Adjacency matrix A_t

	$Sector_1$	$Sector_2$	...	$Sector_k$
$Sector_1$	$a_{1,1,t}$	$a_{1,2,t}$	...	$a_{1,k,t}$
$Sector_2$	$a_{2,1,t}$	$a_{2,2,t}$	...	$a_{2,k,t}$
...		...		
$Sector_k$	$a_{k,1,t}$	$a_{k,2,t}$	...	$a_{k,k,t}$

Finally, we extrapolate the sector-level adjacency matrix to the firm level by directly imposing sectoral risk spillover onto firms' risk spillover, based on each firm's sector classification. Figure 2 illustrates this extrapolation process.

FIGURE 2 ABOUT HERE

Figure 2 presents an example of extrapolating sector-level idiosyncratic risk spillover to the firm level. Panel A shows an example of the risk spillover among three sectors, while Panel B demonstrates the extrapolation to five firms belonging to these sectors. For instance, if the risk spillover intensity from sector 2 to sector 1 is 0.1, this value is imposed as the spillover intensity

[12] We construct the adjacency matrix using data from t-90 to t to represent edge characteristics at time t. The results remain robust when we lag the edge inputs by 1 month. Prior literature (e.g., Kryzanowski, Perrakis, and Zhong (2017)) indicates no clear lead-lag relationship between the CDS and equity markets in response to positive news, but suggests the CDS market leads the equity market in response to negative news by a matter of days to a few weeks. Therefore, we conduct a robustness test by lagging the edge characteristics (from the equity market) by 1 month to ensure we do not use future information for predictions.

from any firm in sector 2 to any firm in sector 1.

By imposing the sector-level adjacency matrix onto the firm level, we effectively map inter-sectoral dependencies to inter-firm dependencies. This approach assumes that each firm in the sample is representative of its sector. Intuitively, we can model the market with a production-based multi-sector model with n sectors, each producing its own good used as inputs for other sectors. Each sector features a representative firm, and sector-level idiosyncratic risk spillover captures dynamics for the broad market.

The use of stock data for price discovery in the CDS market can be justified through several considerations. First, stock return can be used to infer firm value and CDS spread. Based on the Merton's model, a firm's equity can be viewed as a call option on its assets, with the strike price equal to the face value of its debt. Consequently, stock data provides valuable information about changes in firm value, which in turn directly influence the firm's default risk and its CDS spread. Additionally, Kitwiwattanachai and Pearson (2015) show that the stock correlation can be inferred from CDS spread and the inferred correlation can capture default dependence. While bond data could theoretically be used to infer CDS spreads as well, it has practical limitations for our analysis. Bond market data is of low-frequency and cannot generate a comparable risk spillover measure.

Second, the volatility spillover among stocks captures the network of firms' fundamentals. The input-output relationship among firms leads to the connection among firms' cash flows, dividends, and stock returns. As stock returns are time-varying, this also leads to stock volatility spillover. Thus, the measure of stock risk spillover not only reflects the network among expected stock returns, but also captures the network among firms' fundamentals. This measure should have predictive power for CDS spreads. Additionally, Chen (2023) show that the dynamics of idiosyncratic volatility spillover contain systematic risk and have long-run impacts on uncertainty. This risk spillover measure provides a cleaned and high-frequency measure of volatility risk. Since this

measure captures uncertainty, it should affect the cross-section of firm values, default intensity, and CDS spreads. Therefore, the predictive power of this measure comes through the uncertainty channel.

Throughout this subsection, we characterize the dynamic network structure by utilizing the information of idiosyncratic risk spillover. This measure is asymmetric, capturing the asymmetric interdependence among firms, and dynamic, capturing the time-varying network structure.

3.3.2 Implementation Details

We use daily Markit CDS data for all U.S. firms spanning from January 2005 to December 2020.[13] We focus on the 5-year tenor and senior unsecured contract, recognized as the most liquid. Furthermore, we use the CDS spread under the most prevalent XR14 contract. The spread for 'XR' contracts reflects default risk, excluding restructuring risk, which aligns with the structural model.[14] We construct monthly panel data by retaining the most recent spread of each month. Our dataset comprises 678 firms with 130,176 observations in total over the period from January 2005 to December 2020.

TABLE 1 AND FIGURE 3 ABOUT HERE

Table 1 presents summary statistics for the distribution of log CDS spreads, along with corresponding firms' market capitalization and implied ratings distribution. Columns 1-2 show that CDS spreads exist for a median duration of 137 months (11.4 years), with a minimum of 2 months and a maximum of 192 months (16 years). Columns 3-4 demonstrate that our sample spans from very small firms (with capitalization of $700) to very large firms (with capitalization of $1.73 billion).

[13] Prior to 2005, the availability of CDS data was limited, rendering it suboptimal for machine learning applications.

[14] As noted by Liu (2022), 'XR' contracts became the standard for U.S. corporates following the 2009 CDS Big Bang, while 'MR' (modified restructuring) contracts were more commonly used prior to this event. The spread for 'XR' contracts reflects default risk, while that for 'MR' reflects both default and restructuring risks.

Columns 5-6 indicate that the firms range from very high credit quality (AA rating) to very low credit quality (CCC and D ratings).[15]These statistics underscore the broad market representation of our sample with respect to duration, size, and credit quality.

Figure 3 displays the histogram of log CDS spreads for all contracts in the sample. We use the logarithm of CDS spreads to normalize the right-skewed distribution of raw spreads, which serves as our target variable. The figure shows that the mean log CDS spread is approximately -4.69 with a standard deviation of 0.94.

For temporal node characteristics, we construct 94 firm-specific features spanning our sample period, following the procedures outlined by Gu, Kelly, and Xiu (2020). Of these 94 characteristics, 61 are updated annually, 13 quarterly, and 20 monthly. These features contain independent and nonredundant information.[16]Additionally, we incorporate 8 macroeconomic variables sourced from Welch and Goyal (2008): dividend-price ratio (dp), earnings-price ratio (ep), book-to-market ratio (bm), net equity expansion (ntis), Treasury-bill rate (tbl), term spread (tms), default spread (dfy), and stock variance (svar).[17]In total, we have 112 node-level characteristics. For normalization, we follow Kelly, Pruitt, and Su (2019) and Freyberger, Neuhierl, and Weber (2020), ranking all characteristics for each month and mapping them into the interval $[-1, 1]$, while maintaining all other variables at their median value of zero.

Our model training utilizes 72-month data (200501–201012), with a 12-month validation set (201101–201112) for hyperparameter tuning and overfitting prevention. The testing data comprises

[15]The total firm count exceeds 678 because some firms experienced rating changes, resulting in individual firms being associated with multiple ratings throughout the sample period. The distribution of CDSs across sectors are presented in the Internet Appendix A.

[16]We account for data release delays by following Gu, Kelly, and Xiu (2020) and Gu, Kelly, and Xiu (2021). The code and data for early years are made available by the authors on their website. Related literature includes Fama and French (2016), and Green, Hand, and Zhang (2017), Hou, Xue, and Zhang (2020), Gu, Kelly, and Xiu (2021) and Kelly, Malamud, and Zhou (2024). The details of the 94 characteristics are presented in the Internet Appendix B.

[17]Incorporating interactions between the 94 firm-level characteristics and 8 macro characteristics delivers similar out-of-sample results.

samples from the following month, providing out-of-sample assessments of model performance. We recursively refit the models each month to incorporate the latest input information, despite the computational expense. With each refit, we increase the training sample by 1 month, maintaining the same size of the validation sample but rolling it forward to include the most recent month. This approach yields 108 months (201201-202012) for out-of-sample testing.

For both GNN and GNN-attention models, we use the Stochastic Gradient Descent (SGD) optimizer. To mitigate overfitting, we implement early stopping and fine-tune all hyperparameters based on the mean squared error observed on the validation set. We use simple unidirectional algorithm for LSTM and limit the number of hidden layers to either 1 or 2 for both GCN and LSTM to avoid model misspecification. The optimal configuration consists of 1 hidden layer for GCN with 12 neurons, and 2 hidden layers for LSTM with 12 and 6 neurons, respectively. For the GNN-attention model, we implement a dual-head attention mechanism and use the mean of their outputs as the final attention scores. Hyperparameter details are provided in the Internet Appendix D.

In addition to GNN and GNN-attention, we employ five classic machine learning algorithms as competing benchmarks: Principal Component Regression (PCR), Partial Least Squares (PLS), Gradient Boosting Regression Trees (GBRT), Random Forest (RF), and Convolutional Neural Networks (CNN). These algorithms utilize the same node characteristics as inputs but do not incorporate edge characteristics. They are trained using the same configurations as GNN and GNN-attention. Detailed descriptions of these algorithms are provided in the Internet Appendix C.

4 Empirical Results

4.1 Out-of-sample Results

Our results are evaluated based on out-of-sample (OOS) prediction accuracy. The out-of-sample data is used to provide an objective assessment of the performance of different algorithms. The primary metric for evaluating OOS predictions is the root mean square error (RMSE). Table 2 presents the pooled out-of-sample RMSE for various machine learning algorithms, and Figure 4 visualizes these RMSEs using bar plots.

TABLE 2 AND FIGURE 4 ABOUT HERE

In Table 2, Column 1 shows RMSE for the entire sample from February 2005 to December 2020. Columns 2 and 3 report RMSE for investment-grade (BBB and above) and high-yield (BB and below) firms, respectively. Columns 4 and 5 display RMSE for small (below median market capitalization) and large (at or above median market capitalization) firms, respectively.

The results presented in column 1 show that the overall RMSE for GNN and GNN-attention is 0.837 and 0.828, respectively, which is less than half that of other algorithms, such as CNN at 2.088. CNN serves as an appropriate comparison to GNN, as both utilize neural network architectures. However, GNN incorporates edge inputs in addition to node characteristics, whereas CNN relies solely on the latter. Under identical training conditions, GNN models substantially outperform CNN, highlighting the crucial role of network effects in CDS prediction.

The results in columns 2 and 3 demonstrate enhanced prediction accuracy of GNN models for both investment-grade and high-yield firms. Interestingly, while classic algorithms consistently show better accuracy for high-yield firms (e.g., SVR with RMSE of 1.964 for high-yield vs. 2.150 for investment-grade), this pattern is reversed for GNN models. For instance, GNN achieves an

RMSE of 0.752 for investment-grade firms compared to 0.978 for high-yield firms. This suggests that the inter-firm risk spillover measure is particularly effective in determining CDS spreads for investment-grade firms. Intuitively, investment-grade firms, which are generally stronger and have extensive trade relationships with other firms, typically occupy central positions in the financial network. The measure of inter-firm idiosyncratic volatility spillover captured by GNN models can effectively cover the necessary network effects that influence these firms' CDS spreads. Columns 4 and 5 show that GNN models improve prediction accuracy for both small and large firms compared to classic machine learning algorithms. For small firms, the RMSE decreases from 2.133 with CNN to 0.807 and 0.810 with GNN and GNN-attention, respectively. Similar improvements are observed for large firms.

Overall, Table 2 shows that GNN models yield superior out-of-sample prediction accuracy consistently across firm ratings and sizes. This underscores the importance of considering network effects, specifically the idiosyncratic risk spillover among firms, when assessing CDS spreads.

4.2 Attention-layers

4.2.1 Node Attention

We examine the node attention scores in Equation (10) from the GNN-attention model with a single hidden layer. These scores form an $n \times 1$ vector revealing the attention weight assigned to each node (i.e., firm). For each out-of-sample testing month, we can output the normalized node attention, resulting in time-varying total attention. Let $\mathcal{N}_{T\times n}^{firm}$ denote the node attention matrix, where T is the number of time periods, and n is the number of firms. This matrix comprises n-dimensional column vectors, each corresponding to the node attention at a time point t. The node attention layer illuminates which firms receive more attention from the neurons, thereby identifying key firms in

the GNN forecasting setting.

Assuming each firm represents its sector, we can interpret the importance of a firm in terms of its sector's importance. To provide a coherent interpretation of the node attention results, we aggregate the firm-level attention to the sector level. This aggregation allows us to identify pivotal sectors in the GNN model and enhance our understanding of the financial network. We denote the sector-level node attention as matrix $\mathcal{N}_{T\times k}^{sector}$, where k is the number of sectors. We derive this sector-level matrix from the firm-level matrix using a mapping function $\mathcal{T}_{n\times k}$:

$$\mathcal{T}_{n\times k,t}(i,j) = \begin{cases} 0 & \text{if firm } i \text{ does not belong to sector } j \text{ at time } t \\ \text{Firm } i\text{'s value weight in sector } j & \text{if firm } i \text{ belongs to sector } j \text{ at time } t \end{cases} \tag{15}$$

At each time point t, we derive the sector-level node attention matrix $\mathcal{N}_{1\times k,t}^{sector}$ by multiplying the firm-level node attention matrix $\mathcal{N}_{1\times n,t}^{firm}$ with the mapping matrix $\mathcal{T}_{n\times k,t}$:

$$\mathcal{N}_{1\times k,t}^{sector} = \mathcal{N}_{1\times n,t}^{firm} \times \mathcal{T}_{n\times k,t}. \tag{16}$$

The mapping function $\mathcal{T}_{n\times k}$ in Equation (16) transforms the $T \times n$ firm-level node attention matrix into a $T \times k$ sector-level node attention matrix. Each element within $\mathcal{T}_{n\times k}$ allocates the node attention of a firm to its respective sector, considering the firm's affiliation and its relative importance (i.e., value weight) within that sector. Consequently, the resulting $\mathcal{N}_{T\times k}^{sector}$ matrix offers a comprehensive view of sector-level node attention over time, extrapolated from the firm-level attention data.

FIGURE 5 ABOUT HERE

Figure 5 presents the time series distribution of the sector-level node attention. Notably, two sectors consistently receive heightened attention from the GNN algorithm: the Finance and

Insurance sector and the Manufacturing sector. Intuitively, these two sectors are relatively central in a production-based trade network. In the input-output network, each sector produces unique goods used as inputs for their customers and the customers of customers. Sectors like manufacturing and financial sectors, consistently produce general goods that serve as common inputs to other sectors.[18] Consequently, the health of manufacturing and financial firms can impact more of the CDS spreads. Additionally, financial firms lend to other sectors, affecting their financial well-being. Therefore, by paying more attention to the health of manufacturing and intermediary sectors, we can achieve better prediction for CDS spreads.

Apart from the manufacturing and finance insurance sectors, the construction and mining oil sectors (or the 'energy' sector) also stand out at certain periods, though not as consistently throughout the time. These sectors are important in tail risk spillover. According to Dew-Becker (2023), sectors providing complementary inputs to others can transmit extremely negative shocks downstream to their customers. The construction and mining oil sectors, supplying complementary goods to various industries, can thus play a key role in transmitting tail risk. Therefore, by paying attention to these sectors, the GNN algorithm can improve CDS prediction accuracy.[19]

4.2.2 Edge Attention

We further examine the edge attention scores from the GNN-attention model with a single hidden layer. We output the multiplication item '$\left(D^{-1}\hat{A}\right) \odot$ edge attention' from Equation (11), which is an $n \times n$ matrix, to reveal the attention scores assigned to each inter-firm edge. For each out-of-sample testing month, we can output the edge attention, resulting in time-varying total attention. We denote

[18] This is supported by empirical evidence in Acemoglu et al. (2012), which shows that the output distribution always exhibits a heavy tail as certain sectors like manufacturing and financial services always serve as the largest suppliers.

[19] However, it's important to acknowledge that we are not claiming the GNN algorithm always accurately focuses on the economically meaningful nodes in the real world. While the node attention results generally align with intuitive expectations, there may be instances where the algorithm's focus does not perfectly correspond to real-world events.

$\mathcal{E}^{firm}_{n\times n,t}$ as the edge attention matrix among n firms at time t. In this matrix, each element $\mathcal{E}^{firm}_{i,j,t}$ represents the attention that the neural network allocates to the edge (i.e., idiosyncratic volatility spillover) from firm j to firm i at time t.

To provide a more comprehensive view of the edge attention results, we aggregate the firm-level edge attention to the sector level. We denote the sector-level edge attention matrix as $\mathcal{E}^{sector}_{k\times k,t}$, where k is the number of sectors. The sector-level edge attention is derived by extrapolating from firm-level attention:

$$\mathcal{E}^{sector}_{k\times k,t} = \left(\mathcal{E}^{firm}_{n\times n,t} \times \mathcal{T}_{n\times k,t}\right)^T \times \left(\mathcal{E}^{firm}_{n\times n,t} \times \mathcal{T}_{n\times k,t}\right), \tag{17}$$

where $\mathcal{T}_{n\times k}$ is defined in Equation (15). Each entry in the sector-level edge attention matrix, $\mathcal{E}^{sector}_{k\times k,t}$ in Equation (17), is extrapolated from the firm-level attention by considering the firm's affiliation and its relative importance (i.e., value weight) within that sector. By examining the sector-level edge attention matrix, we can identify key inter-sector edges (i.e., idiosyncratic volatility spillover edges) that play a significant role in predicting CDS spreads.

By compiling the edge attention data across different time points, we can construct a three-dimensional plot that shows the dynamic attention allocated to the $k \times k$ inter-sector edges. To ensure comparability of the time series edge attention data, we also normalize the edge attention at each time point.

FIGURE 6 ABOUT HERE

Figure 6 presents a three-dimensional representation of the $k \times k$ edge attention matrix, stacked along the time series (z-axis). At each time point, the x-axis (visible from the right-hand side of the 3D cube) and the y-axis (visible from the left-hand side) indicate the relative attention that the neural network pays to each inter-sector edge. The point (i, j, t) in the cube represents the attention given to the idiosyncratic volatility spillover edge from sector j to sector i at time t.

To facilitate a clearer understanding of which edges receive more attention at each time point, we examine cross-sectional cuts of the 3-D cube at different time periods. Figure 7 presents cross-sectional snapshots of the 3D sector-to-sector attention. Panels (1)-(4) display the attention patterns in January of 2012, 2014, 2016, and 2018, respectively.

FIGURE 7 ABOUT HERE

Panel (1) of Figure 7 shows that for the 2012 predicting period, the algorithm pays more attention to edges between other sectors and the information sector or the retail trade sector. Intuitively, there were significant technological advancements and changes in consumer behavior during this period, with various sectors increasingly interacting with the information and retail trade sectors. This suggests we should pay more attention to these sectors' impact on others.

Panel (2) shows that in 2014 there is a more complex pattern of edge attention, with notable edges to the construction, administrative, and real estate sectors. This complexity mirrors the intricate nature of financial networks, influenced by diverse factors like economic conditions, government policies, and global events.

Panels (3) and (4) show that in more recent years, the neurons pay more attention to edges with the retail trade, finance insurance, and health sectors. Intuitively, this reflects the growing importance of intermediary sectors and a heightened focus on health-related issues in recent years. By prioritizing these critical inter-sector edges, the GNN algorithm can greatly enhance the accuracy of CDS spread predictions.

Across all four cross-sectional plots, it's evident that the financial network is dynamic, with varying importance placed on different inter-sector edges over time. Incorporating idiosyncratic volatility spillover information among sectors and focusing on key inter-sector edges can greatly help in accurately predicting CDS spreads. Sectors like retail trade, construction, and finance

insurance, often linked to economic state or supplier-customer conditions, play a significant role in CDS forecasting. Our analysis sheds light on the importance of identifying and emphasizing these crucial edges in financial market predictions.

4.3 Statistical Comparison of Predictive Power

To compare the predictive power of various machine learning algorithms, we apply the Diebold and Mariano (DM) test (Diebold and Mariano, 2002). The DM statistic is calculated as follows:

$$\begin{cases} d_{A1,A2,t+1} = \frac{1}{N_{i,t+1}} \sum_{i} \left(e^2_{1,i,t+1} - e^2_{2,i,t+1} \right) \\ d_{A1,A2} = \frac{1}{T} \sum_{t+1} d_{A1,A2,t+1} \end{cases} \tag{18}$$

where $N_{i,t+1}$ is the number of firms at time $t+1$ and T is the total number of time periods. In equation (18), $e_{1,i,t+1}$ and $e_{2,i,t+1}$ are the prediction errors for each firm i at time $t+1$ using two different machine learning algorithms $A1$ and $A2$. The Newey-West standard error $\sigma_{A1,A2}$ of the sequence $d_{A1,A2,t+1}$ is computed with a lag k[20], and the DM statistic is defined as:

$$DM = \frac{d_{A1,A2}}{\sigma_{A1,A2}}. \tag{19}$$

Under the null hypothesis that there is no difference between the two algorithms, the DM statistic in Equation (19) follows a standard normal distribution.

Given that the DM test tends to reject the null hypothesis with small sample sizes, we also apply

[20] A common practice is to use the value $k = n^{\frac{1}{3}} + 1$. In our reporting, we use lag k=6.

the HLN test (Harvey, Leybourne, and Newbold, 1998):

$$HLN = DM\sqrt{\frac{T+1-2k+k(k-1)}{T}} \sim \text{two-tailed t distribution (T-1)}, \tag{20}$$

where T is the number of time periods and k is the lag used in the Newey-West standard error calculation. The HLN test in Equation (20), provides a small-sample correction to the DM test.

TABLE 3 ABOUT HERE

Table 3 presents pairwise comparisons of out-of-sample firm-level prediction performance across thirteen models. Panel A reports Diebold-Mariano (DM) test statistics, while Panel B shows HLN test statistics. Positive values indicate the column model outperforms the row model. P-values are provided in parentheses. Both panels show that GNN and GNN-attention significantly outperform all other algorithms, with GNN-attention slightly outperforming GNN (DM statistic of 1.89, p-value of 0.06; HLN statistic of 2.05, p-value of 0.04).

These results indicate that in predicting CDS spreads, dimension reduction methods and random forest perform well while graph neural networks consistently show the best performance. This implies that network effects are crucial in predicting firm CDS spreads.

4.4 Long-Short Strategy

In this section, we explore the asset pricing implications of machine learning forecasts, as CDS itself is a tradable asset. We construct a new set of CDS portfolios based on 1-month-ahead out-of-sample predictions generated by each method at the end of each month. Firms are allocated to value-weighted quintile portfolios according to their predicted CDS spread for the following month. For each algorithm, we report the predicted mean and standard deviation, as well as the true mean

and standard deviation of log CDS spreads.

TABLE 4 ABOUT HERE

Table 4 presents the performance of CDS portfolios sorted based on predicted spreads during the out-of-sample testing period. Among all classic machine learning techniques, Random Forest (RF) performs well in capturing the true volatility of CDS for the most extreme quintile, but it overshoots the mean of the CDS for each quintile. CNN overestimates both the portfolio mean and standard deviation.

Remarkably, for GNN and GNN-attention algorithms, there is an extraordinary quantitative match between the predicted and realized CDS means and volatilities, both overall and for the high-low (H-L) quintile. The long-short strategy based on GNN and GNN-attention predictions generates significant return spreads, which indicates that it is necessary to take into account the network effects when trading CDSs.

4.5 Variable Importance

Lastly, we investigate the relative importance of individual covariates for the performance of each machine learning model. Specifically, for each algorithm, we assess the impact of each predictor by calculating the increase in Root Mean Square Error (RMSE) when all values of a given predictor are set to zero within each training sample. These increases are then averaged to derive a single importance measure for each predictor. The variable importance within each model is normalized to sum to one, facilitating the interpretation of relative importance for that particular model.

FIGURE 8 ABOUT HERE

Figure 8 reports the overall rankings of firm-level characteristics across all models. We rank the importance of each characteristic for each method and then sum their ranks. The characteristics are ordered such that those with the highest total ranks appear at the top, and those with the lowest ranks are at the bottom. The color gradient within each column indicates the model-specific ranking of characteristics from most to least important (darkest to lightest).

The models generally agree on the most influential predictors. One group of important predictors includes risk measures such as total volatility (retvol), dollar volume volatility (std_dolvol), earnings volatility (roavol), market beta (beta), and idiosyncratic return volatility (idiovol). Another important group comprises liquidity-related measurements including Amihud illiquidity (ill), turnover volatility (std_turn), log market equity (mvel1), number of zero trading days (zerotrade), bid-ask spread (baspread), cash holdings (cash), and current ratio (currat). Dimension reduction models tend to heavily favor risk-based variables, while neural networks draw predictive information from a broader set of characteristics.

We further show the relative importance of 8 macro variables across different machine learning algorithms.

FIGURE 9 ABOUT HERE

Figure 9 presents the overall rankings of macroeconomic characteristics across all models. Among all algorithms, three macro variables stand out as the most important: stock variance (svar), net equity expansion (ntis), and the earnings-price ratio (ep). These variables are more related to the stock market.

It is observed that Partial Least Squares (PLS) and Principal Component Regression (PCR) assign similar weights to various predictors with slightly more weight to the earnings-price ratio (ep). Tree models like Gradient Boosting Regression Trees (GBRT) and Random Forest (RF) assign more weight to term spread (tms). GNNs assign weights to a broader set of variables due

to the high correlation among them. The GNNs could pick up some bond market-related variables that are neglected by other algorithms, such as the Treasury-bill rate (tbl) and the default spread (dfy).

5 Conclusion

This paper explores the role of intricate network effects in predicting CDS spreads using a machine learning framework. By leveraging Graph Neural Networks (GNNs), we incorporate both node characteristics and inter-firm edge characteristics to predict cross-sectional CDS spreads. The GNN reduces out-of-sample prediction error by more than half compared to Convolutional Neural Networks (CNN), which we use as a benchmark, highlighting the critical importance of network effects in CDS prediction.

We further enhance GNNs by integrating node and edge attention mechanisms, effectively opening the "black box" of GNNs and revealing how neurons focus on specific nodes and edges. The temporal node and edge attention from GNN-attention identifies (1) key nodes, such as firms in the Finance, Insurance, and Manufacturing sectors, and (2) significant linkages, particularly between intermediary firms, retail trade, or information firms and others, in determining the cross-section of firms' CDS spreads.

Our paper makes a significant contribution to CDS prediction by employing advanced machine learning techniques and enriching CDS pricing models with insights from financial networks. The findings offer important policy implications: policymakers should target key firms and critical inter-firm relationships to prevent market-wide contagion effects.

Finally, our results justify the growing role of machine learning—particularly algorithms that account for network effects—within the architecture of the fintech industry. Network effects pervade

the financial system. They exist not only among assets and firms, but also among agents and firm decision-makers. Future research could extend this paper to explore network applications in these areas.

Table 1. Summary Statistics of CDS Contracts and Issuing Firms
This table presents summary statistics for CDS contracts and their issuing firms from January 2005 to December 2020. Columns 1-2 show CDS contract duration statistics. Columns 3-4 show firms' market capitalization statistics. Columns 5-6 present the distribution of firms' credit ratings.

CDS Duration		Firm Market Capitalization		Firm Rating Distribution	
	Month		Dollar($)	Rating	Firm Count
mean	117	mean	2.65×10^7	AA	97
std	69	std	5.38×10^7	A	144
min	2	min	7.01×10^2	BBB	192
25%	50	25%	3.57×10^6	BB	156
50%	137	50%	1.01×10^7	B	88
75%	188	75%	2.64×10^7	CCC	59
max	192	max	1.71×10^9	D	3

Table 2. Pooled Out-of-Sample RMSE

This table presents the pooled out-of-sample root mean square error (RMSE) for various machine learning algorithms. Column 1 shows RMSE for the entire sample from February 2005 to December 2020. Columns 2 and 3 report RMSE for investment-grade (BBB and above) and high-yield (BB and below) firms, respectively. Columns 4 and 5 display RMSE for small (below median market capitalization) and large (at or above median market capitalization) firms, respectively.

	All	Investment Grade	High Yield	Small	Big
SVR	2.087	2.150	1.964	1.640	1.498
PCR	2.198	2.245	2.107	1.955	1.400
PLS	1.554	1.660	1.332	1.342	1.072
GBRT	1.744	1.838	1.550	1.514	1.287
RF	1.787	2.025	1.217	1.410	1.858
CNN	2.088	2.126	2.014	2.133	1.494
GNN	0.837	0.752	0.978	0.807	0.797
GNN_attention	0.828	0.735	0.981	0.810	0.776

Table 3. Pairwise Comparison of Out-of-Sample Prediction Performance
This table presents pairwise comparisons of out-of-sample firm-level prediction performance across thirteen models. Panel A reports Diebold-Mariano (DM) test statistics, while Panel B shows HLN test statistics. Positive values indicate the column model outperforms the row model. P-values are provided in parentheses.

Panel A: DM Test

	SVR	PCR	PLS	GBRT	RF	CNN	GNN	GNN_attention
SVR		-1.33	4.90	3.17	3.21	0.17	7.01	7.02
		(1.82)	(0.00)	(0.00)	(0.00)	(0.86)	(0.00)	(0.00)
PCR			15.09	11.83	11.21	0.48	13.52	13.45
			(0.00)	(0.00)	(0.00)	(0.63)	(0.00)	(0.00)
PLS				-6.12	-16.08	-1.30	11.61	11.47
				(2.00)	(2.00)	(1.81)	(0.00)	(0.00)
GBRT					-1.23	-0.82	11.17	11.06
					(1.78)	(1.59)	(0.00)	(0.00)
RF						-0.71	13.92	13.74
						(1.52)	(0.00)	(0.00)
CNN							2.74	2.75
							(0.01)	(0.01)
GNN								1.89
								(0.06)

Panel B: HLN Test

	SVR	PCR	PLS	GBRT	RF	CNN	GNN	GNN_attention
SVR		-1.44	5.32	3.43	3.48	0.19	7.60	7.61
		(0.15)	(0.00)	(0.00)	(0.00)	(0.85)	(0.00)	(0.00)
PCR			16.36	12.83	12.15	0.52	14.66	14.59
			(0.00)	(0.00)	(0.00)	(0.61)	(0.00)	(0.00)
PLS				-6.63	-17.43	-1.41	12.59	12.44
				(0.00)	(0.00)	(0.16)	(0.00)	(0.00)
GBRT					-1.33	-0.89	12.11	12.00
					(0.19)	(0.37)	(0.00)	(0.00)
RF						-0.77	15.09	14.90
						(0.44)	(0.00)	(0.00)
CNN							2.97	2.99
							(0.00)	(0.00)
GNN								2.05
								(0.04)

Table 4. Performance of Prediction-Sorted CDS Portfolios

This table presents the performance of CDS portfolios sorted based on predicted spreads during the out-of-sample testing period. Firms are allocated to value-weighted quintile portfolios according to their predicted CDS spread for the following month. For each algorithm, we report the predicted mean and standard deviation, as well as the true mean and standard deviation of log CDS spreads.

	SVR					PCR			
	Prediction		True			Prediction		True	
	mean	std	mean	std		mean	std	mean	std
L	-6.17	0.42	-5.60	0.18	L	-5.27	0.13	-5.58	0.18
2	-4.81	0.28	-5.28	0.21	2	-4.16	0.11	-5.17	0.23
3	-3.84	0.20	-4.99	0.25	3	-3.31	0.09	-4.71	0.21
4	-2.72	0.33	-4.77	0.28	4	-2.22	0.21	-4.37	0.24
H	-0.81	1.07	-4.79	0.38	H	-0.70	0.44	-4.95	0.33
H-L	5.35	1.44	0.80	0.33	H-L	4.57	0.43	0.63	0.24
	PLS					GBRT			
	Prediction		True			Prediction		True	
	mean	std	mean	std		mean	std	mean	std
L	-5.33	0.09	-5.60	0.19	L	-5.30	0.21	-5.61	0.18
2	-4.43	0.10	-5.16	0.18	2	-4.49	0.15	-5.15	0.20
3	-3.77	0.08	-4.69	0.20	3	-3.69	0.17	-4.87	0.26
4	-2.96	0.10	-4.36	0.24	4	-2.61	0.22	-4.73	0.37
H	-1.94	0.28	-4.89	0.37	H	-1.47	0.31	-4.82	0.38
H-L	3.39	0.35	0.70	0.26	H-L	3.83	0.44	0.79	0.32
	RF					CNN			
	Prediction		True			Prediction		True	
	mean	std	mean	std		mean	std	mean	std
L	-3.73	0.12	-5.56	0.17	L	-5.33	1.10	-5.58	0.18
2	-3.38	0.09	-5.08	0.24	2	-4.32	1.10	-5.20	0.24
3	-3.17	0.09	-4.78	0.24	3	-3.56	1.23	-4.79	0.25
4	-2.90	0.19	-4.36	0.23	4	-2.70	1.37	-4.52	0.33
H	-2.40	0.22	-4.91	0.34	H	-1.71	1.36	-4.79	0.38
H-L	1.33	0.30	0.66	0.27	H-L	3.62	1.13	0.79	0.29
	GNN					GNN_attention			
	Prediction		True			Prediction		True	
	mean	std	mean	std		mean	std	mean	std
L	-5.72	0.19	-5.54	0.18	L	-5.76	0.20	-5.61	0.24
2	-5.47	0.20	-5.37	0.23	2	-5.42	0.20	-5.35	0.24
3	-4.91	0.20	-4.72	0.20	3	-4.82	0.22	-4.77	0.24
4	-4.22	0.22	-4.03	0.29	4	-4.45	0.21	-4.21	0.25
H	-4.16	0.21	-4.01	0.24	H	-4.25	0.22	-4.06	0.27
H-L	1.56	0.15	1.53	0.18	H-L	1.51	0.23	1.55	0.27

Figure 1. Graph Neural Network (GNN) Architecture
This figure illustrates the GNN algorithm structure. Panel A depicts the intra-layer scheme, while Panel B shows the inter-layer scheme.

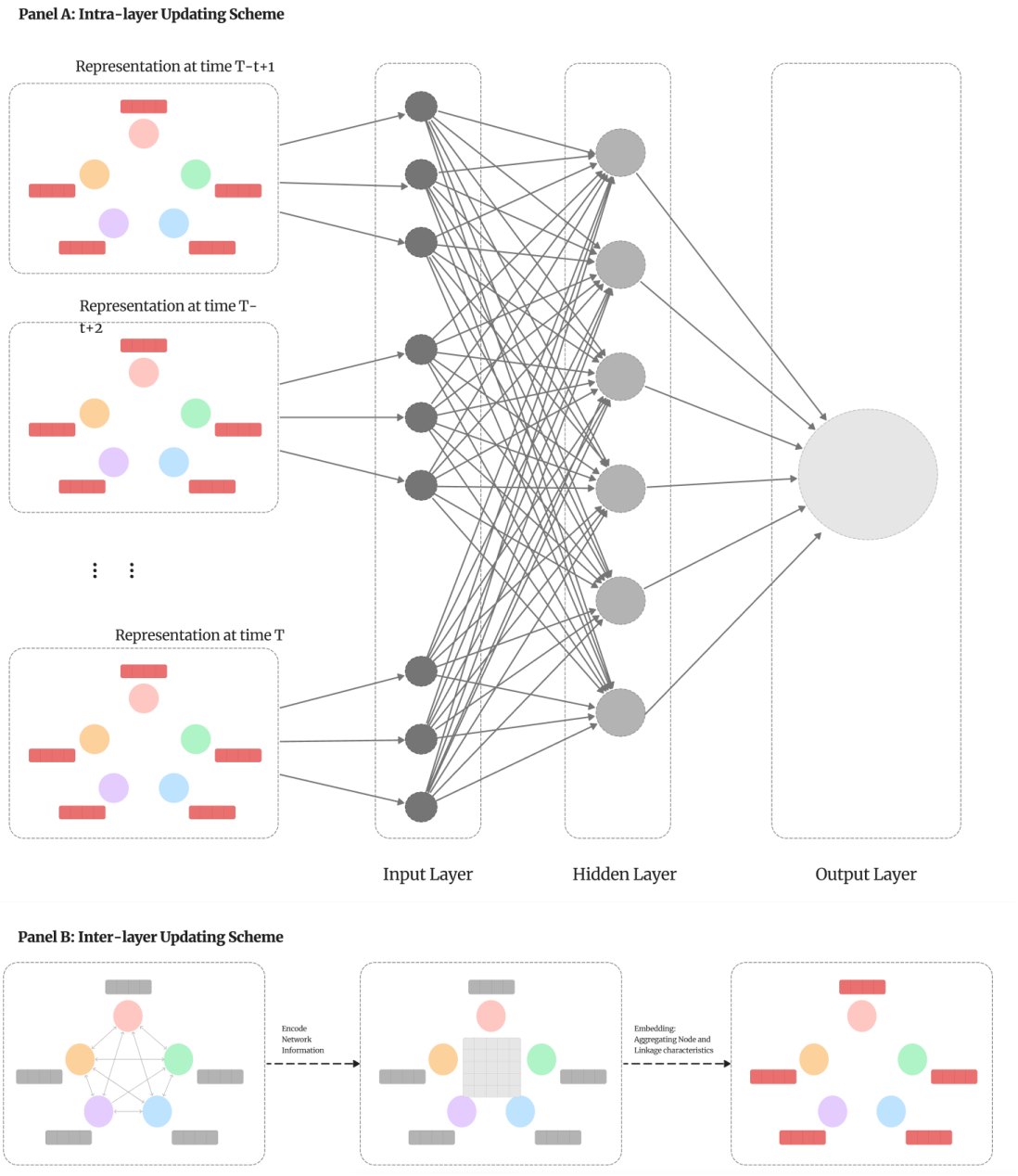
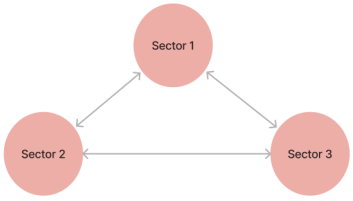


Figure 2. Extrapolation of Sector-Level Network Effects to Firm Level
This figure illustrates the process of extrapolating sector-level idiosyncratic risk spillover to the firm level. Panel A depicts risk spillover among three sectors, estimated from the Generalized Variance Decomposition (GVD) of VAR forecast errors. In the adjacency matrix, element (i,j) indicates the risk spillover intensity from sector j to sector i. Panel B demonstrates the extrapolation of this risk spillover to five firms across the three sectors.

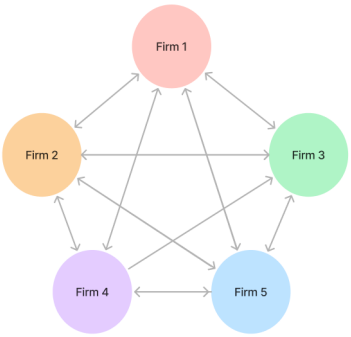
Panel A: Example for risk spillover among 3 sectors



	Sector 1	Sector 2	Sector 3
Sector 1	0.4	0.1	0.5
Sector 2	0.25	0.6	0.15
Sector 3	0.25	0.45	0.3

Panel B: Example for extrapolation

Firm	Sector 1	Sector 2	Sector 3
Firm 1	1	0	0
Firm 2	0	1	0
Firm 3	1	0	0
Firm 4	0	0	1
Firm 5	0	1	0



	Firm 1	Firm 2	Firm 3	Firm 4	Firm 5
Firm 1	0.4	0.1	0.4	0.5	0.1
Firm 2	0.25	0.6	0.25	0.15	0.6
Firm 3	0.4	0.1	0.4	0.5	0.1
Firm 4	0.25	0.45	0.25	0.3	0.45
Firm 5	0.25	0.6	0.25	0.15	0.6

Figure 3. Distribution of Log CDS Spreads
This figure displays the histogram of log CDS spreads for 5-year tenor, senior unsecured contracts from January 2005 to December 2020.

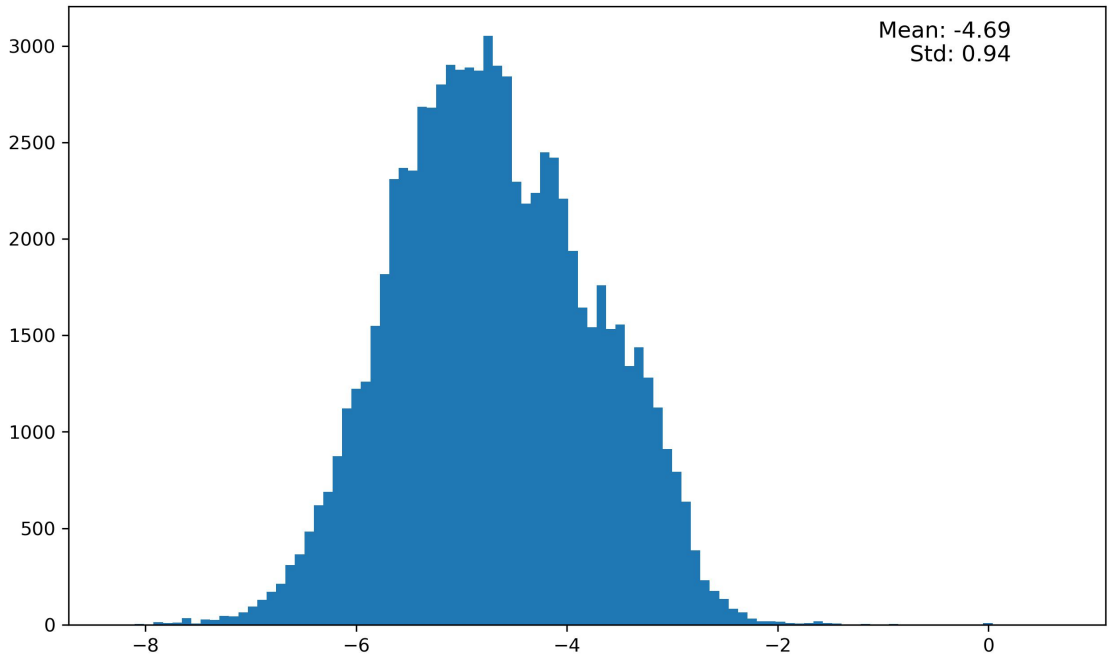
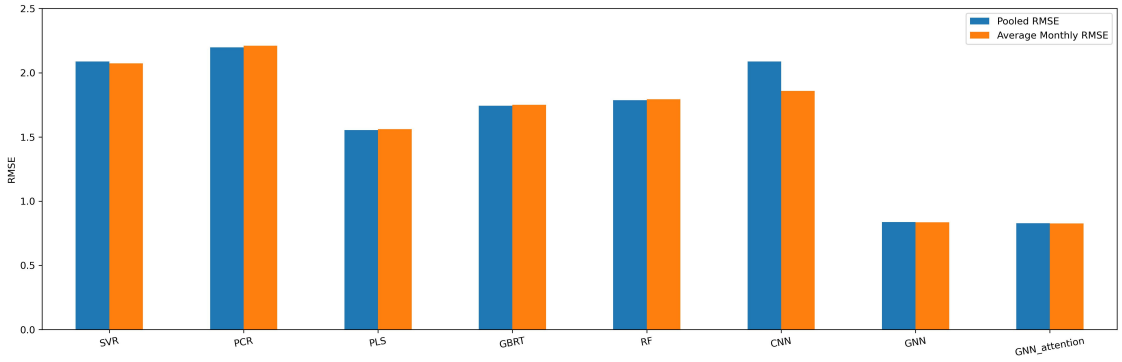
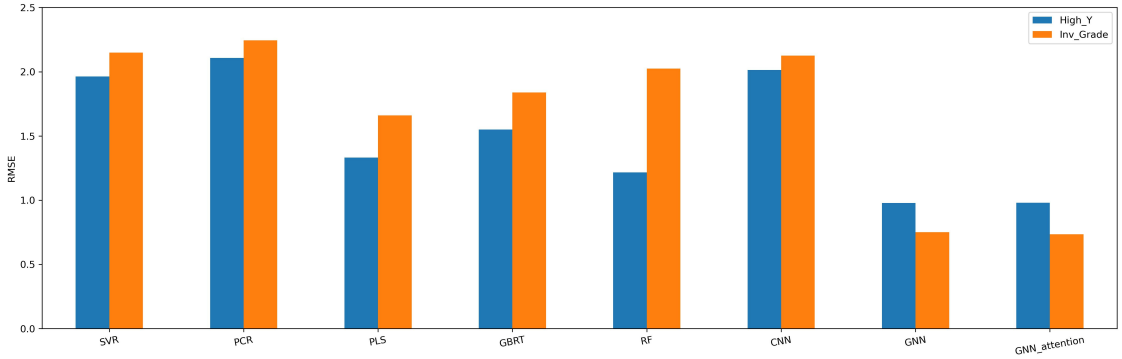


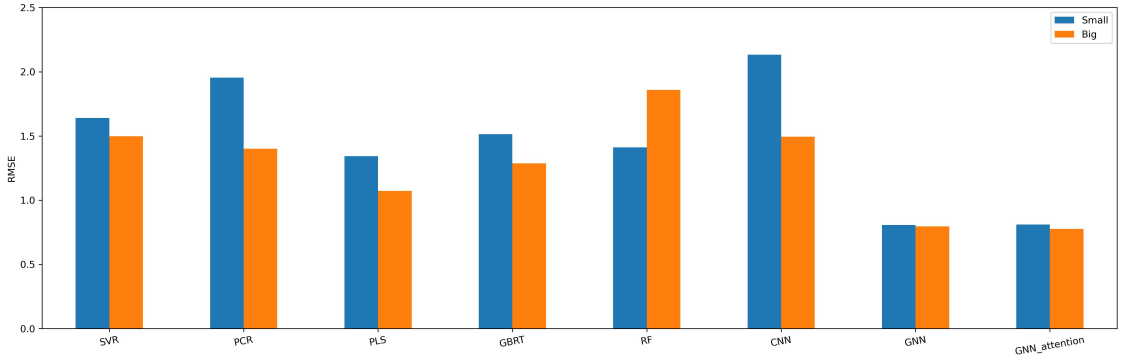
Figure 4. Out-of-Sample Prediction Error of Machine Learning Algorithms
This figure presents the out-of-sample prediction error of various machine learning algorithms using pooled Root Mean Square Error (RMSE). Panel A displays the overall RMSE across all out-of-sample periods and the average monthly RMSE. Panel B presents the overall RMSE by firm credit rating, with investment grade defined as BBB or above and high-yield as BB or below. Panel C shows the overall RMSE by firm size, with small firms defined as those below the median market capitalization and large firms as those at or above the median.



(a) Pooled RMSE and Average Monthly RMSE



(b) Pooled RMSE by Firm Credit Rating



(c) Pooled RMSE by Firm Market Capitalization

Figure 5. Sector-Level Node Attention in GNN-Attention Model
This figure illustrates the sector-level node attention derived from a single hidden layer of the GNN-attention model. For each out-of-sample (OOS) testing period, we normalize firm-level node attention values and aggregate them to the sector level (NAICS 2-digit). The figure displays the time series distribution of this sector-level attention.

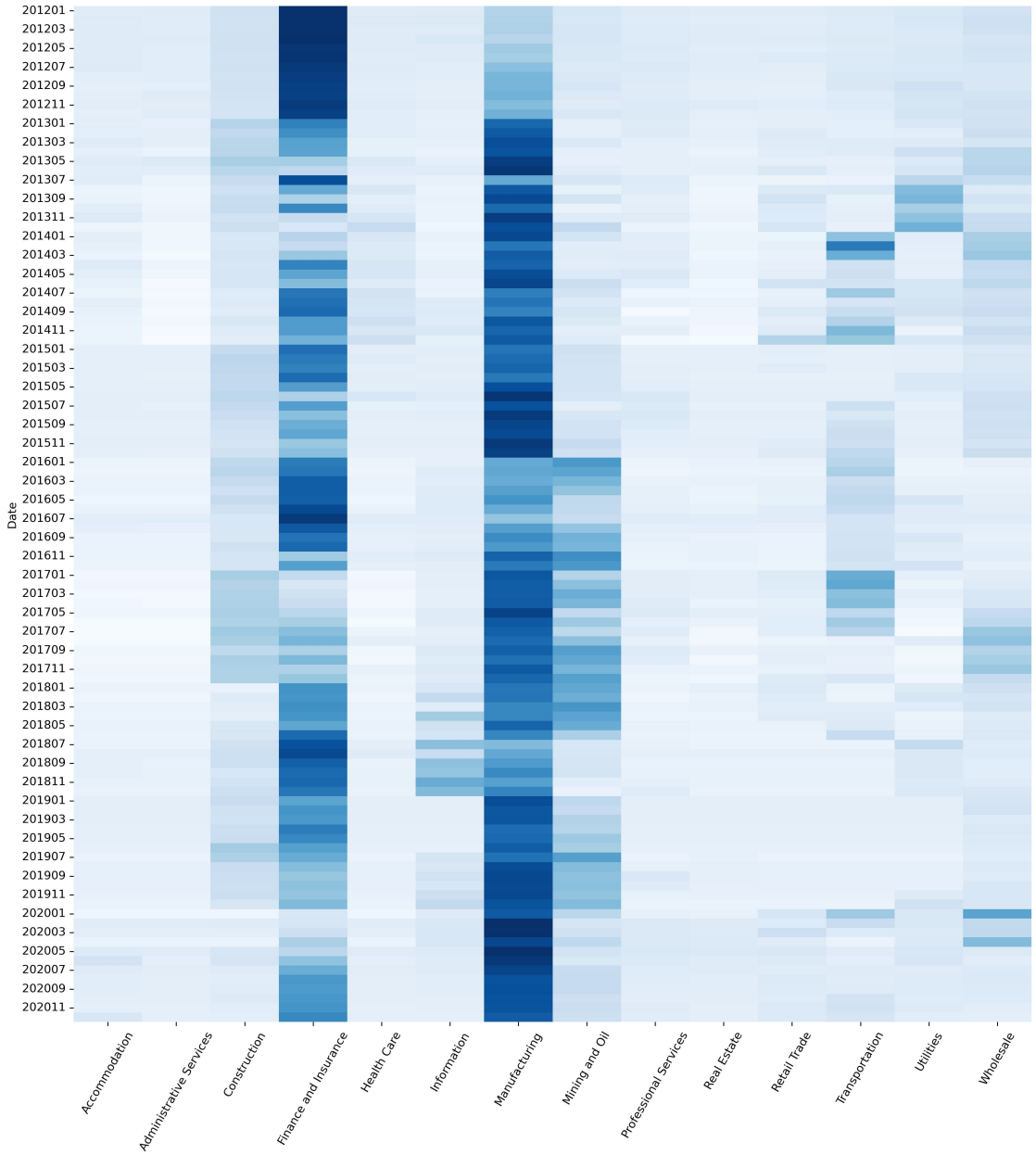


Figure 6. Sector-to-Sector Edge Attention in GNN-Attention Model
This figure illustrates the sector-to-sector edge attention derived from a single hidden layer of the GNN-attention model. For each out-of-sample (OOS) testing period, we normalize firm-level edge attention values and aggregate them to the sector level (NAICS 2-digit). The figure displays the time series distribution of this sector-to-sector attention.

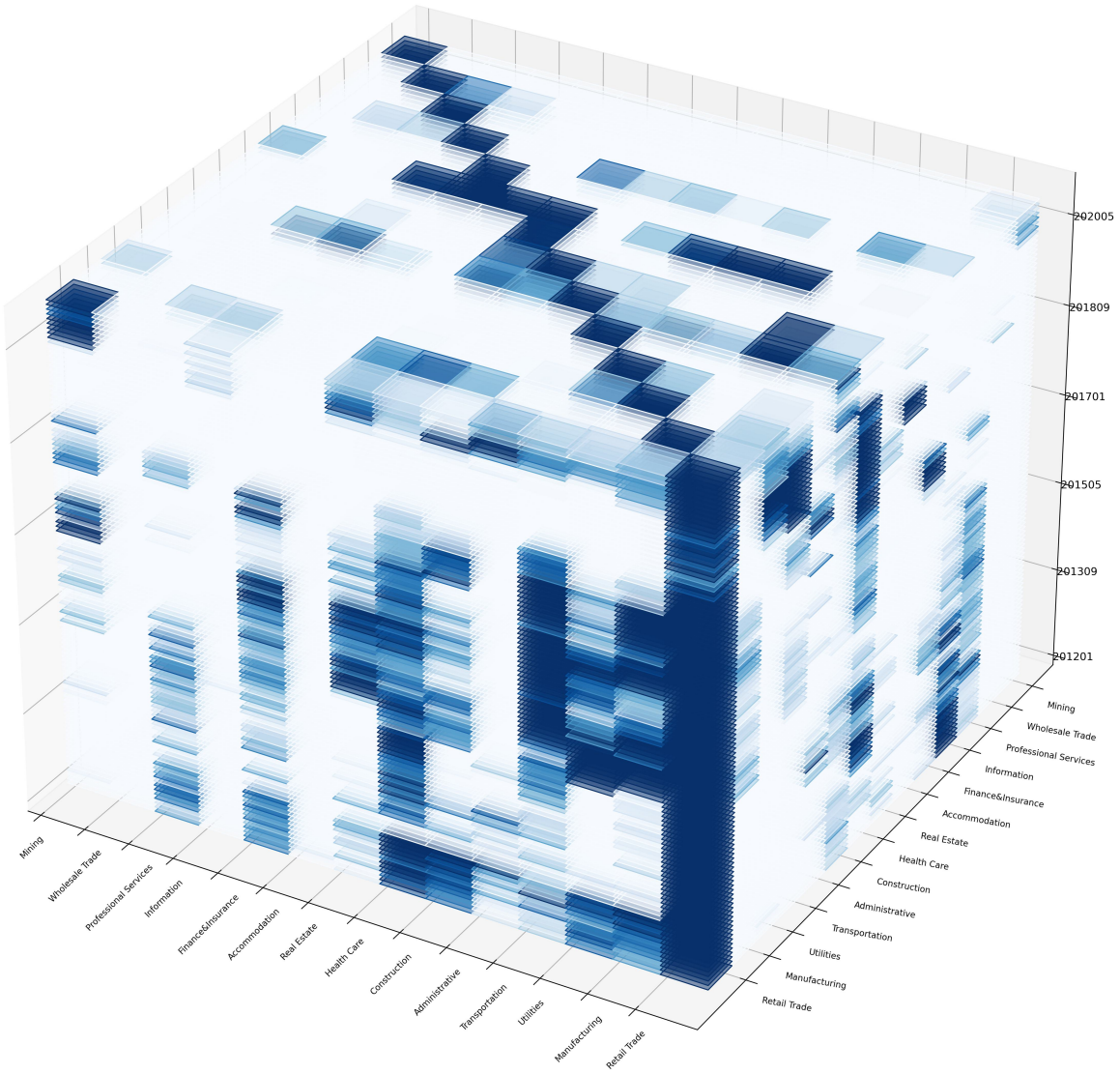


Figure 7. Cross-sectional Views of Sector-to-Sector Edge Attention
This figure presents cross-sectional snapshots of the 3D sector-to-sector attention. Panels (1)-(4) display the attention patterns in January of 2012, 2014, 2016, and 2018, respectively.

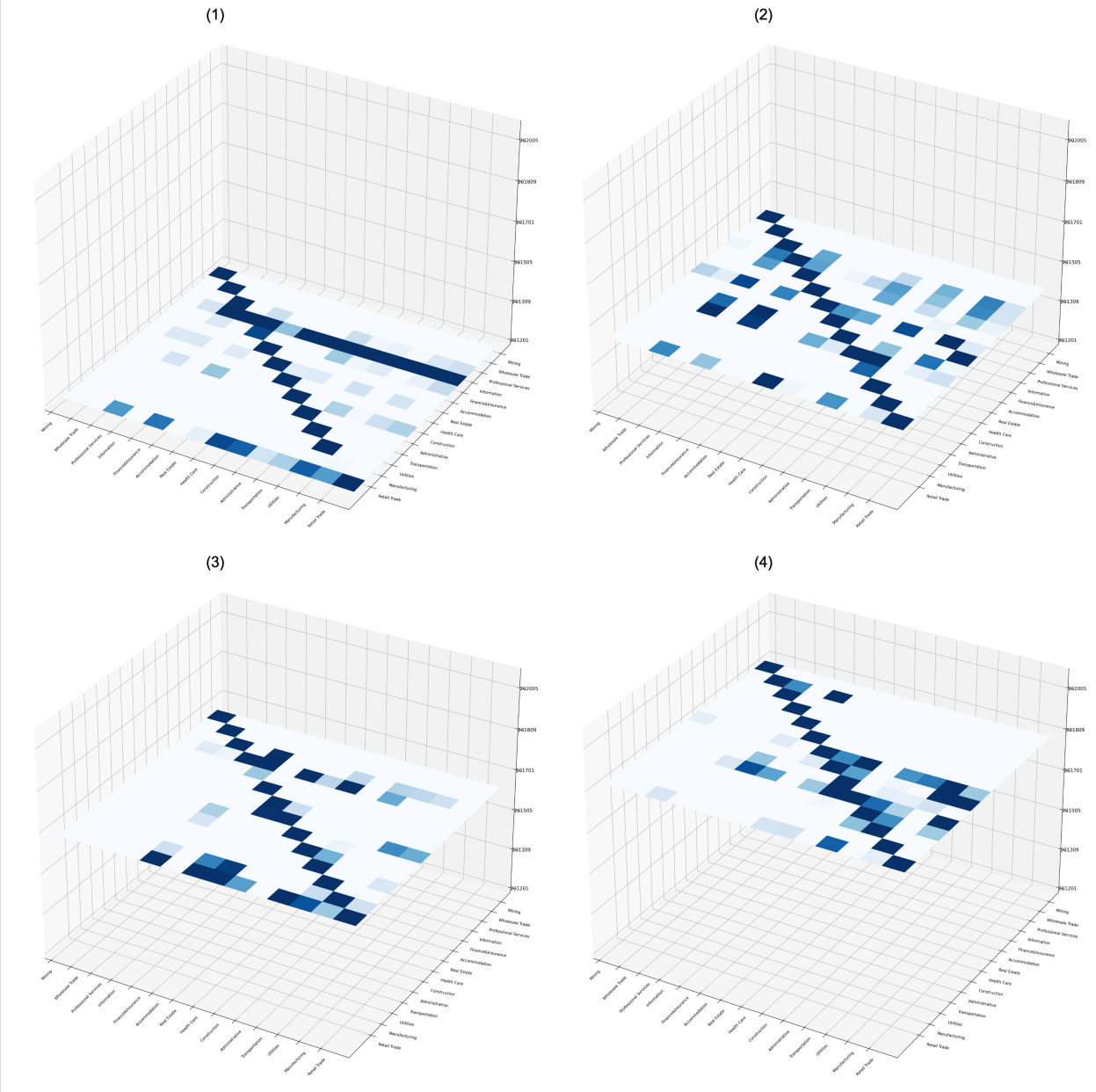


Figure 8. Relative Importance of Firm-Level Characteristics
This figure shows the relative importance of 94 firm-level characteristics across different algorithms. Importance is measured by the average increase in Root Mean Square Error (RMSE) when a predictor's values are set to zero in each training sample. Characteristics are ranked based on their summed importance across all methods, with the most important at the top. The color gradient (dark to light) within each column represents the model-specific ranking from most to least important.

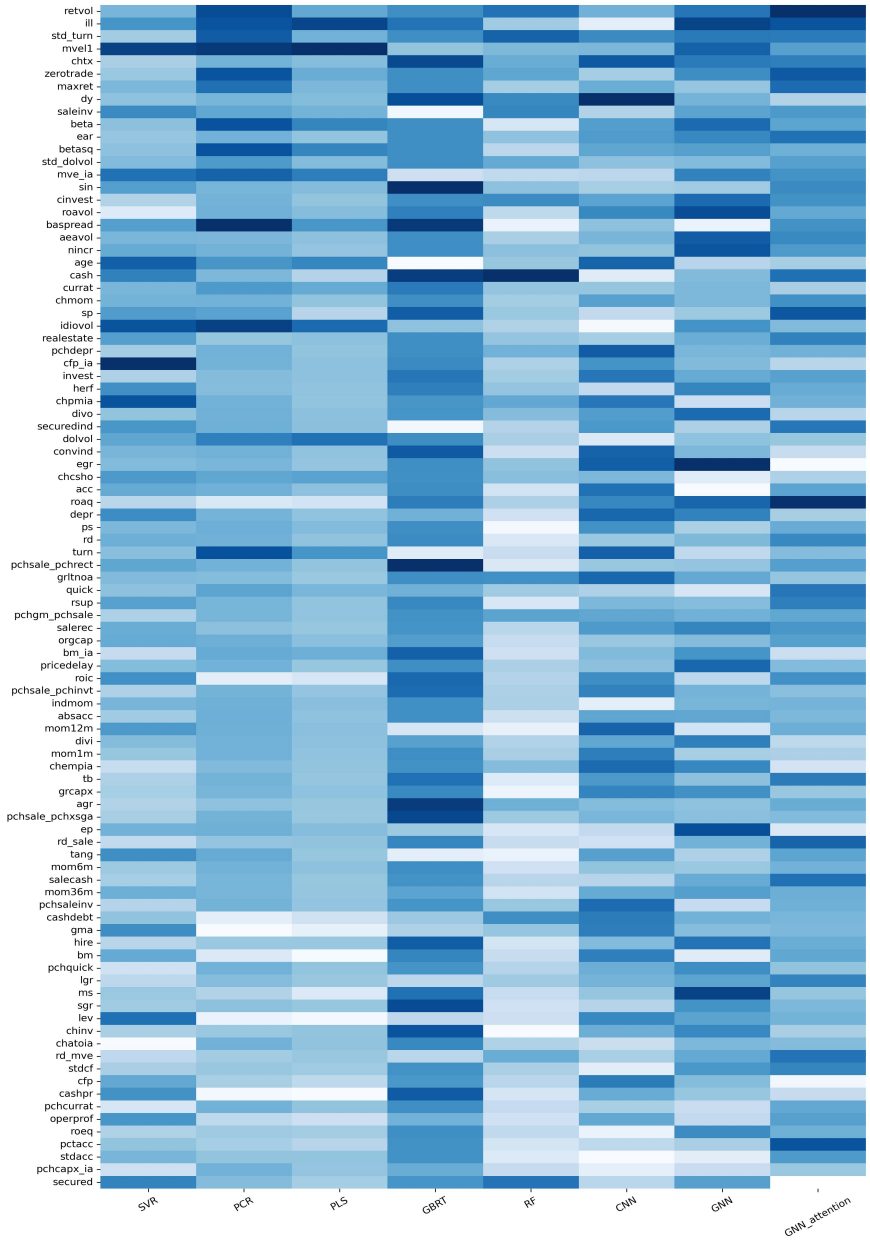
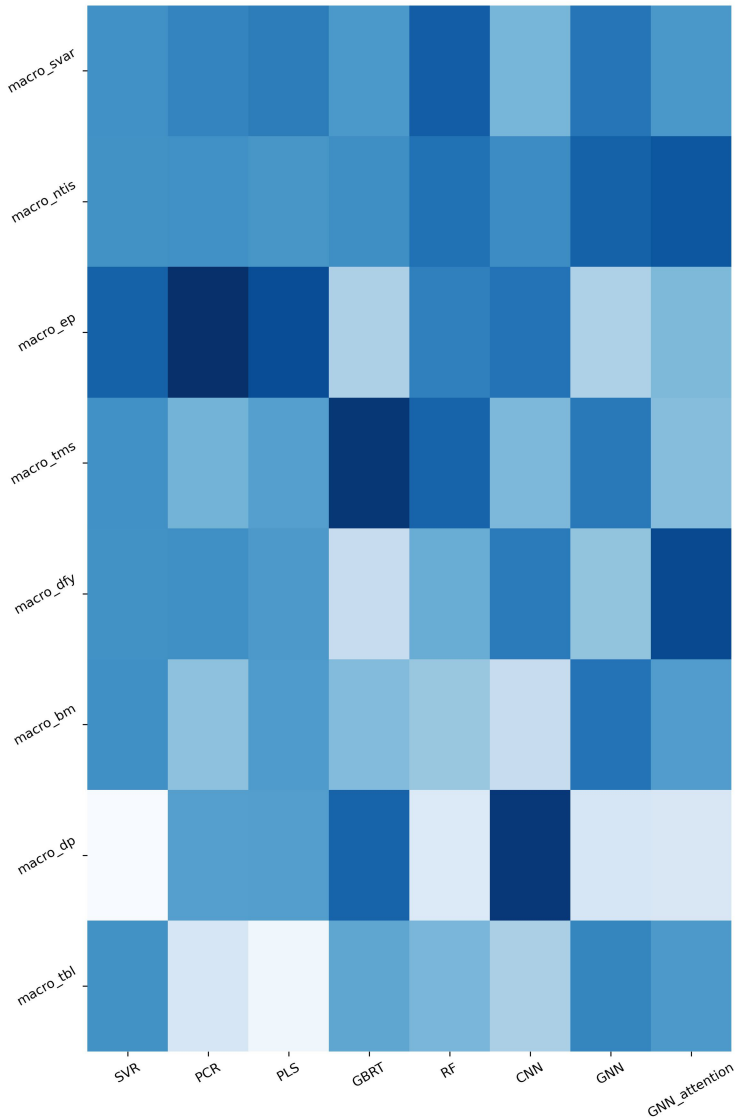


Figure 9. Relative Importance of Macroeconomic Characteristics
This figure shows the relative importance of 8 macroeconomic characteristics across different algorithms. Importance is measured by the average increase in Root Mean Square Error (RMSE) when a predictor's values are set to zero in each training sample. Characteristics are ranked based on their summed importance across all methods, with the most important at the top. The color gradient (dark to light) within each column represents the model-specific ranking from most to least important.



References

Acemoglu, Daron, Ufuk Akcigit, and William Kerr, 2016, Networks and the macroeconomy: An empirical exploration, *Nber Macroeconomics Annual* 30, 273–335.

Acemoglu, Daron, Vasco M Carvalho, Asuman Ozdaglar, and Alireza Tahbaz-Salehi, 2012, The network origins of aggregate fluctuations, *Econometrica* 80, 1977–2016.

Acemoglu, Daron, Asuman Ozdaglar, and Alireza Tahbaz-Salehi, 2017, Microeconomic origins of macroeconomic tail risks, *American Economic Review* 107, 54–108.

Almeida, Heitor, and Thomas Philippon, 2007, The risk-adjusted cost of financial distress, *The Journal of Finance* 62, 2557–2586.

Altman, Edward I, 1968, Financial ratios, discriminant analysis and the prediction of corporate bankruptcy, *The Journal of Finance* 23, 589–609.

Bao, Jack, Kewei Hou, and Shaojun Zhang, 2023, Systematic default and return predictability in the stock and bond markets, *Journal of Financial Economics* 149, 349–377.

Beaver, William H, 1966, Financial ratios as predictors of failure, *Journal of Accounting Research* 71–111.

Bharath, Sreedhar T, and Tyler Shumway, 2008, Forecasting default with the merton distance to default model, *The Review of Financial Studies* 21, 1339–1369.

Blasques, Francisco, Siem Jan Koopman, Andre Lucas, and Julia Schaumburg, 2016, Spillover dynamics for systemic risk measurement using spatial financial time series models, *Journal of Econometrics* 195, 211–223.

Breiman, Leo, 2001, Random forests, *Machine learning* 45, 5–32.

Campbell, John Y, Jens Hilscher, and Jan Szilagyi, 2008, In search of distress risk, *The Journal of Finance* 63, 2899–2939.

Carvalho, Vasco, and Xavier Gabaix, 2013, The great diversification and its undoing, *American Economic Review* 103, 1697–1727.

Carvalho, Vasco M, 2008, *Aggregate fluctuations and the network structure of intersectoral trade* (The University of Chicago).

Chen, Belinda, 2023, Network factors for idiosyncratic volatility spillover, *Available at SSRN 4579385* .

Chen, Cathy Yi-Hsuan, Wolfgang Karl Härdle, and Yarema Okhrin, 2019, Tail event driven networks of sifis, *Journal of Econometrics* 208, 282–298.

Das, Sanjiv R, Paul Hanouna, and Atulya Sarin, 2009, Accounting-based versus market-based cross-sectional models of cds spreads, *Journal of Banking & Finance* 33, 719–730.

Demirer, Mert, Francis X Diebold, Laura Liu, and Kamil Yilmaz, 2018, Estimating global bank network connectedness, *Journal of Applied Econometrics* 33, 1–15.

Dew-Becker, Ian, 2023, Tail risk in production networks, *Econometrica* 91, 2089–2123.

Diebold, Francis X, and Robert S Mariano, 2002, Comparing predictive accuracy, *Journal of Business & economic statistics* 20, 134–144.

Diebold, Francis X, and Kamil Yılmaz, 2014, On the network topology of variance decompositions: Measuring the connectedness of financial firms, *Journal of Econometrics* 182, 119–134.

Duan, Jin-Chuan, Jie Sun, and Tao Wang, 2012, Multiperiod corporate default prediction—a forward intensity approach, *Journal of Econometrics* 170, 191–209.

Duffie, Darrell, 1999, Credit swap valuation, *Financial Analysts Journal* 55, 73–87.

Engle, Robert, and Bryan Kelly, 2012, Dynamic equicorrelation, *Journal of Business & Economic Statistics* 30, 212–228.

Fama, Eugene F, and Kenneth R French, 2016, Dissecting anomalies with a five-factor model, *The Review of Financial Studies* 29, 69–103.

Freyberger, Joachim, Andreas Neuhierl, and Michael Weber, 2020, Dissecting characteristics nonparametrically, *The Review of Financial Studies* 33, 2326–2377.

Gabaix, Xavier, 2011, The granular origins of aggregate fluctuations, *Econometrica* 79, 733–772.

Galil, Koresh, Offer Moshe Shapir, Dan Amiram, and Uri Ben-Zion, 2014, The determinants of cds spreads, *Journal of Banking & Finance* 41, 271–282.

Gouriéroux, Christian, Alain Monfort, and Jean-Paul Renne, 2014, Pricing default events: Surprise, exogeneity and contagion, *Journal of Econometrics* 182, 397–411.

Green, Jeremiah, John RM Hand, and X Frank Zhang, 2017, The characteristics that provide independent information about average us monthly stock returns, *The Review of Financial Studies* 30, 4389–4436.

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

Gu, Shihao, Bryan Kelly, and Dacheng Xiu, 2021, Autoencoder asset pricing models, *Journal of Econometrics* 222, 429–450.

Härdle, Wolfgang Karl, Weining Wang, and Lining Yu, 2016, Tenet: Tail-event driven network risk, *Journal of Econometrics* 192, 499–513.

Harvey, David I, Stephen J Leybourne, and Paul Newbold, 1998, Tests for forecast encompassing, *Journal of Business & Economic Statistics* 16, 254–259.

Herskovic, Bernard, 2018, Networks in production: Asset pricing implications, *The Journal of Finance* 73, 1785–1818.

Herskovic, Bernard, Bryan Kelly, Hanno Lustig, and Stijn Van Nieuwerburgh, 2016, The common factor in idiosyncratic volatility: Quantitative asset pricing implications, *Journal of Financial Economics* 119, 249–283.

Herskovic, Bernard, Bryan Kelly, Hanno Lustig, and Stijn Van Nieuwerburgh, 2020, Firm volatility in granular networks, *Journal of Political Economy* 128, 4097–4162.

Hou, Kewei, Chen Xue, and Lu Zhang, 2020, Replicating anomalies, *The Review of financial studies* 33, 2019–2133.

Ioffe, Sergey, and Christian Szegedy, 2015, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in *International Conference on Machine Learning*, 448–456, PMLR.

Jacobson, Tor, and Erik Von Schedvin, 2015, Trade credit and the propagation of corporate failure: An empirical analysis, *Econometrica* 83, 1315–1371.

Jankowitsch, Rainer, Rainer Pullirsch, and Tanja Veža, 2008, The delivery option in credit default swaps, *Journal of Banking & Finance* 32, 1269–1285.

Jarrow, Robert A, and Fan Yu, 2001, Counterparty risk and the pricing of defaultable securities, *The Journal of Finance* 56, 1765–1799.

Kelly, Bryan, and Hao Jiang, 2014, Tail risk and asset prices, *The Review of Financial Studies* 27, 2841–2871.

Kelly, Bryan, Semyon Malamud, and Kangying Zhou, 2024, The virtue of complexity in return prediction, *The Journal of Finance* 79, 459–503.

Kelly, Bryan T, Seth Pruitt, and Yinan Su, 2019, Characteristics are covariances: A unified model of risk and return, *Journal of Financial Economics* 134, 501–524.

Kipf, Thomas N, and Max Welling, 2016, Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907* .

Kitwiwattanachai, Chanatip, 2015, Learning network structure of financial institutions from cds data, *Available at SSRN 2533606* .

Kitwiwattanachai, Chanatip, and Neil D. Pearson, 2015, Inferring Correlations of Asset Values and Distances-to-Default from CDS Spreads: A Structural Model Approach, *The Review of Asset Pricing Studies* 5, 112–154.

Kryzanowski, Lawrence, Stylianos Perrakis, and Rui Zhong, 2017, Price discovery in equity and cds markets, *Journal of Financial Markets* 35, 21–46.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton, 2015, Deep learning, *nature* 521, 436–444.

Liu, Lily Y, 2022, Estimating loss given default from cds under weak identification, *Journal of Financial Econometrics* 20, 310–344.

Merton, Robert C, 1974, On the pricing of corporate debt: The risk structure of interest rates, *The Journal of finance* 29, 449–470.

Thekumparampil, Kiran K, Chong Wang, Sewoong Oh, and Li-Jia Li, 2018, Attention-based graph neural network for semi-supervised learning, *arXiv preprint arXiv:1803.03735* .

Uddin, Ajim, 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.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, 2017, Attention is all you need, *Advances in neural information processing systems* 30.

Veličković, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, 2017, Graph attention networks, *arXiv preprint arXiv:1710.10903* .

Wang, Daixin, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi, 2019, A semi-supervised graph attentive network for financial fraud detection, in *2019 IEEE International Conference on Data Mining (ICDM)*, 598–607, IEEE.

Welch, Ivo, and Amit Goyal, 2008, A comprehensive look at the empirical performance of equity premium prediction, *The Review of Financial Studies* 21, 1455–1508.

Yang, ZR, Marjorie B Platt, and Harlan D Platt, 1999, Probabilistic neural networks in bankruptcy prediction, *Journal of Business Research* 44, 67–74.

Zhang, Chao, Xingyue Pu, Mihai Cucuringu, and Xiaowen Dong, 2023, Graph neural networks for forecasting multivariate realized volatility with spillover effects, *arXiv preprint arXiv:2308.01419* .

Zmijewski, Mark E, 1984, Methodological issues related to the estimation of financial distress prediction models, *Journal of Accounting Research* 59–82.

Internet Appendix

A CDS Distribution Across Sectors **52**

B Firm Level Characteristics **53**

C Comparison Algorithms **55**

C.1 CNN 55

C.2 PCR and PLS 56

C.3 GBRT and RF 58

C.4 SVR 59

D Model Complexity and Stability **60**

E Robustness of GNN Under Various Specifications **61**

A CDS Distribution Across Sectors

Table 5. Firm Distribution Across Sectors
This table presents the distribution of firms in our sample across sectors, categorized by the first two digits of their NAICS code. It reports the number of firms and the average log CDS spread for each sector.

Sector Name	First 2 Digit of NAICS	Firm Count	Mean log CDS
Accommodation and Food Services	72	16	-4.79
Administrative and Support and Waste Management and Remediation Services	56	11	-5.09
Construction	23	14	-4.10
Finance and Insurance	52	117	-4.73
Health Care and Social Assistance	62	16	-4.20
Information	51	54	-4.42
Manufacturing	31, 32, 33	268	-4.83
Mining, Quarrying, and Oil and Gas Extraction	21	48	-4.51
Professional, Scientific, and Technical Services	54	26	-4.56
Real Estate and Rental and Leasing	53	33	-4.33
Retail Trade	44, 45	45	-4.58
Transportation and Warehousing	48, 49	31	-4.70
Utilities	22	55	-4.88
Wholesale Trade	42	38	-4.68

B Firm Level Characteristics

Table 6. Firm-Level Characteristics
This table presents 94 firm-level characteristics constructed following Gu, Kelly, and Xiu.

Acronym	Firm characteristic	Acronym	Firm characteristic
absacc	Absolute accruals	*divo*	Dividend omission
acc	Working capital accruals	*dolvol*	Dollar trading volume
aeavol	Abnormal earnings announcement volume	*dy*	Dividend to price
age	# years since first Compustat coverage	*ear*	Earnings announcement return
agr	Asset growth	*egr*	Growth in common shareholder equity
baspread	Bid-ask spread	*ep*	Earnings to price
beta	Beta	*gma*	Gross profitability
betasq	Beta squared	*grcapx*	Growth in capital expenditures
bm	Book-to-market	*grltnoa*	Growth in long term net operating assets
bm_ia	Industry-adjusted book to market	*herf*	Industry sales concentration
cash	Cash holdings	*hire*	Employee growth rate
cashdebt	Cash flow to debt	*idiovol*	Idiosyncratic return volatility
cashpr	Cash productivity	*ill*	Illiquidity
cfp	Cash flow to price ratio	*indmom*	Industry momentum
cfp_ia	Industry-adjusted cash flow to price ratio	*invest*	Capital expenditures and inventory
chatoia	Industry-adjusted change in asset turnover	*lev*	Leverage
chcsho	Change in shares outstanding	*lgr*	Growth in long-term debt
chempia	Industry-adjusted change in employees	*maxret*	Maximum daily return
chinv	Change in inventory	*mom12m*	12-month momentum
chmom	Change in 6-month momentum	*mom1m*	1-month momentum
chpmia	Industry-adjusted change in profit margin	*mom36m*	36-month momentum
chtx	Change in tax expense	*mom6m*	6-month momentum

Table 7. Firm-Level Characteristics (Continued)

cinvest	Corporate investment	*ms*	Financial statement score
convind	Convertible debt indicator	mvel1	log market equity
currat	Current ratio	*mve_ia*	Industry-adjusted size
depr	Depreciation / PP&E	*nincr*	Number of earnings increases
divi	Dividend initiation	*operprof*	Operating profitability
orgcap	Organizational capital	*roeq*	Return on equity
pchcapx_ia	Industry adjusted % change in capital expenditures	*roic*	Return on invested capital
pchcurrat	% change in current ratio	*rsup*	Revenue surprise
pchdepr	% change in depreciation	*salecash*	Sales to cash
pchgm_pchsale	% change in gross margin - % change in sales	*saleinv*	Sales to inventory
pchquick	% change in quick ratio	*salerec*	Sales to receivables
pch-sale_pchinvt	% change in sales - % change in inventory	*secured*	Secured debt
pch-sale_pchrect	% change in sales - % change in A/R	*securedind*	Secured debt indicator
pch-sale_pchxsga	% change in sales - % change in SG&A	*sgr*	Sales growth
pchsaleinv	% change sales-to-inventory	*sin*	Sin stocks
pctacc	Percent accruals	*SP*	Sales to price
pricedelay	Price delay	*std_dolvol*	Volatility of liquidity (dollar trading volume)
ps	Financial statements score	*std_turn*	Volatility of liquidity (share turnover)
quick	Quick ratio	*stdacc*	Accrual volatility
rd	R&D increase	*stdcf*	Cash flow volatility
rd_mve	R&D to market capitalization	*tang*	Debt capacity/firm tangibility
rd_sale	R&D to sales	*tb*	Tax income to book income
realestate	Real estate holdings	*turn*	Share turnover
retvol	Return volatility	*roavol*	Earnings volatility
roaq	Return on assets	*zerotrade*	Zero trading days

C Comparison Algorithms

To provide a comparison, we also employ several nonlinear machine learning algorithms to predict CDS spreads, including convolutional neural network, principal component regression, partial least squares, random forest, gradient boosting regression tree and support vector regression. Consistent with GNN models, we employ the same firm-level characteristics as inputs, and the log CDS spread as the target variable. The division of the data into training, validation, and test samples is split in the same manner as in the GNN models. All hyperparameters are fine-tuned through the validation, and out-of-sample predictions are generated for the subsequent month. In this section, we discuss about the detailed algorithms of each of these machine learning models, which serve as the competing benchmarks.

C.1 CNN

We begin by illustrating the algorithm of the Convolutional Neural Network (CNN), which serves as an ideal benchmark for the Graph Neural Network (GNN). The key distinction between these two models lies in their input data: while the CNN incorporates only node characteristics, the GNN includes both node characteristics and a set of adjacency matrices. Thus, the CNN serves as a control group in our analysis, with the GNN representing a treatment group with additional edge information. In the field of machine learning, CNN is known for its powerful predictive capabilities, attributed to its complexity and nonlinearity (LeCun, Bengio, and Hinton (2015)). For instance, Gu, Kelly, and Xiu (2020) demonstrated that CNNs can predict the cross-section of stock returns more effectively than other machine learning techniques. Our CNN algorithm, designed to predict firm CDS spreads, mirrors the CNN structure in their study. Unlike the GNN architecture, the CNN architecture solely contains the updating scheme (or 'intra-layer design') and omits the inter-layer design, as there is no need to embed edge information. The intra-layer design closely resembles that of the GNN.

$$h_v^{(k+1)} = AGG\left\{ACT\left(DROPOUT\left(BN\left(\omega^{(k)}h_v^{(k)} + b^{(k)}\right)\right)\right), v = 1, 2, ..., N\right\} \tag{21}$$

The equation (21) presented in this subsection defines the update rule for the node characteristics in a Convolutional Neural Network (CNN) with one hidden layer. The node characteristics at layer

$k+1$, denoted by $h_v^{(k+1)}$, are computed as a function of the characteristics of the nodes in layer k. Specifically, the characteristics of the neighbors are aggregated using a simple linear aggregation function, and the resulting vector is passed through a sequence of modules: batch normalization $BN(\cdot)$, dropout $DROPOUT(\cdot)$, nonlinear activation function $ACT(\cdot)$, and a trainable weight matrix $\omega^{(k)}$ followed by a bias term $b^{(k)}$. The node characteristics at layer k, denoted by $h_v^{(k)}$, represent the characteristics of node v at layer k. It is not the node embedding $\hat{h}_v^{(k)}$ any more as in the GNN.

To ensure a robust comparison with the GNN, we maintain the same nonlinear activation, dropout, and batch normalization rules. The rectified linear unit (ReLU) is used as the nonlinear activation function across all nodes. Dropout is implemented to prevent overfitting, and batch normalization is employed to stabilize and expedite the training process. Therefore, the updating scheme in the CNN is identical to that in the GNN. Furthermore, we consider CNN architectures with up to four hidden layers, denoted as CNN1, CNN2, CNN3, and CNN4. The number of neurons in each layer is selected to match those in the corresponding GNN models.

Finally, the training configuration for the CNN mirrors that of the GNN. All hyperparameters, including the learning rate in SGD, are fine-tuned using the validation set, and early stopping is incorporated to prevent overfitting. In essence, all training rules for the CNN are aligned with those of the GNN, allowing for a precise comparison between the two models.

C.2 PCR and PLS

We employ Principal Components Regression (PCR) as a classic dimension reduction method in our analysis. Dimension reduction, unlike predictor selection, averages all predictors, which can help reduce noise and better isolate the signal within them. This technique is particularly beneficial when the predictors are highly correlated. PCR consists of two main steps: PCA and regression. Initially, PCA is conducted to construct a few principal components (say p components) that preserve the covariance structure of all regressors. Then these principal components are used in a standard regression model. [21] The PCR model is represented as:

$$\mathbf{CDS} = Z\beta + E$$

[21] PCA's role is to condense a large number of predictors into a few significant components before examining their relationship with the target variable.

where **CDS** is the $NT \times 1$ vector of the CDS spread, Z is the $NT \times k$ matrix of stacked predictors, and E is the vector of residuals. PCR seeks the p principal components Ω so that

$$\mathbf{CDS} = (Z\Omega_p)\beta_p + \hat{E}$$

Each column of Ω_p, ω_j is a linear combination of the original predictors. PCR solves the following optimization function:

$$\omega_j = \arg \max_{\omega} \text{Var}(Z\omega)$$

s.t. $\omega'\omega = 1$ and $\text{Cov}(Z\omega, Z\omega_k) = 0$ fpr all $k = 1, 2, \cdots, k-1$.

The number of principal components is determined through the validation set. PCR seeks linear combinations of the full set of predictors that best mimic the full predictors, without directly incorporating the forecasting objective into the initial dimension reduction step. This approach sometimes leads to criticism that PCR focuses more on finding principal components than on the final prediction target.

Partial Least Squares (PLS) is another dimension reduction method that considers the correlation of initial predictors with the target during dimension reduction. For each predictor i, PLS estimates its coefficient ψ_i through a univariate OLS regression. PLS then averages all predictors into a single aggregate component, weighting them proportionally to ψ_i, thus giving more weight to stronger univariate predictors. PLS effectively averages the "partial" sensitivity of the target variable to each predictor. Multiple components can also be formed, with the target and all predictors orthogonalized with respect to previously constructed components, and the procedure repeated on the orthogonalized dataset until the desired number of PLS components is reached.

Similar to PCR, PLS also seeks the p principal components Ω so that

$$\mathbf{CDS} = (Z\Omega_p)\beta_p + \hat{E}$$

and this number is tuned from the validation set.

However, its optimization goal is different:

$$\omega_j = \arg \max_{\omega} \text{Cov}^2(\mathbf{CDS}, Z\omega)$$

s.t. $\omega'\omega = 1$ and $\text{Cov}(Z\omega, Z\omega_k) = 0$ for all $k = 1, 2, \cdots, k-1$.

PLS, therefore, trades off some accuracy in principal component identification for improved prediction of the initial target.

C.3 GBRT and RF

'Ensemble' tree models are also popular in machine learning for their ability to handle nonlinearity and variable interactions. Classic tree-based models are prone to overfitting and thus require substantial regularization. Following the approach in Gu, Kelly, and Xiu (2020), we employ two regularized tree-based methods as benchmarks: Gradient Boost Regression Trees (GBRT) and Random Forest (RF).

GBRT is a method that iteratively combines predictions from multiple simple, or 'weak' trees to form a more robust, 'strong' learner. It begins by fitting a shallow tree, typically with limited depth (e.g., depth $L = 1$), which, due to its simplicity, is likely to be a weak predictor with large bias. Then a second simple tree of the same depth is used to fit the residuals from the first tree. The predictions from these two trees are combined, with the contribution from the second tree scaled down by a factor $\nu \in (0, 1)$ to mitigate overfitting. This process is repeated, with each new tree fitting the residuals from the previous ensemble, until the ensemble comprises B trees. The final model is an additive combination of these shallow trees, with the tuning parameters (depth L, shrinkage factor ν, and number of trees B) determined during the validation phase. The optimization at each tree branch aims to minimize the L_2 loss function.

Random Forest, like GBRT, is another ensemble method that aggregates predictions from multiple trees. It is a variation of the bootstrap aggregation, or 'bagging' (see Breiman (2001)). B different bootstrap samples of the data are drawn, a separate regression tree is fitted to each sample, and their forecasts are averaged. While trees fitted to individual bootstrap samples tend to be deep and overfit, averaging across multiple predictions reduces this variability. RF introduces an additional layer of regularization by considering only a randomly selected subset of predictors for splitting at each branch, a technique akin to dropout, to further prevent overfitting. The depth L of the trees, the number of predictors considered at each split, and the number of bootstrap samples B are all key parameters, tuned through validation.

C.4 SVR

Support Vector Regression (SVR) is a machine learning method that originates from the broader concept of Support Vector Machines (SVMs). SVR is good at modeling relationships between input features and target variables. It does so by trying to optimize a hyperplane that accurately captures the underlying data patterns, while also minimizing prediction errors. SVR is grounded in a margin maximization strategy, rendering it particularly effective for both linear and non-linear regression challenges. SVR aims to identify a function $f(x)$ that predicts continuous target values from input data x. The objective is to establish a function $f(x)$ that maintains a maximal margin between the predicted values and a specified margin threshold.

Mathematically, SVR is formulated as an optimization problem. Given a training dataset $\{(x_i, y_i)\}_{i=1}^{N}$, where x_i is the input feature vector of dimension d, and y_i is the corresponding target value (i.e. log CDS spread), SVR seeks to solve the following optimization problem:

$$
\begin{aligned}
\text{Minimize:} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right) \\
\text{Subject to:} \quad & y_i - \left(w^T\phi(x_i) + b\right) \leq \epsilon + \xi_i \\
& \left(w^T\phi(x_i) + b\right) - y_i \leq \epsilon + \xi_i^* \\
& \xi_i, \xi_i^* \geq 0
\end{aligned}
$$

Here, w is a weight vector, and b is a bias term that defines the hyperplane in the feature space. $\phi(x_i)$ represents the mapping of input data x_i into a higher-dimensional feature space. This mapping allows SVR to capture non-linear relationships between input features and target values. ϵ is a margin threshold, and ξ_i and ξ_i^* are slack variables allowing for some deviation from the margin threshold. The regularization parameter C balances maximizing the margin and minimizing the errors. [22]

The optimization problem aims to minimize the norm of the weight vector $\frac{1}{2}\|w\|^2$ while ensuring that the errors (ξ_i and ξ_i^*) are within the margin threshold ϵ. The solution yields the weight vector w and bias term b, and once trained, the SVR model can make predictions for new

[22] A smaller C encourages a wider margin but allows for more errors, while a larger C enforces stricter error penalties.

data points by evaluating $w^T \phi(x) + b$. The choice of the kernel function $\phi(x)$ plays a crucial role in SVR. We employ a polynomial kernel, with its parameters optimized through the validation set.

SVR's strength lies in its capacity to identify a margin containing most training data points, while tolerating some points to reside outside the margin within the threshold ϵ. This attribute makes SVR a robust regression method, particularly effective in scenarios involving noisy or non-linear data.

D Model Complexity and Stability

For both GNN and GNN-attention models, we use the Stochastic Gradient Descent (SGD) optimizer. To mitigate overfitting, we implement early stopping. This mechanism involves continuous monitoring of the model's performance on a separate validation dataset throughout the training process. Should there be any indication of deteriorating performance on this validation set, such as an increase in validation mean squared error, the training process is preemptively halted. This strategy is crucial for preventing the model from excessively adapting to the training data at the expense of its predictive performance on unseen data.

Additionally, we fine-tune all hyperparameters (including dropout rate, learning rate, weight decay in SGD, patience, number of hidden layers, hidden layer size of GCN and LSTM, neuron kernel initialization methods like Glorot or Kaiming, and the number of attention heads) based on the mean squared error observed on the validation set. We use simple unidirectional algorithm for LSTM and limit the number of hidden layers to either 1 or 2 for both GCN and LSTM to avoid model misspecification. The optimal configuration consists of 1 hidden layer for GCN with 12 neurons, and 2 hidden layers for LSTM with 12 and 6 neurons, respectively. For the GNN-attention model, we implement a dual-head attention mechanism and use the mean of their outputs as the final attention scores.

To calculate the number of parameters, we break down the model into its components and compute the parameters for each part. The architecture includes a GCN layer with 12 neurons, followed by a LSTM network with two layers comprising 12 and 6 neurons, respectively. The model also incorporates an attention mechanism with two attention heads. The input to the GCN layer consists of 112 variables.

First, the GCN layer applies a weight matrix to the input features, resulting in $112 \times 12 = 1,344$ parameters for the weights, plus 12 parameters for the biases, leading to a total of 1,356 pa-

rameters for the GCN layer. Next, the LSTM network requires four weight matrices for each gate (input, forget, cell, output), with dimensions based on the input size and the number of hidden units. For the first LSTM layer with 12 neurons, the number of parameters is calculated as $4 \times ((12 \times 12) + (12 \times 12) + 12) = 4 \times 300 = 1,200$ parameters. The second LSTM layer with 6 neurons requires $4 \times ((12 \times 6) + (6 \times 6) + 6) = 4 \times 114 = 456$ parameters. Finally, the attention mechanism introduces its parameters. Assuming each attention head operates on the 6-dimensional output from the last LSTM layer, and each head produces a scalar output, each attention head contributes $6 \times 1 = 6$ parameters, leading to $6 \times 2 = 12$ parameters for the two heads. Consequently, the total number of parameters in the TGNN is $1,356$ (GCN) + $1,200$ (LSTM Layer 1) + 456 (LSTM Layer 2) + 12 (Attention Heads) $= 3,024$ parameters. The total number of observation is $72 \times 678 = 48816$.

The ratio of the number of observations to the number of parameters is around 15. This ratio is generally considered a good ratio in deep learning contexts as it suggests that the model has a reasonable amount of data relative to its complexity, which can help in achieving good generalization and avoiding overfitting.

E Robustness of GNN Under Various Specifications

For robustness checks, we present out-of-sample prediction results with modified input specifications for the GNN algorithm.

In our baseline results, we construct inter-firm edge characteristics by computing idiosyncratic volatility spillover among sectors and extrapolating it to the firm level, using NAICS two-digit codes for sector classification. We now consider two alternatives: (1) Fama-French 48 sectors; and (2) Fama-French 12 sectors. The results are reported in the first two rows of Table 8.

Additionally, we explore different specifications for calculating idiosyncratic volatility. In the baseline, idiosyncratic volatility is computed using the monthly standard deviation of daily idiosyncratic returns, where idiosyncratic return is the residual after removing Fama-French 3 factors. We consider three alternatives: (1) Removing only the CAPM factor; (2) Removing PC 5 factors; and (3) Not removing any common factors. The results are reported in row 3-5 of Table 8.

Table 8. Pooled Out-of-Sample RMSE Under Various Input Specifications
This table presents the pooled out-of-sample root mean square error (RMSE) for modified input specifications for the GNN algorithm. Column 1 shows RMSE for the entire sample from February 2005 to December 2020. Columns 2 and 3 report RMSE for investment-grade (BBB and above) and high-yield (BB and below) firms, respectively. Columns 4 and 5 display RMSE for small (below median market capitalization) and large (at or above median market capitalization) firms, respectively.

	All	Investment Grade	High Yield	Small	Big
Fama-French 48 sectors	0.850	0.775	0.977	0.822	0.816
Fama-French 12 sectors	0.820	0.726	0.974	0.800	0.770
CAPM factor	0.802	0.709	0.877	0.878	0.865
PC 5 factors	0.701	0.704	0.697	0.629	0.637
no factor	0.902	0.714	1.079	0.888	0.733

Overall, these robustness checks confirm the reliability and effectiveness of the edge characteristics design in our GNN algorithms, demonstrating their robustness across various specifications.