CATCHM: A novel network-based credit card fraud detection method using node representation learning

Rafaël Van Belle[a,*], Bart Baesens[a,b], Jochen De Weerdt[a]

[a]*Faculty of Business and Economics, KU Leuven, Naamsestraat 69, 3000 Leuven, Belgium*
[b]*Southampton Business School, University of Southampton, Highfield, Southampton SO17 1BJ, UK*

Acknowledgments

This research was supported by the Research Foundation Flanders [Grant number 180319] and was financed in part by the EC H2020 MSCA RISE NeEDS Project [Grant agreement ID: 822214].

*Corresponding author
Email address: rafael.vanbelle@kuleuven.be (Rafaël Van Belle)

CATCHM: A novel network-based credit card fraud detection method using node representation learning

Abstract

Advanced fraud detection systems leverage the digital traces from (credit-card) transactions to detect fraudulent activity in future transactions. Recent research in fraud detection has focused primarily on data analytics combined with manual feature engineering, which is tedious, expensive and requires considerable domain expertise. Furthermore, transactions are often examined in isolation, disregarding the interconnection that exists between them.

In this paper, we propose *CATCHM*, a novel network-based credit card fraud detection method based on representation learning (RL). Through innovative network design, an efficient inductive pooling operator, and careful downstream classifier configuration, we show how network RL can benefit fraud detection by avoiding manual feature engineering and explicitly considering the relational structure of transactions. Extensive empirical evaluation on a real-life credit card dataset shows that *CATCHM* outperforms state-of-the-art methods, thereby illustrating the practical relevance of this approach for industry.

Keywords: Network Representation Learning, DeepWalk, Credit Card Fraud, Fraud Detection

1. Introduction

The advent of e-commerce and digital payment solutions has profoundly changed the way we pay for goods and services. The number of credit and debit card transactions rose dramatically in recent decades and this has attracted criminals [1, 2]. In response to this threat, financial institutions have tried to improve security measures to prevent fraud and, deploy systems to detect fraud

[3]. Automated fraud detection originally relied on domain expert knowledge to create a set of differentiating rules [4]. More recently, machine learning (ML) has been applied to mine fraudulent patterns in transaction logs of financial organizations [5, 6, 7]. To date, however, research in fraud detection has been confided to either domain expertise or extensive feature engineering, both of which have important limitations in terms of classification performance, operational efficiency, maintainability, and cost. This is exactly what we aim to address in this paper.

Concretely, this work contributes to the literature by proposing *CATCHM*, an entirely novel fraud detection algorithm that relies on customized network representation learning. We demonstrate that accurate fraud detection can be achieved without the aforementioned domain knowledge or manual feature engineering. A successful fraud detection system (FDS) should be capable of dealing with three crucial challenges. First, fraud is *hidden*. Only a fraction of transactions is fraudulent, which results in a severe class imbalance [3, 8]. Globally, 6.86 dollar cents per 100$ are lost due to fraud, while in the SEPA zone, approximately 0.24% of all transactions turn out to be fraudulent [1]. This means that there are very few examples of fraudulent activity to learn from, which, in turn, can cause a high number of false positives (legitimate transactions falsely tagged as fraud). Second, fraud is *dynamic*. Fraudsters adapt quickly to new security measures and detection methods [9, 3]. Novel modi operandi will emerge quickly and fraud detection systems should adapt accordingly. Third, *time is scarce*. Increases in the volume of transactions[1], changing legislation and rising customer expectations have led to a dramatic decrease in payment processing times [10, 11]. Currently, a transaction can be authorized in less than one hundred milliseconds [12, 13]. As a result, an FDS has to scale and be fast. *CATCHM* addresses these challenges by means of three crucial elements. First, we propose an innovative tripartite network design including an artificial node to directly incorporate fraud information available

[1] Visa network: 188B transactions in 2020, Mastercard network: 113B in 2020

in training data, which strongly impacts the quality of the node embeddings obtained from the representational learner. Second, we present an inductive extension for random-walk based network representation learning. In this way, a transaction can be processed quickly, and operational time requirements can be met. Third, we propose an optimal downstream classifier optimization through careful hyperparameter tuning and model stacking.

In an extensive experimental evaluation, *CATCHM* is shown to outperform relevant benchmarks on actual credit card fraud data, containing more than three million transactions. More specifically, we demonstrate how our solution can be deployed satisfying the stringent operational time constraints while concurrently optimizing prediction quality metrics important to financial organizations.

The remainder of this article is structured as follows. After introducing related work in Section 2, *CATCHM* is described in Section 3. Sections 4 and 5 describe the benchmarks and the experimental design respectively, before the results are presented in Section 6. The paper is concluded in Section 7.

2. Related Work

In this section, we provide an overview of related work in terms of (1) general (supervised) machine learning techniques for fraud detection, (2) network-based fraud detection, and (3) graph representation learning. To conclude this section, we clarify the research gap and our contribution.

2.1. Machine Learning for Fraud Detection

Thanks to the digitization of the payment process, more data are being stored for each transaction. These data enable statistical learning techniques, including ML, to discern genuine and fraudulent transactions. The most common ML technique in the fraud detection literature is neural networks (NNs). Initially, research was focused on data preprocessing and in particular, feature engineering [6, 14, 15]. In the last decade, the focus shifted to the architecture,

and novel NN designs for fraud detection were introduced [16, 17, 18, 19]. Decision tree (DT) algorithms have also been used for fraud detection [5, 9, 20, 21]. An important advantage of DTs is that the models are relatively straightforward to explain and can easily be interpreted as a rule set. Hence, DTs can augment existing rules from domain experts. To deal with challenges such as concept drift and the volume of transactions, scholars have experimented with novel DT algorithms that allow for online learning [21]. Researchers have also studied evolutionary computing techniques [22, 23], Bayesian networks [7, 24] and support vector machines [25]. For a more in-depth overview and analysis of ML-based fraud detection methods, we refer to [8].

2.2. Social Network-driven Fraud Detection

Fraudsters often cooperate and organize in gangs [26]. Moreover, a single fraud case can span multiple transactions, involving more than one cardholder and/or merchant [27], resulting in a complex network of interactions underlying each fraud case. While anomaly detection in graphs has been considered for fraud detection [28], our work focuses on supervised learning. In this context, a prime focus is on feature engineering via social network analysis (SNA). Social network analysis investigates social structures using graph theory and provides a wide array of network metrics and algorithms that are used to derive a set of features summarizing the network topology [29]. These features are subsequently used to train a machine learning classifier for the task of fraud detection. For example, in [30], egonet features and metrics from shortest paths and strongly connected components were used to create a fraud detection model. In [31], a heterogeneous network consisting of vertices representing users, reviews and products was characterized via metapath analysis for spam detection. With auction fraud, the authors of [32] found that fraudsters tended to have more intense links with their neighbors. Hence, they used the weighted degree centrality as an indicator for fraudulent behavior.

Algorithms are often required to calculate more intricate measures characterizing a network. For fraud detection, collective classification algorithms have

been used frequently, particularly propagation algorithms. Propagation algorithms rely on the network topology to spread a fraud signal originating from confirmed fraud nodes across the network. The amount of signal received by each node can then be used as a suspicion score. In [33], the authors used a Markov random field to create a semi-supervised algorithm for fraud detection. An adaptation of the belief propagation algorithm the incorportation of transaction information and observed fraud labels into the optimization of the Markov random field. Personalized PageRank (PPR) is also a propagation algorithm used frequently for fraud detection. PPR calculates a score for each node indicating its proximity to a set of source nodes, in the case of fraud detection, confirmed fraud transactions act as source nodes. In [34], the birank algorithm, an adaptation of PageRank for bipartite graphs, was used to detect insurance fraud. In the same vein, the authors of [35] used PageRank to detect social insurance fraud. In [36], Personalized PageRank was applied specifically for credit card fraud detection. The technique in [36] has been improved with (1) a regularized commute time kernel to enhance the treatment of hub nodes in the network [2] [37], (2) a feedback loop to include the results from fraud investigators in the PageRank model [37] and (3) the free energy distance measure in [38].

2.3. Representation Learning in Graphs

Various studies illustrated how the network topology can be used for fraud detection. The key challenge is to transform the transaction network into a format suitable for vector-based machine learning algorithms without being negatively affected by the challenges introduced in Section 1. Recently, this has been addressed by scholars in new ways, which has led to the introduction of novel techniques such as graph neural networks and matrix factorization algorithms [39, 40, 41], collectively called network representation learning (NRL) techniques [42]. NRL transforms network vertices into latent, low-dimensional vector representations, which are optimized to preserve the original network

[2] Nodes with substantial first-order neighborhoods.

topology, node attributes and auxiliary information [42].

Given the breadth of the research field, this study will focus on a subfield of NRL, namely, random walk-based network representation algorithms, such as DeepWalk [43] and Node2Vec [44]. DeepWalk [43] is a technique inspired by the field of natural language processing (NLP). It uses the Word2Vec model [45], which transforms words into vectors by analyzing the sentences in which they appear. In DeepWalk, each node is considered a 'word' and 'sentences' are generated by performing random walks in the network. Through optimization, DeepWalk provides a latent, continuous, vector representation for each node in the original network.

Network representation learning has also proven useful for fraud detection in insurance [27], mobile advertising [46], online fraud [47] and transaction fraud [48]. Despite these studies showing promising results, no studies have addressed the challenges associated with credit card fraud detection. In particular, existing techniques are transductive, label information is not incorporated into the network topology and concept drift is not treated adequately. In addition, classification performance is reported using default classification metrics, ignoring the fact that fraud detection is a business process, for which performance is often measured differently. These gaps are specifically addressed in this work.

2.4. Our Contribution

Fraud detection faces three crucial challenges: fraud is hidden, fraud is dynamic, and detection time is scarce. These challenges give rise to a few important, industry-recognized, limitations of currently applied, state-of-the-art methods in terms of classification performance, operational efficiency, maintainability, and costs.

Classification performance. First, the dynamic nature of fraud causes novel fraud patterns (modi operandi) to emerge more frequently, reducing the detection accuracy of expert-based models over time [8, 28].

Operational efficiency. Efficiency is key in card transactions [13]. Machine learning models for fraud detection rely on extensive feature engineering that scales poorly [28]. Likewise, traditional graph-based models rely on complex aggregated network features that require offline calculation, hampering real-time fraud detection [9].

Maintainability. Third, the dynamic nature of fraud demands frequent model updates. This is cumbersome for rule-based models with expansive rule bases, as the impact of adding or altering rules becomes unclear [8].

Cost. Finally, contemporary fraud detection systems suffer from gradually declining accuracy, inadequate operational efficiency, and poor maintainability. This eventually results in increased costs for the system owner, despite cost being a major driver within the payments industry [3, 8]. In addition, both expert-based rule creation and feature engineering are time-consuming and labor intensive [8, 28].

In this work, we present *CATCHM*, a fraud detection algorithm that directly addresses the limitations listed above. First, *CATCHM* is rooted in graph-based ML and can reveal hidden, previously unknown fraud patterns. Second, *CATCHM* is equipped with a clever inductive pooling operator and allows for near real-time application, avoiding expensive feature computation or even the need to retrain models instantly, thus meeting operational constraints. Third, by relying on automated feature engineering, maintainability is drastically improved, since the models can be retrained, redeployed, and managed with much less effort compared to rule-based expert systems. Finally, *CATCHM* avoids costly manual rule creation and feature engineering.

In Section 5.2, the challenges of classification performance and operational efficiency are examined more thoroughly. In addition, relevant metrics to quantify and compare the performance of *CATCHM* on both challenges are introduced (see Section 6).

CATCHM is part of a larger body of work in the area of network representation learning-based fraud detection [49, 50, 51]; however, it differs in various aspects from this previous work. In [49, 50], the first attempts at creating inductive solutions, including a nearest-neighbor approach, were introduced. In [51], we compared contemporary inductive graph neural networks, including GraphSAGE.

3. Network Representation Learning for Fraud Detection

In this section, we present *CATCHM*, our fraud detection algorithm based on network representation learning in greater detail (see Figure 1). We discuss (1) network design, (2) the inductive extension for predicting unseen nodes, and (3) downstream classifier optimization. The implementation is available on Github.[3]

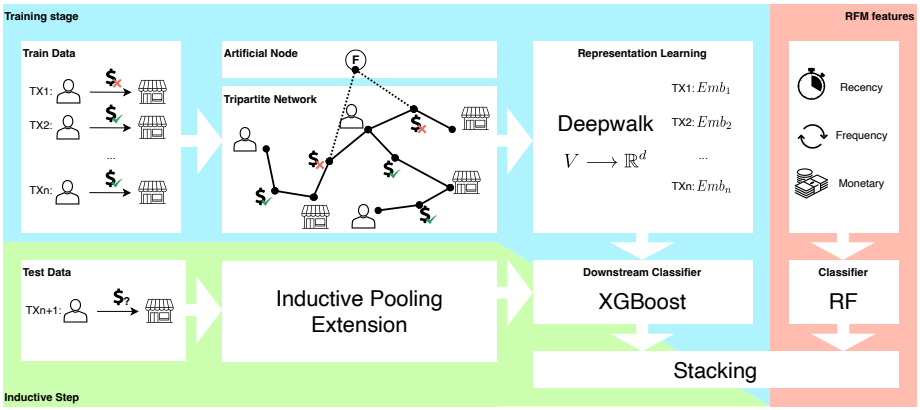

Figure 1: Schematic overview of *CATCHM*. In the training stage, input transaction data are transformed into a tripartite transaction network. Fraudulent transactions are connected to an additional artificial fraud node. Next, a representation learning algorithm is applied to the network to obtain a set of node embeddings. These embeddings are used to train an XGBoost classification model. In the inductive step, each entry in the test data is parsed through the inductive pooling extension, which generates a new embedding based on the tripartite network and embeddings from the training data. Optionally, RFM features are used to train a separate random forest classifier. Predictions for both classifiers are combined by stacking.

[3] `URLHIDDENFORDOUBLEBLIND`

3.1. Network Design

3.1.1. Tripartite Network

Every (credit-card) transaction has two parties involved: a cardholder and a merchant. Intuitively, both parties can be represented as a vertex (or node) in a network. An edge between two parties indicates that a transaction took place between the cardholder and the merchant. This setup is known as a bipartite graph (see Figure 2). The network has two distinct types of nodes (cardholders and merchants), and edges can only exist between different types of nodes. In general, network representation learning yields an embedding for each vertex (node) in the network, leaving the transaction edges without embedding. While techniques exist to combine node embeddings into an edge embedding, this would result in identical embeddings for all transactions sharing the same cardholder and merchant, when not all those transactions have identical class labels.

A different approach is to create a graph with a single node type. Then, nodes depict transactions, and two transactions are connected if they share the same merchant and/or cardholder. This network design, however, leads to an extremely connected or dense network, which increases the computational burden of (certain) RL algorithms considerably.

We propose a tripartite network design in which cardholders, merchants and transactions are modeled as distinct node types. This design ensures that we obtain embeddings for cardholders, merchants and transactions while keeping the network sparsely connected. The edges in the tripartite network are undirected and unweighted.

3.1.2. Artificial Node

A second modification to the typical transaction network design is the addition of a single artificial node. The authors of [52] illustrated how artificial nodes can be used to inject attribute information into the network and influence the learned node representations. The representation learning algorithm used in *CATCHM*, DeepWalk [43], is an unsupervised algorithm. For DeepWalk to

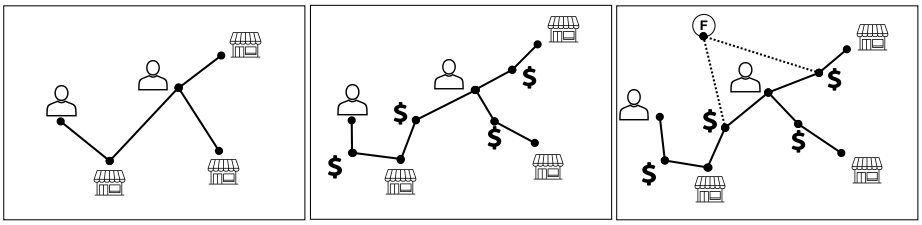

Figure 2: **Left**: bipartite graph with cardholders and merchants connected through edges representing transactions. **Middle**: tripartite graph with transactions transformed into nodes (dollar signs). **Right**: an artificial 'fraud' (F) node is added to the graph and connected with all fraudulent transactions.

create embeddings that can better distinguish fraud from genuine transactions, we thus propose introducing a single artificial 'fraud node' (see Figure 2)[4] to which all fraudulent transactions are connected before RL is applied.

The design of the network structure is important for the downstream classification task. In particular, the artificial node helps to combat the challenge of class imbalance by increasing the connectivity among fraudulent nodes in the network, resulting in a more clustered set of embeddings and subsequently improving the training of the classifier.

3.2. Inductive Pooling Extension

Random walk-based representation learning in general and DeepWalk [43] in particular are transductive algorithms. This implies that obtaining an embedding for nodes not seen during training (i.e., a new transaction) requires complete retraining of the DeepWalk model. This is particularly problematic for DeepWalk, as subsequent runs of the model will not necessarily result in similar embeddings. To avoid such expensive and time-intensive retraining, we devised an inductive extension that combines existing embeddings of nodes from the training data by means of a pooling operator. This extension is detailed as pseudo code in Algorithm 1.

The algorithm starts from a tripartite graph G with a set of vertices V and edges E. This graph contains all information from the training data. The

[4]Please note that only label information on training data can be used for this purpose

algorithm receives a single incoming transaction x, which is part of the test data. In theory, five distinct scenarios can be identified for an incoming transaction. First, both the cardholder and merchant are known and have one or more mutual transaction in the training data; Second, both parties are in the training data, but they do not have any common transactions; Third, only the cardholder is found in the training data; Fourth, only the merchant is found in the training data; Fifth, neither the cardholder nor the merchant of the new transaction are seen during training.

In the first case (Algorithm 1, lines 5-7), both parties are known and have interacted before in the training data. Here, the embedding from the most recent transaction between cardholder and merchant is used as the new embedding.

In the second case, both parties are found in the training data. Consequently, information from both parties could be used. However, pooling embeddings from both cardholder and merchant transactions leads to very poor predictive performance. This is because the combination of embeddings from nodes in distinct regions in a network results in an embedding positioned somewhere in the middle of both regions, thus making no sense from a predictive perspective. As a result, when both parties are in the training data, we rely on the information from the historical transactions from the cardholder only. (Algorithm 1, lines 8-9) [5]

In the third and fourth cases, (Algorithm 1, lines 8-11), either the cardholder or the merchant are in the training data, but not both.

Finally, in the fifth case (Algorithm 1, lines 12-13) there is no useful information in the training network. Therefore, an average of all existing embeddings is calculated and used for the unseen transaction node. [6] Fortunately, this case is uncommon (with 4 days of training data, only 5% of all transactions).

[5] Using transactions in which only the merchant was involved would also make sense, but in our case, using cardholder transactions systematically results in better performance.

[6] This might seem nonsensical at first, but it guarantees that the trained classifier will still work on this incoming transaction not breaking the pipeline.

Algorithm 1: Inductive Pooling Extension for DeepWalk Representation Learning

Data: Tripartite graph $G = (V, E)$, transaction x with cardholder c and merchant m

Result: emb_x for transaction x

$T_m = \{i | i \in V \wedge (i, m) \in E\}$ `// set of all transactions executed by merchant` m

$T_c = \{i | i \in V \wedge (i, c) \in E\}$ `// set of all transactions executed by cardholder` c

$t : V \to \mathbb{R}; i \mapsto t(i)$ `// timestamp function`

$emb : V \to \mathbb{R}^d; i \mapsto emb(i)$ `// embedding lookup function`

if $m \in V \wedge c \in V \wedge (T_c \cap T_m) \neq \emptyset$ **then**

  $i^* = \underset{i \in (T_c \cap T_m)}{\text{argmax}}\ t(i)$ `// most recent mutual transaction`

  $emb(x) \leftarrow emb(i^*)$

else if $c \in V$ **then**

  $emb_x \leftarrow \frac{1}{|T_c|} \sum_{i \in T_c} emb_i$

else if $m \in V$ **then**

  $emb_x \leftarrow \frac{1}{|T_m|} \sum_{i \in T_m} emb_i$

else

  $emb_x \leftarrow \frac{1}{|V|} \sum_{i \in V} emb_i$

3.3. Downstream Classifier Optimization

Representation learning transforms a network or graph into a representation suitable for classification. Concretely, every node is translated into an embedding, positioning the node in a generated embedding space. Note that this representation procedure is unsupervised. Hence, the downstream classifier requires careful hyperparameter tuning to learn from unsupervised embeddings and their associated class labels.

3.3.1. Hyperparameter Tuning

Extreme gradient boosting (XGBoost) [53] is a machine learning algorithm that creates an ensemble of weak learners, typically decision trees. The weak learners are trained sequentially with each successor attempting to correct the errors of its predecessor. XGBoost is widely used by data scientists and has achieved state-of-the-art performance in many applications across various fields

[53].

The choice for XGBoost is well considered. First, fraud detection is a challenging application (see Section 1) requiring a scalable machine learning algorithm with a high capacity. Second, the feature space created by representation learning is complex, resulting in an even stronger requirement for substantial model capacity to successfully train a machine learning model. Third, the node embeddings have no supervised signal encapsulated. Hence, the downstream classifier (i.e., XGBoost) bears even more importance to learn how node embeddings (input features) link to fraud.

Tuning an XGBoost model is crucial and a fortiori in a complex feature space, such as node embeddings. First, the *number of trees* has an important impact on the model capacity. Higher values lead to a more complex model. Hence, this parameter should have a sufficiently high value to learn the complex embedding space. Second, *feature sampling* for training *decision trees* restricts the feature set available to any individual tree in the XGBoost model (*colsample by tree*). In addition, the feature set can be further restricted for each individual decision point in a tree (*colsample by level*). This forces the XGBoost algorithm to consider different feature subsets in each step. It helps to exploit all relevant dimensions of the feature space and helps to increase the robustness of the model. Third, the *learning rate* or *eta* influences the model's learning speed. With a greater *number of trees*, the risk of overfitting increases; hence, it is recommended to lower the *learning rate* to avoid overfitting the training data too quickly.

Our experiments showed that the combination of more trees, feature sampling and slower learning increased performance considerably. Hyperparameters act in concert, which renders it difficult to determine values upfront. As a result, we relied on a grid search to find an optimal combination of hyperparameter values from the parameter grid (see Table 1) in terms of classification performance.

	Values
Number of trees	100, 300
Learning rate	0.1, 0.05, 0.01
Colsample by tree	0.3, 0.6, 0.9
Colsample by level	0.6, 0.9
Subsample	0.5, 0.9

Table 1: Hyperparameter grid for XGBoost

3.3.2. Model Stacking

To demonstrate how *CATCHM* can be integrated with existing FDS, we mimiced a fraud detection model that uses recency, frequency and monetary value features (RFM). These features were inspired by [35] and are closely related to the features currently in use at financial institutions. In *CATCHM*, instead of combining all features (i.e., the RFM features and the embeddings from the representation learner) in one model, two separate models are built. A random forest classifier [54] is trained on the RFM features, and an XGBoost model [53] is trained for the node embeddings.

The predicted class probabilities of both models are combined by means of model stacking. The outputs of multiple machine learning classifiers are aggregated by means of a meta-learner. In this study, we opted for logistic regression as the meta-learner, given that the predictions of only two base learners are combined. The choice for separate models and model stacking was deliberate. The RFM-inspired features are the result of elaborate (manual) feature engineering, often with the help of domain experts. Hence, their individual discriminatory power is relatively high, which combined with a high capacity classifier could lead to overfitting the training data. In contrast, the discriminatory power of unsupervised node embeddings is less obvious and thus requires a downstream classifier with substantial capacity to avoid underfitting.

As a result, *CATCHM* stacks two different models. On the one hand, a bagging classifier for the RFM features, which aids in lowering the variance and avoids overfitting, and, on the other hand, a high capacity boosting classifier for the node embeddings, which helps to reduce bias and avoids underfitting.

4. Benchmarks

To verify the performance of *CATCHM*, we compared it to a number of benchmarks: DeepWalk [43], Node2Vec [44], PageRank [55] and GraphSAGE [40], which are described in more detail below.

4.1. Transductive and Inductive DeepWalk

In Section 3.2, we introduced a tailored pooling extension that enables generating embeddings inductively, without the need for retraining the model. To measure the performance of our pooling extension, it is compared against the original DeepWalk algorithm [43], which learns node embeddings transductively. Transductive network representation learning implies knowledge of the entire transaction network before applying the RL algorithm. There are four important hyperparameters: 1) number of walks per node (10), 2) length of a single walk (80), 3) dimensionality of the node embeddings (128), and 4) window size (5). The choice of hyperparameter values is based on insights from [44, 43]. Given that *CATCHM* includes a network design with an artificial fraud node, we include Inductive DeepWalk (with the inductive pooling extension but without an artificial fraud node) as an alternative benchmark in the evaluation of the experiment.

4.2. Node2Vec + Inductive Pooling

Node2Vec [44] is a transductive algorithm and is almost identical to DeepWalk. The main difference is the random walk procedure; Node2Vec utilizes a biased random walk, which improves its expressive power and influences the node neighborhoods to explore [44]. This is achieved by introducing two additional parameters. Parameter p influences the likelihood of immediately revisiting a node in the walk. Parameter q differentiates between breadth-first ($q > 1$) and depth-first ($q < 1$). Two parameter choices are evaluated in our experiments: $(q = 0.5, p = 1)$ and $(q = 2, p = 1)$.[7]

[7] DeepWalk is a specific case of Node2Vec with $p = 1$ and $q = 1$.

To make Node2Vec inductive, the same inductive pooling extension (see Section 3.2) is applied. Node2Vec generates embeddings for all nodes in the training data, and subsequently, the inductive pooling extension creates embeddings for new incoming transactions.

4.3. PageRank Inductive

The personalized PageRank algorithm has already been used successfully for fraud detection [35, 36, 37, 38]. PageRank [55] iteratively calculates a suspicion score indicating the likelihood of fraudulent behavior; see Eq. 1.

$$\overrightarrow{\varepsilon_k} = \alpha \cdot Q_{norm} \cdot \overrightarrow{\varepsilon}_{k-1} + (1-\alpha) \cdot \overrightarrow{z}_{norm} \tag{1}$$

Where $\overrightarrow{\varepsilon_k}$ is the vector containing the suspicion scores for each node after k iterations, $\overrightarrow{\varepsilon_0}$ is a random start vector with values between $[0, 1]$, $(1-\alpha)$ is the restart probability ($\alpha = 0.85$), Q_{norm}, is the column-normalized weight matrix and $\overrightarrow{z}_{norm}$ is the normalized starting vector, containing nonzero weights for confirmed fraud cases.

In this work, the approach introduced in [35] is applied for the experiments involving PageRank. In particular, the PageRank algorithm is applied three times with different recency-based edge weights in the transaction network.

4.4. GraphSAGE

GraphSAGE [40] is an NRL technique that differs from DeepWalk and Node2Vec in that it learns an embedding generating function based on sampled and aggregated node features from a node's local neighborhood rather than training individual embeddings. This enables the algorithm to leverage node features to generate embeddings for unseen nodes, which makes it inherently inductive without modifications. The authors of [40] introduced several aggregator functions. In this study, we chose the mean pooling aggregator.

The GraphSAGE algorithm allows for supervised and unsupervised training, depending on the loss function [40]. Unsupervised training relies on a graph-based loss function, which encourages neighboring nodes to have similar

embeddings. In contrast, supervised training uses a cross entropy loss emphasizing the prediction of class labels. Given the important class imbalance in our dataset, the supervised variant is more appropriate. In fact, it resembles the scenario where an artificial node is injected into the network (see Section 3.1.2). However, instead of using the predictions of a supervised GraphSAGE model directly, we chose to extract the embeddings and use them to train a downstream classifier, as this results in a better classification performance.

To accommodate the tripartite network structure, we used the HinSAGE algorithm (heterogeneous graphSAGE) [56]. GraphSAGE requires all nodes to have the same set of features, while HinSAGE handles different node types with different sets of features. The cardholder and merchant node types have no features; hence, a single dummy feature was added to both nodes. Important hyperparameter values were depth (2), neighborhood sample size (2,32), and dimension of the embeddings (128). Apart from the modifications introduced in HinSAGE, the implementation of the GraphSAGE algorithm is plain vanilla. We are aware that further engineering of the GraphSAGE model for fraud detection could improve its performance considerably. However, this is considered out of the scope for this paper.

5. Experimental Design

5.1. Data & Benchmarks

The dataset contained 3,240,339 credit card transactions, with a fraud rate of 0.32%. The data labels were assigned by domain experts after suspicious transactions were investigated and confirmed. A rolling window approach was used to create consecutive splits of training, validation and test sets, which allowed us to repeat every experiment 10 times (see Figure 3). Three different settings for the training set size were evaluated: 1, 2 or 4 days of training data. Each day, approximately 100,000 transactions were processed. The size of the test set was always a single day, and the validation set was 20% of the training data. See Table 2 for a small excerpt of data. Fraud detection

systems could be limited in the amount and type of data they can utilize because of data protection laws (e.g., GDPR). However, these only marginally impact CATCHM, as it relies on anonymized IDs of the transacting parties. The RFM features (see Section 3.3.2) are built on transaction details (see Table 2) and are considered nonintrusive for data privacy considerations. For a detailed overview of data privacy in fraud detection, see [57]

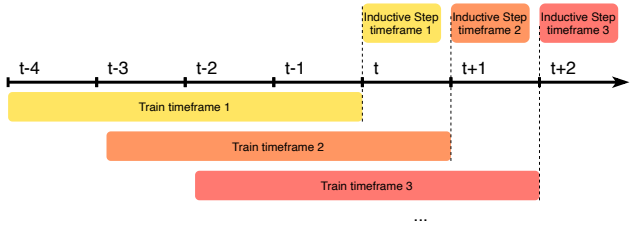

Figure 3: Rolling window of train-test split, with the test set used for the inductive step. For each replication, either 1,2 or 4 days of transaction data were used for training and 1 day for testing.

TX	Cardholder	Merchant	Cat.	Country	Amount	Timestamp	Fraud
t0	AC83FD	m000174	4816	USA	7.37	2013-10-01 01:00:06	False
t1	1CD10E	m207001	5735	LUX	6.25	2013-10-01 01:00:08	False
t2	4ECA55	m003020	7523	CAN	7.18	2013-10-01 01:00:08	False
t3	74186F	m800002	4812	USA	154.93	2013-10-01 01:00:09	True
t4	8777F3	m000102	7399	BEL	15.00	2013-10-01 01:00:10	False

Table 2: Excerpt from the credit card fraud dataset used in this paper. Each line represents one transaction and contains identifiers of parties involved (Cardholder, Merchant) along with timestamp information (Timestamp) and monetary amount of the transaction (Amount). In addition, the country of the Merchant (Country) and Merchant category are reported (Cat.).

In addition to the benchmark techniques described in Section 3, and the novel techniques presented in this work, we also included a baseline model exploiting the original features from the dataset. A second collection of experiments combined the aforementioned techniques with *RFM* features (recency, frequency and monetary value). These features mimic the traditional features found in fraud detection [35]. The aim was to assess the complementarity of network features with traditional RFM features.

5.2. Evaluation Metrics

It is crucial for machine learning models to be evaluated properly. Especially for fraud detection, the extreme class imbalance poses a challenge for traditional metrics (e.g., accuracy).

5.2.1. Classification Performance

The precision-recall curve shows the trade-off between precision (y-axis) and recall (x-axis) for different values of the classification threshold. The Area Under the Precision-Recall Curve ($AUCPR$) offers a summary statistic corresponding to this curve (see Equation 2). For fraud detection, it provides an ideal metric because 1) it is threshold independent, 2) it illustrates the trade-off between catching more criminals and yielding more false alarms, and 3) it takes precision into account (which the classical ROC curve does not).

$$AUCPR = \int_0^1 precision(recall)\, d(recall) \tag{2}$$

Research in machine learning often settles for the receiver operating curve (ROC) or area under the ROC curve. Despite its usefulness, it is a mediocre metric, particularly for fraud detection and, in general, for all applications with extreme class imbalance [58].

A threshold-dependent metric suitable for fraud detection is the *F1* score, which is the harmonic mean of precision and recall (see Equation 3). Given the influence of the threshold, it is optimized on the validation set to maximize the *F1* score.

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

5.2.2. Operational Efficiency

In practice, financial institutions are confronted with a variety of operational constraints. Two of them are crucial for ML-based fraud detection models: (1) fraud detection resources are limited, and (2) real-time fraud detection should be time efficient. For the first constraint, the number of fraud cases that can be

dealt with on a daily basis is limited. Hence, from a practitioner's perspective, it is relevant to know how many fraud cases (true positives) can be discovered among the number of cases that can be investigated ($TP@k$). Let t^* be the classification threshold for which the total number of positive cases is less than or equal to k: $t^* = min\{t|\, TP(t) + FP(t) \leq k \wedge t \in [0,1]\}$; then, $TP@k$ can be defined as:

$$TP@k = TP(t^*) \tag{4}$$

As for the second constraint, the computational burden of the FDS should correspond to the stringent time constraints for transaction authorization. Although little is known about the actual constraints applied in industry, we do know that the maximum authorization time interval has been reduced considerably in recent years. In 2016, the authors of [35] reported six seconds per transaction; currently, a credit card transaction can be processed and authorized within less than one hundred milliseconds [12]. As such, in addition to $TP@k$, we also report on the average prediction processing time (PPT_A). Let n be the number of samples in the test set; then, PPT_A can be defined as:

$$PPT_A = \frac{1}{n}\sum_{i=1}^{n} PPT_i, \tag{5}$$

Given current practices, we expect that the processing time of a single transaction in the test phase should not exceed one hundred milliseconds. Moreover, throughout the experiments, we also applied a one-hour time limit on the total training time, which suffices for daily retraining of the model. The frequent retraining of the model helps to manage the constant change in modi operandi.

5.2.3. Bayesian Model Comparison

While the aforementioned metrics are useful to gauge model performance, we computed averages over ten independent replications. Averages can hide important differences between replications. To compare all approaches in a statistically sound manner, we utilized the Bayesian model comparison presented

in [59]. Specifically, a Bayesian version of signed-ranks [59] was applied to compare our approach to all benchmarks (see Section 5.1). The Bayesian test has a *region of practical equivalence (ROPE)*; this region contains differences in performance that are considered too small to be practically relevant. In this study, the ROPE was 5%, which implies that we only consider method A better than method B when the performance metric is at least 5 percentage points better. [8]

6. Results

This section presents the results from the experiments in which *CATCHM* is compared against relevant benchmarks (see Section 5.1). First, general classification performance is presented. Second, significance tests are executed through Bayesian model comparison, and finally, the operational efficiency of *CATCHM* within capacity and time constraints is discussed.

6.1. Classification Performance

Transductive vs. Inductive. Table 3 presents the results in terms of *AUCPR* and *F1* scores. First, the beneficial impact of more training data on the classification results was confirmed. Both the *AUCPR* and *F1* scores improved considerably with an increased number of training days, for the majority of experiments. Nonetheless, one must keep in mind the trade-off between classification performance and the computational burden associated with larger networks.

Table 3 also reveals that the performance of the baseline was very poor, which highlights the importance of feature engineering in the case of fraud detection. Merely applying a powerful (XGBoost) classifier does not yield a capable fraud detection model.

DeepWalk Transductive outperformed the baseline by a considerable margin. However, this transductive technique was unable to generalize to incoming

[8] This ROPE is considerably larger than the ROPE suggested in [59], leading to a stricter evaluation.

transactions and hence cannot work in an online FDS. Transductive DeepWalk can be applied post hoc to search for fraud cases after the transactions are executed.

For this reason, the results of inductive models are presented separately. These models can be employed in an online FDS. In other words, feature engineering generalizes to new transactions, and hence, features for incoming transactions can be generated without noticeable delay. In general, the performance of the inductive techniques was still considerably better than that of the baseline but worse than that of the transductive variant. This is to be expected, as the transductive models have access to all transactions during training, in contrast to the inductive methods.

CATCHM was the best-performing inductive method. To recognize the benefit of the artificial node, we compared it with the DeepWalk + Inductive Pooling technique. With 4 days of training data, *CATCHM* improved $AUCPR$ by 35% and *F1* by 20%. The artificial class label information injected into the network helped to distinguish embeddings from fraudulent and nonfraudulent transactions.

Three more algorithms were evaluated, namely, PageRank Inductive, Node2Vec and GraphSAGE. The performance of PageRank Inductive was similar to that of DeepWalk + Inductive Pooling. This illustrates how network representation learning is a reasonable alternative for network feature engineering.

For Node2Vec, the results were in line with DeepWalk + Inductive Pooling. Hence, the p and q hyperparameters did not offer much benefit in the transaction network. In theory, Node2Vec has a similar complexity as DeepWalk, in practice, the algorithm does not scale well and requires substantially more computing power than DeepWalk. Therefore, no results were available for '4 days' of training data, as no model could be trained within the predetermined maximum training time of one hour. Given their similar performance, it is highly recommended to use DeepWalk rather than Node2Vec for this application.

There was insufficient data available for GraphSAGE to train properly with only 1 or 2 days of training data. Performance in terms of both $AUCPR$ and

F1 for 4 days of training data was low compared with the other inductive models. Note, however, that we used a plain vanilla GraphSAGE model with no modifications.

	Classification Score	*AUCPR*			*F1*		
		1 day	2 days	4 days	1 day	2 days	4 days
	Baseline	0.06 (0.04)	0.10 (0.04)	0.15 (0.07)	0.14 (0.06)	0.19 (0.06)	0.23 (0.07)
	DeepWalk Transductive	0.34 (0.12)	0.40 (0.10)	0.46 (0.07)	0.31 (0.07)	0.36 (0.08)	0.39 (0.11)
Inductive	PageRank Inductive	0.25 (0.08)	0.30 (0.07)	0.31 (0.10)	0.43 (0.10)	0.47 (0.08)	0.53 (0.05)
Inductive	DeepWalk + Inductive Pooling	0.24 (0.10)	0.31 (0.10)	0.31 (0.11)	0.39 (0.10)	0.46 (0.08)	0.46 (0.10)
Inductive	Node2Vec (0.5) + Inductive Pooling	0.24 (0.10)	0.30 (0.09)	-	0.39 (0.09)	0.47 (0.09)	-
Inductive	Node2Vec (2) + Inductive Pooling	0.23 (0.09)	0.29 (0.09)	-	0.38 (0.09)	0.45 (0.09)	-
Inductive	GraphSAGE	-	-	0.22 (0.08)	-	-	0.29 (0.08)
Inductive	*CATCHM*	**0.28** (0.08)	**0.37** (0.07)	**0.42** (0.09)	**0.43** (0.08)	**0.51** (0.08)	**0.55** (0.09)
RFM	RFM Baseline	0.29 (0.13)	0.32 (0.16)	0.35 (0.12)	0.36 (0.11)	0.38 (0.13)	0.40 (0.09)
RFM	PageRank Inductive	0.34 (0.07)	0.43 (0.08)	0.42 (0.11)	0.40 (0.07)	0.46 (0.15)	0.49 (0.09)
RFM	DeepWalk + Inductive Pooling	0.41 (0.11)	0.49 (0.10)	0.50 (0.10)	0.48 0.08)	0.55 (0.08)	0.56 (0.08)
RFM	Node2Vec (0.5) + Inductive Pooling	0.43 (0.11)	0.48 (0.11)	-	0.49 (0.09)	0.54 (0.08)	-
RFM	Node2Vec (2) + Inductive Pooling	0.41 (0.11)	0.47 (0.11)	-	0.47 (0.09)	0.54 (0.08)	-
RFM	GraphSAGE	-	-	0.40 (0.13)	-	-	0.44 (0.11)
RFM	*CATCHM*	**0.45** (0.10)	**0.53** (0.08)	**0.57** (0.09)	**0.52** (0.07)	**0.59** (0.06)	**0.63** (0.06)

Table 3: *AUCPR* and *F1* results with each column representing a different number of training days. Cells contain the 10-fold average with standard deviation between parentheses. The best performing technique per group is denoted in bold.

RFM. The results from experiments with additional RFM features are presented in the lower half of Table 3. To evaluate the improvement in classification performance, an RFM baseline containing only recency, frequency and monetary features is introduced. As conventional FDS often rely on RFM features, these experiments investigate to what extent RFM features and network features are complementary.

As seen in Table 3, the RFM + DeepWalk + Inductive Pooling experiment, which combines RFM features with network features from DeepWalk, showed an important improvement in *AUCPR* and *F1* compared to the RFM baseline.

What stands out is the combination of RFM features with *CATCHM* yielding the best results overall. These results illustrate how network features carry information not present in the conventional RFM features.

In contrast with the important PageRank Inductive benchmark, which has been used frequently in recent fraud research (see Section 2), *CATCHM* is substantially more complementary to the RFM features, outperforming the PageRank benchmark by a considerable margin both in terms of *AUCPR* (+62%) and *F1* (+58%).

Together, these experiments show that *CATCHM* was successful for credit card fraud detection, achieving excellent classification performance and outperforming all relevant benchmarks.

Bayesian Model Selection. Table 4 contains the results from the pairwise Bayesian signed-rank tests. Each cell contains a probability $P_{i,j}$ measuring the likelihood that method i outperforms method j by more than 5 percentage points on *AUCPR* performance (see Section 5.2). As seen on the left side of Table 4, all methods except GraphSAGE outperformed the baseline with a probability higher than or equal to 95%. *CATCHM* outperformed all other techniques almost with certainty. The right-side of Table 4 provides the probabilities for experiments with RFM features. Again, all experiments outperformed the baseline, except GraphSAGE, with a probability of 95% or more. *CATCHM* combined with RFM features performed significantly better than PageRank Inductive and GraphSAGE with RFM features.

6.2. Operational Efficiency

Investigation Capacity. In reality, the total number of flagged cases per day is limited due to constraints in time, budget and staff. As a result, the main objective of a fraud detection system is to have a high percentage of actual fraud cases in the total number of transactions flagged by the FDS.

To this end, we introduced $TP@k$ in Section 5.2. Table 5 shows the number of fraudulent transactions discovered among 300 flagged transactions. The

	Baseline	PageRank Ind.	GraphSAGE	*CATCHM*	RFM Baseline	RFM + PageRank Ind.	RFM + GraphSAGE	RFM + *CATCHM*
Baseline	-							
PageRank Inductive	**1**	-	0.91					
GraphSAGE	0.8	0.01	-					
CATCHM	**1**	**0.99**	**1**	-				
RFM Baseline					-	0.01		
RFM + PageRank Inductive					**0.96**	-	0.46	
RFM + GraphSAGE					0.61	0.1	-	
RFM + *CATCHM*					**1**	**1**	**1**	-

Table 4: Probabilities $P_{i,j}$ from pairwise Bayesian signed rank tests [59] comparing *AUCPR* results with 4 days of training data. $P_{i,j}$ indicates the likelihood of method i outperforming method j with at least five percentage points difference in *AUCPR* (ROPE = 5%). Only non-zero values are reported.

higher the number of actual fraud cases, the better. All techniques outperform the baseline considerably, with at least twice the number of fraudulent transactions discovered. *CATCHM* returns the highest number of caught fraudsters: with 4 days of training data, almost 200 criminals are caught. The combination with RFM features leads to even better results (see Table 5. *CATCHM* yields on average 213 true positives, which is a 40% increase compared to the *baseline* and almost 20% better than PageRank Inductive.

Revenue. When criminals successfully execute a fraudulent transaction, the loss is either incurred by the cardholder or the financial institution handling the payment. Hence, the performance of the FDS can be measured in terms of avoided losses. In the 10-fold test data, a total of €720K of funds was lost over a period of ten days. Figure 4 shows the revenues (300 flagged transactions/day) for each technique *without* (w/o) and *with* (w/) RFM features. All techniques outperformed the baseline both with and without RFM features. Not only did *CATCHM* detect the highest number of fraud cases, *CATCHM* also yielded the highest revenue overall (42.9% of €720K). Note that the revenue of *CATCHM* was similar to the stacked version with RFM features, which was due to the fact

	True Positives @ 300	1 day avg (std)	2 days avg (std)	4 days avg (std)
	Baseline	39.90 (24.98)	66.40 (36.12)	86.40 (41.27)
	DeepWalk Transductive	131.40 (58.32)	164.60 (58.72)	167.50 (56.46)
Inductive	PageRank Inductive	156.50 (49.41)	167.00 (42.54)	189.60 (39.55)
Inductive	DeepWalk + Inductive Pooling	146.30 (56.59)	173.60 (50.96)	168.60 (46.50)
Inductive	Node2Vec (0.5) + Inductive Pooling	148.80 (57.92)	172.40 (51.03)	-
Inductive	Node2Vec (2) + Inductive Pooling	147.00 (57.43)	173.50 (56.19)	-
Inductive	GraphSAGE	-	-	108.50 (41.27)
Inductive	**CATCHM**	**157.10** (44.57)	**197.90** (61.53)	**198.80** (42.55)
RFM	RFM Baseline	130.00 (54.23)	140.50 (69.71)	151.20 (61.62)
RFM	PageRank Inductive	152.80 (54.06)	186.10 (47.02)	180.40 (61.30)
RFM	DeepWalk + Inductive Pooling	174.00 (49.19)	196.20 (49.08)	198.50 (44.70)
RFM	Node2Vec (0.5) + Inductive Pooling	172.80 (49.06)	194.70 (53.80)	-
RFM	Node2Vec (2) + Inductive Pooling	170.80 (53.80)	191.90 (48.47)	-
RFM	GraphSAGE	-	-	164.10 (62.68)
RFM	**CATCHM**	**182.30** (44.97)	**204.70** (41.34)	**213.00** (38.90)

Table 5: True Positives @ 300 for transductive and inductive experiments. Each column shows the results for a different number of training days. Cells contain the 10-fold average with standard deviation between parentheses.

that the additional fraud cases had low transaction amounts.

Processing Time. Finally, Figure 5 shows the processing time for incoming, unseen transactions. All techniques remained well below 100 milliseconds, which is acceptable for use in production. What stands out in the figure is the extremely low processing time for GraphSAGE. The *CATCHM* technique took an average of 10 milliseconds to process a transaction (with 4 days of training data).

In conclusion, combining the operational efficiency of *CATCHM* with the results in Table 3, we can confidently state that our algorithm overall outperformed the benchmarked techniques.

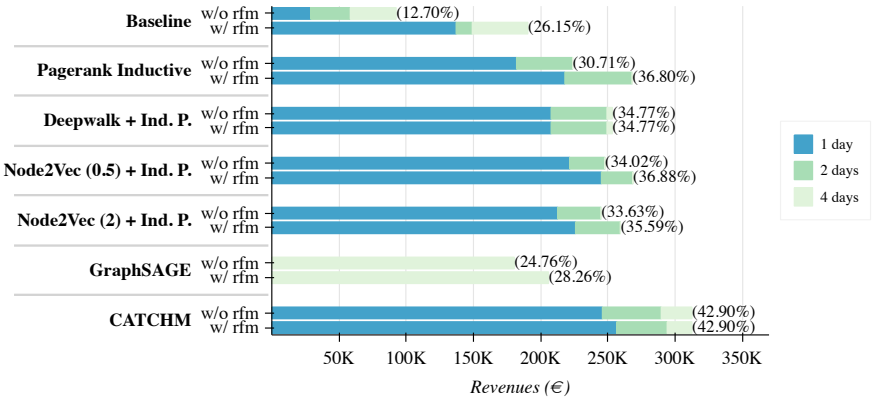

Figure 4: Total revenue considering 300 flagged transactions per day, *with* (w/) and *without* (w/o) RFM features. The different colors represent a different number of training days. A fraudulent transaction blocked by a detection model (TP) prevents the funds from being stolen. These funds are considered 'revenues' of the model. The results are summed over 10-fold. Between parentheses, the recovered funds are expressed as a percentage of the total value of all fraudulent transactions (€720K)

7. Conclusion

In this work, we proposed *CATCHM*, a novel network-based credit card fraud detection approach using node representation learning. *CATCHM* was designed to tackle the challenges of fraud detection and overcome the limitations of current detection techniques. First, important adaptations were introduced to improve representation learning for fraud detection:

1) Network design: By means of a tripartite network rather than the classical bipartite graph the network is kept sparse, while each transaction obtains an individual embedding. In addition, inspired by [52], an artificial node was injected into the network. The results indicated that the augmented transaction network improved classification performance considerably.

2) Inductive Pooling Extension: An efficient Inductive Pooling extension for the transductive algorithm DeepWalk was introduced. The inductive extension avoids costly retraining and potential rotation of the embedding space, which in turn improves the maintainability of *CATCHM*.

3) Downstream classifier optimization: Node embeddings require careful hyperparameter tuning of the classifier to unlock their full potential. In addition,

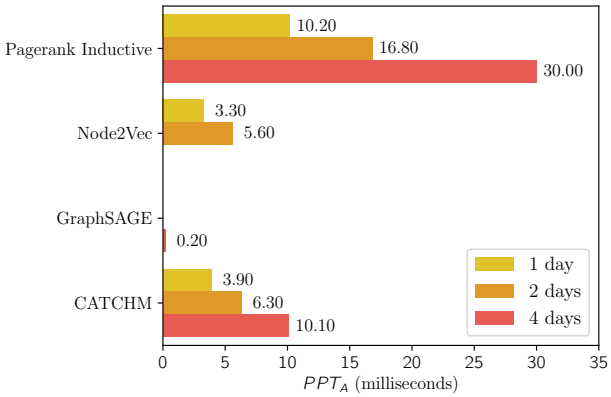

Figure 5: Average prediction processing time (PPT_A) in milliseconds for a single incoming transaction. Processing time was measured at the batch level and divided by the number of transactions in the batch.

model stacking was suggested to combine *CATCHM* with existing FDS.

The aforementioned adaptations tackle the inherent challenges of fraud detection, namely: class imbalance, concept drift and time constraints. Furthermore, combining representation learning with these adaptations overcomes the limitations of classification performance, maintainability and costs experienced by contemporary fraud detection systems.

Second, the classification performance was assessed by means of an extensive empirical evaluation. *CATCHM* was shown to outperform state-of-the-art methods from both the fraud detection and network representation learning literature.

Third, we expanded upon the practical relevance of our research. Both time and capacity constraints were explicitly considered to evaluate the operational efficiency of *CATCHM*. In addition to conventional performance measures, the $TP@k$ metric was put forward to demonstrate the performance while taking operational constraints into account, including a complementary analysis in terms of realized revenue. Finally, given that scalability is crucial in industry, the inductive extension was designed for parallel execution.

The value of artificial nodes for representation learning is intriguing and its

usefulness could be further explored in future research. In this work, only class label information was carried by an artificial node, but other features could also be represented as an artificial node in the network, expanding the information available to the RL algorithm. A future study could also investigate different choices for the inductive extension. We relied exclusively on a mean pooling operator, although a wide range of alternatives is feasible. We believe optimizing the aggregator could further improve the final predictive performance.

Acknowledgments

This research was supported by HIDDEN FOR DOUBLE BLIND and was financed in part by the HIDDEN FOR DOUBLE BLIND.

References

[1] European Central Bank, Seventh report on card fraud (2021) (Oct. 2021).

[2] The Federal Reserve, The 2019 payments study (Jan. 2019).

[3] R. J. Bolton, D. J. Hand, Statistical fraud detection: A review, Statistical science 17 (3) (2002) 235–255.

[4] K. J. Leonard, The development of a rule based expert system model for fraud alert in consumer credit, Eur. J. Oper. Res. 80 (2) (1995) 350–356.

[5] S. Stolfo, D. W. Fan, W. Lee, A. Prodromidis, P. Chan, Credit card fraud detection using meta-learning: Issues and initial results, in: AAAI-97 Workshop on Fraud Detection and Risk Management, 1997, pp. 83–90.

[6] J. R. Dorronsoro, F. Ginel, C. Sgnchez, C. S. Cruz, Neural fraud detection in credit card operations, IEEE Trans. Neural Netw. 8 (4) (1997) 827–834.

[7] S. Maes, K. Tuyls, B. Vanschoenwinkel, B. Manderick, Credit card fraud detection using bayesian and neural networks, in: Proceedings of the 1st international naiso congress on neuro fuzzy technologies, Vol. 7, 2002.

[8] N. F. Ryman-Tubb, P. Krause, W. Garn, How artificial intelligence and machine learning research impacts payment card fraud detection: A survey and industry benchmark, Eng. Appl. Artif. Intell. 76 (2018) 130–157.

[9] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, G. Bontempi, Credit card fraud detection: A realistic modeling and a novel learning strategy, IEEE Trans Neural Netw Learn Syst 29 (8) (2018) 3784–3797.

[10] P. de Spain, G. Evers, J. Sherwin Smith, C. Ramsey, D. Wallace, K. Brown, Keeping pace in a global economy: Helping financial institutions get up to speed with real-time, Tech. rep., Mastercard (2021).

[11] SEPA instant credit transfer, https://www.europeanpaymentscouncil.eu/what-we-do/sepa-instant-credit-transfer, accessed: 2022-5-1.

[12] B. Stackpole, How big firms leverage artificial intelligence for competitive advantage, https://mitsloan.mit.edu/ideas-made-to-matter/how-big-firms-leverage-artificial-intelligence-competitive-advantage, accessed: 2021-9-27 (May 2021).

[13] Mastercard, Annual report, Tech. rep. (2012).

[14] C. Tsung-Nan, A novel prediction model for credit card risk management, Innovative Computing, Information and Control (2007).

[15] T. Guo, G.-Y. Li, Neural data mining for credit card fraud detection, in: 2008 International Conference on Machine Learning and Cybernetics, Vol. 7, ieeexplore.ieee.org, 2008, pp. 3630–3634.

[16] C. C. Lee, J. W. Yoon, A data mining approach using transaction patterns for card fraud detection, arXiv preprint arXiv:1306.5547 (Jun. 2013). arXiv:1306.5547.

[17] M. K. Mishra, R. Dash, A comparative study of chebyshev functional link artificial neural network, multi-layer perceptron and decision tree for credit card fraud detection, in: 2014 International Conference on Information Technology, ieeexplore.ieee.org, 2014, pp. 228–233.

[18] N. Mahmoudi, E. Duman, Detecting credit card fraud by modified fisher discriminant analysis, Expert Syst. Appl. 42 (5) (2015) 2510–2516.

[19] A. Zakaryazad, E. Duman, A profit-driven artificial neural network (ANN) with applications to fraud detection and direct marketing, Neurocomputing 175 (2016) 121–131.

[20] R. Brause, T. Langsdorf, M. Hepp, Neural data mining for credit card

fraud detection, in: Proceedings 11th International Conference on Tools with Artificial Intelligence, ieeexplore.ieee.org, 1999, pp. 103–106.

[21] T. Minegishi, A. Niimi, Proposal of credit card fraudulent use detection by online-type decision tree construction and verification of generality, International Journal for Information Security Research (IJISR) 1 (4) (2011) 229–235.

[22] M. F. A. Gadi, X. Wang, A. P. d. Lago, Credit card fraud detection with artificial immune system, in: International conference on artificial immune systems, Springer, 2008, pp. 119–131.

[23] Bhusari, S. Patil, Application of hidden markov model in credit card fraud detection, Int. J. Parallel Emergent Distrib. Syst. 2 (6) (2011) 203–211.

[24] A. C. Bahnsen, A. Stojanovic, D. Aouada, B. Ottersten, Cost sensitive credit card fraud detection using bayes minimum risk, in: 2013 12th International Conference on Machine Learning and Applications, Vol. 1, ieeexplore.ieee.org, 2013, pp. 333–338.

[25] V. Dheepa, R. Dhanapal, Behavior based credit card fraud detection using support vector machines, ICTACT Journal on Soft computing (2012).

[26] A. J. Reiss, Co-Offending and criminal careers, Crime and Justice 10 (1988) 117–170.

[27] C. Chen, C. Liang, J. Lin, L. Wang, Z. Liu, X. Yang, J. Zhou, Y. Shuang, Y. Qi, InfDetect: A large scale graph-based fraud detection system for e-commerce insurance, in: 2019 IEEE International Conference on Big Data (Big Data), IEEE, 2019, pp. 1765–1773.

[28] T. Pourhabibi, K.-L. Ong, B. H. Kam, Y. L. Boo, Fraud detection: A systematic literature review of graph-based anomaly detection approaches, Decis. Support Syst. 133 (2020) 113303.

[29] M. Newman, Networks, Oxford University Press, 2018.

[30] I. Molloy, S. Chari, U. Finkler, M. Wiggerman, C. Jonker, T. Habeck, Y. Park, F. Jordens, R. van Schaik, Graph analytics for real-time scoring of cross-channel transactional fraud, in: 20th International Conference on Financial Cryptography and Data Security, FC 2016, Vol. 9603 LNCS,

Springer Verlag, 2017, pp. 22–40.

[31] S. Shehnepoor, M. Salehi, R. Farahbakhsh, N. Crespi, NetSpam: A Network-Based spam detection framework for reviews in online social media, IEEE Trans. Inf. Forensics Secur. 12 (7) (2017) 1585–1595.

[32] P. Bangcharoensap, H. Kobayashi, N. Shimizu, S. Yamauchi, T. Murata, Two step graph-based semi-supervised learning for online auction fraud detection, in: Joint European conference on machine learning and knowledge discovery in databases, Springer, 2015, pp. 165–179.

[33] Y. Li, Y. Sun, N. Contractor, Graph mining assisted semi-supervised learning for fraudulent cash-out detection, in: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, ASONAM '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 546–553.

[34] M. Óskarsdóttir, W. Ahmed, K. Antonio, B. Baesens, R. Dendievel, T. Donas, T. Reynkens, Social network analytics for supervised fraud detection in insurance, Risk Anal. (Feb. 2021).

[35] V. Van Vlasselaer, C. Bravo, O. Caelen, T. Eliassi-Rad, L. Akoglu, M. Snoeck, B. Baesens, APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions, Decis. Support Syst. 75 (2015) 38–48.

[36] V. Van Vlasselaer, T. Eliassi-Rad, L. Akoglu, M. Snoeck, B. Baesens, GOTCHA! Network-Based fraud detection for social security fraud, Manage. Sci. 63 (9) (2017) 3090–3110.

[37] B. Lebichot, F. Braun, O. Caelen, M. Saerens, A graph-based, semi-supervised, credit card fraud detection system, in: International Workshop on Complex Networks and their Applications, Springer, 2016, pp. 721–733.

[38] S. Courtain, B. Lebichot, I. Kivimäki, M. Saerens, Graph-based fraud detection with the free energy distance, in: International Conference on Complex Networks and Their Applications, Springer, 2019, pp. 40–52.

[39] S. Cao, W. Lu, Q. Xu, GraRep: Learning graph representations with global structural information, in: Proceedings of the 24th ACM International

on Conference on Information and Knowledge Management, CIKM '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 891–900.

[40] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 1025–1035.

[41] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th international conference on world wide web, 2015, pp. 1067–1077.

[42] D. Zhang, J. Yin, X. Zhu, C. Zhang, Network representation learning: A survey, IEEE transactions on Big Data 6 (1) (2018) 3–28.

[43] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[44] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.

[45] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems 26, Curran Associates, Inc., 2013, pp. 3111–3119.

[46] J. Hu, T. Li, Y. Zhuang, S. Huang, S. Dong, GFD: A weighted heterogeneous graph embedding based approach for fraud detection in mobile advertising, Secur. Commun. Netw. 2020 (2020) 1–12.

[47] C. Xu, Z. Feng, Y. Chen, M. Wang, T. Wei, Featnet: large-scale fraud device detection by network representation learning with rich features, in: Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security, 2018, pp. 57–63.

[48] A. Khazane, J. Rider, M. Serpe, A. Gogoglou, K. Hines, C. B. Bruss, R. Serpe, DeepTrax: Embedding graphs of financial transactions, in: 2019 18th IEEE International Conference On Machine Learning And Applica-

tions (ICMLA), 2019, pp. 126–133.

[49] R. Van Belle, S. Mitrovic, J. De Weerdt, Graph representation learning for fraud prediction: A nearest neighbour approach, https://grlearning.github.io/papers/ (2019).

[50] R. Van Belle, S. Mitrović, J. De Weerdt, Representation learning in graphs for credit card fraud detection, in: Mining Data for Financial Applications, Springer International Publishing, 2020, pp. 32–46.

[51] R. Van Belle, C. Van Damme, H. Tytgat, J. De Weerdt, Inductive graph representation learning for fraud detection, Expert Systems with Applications (2022) 116463.

[52] S. Mitrović, B. Baesens, W. Lemahieu, J. De Weerdt, tcc2vec: RFM-informed representation learning on call graphs for churn prediction, Inf. Sci. (Feb. 2019).

[53] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[54] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[55] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web., Tech. rep., Stanford InfoLab (1999).

[56] C. Data61, Stellargraph machine learning library, `https://github.com/stellargraph/stellargraph` (2018).

[57] L. Găbudeanu, I. Brici, C. Mare, I. C. Mihai, M. C. Șcheau, Privacy intrusiveness in Financial-Banking fraud detection, Risks 9 (6) (2021) 104.

[58] J. Davis, M. Goadrich, The relationship between Precision-Recall and ROC curves, in: Proceedings of the 23rd international conference on Machine learning, ICML '06, Association for Computing Machinery, New York, NY, USA, 2006, pp. 233–240.

[59] A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, The Journal of Machine Learning Research 18 (1) (2017) 2653–2688.