Intelligent Systems with Applications 19 (2023) 200240

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Predicting systemic risk in financial systems using Deep Graph Learning ☆

Vicente Balmaseda [a], María Coronado [b,*], Gonzalo de Cadenas-Santiago [c]

[a] Universidad Pontificia Comillas, ICAI, School of Engineering and ICADE, Faculty of Economics and Business Administration, Alberto Aguilera 23, 28015 Madrid, Spain
[b] Universidad Pontificia Comillas, ICADE, Faculty of Economics and Business, Alberto Aguilera 23, 28015 Madrid, Spain
[c] MAPFRE Economics, Economic Research Department, Carretera de Pozuelo 52, ed.1. 28222 Majadahonda, Madrid, Spain

ARTICLE INFO

Keywords:
Graph neural networks (GNN)
Financial networks modeling
Model selection
Neural networks
Label regression
Network simulation

ABSTRACT

Systemic risk is the risk of infection from an individual financial entity to the financial system due to existing interconnections. Having powerful tools to analyze and predict systemic risk in large financial networks is essential to ensure the stability of the financial system, avoiding the negative externalities derived from the failure of a systemically important financial institution. In this context, Machine Learning (ML) has proved to be a useful tool thanks to its ability to deal with complex relations. However, traditional techniques are limited in their use of the interactions between entities and the network structure, which has been shown to be of great importance for systemic risk. Thus, this work proposes Graph Neural Networks (GNNs) for systemic risk analysis. GNNs use the network structure and feature information to deal with large-scale financial networks, providing the benefits of ML while using all the available information (node inter-relations and node, edge, and graph features). We also present C2R, an approach to reduce the pre-labeling effort for costly systemic risk metrics by pre-labeling into a small number of classes while predicting continuous risk scores. We have tested GNNs against traditional ML in classifying entities by systemic risk importance in two different networks, comparing their generalization capabilities with different amounts of available data. GNNs achieve a 94% and 15% Matthew's Correlation Coefficient (MCC) average percentage increase compared to ML, achieving statistically significant MCC improvements in most scenarios. When combining C2R with GNNs to predict the systemic risk quantile from the class labels, the models achieve statistically significant improvements in the quantile RMSE. From our experiments, we can confirm that GNN models are better suited for systemic risk prediction on financial networks and should be preferred over traditional Machine Learning. The results obtained also confirm the fact that the network structure and the features of the relations (edge features) hold useful information for our task.

1. Introduction

The importance of systemic risk in the financial sector has increased once again with the Covid-19 crisis and the Ukrainian war. Systemic risk is the risk of infection, due to losses or defaults, from an individual financial entity to the financial system due to existing interconnections. The recent failures of Silicon Valley Bank (SVB) in the USA and Credit Suisse in Europe have brought back the threat of a contagion effect into the whole banking sector, since the aftermath of the Global Financial Crisis of 2007-2009. A robust financial network plays a key role in the stability of the financial system. Through it, entities can borrow from each other, having a stabilizing effect by effectively redistributing funds. However, at the same time, it can make the system prone to financial contagion through the existing linkages. This gives the analysis of financial networks a special interest. In this context, systemic risk has become an area of increased relevance and attention.

Supervisory authorities or individual institutions can analyze the financial network to gather insights concerning contagion risk from the channels through which shocks propagate. Moreover, the resilience of a network can be evaluated, and systemically significant nodes can be identified. Regulators have imposed specific capital requirements on these systemically important financial Institutions (G-SIFIs), which they identify as G-SIBs (Global Systemically Important Banks) and G-SIIs (Global Systemically Important Insurers). Network analysis also

☆ The code is available on https://github.com/vibalcam/gnn-systemic-risk.

* Corresponding author.
E-mail address: mcoronado@comillas.edu (M. Coronado).

https://doi.org/10.1016/j.iswa.2023.200240
Received 26 February 2023; Received in revised form 8 May 2023; Accepted 27 May 2023
Available online 5 June 2023

provides an empirical tool for evaluating the effectiveness of macroprudential policies whose objective is to limit the impact of financial shocks on the real economy.

The importance of the linkages among entities has led financial networks to be modeled as graphs where nodes are financial entities, regions, or countries; and edges are the connections between them. In doing so, we can study the impact that a shock on one or more entities produces on the entire network, obtain systemic risk metrics, and study how different entities (banks, insurers, mutual funds, central banks, and so on) complement each other. This structure suggests the use of Graph Theory, applied in other fields, to the analysis of systemic risk.

Systemic risk is always potentially present in modern large-scale financial systems. The increasing globalization of financial markets is extending these linkages internationally, creating a substantial web of interconnections across a large number of heterogeneous entities, and making the analysis of these networks increasingly difficult.

Moreover, systemic risk is a complex nonlinear phenomenon, which originates from multiple contagion channels and interactions between entities. In these large-scale scenarios, traditional network models struggle to capture complex dynamics and nonlinear interactions between entities. Moreover, these tools cannot process all the available information, being unable to use network information such as node or edge features. Machine Learning techniques can be used to deal with large and complex networks, identifying patterns to analyze and predict systemic risk in these large networks, thus bringing new insights into the current knowledge.

The traditional Machine Learning techniques currently used have achieved success in this field. However, these techniques consider each observation independently, not using the information provided by the interactions between entities, which has been shown to be of great importance for systemic risk. Moreover, these techniques require using metrics such as centrality to include network structure information, thus forcing explicit feature engineering and limiting the network information used. Therefore, there is a trade-off between using Network models, taking advantage of the graph structure, and using Machine Learning techniques, taking advantage of feature information, and dealing with large networks.

Graph Neural Networks (GNNs) are Deep Learning models that bring the benefits of Neural Networks to the graph realm. GNNs have been designed to work with graph-structured data, which allows them to model a nonlinear systemic risk metric from the graph representing the financial network, modeling the complex interactions between entities. Therefore, using GNNs avoids the trade-off between Machine Learning and network structure, dealing with large-scale networks while making use of all the available information, both the network structure and node, edge, and graph features. In addition, GNNs are flexible models that can be applied to the same supervised and unsupervised tasks as traditional Machine Learning and can serve as a meeting point between Machine Learning and network techniques for systemic risk. Thus, the first aim of this paper is to compare the effectiveness of traditional Machine Learning and GNNs for systemic risk prediction on financial networks.

We test the effectiveness of GNNs compared to traditional Machine Learning in classifying the entities of a network by systemic risk importance. We have compared them in two differently distributed large networks[1] using different amounts of available data to compare their generalization capabilities. Concretely, we have applied 3 different GNNs: Graph Convolutional Networks (GCN) (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), and Graph Attention Networks (GAT) (Veličković et al., 2018). We have compared their effectiveness against traditional Machine Learning techniques: Logistic Regression (LR), Random Forest Classifier (RF), KNN Classifier (KNN), and a Deep Learning Model, Fully Connected Neural Network (FCNN). More specifically, the task consists in classifying the nodes of a large financial network (1500 nodes) into 4 systemic importance classes. The classes have been defined according to a systemic risk metric based on a contagion simulation. For each network, three scenarios have been presented, in which the percentage of pre-labeled nodes available for training varies from 75% to just 10% (Table 3).

A second objective of this paper is motivated by the following fact: The systemic risk literature mostly focuses on supervised Machine Learning, which requires pre-labeled data to train the models. Recent developments in Machine Learning, especially Deep Learning, allow dealing with large networks modeling complex systemic risk metrics. However, these methods often perform best with higher quantities of data. For some continuous metrics, such as expert evaluations, the increase in pre-labeled data leads to higher costs and complexity, potentially rendering the process unfeasible. Using discrete versions of these metrics reduces the pre-labeling effort at the expense of less precise predictions. Thus, we propose an approach to taking advantage of the simple discrete pre-labeling while training a model that predicts a continuous risk score, thus reducing the loss of precision. Moreover, it can also be used with discrete metrics for quantile prediction, thus increasing the granularity and precision of the predictions.

We test the effectiveness of the approach presented by training a GNN from data labeled into a small number of classes (discrete systemic risk metric) to predict a continuous systemic risk score, each node's quantile. More specifically, the input will still consist of a set of pre-labeled nodes into 4 classes (same as for classification), but the model will predict each node's systemic importance quantile instead of its class. This approach presents a way of using a simpler pre-labeling (using 4 ordered classes) to obtain more granular and precise predictions, thus enabling the use of more complex and costly systemic risk metrics. It can be generalized to be used to obtain models that have a discrete input (classes) and a continuous regression output.

Our study has important implications for both academics and practitioners (whether they are policymakers or financial institutions) as we propose analytical tools and new approaches to analyze the behavior of the financial system at large as a complex system, that can be used by central banks, other regulatory authorities and financial institutions to better assess and monitor systemic risk.

Contributions. This work contributes to the literature on applying Graph Neural Networks, and to the literature on financial systemic risk and financial network modeling in two ways. First, while previous works focused either on the network structure, using network theory, or on feature information, using traditional Machine Learning, we discuss and show experimentally the benefits of using Graph Neural Networks (GNNs) to better model financial networks for systemic risk analysis and other tasks. Additionally, we discuss how GNNs can act as a meeting point between Machine Learning and network literature by benefiting from network estimation techniques. Second, to improve the usefulness of Machine Learning to model complex and costly systemic risk metrics, we present an approach to reduce the pre-labeling effort by pre-labeling into a small number of classes while predicting continuous risk scores. To the best of our knowledge, this is the first work that combines the network structure and feature information in a single model for systemic risk prediction, through the use of Graph Neural Networks.

Organization of the paper. The remainder of this paper is structured as follows. Sec. 2 reviews prior research done on the measurement and analysis of systemic risk in the financial sector. Sec. 3 gives a quick overview on Graph Neural Networks. Sec. 4 providing a general flowchart and methodology proposed. Sec. 5 presents the motivation and approach of Class to Regression (C2R). Two case studies, one with synthetic data and another one using real data from 1444 European banks, are presented in Sec. 6. The results and their discussion can be found in Sec. 7. Conclusions and future work are presented at the end of the paper in Sec. 8.

[1] Two financial networks have been used, one generated from historical aggregated data from European banks, and another one generated from aggregated synthetic data.

2. Literature review

In this section, we will conduct a very synthetic review of the literature on the measurement and analysis of systemic risk in the financial sector, with two aims: i) to show the research gap in the literature to which our work responds, and also ii) to mention the already existing techniques in this domain that we are also using in our proposed methodology or even more, that could be used together with our proposed methodology. In addition, we emphasize research from 2019 to 2023 by summarizing it in Table 1.

"Too big to fail." A key aspect of financial stability is large financial institutions. These institutions have been among the various suspects for destabilizing the financial system since their failure would be disastrous to the greater economic system. These institutions are the focus of the "too big to fail" theory. Such financial institutions must be supported by the government when they face financial distress due to their systemic importance and interconnectedness. However, the failure of smaller financial institutions with many connections in the financial system can also be a source of systemic risk. A highly interconnected financial network can absorb shocks through its links and be robust, thus reducing the probability of default since the shock can be shared by many counterparties. However, beyond a given magnitude, the same links may have the opposite effect. The shock will spread into a large part of the system, which can cause a large cascade of defaults (Acemoglu et al., 2015). Thus, the links will function as a propagator mechanism for those shocks leading to financial contagion and systemic risk. This concept is known as robust-yet-fragile, a risk absorber in most cases, but equally a risk spreader where fragility prevails (Haldane, 2009).

Insurers. Most of the research done about systemic risk focuses on banks. However, insurers are also part of the financial system and, as such, are agents that can generate or absorb systemic risk. This has led regulators, until 2019, to include certain insurers as G-SIIs (Global Systemically Important Insurers). Insurers are different from banks in many ways. For example, insurers do not have as their primary method of funding a deposit scheme, tend to follow a more conservative investment policy statement than banks, rely more on internal credit analysis, and are regulated differently (Rudolph, 2017). These differences may cause insurers to have a different effect on systemic risk than banks.

"Traditional" (econometric and statistical) Systemic Risk metrics. The importance of systemic risk in the Global Financial Crisis led many researchers to continue working on metrics to measure it. The two most popular metrics are CoVaR-ΔCoVaR and SRISK. ΔCoVaR represents the Value at Risk (VaR) of an institution (or a set of financial institutions, i.e., the financial system) conditional on an institution being under distress relative to its median state (Adrian & Brunnermeier, 2016). SRISK is defined as the expected capital shortfall of a financial entity conditional on a prolonged market decline. SRISK is a function of the firm's size, its degree of leverage, and its expected equity loss conditional on the market decline, which they call Long Run Marginal Expected Shortfall (LRMES). SRISK is used to construct rankings of systemically risky institutions. The higher the SRISK, the higher the firm's contribution to the undercapitalization of the financial system in times of distress. The sum of SRISK across all firms is used as a measure of overall systemic risk in the entire financial system. It can be thought of as the total amount of capital that the government would have to provide to bail out the financial system in case of a crisis (Brownlees & Engle, 2017). Other metrics also used to measure systemic risk are SES (propensity to be undercapitalized when the system is undercapitalized) (Acharya et al., 2017), CoCVaR (the conditional Value at Risk, CVaR, of the financial system conditional on an institution being in financial distress) (Huang & Uryasev, 2018) and CoCDaR-mCoCDaR (conditional drawdown-at-risk conditioned on an institution) (Ding & Uryasev, 2020). Other work in this field combines multiple metrics to measure systemic risk, such as Giglio et al. (Giglio et al., 2016) which combines 19 different metrics.

Network models. Deeper insights than the ones obtained by "traditional" (econometric and statistical) systemic risk metrics can be obtained by modeling the financial system as a network of interconnected financial institutions (Hautsch et al., 2014; Giudici and Spelta, 2016; Giudici et al., 2020; Cerchiello and Giudici, 2016; Härdle et al., 2016). Financial systems can be modeled by networks in several ways, such as multiplex networks, single-layer networks, and correlation and similarity-based networks. By using graphs to model the systemic risk, we can apply metrics from Mathematics and Physics, where networks have a long history of research (Bardoscia et al., 2021). An example of this application can be found by estimating the bilateral exposures matrix using aggregate financial data on loans and deposits from Bankscope. That data can then be used to model the financial system as a hierarchical network where we can analyze interconnectedness using network centrality measures. The result of this analysis confirms that most of the banks designated as G-SIBs play a central role in the global interbank market (Kanno, 2015). Moreover, networks can be used to compare the fragility of several network topologies. An example of this is Leventides et al. (2019), who made use of Monte Carlo simulations and a simple default model of contagion applied on interbank networks of varying sizes. They then triggered a series of banking crises by exogenously failing each bank in the system and observed the propagation mechanisms that take effect within the system under different scenarios. One of the main findings of this work is the importance of heterogeneity in the stability of the financial system. When heterogeneity is introduced in the size of the banks, the system's shock absorption capacity is enhanced. Other research focuses on comparing network metrics of systemic risk with "traditional" metrics. Studies comparing the difference in informative content between quantile-based network measures and systemic risk indicators, such as the loss measure ΔCoVaR, showed that these metrics are not highly correlated (Billio et al., 2017). Moreover, centrality measures can explain the maximum financial loss percentage more than the loss measures based on ΔCoVaR. These results confirm the importance of using networks to measure systemic risk since they capture the "indirect" effects of risk spillovers better (Billio et al., 2017). Keilbar and Wang (2022) observed that "traditional" systemic risk measures, specifically CoVaR in its original approach (Adrian & Brunnermeier, 2016), are restricted to analyzing systemic risk in a linear and bivariate context. Thus, they provide a new perspective for estimating CoVaR, using a neural network quantile regression and thus allowing to capture of possible nonlinear dependencies since nonlinear effects are crucial for modeling systemic risk. In addition, they propose three network-based measures for systemic risk: the Systemic Network Risk Index (SNRI) as a measure for total systemic risk, and two firm-specific measures, the Systemic Fragility Index (SFI), which identifies the most vulnerable banks in a given financial network, and the Systemic Hazard Index, which identifies the financial institutions which potentially pose the largest risk to the financial system.

DebtRank. In addition to using "traditional" metrics and network metrics to measure systemic risk, financial networks also have specific metrics. An example of this is DebtRank, also known as linear propagation. DebtRank is a financial-oriented centrality measure that is able to capture the banks' distress levels even at mild macroeconomic conditions or low levels of distress (Bardoscia et al., 2015, Battiston et al., 2012). DebtRank is sensitive to small shocks because it considers that financial assets deteriorate as a function of the net worth of the borrower. Then, as the borrower becomes financially distressed, the corresponding creditors immediately recognize losses on the financial assets they have against that borrower. In this way, it can supply information on how far banks are from insolvency. Compared with loss-based metrics, it is not a binary metric (either one bank can honor or not their liabilities in full), and it can also estimate potential losses in a financial system using the concept of financial stress (capacity of banks to absorb losses rather than their payment ability). DebtRank is the contagion algorithm we use in our case studies (see Sec. 6.1). Niu et al. (2021) also use the other most popular contagion algorithm, Threshold propaga-

Table 1
Brief review of research from 2019 to 2023 on the measurement and analysis of systemic risk in the financial sector.

Author	Approach used	Main findings	Does it apply GNN?
Giudici et al. (2020)	Network model	Trying to address the interconnected nature of financial systems, these authors compare classical networks (or direct exposure networks, also known as market-based estimated networks) and correlation-based networks (which leverage the multivariate network structure and indirect or common exposures), in the modeling of interbank market flows between countries. Correlation network models. They find that while linkages based upon common exposures and a combination of direct and common exposures perform equally well in forecasting crisis episodes, the predictive power obtained combining the two type of network is superior especially in periods of financial crisis.	No
Bardoscia et al. (2021)	Network model	They review how statistical physics codeveloped new metrics and models for the study of financial network structure, dynamics, and stability and instability. The authors introduce network representations originating from different financial relationships, including direct interactions such as loans, similarities such as co-ownership and higher-order relations such as contracts involving several parties (for example, credit default swaps) or multilayer connections (possibly extending to the real economy). They then review models of financial contagion capturing the diffusion and impact of shocks across each of these systems. They also discuss different notions of 'equilibrium' in economics and statistical physics, and how they lead to maximum entropy ensembles of graphs, providing tools for financial network inference and the identification of early-warning signals of system-wide instabilities	No
Leventides et al. (2019)	Network model	They use networks to compare the fragility of several network topologies. By running Monte Carlo simulations and a simple default model of contagion applied on interbank networks of varying sizes. They then triggered a series of banking crises by exogenously failing each bank in the system and observed the propagation mechanisms that take effect within the system under different scenarios. One of the main findings of this work is the importance of heterogeneity in the stability of the financial system. When heterogeneity is introduced in the size of the banks, the system's shock absorption capacity is enhanced.	No
Keilbar and Wang (2022)	Neural Network quantile regression (Network Model +ML)	The authors observed that "traditional" systemic risk measures, specifically CoVaR in its original approach (Adrian and Brunnermeir, 2016), are restricted to analyzing systemic risk in a linear and bivariate context. Thus, they provide a new perspective for estimating CoVaR, using a neural network quantile regression and thus allowing to capture of possible nonlinear dependencies since nonlinear effects are crucial for modeling systemic risk. They find that neural networks outperform the linear baselines. In addition, they propose three network-based measures for systemic risk: the Systemic Network Risk Index (SNRI) as a measure for total systemic risk, and two firm-specific measures, the Systemic Fragility Index (SFI), which identifies the most vulnerable banks in a given financial network, and the Systemic Hazard Index, which identifies the financial institutions which potentially pose the largest risk to the financial system	No
Niu et al. (2021)	Network Model	They propose a practical visual analytical approach to support the exploration and evaluation of regulatory intervention measurements to mitigate systemic risk in financial networks, such as removing nodes (entities) of interest, partitioning the network, replacing the node with specified one, and many others. They use the two most popular contagion algorithms, Threshold propagation (also known as Default Cascades), and Linear Propagation (DebtRank)and a combination of them. When a regulatory intervention measure is taken, the authors carry out the contagion process with the new network to analyze the effects of the measure taken	No
Kou et al. (2019)	ML	The authors provide an overview of different Machine Learning techniques for systemic risk analysis in financial sectors, such as Support Vector Machines, Gradient Boosting Decision Trees, Neural Networks and many others	No
Yu and Zhao (2020)	ML	They use Gradient Boosting Decision Trees to examine the influence of network structures and different network features to forecast financial crises. The authors represent the financial system using a directed graph and used Gradient Boosting Decision Trees instead of graph properties to estimate the importance of each variable. They conclude that, for any attack strategy adopted to trigger the initial failures, a capital reserve ratio higher than 20% would control the crisis. On the other hand, the crisis would be nearly inevitable when the capital reserves ratio is less than 5%.	No
Cheng et al. (2020)	GNN	They propose the application of GNNs for detecting and predicting risk guarantees in networked-loans. In particular, they propose a dynamic graph-based attention neural network (DGANN) where each guarantee is represented as an edge in dynamic loan networks, while companies are denoted as nodes. Thus, they do edge classification instead of node classification.	Yes (not for predicting systemic risk but risk guarantees in networked loans)
Li et al. (2021)	GNN	The authors apply GNN for merger and acquisition (MA) prediction. While the traditional methods of predicting MA probability are only based on the company's fundamentals, such as revenue, profit, or news, GNN takes full advantage of those relationship data to expand the feature dimension and improve the prediction result. Their approach achieves a high Area-Under-Curve score (AUC) of 0.952, which is better than the previous record of 0.888. The true positive rate is 83% with a low false positive rate of 7.8%, which performance is better than the previous benchmark record of 70.9%/10.6%	Yes (not for predicting systemic risk but MAs probability)

tion (also known as Default Cascades) and a combination of linear and threshold propagation.

Machine Learning. Systemic risk is always potentially hidden in modern large-scale financial systems. Big data on financial transactions, market sentiments, etc., are sources of great amounts of unexploited information. Machine learning methods allow us to exploit this information to assess systemic risk. An overview of different Machine Learning techniques for systemic risk analysis in financial sectors can be found in Kou et al. (Kou et al., 2019). Thus, the use of Machine Learning and networks together has the potential to bring new insights into the current knowledge. Machine learning has proved to be a useful tool for systemic risk analysis thanks to its ability to analyze complex networks with non-linear relations and its flexibility. Traditional Machine Learning techniques, such as Support Vector Machines, have been used to develop systems able to predict systemic risk from a given set of risk indicators (Li et al., 2013). Others, such as Gradient Boosting Decision Trees, have been used to examine the usefulness of different indicators to predict recessions (Döpke et al., 2017) or the influence of network structures and different network features to forecast financial crisis (Yu & Zhao, 2020). Yu and Zhao (2020) represented the financial system using a directed graph and used Gradient Boosting Decision Trees instead of using graph properties to estimate the importance of each variable. They concluded that, for any attack strategy adopted to trigger the initial failures, a capital reserve ratio higher than 20% would control the crisis. On the other hand, the crisis would be nearly inevitable when the capital reserves ratio is less than 5%. Other research uses, not only numerical data but also text information to model the financial relationships between institutions. Data can be extracted from various text sources, such as social media like Twitter (Cerchiello et al., 2017), news like the Reuters online news archive (Rönnqvist & Sarlin, 2015), and online discussion forums like Kauppalehti (Rönnqvist & Sarlin, 2014). Neural Networks have also been used for systemic risk prediction. Keilbar and Wang (2022) used Neural Networks to calibrate the conditional Value-at-Risk (CoVaR), which outperformed linear baselines the reason being the importance of nonlinearities on risk measures due to the complex relations between financial institutions.

Graph Neural Networks (GNN). However, these Machine Learning techniques have a major flaw: they do not take into account the network structure or they do so, but not in a full manner (as in the case of neural networks). Therefore, they cannot be applied directly to financial networks. This can be alleviated by using Graph Neural Networks, as they are a class of Deep Learning algorithms designed to deal with data presented in graphs, allowing us to perform node-level, edge-level, or graph-level tasks. Hence, using GNNs seems to be the natural step. Thus, we propose to go this step further and use GNNs for systemic risk analysis to take advantage of network information in a more granular way, not just the aggregate effect on a node. Hence, our study contributes to the contemporary literature on analyzing systemic risk by filling this gap. Concretely, we use GNNs for classification into systemic risk classes, but ultimately they could also be used to predict other systemic risk measures such as the ones reviewed in this section. For instance, we could train a GNN in a similar way as Keilbar and Wang (2022) do for a neural network. This model could then be used to estimate the CoVaR and use it as a risk measure in the tool proposed by said paper. Even more, it would be interesting as future work to replicate their results and test whether using GNNs improves them. We are willing to explore this as a further line of research, as stated in section 8.2. GNNs have been applied in various domains and tasks (other than financial ones) such as computer vision, Natural Language Processing (NLP), traffic forecasting, recommender systems, physics, chemistry, biology, and many others – as detailed by the surveys on practical applications of GNNs by Wu et al. (2021) and Zhou et al. (2020). Zhao et al. (2021) propose a representation iterative fusion based on heterogeneous GNNs for relation extraction (RIFRE) tasks, to construct knowledge from unstructured text in the NLP domain. In the financial domain, Li et al. (2021) apply GNN for merger and acquisition (M&A) prediction. Cheng et al. (2020) propose the application of GNNs for detecting and predicting risk guarantees in networked-loans. In particular, they propose a dynamic graph-based attention neural network (DGANN) where each guarantee is represented as an edge in dynamic loan networks, while companies are denoted as nodes. Thus, predicting risk guarantees is modeled as an edge classification task.

Table 1 summarizes a brief review of research from 2019 to 2023 on the measurement and analysis of systemic risk in the financial sector.

The network structure has been shown to be of great importance for systemic risk measures. While Machine Learning techniques used in the literature have proved to be successful for systemic risk analysis thanks to their ability to deal with the complex relations produced by the interactions between financial institutions, they cannot use the network structure and information in their predictions and, thus, do not exploit these interactions directly. By default, these techniques consider each observation independently and have to make use of centrality and other network metrics as proxies to include network information in their predictions. This is suboptimal since it limits the usable network information and forces explicit feature engineering. Moreover, it greatly reduces the benefits of Neural Networks, which are designed to deal with high-dimensional data to extract features automatically. GNNs extend the benefits of Neural Networks to graphs by modeling systemic risk measures using the graph itself, taking advantage of all the information available (node inter-relations and node, edge, and graph features). Thus, GNNs consider the interactions between financial institutions for modeling and do not require explicit network feature engineering. We expect GNNs to act as a meeting point between the literature on Machine Learning and networks for systemic risk analysis.

3. Graph neural networks

Graph Neural Networks (GNNs) are a class of deep learning algorithms designed to deal with data presented in graphs. As such, they calculate each node's representation considering its features and its neighbors' representations, thus aggregating information from the graph. GNNs can be directly applied to graphs to perform node-level, edge-level, or graph-level tasks.

- **Node-level** focus on the nodes. Some tasks are node classification (predicting its class), node clustering (separating nodes into groups of similar nodes), and node regression (predicting a continuous feature of the nodes).
- **Edge-level** focus on the edges. Some tasks are edge classification (predict its class) or link prediction (predict if an edge exists between two nodes).
- **Graph-level** focus on the graph as a whole.

Many different GNN architectures have been proposed (Wu et al., 2021). A brief description of GNNs is provided as follows.

The input is a graph with n nodes and m edges represented by its adjacency matrix A $(n \times n)$, a node feature matrix X $(n \times f)$ with f input features (x_v is the feature vector for node v), and an edge feature matrix X^e $(m \times f^e)$ with f^e input features (x^e_{vu} is the feature vector for edge (v,u)).

One widely used framework that generalizes spatial-based convolutional GNNs is the Message Passing Neural Networks (Gilmer et al., 2017). The message passing process with L layers is defined as follows

$$h_v^{(0)} = x_v$$

$$h_v^{(k)} = U_k(h_v^{(k-1)}, \sum_{u \in N(v)} M_k(h_v^{(k-1)}, h_u^{(k-1)}, x^e_{vu}))$$

where U_k and M_k are functions with learnable parameters and $N(v)$ are the neighbors of v. The output of the last layer $z_v = h_v^{(}L)$ is the node embedding of v, which is a vector that represents node v.

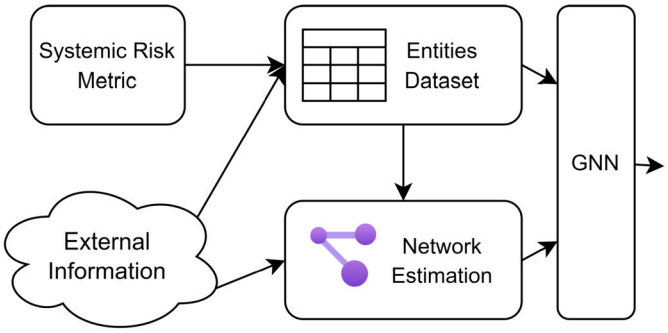


Fig. 1. Flowchart of the methodology followed.

Graph Convolutional Networks (GCN) (Kipf & Welling, 2017) generalize convolutional neural networks to graphs. The propagation rule for GCN is as follows

$$H^{(k)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(k-1)}W^{(k-1)})$$

where $H^(k)$ is the matrix of node representations (row v is $h_v^{(k)}$), $\sigma(\cdot)$ is an activation function (typically tanh or ReLU), $W^{(k)}$ is the matrix of learnable weights for layer k, $\hat{A}$ is the adjacency matrix with added self-loops ($\hat{A} = A + I$), and $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$.

As the number of neighbors of a node increases, it becomes inefficient to use all neighbors. GraphSAGE (Hamilton et al., 2017) samples the neighborhood to obtain a fixed number of neighbors for each node.

$$h_v^{(k)} = \sigma(W^{(k)} \cdot f_k(h_v^{(k-1)}, \{h_u^{(k-1)} \mid u \in S_{N(v)}\})$$

where $f_k(\cdot)$ is an aggregation function invariant to permutations (such as sum or mean), and $S_{N(v)}$ is a random sample of the node v's neighbors.

GCN assumes the contribution of each neighbor u of v to v's embedding is pre-determined ($\frac{1}{\sqrt{deg(u)deg(v)}}$). GraphSAGE assumes they are identical. Graph Attention Networks (Veličković et al., 2018) adopts attention mechanisms to learn the relative weights between two connected nodes.

$$h_v^{(k)} = \sigma(\sum_{u \in N(v) \cup v} \alpha_{vu}^{(k)} W^{(k)} h_u^{(k-1)})$$

where $\alpha_{vu}^{(k)}$ is the attention weight which weights the connection between v and u

$$\alpha_{vu}^{(k)} = \frac{\exp(g(a^T[W^{(k)}h_v^{(k-1)} || W^{(k)}h_u^{(k-1)}]))}{\sum_{k \in N(v)} \exp(g(a^T[W^{(k)}h_v^{(k-1)} || W^{(k)}h_k^{(k-1)}]))}$$

where $g(\cdot)$ is a LeakyReLU activation function, a is a vector of learnable parameters, and $||$ is the concatenation operation.

4. Methodology

This work first compares Graph Neural Networks (GNNs) to traditional Machine Learning techniques in dealing with financial networks. It then presents an approach to convert a classification task into regression to obtain more granular predictions. This section introduces the general methodology followed.

Fig. 1 provides a flowchart of the approach followed.

4.1. Entity dataset

First, we define the financial network which we want to model. This requires defining what entities will be included, which metric we will use to measure systemic risk, and what feature information will be used as input for the model.

There is no specific limitation regarding the metric used for systemic risk, apart from it being a function, possibly non-linear, of the feature information in order to ensure it can be modeled. Depending on the use case, this metric can be continuous (e.g., Marginal Expected Shortfall or CoVaR) or discrete (e.g., ranking or categorical classification), or even computationally costly (e.g., simulation-based metrics, high-dimensional data analysis, or expert or qualitative assessments). Since we are dealing with a supervised task, we will also need to pre-label a percentage of those entities, which will be given to the GNN during the training process to model the systemic risk metric. This entails calculating the chosen metric for a percentage of the entities.

Regarding the feature information, we will include prediction indicators for the systemic risk metric of choice. One of the strengths of Deep Learning models is their ability to deal with high-dimensional data. GNNs can automatically extract features from high-dimensional data without the need for explicit feature engineering. This allows using a large number of features can be used as input to model the chosen systemic risk metric, allowing for complex relations.

4.2. Network estimation

The systemic risk importance of an entity does not solely depend on that entity by itself. The network structure is important for systemic risk because it captures the interconnectedness and dependencies among financial institutions or markets, which can amplify and propagate shocks and failures throughout the financial system. We also provide the financial network to model the systemic risk metric and take advantage of the information available in the linkages through the use of GNNs. This network is provided as a directed weighted graph.[2] This differs from the traditional Machine Learning pipeline.

While financial network data is available to financial institutions, it is usually confidential and not of public access. However, there are multiple techniques available in the literature that allow estimating the financial network from aggregated data, which is more accessible. Two popular network reconstruction methods for financial networks are the maximum entropy (Upper & Worms, 2004) and the minimum density (Anand et al., 2015) estimations. The maximum entropy method assumes that each node tries to diversify its exposures as evenly as possible, while the minimum density method assumes financial networks are sparse and disassortative. Both methods ensure the restrictions of assets and liabilities are satisfied.

Moreover, GNNs enable including additional information into the model through the network's construction. Some examples in the literature are estimating the financial network based on text co-mentions (Rönnqvist & Sarlin, 2015, 2014) or based on a combination of financial data and financial tweets (Cerchiello et al., 2017).

4.3. Training GNN

The entity feature information, with a percentage of the entities pre-labeled, and the graph are used as input to train a GNN. This is typically done by defining a loss function which is minimized through the backpropagation algorithm. The GNN can be represented as a function $f(G, X)$, where G is the directed weighted graph representing the financial network and X is the feature matrix. Depending on the systemic risk metric chosen the output might be treated differently. For example, if the metric is continuous it can be modeled as $y = f(G, X)$, while if the metric is discrete with m levels, the probability of an entity being classified as level i can be modeled as $[softmax(f(G, X))]_i$, with $[softmax(z)]_i = \frac{\exp(z_i)}{\sum_{j=1}^{m}\exp(z_j)}$.

5. Classes to regression

Depending on the context, discrete systemic risk metrics might be preferred over continuous metrics. Discrete metrics provide a clear-cut

[2] Some GNN models can only model unweighted graphs.

categorization of the institutions into different risk levels, which can be particularly useful for policymakers and regulators who need to make decisions based on the risk level of individual institutions. Moreover, in situations where it is costly or complex to calculate the metric for each institution, such as expert or qualitative assessment, it is simpler, less error-prone, and more consistent to classify each institution into one of a k categories instead of giving each one a score. Moreover, similar entities close to the class borders might be classified into different classes and thus thought to be different, thus providing a misrepresentation of their true risk score. However, using continuous metrics allows for a better utilization of the information allowing to capture subtle changes, is more robust to label noise, and provides more granular and precise risk estimation.

We present an approach to take advantage of the simple pre-labeling provided by discrete ordinal systemic risk metrics while training a model that predicts a continuous risk score. During training, the input of the model will be the same as for classification, i.e., the graph G representing the financial network, the entity feature information matrix X, and the class labels provided by the discrete systemic risk metric for a percentage of the entities. However, instead of modeling the predicted class of an unseen entity, the model will predict its quantile (score in the $[0, 1]$ range).

5.1. Process overview

The model architecture used is the same as for classification, being the predicted score by the GNN $f(G, X)$. When training a Deep Learning model, we have to define a loss function that represents the error of the model's prediction. During training, the model optimizes its predictions by minimizing the loss. Since the output for this scenario is continuous, we will use a regression loss function such as the Mean Squared Error loss (MSE) and Mean Absolute Error loss (MAE). Algorithm 1 presents the process used to calculate the loss value for backpropagation.

To calculate the regression loss we need to establish a conversion between the class label provided for training and the target output of the model used to calculate the loss. Thus, we need to define a function $get\_target(\cdot)$ which returns the target used to compute the training loss (line 1.3) given the true label.

Algorithm 1 Loss value calculation in Class to Regression.

1: **Input.** $x \leftarrow$ the model's raw output
   $l \leftarrow$ target labels
2: $q \leftarrow sigmoid(x)$ ▷ scale to range (0,1), where $sigmoid(\cdot) = \frac{1}{1+e^{-x}}$
3: $t \leftarrow get\_target(l)$ ▷ $get\_target(\cdot)$ gets the target value from the label
4: $loss\_value \leftarrow loss\_f(q, t)$ ▷ $loss\_f(\cdot)$ is the loss function
5: **return** $loss\_value$

5.2. Problem definition

We will assume that we have N classes and that each class represents an equal-length interval of the quantile range. If this condition is not met and instead each class represents a rational length interval, the problem can be simplified by transforming it into an equal-length problem. Given that D is the common denominator of all the interval fractions, the length of the interval k is $\frac{n_k}{D}$. Then, the quantile range is divided into D equal-length intervals and each class k will be assigned n_k intervals (in order) which are mapped to class k.

From now on we will assume equal-length intervals. We can then assign to each class $k \in \{0, \dots, N-1\}$, the quantile range $\left[\frac{k}{N}, \frac{k+1}{N}\right)$.[3] Given a loss function $loss(\cdot)$, the true label for a sample l, the true quantile for that sample q, and the model's predicted quantile x, the objective is to define a function $get\_target(\cdot) : \{0, \dots, N-1\} \rightarrow [\not{\jmath}, \not{\jmath}]$ such that a model trained to minimize $loss(get\_target(l), x)$ can closely predict the true quantile q ($f(G, X) + \epsilon = q$ with ϵ being the error).

[3] Except class $N-1$ which will be assigned the closed interval $\left[\frac{N-1}{N}, 1\right]$.

5.3. Naive approach

A naive approach, to which we will refer to as *base_n*, could be achieved by dividing the label by the number of classes. Therefore

$$get\_target(l) = \frac{l}{N} \tag{1}$$

To get back the predicted class from the predicted quantile

$$\begin{aligned} to\_class(x) &= k \mid x \in \left[\frac{k}{N}, \frac{k+1}{N}\right) \\ &= \max(\lfloor N \cdot x \rfloor, N-1) \end{aligned} \tag{2}$$

where $\lfloor x \rfloor$ is the floor of x.

5.4. Equal-scale approach

When the node's label is $N-1$ and $f(G, X) > \frac{N-1}{N}$, *base_n* approach implies $f(G, X) > \frac{N-1}{N} = get\_label(N-1)$. Thus the models predicted quantile for class $N-1$ are pushed towards the class border around $\frac{N-1}{N}$. The underlying issue is that quantiles take values in the range $[0, 1]$ while $get\_target(\cdot) \in [0, \frac{N-1}{N}]$.

This issue can be solved by correctly scaling the target values so its range matches the quantile interval. Therefore, the new $get\_target(\cdot)$ function that replaces Equation (1) is

$$get\_target(l) = \frac{l}{N-1} \tag{3}$$

5.5. Equal-distribution approach

The loss of the model is computed as $loss(get\_target(l), f(G, X))$. The model will try to minimize the loss by predicting values close to the target. Therefore, the target label should be a good representative of the class range. Depending on the data and the class, this value can be calculated in different ways.

If the original distribution for each range is known (or can be approximated) it is a good idea to choose the class representative/target such that the predictions' distribution is similar to the original distribution for each class. By doing so, the knowledge of the distribution for each class will be incorporated into the model.

A good (and simple) approach to select the representative is to use a centrality measure. This encourages the model to make its predictions around that value, obtaining a distribution of predictions with a similar centrality measure to the original distribution. Moreover, this can be used in scenarios where the true distribution is unknown but its centrality measures are known.

For instance, we could use the mean/median as target. Since $f(G, X)$ is a predicted quantile and the quantiles follow a uniform distribution, their mean/median has the same value, which is the middle of each quantile interval.

$$get\_target(l) = \frac{2l+1}{2N} \tag{4}$$

To show the difference between using the mean/median as the target and following Equation (3) (*equal-scale* approach), we calculate the distance between the target for label l obtained through Equation (3) and the upper and lower limits of its quantile range.

$$d_l = \left| \frac{l}{N-1} - \frac{l}{N} \right| = \frac{l}{N(N-1)}$$

$$D_l = \left| \frac{l+1}{N} - \frac{l}{N-1} \right| = \frac{N-1-l}{N(N-1)} = \frac{1}{N} - d_l$$

These distances are generally not equal (only for $l = \frac{N-1}{2}$). Assume we have a model that obtains a loss of zero. Then, $f(G, X) =$

$get\_target(l) = \frac{l}{N-1}$ which as we have seen is not the mean of the quantile interval. This results in the model's predictions for class l levitating towards the limits of the quantile range, instead of towards its mean.

5.6. Extensions for equal-distribution approach

5.6.1. Known true/expected output distribution

If we have some knowledge about the true, or expected, output distribution, we can incorporate that information into the model by using a distribution similarity score, such as the Kullback-Leibler divergence or the Jensen-Shannon divergence. Then the model will also minimize the similarity between the predicted distribution and the true output distribution. This can be considered as using the true distribution as a regularization to incorporate prior knowledge into the model. The model still needs to learn to generalize to predict the quantiles, but it now has additional information that it can use to adjust its predictions.

This can be done by combining the regression loss with a loss based on the distribution similarity score.

$$loss = regression\_loss + \alpha \cdot distribution\_similarity$$

with α being the weight of the distribution similarity compared to the regression loss.

5.6.2. Learned target

This approach utilizes the learning capabilities of Neural Networks (or another architecture) to obtain the best target value. Instead of using a pre-chosen target value, the $get\_target(\cdot)$ function (line 1.3) is learned by a model.

- **Input.** The input of the model is the true label l. Additional features from that label's quantile distribution (e.g., its mean, median, variance, or an embedding for the distribution) can be used.
- **Output.** The output will be the target value used to compute the main model's loss which for label l must be between $\left[\frac{l}{N}, \frac{l+1}{N}\right)$.
- **Loss.** The target model will use the same loss as the main model for back-propagation.

6. Case studies

This section compares the effectiveness of traditional Machine Learning and GNNs for systemic risk prediction on financial networks. We compare them on a classification task since it is one of the most widely used data science tasks in business.

We compare the GNNs' performance on two different financial networks. We first compare their effectiveness using synthetic data, and then using real historical data from 2017 for 1444 European banks. For both networks, we test different percentages of pre-labeled data to test their generalization capabilities. We utilize the estimated interbank network in our research as common bilateral exposure data are confidential. Specifically, we use the minimum density approach, which assumes financial networks are sparse and disassortative.

6.1. Systemic risk metric

For the systemic risk metric, we consider 4 systemic importance classes based on the additional system stress measured by a contagion process with linear propagation. The node features used as input of the model are the features used for the contagion process, which are the entity's total inter-node assets, total inter-node liabilities, and buffer (the amount that the entity can lose before defaulting), plus the entity's weight (which represents the entity's relevance in the network). The systemic risk metric obtained will have a non-linear relationship with the node features due to the importance of the connections.

More specifically, given the financial network and the total assets, liabilities, and buffer of each entity, we run a contagion algorithm (line 2.3) to estimate the additional system stress produced by the default of each entity. This is done using linear propagation, also known as DebtRank (Bardoscia et al., 2015, Battiston et al., 2012), which accounts for the loss of value of a node even if it does not fully default.[4] The entities are then ordered by their additional system stress and are classified into one of the 4 classes corresponding to their quartile. Thus, the classes are ordered by level of importance, being class 0 the most important and class 3 the least important. The number of classes has been chosen to simulate a feasible number in which entities could be classified through expert assessment in a real data scenario. The process followed to obtain the importance classes/labels of each of the network's nodes is presented in Algorithm 2.

Algorithm 2 Node Labeling Process.

1: **Input.** $a \leftarrow$ vector of inter-node assets
$l \leftarrow$ vector of inter-node liabilities
$b \leftarrow$ vector of capital buffers
$w \leftarrow$ vector of weights
2: Exposure matrix $E \leftarrow mimimum\_entropy\_reconstruction(a, l)$
3: Additional system stress $\Delta ss \leftarrow debtrank\_contagion(E, b, w)$
4: Calculate the Δss quantiles $Q_{0.25}$, $Q_{0.5}$ and $Q_{0.75}$
5: **for** $n \in nodes$ **do**
6: Label node n such that

$$label(n) = \begin{cases} 0 & \text{if } \Delta ss_n \geq Q_{0.75} \\ 1 & \text{if } Q_{0.5} < \Delta ss_n \leq Q_{0.75} \\ 2 & \text{if } Q_{0.25} < \Delta ss_n \leq Q_{0.5} \\ 3 & \text{if } \Delta ss_n \leq Q_{0.25} \end{cases}$$

7: **end for**

The systemic risk metric has been chosen to focus on the effects that the failure of a node produces on the rest of the nodes in the system. While it could be considered that the *total system stress* produced by the default of a node is a good indicator, we considered the *additional system stress* to be a better indicator. For each node k

$$\begin{aligned} \text{total system stress}_k &= \text{original system stress}_k + \text{additional system stress}_k \\ &= \frac{\text{weight}_k}{\sum_{n \in \text{nodes}} \text{weight}_n} + \text{additional system stress} \end{aligned} \tag{5}$$

the *total system stress* is directly dependent on the node's weight, giving less importance to how node k affects the rest of the nodes in the system. In contrast, the *additional system stress* is only indirectly affected by the node's features through the node's links, making this variable a better indicator.

6.2. Synthetic case study: SymNet

We first compare GNNs and traditional Machine Learning on a synthetic heavy-tailed financial network composed of 1500 nodes. Each node has been generated by randomly sampling from a heavy-tailed probability distribution.

Inter-node assets and liabilities. Both are sampled from a lognormal distribution with mean 0 and standard deviation 2 ($x \sim LogN(0, 4)$). After sampling from this distribution, we filter out those banks with a value lower than 4 ($p(x < 4) = 0.76$), with the rest being our "big nodes". For the remainder of the nodes, we will sample from a uniform distribution $U(0, 1)$, thus obtaining a heavy-tailed distribution. This process is done for the assets and liabilities separately, and then it is ensured that total assets equal total liabilities.

Buffer. It is calculated as a percentage of the liabilities, which represents the debt that the node is covering, plus a fixed value, which represents the node's extra capital. The percentage is sampled from a

[4] With DebtRank, when a node loses a certain percentage of its capital buffer, that same percentage of its inter-node liabilities is propagated to its creditors.

Fig. 2. SymNet Network.

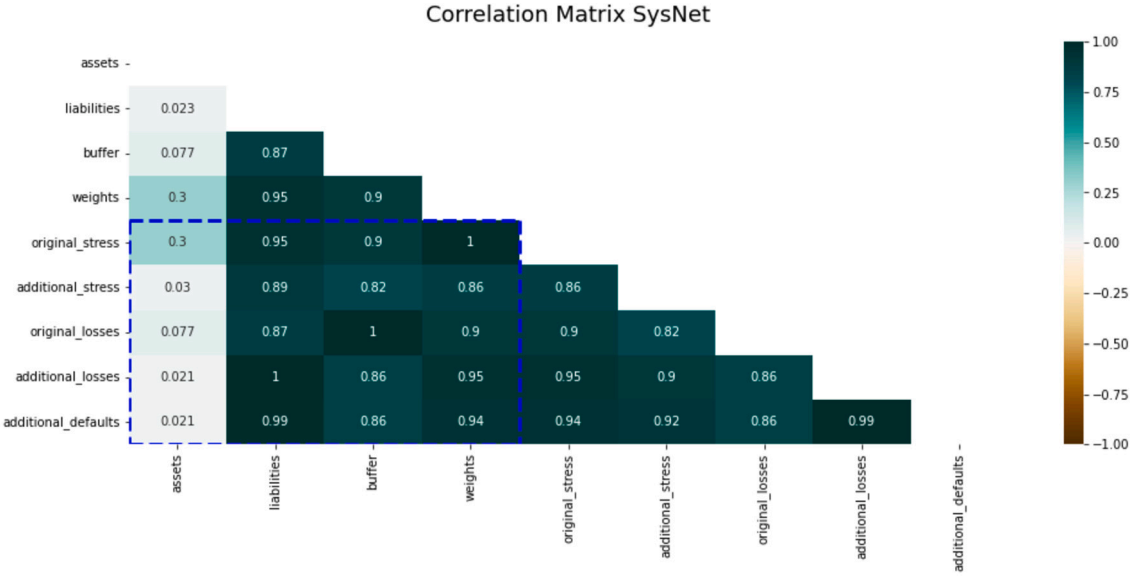


Fig. 3. SymNet correlation of the outputs and inputs of the contagion process.

$U(0,1)$ and the fixed portion is sampled from the absolute value of a normal distribution.

Weights. The importance of the node in the system has been represented as the sum of the node's assets, liabilities, buffer, and a $LogN(0,1)$.

Resulting Network. The resulting network is presented in Fig. 2.

Contagion Variables Correlation. In the blue square of the correlation matrix presented in Fig. 3 it can be observed the correlation between the outputs and inputs. The correlation matrix further justifies choosing *additional_stress* as the base for the systemic risk metric. It is important to note that *original_stress* and *original_losses* are directly related to the node attributes. Specifically, *original_losses* highly correlates with *buffer*, and *original_stress* correlates with *weights* as showed in Equation (5).

Resulting Additional Stress. When plotting each node's original system stress against the additional system stress generated by their default, we can observe that there is no clear relationship between these two variables. This can be better appreciated when looking into the smaller nodes in Fig. 4, where nodes with the same original stress, generate very different additional stress.

In Fig. 4 the quartiles that separate the labels have also been drawn. It is important to note that quartiles Q_0 and $Q_{0.25}$ are very close together, so we expect our models to have difficulties separating labels 2 and 3.

6.2.1. *Real case study: EurNet*

We then compare GNNs and traditional Machine Learning on a financial network with 1444 European banks. We have used historical data from 2017 gathered from the S&P Global platform.[5] By doing so we obtain a distribution of inter-bank assets and liabilities that match the real European bank network to the extent possible.

Inter-node Assets and Liabilities. The bank's *Net Loans to Banks* have been used as inter-node assets, and the *Total Deposits from Banks* have been used as inter-node liabilities. It has been ensured that the sum over the whole network of assets is equal to the sum of liabilities.

Buffer. The bank's *Total Capital*, the sum of *TIER1* and *TIER2*, has been used as the capital buffer.

[5] We appreciate AFI (Analistas Financieros Internacionales) for their help and support in obtaining this data.

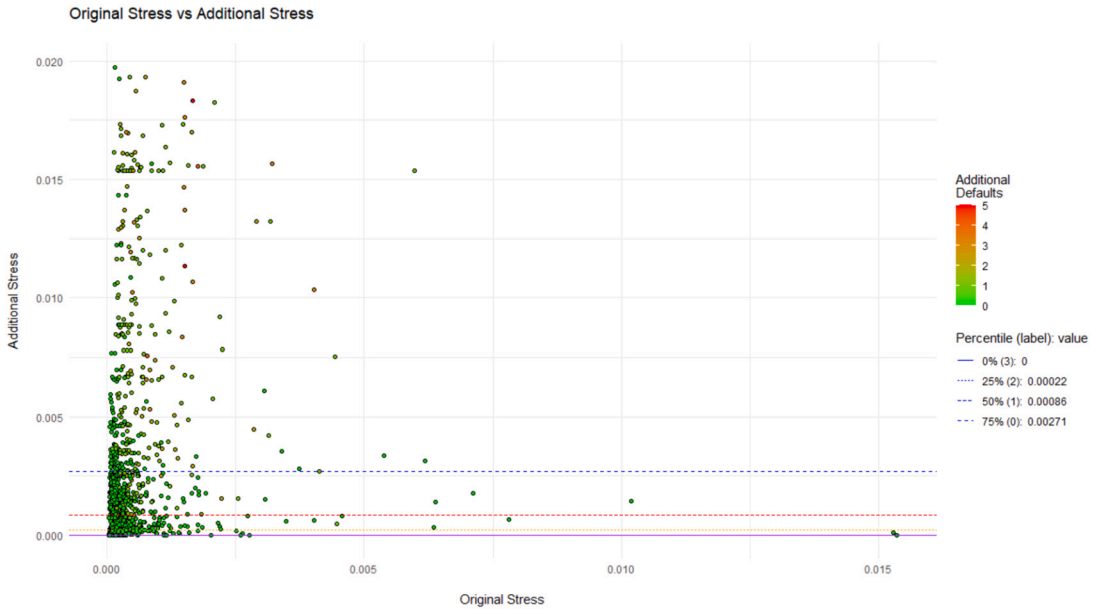


Fig. 4. Symnet original vs additional system stress for smaller nodes.

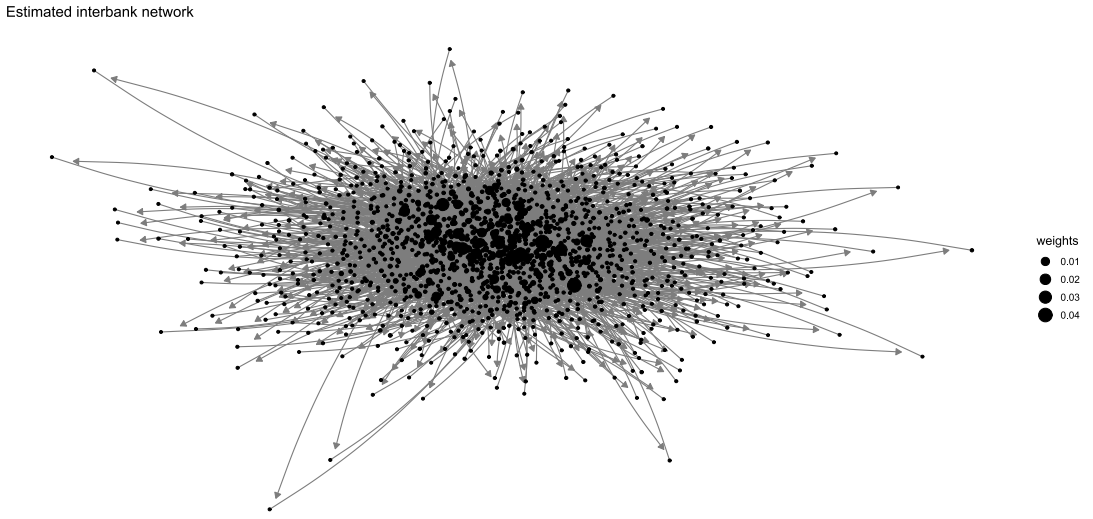


Fig. 5. EurNet Network.

Weights. The importance of each bank over the system has been set to its *Total Liabilities*. This variable refers to the total amount of debt across the whole financial system which would be affected by the bank's default, not only the interbank network. Thus, it represents the overall importance of the bank in the financial system.

Resulting Network. The resulting network is presented in Fig. 5. It can be observed that the nodes further away from the center only have connections with a few other nodes, while the ones in the center have a higher number of connections.

Contagion Variables Correlation. In the correlation matrix presented in Fig. 6, it can be observed how *original_stress* and *original_losses* have a high correlation with the input variables while the rest of the output variables have a very low correlation. This information further justifies basing the systemic risk metric on the *additional_stress*.

Resulting Additional Stress. When plotting each node's original system stress against the additional system stress generated by their default (Fig. 7), it can be observed that there is no clear relationship between these two variables. It can also be observed that *additional stress* is very concentrated around two values, with quartiles $Q_{0.75}$ and $Q_{0.5}$, and quartiles $Q_{0.25}$ and Q_0, having very similar values. Therefore, entities classified as class 0 and 1 will be highly similar. The same can be said for entities classified as classes 2 and 3.

Table 2
SymNet and EurNet quantiles of additional system stress.

Quantile	Q_0	$Q_{0.25}$	$Q_{0.5}$	$Q_{0.75}$	Q_1
SymNet	0,0000	0,0002	0,0009	0,0027	0,2851
EurNet	0,00000	0,00001	0,45093	0,45096	0,47666

Table 2 presents the quantile values of additional system stress which are the thresholds for each of the systemic risk classes.

6.3. Scenarios

For each network, we have considered three scenarios with 75%, 40%, and 10% of pre-labeled entities. Doing so allows us to compare the generalization capabilities of GNNs and traditional Machine Learning when training data becomes scarce. This is also useful to observe how

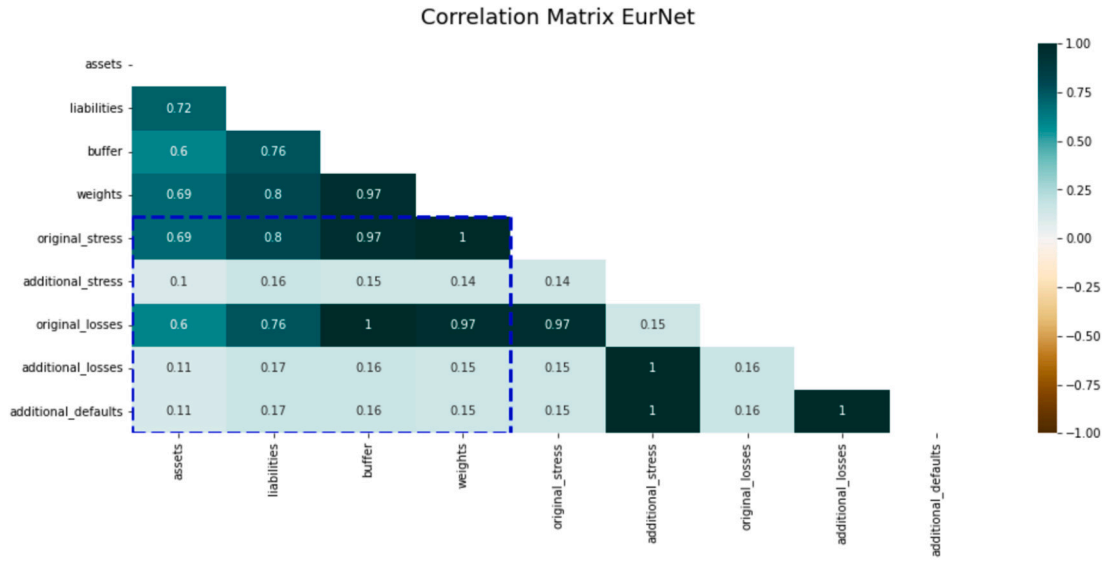


Fig. 6. EurNet correlation of the outputs and inputs of the contagion process.

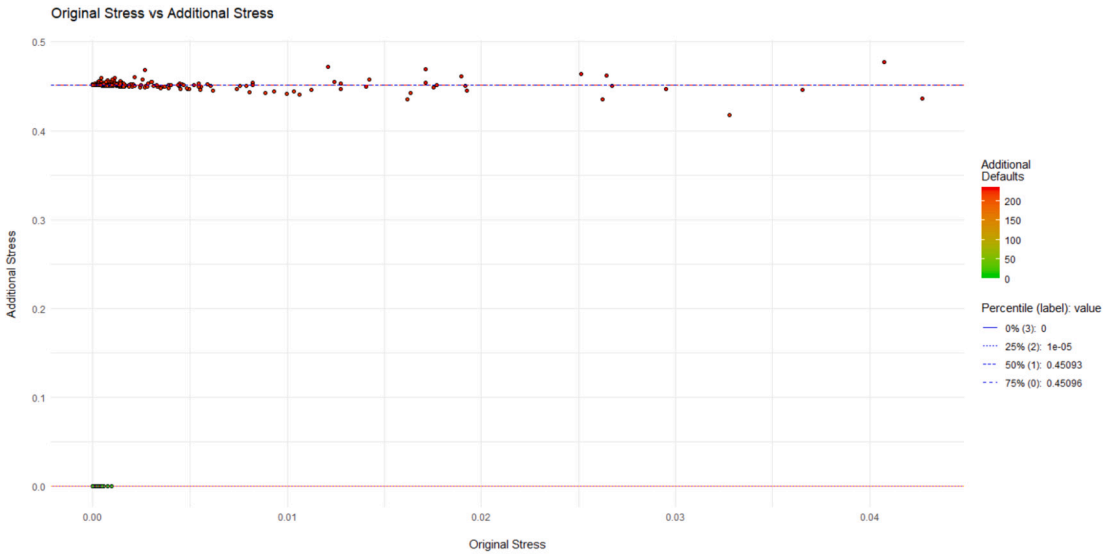


Fig. 7. EurNet original vs additional system stress.

Table 3
SymNet and EurNet scenarios details.

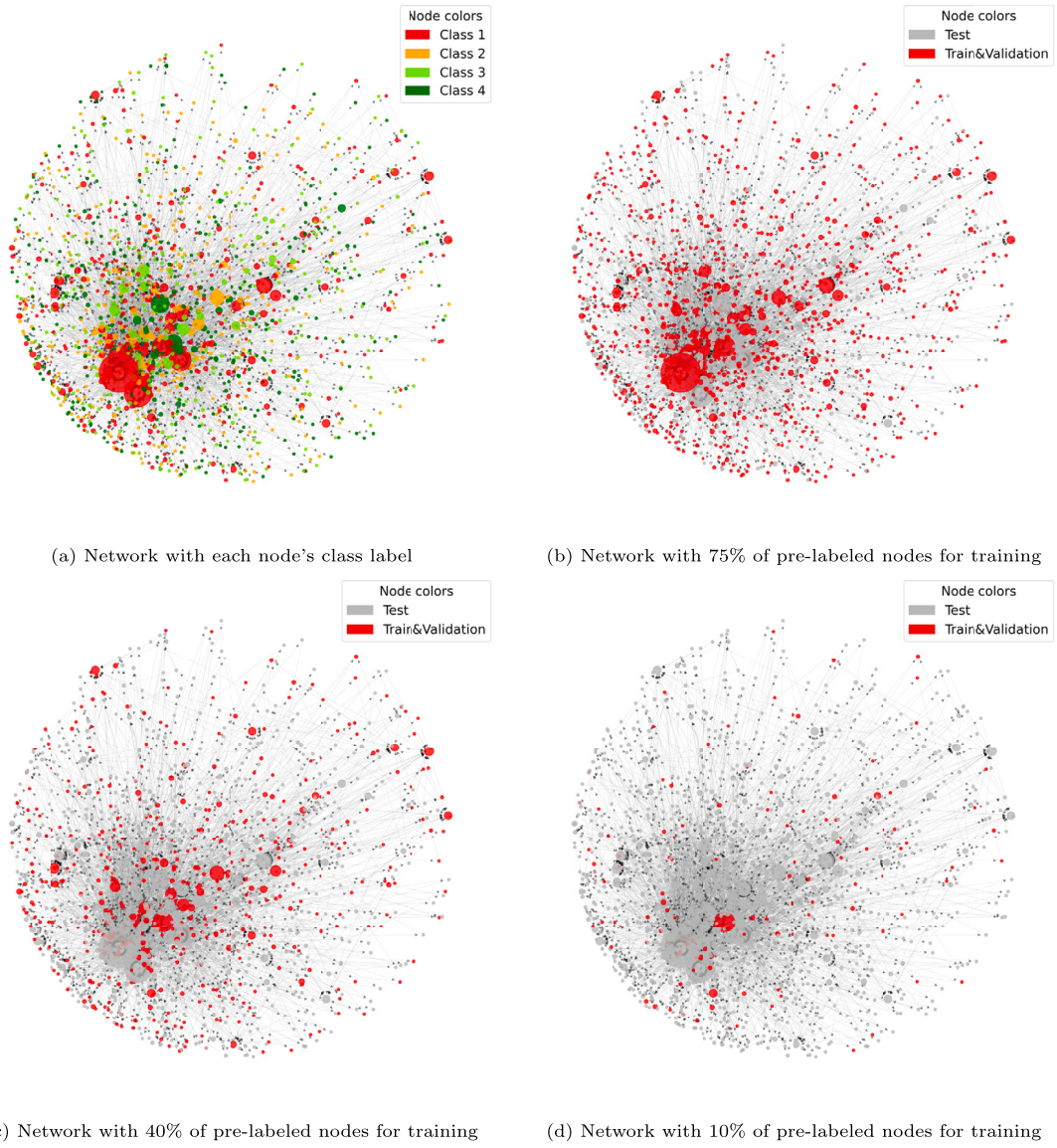
Dataset	Nodes	Edges	Scenario	Train/Validation/Test %	Train/Validation/Test Nodes
SymNet	1500	3006	SymNet_75	60/15/25	900/225/375
			SymNet_40	32/8/60	480/120/900
			SymNet_10	8/2/90	120/30/1350
EurNet	1444	2893	EurNet_75	60/15/25	866/216/362
			EurNet_40	32/8/60	462/115/867
			EurNet_10	8/2/90	115/28/1301

sensible the models are to a reduction in the training data and what results to expect if we were to pre-label a small percentage of the nodes and use the model to classify the rest. Plots of the networks and the amount of pre-labeled nodes are presented in Fig. 8 for SymNet and Fig. 9 for EurNet.

The scenarios are presented in Table 3. The suffix in the scenario name indicates the percentage of the total nodes that have been pre-labeled. The pre-labeled nodes have been split into two sets, training and validation, with 80% and 20% of the pre-labeled data respectively. The training set is used to fit the models and the validation set is used to tune the hyperparameters. The rest of nodes that have not been pre-labeled are considered the test set, which is used to compute an unbiased estimate of the model's performance.

6.4. Models

The baseline models are traditional Machine Learning models which do not use the graph structure of the data. The baseline models used in the experiments are Logistic Regression (LR), Random Forest Classi-



(a) Network with each node's class label

(b) Network with 75% of pre-labeled nodes for training

(c) Network with 40% of pre-labeled nodes for training

(d) Network with 10% of pre-labeled nodes for training

Fig. 8. Representation for the SymNet network. Each circle represents a node, with the size of the circle being proportional to the node's degree.

fier (RF), KNN Classifier (KNN), and a Fully Connected Neural Network (FCNN). The FCNN has been included to compare a non-graph Deep Learning model with the GNNs. As for the GNN models, the models used are Graph Convolutional Networks (GCN) (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), and Graph Attention Networks (GAT) (Veličković et al., 2018).

The experiments have been run using Python except for the contagion process and network estimation which have used the implementation from the R library NetworkRiskMeasures. For the baseline models, we have used the implementation provided by skicit-learn. For the FCNN and the GNNs, we have used PyTorch and DGL, and a GPU RTX 2070[6] for training.[7]

For each model we tested multiple hyperparameter combinations, choosing the highest Matthews Correlation Coefficient (MCC) on the test set for comparison. The MCC has been chosen as the main metric, given its advantages over the accuracy and F1 score (Chicco & Jurman, 2020).

[6] Some GNN models trained with a GPU with Pytorch are nondeterministic, thus, there may be small variations.

[7] The code is available at https://github.com/vibalcam/gnn-systemic-risk.

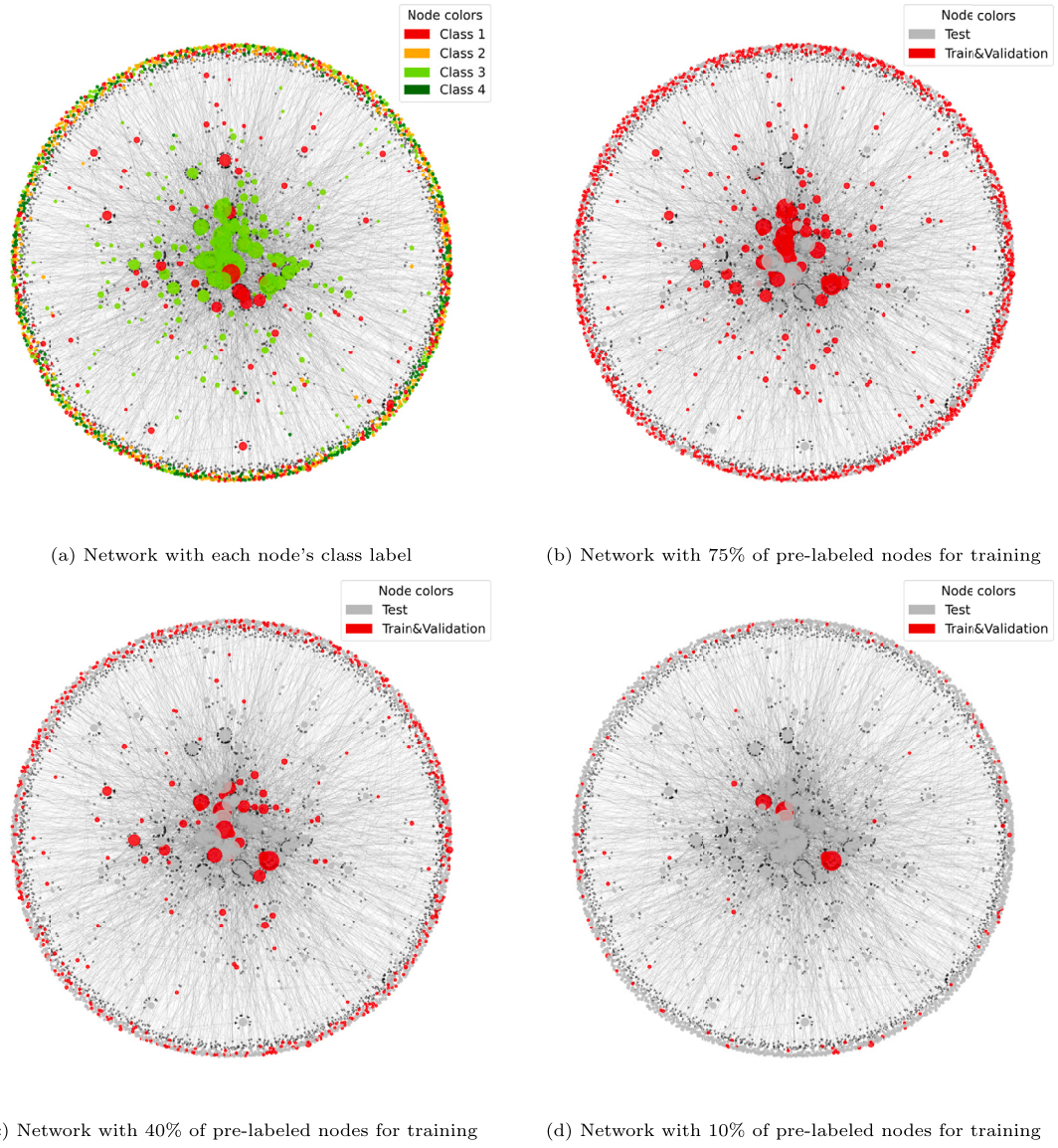
Table 4
Inputs/output of baseline and GNN classification models.

Attributes		Baseline	GNN
Inputs	Total Inter-node Assets	✓	✓
	Total Inter-node Liabilities	✓	✓
	Buffer	✓	✓
	Weight	✓	✓
	Network[a]	✗	✓
Output		Label	Label

[a] Weighted directed graph where w_{ij} means i has an asset on j of value w_{ij}.

The inputs and outputs of the models are summarized in Table 4. The only difference between the baseline and GNN models is that the GNNs can take advantage of the network structure of the data.

The systemic risk metric chosen is discrete and ordinal. Thus, we can apply the Class to Regression (C2R) approach presented in Sec. 5 to predict the quantile for each node. We have used a Mean Squared Error (squared L2 norm) loss and the same models as for the multi-class classification task. The best hyperparameter combination is chosen to minimize the Root Mean Squared Error (RMSE) between the predicted and true quantile on the test set. Table 5 compares the inputs and output



(a) Network with each node's class label

(b) Network with 75% of pre-labeled nodes for training

(c) Network with 40% of pre-labeled nodes for training

(d) Network with 10% of pre-labeled nodes for training

Fig. 9. Representation for the EurNet network. Each circle represents a node, with the size of the circle being proportional to the node's degree.

Table 5
Inputs/output of classification and C2R GNN models.

Attributes		Classification	C2R
Inputs	Total Inter-node Assets	✓	✓
	Total Inter-node Liabilities	✓	✓
	Buffer	✓	✓
	Weight	✓	✓
	Network[a]	✓	✓
Output		Label	Continuous (Regression)
Target		Label	Label

[a] Weighted directed graph where w_{ij} means i has an asset on j of value w_{ij}.

of the classification and C2R models. The available information for both is the same, being the only difference in the output of the model, a label for classification, and a continuous value for C2R.

6.4.1. Baseline

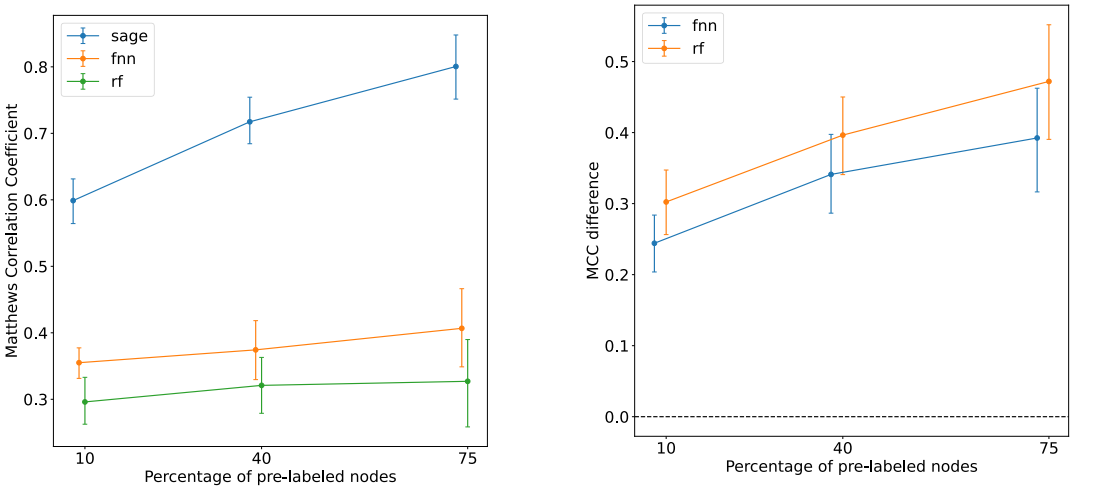
For the LR, RF, and KNN models, we have fitted the models using both the train and validation sets. For the RF, 20 models have been trained with 1 to n number of estimators, being n the number of nodes in the network.

For the FCNN, several 1 to 3-layer models have been trained. We tested hidden sizes in the range $[5, 30]$ for SymNet and $[100, 800]$ for EurNet. All the models have used a ReLU activation function and dropout values of $\{0, 0.2\}$. We have tried trained models without normalization, with BatchNorm and with GraphNorm (Cai et al., 2020). More details on the hyper-parameters can be found in Sec. Appendix B. The input is first passed through a normalization layer, and then through the FCNN layers which are composed of

- Fully connected layer
- Normalization layer
- Activation function. A ReLU activation function has been used.
- Dropout

6.4.2. Graph neural networks

For the GNNs, we first preprocessed the graph. We have added self-loops, which is necessary for some models that give invalid output values for zero in-degree nodes. We have also used DropEdge (Rong et al., 2019), which consists in removing a certain percentage of the graph's edges during training, acting as a data augmenter. DropEdge has been shown to increase generalization capabilities by alleviating over-fitting and over-smoothing.



(a) MCC confidence interval

(b) Confidence interval of the MCC improvement between GraphSAGE and baseline

Fig. 10. GraphSAGE achieves statistically significant MCC improvements over the traditional Machine Learning baseline on SymNet.

We have tested for GCN, 2 to 4-layer models; for GraphSAGE, 2 to 3-layer models with different aggregators (LSTM achieves the best results); and for GAT, 1 to 3 multi-head GAT layers. We tested hidden sizes in the range $[5, 30]$ for SymNet and $[100, 800]$ for EurNet. All the models have used a ReLU activation function and dropout (feature and/or node dropout) values of $\{0, 0.2\}$. We have tested models without normalization, with BatchNorm and with GraphNorm. More details on training and hyperparameters can be found in Sec. Appendix B. Each of the GNN layers follows this structure

- **GNN layer.** GraphConv, SAGEConv or GATConv layer depending on the model.
- **Activation function.** A ReLU activation has been used.
- **Node normalization layer.** Can be none, batch normalization or GraphNorm (Cai et al., 2020). GraphNorm is a learnable normalization designed for GNNs which has been shown to work better than BatchNorm and to improve the generalization capabilities of GNNs.
- **Dropout.** With the exception of GATs, which do not have this layer.

Apart from node features, edge weights were also incorporated into the models as an edge feature, except GATs which currently do not support edge features. The optimizer used for training is AdamW (Loshchilov & Hutter, 2019). For the classification task, a cross-entropy loss with $ls \in \{0, 0.2\}$ label smoothing (Szegedy et al., 2015) has been used. The models have been trained for 100 epochs, except GraphSAGE on EurNet_75 and EurNet_40 which have trained for 2000 epochs, and GraphSAGE on EurNet_10 which has trained for 1000 epochs.

7. Results analysis and discussion

For the multi-class classification problem, we will provide the model's results using the Matthews Correlation Coefficient (MCC) as our main metric, given its advantages over the accuracy and F1 score (Chicco & Jurman, 2020). We will also report the accuracy since the labels are balanced. Moreover, since the classes have an order, the closer our prediction is to the true class, the better. We will measure this using the RMSE.

$$ClassRMSE(c, \hat{c}) = \sqrt{\frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (c_i - \hat{c}_i)^2} \tag{6}$$

where c is the true class and $\hat{c}$ is the predicted class.

The non-parametric bootstrap was used to estimate the variability around model performance. A total of 1000 bootstrap replicates from the test set were drawn, and the MCC and its corresponding differences were calculated for the best baseline and best GNN models. This produced a distribution for each estimate, and the 95% bootstrap percentile intervals were computed to assess significance at the $p = 0.05$ level.

Some tables will have a model named *Best Baseline*. This model is the best result achieved by a baseline model in each metric category. Thus, for the same scenario, different *Best Baseline* metrics may have been achieved by different baseline models.

Table 6
SymNet baseline test metrics.

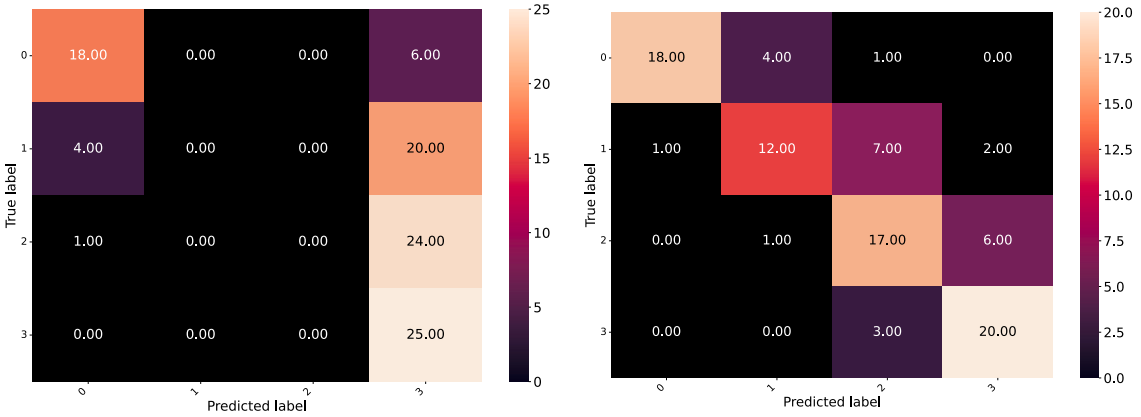
Scenario	Metric	LR	RF	KNN	FCNN
SymNet_75	MCC	0.335	0.328	0.188	**0.405**
	Accuracy	0.488	0.496	0.392	**0.525**
	Class RMSE	0.961	0.977	1.137	**0.917**
SymNet_40	MCC	0.331	0.321	0.196	**0.36**
	Accuracy	0.483	0.49	0.397	**0.496**
	Class RMSE	**1.006**	1.024	1.22	1.093
SymNet_10	MCC	0.282	0.297	0.18	**0.342**
	Accuracy	0.456	0.472	0.383	**0.487**
	Class RMSE	**1.023**	1.04	1.259	1.088

7.1. SymNet

The baseline models' test results are summarized in Table 6. It can be observed that the FCNN achieves the best baseline results for all scenarios, being closely followed by Random Forest. It is important to mention that Random Forest is faster to train than FCNN, which, depending on the use case, might compensate for the lower performance.

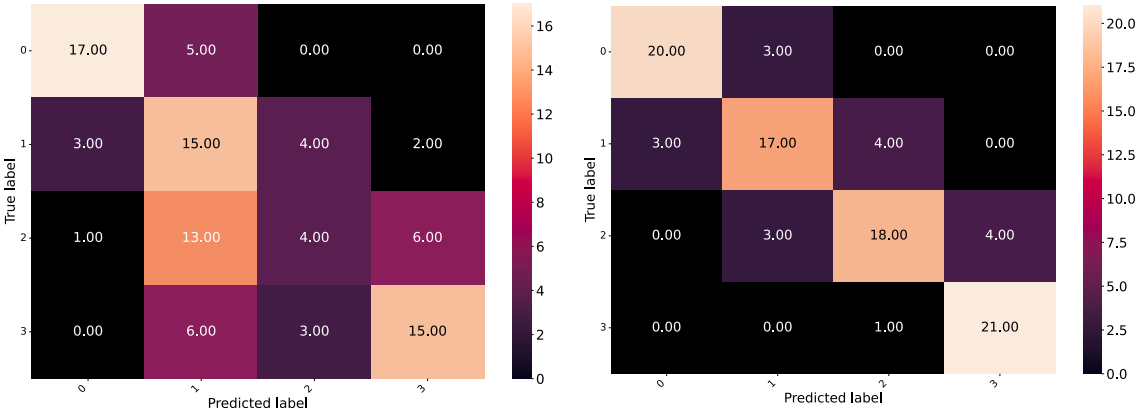
GraphSAGE demonstrated statistically significant performance gains over the traditional Machine Learning baseline. Fig. 10 presents the MCC bootstrap confidence intervals for GraphSAGE, RF, and FCNN, and the difference between GraphSAGE and the baseline models. The MCC for GraphSAGE is consistently higher than the MCC for the baseline models, showing that GraphSAGE models better the systemic risk metric than the baseline models. With 10% of pre-labeled nodes, the improvement in performance is 0.244 (95% CI 0.204, 0.284), which increases with the amount of data used for training, reaching 0.392 (95% CI 0.316, 0.462) with 75% of pre-labeled nodes.

The GNN models' test results are summarized in Table 7. GraphSAGE achieves better results in all metrics, surpassing the baseline models in all scenarios. GraphSAGE does not only obtain better overall results but also per-class results. When looking at per-class accuracy (Table 8),



(a) FCNN (b) GraphSAGE

Fig. 11. Normalized confusion matrices for SymNet_10 (percentage of total).



(a) FCNN (b) GraphSAGE

Fig. 12. Normalized confusion matrices for SymNet_40 (percentage of total).

Table 7
SymNet baseline vs GNN test metrics.

Scenario	Metric	Best Baseline	GCN	GraphSAGE	GAT
SymNet_75	MCC	0.405	0.43	**0.802**	0.372
	Accuracy	0.525	0.544	**0.851**	0.528
	Class RMSE	0.917	0.921	**0.482**	0.947
SymNet_40	MCC	0.36	0.419	**0.764**	0.407
	Accuracy	0.496	0.533	**0.822**	0.548
	Class RMSE	1.006	0.894	**0.481**	0.956
SymNet_10	MCC	0.342	0.408	**0.603**	0.341
	Accuracy	0.487	0.523	**0.701**	0.49
	Class RMSE	1.023	0.942	**0.643**	1.087

GraphSAGE obtains better results in almost all cases when compared to FCNN, which was the best baseline model.

GraphSAGE achieves 0.603 MCC, 70% accuracy, and 0.643 *Class RMSE* in the SymNet_10 scenario. This last metric can be interpreted as, by providing the model with 150 labeled nodes, the model is able to classify the other 1350 missing the true class by 0.643. Thus, on average, it either correctly predicts the true class or predicts a neighboring class. This is confirmed when looking at the model's confusion matrix Fig. 11. We can observe that the model's predictions are mostly in the diagonal or around it, confirming our statement. This is also the case for the other scenarios (Fig. 12 and 13). While FCNN's predictions are spread out among all the classes without any particular pattern, GraphSAGE's predictions are concentrated around the diagonal. This also entails that

Table 8
SymNet FCNN vs GraphSAGE test accuracy per class.

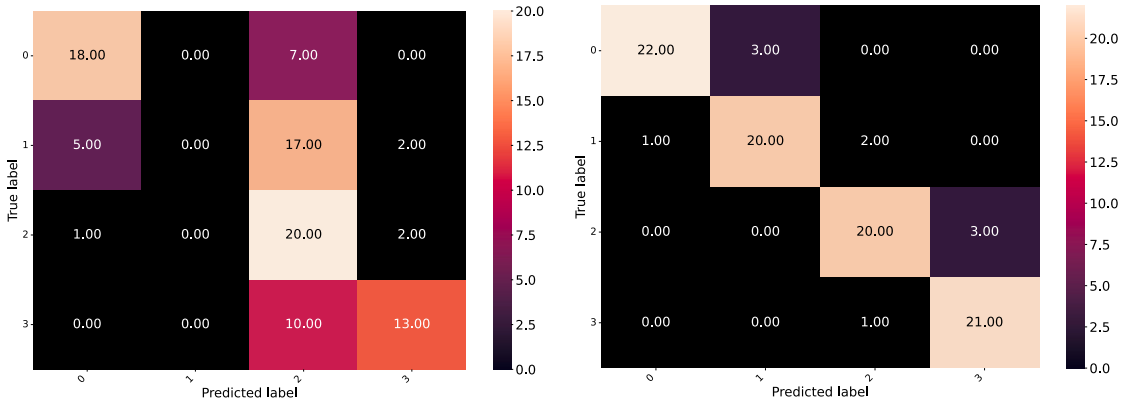
Scenario	Model	Class 0	Class 1	Class 2	Class 3
SymNet_75	FCNN	0.704	0	0.828	0.551
	GraphSAGE	**0.847**	**0.811**	**0.839**	**0.91**
SymNet_40	FCNN	0.732	0.587	0.173	0.616
	GraphSAGE	**0.841**	**0.696**	**0.727**	**0.895**
SymNet_10	FCNN	**0.74**	0	0	**1**
	GraphSAGE	0.755	**0.524**	**0.672**	0.832

GraphSAGE is capturing the characteristics that make up each class, thus being able to closely rank each node. These results improve as the non-test data used by the model increases, reaching 0.802 MCC, 85% accuracy, and 0.482 *Class RMSE* in SymNet_75.

7.2. EurNet

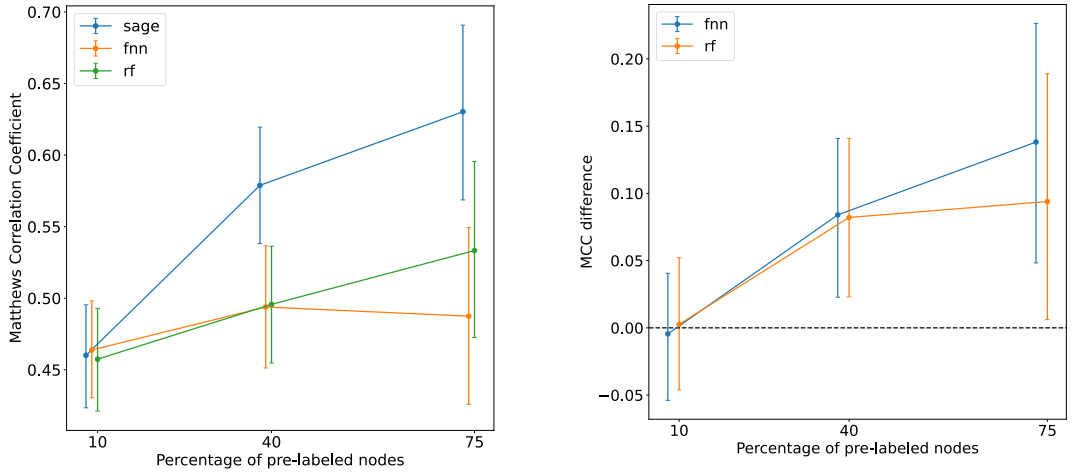
The baseline models' test results are summarized in Table 9. It can be observed that Random Forest achieves the best baseline results for all scenarios, being closely followed by the FCNN.

Given the superior performance of GraphSAGE on SymNet, we have only trained this GNN model on EurNet. The test results are summarized in Table 10. When looking at the per-class accuracy (Table 11), the baseline concentrates on lower classes while doing poorly on higher classes. GraphSAGE instead has a more uniform performance across



(a) FCNN (b) GraphSAGE

Fig. 13. Normalized confusion matrices for SymNet_75 (percentage of total).



(a) MCC confidence interval

(b) Confidence interval of the MCC improvement between GraphSAGE and baseline

Fig. 14. GraphSAGE achieves statistically significant MCC improvements over the traditional Machine Learning baseline on EurNet scenarios with higher amounts of training data.

Table 9
EurNet baseline test metrics.

Scenario	Metric	LR	RF	KNN	FCNN
EurNet_75	MCC	0.332	**0.525**	0.377	0.494
	Accuracy	0.474	**0.643**	0.529	0.612
	Class RMSE	1.223	0.953	1.182	**0.944**
EurNet_40	MCC	0.365	**0.495**	0.361	0.482
	Accuracy	0.507	**0.617**	0.518	0.607
	Class RMSE	1.188	**0.932**	1.155	0.967
EurNet_10	MCC	0.272	**0.462**	0.276	0.443
	Accuracy	0.429	**0.593**	0.45	0.573
	Class RMSE	1.228	**0.979**	1.303	0.982

Table 10
EurNet baseline vs GraphSAGE test metrics.

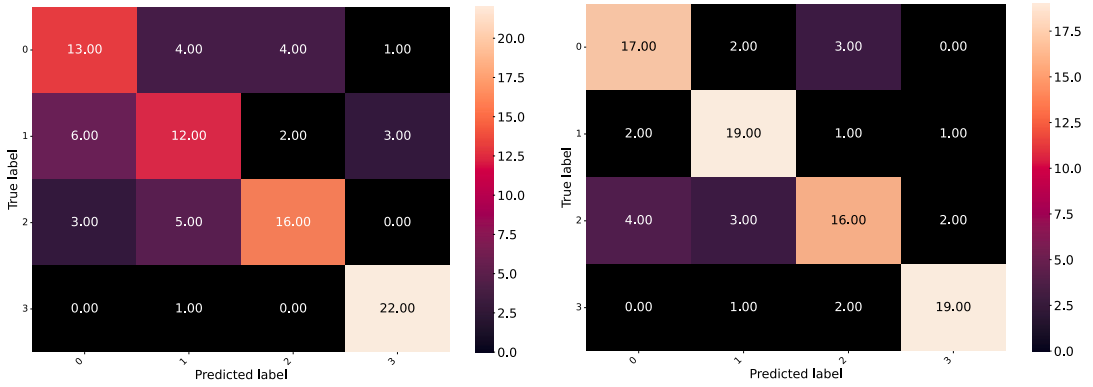
Scenario	Metric	Best Baseline	GraphSAGE
EurNet_75	MCC	0.525	**0.636**
	Accuracy	0.643	**0.726**
	Class RMSE	0.944	**0.803**
EurNet_40	MCC	0.495	**0.571**
	Accuracy	0.617	**0.678**
	Class RMSE	**0.932**	0.973
EurNet_10	MCC	0.462	**0.509**
	Accuracy	0.593	**0.63**
	Class RMSE	**0.979**	1.123

classes which showing its greater capacity to adjust to more complex patterns.

GraphSAGE demonstrated statistically significant performance gains with higher amounts of training data over the traditional Machine Learning baseline on EurNet. Fig. 14 presents the MCC bootstrap confidence intervals for GraphSAGE, RF, and FCNN, and the difference between GraphSAGE and the baseline models. Except for the 10% scenario, the MCC for GraphSAGE is higher than the MCC for the baseline models, showing that GraphSAGE models better the systemic risk metric than the baseline models with sufficient training data. With 40% of

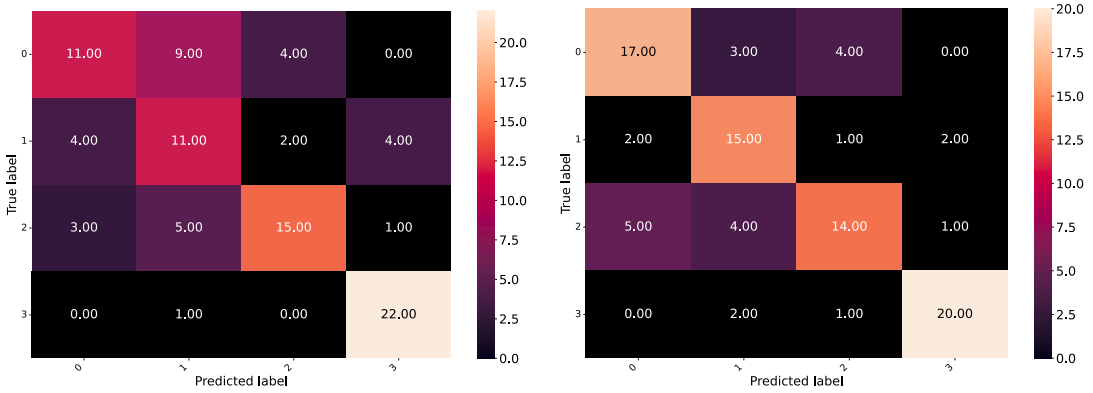
Table 11
EurNet baseline vs GraphSAGE test accuracy per class.

Scenario	Model	Class 0	Class 1	Class 2	Class 3
EurNet_75	Best Baseline	0.563	0.506	**0.619**	**0.92**
	GraphSAGE	**0.724**	**0.758**	0.6	0.816
EurNet_40	Best Baseline	0.426	0.531	**0.606**	**0.912**
	GraphSAGE	**0.665**	**0.704**	0.553	0.819
EurNet_10	Best Baseline	0.492	0.533	0.457	**0.871**
	GraphSAGE	**0.652**	**0.562**	**0.518**	0.643



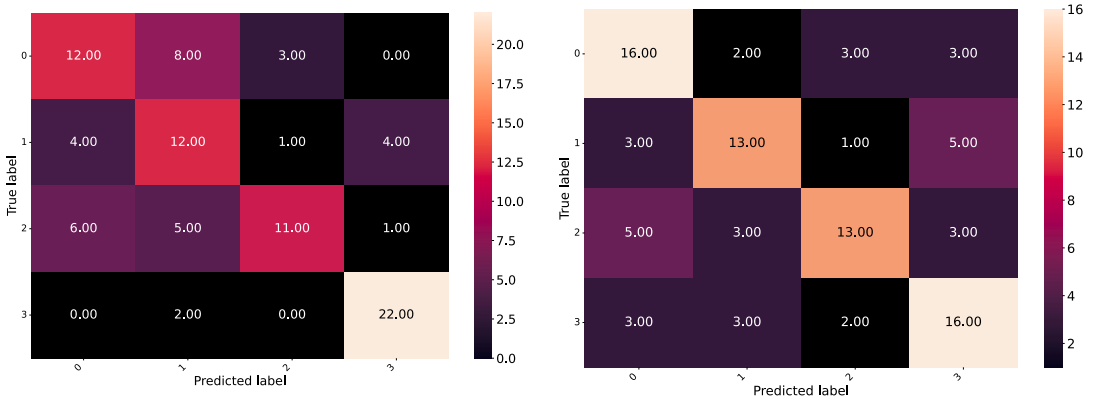
(a) Random Forest (b) GraphSAGE

Fig. 15. Normalized confusion matrices for SymNet_75 (percentage of total).



(a) Random Forest (b) GraphSAGE

Fig. 16. Normalized confusion matrices for SymNet_40 (percentage of total).



(a) FCNN (b) GraphSAGE

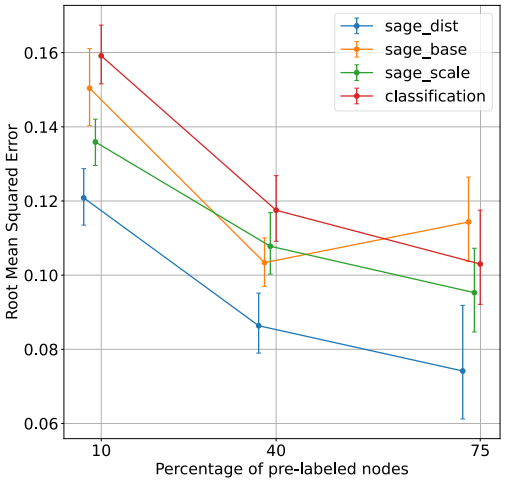
Fig. 17. Normalized confusion matrices for SymNet_10 (percentage of total).

pre-labeled nodes, the improvement in performance is 0.082 (95% CI 0.023, 0.141), which increases with the amount of data used for training, reaching 0.094 (95% CI 0.006, 0.189) with 75% of pre-labeled nodes.

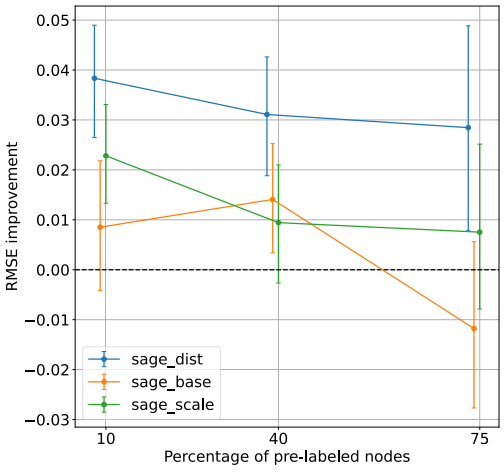
The confusion matrices for each of the scenarios are presented in Fig. 15, 16, and 17. It can be observed that both FCNN and GraphSAGE perform worst on EurNet than on SymNet. However, when a higher amount of training data is available, GraphSAGE performs better, being closer to the desired diagonal shape as can be seen by the higher percentages of predictions on the diagonal. The models are not completely able to correctly capture the characteristics used to rank the nodes. This can be partially due to the target ranking used, since, as we saw in Sec. 6.2.1, the target attribute is concentrated around two values.

7.3. Class to regression

Table 12 summarizes the results obtained and the *equal-mean* approach (Sec. 5.5). It can be observed that GraphSAGE achieves the best results in all metrics. GraphSAGE, with 75% / 40% / 10% of the nodes, pre-labeled in 4 classes, is able to predict the quantile of the other 25% / 60% / 90% of the nodes with a mean absolute error of 0.053 / 0.063



(a) RMSE confidence interval



(b) Confidence interval of the RMSE improvement between C2R approaches and classification

Fig. 18. C2R *equal-mean* achieves statistically significant RMSE improvements over the classification models on SymNet.

Table 12
SymNet C2R: test metrics using *equal-distribution* approach.

Scenario	Metric	FCNN	GCN	GraphSAGE	GAT
SymNet_75	RMSE	0.188	0.211	**0.076**	0.162
	MAE	0.159	0.175	**0.053**	0.124
	MCC	0.311	0.283	**0.784**	0.439
SymNet_40	RMSE	0.196	0.213	**0.089**	0.173
	MAE	0.164	0.177	**0.063**	0.137
	MCC	0.293	0.299	**0.773**	0.387
SymNet_10	RMSE	0.212	0.215	**0.122**	0.207
	MAE	0.175	0.18	**0.084**	0.163
	MCC	0.259	0.331	**0.68**	0.304

Table 13
SymNet C2R: test MAE for each true class.

Scenario	Model	Class 0	Class 1	Class 2	Class 3	Global
SymNet_75	GraphSAGE_dist	**0.0551**	**0.0546**	**0.0539**	**0.0492**	**0.053**
	GraphSAGE_scale	0.0909	0.0981	0.0687	0.0518	0.078
	GraphSAGE_base	0.0665	0.0774	0.0623	0.1232	0.082
SymNet_40	GraphSAGE_dist	0.0687	**0.0586**	**0.0597**	0.0654	**0.063**
	GraphSAGE_scale	0.0757	0.1013	0.1015	**0.0565**	0.084
	GraphSAGE_base	**0.0676**	0.0866	0.0921	0.1684	0.103
SymNet_10	GraphSAGE_dist	0.0841	**0.0813**	**0.0821**	0.0877	**0.084**
	GraphSAGE_scale	0.0885	0.1163	0.1121	**0.0742**	0.098
	GraphSAGE_base	**0.0832**	0.1137	0.094	0.1456	0.109

/ 0.084 quantile points. Although we have decreased the training data available by 85%, the model only performs 58% worst.

7.3.1. Approaches comparison

Table 13 presents the results obtained by GraphSAGE models using different approaches (*base_n*, *equal-scale*, and *equal-mean*) for each class. *Equal-mean* approach achieves the best overall results, followed by *equal-scale* approach. The *base_n* approach obtains significantly worst results for class 3, which confirms the issue we stated in Sec. 5.3.

7.3.2. C2R vs classification

To compare the classification models obtained in Sec. 6.4 with C2R, the predicted labels must be converted to quantiles. This is accomplished by converting all the samples that belong to a class to the mean of the corresponding quantile range, to minimize the RMSE, or to its median, to minimize the MAE.[8] Since the quantiles are uniformly distributed, the mean and the median are the same, the middle-value class interval. Thus, for comparison, we will consider the classification model to be predicting the middle value of the quantile range of the predicted class.

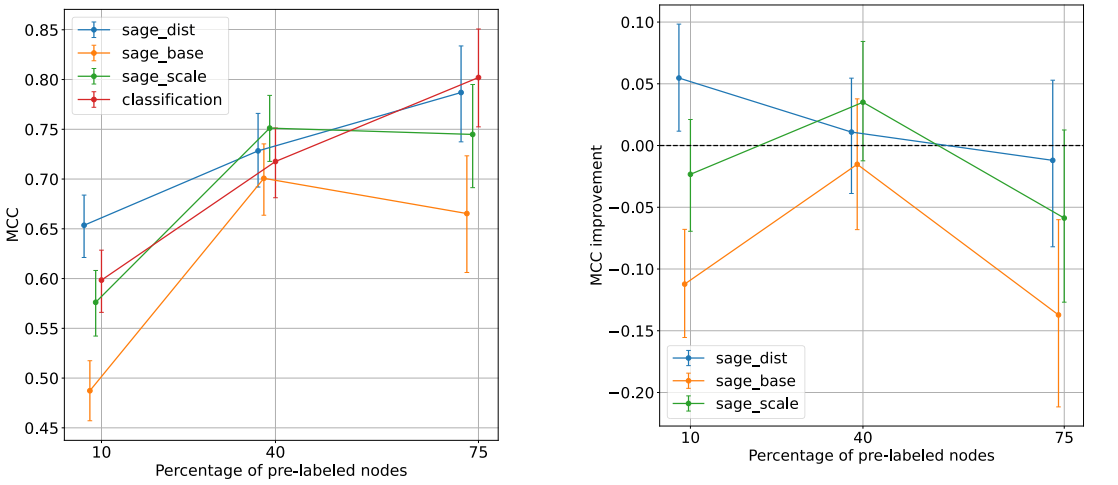
Equal-mean C2R demonstrated statistically significant performance gains over the classification GNN models on SymNet. This improvement is higher for lower percentages of pre-labeled nodes. Fig. 18 presents the RMSE bootstrap confidence intervals for different C2R approaches and the best classification model. It also presents the difference between each of the approaches and the classification model. The RMSE for *equal-mean* is consistently higher, showing that training with the *equal-mean* approach achieves more precise and granular predictions than standard classification for discrete ordinal systemic risk metrics. With 10% of pre-labeled nodes, the improvement in performance is 0.038 (95% CI 0.026, 0.049), which decreases with the amount of data used for training, reaching 0.028 (95% CI 0.008, 0.049) with 75% of pre-labeled nodes.

Even though *Equal-mean* C2R focuses on minimizing the RMSE of the predicted quantile, it does not achieve a statistically significant reduction on its classification MCC performance. Moreover, it achieves a statistically significant improvement in the classification MCC with respect to the classification models when only 10% of pre-labeled data as shown in Fig. 19 (0.055 improvement with 95% CI 0.012, 0.098).

The comparison between the best classification model obtained in Sec. 7.1 and the best C2R model is presented in Table 14. Overall the C2R models obtain better results, mostly when the amount of data available is lower even surpassing the classification models' results in the classification metrics.

Moreover, the application of a classification model to the regression task is limited. Considering a classification model with 100% accuracy, it would provide a quantile range of length 25 to which the node belongs, and thus obtain an MAE of approximately 6.25 percentile points. This is the minimum MAE that our classification models could achieve. In contrast, our model obtains an MAE of 5.3 / 6.3 / 8.4 (SymNet_75 / SymNet_40 / SymNet_10) percentile points. Therefore, our C2R model is even capable of determining the importance of a node in the SymNet_75 scenario more precisely than a perfect 4-class classification model.

[8] The mean minimizes the mean of the squared deviations (MSE), and the median minimizes the mean of the absolute deviations (MAE).



(a) MCC confidence interval

(b) Confidence interval of the MCC improvement between C2R approaches and classification

Fig. 19. C2R *equal-mean* achieves on SymNet statistically significant MCC improvements over the classification models with 10%.

Table 14
SymNet: classification vs C2R best models.

Scenario	Metric	Classification	C2R
SymNet_75	RMSE	0.12	**0.076**
	MAE	0.086	**0.053**
	MCC	**0.802**	0.784
	Accuracy	**0.851**	0.837
SymNet_40	RMSE	0.112	**0.089**
	MAE	0.084	**0.063**
	MCC	0.764	**0.773**
	Accuracy	0.822	**0.829**
SymNet_10	RMSE	0.153	**0.122**
	MAE	0.112	**0.084**
	MCC	0.603	**0.68**
	Accuracy	0.701	**0.759**

7.4. Overall results

For the baselines, Random Forest and FCNN have achieved the best results overall. However, they have been surpassed by the GNN models, with GraphSAGE being the model that achieved the best results. GNNs have shown statistically significant improvements (5% significance) on the MCC for all scenarios except for EurNet 10%, meaning that they outperform traditional Machine Learning even when fewer training data is available. Not only GNNs achieve better classification metrics, but overall also have a better RMSE, which shows they are learning better ranking functions for the systemic risk metric.

C2R models outperform the classification models in regression metrics, achieving statistically significant improvements with 5% significance. This shows that they learn a better approximation to the true systemic risk metric, providing more granular and precise predictions. Moreover, although being trained for quantile regression, C2R models do not show a statistically significant deterioration in classification MCC performance, even achieving a statistically significant improvement with 10% of training data. Overall, C2R models have been more robust to data scarcity.

From our experiments, we can confirm that GNN models are better suited for systemic risk prediction on financial networks and should be preferred over traditional Machine Learning. The results obtained also confirm the fact that the network structure and the features of the relations (edge features) hold useful information for our task. In other words, our ranking does not solely depend on the node's features, but also on the relations between the nodes. In this sense, we can interpret the difference between the baseline models' and the GNN's results as the importance of the network structure and its relations over the target labels compared to the node's features. Therefore, when the baseline models achieve similar results to the GNNs, such as on EurNet (Sec. 7.2), it can be interpreted that the node features' importance overshadows the importance of the network. In contrast, when this difference is greater, such as on SymNet (Sec. 7.1), it can be interpreted that the network structure has greater importance on the labels.

8. Conclusion and future work

8.1. Conclusion

We have applied Graph Neural Networks (GNNs) to analyze financial networks. In contrast with network metrics and traditional Machine Learning, GNNs are flexible models capable of using both the network structure and the features of the nodes and edges to perform edge-level, node-level, or graph-level tasks on large networks. To compare the effectiveness of GNNs against traditional Machine Learning techniques (not graph oriented), we have tackled a task in which the objective is to classify the nodes of a network by their systemic importance with respect to the rest of the nodes. This classification has been presented as complex and can not be automated. Therefore, it is solved by pre-labeling some nodes into 4 classes and using a model to classify the rest of the nodes. We have run this task in two differently distributed networks and three scenarios, with 75%, 40%, and 10% of pre-labeled data (see Table 3). The two financial networks used are generated using network reconstruction on aggregated data from European banks (EurNet, Sec. 6.2.1) and synthetic data (SymNet, Sec. 6.2).

The GNNs, specially GraphSAGE, surpass the traditional Machine Learning techniques in dealing with this task, obtaining better results (see Table 7 and 10). While traditional Machine Learning techniques have obtained MCC (Matthew's correlation coefficient) 0.41/0.36/0.34 in SymNet and 0.53/0.5/0.46 in EurNet (75%, 40%, and 10% scenarios), GNNs have obtained MCC 0.8/0.76/0.6 in SymNet and 0.64/0.57/0.51 in EurNet. This results in an average increase in MCC of 0.35 (94% average percentage increase) for SymNet and 0.08 (15%) for EurNet. Thus, GNNs have verified their superior capabilities to deal with network data. These improvements suggest the relevance of the interconnections among nodes relative to node feature information.

In Sec. 5, we present an approach that allows using data labeled into a small number of classes (discrete input) to train models to predict a regression output. This approach uses a simple pre-labeling (using

4 classes) to obtain granular and more precise predictions. The only requirements are that the input classes have an order and that each class is associated with a subinterval of the output.

We have applied this approach to the above-stated problem to obtain each node's systemic importance percentiles, achieving a better approximation of the true importance of the nodes (see Table 14). While the classification models provide a percentile ($[0, 100]$ range) MAE of 8.6/8.4/11.2 (SymNet_75 / SymNet_40 / SymNet_10), the classes to regression (C2R) models improved these results achieving 5.3/6.3/8.4. In sum, C2R models obtained an MAE reduction of 3 percentile points (30% average percentage reduction). Moreover, a perfect classification model would provide a percentile MAE of approximately 6.25 percentile points, the minimum MAE our classification models could achieve. Therefore, our C2R model has determined the importance of a node in the SymNet_75 scenario more precisely than a perfect 4-class classification model.

Graph Neural Networks are flexible architectures that can be applied to many tasks and that combine two important analytical tools in this area, network theory and Machine Learning. We expect that this work will open up the exploration of the use of Graph Neural Networks to analyze systemic risks and, more generally, solve tasks in financial networks, bringing the improvements made in Deep Learning to the analysis of financial networks.

8.2. Future work

The application of GNNs to financial networks is still in its initial stages. This section provides an overview of some areas of research aimed at increasing their use, applicability, and capabilities in the analysis of financial networks.

Reduced Data Requirement. This area would consist in improving the flexibility of the applicability of GNN models to financial networks, making sure they use all the available information but remain flexible to shortages. In particular, the models in this work make use of the whole financial network. This requirement could be too strict for its applicability. It would be beneficial to test its usefulness with a small neighborhood of each node, using only nodes at a distance k. This allows to greatly reduce the data required to train the model and to make predictions, since only a small neighborhood of the node of interest is used. On this same note, it would be interesting to check the GNNs' effectiveness without edge weights and how to better make use of edge features.

Adding information through network construction. GNNs enable including additional information into the model through the network's construction. It would be useful to study the situation when the graph information is not available. In this situation, instead of using traditional Machine Learning, the network can be constructed through additional information, the use of techniques in the literature, or expert knowledge. Even though the real network structure is unknown, the GNNs might be able to outperform other techniques if the network estimation procedure is sufficiently good. Some techniques that can be tested can be found in the literature, such as estimating the financial network based on text co-mentions (Rönnqvist & Sarlin, 2015, 2014) or based on a combination of financial data and financial tweets (Cerchiello et al., 2017).

Regression from Classes. This area would consist on expanding Sec. 5 capabilities.

- Predicting more complex distributions (apart from the uniform distribution used for quantiles). For example, predicting directly a systemic risk score from the classes, instead of the quantiles, would increase its usefulness.
- Test the utility of the extensions mentioned in Sec. 5.6, especially losses based on distribution similarity, which can produce good results in complex scenarios.

Other Datasets and Tasks. This area consists in testing the GNN's effectiveness in other synthetic or real datasets, and/or different tasks. By doing so, we can show consistent overall results when compared to traditional techniques. Concretely, one interesting line of further research consists of applying GNNs to predict other already existing systemic risk measures such as CoVaR. For instance, training a GNN in a similar way as (Keilbar & Wang, 2022) do for a neural network. This model could then be used to estimate the CoVaR and use it as a risk measure in the tool proposed by said paper, replicating their results and testing whether using GNNs improves them. We would also like to apply GNNs in the context of the paper by (Niu et al., 2021), who propose a practical visual analytical approach to support the exploration and evaluation of regulatory intervention measurements to mitigate systemic risk in financial networks, such as removing nodes (entities) of interest, partitioning the network, replacing the node with specified one, and many others. When a regulatory intervention measure is taken, the authors carry out the contagion process with the new network to analyze the effects of the measure taken. However, removing any entity (node) will most likely cause the rest of the entities (nodes) to change their interconnections (if an entity can't lend with one bank, it will try another). We could train a GNN on the initial network and, after taking the intervention measurement and removing a connection, run the model to predict if there will be a connection between two nodes. If the model is capable of modeling the patterns of connections, it can be used to predict the changes in the connections, providing more realistic simulations.

CRediT authorship contribution statement

Vicente Balmaseda: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **María Coronado:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Gonzalo de Cadenas-Santiago:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data is available at https://github.com/vibalcam/gnn-systemic-risk. More details on data can be found on the Supplementary Material File: Appendix A. Data Appendix.

Appendix A. Glossary

Table A.1 provides a list of the abbreviations used in this work.

Appendix B. More details on experiments

B.1. Network estimation

Table B.1 provides the inputs for maximum entropy and minimum entropy network estimation.

Table A.1
List of abbreviations.

GNN	Graph Neural Network
MCC	Matthews Correlation Coefficient
ML	Machine Learning
RF	Random Forest
LR	Logistic Regression
KNN	K-Nearest Neighbors
GAT	Graph Attention Network
GCN	Graph Convolutional Network
FCNN	Fully Connected Neural Network
FNN	Feedforward Neural Network
C2R	Class to regression
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
ReLU	Rectified Linear Unit
Sage	GraphSAGE

Table B.1
Inputs of maximum entropy and minimum density network estimation.

Inter-node assets	Net loans to other entities in the network
Inter-node liabilities	Total deposits from entities in the network
Buffer	Amount of losses an entity can withstand before it defaults
Weights	Relevance of the entity

Table B.2
FCNN hyper-parameters.

Hyper-parameter	Description
in_features	input feature size
h_features	list of hidden feature sizes
out_features	out feature size
activation	activation function applied to the updated node features
norm_nodes	normalization applied to the node features
dropout	dropout rate applied to intermediate layers

Table B.3
GCN hyper-parameters.

Hyper-parameter	Description
in_features	input feature size
h_features	list of hidden feature size
out_features	out feature size
activation	activation function applied to the updated node features
norm_edges	applies normalization to the edge weights (EdgeWeightNorm)
norm_nodes	normalization applied to the node features
dropout	dropout rate applied to intermediate layers

B.2. Models

The descriptions of each of the model's hyper-parameters are presented in Table B.2 for FCNN, Table B.3 for GCN, Table B.4 for GraphSAGE, and Table B.5 for GAT.

Classification. The classification models output a vector of size n, being n the number of classes. These n outputs are the result of passing the output of the last layer through a $softmax(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_{j=0}^{n-1} e^{x_j}}$ function. Index i corresponds to the probability of the node belonging to class i.

Classes to Regression. The C2R models output a single value, which is obtained by applying a $sigmoid(x) = \frac{1}{1+e^{-x}}$ function after the last layer. This value is the predicted quantile of the node and can be transformed to a class prediction using Equation (2).

B.3. Training

We have used the scikit-learn library for the traditional Machine Learning models, and the DGL library with PyTorch backend to develop and train the GNN models. The models have been trained on an RTX 2070 GPU. A random seed has been used to ensure reproducible results to the extent possible. However, fully deterministic behavior cannot be ensured due to the unavailability in DGL and PyTorch of deterministic algorithms for training on the GPU.

Table B.4
GraphSAGE hyper-parameters.

Hyper-parameter	Description
in_features	input feature size
h_features	list of hidden feature size
out_features	out feature size
activation	activation function applied to the updated node features
norm_edges	applies normalization to the edge weights (EdgeWeightNorm)
norm_nodes	normalization applied to the node features
aggregator_type	aggregator type to use (mean, gcn, pool, lstm).
feat_drop	dropout rate on features

Table B.5
GAT hyper-parameters.

Hyper-parameter	Description
in_features	input feature size
h_features	list of hidden feature size
out_features	out feature size
num_heads	list of number of heads in Multi-Head Attention
norm_nodes	normalization applied to the node features
activation	activation function applied to the updated node features
negative_slope	LeakyReLU angle of negative slope
attn_drop	dropout rate on attention weights
feat_drop	dropout rate on features
residual	whether to use residual connections

Table B.6
Classification training hyper-parameters.

Hyper-parameter	Description
lr	learning rate for the training
optimizer_name	optimizer used for training
n_epochs	number of epochs of training
scheduler_mode	scheduler mode to use for the learning rate scheduler
label_smoothing	label smoothing applied to CrossEntropyLoss
use_edge_weight	If true, it uses edge weights for training when possible
scheduler_patience	value used as patience for the learning rate scheduler

Table B.7
Percentile regression training hyper-parameters.

Hyper-parameter	Description
lr	learning rate for the training
optimizer_name	optimizer used for training
n_epochs	number of epochs of training
scheduler_mode	scheduler mode to use for the learning rate scheduler
use_edge_weight	If true, it uses edge weights for training when possible
scheduler_patience	value used as patience for the learning rate scheduler
loss_type	regression loss to use (MSE, MAE)
approach	approach used to convert pseudo percentiles into labels

The descriptions of the training hyper-parameters are summarized in Table B.6 and B.7. The optimizer used for training is AdamW (Loshchilov & Hutter, 2019). For the classification task, a cross-entropy loss with $ls \in \{0, 0.2\}$ label smoothing (Szegedy et al., 2015) has been used. For the percentiles regression task, the Mean Squared Error (MSE) loss has been used. The training process incorporates a learning rate scheduler that reduces the learning rate when the *scheduler_mode* metric does not improve for *scheduler_patience* epochs. The metric followed for classification is the Matthews Correlation Coefficient (MCC) on the validation set. The models have been trained for 100 epochs, except GraphSAGE on EurNet_75 and EurNet_40 which have trained for 2000 epochs, and GraphSAGE on EurNet_10 which has trained for 1000 epochs.

Table B.8
Dataset hyper-parameters.

Hyper-parameter	Description
drop_edges	percentage of edges to remove. Value in [0,1]
add_self_loop	If true, it adds non duplicated self-loops
sets_lengths	tuple with percentage of train, validation and test samples
target	column to use as target for label calculation
node_attributes	list of names of the columns to use as node features
num_classes	number of classes, n-tiles which the labels will represent

The descriptions of the dataset's hyper-parameters are presented in Table B.8. We incorporated self-loops into the graphs during training and testing and used DropEdge values of $\{0, 0.2\}$.

Appendices. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.iswa.2023.200240.

References

Acemoglu, D., Ozdaglar, A., & Tahbaz-Salehi, A. (2015). Systemic risk and stability in financial networks. *American Economic Review*, *105*(2), 564–608.

Acharya, V. V., Pedersen, L. H., Philippon, T., & Richardson, M. (2017). Measuring systemic risk. *The Review of Financial Studies*, *30*(1), 2–47.

Adrian, T., & Brunnermeier, M. K. (2016). CoVaR. *American Economic Review*, *106*(7), 1705–1741.

Anand, K., Craig, B., & von Peter, G. (2015). Filling in the blanks: Network structure and interbank contagion. *Quantitative Finance*, *15*(4), 625–636.

Bardoscia, M., Barucca, P., Battiston, S., Caccioli, F., Cimini, G., Garlaschelli, D., Saracco, F., Squartini, T., & Caldarelli, G. (2021). The physics of financial networks. *Nature Reviews Physics*, *3*(7), 490–507.

Bardoscia, M., Battiston, S., Caccioli, F., & Caldarelli, G. (2015). DebtRank: A microscopic foundation for shock propagation. *PLoS ONE*, *10*(6), 1–13.

Battiston, S., Puliga, M., Kaushik, R., Tasca, P., & Caldarelli, G. (2012). DebtRank: Ooo central to fail? Financial networks, the FED and systemic risk. *Scientific Reports*, *2*, 1–6.

Billio, M., Costola, M., Panzica, R., & Pelizzon, L. (2017). Systemic risk and financial interconnectedness: Network measures and the impact of the indirect effect. In M. Billio, et al. (Eds.), *Systemic risk tomography: Signals, measurement and transmission channels* (pp. 43–72).

Brownlees, C., & Engle, R. F. (2017). SRISK: A conditional capital shortfall measure of systemic risk. *The Review of Financial Studies*, *30*(1), 48–79.

Cai, T., Luo, S., Xu, K., He, D., Liu, T.-Y., & Wang, L. (2020). GraphNorm: A principled approach to accelerating graph neural network training.

Cerchiello, P., & Giudici, P. (2016). Conditional graphical models for systemic risk estimation. *Expert Systems with Applications*, *43*, 165–174.

Cerchiello, P., Giudici, P., & Nicola, G. (2017). Twitter data models for bank risk contagion. *Neurocomputing*, *264*, 50–56.

Cheng, D., Wang, X., Zhang, Y., & Zhang, L. (2020). Risk guarantee prediction in networked-loans. In C. Bessiere (Ed.), *Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI-20* (pp. 4483–4489). International Joint Conferences on Artificial Intelligence Organization. Special Track on AI in FinTech.

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*, *21*(1), 1–13.

Ding, R., & Uryasev, S. (2020). CoCDaR and mCoCDaR: New approach for measurement of systemic risk contributions. *Journal of Risk and Financial Management*, *13*(11), 270.

Döpke, J., Fritsche, U., & Pierdzioch, C. (2017). Predicting recessions with boosted regression trees. *International Journal of Forecasting*, *33*(4), 745–759.

Giglio, S., Kelly, B., & Pruitt, S. (2016). Systemic risk and the macroeconomy: An empirical evaluation. *Journal of Financial Economics*, *119*(3), 457–471.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. Retrieved from arXiv:1704.01212.

Giudici, P., Sarlin, P., & Spelta, A. (2020). The interconnected nature of financial systems: Direct and common exposures. *Journal of Banking & Finance*, *112*.

Giudici, P., & Spelta, A. (2016). Graphical network models for international financial flows. *Journal of Business and Economic Statistics*, *34*(1), 128–138.

Haldane, A. G. (2009). Rethinking the financial network. In *Bank of England. Speech given at the financial student association, Amsterdam*.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems, 2017-Decem(Nips)* (pp. 1025–1035).

Hautsch, N., Schaumburg, J., & Schienle, M. (2014). Forecasting systemic impact in financial networks. *International Journal of Forecasting*, *30*(3), 781–794.

Huang, W. Q., & Uryasev, S. (2018). The CoCVaR approach: Systemic risk contribution measurement. *The Journal of Risk*, *20*(4), 75–93.

Härdle, Karl, W., Wang, W., & Yu, L. (2016). TENET: Tail-event driven NETwork risk. *Journal of Econometrics*, *192*(2), 499–513.

Kanno, M. (2015). Assessing systemic risk using interbank exposures in the global banking system. *Journal of Financial Stability*, *20*, 105–130.

Keilbar, G., & Wang, W. (2022). Modelling systemic risk using neural network quantile regression. *Empirical Economics*, *62*(1), 93–118.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th international conference on learning representations, ICLR 2017 - conference track proceedings* (pp. 1–14).

Kou, G., Chao, X., Peng, Y., Alsaadi, F. E., & Herrera-Viedma, E. (2019). Machine learning methods for systemic risk analysis in financial sectors. *Technological and Economic Development of Economy*, *25*(5), 716–742.

Leventides, J., Loukaki, K., & Papavassiliou, V. G. (2019). Simulating financial contagion dynamics in random interbank networks. *Journal of Economic Behavior & Organization*, *158*, 500–525.

Li, Y., Shou, J., Treleaven, P., & Wang, J. (2021). Graph neural network for merger and acquisition prediction.

Li, S., Wang, M., & He, J. (2013). Prediction of banking systemic risk based on support vector machine. *Mathematical Problems in Engineering*, *2013*, 1–5.

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. In *7th international conference on learning representations, ICLR 2019*.

Niu, Z., Wu, J., Cheng, D., & Zhang, J. (2021). Regshock: Interactive visual analytics of systemic risk in financial networks. Retrieved from arXiv:2104.11863.

Rong, Y., Huang, W., Xu, T., & Huang, J. (2019). DropEdge: Towards deep graph convolutional networks on node classification, pp. 1–18.

Rönnqvist, S., & Sarlin, P. (2014). From text to bank interrelation maps. In *IEEE/IAFE conference on computational intelligence for financial engineering, proceedings (CIFEr)* (pp. 48–54).

Rönnqvist, S., & Sarlin, P. (2015). Bank networks from text: Interrelations, centrality and determinants. *Quantitative Finance*, *15*(10), 1619–1635.

Rudolph, M. J. (2017). Reviewing systemic risk within the insurance industry. Society of Actuaries, Technical report.

Szegedy, C., Vanhoucke, V., Ioffe, S., & Shlens, J. (2015). Rethinking the inception srchitecture for computer vision.

Upper, C., & Worms, A. (2004). Estimating bilateral exposures in the German interbank market: Is there a danger of contagion? *European Economic Review*, *48*(4), 827–849.

Veličković, P., Casanova, A., Liò, P., Cucurull, G., Romero, A., & Bengio, Y. (2018). Graph attention networks. In *6th international conference on learning representations, ICLR 2018 - conference track proceedings* (pp. 1–12).

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24.

Yu, J., & Zhao, J. (2020). Prediction of systemic risk contagion based on a dynamic complex network model using machine learning algorithm. *Complexity*, *2020*.

Zhao, K., Xu, H., Cheng, Y., Li, X., & Gao, K. (2021). Representation iterative fusion based on heterogeneous graph neural network for joint entity and relation extraction. *Knowledge-Based Systems*, *219*, Article 106888.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57–81.