Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Inductive Graph Representation Learning for fraud detection

Rafaël Van Belle [*], Charles Van Damme, Hendrik Tytgat, Jochen De Weerdt

Research Centre for Information Systems Engineering (LIRIS), KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium

ARTICLE INFO

Keywords:
Inductive Representation Learning
Graph embeddings
Class imbalance
Fraud detection
GraphSAGE

ABSTRACT

Graphs can be seen as a universal language to describe and model a diverse set of complex systems and data structures. However, efficiently extracting topological information from dynamic graphs is not a straightforward task. Previous works have explored a variety of inductive graph representation learning frameworks, but despite the surge in development, little research deployed these techniques for real-life applications. Most earlier studies are restricted to a set of benchmark experiments, rendering their practical generalisability questionable. This paper evaluates the proclaimed predictive performance of state-of-the-art inductive graph representation learning algorithms on highly imbalanced credit card transaction networks. More specifically, we assess the inductive capability of GraphSAGE and Fast Inductive Graph Representation Learning in a fraud detection setting. Credit card transaction fraud networks pose two crucial challenges for graph representation learners: First, these networks are highly dynamic, continuously encountering new transactions. Second, they are heavily imbalanced, with only a small fraction of transactions labelled as fraudulent. This paper contributes to the literature by (i) proving how inductive graph representation learning techniques can be leveraged to enhance predictive performance for fraud detection and (ii) demonstrating the benefit of graph-level undersampling for representation learning in imbalanced networks.

1. Introduction

In the field of computer science and mathematics, graphs are used as ubiquitous data structures. Many domains ranging from disease gene networks to communication networks are mathematically represented using graphs, making them the backbone of numerous systems. Consequently, machine learning on graphs creates enormous potential for a variety of tasks such as node classification, structural hole detection and link prediction (Hamilton et al., 2017b). Graphs are, however, high-dimensional, non-Euclidean structures that cannot easily be processed by off-the-shelve machine learning models. Therefore, graph representation learning algorithms are used to generate low-dimensional numeric vectors that represent the original network structure (Cui et al., 2018). Previous years have witnessed the success of graph representation learning algorithms in achieving the aforementioned machine learning tasks (Ahmed et al., 2020; Guo et al., 2018; Rossi et al., 2018). Additionally, multiple recent studies reveal the success of inductive graph representation learning (IGRL) (Liang et al., 2018; Liu et al., 2019; Liu, Xie et al., 2020; Yang et al., 2016). Motivated by nearly $25 billion in losses due to credit card fraud worldwide (SHIFT, 2018) and building on previous work by Van Belle et al. (2019) and Van Vlasselaer et al. (2015), this research investigates the combination of embedded credit card network data and machine learning to detect fraudulent transactions in an inductive setting effectively. More specifically, we investigate how features originating from the network of transactions, rather than intrinsic transaction features, can benefit fraud detection. This is researched by applying two promising inductive graph representation learning techniques: GraphSAGE (Hamilton et al., 2017a) and Fast Inductive Graph Representation Learning, hereafter referred to as FI-GRL (Jiang et al., 2018). The embeddings generated by these algorithms are leveraged to help detect fraud. Concretely, this paper addresses the following research questions:

RQ1 How do state-of-the-art inductive graph representation learning techniques perform for fraudulent transaction classification?

RQ2 What is the empirical effect of employing node embeddings on the classification performance of fraud detection on a real world fraud dataset? Put differently, can the classification performance of a fraud detection model be improved by adding node embeddings to the original set of transaction features.

RQ3 What is the empirical effect of graph-level undersampling on the quality of inductively learned embeddings and associated classification performance for a highly imbalanced real world fraud classification dataset?

* Corresponding author.
E-mail addresses: rafael.vanbelle@kuleuven.be (R. Van Belle), charles.vandamme@student.kuleuven.be (C. Van Damme), hendrik.tytgat@student.kuleuven.be (H. Tytgat), jochen.deweerdt@kuleuven.be (J. De Weerdt).

https://doi.org/10.1016/j.eswa.2021.116463
Received 9 June 2020; Received in revised form 10 December 2021; Accepted 23 December 2021
Available online 10 January 2022

Please note that the conclusions presented in this paper are confined to the specific dataset used. Public, labelled fraud datasets are rare and hence, reaching general conclusions in this domain is particularly arduous. Nevertheless, we employ real fraud data labelled by domain experts. The number of transactions, both licit and illicit, increases our confidence in the broader relevance of the results.

The remainder of this paper is organised as follows: Section 2 discusses related work on graph representation learning techniques and fraud detection. Section 3 explains the inner working of the two selected IGRL algorithms. Section 4 describes the experimental setup and Section 5 evaluates the results. Finally, Section 6 concludes this paper by providing an answer to the research questions and discusses further research.

2. Related work

Machine learning models work efficiently for feature vectors and hence tabular data. Unlike these data formats, networks have complex topological structures, no fixed reference point or node ordering, and dynamic features (Leskovec et al., 2018). This discrepancy has been addressed with the introduction of graph representation learning techniques which transform networks into vector space representations. These methods try to capture the relational structure of a graph in a Euclidean vector space (Hamilton et al., 2017b). In this work, we focus on the task of node classification which aims to predict the label of nodes based on a combination of feature and label information of other nodes.

2.1. Graph representation learning

Learning node embeddings is a branch of graph representation learning conceived for the task of node classification. Goyal and Ferrara (2018) suggest three broad categories of node embedding techniques: (1) matrix factorisation, (2) random walk based, and (3) deep learning.

Matrix factorisation techniques require a matrix that summarises node connectivity (e.g. adjacency matrix, Laplacian) to extract a latent vector (node embeddings) for each node in the network. These techniques are often referred to as spectral given that they rely on the eigenanalysis for studying properties of the graph (Chung & Graham, 1997). Well-known factorisation techniques include Graph Factorisation, GraRep and HOPE (Ahmed et al., 2013; Cao et al., 2015; Ou et al., 2016). Factorisation techniques have, on average, a higher time complexity in comparison with other graph embedding techniques. In addition, the majority of these techniques only learns on static networks and, hence, are inherently transductive.

Random walk based techniques, such as Deepwalk (Perozzi et al., 2014) and Node2Vec (Grover & Leskovec, 2016), perform a series of truncated random walks which are used to train a shallow single-layer neural network. The neural network is optimised to maximise the probability of observing nodes encountered in close proximity to each other on a random walk (Perozzi et al., 2014). In contrast to the spectral matrix factorisation based approaches, random walk techniques are considered to be spatial representation learners, to the extent that embeddings are created based on neighbourhood information extracted from the walks. However, recent work by Levy and Goldberg (2014) has shown that the Skipgram algorithm underlying Deepwalk and related approaches is an implicit matrix factorisation, blurring the lines between spectral and spatial representation learning. Much like matrix factorisation, most random walk techniques are restricted to transductive learning. In addition, random walk based techniques are unsupervised: label and feature information is typically not exploited for learning the embeddings.

The recent surge in deep learning research has also explored graph data. The most important contribution is Graph Convolutional Networks (GCN) (Defferrard et al., 2016; Kipf & Welling, 2016). An aggregator function iteratively aggregates feature information from neighbouring nodes. In contrast with matrix factorisation techniques, GCNs do not rely on the adjacency matrix and only need neighbourhood information, making them more efficient on large and sparse graphs. Unlike random walk techniques, GCNs rely heavily on node features and they support supervised learning.

2.2. Transductive and inductive representation learning

Node embedding algorithms can also be classified based on their capacity to handle unseen nodes. Two categories can be discerned: transductive and inductive algorithms. Currently, most graph representation learning techniques belong to the transductive class. These techniques require the presence of the entire graph during node embedding generation, which makes it challenging to generalise towards unseen nodes.

The limitations of transductive approaches in a fraud detection application are twofold. First, transductive techniques in a highly dynamic transaction network need to continuously re-train to learn embeddings for unseen nodes (Hamilton et al., 2017b), when fraud detection applications require fast solutions that are able to predict fraud quickly. Second, most transductive techniques leave out node and edge features, which are important network characteristics that have been relatively successful in traditional fraud detection (Leskovec et al., 2018).

As opposed to transductive techniques, inductive representation learners do not require re-training for unseen nodes. For these previously unobserved nodes, inductive algorithms have the ability to *induce* the embedding based on their position in the network. In dynamic graphs, such as credit card transaction networks, this inductive characteristic is of utmost importance to achieve operational scalability. Without this inductive capability, the graph representation learning algorithm would have to reprocess the entire network for each new transaction that appears. Therefore, techniques that need to translate dynamic networks into node embeddings for a high-throughput fraud detection application require an inductive approach (Hamilton et al., 2017b). Originally, Graph Convolutional Networks (GCNs) (Defferrard et al., 2016; Kipf & Welling, 2016) lacked the ability for inductive learning. Hamilton et al. (2017a) introduced GraphSAGE: a GCN framework with a neighbourhood sampling layer allowing for inductive learning. For matrix factorisation algorithms, dealing with unseen nodes is even harder because altering the adjacency matrix after completing the factorisation is intricate. Fast Inductive Graph Representation Learning (FI-GRL) (Jiang et al., 2018) is one of the few matrix factorisation embedding techniques capable of inductive learning. Random walk techniques that support inductive learning are also scarce. SPINE (Guo et al., 2018) and Planetoid (Yang et al., 2016) are two representative examples.

2.3. From graph representation learning to fraud detection

Fraud is still an important issue causing financial losses for many organisations worldwide. The advent of e-commerce and the declining popularity of cash has triggered an increase in digital transactions which are susceptible to fraud (European Central Bank, 2020). Machine learning is a highly relevant domain for fraud detection (Bolton & Hand, 2002; Fawcett & Provost, 1997). Both supervised and unsupervised techniques have been investigated for fraud detection (Ryman-Tubb et al., 2018). Supervised fraud detection is characterised by a high complexity originating from the numerous legitimate compared to fraudulent transactions. This creates a severe class imbalance and makes detection and machine learning challenging. Further approaches addressing the high class imbalance problem range from data level methods (e.g. over- and undersampling) to algorithm level methods (such as cost-sensitive and hybrid/ensemble learning) (Leevy et al., 2018). Despite being a critical topic that has gained widespread attention, the number of works published on fraud detection with machine learning is fairly limited. The most commonly deployed techniques for fraud detection are Bayesian networks, decision trees, and artificial neural networks (Bahnsen et al., 2013; Dal Pozzolo et al., 2018; Ryman-Tubb, 2016).

An effective fraud detection system has to look beyond individual transactions (Bolton & Hand, 2002). Hence, most research focuses on historical transaction patterns of customers or peer groups. More recent work has explored the relationships between multiple customers and their respective transactional history. Simply put, the network representing the transactions between cardholders and merchants has become an important artefact in recent work on supervised fraud detection. While unsupervised fraud detection often relied on the transaction network for anomaly detection, supervised fraud detection employing network information is a more recent phenomenon (Pourhabibi et al., 2020).

One of the first studies to use network information for fraud detection in a supervised context is APATE (Van Vlasselaer et al., 2015). The authors propose an extensive feature engineering approach which contains personalised Pagerank scores extracted from the transaction network. These scores and other generated features are then used as input for a downstream classifier. The APATE approach has been further refined by Courtain et al. (2019) and Lebichot et al. (2016).

Feature engineering is a tedious and error-prone process which often requires substantial domain expertise. Developing features from network data requires both knowledge of the fraud and network domain. Hence, to alleviate this problem, graph representation learning techniques propose an automated procedure to extract relational features from an input graph. Van Belle et al. (2019) applied a random walk based node embedding technique, called Node2Vec (Grover & Leskovec, 2016) to extract node embeddings which were subsequently used in a downstream classifier for fraud detection. Several difficulties arise when applying shallow, random walk based node embedding algorithms: (1) transductive learning, (2) scalability, (3) exclusion of label and intrinsic features, (4) unsupervised, resulting in a two-step approach for supervised fraud detection (Hamilton et al., 2017b).

Graph Neural Networks solve many of the aforementioned issues. Recently, there has been an increase in the research combining graph neural networks and fraud detection. Liu, Dou et al. (2020) introduced important inconsistencies that arise when applying convolutional graph neural networks for fraud detection. The authors introduce a custom GCN framework that tackles the inconsistencies through various modifications. Dou et al. (2020) noticed that fraudsters tend to camouflage themselves in order to avoid raising suspicion. The authors improve the GCN aggregation process by introducing a label-aware similarity measure which is used for neighbour selection. In addition, reinforcement learning is leveraged to determine the optimal number of neighbours. In similar vein, Cheng et al. (2020) introduce a framework consisting of a GNN layer, spatial–temporal attention layer and multiple 3D convolutional layers. The different components of the framework are optimised in an end-to-end way.

In sum, inductive graph representation learning techniques hold potential for fraud detection. Two graph representation learning algorithms were selected based on three essential requirements for fraud detection: scalability,[1] ability to generalise towards unseen nodes, and inductive speed. As a result we investigate two inductive representation learning techniques with different characteristics: GraphSAGE and FI-GRL. All of the aforementioned research on graph convolutional networks for fraud detection are custom designed GCN frameworks which often have not been tested on large networks. In contrast, GraphSAGE (Hamilton et al., 2017a), which is an extension of the GCN framework, has been evaluated on large graphs (Ying et al., 2018) and, in addition, supports inductive learning. Matrix factorisation techniques have been largely ignored in recent years for fraud detection, for reasons such as poor scalability, computational burden and the inability to work with dynamic graphs. FI-GRL (Jiang et al., 2018) is an exception and presents a fast, scalable and inductive matrix factorisation representation learning framework. A final reason for choosing these specific algorithms is their difference in the graph analysis paradigm they represent. GraphSAGE draws from spatial graph theory, while FI-GRL relies on graph Laplacian factorisation which makes it a spectral algorithm.

[1] **Scalability:** the computational complexity concerning the number of nodes in the training step.

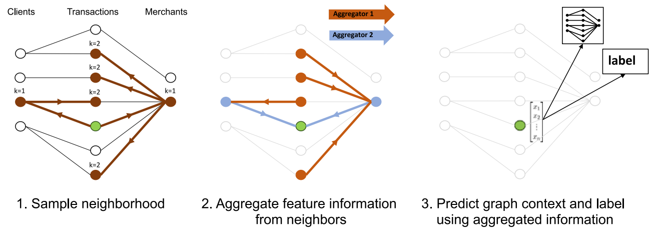


Fig. 1. Embedding generation for the unseen green node using the GraphSAGE framework (Hamilton et al., 2017a).

3. Inductive graph representation learning

This section discusses how GraphSAGE and FI-GRL approach network representation and why it is interesting to compare the performance of their inductive embeddings for node classification.

3.1. GraphSAGE

GraphSAGE, short for graph sample and aggregate, leverages node features to learn both the distribution of features in a particular node's local neighbourhood as well as the network structure. In essence, GraphSAGE trains a set of functions that aggregate the acquired knowledge about the surrounding feature information of a node's neighbourhood. During inference time, these learned aggregation functions are applied to generate embeddings for unseen nodes (Hamilton et al., 2017a).

At its core, GraphSAGE assumes that nodes with similar neighbourhoods in the original network should have similar embeddings. The intuition behind GraphSAGE is shown in Fig. 1. In a first step, a fixed-size set of neighbours is sampled per neighbourhood k, $\forall k \in \{1, \ldots, K\}$. The depth and number of neighbours to sample in this illustration are respectively set to 2 and $\{2, 3\}$. The second step aggregates the feature information from the sampled neighbours by applying aggregation functions, which were learned and optimised for each depth k during training time. Hamilton et al. (2017a) propose in their paper three types of aggregation functions: mean, LSTM, and pooling. GraphSAGE aggregates the representations of the nodes in its immediate neighbourhood first. Next, GraphSAGE iterates over nodes per neighbourhood depth k while incrementally adding information from the further neighbourhood with the aggregation function. This allows the embedding algorithm to gain structural insights of the graph. The third and last step is feeding the resulting numeric vector, or node embedding, to off-the-shelve machine learning models for a variety of tasks. However, before learning any inductive node embeddings, the aggregation functions have to be trained. These functions are part of a neural network that is controlled by either a supervised or unsupervised loss function and learning rate. When embeddings are learned in a completely unsupervised setting, the graph-based loss function is applied. Intuitively, this loss function encourages neighbouring nodes to have similar embeddings while it makes the embeddings of distant nodes different. Stochastic Gradient Descent is used to optimise the parameters of the neural network, and at the same time, to optimise the hyperparameters of the selected aggregation functions, which form a part of this neural network. The graph-based loss function in the

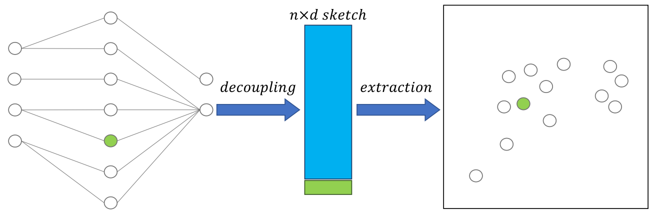


Fig. 2. A high-level illustration of embedding generation for the unseen green node using the FI-GRL framework (Jiang et al., 2018).

unsupervised setup can easily be replaced with cross-entropy loss if the node embeddings have to be optimised for a downstream task, such as (supervised) fraud detection (Hamilton et al., 2017a).

In their paper, Hamilton et al. (2017a) perform inductive learning for multi-class node classification with GraphSAGE on two rapidly evolving graphs: a Web of Science (WoS) citation network consisting of 302,423 nodes and a Reddit posts network containing 232,965 nodes. GraphSAGE outperformed a raw features baseline and DeepWalk (Perozzi et al., 2014) for these classification tasks based on micro-averaged F1 scores[2] Hamilton et al. (2017a). Given GraphSAGE's success on these inductive node classification tasks, our research investigates if this performance holds on binary classification in highly imbalanced credit card transaction networks with millions of nodes and edges.

3.2. Fast inductive-graph representation learning

Jiang et al. (2018) propose FI-GRL as a fast and flexible node embedding framework that tries to preserve graph topological information. This framework follows a two-stage approach: decoupling and feature extraction. Decoupling starts from a graph $\mathcal{G}$, creating a matrix sketch $M \in \mathbb{R}^{n\times d}$ where d is the dimension of the sketch size and n the number of nodes in the original graph. In the matrix sketch M, $d \ll n$, and $d \& n$ are automatically determined. M approximates the original graph well with theoretical guarantees. The decoupling stage is designed to separate the nodes' relations by utilising the Johnson–Lindenstrauss random projection (Jiang et al., 2018). The result of this random projection is embedding points of a high-dimensional space into a much lower dimensional space while almost perfectly preserving the distance between the points (Cohen et al., 2015). After the decoupling, the feature extraction stage solves a minimisation problem of a cost function that calculates the fit between the original graph $\mathcal{G}$ and the matrix sketch M, given the constraint that M has a reduced rank. Intuitively, this second stage extracts meaningful features contained in the matrix sketch by using low-rank approximation (a minimisation problem, measuring the fit between the approximating matrix and the given matrix). Dimension reduction is achieved in both of these two stages. The resulting representations are theoretically guaranteed[3] and perform well on constrained low-rank approximation tasks such as k-means clustering (Jiang et al., 2018).

The inductive step of this algorithm is visualised in Fig. 2. FI-GRL generalises to an inductive setting by utilising a fold-in technique on the matrix sketch M to learn the embedding of the new node. This fold-in technique is the process of adding the compressed vector of the new node to the pre-calculated matrix sketch M. Specifically, a normalised random walk matrix is created out of the new adjacency matrix, and from this normalised random walk matrix, the row vector of the new node is extracted. When combining this with the Johnson–Lindenstrauss random projection, the new compressed vector is created. The extraction stage now applies the low-rank approximation to map this compressed vector to the embedding space (Jiang et al., 2018).

The empirical study of Jiang et al. (2018) contains extensive experiments of both clustering and structural hole detection applications. These experiments compare the FI-GRL framework to GraphSAGE (Hamilton et al., 2017a), Deepwalk (Perozzi et al., 2014), node2vec (Grover & Leskovec, 2016) and LINE (Tang et al., 2015). In addition, FI-GRL is matched against structural hole detection methods like PageRank. All networks used in the empirical study were undirected, homogeneous graphs widely varying in network type and scale. The first comparison of FI-GRL was performed on a clustering task using evaluation metrics modularity[4] and permanence.[5] While Jiang et al. (2018) demonstrated FI-GRL's superiority in terms of the training runtime and aforementioned evaluation metrics, they did not thoroughly compare FI-GRL's inductive performance against other inductive graph representation learning methods. Additionally, the most extensive graph used for the inductive learning setting contained only 23,133 nodes. However, FI-GRL is theoretically guaranteed to transform the topological structure of a graph into a low dimensional space without losing essential information (Jiang et al., 2018). Given the framework's theoretical basis, our research investigates whether the performance holds in a realistic environment with a highly imbalanced fraud dataset. Furthermore, we analyse the inductive capability of FI-GRL and evaluate how it compares against GraphSAGE.

3.3. Fraud detection: GraphSAGE vs. FI-GRL

GraphSAGE and FI-GRL are both inductive graph representation learning algorithms that differ fundamentally in how they approach the problem. GraphSAGE uses a feature-oriented approach in which it first samples from a node's neighbourhood and then proceeds to aggregate the observed information. FI-GRL, on the other hand, applies matrix reduction techniques in two stages, namely decoupling and feature extraction. These matrix reduction techniques are unable to exploit node features, causing FI-GRL's resulting embeddings to be solely based on the network structure. Nevertheless, both algorithms are inductive, scalable, and generate node embeddings that capture structural network information, which can be leveraged for our fraud detection application.

In their paper, Hamilton et al. (2017a) demonstrate GraphSAGE's outstanding performance concerning node classification problems. Our research investigates whether this performance scales to highly imbalanced credit card transaction networks and tests the quality of the resulting supervised embeddings for fraud detection. Since FI-GRL operates in a fully unsupervised manner, the empirical study by Jiang et al. (2018) does not include node classification. Hence, FI-GRL is unable to exploit fraud labels in creating distinct embeddings for fraudulent and genuine transactions. This, combined with the aforementioned inability to leverage node features, provides a unique setting to test whether purely topological node embeddings contain novel information when used for fraud detection. In addition, selecting these algorithms allows to compare the predictive quality of embeddings generated by supervised and unsupervised graph representation learners.

[2] **Micro-Averaged F1 Scores:** F1 Scores are the weighted average of the precision and recall. Micro-averaged F1 scores denote the aggregated contributions of all classes to calculate the average F1 Score.

[3] **Theoretically guaranteed:** Jiang et al. (2018) mathematically prove to extract meaningful information out of the matrix sketch M.

[4] **Modularity:** high modularity means that the resulting clustering has sparse inter-cluster connections but dense intra-cluster connections.

[5] **Permanence:** permanence is based on two factors (1) the number of intra-cluster connections of the vertex versus inter-cluster connections, and (2) how dense the vertex is connected internally (Chakraborty et al., 2016).

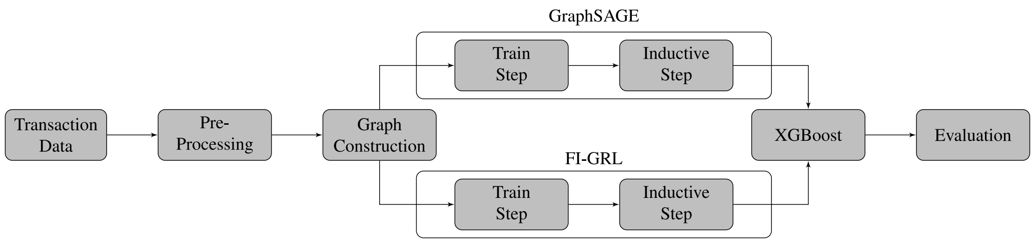


Fig. 3. Visualisation of experimental pipeline described in Section 4.

4. Experimental setup

Our experimental setup is designed with the goal of (i) evaluating the inductive performance of FI-GRL and GraphSAGE for fraud detection and (ii) investigating the influence of undersampled input graphs on the predictive quality of the inductively generated embeddings. To draw an objective comparison between both representation learners, as many factors as possible are kept constant. Concretely, FI-GRL and GraphSAGE learn on identical input graphs, and an identical downstream classifier evaluates their inductive embeddings. The experimental pipeline is visualised in Fig. 3 and our code is publicly available.[6]

4.1. Data

The real-life dataset used to construct our credit card transaction networks is provided by a credit card issuer and contains 3,724,348 transactions. This dataset includes information on the following features: anonymised identification of clients and merchants, merchant category code, country, monetary amount, time, acceptance,[7] and fraud label. The aforementioned number of transactions resulted from the interaction between 1,325,070 clients and 144,439 merchants gathered over five weeks during October–November 2013. This real-life dataset is highly imbalanced and contains only 0.65% fraudulent transactions.

4.2. Pre-processing

The second step in the pipeline is transforming the input data into a compatible format for the downstream graph representation learning algorithm. Specifically, the features needed to be transformed into a numeric format. The timestamp attribute was reshaped into separate year, month, day, hour, minute and second attributes. Furthermore, the acceptance and fraud label attributes were converted into binary values. An additional transformation was the creation of dummy variables for the country attribute. After these pre-processing steps, the number of attributes increased from 8 to 214 numeric variables.

To improve the robustness of our results, we devised a manual 5-fold out-of-time validation, by dividing the dataset based on a rolling window approach. Fig. 4 illustrates how the dataset was split up into five smaller timeframes. Each window has a size of 17 days, with the start date of each window five days apart. Within particular windows, a train–test split was made: the first twelve days were used to train the representation learners, leaving the last five days to test the inductive capability of the algorithms.

[6] https://github.com/Charlesvandamme/Inductive-Graph-Representation-Learning-for-Fraud-Detection.

[7] **Acceptance:** a boolean variable indicating whether a transaction was accepted based on a number of sanity checks.

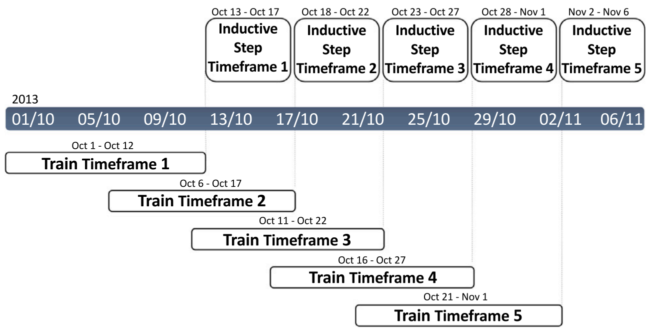


Fig. 4. Rolling window visualisation of train–test split per timeframe, with the test set used for the inductive step. For each timeframe, 12 days of transaction data is used for training and 5 days for testing.

Table 1
Number of transactions and fraud distribution in the training and test sets for each timeframe.

	Train		Test (Inductive)	
	Transactions	Fraud	Transactions	Fraud
Timeframe 1	1,203,442	0.57%	501,601	0.38%
Timeframe 2	1,205,046	0.46%	501,461	0.32%
Timeframe 3	1,204,232	0.36%	508,861	0.97%
Timeframe 4	1,211,352	0.60%	505,898	1.16%
Timeframe 5	1,215,582	0.97%	503,085	0.61%

4.3. Graph construction

The next step in the pipeline is constructing the graphs that will be used by FI-GRL and GraphSAGE to learn node embeddings. We designed the credit card transaction networks as heterogeneous tripartite graphs containing client, merchant, and transaction nodes, following the tripartite graph construction approach of Van Belle et al. (2019), Van Vlasselaer et al. (2015) (see Fig. 5). Because of this tripartite setup, representations can be learned for the transaction nodes. Client and merchant nodes are determined by their anonymised identification. The 212 remaining attributes, hereafter referred to as the original transaction features, reside with the corresponding transaction nodes. A graph was constructed for each timeframe and used as input to the graph representation learning algorithm. Table 1 illustrates the number of transactions and fraud distribution in the training and test sets for each timeframe. The percentage fraud fluctuates substantially between timeframes, characterising an additional difficulty to obtain stable results. Given the highly imbalanced nature of our dataset, we hypothesised that without sampling the graph representation learning algorithms will have a blurred view on the structure of fraudulent transactions. To test this hypothesis, we fed graphs with varying sampling rates to the representation learners.

4.4. GraphSAGE configuration

The aforementioned graphs are fed to GraphSAGE to learn embeddings of transaction nodes. Since our credit card transaction networks

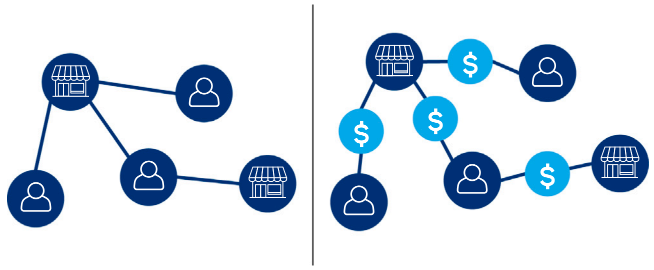


Fig. 5. Example of bipartite transaction network (left) and tripartite transaction network (right). The bipartite network contains cardholder and merchant nodes connected with transaction edges. The tripartite network has cardholder, merchant and transaction node types.

Table 2
Hyperparameter settings for GraphSAGE in our experimental setup. Depth K defines the extent of the random walks, the neighbours define the number of neighbours to sample in each neighbourhood and the final dimension represents the size of the resulting embeddings.

	Depth (K)	Neighbours	Final dimension
GraphSAGE	2	{2,32}	64

are heterogeneous graphs, we used HinSAGE as a heterogeneous implementation of the GraphSAGE framework, which is part of the Python StellarGraph package (Data61, 2018). The implementation of this algorithm requires all nodes in the graph to have features, which is why we added a dummy feature with value 1 to all client and merchant nodes. The original transaction features described in the graph construction section remain at the transaction nodes. The supervised version of HinSAGE was deployed and trained with a binary cross-entropy loss function. The training records in Fig. 4 were further divided into a training and validation set. Concretely, the first ten days were used for training, leaving the last two days for validation. The out-of-time test set was not used for training of the HinSAGE model. In our setup, HinSAGE was implemented with Tensorflow using the Adam optimiser, a 0.001 learning rate, and a batch size of 50.

Hyperparameters. Table 2 summarises the hyperparameter settings for GraphSAGE. The depth K of random walks and number of neighbours to sample per $k, \forall k \in \{1,\dots,K\}$ are respectively set to 2 and {2,32}. Hamilton et al. (2017a) report in their parameter sensitivity analysis that a depth of 2 provides the best trade-off between accuracy and time complexity. Only two neighbours were sampled in the first neighbourhood (clients and merchants) since each transaction has exactly one client and one merchant. Up to 32 nodes were sampled in the second neighbourhood (transactions) because we observed diminishing returns for sampling a larger number of transactions. The mean aggregator, which is currently the only supported aggregator in StellarGraph's HinSAGE implementation, was selected.

Inductive step. The inductive step uses the mean aggregation functions, optimised during training, to induce embeddings of size 64 for the unseen transactions when they are added to the graph.

4.5. FI-GRL configuration

FI-GRL is based on matrix multiplication. Hence, the algorithm requires the input graph to be homogeneous and without features. Therefore, the heterogeneous input graphs had to be transformed into a matrix-compatible format, which means that the alphanumerical node identifiers in the original graph needed to be transformed into consecutive integers. This transformation gave each node a unique integer that defined its position in the network's adjacency matrix. Our setup used the current publicly available Matlab distribution of FI-GRL (Jiang et al., 2018).

Table 3
Hyperparameter configuration for the FI-GRL framework, following Formula (1). The intermediate dimension d, approximation ratio ϵ and final dimension of the resulting embeddings k.

	Intermediate (d)	ϵ	Final dimension (k)
FI-GRL	400	0.4	64

Hyperparameters. The first hyperparameter of FI-GRL is the intermediate dimension (i.e., the dimension of the matrix sketch M). This dimension can be mathematically derived as follows:

$$d = max\left\{\frac{4\log(n)}{\epsilon}, \frac{k}{\epsilon^2}\right\} \tag{1}$$

where the constant n represents the number of nodes in the train graph, ϵ the desired approximation ratio, and k the final dimension of the node embeddings. When 2 out of 3 hyperparameters are set (d, k, and ϵ), the third is determined by Formula (1). The lower the approximation ratio, the higher the accuracy of the low-rank approximation problem. However, this also significantly increases the computational complexity (Jiang et al., 2018). After a thorough sensitivity analysis for the FI-GRL configuration, we set the intermediate dimension for the configurations to achieve an approximation ratio of 0.4 and a final embedding size of 64. This approximation ratio captures the macroscopic structure of the network while still delivering scalable results. Table 3 summarises the hyperparameter settings used for FI-GRL in our experiments.

Inductive step. The inductive step uses the during training established matrix sketch M with intermediate dimension size d to induce embeddings of size 64.

4.6. XGBoost: From embedding to prediction

The penultimate component in the experimental pipeline before evaluation is a machine learning algorithm that classifies transactions based on the generated node embeddings. Previous graph representation learning research by Grover and Leskovec (2016) and Mitrović et al. (2019) reported promising results with logistic regression. However, Van Belle et al. (2019) showed in a similar fraud detection setup that eXtreme Gradient Boosting, or XGBoost, performs well on numeric vectors that represent the network structure. Based on this finding, we select XGBoost as the machine learning classifier for our pipeline. Besides, we considered applying oversampling with Adaptive Synthetic Sampling Approach for Imbalanced Learning (He et al., 2008) on XGBoost level using the embeddings generated on non-sampled graphs as the original data. However, we observed a preliminary negative trend on predictive performance and computational efficiency.

Configuration. To achieve an objective comparison between the two graph representation learning algorithms, no XGBoost hyperparameter tuning was performed. XGBoost was used with default settings: logistic regression for binary classification objective function, a learning rate of 0.3, and 100 boosting rounds. The evaluation metrics for fitting our XGBoost classifier were the mean average precision (MAP), the area under the precision–recall curve (PR-AUC), and the binary classification error rate (error). The embeddings generated by the representation learners were used to train and evaluate XGBoost using the same data split as in Fig. 4. The Section 5 thoroughly evaluates the predictive quality of the inductive embeddings based on the output of XGBoost.

4.7. Evaluation

Given our highly imbalanced data, the resulting XGBoost predictions were evaluated on Precision–Recall (PR) curves rather than on the more common Receiver Operating Characteristic (ROC) plots. The area under these ROC curves (ROC-AUC) assesses the overall classification performance, which renders the metric less sensitive than the area under the PR curve (PR-AUC) to accurately classify the minority class (Davis & Goadrich, 2006). A brief illustration shows the importance of working with PR curves in this context: suppose that two algorithms try to find a total of 3000 fraud cases in a network of 3 million transactions. Imagine algorithms A and B respectively classify 3000 and 60,000 transactions as fraudulent, of which 2400 are correct. The first algorithm achieves superior results compared to the second, by returning fewer false positives. However, because the number of legitimate transactions is so large, this difference is mostly lost with the ROC measures of TPR and FPR[8] (true and false positive rate). Precision and recall are not impacted by true negatives, making them more robust to the class imbalance.[9] This example is based on a specific threshold in the PR and ROC space. Nonetheless, if these differences persist over all thresholds, the resulting AUC difference for ROC curves will be much smaller than for their PR counterparts, making the latter more suitable for evaluation purposes in a fraud detection context.

The importance of achieving a low number of false positives stems from the limited capacity within the credit card industry to inspect flagged fraud cases (Van Belle et al., 2019). This limited capacity causes the industry to review only a top percentile of the predicted fraud probabilities. Lift scores define how many more fraud cases are present in a specific percentile when applying the prediction model, using the class distribution as a reference point. Therefore, the resulting XGBoost predictions were also evaluated on the top 1% Lift scores.

5. Results

This section investigates the predictive performance of inductive graph representation learning for fraud detection using the aforementioned experimental setup.

5.1. Predictive value of inductive embeddings

Fig. 6 depicts the average PR curves over the five timeframes for predictions made with XGBoost on GraphSAGE and FI-GRL inductive embeddings. GraphSAGE is evaluated twice: once on the original graphs and once on graphs where the transaction features are replaced with network information (degree, average neighbour degree, and PageRank).[10] The latter configuration allows testing GraphSAGE's embedding quality when trained with topological network information only. Note that the PR curve of predictions made with FI-GRL embeddings lies significantly lower than the *GraphSAGE with original features* curve. This difference could be due to three main limitations: First, since FI-GRL is inherently unsupervised, the algorithm is unable to make purposely different representations for fraudulent and legitimate transactions. Second, the algorithm cannot leverage the difference between client, merchant, and transaction nodes as it treats our credit card transaction networks as homogeneous graphs. Third, FI-GRL does not leverage the original transaction features. Because of these constraints, the algorithm's node embeddings are solely based on the network's topological information. This, in combination with both GraphSAGE curves outperforming a no skill classifier (majority class prediction), demonstrates the predictive value of inductively generated node embeddings.

Each of FI-GRL's mentioned limitations could explain why *GraphSAGE with original features* performed significantly better on precision

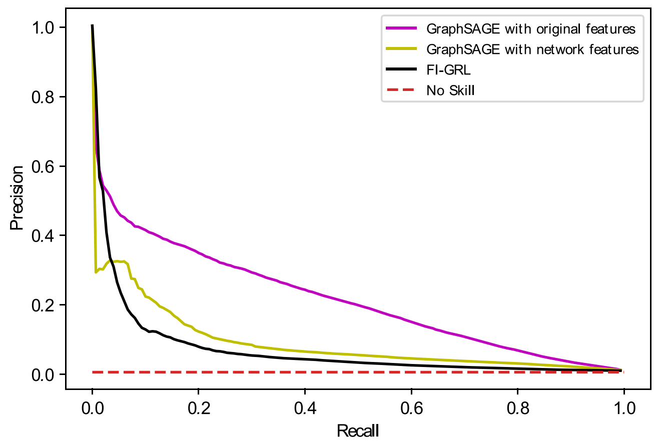


Fig. 6. Average PR curves of XGBoost predictions over five timeframes on FI-GRL and GraphSAGE's inductive embeddings. GraphSAGE generates embeddings on two types of input graphs: one where transaction nodes have network features (degree, average neighbour degree and PageRank) and one where they have original transaction features. To contextualise, a no skill classifier that labels all instances as non-fraud is added.

Table 4
Average and standard deviation of PR-AUC and 1% Lift, for XGBoost predictions over five timeframes, on the original transaction features (Baseline) and on the combination of these features with the inductive embeddings of GraphSAGE and FI-GRL. The two rightmost columns represent the % gain of the inductive embeddings with the original transaction features over the baseline.

	Baseline	GraphSAGE	FI-GRL	% gain w.r.t baseline	
				GraphSAGE	FI-GRL
PR-AUC	0.196 ± 0.1139	0.276 ± 0.1704	0.270 ± 0.1497	**40.81%**	**37.78%**
1% Lift	32.86 ± 5.4248	39.96 ± 7.9849	38.29 ± 6.2436	**21.61%**	**16.52%**

and recall. However, when we replace the original transaction node features of GraphSAGE's input graphs, its PR curve drops dramatically. This reduction suggests that the algorithm's superior performance stems mainly from its ability to leverage the original transaction features, rather than its heterogeneous and supervised capacity. The next subsection evaluates the importance of transaction features and their complementarity to an inductively learned embedding.

5.2. Novelty of inductive embeddings

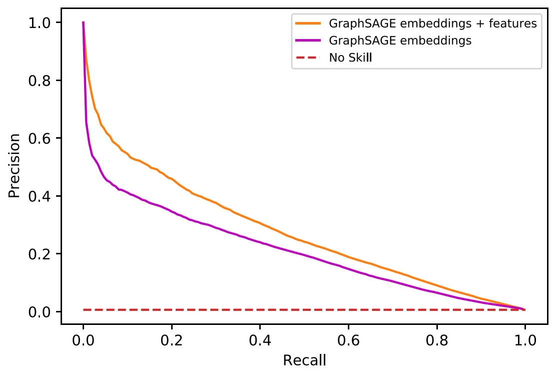
Considering that the inductive embeddings of both algorithms have predictive value, we research if these embeddings are complementary to the original transaction features. Fig. 7a exhibits the difference between, on the one hand, GraphSAGE embeddings and, on the other hand, the combination of GraphSAGE embeddings with these transaction features. Fig. 7b depicts the same situation for FI-GRL embeddings. Both figures demonstrate that predictions become more accurate when the inductive embeddings are combined with transaction features. This increased performance illustrates the usefulness of the original transaction features to predict fraudulent transactions. However, when the transaction features are added to the inductive embeddings, FI-GRL obtains a more substantial performance gain than GraphSAGE, which already leverages these features. This indicates the importance of the underlying transaction features for fraud detection and raises a question: do inductive embeddings capture valuable information that is not yet reflected in the original transaction features?

Fig. 7c answers this question by comparing the average PR curves of inductive embeddings merged with the original transaction features to a baseline. The baseline is the situation where XGBoost only uses the original transaction features as input, without embeddings. This comparison shows the net impact of the network structure on predictions. We observe that the combination of transaction features and inductively generated embeddings by FI-GRL and GraphSAGE score strictly

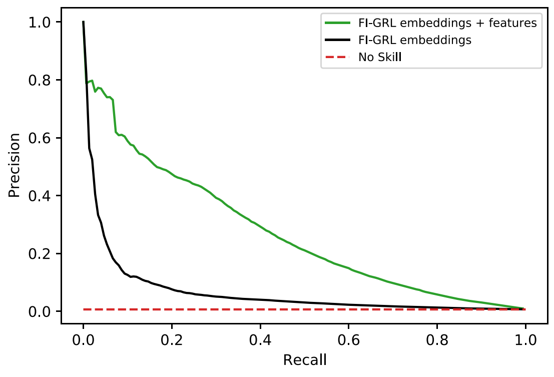
[8] **ROC measures:** A: TPR = 0.8, FPR = 0.0002; B: TPR = 0.8, FPR = 0.0192 (difference of 0.019).

[9] **PR measures:** A: recall = 0.8, precision = 0.8; B: recall = 0.8, precision = 0.04 (difference of 0.76).

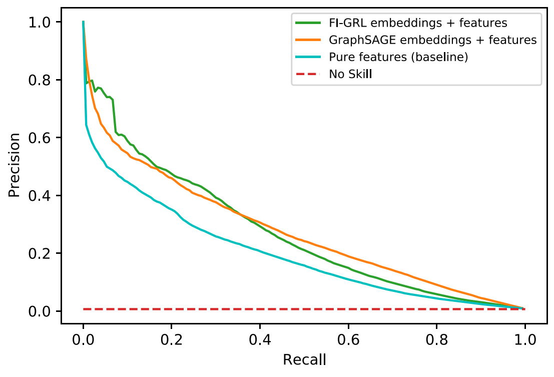
[10] The degrees and PageRanks are initially calculated for the train graphs and updated when new transactions are added in the inductive step.



(a) GraphSAGE embeddings and GraphSAGE embeddings plus the original transaction features. GraphSAGE is trained with the original features in both cases. In the setting 'GraphSAGE + features' the original features are concatenated with the trained embeddings and used in the downstream XGBoost classifier.



(b) FI-GRL embeddings and FI-GRL embeddings plus the original transaction features. FI-GRL makes no use of the original features. In the setting 'FI-GRL + features' the original features are concatenated with the trained FI-GRL embeddings and used in the downstream XGBoost classifier.



(c) GraphSAGE and FI-GRL's inductive embeddings plus the original transaction features. The baseline represents the predictions on the original transaction features, without embeddings.

Fig. 7. Average PR curves of XGBoost predictions over five timeframes. To contextualise, a no skill classifier that labels all instances as non-fraud is added. '+ `features`' indicates that the original features are concatenated with the embeddings generated by the corresponding embedding technique before being utilised by the XGBoost classifier.

higher than the baseline. Even when adding a few simple network features such as degree, average neighbour degree, and PageRank score, the representation learning-based approaches outperform our baseline. Table 4 depicts the average PR-AUC and 1% Lift along with their standard deviations for both algorithms and the baseline. Looking at percentage differences, we conclude that GraphSAGE, in combination with the original transaction features, outperforms the baseline PR-AUC and Lift with respectively 40.81% and 21.61%. Similarly, FI-GRL plus the original transaction features exceeds our baseline on these metrics with 37.78% and 16.52%. Hence, we observe little impact of the (un)supervised characteristic on embeddings generated by graph representation learners for our fraud detection setup. High standard deviations are a result of the PR-AUC metric's sensitivity to the class imbalance of our inductive sets (see Table 1). Despite this high variability, the PR curves of the representation learners strictly outperformed the baseline for each timeframe. These results prove that the node embeddings of both algorithms contain novel information that enhances the predictive performance of XGBoost. However, as many new features are provided to the classifier, the curse of dimensionality could occur. Considering the machine learning criterion of having at least five training examples for each feature in the representation (Theodoridis et al., 2010), our setup can exclude this dimensionality concern.[11] In conclusion, the experiments confirm that inductively learned embeddings contain novel information not yet captured by the original transaction features, therefore making them useful in a fraud detection application.

5.3. Graph-level undersampling

The presence of class imbalance has a significant influence on the predictive performance of classifiers and the favourable effects of sampling imbalanced datasets in traditional data mining have been observed frequently (Lauron & Pabico, 2016). Therefore, our research hypothesised that without sampling, inductive graph representation learning algorithms are restrained in learning the structure of the minority class. To test this hypothesis, we perform undersampling,[12] which has proven to be an efficient and straightforward method for class-imbalance learning (Liu et al., 2009). Concretely, this means that both FI-GRL and GraphSAGE train on undersampled input graphs before generating embeddings for unseen transactions using the inductive mechanism. Fig. 8a depicts the average PR-AUC over five timeframes for embeddings plus features and our baseline with sampling rates between 0.1 and 0.5. Note that two curves represent GraphSAGE: *GraphSAGE full graph* is the setting in which the transactions used for training the algorithm's aggregation functions are sampled, but the random walks can still access the entire network. *GraphSAGE limited graph* is the setting where the graphs used for training are sampled, containing only the sampled transactions along with their clients and merchants. Through comparison against a baseline of only original transaction features, the net value of embeddings can be analysed. Surprisingly, we observe that all configurations exhibit a decreasing trend, set by the baseline. This suggests that there is no net improvement in embedding quality for large sampling rates. An explanation for this general diminishing trend could be the design of the PR-AUC metric, which is sensitive to the number of false positives. However, FI-GRL's curve demonstrates an initial increase, contrary to the overall declining tendency.

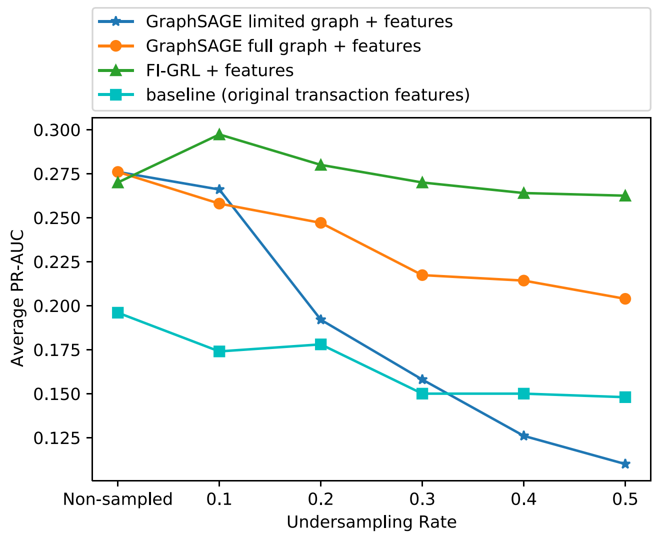
To analyse this initial improvement, Fig. 8b illustrates the same situation for smaller sampling rates. On the one hand, *GraphSAGE full graph* curve remains unaffected by small sampling rates, as the random walks still aggregate information from the entire network. On the other hand, with a sampling rate of 0.02, both FI-GRL and *GraphSAGE limited graph* exhibit substantial improvement over the non-sampled setup. Specifically, Table 5 depicts that with a 0.02 undersampling rate, the representation learning-based approaches obtain an increase of approximately 10% on PR-AUC and Lift@1% over their non-sampled counterparts. Considering that the baseline's performance remained constant, this increase is solely due to the enhanced quality of inductive

[11] **Curse of Dimensionality:** 212 original transaction features plus 64 node embedding features with on average 1,207,931 train transactions per timeframe equals an average of 4376 training examples per feature.

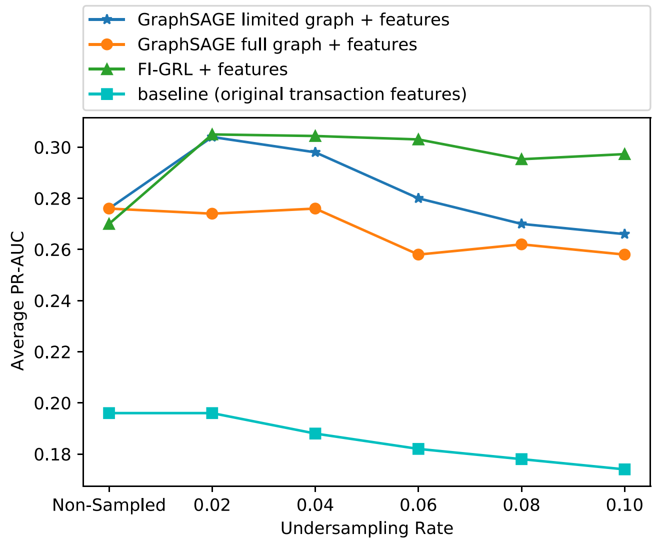
[12] **Undersampling:** a random subgroup of the majority class is ignored to achieve a new predefined and more balanced class ratio in the training set.

Table 5
Average and standard deviation of PR-AUC and 1% Lift, for XGBoost predictions over five timeframes, with a 2% undersampling rate on the original transaction features (Baseline) and the combination of these features with the inductive embeddings of *GraphSAGE limited graph* and FI-GRL. The two rightmost columns represent the % gain of the representation learning-based approaches over their non-sampled counterparts mentioned in Table 4.

	Baseline	GraphSAGE	FI-GRL	% gain w.r.t Non-sampled	
				GraphSAGE	FI-GRL
PR-AUC	0.196 ± 0.1103	0.304 ± 0.1704	0.305 ± 0.1499	**10.14%**	**12.96%**
1% Lift	33.94 ± 5.1924	43.17 ± 7.0518	42.27 ± 5.2436	**8.03%**	**10.39%**



(a) High-level representation of undersampling trend.



(b) Detailed representation of the samping rates between non-sampled and 0.1.

Fig. 8. Average PR-AUC of XGBoost predictions over five timeframes for increasing sampling rates on FI-GRL and GraphSAGE's inductive embeddings with transaction features, and the baseline of the original transaction features. GraphSAGE is used to generate embeddings twice: once receiving full input graphs to train on and once with networks of only the sampled transactions along with their clients and merchants.

embeddings. These findings show that small undersampling rates at graph-level increase the predictive quality of inductive embeddings generated for our highly imbalanced network. Additionally, representation learning on undersampled graphs is more efficient thanks to the reduction of the total network size.

5.4. Time complexity

In a real-world fraud detection application, fast processing of incoming transactions is of high importance. Credit card issuers are necessitated to decide whether a transaction is fraudulent in less than ten milliseconds (Ravenscraft, 2016). Table 6 summarises the runtimes of both representation learners. On average, graphs used during training have 1,936,759 nodes and 2,415,867 edges, while the inductive sets contain approximately 504,181 transactions.

A first observation is that FI-GRL's training time is well beneath GraphSAGE. However, in a fraud detection context, rather than training time, the time delay for inductive embedding generation is decisive. Considering that credit card transactions are processed on a per transaction basis, the most relevant column is the runtime for individual embedding generation. As a result of this, GraphSAGE outperforms FI-GRL and generates the embedding for an unseen transaction well below the time limit of 10 ms. Based on the increased runtime from one node to a batch of approximately 500,000, we observe an inductive time complexity of $O(n)$ for GraphSAGE and $O(\log n)$ for FI-GRL. However, the absolute training and inductive runtime of both algorithms is highly influenced by the hyperparameter configuration (Tables 2 and 3). GraphSAGE's ability to quickly generate embeddings for new transactions makes the algorithm a more realistic representation learner for a fraud detection system. While FI-GRL does not meet the time limit set by the credit card industry, the algorithm's rapid batch processing speed can be highly desired in other application fields such as graph representation learning in protein networks.

Hardware. For all experiments, a single shared Windows server with an Intel Xeon$^{®}$ (E5 - 2699 v3 @ 2,3 GHz) and 256 GB of RAM was used. The limiting factor to running the experiments on a local machine is RAM rather than CPU power.

6. Conclusion and further research

In this paper, two state-of-the-art inductive graph representation learning algorithms were applied to highly imbalanced credit card transaction networks. GraphSAGE and Fast Inductive Graph Representation Learning were juxtaposed against each other to evaluate the predictive value of their inductively generated embeddings for a fraud detection application. In answer to **RQ1** and **RQ2**, our results show that these embeddings provide novel information and significantly improve fraud detection when combined with the original transaction features. Concretely, we achieved a mean increase in average precision of approximately 40% with a non-optimised classifier. Additionally, the experiments reconfirm the importance of original transaction features for credit card fraud detection. In response to **RQ3**, the results demonstrate a general negative effect of graph-level undersampling on the predictive performance of embeddings generated by GraphSAGE and FI-GRL for our fraud dataset. However, we observed an increase of roughly 10% in average precision for small sampling rates, demonstrating that undersampling can be beneficial for graph representation learners both in terms of predictive performance and efficiency. Finally, GraphSAGE proved to be the most realistic representation learner for a fraud detection application, generating embeddings for new transactions in less than 3 ms. Future research possibilities for

Table 6
Average and standard deviation of runtime for GraphSAGE and FI-GRL, over five timeframes, during the training and inductive step, with the latter divided into batch and individual processing. Train graphs have an average size of 1,936,759 nodes and 2,415,867 edges. The inductive batch contains an average of 504,181 transactions.

	Train time	Inductive time	
		Batch	Individual
GraphSAGE	17,774.94 s ± 1,074.52 s	717.72 s ± 59.61 s	0.0027 s ± 0.0001 s
FI-GRL	564.18 s ± 48.26 s	60.32 s ± 1.99 s	3.05 s ± 0.25 s

taking this work further could focus on: (1) discovering the optimal graph-level sampling rate and technique for a given class imbalance; (2) investigating how the combination of transaction features and inductive embeddings compete against fraud detection techniques in the industry; (3) generalising the power of inductive graph representation learning techniques by deploying them for other highly imbalanced networks such as the detection of malicious websites; (4) researching how the time dimension can be leveraged for fraud detection by, for instance, designing GraphSAGE's random walks to be time-dependent; (5) analysing the effect of extensive feature engineering in heterogeneous networks on embeddings generated by feature-oriented graph representation learners.

CRediT authorship contribution statement

Rafaël Van Belle: Conceptualization, Methodology, Software, Data curation, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Charles Van Damme:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Visualization. **Hendrik Tytgat:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Visualization. **Jochen De Weerdt:** Conceptualization, Methodology, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

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.

Acknowledgements

The authors would like to thank Dr. Sandra Mitrović for her knowledgeable feedback on the research paper. In addition, we acknowledge the support given by the Research Fund - Flanders (FWO) (Grant No. 11H0219N).

References

Ahmed, N., Rossi, R. A., Lee, J., Willke, T., Zhou, R., Kong, X., & Eldardiry, H. (2020). Role-based graph embeddings. *IEEE Transactions on Knowledge and Data Engineering*.
Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., & Smola, A. J. (2013). Distributed large-scale natural graph factorization. In *WWW '13, Proceedings of the 22nd international conference on world wide web* (pp. 37–48). New York, NY, USA: Association for Computing Machinery.
Bahnsen, A. C., Stojanovic, A., Aouada, D., & Ottersten, B. (2013). Cost sensitive credit card fraud detection using Bayes minimum risk. In *2013 12th international conference on machine learning and applications, Vol. 1* (pp. 333–338). ieeexplore.ieee.org.
Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. *Statistical Science*, *17*(3), 235–255.
Cao, S., Lu, W., & Xu, Q. (2015). GraRep: Learning graph representations with global structural information. In *CIKM '15, Proceedings of the 24th ACM international on conference on information and knowledge management* (pp. 891–900). New York, NY, USA: Association for Computing Machinery.
Chakraborty, T., Srinivasan, S., Ganguly, N., Mukherjee, A., & Bhowmick, S. (2016). Permanence and community structure in complex networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *11*(2), 1–34.
Cheng, D., Wang, X., Zhang, Y., & Zhang, L. (2020). Graph neural network for fraud detection via spatial-temporal attention. *IEEE Transactions on Knowledge and Data Engineering*.
Chung, F. R. K., & Graham, F. C. (1997). *Spectral graph theory*. American Mathematical Soc..
Cohen, M. B., Elder, S., Musco, C., Musco, C., & Persu, M. (2015). Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on theory of computing* (pp. 163–172).
Courtain, S., Lebichot, B., Kivimäki, I., & Saerens, M. (2019). Graph-based fraud detection with the free energy distance. In *International conference on complex networks and their applications* (pp. 40–52). Springer.
Cui, P., Wang, X., Pei, J., & Zhu, W. (2018). A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, *31*(5), 833–852.
Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2018). Credit card fraud detection: A realistic modeling and a novel learning strategy. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(8), 3784–3797.
CSIRO's Data61 (2018). StellarGraph machine learning library. https://github.com/stellargraph/stellargraph.
Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In W. W. Cohen, & A. W. Moore (Eds.), *ACM international conference proceeding series: vol. 148, Machine learning, proceedings of the twenty-third international conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006* (pp. 233–240). ACM.
Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. arXiv preprint arXiv:1606.09375.
Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., & Yu, P. S. (2020). Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *CIKM '20, Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 315–324). New York, NY, USA: Association for Computing Machinery.
European Central Bank (2020). *Sixth report on card fraud: Technical report*, European Central Bank.
Fawcett, T., & Provost, F. (1997). Adaptive fraud detection. *Data Mining and Knowledge Discovery*, *1*(3), 291–316.
Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, *151*, 78–94.
Grover, A., & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864).
Guo, J., Xu, L., & Liu, J. (2018). Spine: structural identity preserved inductive network embedding. arXiv preprint arXiv:1802.03984.
Hamilton, W. L., Ying, R., & Leskovec, J. (2017a). Inductive representation learning on large graphs. arXiv preprint arXiv:1706.02216.
Hamilton, W. L., Ying, R., & Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. arXiv preprint arXiv:1709.05584.
He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)* (pp. 1322–1328). IEEE.
Jiang, F., Zheng, L., Xu, J., & Yu, P. (2018). Fi-grl: Fast inductive graph representation learning via projection-cost preservation. In *2018 IEEE international conference on data mining (ICDM)* (pp. 1067–1072). IEEE.
Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
Lauron, M. L. C., & Pabico, J. P. (2016). Improved sampling techniques for learning an imbalanced data set. arXiv preprint arXiv:1601.04756.
Lebichot, B., Braun, F., Caelen, O., & Saerens, M. (2016). A graph-based, semi-supervised, credit card fraud detection system. In *International workshop on complex networks and their applications* (pp. 721–733). Springer.
Leevy, J. L., Khoshgoftaar, T. M., Bauder, R. A., & Seliya, N. (2018). A survey on addressing high-class imbalance in big data. *Journal of Big Data*, *5*(1), 1–30.
Leskovec, J., Hamilton, W. L., & R. Ying, R. S. (2018). Representation learning on networks. (pp. 1–15).
Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 27* (pp. 2177–2185). Curran Associates, Inc..
Liang, J., Jacobs, P., Sun, J., & Parthasarathy, S. (2018). Semi-supervised embedding in attributed networks with outliers. In *Proceedings of the 2018 SIAM international conference on data mining* (pp. 153–161). SIAM.

Liu, Z., Dou, Y., Yu, P. S., Deng, Y., & Peng, H. (2020). Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 1569–1572).

Liu, X., Hsieh, P.-C., Duffield, N., Chen, R., Xie, M., & Wen, X. (2019). Real-time streaming graph embedding through local actions. In *Companion proceedings of the 2019 world wide web conference* (pp. 285–293).

Liu, X., Wu, J., & Zhou, Z. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550.

Liu, X., Xie, M., Wen, X., Chen, R., Ge, Y., Duffield, N., & Wang, N. (2020). Micro-and macro-level churn analysis of large-scale mobile games. *Knowledge and Information Systems*, *62*(4), 1465–1496.

Mitrović, S., Baesens, B., Lemahieu, W., & De Weerdt, J. (2019). Tcc2vec: RFM-informed representation learning on call graphs for churn prediction. *Information Sciences*, 1–16.

Ou, M., Cui, P., Pei, J., Zhang, Z., & Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *KDD '16*, *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1105–1114). New York, NY, USA: Association for Computing Machinery.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710).

Pourhabibi, T., Ong, K.-L., Kam, B. H., & Boo, Y. L. (2020). Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, Article 113303.

Ravenscraft, E. (2016). This is why your credit card transactions take so long to clear. https://www.lifehacker.com.au/2016/09/this-is-why-your-credit-card-transactions-take-so-long-to-clear/. Accessed: 2021-4-18.

Rossi, R. A., Zhou, R., & Ahmed, N. K. (2018). Deep inductive network representation learning. In *WWW '18*, *Companion proceedings of the the web conference 2018* (pp. 953–960). Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee.

Ryman-Tubb, N. (2016). *Understanding payment card fraud through knowledge extraction from neural networks using large-scale datasets* (Ph.D. thesis), University of Surrey.

Ryman-Tubb, N. F., Krause, P., & Garn, W. (2018). How Artificial Intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark. *Engineering Applications of Artificial Intelligence*, *76*, 130–157.

SHIFT (2018). Credit card fraud statistics. https://shiftprocessing.com/credit-card/. Accessed: 2021-4-18.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077).

Theodoridis, S., Pikrakis, A., Koutroumbas, K., & Cavouras, D. (2010). *Introduction to pattern recognition: a matlab approach*. Academic Press.

Van Belle, R., Mitrović, S., & De Weerdt, J. (2019). Representation learning in graphs for credit card fraud detection. In *Workshop on mining data for financial applications* (pp. 32–46). Springer.

Van Vlasselaer, V., Bravo, C., Caelen, O., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2015). A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, *75*, 38–48.

Yang, Z., Cohen, W., & Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In M. F. Balcan, & K. Q. Weinberger (Eds.), *Proceedings of machine learning research*: *vol. 48*, *Proceedings of the 33rd international conference on machine learning* (pp. 40–48). New York, New York, USA: PMLR.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for Web-Scale recommender systems. In *KDD '18*, *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 974–983). New York, NY, USA: Association for Computing Machinery.