Attention-based dynamic multilayer graph neural networks for loan default prediction

Sahab Zandi[a,*], Kamesh Korangi[b,c], María Óskarsdóttir[d], Christophe Mues[b,c], Cristián Bravo[a]

[a]*Department of Statistical and Actuarial Sciences, Western University, 1151 Richmond Street, London, Ontario, N6A 5B7, Canada.*
[b]*Department of Decision Analytics and Risk, Southampton Business School, University of Southampton, University Road, SO17 1BJ, United Kingdom.*
[c]*Centre for Operational Research, Management Sciences and Information Systems, University of Southampton, University Road, SO17 1BJ, United Kingdom.*
[d]*Department of Computer Science, Reykjavík University, Menntavegur 1, 102 Reykjavík, Iceland.*

Abstract

Whereas traditional credit scoring tends to employ only individual borrower- or loan-level predictors, it has been acknowledged for some time that connections between borrowers may result in default risk propagating over a network. In this paper, we present a model for credit risk assessment leveraging a dynamic multilayer network built from a Graph Neural Network and a Recurrent Neural Network, each layer reflecting a different source of network connection. We test our methodology in a behavioural credit scoring context using a dataset provided by U.S. mortgage financier Freddie Mac, in which different types of connections arise from the geographical location of the borrower and their choice of mortgage provider. The proposed model considers both types of connections and the evolution of these connections over time. We enhance the model by using a custom attention mechanism that weights the different time snapshots according to their importance. After testing multiple configurations, a model with GAT, LSTM, and the attention mechanism provides the best results. Empirical results demonstrate that, when it comes to predicting probability of default for the borrowers, our proposed model brings both better results and novel insights for the analysis of the importance of connections and timestamps, compared to traditional methods.

Keywords: OR in Banking, Credit Scoring, Dynamic Multilayer Networks, Graph Neural Networks, Recurrent Neural Networks

1. Introduction

Network science provides a beneficial tool to study complex systems of interacting entities that can be found in many areas, such as biology, finance and economics (Barabási & Pósfai, 2016). To represent

[*]Corresponding author

Email addresses: `szandi@uwo.ca` (Sahab Zandi), `k.korangi@soton.ac.uk` (Kamesh Korangi), `mariaoskars@ru.is` (María Óskarsdóttir), `c.mues@soton.ac.uk` (Christophe Mues), `cbravoro@uwo.ca` (Cristián Bravo)

connections between these entities, graphs are a common representation method, which have diverse applications in social network analysis (Haythornthwaite, 1996), computational finance (Wang et al., 2022), and recommender systems (Wang et al., 2021), among many others. Graph Neural Networks (GNNs) are models in the field of deep learning, specifically tailored to perform over graph domains. They have been utilized for different tasks, including node classification (Tang et al., 2021), edge prediction (Zhang & Chen, 2018), and graph clustering (Tsitsulin et al., 2020). In most cases, they have been used with static single layer networks, in which nodes are linked based on one source of connection, and the network remains unchanged over time. In reality, though, nodes could be connected by more than one source of connection, as is the case in our application setting. Such networks are generally called multilayer networks (Kivelä et al., 2014). Furthermore, the nodes and edges in a graph may evolve. For example, new nodes may appear, node features may change, and a new relation may emerge between two nodes. Being able to capture these changes in the models could lead to higher predictive performance for problems that are characterized by such dynamic graphs.

To enable dynamic graph learning, we consider Recurrent Neural Networks (RNNs) (Elman, 1990). RNNs are used for data that is presented in a sequence, such as time series data or natural language. Their main objective is to create a representation of a series of inputs, usually indexed by time, to predict an output. Because of their powerful learning capacity, they have been applied successfully in various types of tasks, including speech recognition (Graves et al., 2007), acoustic modelling (Qu et al., 2017), trajectory prediction (Altché & de La Fortelle, 2017), sentence embedding (Palangi et al., 2016), and correlation analysis (Mallinar & Rosset, 2018). Since dynamic graphs represented by discrete snapshots can be considered sequence data, RNNs provide a solution to capture the evolution of these graphs. However, it is known that attention-based RNNs outperform encoder-decoder-based RNNs, indicating that the incorporation of attention can improve the prediction performance (Aliabadi et al., 2020).

The application area that we focus on in this paper is credit scoring — one of the prominent applications of data analytics. Lenders build credit scoring models to help adjudge the risk involved in granting a loan and decide on the terms of the loan and the interest rate (Thomas et al., 2017). Traditional credit scoring models use loan- or borrower-level data to assess a loan applicant's default risk, thus treating borrowers as independent entities. While potential default correlation between borrowers has been acknowledged for some time, it is only more recently that this has started to be further investigated using network science. This is part of a broader trend in which credit scoring research increasingly focuses on improving the performance of existing credit scoring models through the incorporation of machine learning methods, and the inclusion of alternative data sources such as network data (Bravo & Óskarsdóttir, 2020).

In this paper, we propose using an attention-based dynamic multilayer graph neural network to model the problem of credit risk across time and explicitly incorporate default correlation between borrowers. This work makes the following contributions. Firstly, our solution, Dynamic Multilayer Graph Neural Networks (DYMGNN), represents a novel approach for node classification in multilayer networks. Secondly, we show how to apply the proposed method to credit risk modelling, using the example context of mortgage loan default prediction. Finally, we show that, in this setting, our model, by considering dynamicity, multilayer effects, and using an attention mechanism, outperforms other baseline methods.

The structure of this paper is as follows. The next section discusses a selection of previous work on GNN and credit risk modelling. Section 3 explains the methodology, multilayer networks, embeddings, and the models used in this paper. Section 4 sheds light on the data, dynamic networks, and the experiments in the paper. Section 5 presents the experimental results and highlights some discussion points relevant to the models. The final section summarizes our conclusions and suggests future work.

2. Previous work

2.1. Graph neural networks

Graphs can be seen in many real-world applications. In some cases, the graph is static, i.e., the graph structure and node features do not change over time. In other cases, the graph is dynamic, i.e., the graph evolves. GNNs are neural models that capture the dependencies between the nodes within a graph through message passing between the nodes of the graph. A comprehensive survey of methods and applications related to GNNs is provided by Zhou et al. (2020). Recently, some types of GNNs such as the Graph Convolutional Network (GCN) (Kipf & Welling, 2017) and Graph Attention Network (GAT) (Veličković et al., 2018) have been widely used for various deep learning tasks, albeit mostly on static graphs.

GCN is an approach for semi-supervised learning on graph-structured data, utilizing an efficient layer-wise linear model based on a first-order approximation of spectral graph convolutions. To prevent overfitting, it simplifies the convolution calculation by constraining the number of parameters, and by minimizing the number of operations, for example, reducing the matrix multiplications per layer. The number of graph edges is linearly scaled in GCN and this model learns hidden layer representations in which both the local graph structure and node features are encoded. GCN has shown acceptable performance on citation networks and knowledge graphs (Kipf & Welling, 2017).

A second type of GNN, known as GAT, uses an attention mechanism to learn node-level representations (Veličković et al., 2018). The encoder-decoder-based neural machine translation system was outperformed by the attention mechanism in natural language processing (Bahdanau et al., 2015). Nowadays, attention models are widely utilized for document categorization (Pappas & Popescu-Belis, 2017), recommendation systems (Xiao et al., 2017), and the creation of image captions (Xu et al., 2015). The attention mechanism concentrates on a few selected relevant attributes while ignoring other irrelevant attributes (Bahdanau et al., 2015). The categories of attention models that are now in use are global and local attention (Luong et al., 2015), soft and hard attention (Bahdanau et al., 2015), and self-attention (Vaswani et al., 2017). Global attention considers all source positions when deriving the context vector, providing a comprehensive overview of the entire input sequence. In contrast, local attention focuses on a subset of source positions, making it more efficient by narrowing the context to relevant parts. Soft attention generates a weighted sum of the attention scores, allowing gradients to pass through and making the model end-to-end trainable. Hard attention, on the other hand, selects a single source position, making it less computationally expensive but more challenging to optimize due to its non-differentiable nature. Self-attention is an attention mechanism that is applied to compute a representation of a single sequence, enabling the model to weigh the importance of different positions in the sequence when generating its output. In conjunction with RNN or convolutions, self-attention can be used in many applications like learning sentence representations (Lin et al., 2017) and machine reading (Cheng et al., 2016). The GAT is a type of GNN that applies an attention mechanism to graph-structured data, so as to classify nodes. It computes the hidden representation of each node by paying attention to its adjacent nodes and then applying a self-attention strategy. GAT achieved state-of-the-art results in both transductive (semi-supervised) and inductive (supervised) settings.

Most of the GNN models have been proposed for static graph learning; however, over the past few years, several machine learning models capturing the structure and evolution of dynamic graphs have been introduced. A complete review of representation learning approaches for dynamic graphs is given in Kazemi et al. (2020), while a more specialized review of GNN-based approaches for dynamic graphs is provided by Skarding et al. (2021). There are many different approaches to modelling spatial and temporal information in graph-structured data. Diffusion Convolutional Recurrent Neural Network (DCRNN; Li et al., 2018) and Spatio-Temporal Graph Convolutional Networks (STGCN; Yu et al., 2018) analyse graph-structured data first and pass the results to sequence-to-sequence models or RNNs. Structural-RNN collects spatial and temporal data synchronously to associate the graph structure with temporal data so that it can apply RNNs on new graphs (Jain et al., 2016). Dynamic Graph Convolutional Networks (DGCN) propose a novel approach that combines RNNs and GCNs to learn long short-term dependencies together with the graph structure (Manessi et al., 2020). EvolveGCN is an approach for graph representation learning in dynamic graphs that uses an RNN to evolve the parameters of a GCN model. By doing so, EvolveGCN can capture the dynamics of the graph sequence without relying on node embeddings (Pareja et al., 2020). Temporal Graph Convolutional Network (T-GCN) is a novel method for real-time traffic forecasting that uses GCN to learn the complex topological structure of the urban road network for spatial dependence. It also employs RNN to capture the dynamic changes in traffic data for temporal dependence (Zhao et al., 2020). Temporal Graph Attention (TGAT) has been proposed for inductive representation learning on temporal graphs that uses a self-attention mechanism and a novel functional time encoding technique to efficiently aggregate temporal-topological neighbourhood features and learn time-feature interactions. TGAT can handle both node classification and link prediction tasks, and can be extended to include temporal edge features (Xu et al., 2020). Joint Dynamic User-Item Embeddings (JODIE) has been proposed to predict future user-item interactions in domains such as e-commerce, social networking, and education. It is a coupled recurrent neural network model that learns the embedding trajectories of users and items through representation learning, and it introduces a novel projection operator to estimate the embedding of the user at any time in the future (Kumar et al., 2019). Dynamic Representation (DyRep) encodes evolving information over dynamic graphs into low-dimensional representations, namely as embeddings, using an inductive deep representation learning framework. The learned embeddings drive the dynamics of two fundamental processes: communication and association between nodes in dynamic graphs (Trivedi et al., 2019). Dynamic Self-Attention Network (DySAT) computes node representations by jointly using self-attention layers along two dimensions: structural neighbourhood and temporal dynamics, and has been evaluated on link prediction experiments (Sankar et al., 2019).

The methods mentioned above are all designed for single layer networks, and would thus not be suitable for the multilayer problem we tackle. Recently, however, several approaches to generalize GNNs to the multilayer case have been proposed. Firstly, Graph Attention Models for Multilayered Embeddings (GrAMME) introduces attention mechanisms and develops two GNN architectures to exploit the inter-layer dependencies: GrAMME-SG and GrAMME-Fusion. GrAMME-SG considers a multilayer network to be a supra graph with implicit edges between layers, whereas GrAMME-Fusion makes use of a supra fusion layer to aggregate embeddings from layerwise attention models (Shanthamallu et al., 2019). Secondly, Multilayer network Embedding via Learning Layer vectors (MELL) incorporates the idea of a layer vector that characterizes the connectivity in a layer. MELL embeds nodes in each layer into the lower embedding space using all layer structures and incorporates layer vectors to differentiate edge probabilities in the layers (Matsuno & Murata, 2018). Thirdly, Multilayer Graph Neural Network (mGNN) presents an innovative way of employing GCN on multilayer networks. In this approach, node feature propagation occurs independently in both intralayer and interlayer edges, and multiple layers can be stacked to capture information from the topology and features further in the network. This method can handle node classification, intra layer link prediction, and network clustering (Grassia et al., 2021).

All the aforementioned methods have been developed for networks that are either static and single layer, static and multilayer, or dynamic and single layer. A unified approach that can handle networks that are both dynamic and multilayer has not been put forward yet. This is significant, as most real-life networks share both characteristics simultaneously. Our approach focuses on solving this problem.

2.2. Credit risk modelling with network data

Credit risk modelling has a long history, and researchers from a broad range of areas have been working on developing credit risk rating systems (Markov et al., 2022). The statistical models that were traditionally used for credit risk modelling seemed to have difficulties dealing with large datasets as they may be characterized by increased noise, heavy-tailed distributions, nonlinear patterns, and temporal dependencies (Gordy, 2000). Advances in computing power and availability of large credit datasets paved the way to artificial intelligence (AI) driven credit risk estimation algorithms such as machine learning and deep learning (Shi et al., 2022). Recently, some machine learning methods were found to outperform conventional models in terms of accuracy when applied to large datasets (Lessmann et al., 2015; Gunnarsson et al., 2021).

Whereas traditional methods such as logistic regression make the IID assumption, treating borrowers as independent observations, it is widely understood that correlated default exists in lending (Óskarsdóttir & Bravo, 2021). Default correlation measures the extent to which the default of one borrower is related to that of another borrower, which may be caused by similar economic conditions affecting both, or, within a sector, by industry-specific reasons. Several researchers have shown that these correlations should be taken into account to avoid misestimating credit risk (Fenech et al., 2015). Some researchers have used alternative data sources such as network data to show the existence of correlation. For this purpose, they used different sources of data such as telephone call data (Óskarsdóttir et al., 2019), app-based marketplace data (Roa et al., 2021), social media data (De Cnudde et al., 2019), and agricultural loan network data (Óskarsdóttir & Bravo, 2021). Some other researchers have used network data from inter-firm transactions to improve credit risk modelling strategies for Small and Medium-sized Enterprises (SMEs) (Vinciotti et al., 2019). A common feature of the aforementioned studies is that they all use numerical measures obtained from the alternative data sources. However, all of them consider networks that are either static or single layer. In fact, nodes could be connected in various ways in a network, and these relationships could change over time, making them dynamic. Our work is an effort to take advantage of dynamic multilayer networks in this field and thus address some of the limitations of previous work.

Whereas traditionally numerical features were extracted from the network and used for credit scoring, more recently, GNNs have been used to develop predictive models in this field, eliminating the need to explicitly extract those features. In one study, GCNs were employed to predict peer-to-peer loan defaults and were found to outperform baseline models such as SVM, Random Forest, and XGBoost (Lee et al., 2021). GNN with Self-attention and Multi-task learning (SaM-GNN) has been proposed for credit default risk prediction. This approach incorporates two parallel tasks based on shared intermediate vectors for input vector reconstruction and credit default risk prediction (Li et al., 2022). Motif-preserving Graph Neural Network with curriculum learning (MotifGNN) has been introduced to jointly learn the lower-order structures from the original graph and higher-order structures from multi-view motif-based graphs for default prediction (Wang et al., 2023). In another study, a novel spatial-temporal aware GNN was proposed to predict SME loan default risk from a network of mined supply chain relationships (Yang et al., 2021). The potential benefits of using networks that are both dynamic and multilayer in credit risk modelling are not yet fully realized. This research presents a novel method for utilizing these networks within this area of study.

3. Methodology

In this section, we describe our methodology. First, we explain our approach for constructing a sequence of snapshots of multilayer networks. Then, we discuss how we employ different types of embeddings to encode topological and temporal dependencies in the networks. After that, we describe different configurations of DYMGNN and their respective architectures.

3.1. Multilayer networks

We start by defining the multilayer network and its node features. Consider an unweighted and undirected network $G = (V, A, X)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes in a layer of the network, $n = |V|$ denotes the number of distinct nodes, and $X \in \mathbb{R}^{n \times d}$ is a feature matrix, in which X_i is a column vector that represents the features of node v_i and d stands for the number of features. A network is represented by its supra adjacency matrix, $A \in \mathbb{R}^{nl \times nl}$, where l denotes the number of layers. This matrix encodes information about connections between pairs of nodes within a layer as well as connections between pairs of nodes from two different layers, i.e., if v_i from layer k and v_j from layer m are connected ($1 \leq k, m \leq l$), then $A_{(k-1)n+i,(m-1)n+j} = 1$; otherwise, the value is 0. In a multilayer network, all layers contain the same set of nodes, while their edge sets are assumed to be different. Each layer focuses on a particular type of relationship, with intra layer edges connecting the nodes that are related. In addition, a series of interlayer edges simply specify which nodes are identical. Fig. 1 shows a multilayer network and its corresponding supra adjacency matrix.

The network dynamics are captured through a sequence of snapshots $[G^{(1)}, ..., G^{(\tau)}]$ where $G^{(t)} = (V^{(t)}, A^{(t)}, X^{(t)})$ for each $t \in \{1, ..., \tau\}$. We are interested in obtaining the node embeddings at $t \leq \tau$

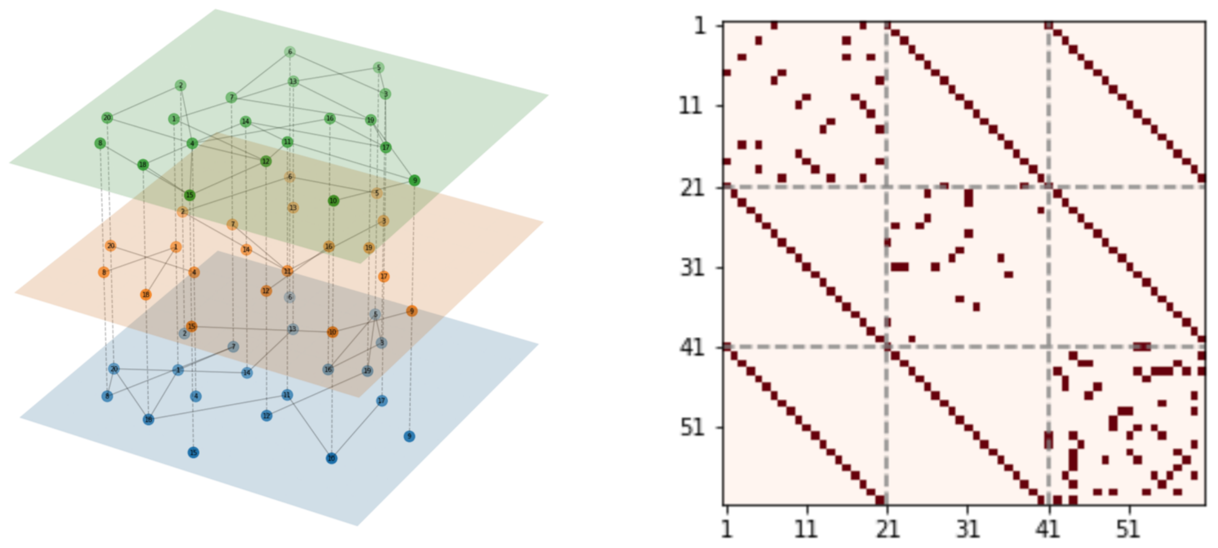


Figure 1: A multilayer network (left) and its supra adjacency matrix (right).

based on snapshots at or before t. For the application in this paper, we assume $V^{(1)} = V^{(2)} = ... = V^{(\tau)}$ and $A^{(1)} = A^{(2)} = ... = A^{(\tau)}$, i.e., nodes and their connections remain constant over time, but features can vary from one snapshot to another. Once the set of multilayer networks is complete, an embedding must be calculated from this data, as explained in the next section.

3.2. Topological embedding

Neural networks are a type of machine learning model inspired by the human brain, consisting of layers of interconnected neurons. Each neuron processes input data, applies a mathematical transformation using weights and biases, and passes the result to the next layer. They are trained through a process called backpropagation, which adjusts the weights and biases to minimize prediction errors measured by a loss function (Goodfellow et al., 2016). GNNs extend these neural networks to handle graph-structured data.

Capturing the topological dependence in a network is a key problem, as neighbouring nodes could influence each other. In this work, we trial two different types of GNNs, i.e., GCN (Kipf & Welling, 2017) and GAT (Veličković et al., 2018), to obtain the topological relationship between a node and its neighbours, encode the topological structure of the network and the features of nodes, capturing the information within the node connections. GCN or GAT is applied to each $G^{(t)}$ to obtain a hidden representation matrix $Z^{(t)}$. Each row of $Z^{(t)}$ contains a node embedding, meaning that for node v_i we have a sequence of embeddings $[Z_i^{(1)}, Z_i^{(2)}, ..., Z_i^{(\tau)}]$.

The GCN formulation performs isotropic aggregation, according to which each neighbour contributes equally to update the representation of the central node. The GCN model for a snapshot can be expressed as follows:

$$Z = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}XW^T. \tag{1}$$

Here, $\tilde{A} = A + I_{nl}$ is the supra adjacency matrix of the snapshot with inserted self-loops. I_{nl} is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the diagonal degree matrix of $\tilde{A}$, and $W^T \in \mathbb{R}^{d \times D}$ is a learnable weight matrix where D is the embedding size.

The GAT model expands the basic aggregation function of the GCN, assigning different importance to each edge through the attention coefficients. It can be formulated as follows:

$$e_{ij} = \text{LeakyReLU}(a^T[WX_i||WX_j]), \tag{2}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k \in N(v_i) \cup \{v_i\}} exp(e_{ik})}, \tag{3}$$

$$Z_i = \sum_{j \in N(v_i) \cup \{v_i\}} \alpha_{ij} WX_j. \tag{4}$$

Equation 2 computes a pairwise denormalized attention score between two neighbours, where $||$ denotes the concatenation operation and $a^T \in \mathbb{R}^{1 \times 2D}$ is a learnable weight vector. The attention score indicates the importance of a neighbour node in the message passing framework. Equation (3) applies a $softmax$ function to normalize the attention scores on each node's incoming edges. This function puts the output of the previous step in a probability distribution and, as a result, the attention scores are more comparable across different nodes. In this equation, $N(v_i)$ represents the neighbourhood of node v_i. Note, we also include the self-edge for each node. In Equation (4), the embeddings from neighbours are aggregated together, scaled by the attention scores. The main objective of this process is to learn a different contribution from each neighbour. The operations from (2) to (4) constitute a single head. The modelling capacity can be improved by considering multiple attention heads, thus allowing for different attention being given to different sets of neighbours. The output representations from the different heads can be aggregated using averaging operations.

3.3. Temporal embedding

Dealing with the temporal dependence is another key problem, as the temporal sequence of connections between nodes could provide useful information. In this work, we use LSTM (Hochreiter & Schmidhuber, 1997) and GRU (Cho et al., 2014) to catch the information related to the evolution of the networks. Both LSTM and GRU use gated mechanisms to memorize as much information as possible; however, there are some differences between these two models (Chung et al., 2014). Comparing these two, LSTM has a more complex structure, more parameters, and longer training time. LSTM is known to be able to deal with long-range dependencies, making it the preferred choice for models built over data of relatively large size (Yang et al., 2020). As our data is of medium size and the interface between GNNs and RNNs is not fully explored, the choice between LSTM and GRU is not obvious. We will, therefore, compare both models in Section 5. After obtaining the sequence of topological embeddings $[Z_i^{(1)}, Z_i^{(2)}, ..., Z_i^{(t)}]$, we need to input it into the RNN model and use the hidden representation of the RNN model as the temporal node embeddings for v_i.

LSTM uses a total of three gates, i.e., input gate, forget gate, and output gate. The input gate determines what information from the current topological embedding and previous temporal embeddings will be cached, or stored for future use, in long term memory. The forget gate decides which information from the long term memory should be maintained or repudiated. The output gate takes the current topological embedding, the previous temporal embedding and the newly computed long term memory to produce the new temporal embedding that will be passed on to the cell in the next time step. The LSTM model can be formulated as follows:

$$I^{(t)} = \sigma(Z^{(t)}W_{ii} + H^{(t-1)}W_{ih} + b_i), \tag{5}$$

$$F^{(t)} = \sigma(Z^{(t)}W_{fi} + H^{(t-1)}W_{fh} + b_f), \tag{6}$$

$$C^{(t)} = F^{(t)} \odot C^{(t-1)} + I^{(t)} \odot tanh(Z^{(t)}W_{ci} + H^{(t-1)}W_{ch} + b_c), \tag{7}$$

$$O^{(t)} = \sigma(Z^{(t)}W_{oi} + H^{(t-1)}W_{oh} + b_o), \tag{8}$$

$$H^{(t)} = O^{(t)} \odot tanh(C^{(t)}). \tag{9}$$

In the equations above, $\odot$ denotes element-wise (Hadamard) product. σ is an activation function (typically *sigmoid*) and *tanh* represents the hyperbolic tangent function. $I^{(t)} \in \mathbb{R}^{nl \times D}$, $F^{(t)} \in \mathbb{R}^{nl \times D}$, and $O^{(t)} \in \mathbb{R}^{nl \times D}$ represent input, forget, and output gates for the nodes, respectively. $C^{(t)} \in \mathbb{R}^{nl \times D}$ and $H^{(t)} \in \mathbb{R}^{nl \times D}$ are memory cell and hidden state for the node embeddings, respectively. $W_{(\cdot\cdot)} \in \mathbb{R}^{D \times D}$ and $b_{(\cdot)} \in \mathbb{R}^{1 \times D}$ are weight matrix and bias vector, respectively. $H^{(0)}$ and $C^{(0)}$ can be initialized with zeros or learned from the data (Mohajerin & Waslander, 2017).

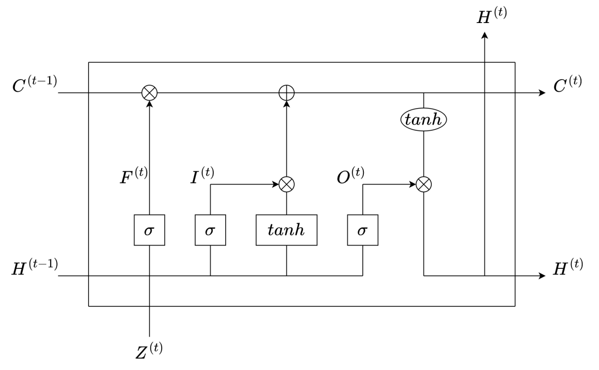
GRU is similar to LSTM, but it incorporates two gates, i.e., an update gate and a reset gate. The reset gate determines how much of the previous temporal embedding should be neglected, while the update gate determines the amount of the new input that needs to be passed along to the next state. The GRU model can be formulated as

$$U^{(t)} = \sigma(Z^{(t)}W_{ui} + H^{(t-1)}W_{uh} + b_u), \tag{10}$$

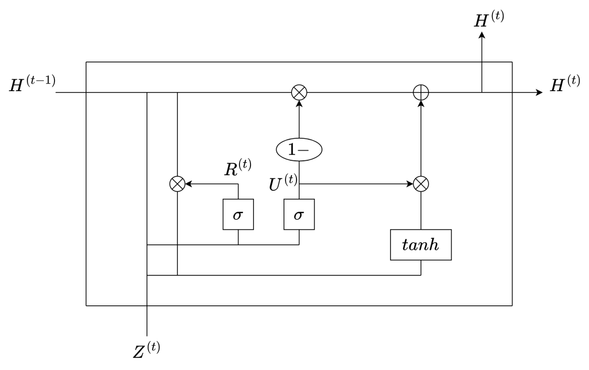
$$R^{(t)} = \sigma(Z^{(t)}W_{ri} + H^{(t-1)}W_{rh} + b_r), \tag{11}$$

$$H^{(t)} = (1 - U^{(t)}) \odot H^{(t-1)} + U^{(t)} \odot tanh[Z^{(t)}W_{hi} + (R^{(t)} \odot H^{(t-1)})W_{hh} + b_h]. \tag{12}$$

In the equations above, $U^{(t)} \in \mathbb{R}^{nl \times D}$ and $R^{(t)} \in \mathbb{R}^{nl \times D}$ represent update and reset gates for the nodes, respectively. All other definitions are the same as LSTM. Fig. 2 depicts the respective structures of the LSTM and GRU models.



(a) LSTM



(b) GRU

Figure 2: The cell structures of RNN models.

3.4. GNN-RNN models

The GNN-RNN model is one of the proposed models for DYMGNN in this work. Within the GNN-RNN framework, there are two configurations, i.e., GNN-LSTM and GNN-GRU, which can be summarized as

$$Z^{(t)} = \text{GNN}(X^{(t)}, A^{(t)}), \tag{13}$$

$$H^{(t)}, C^{(t)} = \text{LSTM}(Z^{(t)}, H^{(t-1)}, C^{(t-1)}) \quad \text{for GNN-LSTM}, \tag{14}$$

$$H^{(t)} = \text{GRU}(Z^{(t)}, H^{(t-1)}) \quad \text{for GNN-GRU}. \tag{15}$$

Fig. 3 displays an overview of these models.

These models are capable of capturing the topological and temporal dependencies of snapshots through combining GNN and RNN, whilst the same importance is assigned to each timestamp. The temporal

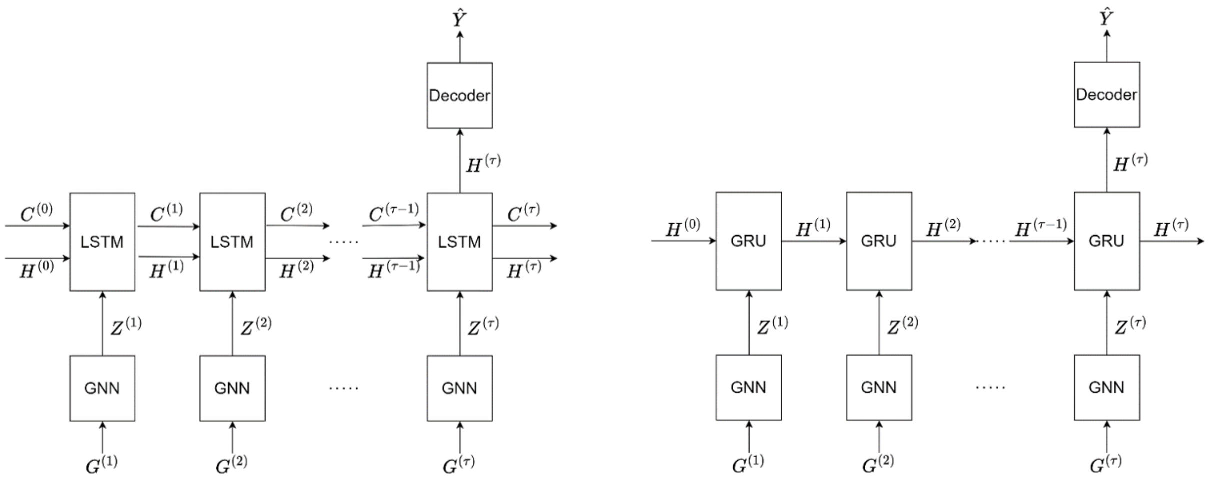


Figure 3: GNN-LSTM (left) and GNN-GRU (right) dynamic models.

embedding for the nodes is obtained by feeding their sequence of embeddings, produced by the GNN model, to the RNN model.

3.5. GNN-RNN-ATT models

The GNN-RNN-ATT model is another proposed model for DYMGNN. In GNN-RNN-ATT models, a soft attention mechanism is applied to assign different importance to each timestamp. This approach contrasts with GNN-RNN models which assign equal importance to every timestamp. The use of attention in GNN-RNN-ATT models allows for a more nuanced weighting of temporal information. Our approach for creating a new hidden state for the node embeddings that is more expressive of the global variation trends can be formulated as follows:

$$s^{(t)} = a_h H^{(t)} W_h, \tag{16}$$

$$\beta^{(t)} = \frac{exp(s^{(t)})}{\sum_{k=1}^{\tau} exp(s^{(k)})}, \tag{17}$$

$$H_{att} = \sum_{t=1}^{\tau} \beta^{(t)} H^{(t)}. \tag{18}$$

First, the hidden states at different timestamps, $H^{(t)}$, are obtained using GNN and RNN, as discussed in the previous model. Equation 16 computes an denormalized attention score for each hidden state, where $a_h \in \mathbb{R}^{1\times nl}$ and $W_h \in \mathbb{R}^{D\times 1}$ are learnable weight vectors. The normalized attention score for each hidden state is computed using a $softmax$ function as shown in Equation 17. In Equation (18), H_{att} is calculated by aggregating the hidden states scaled by the normalized attention scores. The main goal of this process is to re-weight the influence of snapshots at different timestamps. Finally, the final output results can be obtained using H_{att} that can describe the global variation information.

Fig. 4 shows two configurations of GNN-RNN-ATT, i.e., GNN-LSTM-ATT and GNN-GRU-ATT.

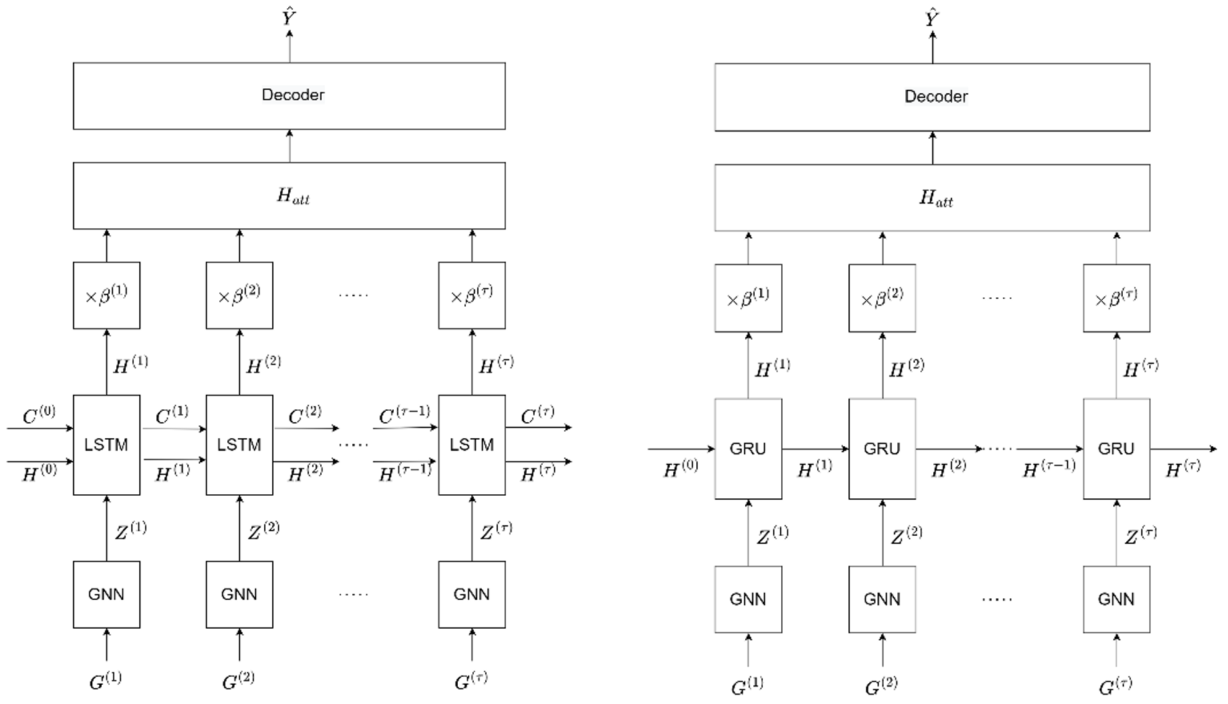


Figure 4: GNN-LSTM-ATT (left) and GNN-GRU-ATT (right) dynamic models. By adding an attention layer to the model, we are able to re-weight the impact of different snapshots.

3.6. Decoder and loss function

A deep neural network model is typically comprised of an encoder and a decoder. The encoder takes input and produces embeddings, whereas the decoder takes the embeddings and performs the prediction task (Goodfellow et al., 2016). In our specific case, GNN, RNN, and ATT comprise the encoder of the full model, while the decoder is a set of feed-forward neural networks applied to the node embeddings, followed by a series of layers that either apply a chosen activation function for non-linearity or dropout function for regularization. The final output is the model prediction for our binary outcome (here, default Y/N), i.e., whether the node v_i belongs to class 1 ($Y_i = 1$) or 0 ($Y_i = 0$). The decoder outputs a vector $\hat{Y}$ where $\hat{Y}_i$ specifies the probability of a node v_i belonging to class 1 given the snapshots $[G^{(1)}, ..., G^{(\tau)}]$. While there is no unique format for the decoder, the architecture used in this work is shown in Fig. 5.

One of the most important aspects of a deep learning model is its loss function. For our work, we use the well-known binary cross-entropy loss function (Gneiting & Raftery, 2007) which can be written as

$$\text{Loss} = \frac{-1}{n} \sum_{i=1}^{n} \Big[Y_i \cdot log(\hat{Y}_i) + (1 - Y_i) \cdot log(1 - \hat{Y}_i) \Big] . \tag{19}$$

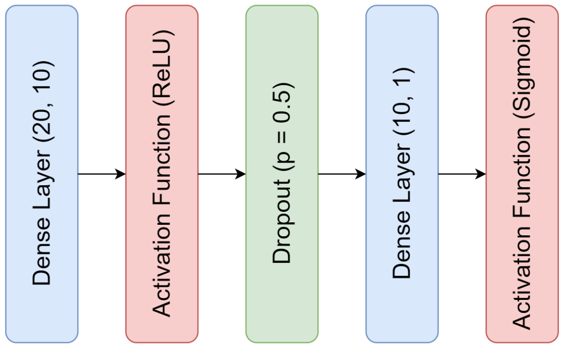


Figure 5: Architecture of the decoder.

4. Experimental setup

4.1. Dataset

In this paper, the goal of our models is to predict one-year-ahead loan default based on borrower or loan characteristics. For this purpose, we use the Single-Family Loan-Level (SFLL) dataset provided by the Federal Home Loan Mortgage Corporation (FHLMC), commonly known as Freddie Mac, which contains loan-level data for a sizable share of mortgage loans in the United States (FreddieMac, 2022). Freddie Mac purchases mortgages on the secondary market, pools them, and sells them as a mortgage-backed security to investors on the open market.

The dataset includes information regarding the loan, such as the amount, the interest rate, the insurance percentage, and the provider, as well as information on the borrower, including the borrower's debt to income ratio and/or unpaid balance, FICO credit score, the geographical area in which they reside, and whether they are a first-time home buyer. It also includes information about the property (type, number of units, etc.). To represent the categorical information, we introduce our own binary features contrasting one category against other categories combined. Numerical node features are normalized using min-max scaling. We clean the data by treating outliers and null values. Specifically, outliers are capped at the 99th percentile and 1st percentile points. There are not many null values, and they are treated with median imputation. Feature descriptions for the data used in model training are given in Table 1. Most of the features are available at the time of loan application and do not change from one month to another; however, a few features such as 'current_upb', 'if_delq_sts', 'mths_remng', and 'current_int_rt' can change from month to month, as they track repayment behaviour over the loan period. More information about the data can be found in Appendix A.

4.2. Dynamic networks

As we are interested in studying the effect of connections between the borrowers and the evolution of those connections over time, we use the data to create a sequence of dynamic networks following the

Feature	Description
fico	Credit score at the time of acquisition
if_fthb	Is the borrower a first-time home buyer?
mi_pct	Mortgage insurance percentage
cnt_units	Number of units in the property
if_prim_res	Is the property a primary residence?
dti	Original debt to income ratio
ltv	Original loan to value ratio
if_corr	Is a correspondent involved in the origination of the mortgage?
if_sf	Is the property a single family home?
if_purc	Is the mortgage loan a purchase mortgage?
cnt_borr	Number of borrowers obligated to repay the mortgage
if_sc	Does the mortgage exceed conforming loan limit?
current_upb	Current unpaid principal balance
if_delq_sts	Are there any payment arrears (between 30 and 90 days)?
mths_remng	Number of remaining months of the mortgage
current_int_rt	Current interest rate
default	Being 90 days or more in payment arrears over next 12 months

Table 1: Description of the node features.

process in subsection 3.1. In particular, we are interested in predicting one-year-ahead loan default based on application information and six months of borrower's repayment behaviour. We choose a six-month period because six and twelve months are common choices for lookback periods (Kennedy et al., 2013). Furthermore, this paper will later show that extending beyond six months offers minimal additional benefits.

For this work, loans originated in 2009 and 2010 are used for training and testing. We select these years to ensure sufficient default information is available for reliably comparing the models. It is important to note, however, that loan population and behaviour change over time, and our sample data may reflect some effects of the global financial crisis. Thus, further research could explore the robustness of our proposed methods across different time periods and financial conditions. We use application data and 18 months of behavioural data, from January 2012 to June 2013, for training. We also use application data and six months of behavioural data of a holdout set, from July 2013 to December 2013, for testing. We consider rolling windows, shifting by one month, for training and testing, with each window containing six snapshots $[G^{(1)}, ..., G^{(6)}]$, and each snapshot corresponding to one month. So, we have 13 windows for training and one window for testing. All snapshots of a specific window have the same set of nodes. However, the node set could be different from one window to another; a loan that has defaulted will remain marked as a defaulter for the observation window but will disappear once the window moves past it. During the training of each window, the goal is to predict default within 12 months following the month of the last snapshot in that window. A one-year horizon is practical for credit management and decision-making, as it balances the need for a sufficiently long period to assess risk while not extending so far that predictions become highly speculative (Lopez & Saidenberg, 2000). Fig. 6 displays the timeline for the windows and their corresponding horizons in model training.

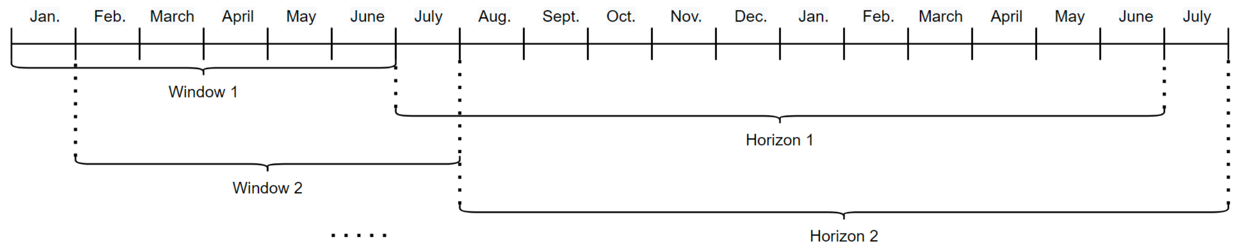


Figure 6: The timeline for windows and their corresponding horizons in model training.

We can train the models using either single layer or double layer networks, with geographical location of the borrower and the company lending the loan being the connector variables. Borrowers whose zip codes have the same first two digits are assumed to be in the same geographical area. In the case of the double layer network, nodes in one layer are also connected to their twins in the other layer. Some previous studies showed that applying some sort of dropout techniques on graph structures could help increase the expressiveness of the GNN models (Shu et al., 2022). Therefore, we decide to randomly select and isolate 50% of the nodes in each snapshot of a window. In other words, in each snapshot, at least half of the nodes do not have any connections with other nodes.

To gain insight into the size and characteristics of the networks, we provide some descriptions in Table 2. Letting $G^{(T)} = \bigcup_{t=1}^{6} G^{(t)}$, Table 2 shows the number of nodes and the number of edges for $G^{(T)}$ created from the snapshots in the first window of the training set, as well as the snapshots of the validation and test sets. It is evident that the single layer networks derived from the lending company are denser than those derived from the geographical area. This is because there are fewer lending companies serving as connector variables compared to the number of geographical areas serving as such.

	Single Layer:Area		Single Layer:Company		Double Layer:Area-Company	
Set	#Nodes	#Edges	#Nodes	#Edges	#Nodes	#Edges
Training	148,520	16,368,244	148,520	91,486,176	297,040	108,151,460
Validation	82,180	4,725,842	82,180	27,735,664	164,360	32,625,866
Test	96,490	6,761,051	96,490	38,404,277	192,980	45,358,308

Table 2: Network description for $G^{(T)}$.

4.3. Experiments

We are interested in comparing the performance of different models on single layer and double layer networks, and in benchmarking them against some baseline methods. The classes are imbalanced in this binary node classification problem, so we use the Area Under the Curve (AUC) and F_1 score to assess the performance of each model. The results are presented with 95% confidence intervals, derived from bootstrap over the test set.

As computational efficiency is another consideration, we also examine the runtime for training the models. In addition, we use the Shapley approach to help us interpret the best performing model and better understand the importance of the different node features. We also look at the attention scores to assess the relative contribution of snapshots at different timestamps.

5. Results and discussion

5.1. Baseline methods

We benchmark our proposed model against a selection of GNN-based and non-GNN-based baseline models. Table 3 shows the results for two GNN-based baseline models, i.e., static GCN and static GAT, whereas Table 4 shows the results for three non-GNN-based baseline models, i.e., Logistic Regression (LR), XGBoost (XGB), and a Deep Neural Network (DNN). LR is popular in the commercial and financial sectors due to its straightforwardness and ease of understanding. Meanwhile, XGB has established itself as a powerful technique for both classification and regression tasks involving structured datasets (Gunnarsson et al., 2021). DNN is fundamental to deep learning and has seen broad usage across a variety of predictive tasks.

For training the GNN-based baseline models, we consider a static network, which is the last snapshot of each window, and values of behavioural features are the mean values of those features across the six snapshots of the respective window. We do not use RNNs for these static models; the decoder and the loss function for these models are the same as those used in the dynamic models.

For non-GNN-based baseline models that rely solely on non-network features, we use a grid search to tune the hyper-parameters for each model using the validation dataset. The LR model is tuned with saga solver and a grid search for the penalty {L1, L2}. The XGB model hyper-parameters are tuned with a grid search for the learning rate {0.001, 0.01, 0.1}, maximum depth {2, 3, 4}, number of estimators {50, 100, 250, 500}, and alpha {0.1,. . . ,0.9}. The architecture of the DNN is given in Appendix B.

	Single Layer:Area		Single Layer:Company		Double Layer:Area-Company	
Model	AUC	F_1	AUC	F_1	AUC	F_1
Static GCN	0.701 ± 0.014	0.802 ± 0.012	0.681 ± 0.014	0.798 ± 0.013	0.729 ± 0.012	0.810 ± 0.012
Static GAT	0.752 ± 0.013	0.814 ± 0.010	0.746 ± 0.011	0.812 ± 0.009	0.763 ± 0.014	0.817 ± 0.012

Table 3: Performance of the GNN-based baseline models.

In Table 3, we can see that the Static GAT performs better than the Static GCN, on both single layer and double layer networks. The difference in performance between them is considerable, and could be

Model	AUC	F_1
LR	0.796 ± 0.020	0.824 ± 0.013
XGB	0.805 ± 0.018	0.837 ± 0.012
DNN	0.803 ± 0.016	0.833 ± 0.014

Table 4: Performance of the non-GNN-based baseline models.

due to the different way in which GAT and GCN aggregate information from the one-hop neighbourhood. Among the non-GNN models (see Table 4), XGB appears to have better performance compared to LR and DNN; however, the differences are fairly small. XGB outperforms LR, suggesting that XGB can capture non-linear relationships better than LR does. It is also not unexpected to see that DNN does not outperform XGB, as this might be the case where the structured data is not very complex or does not contain many features (Borisov et al., 2022; Gunnarsson et al., 2021). Comparing Table 4 against Table 3, it is observed that the performance of each non-GNN-based baseline model surpasses that of the best performing GNN-based baseline model, i.e., Static GAT, on both single layer and double layer networks. This observation is important as it indicates that more complex models do not always yield better performance.

5.2. Performance of the dynamic models

Table 5 and Table 6 show the performance of the different models on our two single layer networks, while Table 7 shows the performance on the double layer network. Out-of-sample performance is again measured in terms of AUC and F_1 score on a test set. The best performing model in each group is highlighted in bold.

Model		AUC		F_1	
Topological	Temporal	Without ATT	With ATT	Without ATT	With ATT
GCN	LSTM	0.804 ± 0.011	0.807 ± 0.012	0.841 ± 0.008	0.847 ± 0.009
	GRU	0.775 ± 0.013	0.780 ± 0.011	0.825 ± 0.006	0.829 ± 0.008
GAT	LSTM	0.806 ± 0.009	$\mathbf{0.810 \pm 0.012}$	0.842 ± 0.005	$\mathbf{0.849 \pm 0.007}$
	GRU	0.793 ± 0.005	0.802 ± 0.014	0.833 ± 0.004	0.840 ± 0.006

Table 5: Performance of the dynamic models on the single layer network derived from the geographical area.

Model		AUC		F_1	
Topological	Temporal	Without ATT	With ATT	Without ATT	With ATT
GCN	LSTM	0.802 ± 0.012	0.804 ± 0.012	0.840 ± 0.009	0.843 ± 0.007
	GRU	0.769 ± 0.013	0.774 ± 0.014	0.818 ± 0.006	0.823 ± 0.006
GAT	LSTM	0.805 ± 0.012	$\mathbf{0.808 \pm 0.010}$	0.840 ± 0.009	$\mathbf{0.846 \pm 0.008}$
	GRU	0.786 ± 0.007	0.795 ± 0.014	0.832 ± 0.004	0.835 ± 0.006

Table 6: Performance of the dynamic models on the single layer network derived from the lending company.

Table 5 and Table 6 show that, for each of the single layer networks, the GAT-LSTM-ATT model produces the highest AUC and F_1 score, while GCN-GRU gives the poorest results. This could be due to the fact that GAT assigns different importance to each edge, and we know that some connections could be more informative than others. Also, the complex structure of LSTM appears to make it the preferred RNN for this problem. Another key observation is that models enhanced with the attention mechanism consistently show better performance compared to those without attention.

Model		AUC		F_1	
Topological	Temporal	Without ATT	With ATT	Without ATT	With ATT
GCN	LSTM	0.806 ± 0.010	0.810 ± 0.009	0.845 ± 0.004	0.848 ± 0.006
	GRU	0.789 ± 0.010	0.793 ± 0.011	0.833 ± 0.005	0.835 ± 0.009
GAT	LSTM	0.807 ± 0.008	$\mathbf{0.812 \pm 0.008}$	0.847 ± 0.005	$\mathbf{0.851 \pm 0.007}$
	GRU	0.800 ± 0.004	0.804 ± 0.006	0.839 ± 0.003	0.843 ± 0.008

Table 7: Performance of the dynamic models on double layer network created with both geographical area and lending company.

As for the double layer network, we can see from Table 7 that, similarly to what was observed for the single layer networks, the GAT-LSTM-ATT model again shows the best performance. The results for the double layer network, however, tend to outperform the single layer ones, which is intuitive as the double layer network is able to consider connections of either type. It is also noticeable that the results obtained from the double layer network have shorter confidence intervals, suggesting greater robustness in these results. Importantly, the best performing dynamic modelling approach, i.e., GAT-LSTM-ATT, performs better on average than the baseline methods presented in the previous subsection. For example, the GAT-LSTM-ATT model for the double-layer network produces AUC and F_1 score of 0.812 and 0.851, respectively, compared to 0.805 and 0.837 for the best baseline model, i.e., XGB (see Table 4). This translates to a 0.87% gain in AUC and a 1.67% gain in F_1 score. Although these numerical gains might seem modest, even a 1% improvement can yield significant financial benefits for some businesses. Interestingly, even when applied to either of the single layer networks, the GAT-LSTM-ATT still tends to perform well compared to the baseline models. This demonstrates that DYMGNN offers an advantage over conventional methods by capturing a richer set of information, thus providing a more comprehensive and realistic picture of a borrower's default probability. Hence, incorporating dynamic network information is able to provide additional information over simply using local features and/or static networks. The most pronounced difference lies between our dynamic model and the static network-based models, demonstrating the importance of capturing network changes over a sufficiently long time window.

5.3. Runtime analysis

Computational complexity of training machine learning models considers two core aspects: time complexity and space complexity. Time complexity relates to the time it takes to train a model and how this is affected by problem size, whereas space refers to how much space a model uses (memory footprint).

As time complexity is a potential consideration in our work, we report the runtimes for the dynamic models in Table 8 and Table 9. The runtimes for the GNN-RNN models and GNN-RNN-ATT models are presented in separate tables as those models' architectures differ from each other. The hyperparameters and resources used for the computations can be found in Appendix C. Note, to allow for easier comparison, the runtimes in each table are also normalized with respect to the lowest number in that table.

Model		Single Layer:Area		Single Layer:Company		Double Layer:Area-Company	
Topological	Temporal	Non-normalized	Normalized	Non-normalized	Normalized	Non-normalized	Normalized
GCN	LSTM	1,690	1.18	7,090	4.93	8,397	5.84
	GRU	1,437	1.00	6,109	4.25	7,221	5.03
GAT	LSTM	2,571	1.79	10,148	7.06	12,171	8.47
	GRU	2,081	1.45	8,390	5.84	10,019	6.97

Table 8: Runtime for training the GNN-RNN models (seconds).

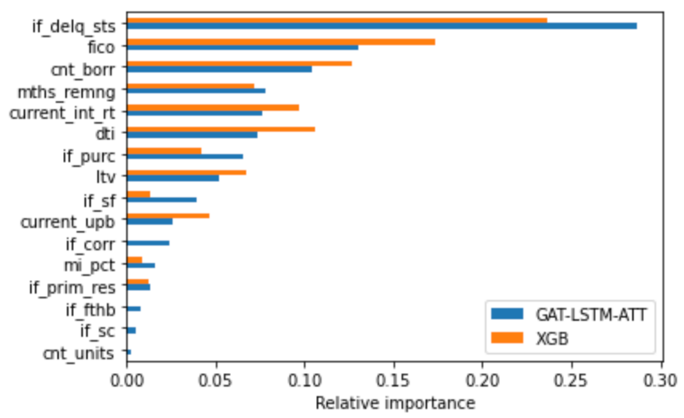
Model		Single Layer:Area		Single Layer:Company		Double Layer:Area-Company	
Topological	Temporal	Non-normalized	Normalized	Non-normalized	Normalized	Non-normalized	Normalized
GCN	LSTM	1,700	1.16	7,104	4.86	8,481	5.80
	GRU	1,463	1.00	6,190	4.23	7,225	4.94
GAT	LSTM	2,597	1.78	10,151	6.94	12,120	8.28
	GRU	2,114	1.44	8,413	5.75	10,054	6.87

Table 9: Runtime for training the GNN-RNN-ATT models (seconds).

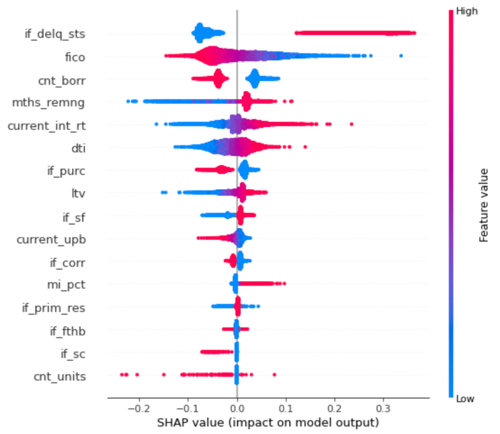
From the tables, we can see that training a model on a single layer network derived from the lending company takes longer than training a model on a single layer network created based on geographical area. This is not unexpected as the former network contains a higher number of connections between the nodes compared to the latter. GAT-LSTM and GAT-LSTM-ATT have the highest training runtimes among the GNN-RNN and GNN-RNN-ATT models, respectively, whereas GCN-GRU and GCN-GRU-ATT have the lowest runtimes. GNN-RNN-ATT models normally have higher runtimes compared to the GNN-RNN models, owing to the complexity added to those models by the attention mechanism. It is worth noting that the runtime for the XGB model is only 151 seconds, highlighting the greater complexity of our proposed models compared to traditional non-network classifiers.

5.4. Interpretability of the architecture

Having established that the best results can be obtained by applying the GAT-LSTM-ATT model to the double layer network, in this section, we employ the Shapley approach (Lundberg & Lee, 2017) to better understand this model. Using this method, we can establish each node feature's relative importance and quantify its contribution to the model output. Fig. 7a displays the relative importance of node features for the best performing proposed model, i.e., GAT-LSTM-ATT, and the best performing baseline model, i.e., XGB. Fig. 7b displays an information-dense summary of how the node features for the best performing proposed model impact its output.



(a) Relative importance for GAT-LSTM-ATT and XGB.



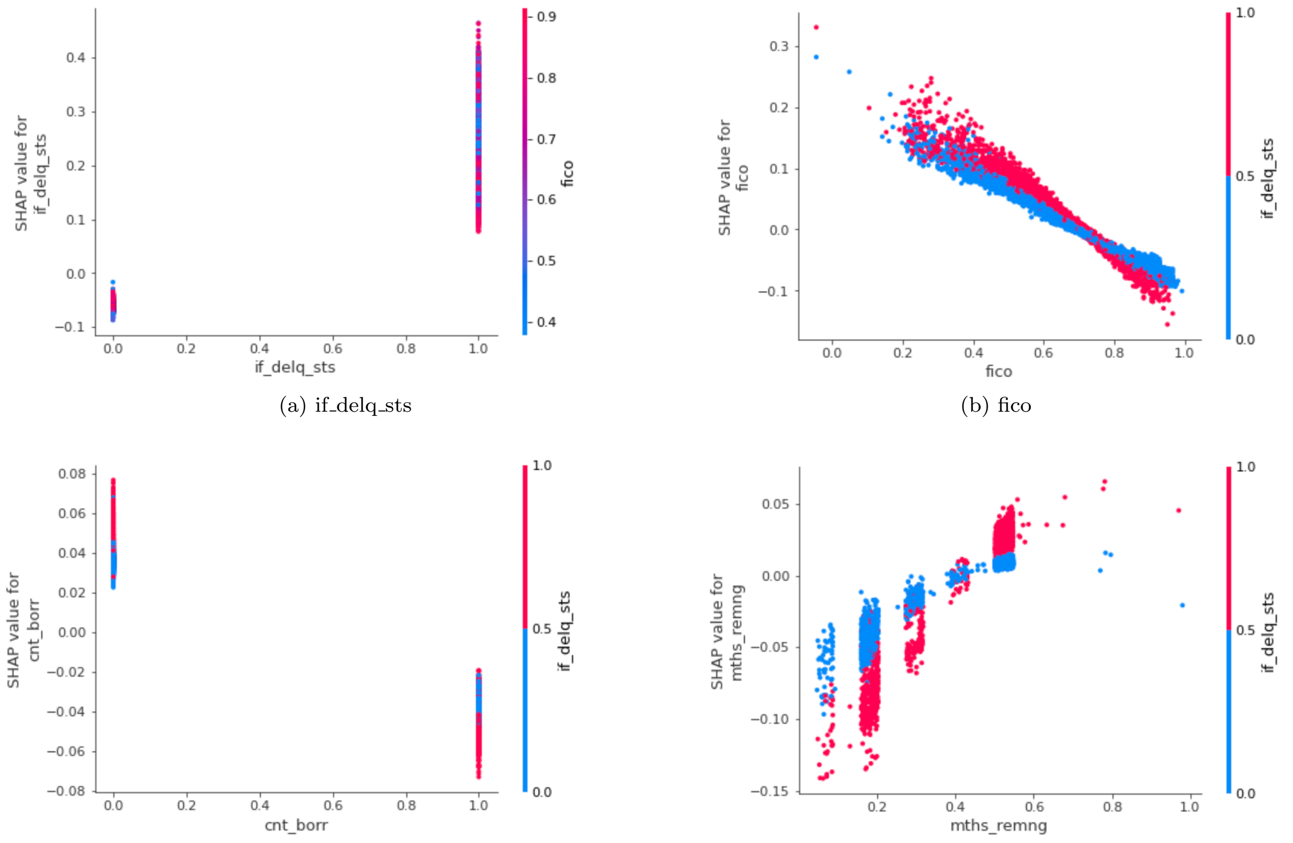
(b) Shapley values for GAT-LSTM-ATT.

Figure 7: Summary of node feature importance.

As seen in Fig. 7a, the presence of overdue payments holds the most significant relative importance compared to other features, for both GAT-LSTM-ATT and XGB. Overdue payments are a strong indicator of a borrower's financial health; similarly, while timely payments generally suggest good financial management, payment arrears can signal financial distress. The FICO credit score has the second highest relative contribution among the features. This is intuitive as this feature summarizes a lot of information about the payment history and financial behaviour of the borrower. For both models, the number of borrowers ranks as the third most crucial feature. For the GAT-LSTM-ATT model, the number of remaining months holds the fourth position in terms of importance, whereas for the XGB model, the debt to income ratio claims the fourth spot. Notably, the disparity in the significance attributed to features by the two models is more pronounced for the top two features.

Fig. 7b shows that payment arrears are highly indicative of default risk. It can also be viewed that borrowers with high credit scores are less likely to default, according to the model, while borrowers with low credit scores are more prone to be classified as defaulters. High values of the number of borrowers are associated with lower default risk, while low values are associated with higher default risk. Higher (lower) number of remaining months is linked with higher (lower) default risk.

Fig. 8 displays the dependency plots of the four most important features. Note, the feature values are scaled to be between 0 and 1 using min-max scaling. Fig. 8a illustrates that borrowers who consistently meet their payment deadlines tend to have high credit scores and are less prone to default. On the other hand, those who experience delays in making payments are more commonly associated with the cohort



(a) if_delq_sts

(b) fico

(c) cnt_borr

(d) mths_remng

Figure 8: Dependency plots of the four most important features for GAT-LSTM-ATT. The colour shows the value of the closest feature by correlation.

of defaulters. Fig. 8b demonstrates that the credit score, by and large, has a linear impact, with higher scores signalling lower risk of default. Lower values of credit score are much more informative than the higher values. Additionally, the model reveals some intriguing interaction effects. The relative impact of credit score on the default risk is more pronounced in the case of borrowers who have a history of late payments. This indicates that while a low credit score is generally a good indicator of high default risk, its predictive influence increases for those who do not consistently make timely payments. Fig. 8c suggests that cases involving fewer borrowers are more likely to default on their loans. This might be attributed to various factors, such as limited financial resources, reduced collective responsibility, or lesser peer pressure to maintain creditworthiness among a smaller group. Fig. 8d indicates that the number of remaining months displays a nearly linear trend, with lower numbers pointing to safer cases. This may stem from the increased uncertainty associated with longer durations (as opposed to shorter durations which signal an approaching end to the financial commitment) or from survival bias. Additionally, the figure points out that this feature's effect on default risk is more significant for borrowers with a pattern of delayed payments.

We also aim to analyse the normalized attention scores from the GAT-LSTM-ATT model to determine the relative importance of each timestamp. Fig. 9 illustrates how these scores vary over time.

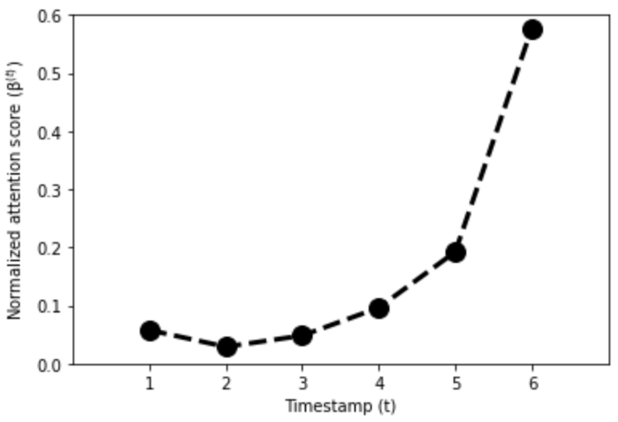


Figure 9: Variation of the normalized attention scores.

The figure shows that for the first few timestamps, the attention score is relatively stable and low, remaining close to 0.1. This indicates a minimal level of attention or importance being assigned during these early timestamps. However, as time progresses, particularly after timestamp 3, there is a noticeable upward trend in the attention score. This increase becomes more pronounced between timestamps 4 and 6, where the attention score rises sharply, peaking just below 0.6 at timestamp 6. This pattern suggests that as time progresses, the snapshots grow in importance. The reason for this progressive increase in attention could be that the most recent information holds greater value. Additionally, it can be inferred that longer lookback periods are unlikely to add anything extra, since the bulk of attention is allocated within a relatively brief period.

6. Conclusions

This study introduced an innovative approach to credit risk assessment through the use of dynamic graph neural networks. We found that this technique outperforms traditional models commonly applied in the sector when tested against US mortgage data. By harnessing the capabilities of both GNN and RNN, our method successfully captures the evolving connections between individual loans. We engineered this methodology to exploit the potential of multilayer networks, rather than the common single layer ones. The findings suggested that models incorporating double layer networks with a customized attention mechanism show enhanced predictive capability.

The evaluation of these models was conducted using a dataset from the mortgage lending domain. In our experiments, we constructed single and double layer networks using the borrower's geographical location and the lending company as the connector variables. Through rigorous testing of various models, we established that the GAT-LSTM-ATT model exhibits the best performance among all configurations of DYMGNN, and other baseline models, both GNN-based and non-GNN-based. Furthermore, this model is able to capture a richer set of information, thereby providing more realistic insights into a borrower's probability of default.

When comparing training times, it became apparent that models employing the attention mechanism exhibit greater complexity and require more extensive training times, yet the runtime remains within acceptable limits. As in any operational research area, explainability is important to consider (De Bock et al., 2023). Therefore, we applied the Shapley approach to decode the model's inner workings, assessing the impact of each node feature on the final output. This analysis revealed the differences in the importance of node features between a baseline model and our DYMGNN model, with particular focus on the four most pivotal features. Additionally, we investigated the relative importance of snapshots at different timestamps by analysing the attention scores associated with each. The results confirmed that the most recent snapshots play a crucial role in influencing the model's output.

Future research could explore wider networks by incorporating additional layers to map more complex inter-individual connections. The method could also be extended by further considering distance information and assigning different weights to the network edges based on geographical proximity between the centroids of neighbouring zip code areas. It might also be beneficial to vary the number of snapshots by adjusting the window length for generating dynamic networks. The exploration of other GNNs and RNNs not considered in this study presents another promising direction. Moreover, gaining a deeper understanding of how different network connections influence default risk could offer valuable insights into credit risk modelling.

Acknowledgements

The first author acknowledges the support of the Natural Sciences and Engineering Research Council (NSERC) of Canada through the Canada Graduate Scholarships – Doctoral (CGS D) program. The second and fourth authors acknowledge the support of the Economic and Social Research Council (ESRC) [grant number ES/P000673/1]. The third author acknowledges the support of the Icelandic Research Fund (IRF) [grant number 228511-051]. The last author acknowledges the support of the NSERC [discovery grant RGPIN-2020-07114]. This research was undertaken, in part, thanks to funding from the Canada Research Chairs program [CRC-2018-00082]. This work was enabled in part by support provided by Compute Ontario (`computeontario.ca`), Calcul Québec (`calculquebec.ca`), and the Digital Research Alliance of Canada (`alliancecan.ca`).

References

Aliabadi, M. M., Emami, H., Dong, M., & Huang, Y. (2020). Attention-based recurrent neural network for multistep-ahead prediction of process performance. *Computers & Chemical Engineering*, *140*, 106931.

Altché, F., & de La Fortelle, A. (2017). An LSTM network for highway trajectory prediction. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)* (pp. 353–359).

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR)*.

Barabási, A.-L., & Pósfai, M. (2016). *Network science*. Cambridge University Press.

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., & Kasneci, G. (2022). Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, *Early Access*, 1–21.

Bravo, C., & Óskarsdóttir, M. (2020). Evolution of credit risk using a personalized pagerank algorithm for multilayer networks. In *KDD MLF 2020: KDD Workshop on Machine Learning in Finance*.

Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 551–561).

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (pp. 103–111).

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*.

De Bock, K. W., Coussement, K., De Caigny, A., Slowiński, R., Baesens, B., Boute, R. N., Choi, T.-M., Delen, D., Kraus, M., Lessmann, S. et al. (2023). Explainable AI for operational research: A defining framework, methods, applications, and a research agenda. *European Journal of Operational Research*, *In press*.

De Cnudde, S., Moeyersoms, J., Stankova, M., Tobback, E., Javaly, V., & Martens, D. (2019). What does your Facebook profile reveal about your creditworthiness? Using alternative data for microfinance. *Journal of the Operational Research Society*, *70*, 353–363.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Fenech, J. P., Vosgha, H., & Shafik, S. (2015). Loan default correlation using an Archimedean copula approach: A case for recalibration. *Economic Modelling*, *47*, 340–354.

FreddieMac (2022). Single family loan-level dataset. URL: `https://www.freddiemac.com/research/datasets/sf-loanlevel-dataset`.

Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, *102*, 359–378.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Gordy, M. B. (2000). A comparative anatomy of credit risk models. *Journal of Banking & Finance*, *24*, 119–149.

Grassia, M., De Domenico, M., & Mangioni, G. (2021). mGNN: Generalizing the graph neural networks to the multilayer case. *arXiv preprint arXiv:2109.10119*, .

Graves, A., Fernández, S., & Schmidhuber, J. (2007). Multi-dimensional recurrent neural networks. In *International Conference on Artificial Neural Networks (ICANN 2007)* (pp. 549–558).

Gunnarsson, B. R., Vanden Broucke, S., Baesens, B., Óskarsdóttir, M., & Lemahieu, W. (2021). Deep learning for credit scoring: Do or don't? *European Journal of Operational Research*, *295*, 292–305.

Haythornthwaite, C. (1996). Social network analysis: An approach and technique for the study of information exchange. *Library & Information Science Research*, *18*, 323–342.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*, 1735–1780.

Jain, A., Zamir, A. R., Savarese, S., & Saxena, A. (2016). Structural-RNN: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5308–5317).

Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., & Poupart, P. (2020). Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, *21*, 1–73.

Kennedy, K., Mac Namee, B., Delany, S. J., O'Sullivan, M., & Watson, N. (2013). A window of opportunity: Assessing behavioural scoring. *Expert Systems with Applications*, *40*, 1372–1380.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR)*.

Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J. P., Moreno, Y., & Porter, M. A. (2014). Multilayer networks. *Journal of Complex Networks*, *2*, 203–271.

Kumar, S., Zhang, X., & Leskovec, J. (2019). Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1269–1278).

Lee, J. W., Lee, W. K., & Sohn, S. Y. (2021). Graph convolutional network-based credit default prediction utilizing three types of virtual distances among borrowers. *Expert Systems with Applications*, *168*, 114411.

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*, 124–136.

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *6th International Conference on Learning Representations (ICLR)*.

Li, Z., Wang, X., Yao, L., Chen, Y., Xu, G., & Lim, E.-P. (2022). Graph neural network with self-attention and multi-task learning for credit default risk prediction. In *23rd International Conference on Web Information Systems Engineering – WISE 2022* (pp. 616–629).

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations (ICLR)*.

Lopez, J. A., & Saidenberg, M. R. (2000). Evaluating credit risk models. *Journal of Banking & Finance*, *24*, 151–165.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)* (pp. 4768—-4777).

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 1412–1421).

Mallinar, N., & Rosset, C. (2018). Deep canonically correlated LSTMs. *arXiv preprint arXiv:1801.05407*, .

Manessi, F., Rozza, A., & Manzo, M. (2020). Dynamic graph convolutional networks. *Pattern Recognition*, *97*, 107000.

Markov, A., Seleznyova, Z., & Lapshin, V. (2022). Credit scoring methods: Latest trends and points to consider. *The Journal of Finance and Data Science*, *8*, 180–201.

Matsuno, R., & Murata, T. (2018). MELL: Effective embedding method for multiplex networks. In *Companion Proceedings of the The Web Conference 2018* (pp. 1261–1268).

Mohajerin, N., & Waslander, S. L. (2017). State initialization for recurrent neural network modeling of time-series data. In *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 2330–2337).

Óskarsdóttir, M., & Bravo, C. (2021). Multilayer network analysis for improved credit risk prediction. *Omega*, *105*, 102520.

Óskarsdóttir, M., Bravo, C., Sarraute, C., Vanthienen, J., & Baesens, B. (2019). The value of big data for credit scoring: Enhancing financial inclusion using mobile phone data and social network analytics. *Applied Soft Computing*, *74*, 26–39.

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., & Ward, R. (2016). Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*, 694–707.

Pappas, N., & Popescu-Belis, A. (2017). Multilingual hierarchical attention networks for document classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 1015–1025).

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., & Leiserson, C. (2020). EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence (Volume 34)* (pp. 5363–5370).

Qu, Z., Haghani, P., Weinstein, E., & Moreno, P. (2017). Syllable-based acoustic modeling with CTC-SMBR-LSTM. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)* (pp. 173–177).

Roa, L., Correa-Bahnsen, A., Suarez, G., Cortés-Tejada, F., Luque, M. A., & Bravo, C. (2021). Super-app behavioral patterns in credit risk models: Financial, statistical and regulatory implications. *Expert Systems with Applications*, *169*, 114486.

Sankar, A., Wu, Y., Gou, L., Zhang, W., & Yang, H. (2019). Dynamic graph representation learning via self-attention networks. In *Workshop on Representation Learning on Graphs and Manifolds, ICLR 2019*.

Shanthamallu, U. S., Thiagarajan, J. J., Song, H., & Spanias, A. (2019). GrAMME: Semisupervised learning using multi-layered graph attention models. *IEEE Transactions on Neural Networks and Learning Systems*, *31*, 3977–3988.

Shi, S., Tse, R., Luo, W., D'Addona, S., & Pau, G. (2022). Machine learning-driven credit risk: A systemic review. *Neural Computing and Applications*, *34*, 14327—-14339.

Shu, J., Xi, B., Li, Y., Wu, F., Kamhoua, C., & Ma, J. (2022). Understanding dropout for graph neural networks. In *Companion Proceedings of the Web Conference 2022* (pp. 1128–1138).

Skarding, J., Gabrys, B., & Musial, K. (2021). Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*, *9*, 79143–79168.

Tang, Y., Huang, Z., Cheng, J., Zhou, G., Feng, S., & Zheng, H. (2021). Graph neural network-based node classification with hard sample strategy. In *2021 International Conference on Cyber-Physical Social Intelligence (ICCSI)* (pp. 1–4).

Thomas, L., Crook, J., & Edelman, D. (2017). *Credit scoring and its applications*. SIAM-Society for Industrial and Applied Mathematics.

Trivedi, R., Farajtabar, M., Biswal, P., & Zha, H. (2019). DyRep: Learning representations over dynamic graphs. In *7th International Conference on Learning Representations (ICLR)*.

Tsitsulin, A., Palowitch, J., Perozzi, B., & Müller, E. (2020). Graph clustering with graph neural networks. In *Proceedings of the 16th International Workshop on Mining and Learning with Graphs (MLG)*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *6th International Conference on Learning Representations (ICLR)*.

Vinciotti, V., Tosetti, E., Moscone, F., & Lycett, M. (2019). The effect of interfirm financial transactions on the credit risk of small and medium-sized enterprises. *Journal of the Royal Statistical Society Series A: Statistics in Society*, *182*, 1205–1226.

Wang, D., Zhang, Z., Zhao, Y., Huang, K., Kang, Y., & Zhou, J. (2023). Financial default prediction via motif-preserving graph neural network with curriculum learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 2233–2242).

Wang, J., Zhang, S., Xiao, Y., & Song, R. (2022). A review on graph neural network methods in financial applications. *Journal of Data Science*, *20*, 111–134.

Wang, S., Hu, L., Wang, Y., He, X., Sheng, Q. Z., Orgun, M. A., Cao, L., Ricci, F., & Yu, P. S. (2021). Graph learning based recommender systems: A review. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)* (pp. 4644–4652).

Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., & Chua, T.-S. (2017). Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)* (pp. 3119–3125).

Xu, D., Ruan, C., Korpeoglu, E., Kumar, S., & Achan, K. (2020). Inductive representation learning on temporal graphs. In *8th International Conference on Learning Representations (ICLR)*.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning (Volume 37)* (pp. 2048–2057).

Yang, S., Yu, X., & Zhou, Y. (2020). LSTM and GRU neural network performance comparison study: Taking Yelp review dataset as an example. In *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)* (pp. 98–101).

Yang, S., Zhang, Z., Zhou, J., Wang, Y., Sun, W., Zhong, X., Fang, Y., Yu, Q., & Qi, Y. (2021). Financial risk analysis for SMEs with graph-based supply chain mining. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence (IJCAI-20)* (pp. 4661–4667).

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)* (pp. 3634–3640).

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*.

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., & Li, H. (2020). T-GCN: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, *21*, 3848–3858.

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.

Appendix A. Statistics of the node features

Feature	Mean	Std. Dev.	Min.	Max.
fico	752.76	44.75	565	832
mi_pct	2.40	7.40	0	35
cnt_units	1.02	0.17	1	4
dti	33.61	11.15	1	65
ltv	69.30	16.07	7	97
cnt_borr	1.50	0.50	1	2
current_upb	173,036.60	97,258.30	13,829.33	716,617.50
mths_remng	304.58	65.55	73	574
current_int_rt	4.88	0.45	3.25	7.25

Table A.1: Descriptive statistics of the non-binary node features. For each loan's behavioural features ('current_upb', 'mths_remng', and 'current_int_rt), the maximum values over all monthly snapshots are considered.

Feature	0s	1s
if_fthb	128,131	20,389
if_prim_res	12,790	135,730
if_corr	89,602	58,918
if_sf	42,195	106,325
if_purc	95,380	53,140
if_sc	147,130	1,390
if_delq_sts	118,184	30,336
default	141,094	7,426

Table A.2: Frequency of the binary node features. For each loan, the maximum value of 'if_delq_sts' over all monthly snapshots is considered.

Appendix B. Architecture of the DNN baseline model

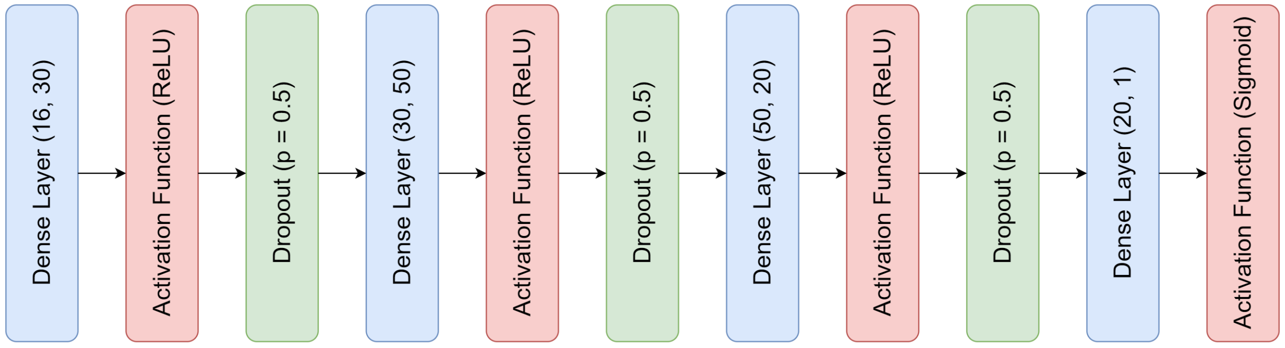


Figure B.1: Architecture of the DNN baseline model.

Appendix C. Hyperparameters for model training and computation resources

Hyperparameter	Value
Epochs	200
Early stop	50
Learning rate	0.001
Optimizer	Adam

Table C.1: Hyperparameters for model training.

Resource	Specification
Processor	AMD Milan 7413 @ 2.65 GHz 128M cache L3
CPU cores per task	2
GPU	NVidia A100
Memory per GPU	40 GB

Table C.2: Computation resources.