Contents lists available at ScienceDirect

Journal of King Saud University - Computer and Information Sciences

journal homepage: www.sciencedirect.com

Full length article

Encoder–decoder graph neural network for credit card fraud detection☆

Asma Cherif[a,b,*], Heyfa Ammar[b,c,d], Manal Kalkatawi[a], Suhair Alshehri[a], Abdessamad Imine[e]

[a] *Department of Information Technology, Faculty of Computing & Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia*
[b] *Center of Excellence in Smart Environment Research, King Abdulaziz University, Jeddah, Saudi Arabia*
[c] *Prince Sultan University, Riyadh, Saudi Arabia*
[d] *RISC-ENIT Research Lab, National Engineering School of Tunis, University of Tunis-Almanar, Tunisia*
[e] *Université de Lorraine, CNRS, INRIA, Nancy, France*

ARTICLE INFO

Keywords:
Credit card fraud detection
Graph neural networks
Encoder–decoder
Feature engineering
Deep learning

ABSTRACT

Credit card fraud is a significant problem, with millions of dollars lost each year. Detecting fraudulent transactions is a challenging task due to the large volume of data and the constantly evolving tactics of fraudsters. Likewise any detection problem, it is important to select relevant features that help distinguishing between fraudulent and non-fraudulent transactions. It is also crucial to design a model that effectively captures the relationships between involved entities such as merchants and customers. In this work, a prior stage of feature engineering is carried out. Then, the use of the Graph Neural Networks (GNNs) for credit card fraud detection is explored by leveraging the relationships between customers and merchants. We propose a novel encoder–decoder based GNNs that effectively captures the complex relationships and dependencies within credit card transaction data. The performance of the model is further improved by employing a graph converter for efficient graph data processing and batch normalization for reducing internal covariate shift and accelerating convergence. Our experiments on a large scale dataset delivered promising results that outperform other models in terms of precision, recall, and F1 score.

1. Introduction

Context. In the modern digital economy, credit card payments have become the dominant form of digital commerce, with key players in the industry constantly adapting to meet evolving user preferences through innovative technologies. This includes the adoption of Internet of Things (IoT) devices, in-app payments, and mobile devices, which have disrupted traditional credit card payment systems. For example, companies like Amazon Go are exploring biometrics-based payments, while tokenization and mobile IoT devices like smartwatches facilitate on-demand payments and enable new communication models for transactions. However, the increased use of credit cards has also led to a rise in fraudulent activities. Fraudsters employ sophisticated techniques such as theft, phishing for credit card information, and producing counterfeit cards to mimic legitimate user behavior. As a result, cardholders and banks suffer significant losses from these fraudulent transactions. To mitigate this, it is crucial for payments providers to modernize their legacy technology. This involves upgrading infrastructure and deployments, enhancing the middleware ecosystem, and improving front-end channels and execution systems. By doing so, payments providers can take advantage of new technologies and innovations to enhance security measures and protect users from fraudulent activities.

The adoption of Artificial Intelligence (AI) and Deep Learning (DL) techniques is becoming more important for banking services and card issuers. AI is being used to develop new approaches to improve credit

☆ This project was funded by the Institutional Fund Projects under grant No. (IFPRC-032-612-2020). The authors, therefore, gratefully acknowledge technical and financial support from the Ministry of Education and King Abdulaziz University, Jeddah, Saudi Arabia.

* Corresponding author at: Department of Information Technology, Faculty of Computing & Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia.

E-mail address: acherif@kau.edu.sa (A. Cherif).

Peer review under responsibility of King Saud University.

https://doi.org/10.1016/j.jksuci.2024.102003
Received 2 October 2023; Received in revised form 17 February 2024; Accepted 29 February 2024
Available online 7 March 2024

card fraud detection. These approaches aim to enhance approval rates, minimize declined transactions, and facilitate proactive monitoring of credit limits. AI is a major project that aims to balance the wide availability of banking and e-commerce services with fine-grained credit card fraud detection.

Problem statement. Building a robust Machine Learning (ML) model is a challenging problem when it comes to solving credit card fraud detection. First, fraudsters are constantly evolving their tactics, making it difficult to track their methods. Next, ML models must regularly rely on large, high-quality datasets to avoid biased results. However, credit card fraud is a relatively rare event compared to legitimate transactions. The rarity of this event leads to imbalanced datasets (Cherif et al., 2023), making it difficult for ML models to accurately identify fraudulent transactions. Finally, adversarial attacks in which fraudsters can intentionally manipulate transactions to fool ML models pose another challenge that can undermine the effectiveness of ML-based fraud detection systems.

To address these challenges, advanced ML techniques, such as anomaly detection using deep learning, are increasingly used. Additionally, the use of high-quality labeled datasets, feature engineering, and continuous model monitoring are essential to improve fraud detection accuracy while ensuring data privacy and security.

Although solutions have been proposed to predict credit card fraud, the intricate relationships between customers and merchants have not been thoroughly studied. Indeed, the relationship between the customer and the merchant may reveal behavioral patterns thus allowing the detection of fraudulent transactions. Such relationship involves many features like the transaction information, the customer and merchant locations, etc. For instance, an online store may be fraudulent, and if a previous fraudulent transaction is captured by the model then new transactions towards the merchant should be detected as a fraud. In the same way, if the customer makes several purchases from a specific merchant without signaling them as fraud, this helps enforcing this relationship. In order to model the links between a customer and the merchants, we propose in this work to exploit the capability of the Graph Neural Networks (GNNs) in dealing with data that could be structured as a graph.

Unlike traditional neural networks, GNNs can process complex and irregular data structures, including social networks (Keramatfar et al., 2022), chemical compounds (Keramatfar et al., 2022), and knowledge graphs (Ye et al., 2022). Very limited research investigate GNN to predict fraudulent credit card transactions. This research aims to explore the feasibility of such a model for predicting credit card fraud and compare its performance to other DL solutions, namely autoencoders for anomaly detection.

Contributions. In our research, we aim to explore the potential of GNNs in accurately predicting credit card fraud by examining the graph representation of fraud data. The contributions of this research can be summarized in three key aspects: Firstly, we conduct an in-depth data exploration of our dataset to identify and incorporate new relevant features. Secondly, we construct a graph representation of the dataset, in which customers and merchants are represented as nodes. Lastly, we implement a Graph Neural Network (GNN) and thoroughly evaluate the performance of our model. Our GNN model is based on encoder–decoder architecture thus being able to capture both the local and global information present in the graph structure. Our extensive experiments on a large-scale dataset yielded good results, demonstrating a significant advancement in performance metrics. Notably, our model achieved a precision of 0.82, a recall of 0.92, and an F1-score of 0.86. Furthermore, the area under the ROC curve (AUC-ROC) reached a notable value of 0.92. These results demonstrate the high performance of our model, outperforming established benchmarks and underlining its effectiveness in real-world applications.

Outline. The paper is structured as follows: Section 2 provides an overview of recent models for detecting credit card fraud. Section 3 presents our methodology. Section 4 showcases the data exploration and preprocessing procedures. Section 5 delves into the implementation details and testing results. Lastly, Section 6 concludes the paper.

2. Related work

Numerous studies have focused on the issue of detecting credit card fraud due to its significant financial repercussions. Besides, the rise of advanced technologies and communication methods has led to a significant increase in credit card fraud (Cherif et al., 2023). Therefore, numerous studies are currently underway to explore more advanced solutions in this area. Cherif et al. (2023) presents a comprehensive analysis of the latest research conducted from 2015 to 2021, focusing on the detection and prediction of fraudulent credit card transactions. This study revealed a lack of extensive research on deep learning in this field, indicating the need for further investigation to tackle the challenges associated with detecting credit card fraud.

Machine learning has been widely used to automate the detection of credit card fraud. Recently, many attempts to use Neural Networks have been proposed. For instance, in Rb and Kr (2021) proposed an Artificial Neural Network (ANN) model that was compared to Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) and achieved the best results in terms of accuracy (0.9992), precision (0.8115), and recall (0.7619). To deal with the class imbalance problem, undersampling was used which reduces the dataset size. Moreover, the authors did not discuss the details about the dataset used or the final size after undersampling. Additionally, the recall rate is relatively low. Singh et al. (2023) proposed a system that combines swarm intelligence and a neural network architecture to detect fraudulent activities. The proposed approach, called FFFW-RBN, utilizes a hybrid swarm intelligence model that incorporates the Fruitfly Optimization Algorithm and Fireworks Algorithm for feature selection. Additionally, it employs the Neural Embeddings Generator and a Shallow Radial Basis Network for localized pattern classification. Through extensive testing, the system achieves an impressive 0.9997 accuracy, 0.91 recall, and 0.94 precision. However, the dataset used (ULB with 284,807 transactions) is old and small and it is not guaranteed that the traditional NN will be efficient with large and complex data. In order to improve the detection performances in terms of recall while maintaining high accuracy, Chung and Lee (2023) proposed to combine three machine learning models, namely the K-nearest neighbors (KNN), the linear discriminant analysis (LDA), and linear regression (LR). The combination of these models aims to better distinguish between both classes "Fraud" and "non-Fraud". The system achieved an average recall of 0.976 and an average accuracy of 0.987 when evaluated on four public datasets. However, this approach compromised the precision metric whose average was 0.104 on the four datasets.

Although detecting fraud using AI has been widely studied throughout the years, it has been demonstrated that more deep learning based models need to be investigated (Cherif et al., 2023) to handle the huge amount of transaction data more efficiently. Besides deep learning models, researchers are still investigating traditional techniques. For example, in a recent study (Jose et al., 2023), authors focused on credit card fraud detection using a synthetic dataset generated by Yeo (2022). Several machine learning algorithms, including Decision Trees (DT), Random Forests (RF), Extra Trees (ET), and Gradient Boosting (GB), were employed. Each algorithm was combined with the AdaBoost method to enhance the classification accuracy. Oversampling techniques were used to solve the class imbalance issue in the dataset. The models with AdaBoost achieved high accuracy rates, with RF at 0.99995, DT at 0.9999, ET at 0.99975, and GB at 0.99995. The study also compared these models with existing frameworks, showing promising results. However, one of the main disadvantage of SMOTE algorithm is that it may lead to overfitting as it will lead to the duplication of instances, selecting from a limited pool of instances will result in the introduction of duplicate data (Sáez et al., 2016; Zhu

Table 1
Related work summary.

Ref	ML	Algorithm(s)	Domain	Dataset	Strengths	Limitations
Rb and Kr (2021)	Traditional	ANN vs. KNN and SVM	Credit card	Public (Kaggle)		Low recall & missing dataset details before and after preprocessing & the use of undersampling reduces the dataset size and may lead to information loss
Singh et al. (2023)	Traditional	Shallow NN	Credit card fraud	Public (ULB)	Use of Hybrid swarm intelligence and neural embedding generator for feature engineering for feature selection	Small dataset + non deep architecture
Chung and Lee (2023)	Traditional	KNN+LDA+LR	Credit card	Public	Applied on 5 datasets + high recall	Low precision
Jose et al. (2023)	Traditional	DT&RF&ET&GB	Credit card	Public & synthetic (Sparkov)	Large dataset	Use of oversampling that may lead to overfitting
Afriyie et al. (2023)	Traditional	LR&DT&RF	Credit card	Public & synthetic (Sparkov)	Large dataset	Undersampling may reduce the dataset size + RF may be complex to build for large sets
Liu et al. (2018)	Deep	GNN	Online payment	Private & collected from Alipay (1 month)	Robust model for large datasets	No discussion of the class imbalance problem
Liu et al. (2021a)	Deep	GNN	Class imbalance for fraud (e.g, opinion, financial, etc.)	Many public dataset collected online	Model tested for different use cases and datasets	Model performance is better for smaller datasets and relatively low performance for financial fraud

et al., 2017). Another recent study that used the synthetic Sparkov dataset was presented in Afriyie et al. (2023). In this study, authors examined the effectiveness of logistic regression (LR), random forest, and decision trees in identifying and predicting credit card fraud. By comparing these models, they found that random forest achieves the highest accuracy of 0.96 (with an area under the curve of 0.989) in detecting fraudulent transactions. However, the proposed model used undersampling which reduces the data set size and may lead to information loss. In addition to building numerous trees and combining their outputs, random forest requires a significant amount of computational power and resources, which may hinder the real-time aspect of the credit card fraud detection problem.

Though many traditional and deep learning models have been suggested and are still being investigated to solve the credit card fraud detection problem, to our knowledge GNN has not been heavily studied for such a problem though being useful in anomaly detection (Chaudhary et al., 2019; Deng and Hooi, 2021). For instance, the work in Chaudhary et al. (2019) and Deng and Hooi (2021) considered using GNN to detect anomalies by utilizing GNN's ability to model complex relationships that conventional machine learning techniques are unable to comprehend. In Liu et al. (2021a), authors addressed the class imbalance issue in fraud detection using GNN. They proposed a novel approach called Pick and Choose Graph Neural Network (PC-GNN). This method focuses on imbalanced supervised learning on graphs. It starts by constructing sub-graphs for mini-batch training by carefully selecting nodes and edges using a devised label-balanced sampler. They model the problem as a node label prediction and studied relations between customers. Then, for each node in the sub-graph, they choose neighbor candidates using a proposed neighborhood sampler. The results from their experiments on both benchmark and real-world graph-based fraud detection tasks, such as opinion fraud and financial fraud, showed that PC-GNN surpasses the performance of existing state-of-the-art methods by a significant margin. However, the model was more efficient for opinion fraud detection. Indeed, the best performance for F-macro (unweighted mean of the F1-score of each class.), AUC, and Gmean achieved were 0.57, 0.81, 0.66 respectively. They also deduced that the model behaves better with smaller opinion fraud dataset as the suggested method achieves better performance in terms of F1-score for the fraudulent class (0.81 for Amazon dataset vs 0.43 for the YelpChi dataset). Liu et al. (2018) developed a GNN based models for fraud detection. They used a heterogeneous graph to model account to device interaction as for detecting for malicious accounts on Alipay. By learning discriminative embeddings from heterogeneous account-device graphs, they employed an attention mechanism to determine the significance of different node types. Testing showed the model achieved an average F1-score and AUC of 0.79 and 0.92, respectively. However, this solution is only applicable on online financial systems. Moreover, the authors did not discuss the class imbalance, nor did they provide the ratio of fraudulent accounts, citing data sensitivity reasons.

Table 1 summarizes the findings of the related work and highlights the limitations of each approach.

3. Methodology

We model the credit card fraud detection using graph neural networks to capture the relationship between customers and merchants.

Graph Neural Networks (GNNs) are a specific kind of neural network that is specifically engineered to process and analyze data that is structured as a graph. In contrast to traditional neural networks that are used for tabular or sequential data, GNNs can handle complex and irregular data structures such as social networks, chemical compounds, and knowledge graphs. Several research works e.g., Bruna et al. (2014), have revisited the challenge of extending neural networks to handle graphs with arbitrary structures (Kipf and Welling, 2017). GNNs are specifically designed to excel in various tasks, including node classification, link prediction, and graph classification. Thomas et al. (2023). The key idea behind GNNs is to learn a representation of each node in a graph by combining information from its neighbors in the graph. GNNs have shown impressive results in a variety of applications, including recommendation systems (Wu et al., 2022), drug discovery (Li et al., 2021), and social network analysis (Liu et al., 2021b; Keramatfar et al., 2022).

The research methodology employed in this study is depicted in Fig. 1. It involves the integration of customer and merchant datasets using the user identifier. Subsequently, the data is preprocessed to enhance its quality and facilitate feature engineering, including the addition of statistical features. Following the preprocessing stage, a graph representation is constructed based on the processed dataset,

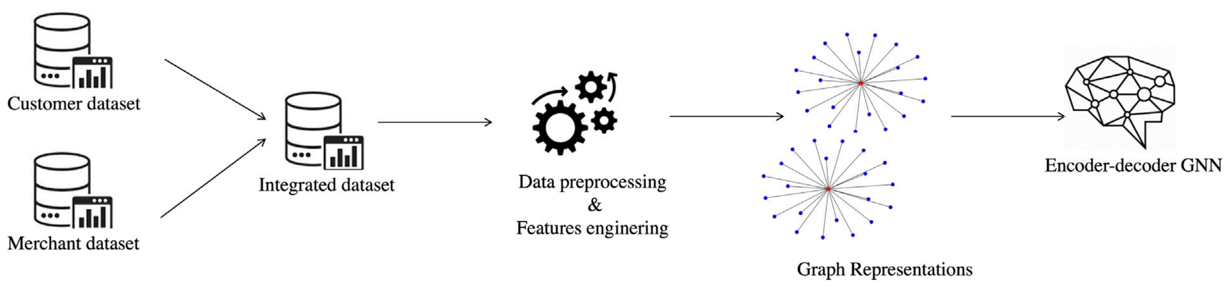


Fig. 1. Research methodology.

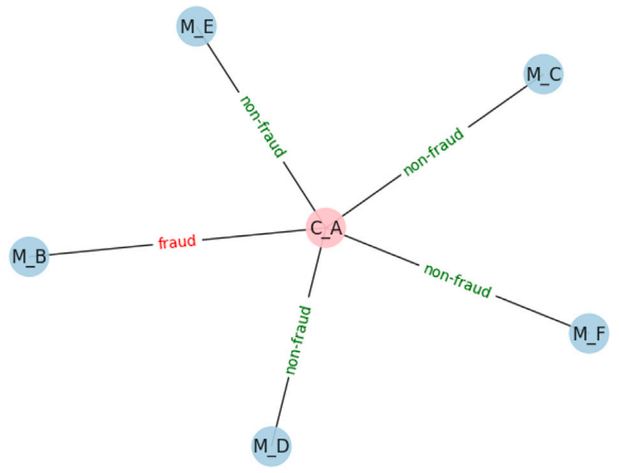


Fig. 2. Dataset modeling as a heterogeneous graph: C_A represents a customer and M_X ($X \in A, B, C, D, E$) represent merchants, while the edge label refers to the transaction type.

which serves as the foundation for training the Graph Neural Network (GNN) model.

Considering a tabular dataset containing customer and merchants information and transactions performed by each customer, we solve the credit card problem using heterogeneous graph representation of our data as illustrated in Fig. 2. In this figure, the central node C_A represents a customer while the other nodes represent merchants.

Moreover, it is crucial to consider both node attributes and link attributes in this task. Link attributes encompass transaction details that are included in the dataset, along with additional engineered features, which will be elaborated on later (see Section 4.2), namely, transaction amount, distance from customer, transaction during night, average transaction amount per day, week, month. Unlike link attributes, which are evident and derived from pre-processed dataset, the information regarding merchants might be limited. As a result, the only viable option to represent node features is by utilizing geographical coordinates. Remarkably, the experimental findings (see Section 5) indicate that incorporating geographical coordinates as node attributes is a viable approach, enabling the reflection of spatial information within the dataset.

Finally, we solve the prediction problem as an edge labeling prediction task where the link label between the customer and each merchant is the prediction task and could be either fraud or non-fraud. We model our data as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of nodes and $\mathcal{E}$ represents the set of edges. Each node $v_i \in \mathcal{V}$ is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^d$, and each edge $(v_i, v_j) \in \mathcal{E}$ is associated with an edge feature vector $\mathbf{e}_{ij} \in \mathbb{R}^f$. In graph neural networks, the aim is to learn node representations that capture the structural information of the graph. The goal is to define a neural network that can aggregate information from neighboring nodes and edges to update the representation of each node iteratively. This is achieved through a technique called message passing, where nodes exchange information with their neighbors to refine their own representations.

3.1. Message passing

Message passing is a crucial process that enables information flow between nodes in a graph. It consists of three steps: message passing, aggregation, and update as illustrated in Fig. 3.

1. Message Passing: In this stage, every node within the graph communicates with its neighboring nodes by sending a message. These messages usually contain relevant details about the node's characteristics or attributes.
2. Aggregation: In this step, each node collects and aggregates the received messages from its neighbors. Aggregation can be performed using various methods like summation, averaging, or max pooling. The goal is to combine the information from neighboring nodes to create a meaningful representation. Let $\mathbf{h}_j^{(t)}$ denote the hidden state of node v_j at iteration t. The information exchange process can be defined as follows:

$$\mathbf{m}_i^{(t)} = \text{AGGREGATE}\left(\{\mathbf{h}_j^{(t-1)}, \mathbf{e}_{ij} \mid v_j \in \mathcal{N}(v_i)\}\right) \tag{1}$$

   where $\mathcal{N}(v_i)$ represents the neighbors of node v_i, and AGGREGATE is a function that aggregates information from neighboring nodes and edges. The aggregated message $\mathbf{m}_i^{(t)}$ captures the information from the neighborhood of node v_i at iteration t.
3. Update: Finally, the aggregated messages are used to update the node's own representation. This update is performed by applying a transformation function UPDATE to the aggregated messages along with the node's current representation. This function is defined by:

$$\mathbf{h}_i^{(t)} = \text{UPDATE}\left(\mathbf{h}_i^{(t-1)}, \mathbf{m}_i^{(t)}\right) \tag{2}$$

   The updated representation captures the node's refined information after considering its neighbors' messages.

By iterating these three steps multiple times, GNNs can capture and propagate information throughout the graph, enabling effective learning and prediction tasks on graph-structured data.

Our GNN architecture is based on encoder–decoder graph neural network (Chami et al., 2022). This architecture leverages an encoder–decoder framework to perform tasks on graph-structured data. The following sections delves into the encoder and decoder components employed in our model, providing a comprehensive understanding of their roles and functionalities.

3.2. Encoder

The encoder–decoder based graph neural network has been successfully applied to various graph-related tasks, such as graph generation, graph translation, and dynamic graph learning (Zhu et al., 2022). It allows for end-to-end learning on graph-structured data and enables the model to capture complex relationships and dependencies within

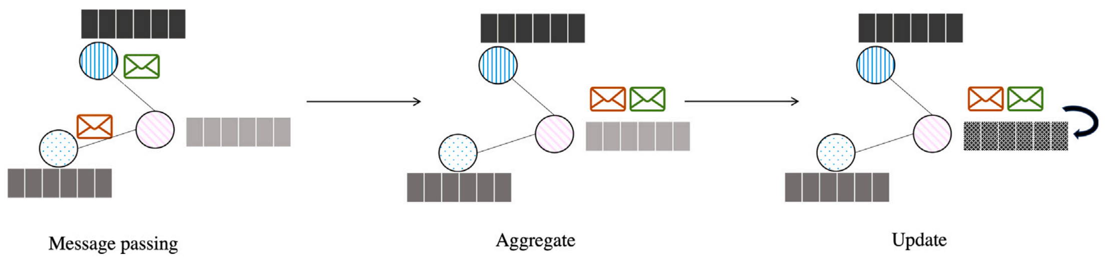


Fig. 3. Learning process in GNNs.

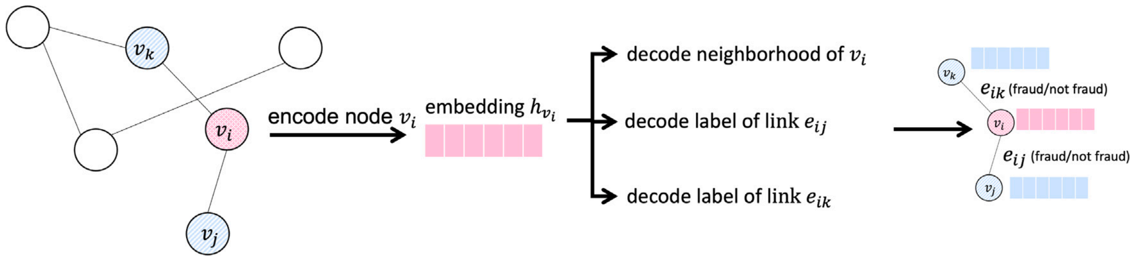

v_k
v_i
v_j
encode node v_i
embedding h_{v_i}
decode neighborhood of v_i
decode label of link e_{ij}
decode label of link e_{ik}
e_{ik} (fraud/not fraud)
e_{ij} (fraud/not fraud)

Fig. 4. Encoder–decoder GNN.

the graph. In the encoder–decoder architecture, the encoder component processes the input graph and learns to encode the graph structure and node features into a latent representation. This latent representation captures the information from the input graph in a meaningful way. The decoder component then takes the encoded representation and decodes it to generate the output graph (see Fig. 4).

The encoder typically consists of multiple layers of graph convolutional operations or other graph neural network operations. These operations perform message passing and update the representations of each node in the graph based on the information from its neighboring nodes. This process allows the model to capture the contextual information and dependencies within the graph.

The encoder maps each node $v_i \in \mathcal{V}$ to a low-dimensional vector representation $\mathbf{h}_{v_i}^{(l)}$ at layer l. This mapping is defined in Eq. (3):

$$\mathbf{h}_{v_i}^{(l+1)} = \sigma\left(\sum_{u\in N(v_i)} \mathbf{W}^{(l)}\mathbf{h}_u^{(l)} + \mathbf{b}^{(l)}\right) \tag{3}$$

where σ is the activation function, $N(v_i)$ represents the neighborhood of node v_i, and $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the learnable weight matrix and bias vector at layer l, respectively.

3.3. Decoder

The decoder, on the other hand, takes the encoded representation and generates the output graph. The decoder may use similar graph convolutional operations to reconstruct the output graph structure and node features from the learned latent representation. It can also incorporate additional layers or operations specific to the task at hand, such as graph pooling or graph classification layers. The decoder aims to decode structural information from the learned node embeddings. It predicts the existence of edges between pairs of nodes.

Fig. 4 illustrates the latent dimension produced by the encoder and the output of the decoder phase. It shows that the encoder first assigns a low-dimensional vector embedding, h_{v_i}, to each node, v_i, taking into account variables such as the node's placement within the graph, its immediate neighboring structure, and/or its distinctive attributes. Subsequently, the decoder extracts specific information specified by the user from the low-dimensional embedding. This information can include details about v_i's local graph neighborhood, such as the identities of its neighbors. By simultaneously optimizing the encoder and decoder, the system is able to efficiently encode the graph structure information into a lower-dimensional embedding space.

In our model, we used the cosine similarity to decode link labels. Indeed, in link prediction tasks, the ranking of node pairs can be determined based on the cosine similarity of their respective node representations. Ma et al. (2020). Cosine similarity is commonly used to measure the similarity between two vectors in a high-dimensional space. In the context of fraud detection, it can be applied to determine the similarity between the node representations of real merchants, fake merchants, and customers. By calculating the cosine similarity between the node vectors of a customer and a real merchant, we can assess the degree of similarity between them. A higher similarity score suggests that the customer's features align more closely with those of a genuine merchant, indicating a lower likelihood of fraud. Conversely, the cosine similarity between a customer and a fake merchant would be lower, as the features of fake merchants differ from those of genuine ones. By comparing the similarity scores, we can identify potential cases of fraud based on the dissimilarity between the customer and the fake merchant. To ensure that the model distinguishes between real and fake merchants effectively, it is important to train the model in a way that encourages higher similarity scores between customers and real merchants, while simultaneously minimizing the similarity between customers and fake merchants. This can be achieved through appropriate training techniques, such as optimizing the model's parameters using loss functions that emphasize these objectives. It is worth noting that during the testing phase, a threshold of 0.5 is set. If the output value of a link falls below this threshold, it indicates that the similarity between the two nodes is relatively low, suggesting a higher likelihood of it being a fraudulent operation.

The overall architecture of the final GNN network utilized in this study is illustrated in Fig. 5. As depicted in the figure, the input graph, generated from our dataset features, undergoes the encoder as the initial step. The encoder comprises multiple sub-layers, starting with the embedding layer. Subsequently, a transformer layer, built upon the multi-head attention mechanism (Shi et al., 2021), follows. To expedite the training process, three linear layers are incorporated, with a batch normalization layer in between (Ioffe and Szegedy, 2015).

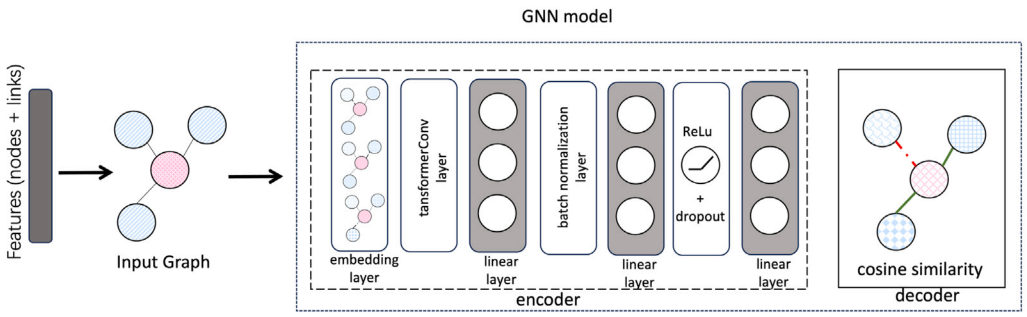


Fig. 5. Encoder–decoder architecture of the proposed model.

Batch normalization facilitates the utilization of higher learning rates and diminishes the necessity for meticulous initialization (Ioffe and Szegedy, 2015). This technique helps in reducing internal covariate shift and accelerates the convergence of the network. It also helps in stabilizing the gradients during the backpropagation process, leading to improved overall performance. Finally, the cosine similarity is computed for each link to yield the label (0 or 1).

4. Dataset description and exploration

To solve the data privacy and quality issue, we use in this study an improved version of the Sparkov dataset (Yeo, 2022). Sparkov-generated synthetic credit card transactions refer to the creation of artificial credit card transactions using the Sparkov platform. Sparkov is an advanced AI-powered system that can generate realistic and synthetic data for various applications, including credit card transactions. Besides, the dataset is suitable for real-time processing since transactions are generated periodically. These synthetic credit card transactions are generated using sophisticated algorithms and machine learning techniques. They mimic the patterns and characteristics of real credit card transactions, including transaction amounts, merchant types, timestamps, and other relevant information.

The dataset consists of two sub-datasets: the transaction dataset and the customer dataset. The transaction dataset comprises 1,391,386 records and 14 columns that document various financial transactions performed by customers. Table 2 lists the features in this dataset with a brief description. Conversely, the customer dataset includes 6450 records and 16 columns that provide customer details. A list of features along with brief descriptions are depicted in Table 3. Together, these datasets offer a comprehensive view of the customers' financial activities and behavior, which can help in developing effective fraud detection models.

4.1. Data exploration

Data exploration is a crucial step in the data analysis process as it allows to gain insights, identify patterns and trends, and uncover hidden relationships within the data. By thoroughly examining our data before implementing any modeling techniques, we can ensure that the results we obtain are accurate and reliable. Additionally, data exploration allows us to detect and address potential issues, such as missing values, outliers, and inconsistencies, which can significantly impact the quality of our analysis. Ultimately, investing time in exploring our data not only enhances our understanding of the dataset but also paves the way for better decision-making and more informed business strategies.

Initially, we investigated the distribution of fraudulent transactions in the dataset, analyzing both the number of fraudulent transactions and the amount involved. Fig. 6 illustrates our findings. Our analysis revealed that the distribution is biased, with only 4.1% of the dataset classified as fraudulent transactions, while the remaining 95.9% comprises normal transactions. Furthermore, we discovered that fraudulent transactions account for a quarter of the total transaction amount.

Table 2
Transaction dataset description.

Feature name	Feature type	Feature description
cc_num	Integer	Customer credit card number
ssn	String	Customer social security number
acct_num	Integer	Account number
profile	String	The profile path which to generate the current transaction
trans_num	Integer	The transaction number
trans_date	Data	The transaction date
trans_time	Time	The transaction time
unix_time	Integer	The Unix time representation for the time and date of the transaction
category	String	Merchant's category
Amt	Float	The amount of money used in the transaction
merchant	String	List of merchandises separated by a comma
Merch_lat	Float	Merchant's latitude
Merch_long	Float	Merchant's longitude
Is_fraud	Binary	If the transaction is a fraud, then this feature is set to 1, else 0.

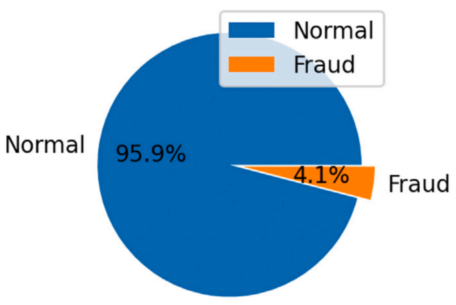
Table 3
Customer dataset description.

Feature name	Feature type	Feature description
cc_num	Integer	Customer's credit card number
ssn	String	Customer's social security number
First	String	Customer's first name
Last	String	Customer 's last name
Gender	Character	Customer's gender. M for male F for female
Street	String	Customer's street address
State	String	Customer's state address
Zip	Integer	Customer's zip code
Lat	Float	Customer's latitude
Long	Float	Customer's longitude
City_pop	Integer	The customer city population count
Job	String	The customer occupation
Dob	Date	Customer date of birth
Acc_num	String	Customer bank account number

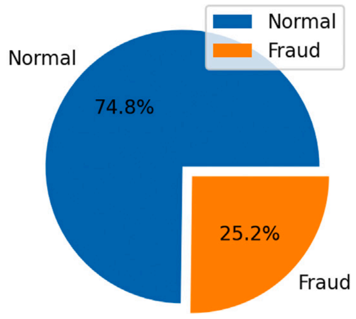
4.2. Pre-processing

Data preprocessing plays a vital role in every data analysis or machine learning endeavor. It involves cleaning, transforming, and organizing raw data to make it suitable for analysis or modeling. This includes tasks such as removing outliers, scaling or normalizing data, and converting categorical data into numerical values. Proper data preprocessing can greatly improve the accuracy and reliability of the final results, as it ensures that the data is consistent and relevant to the problem at hand.

Outliers removal. To identify any outliers in the dataset, we applied the 1.5 times the interquartile range (IQR) rule (Tukey, 1977). This rule involves extracting the first quartile (Q1) and third one (Q3) and calculating the IQR using the formula IQR = Q3 - Q1. Once the IQR is



(a) Fraudulent Transactions Distribution



(b) Fraudulent Amount Distribution

Fig. 6. Distribution of fraudulent samples by transactions number (a) and transactions amount (b).

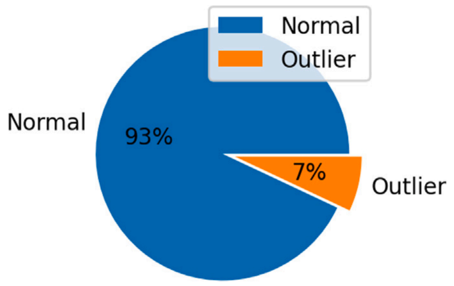


Fig. 7. Outliers distribution.

Table 4
Added features.

Feature name	Feature type	Feature description
distance_from_cust	Float	The distance from the customer and merchant
txn_during_night	Binary	Determine if the transaction occurred during the night or not
trans_weekend	Binary	Determine if the transaction occurred during the weekend or not
avg_txns_amt_1_days	Float	The average amount spent during one day
avg_txns_amt_7_days	Float	The average amount spent during a week
avg_txns_amt_30_days	Float	The average amount spent during a month
nb_txns_1_days	Integer	The number of transactions during one day
nb_txns_7_days	Integer	The number of transactions during a week
Nb_txnx_30_days	Integer	The number of transactions during one month

calculated, the minimum and the maximum range of the distribution are deduced using Eqs. (4) and (5):

$$Min = Q1 - 1.5 \times IQL \tag{4}$$

$$Max = Q3 + 1.5 * IQL \tag{5}$$

By utilizing the aforementioned method, we found that outliers represent 7% of the dataset as depicted in Fig. 7.

One hot-encoding. This step involves transforming categorical attributes such as “state”, “job”, “country”, etc., into numerical data using the one-hot encoding method. This method consists of assigning binary values to categorical columns. For instance the gender (M/F) could be represented as 0/1 values while attributes such as day is encoded using a vector of size 7 (e.g. Monday = [0000000], Sunday =[0000001], etc.).

Data normalization. The dataset features consist of values that span different ranges. To address this, we applied the min–max normalization technique to normalize data. Hence, all the values are transformed to fit into the interval $[0, 1]$.

4.3. Feature engineering

Adding new features from existing ones is a useful way to improve the performance of an AI model as this can provide the model with more information to make better predictions. More precisely, the features that are relevant for detecting fraudulent transactions are those that reflect the customer's behavior and habits. Thus, a data exploration study is conducted in this research. First, the distance between the customer's address and the merchant's address is investigated in order to evaluate the relationship between these features. For instance, if the customer is used to buy from merchants close to his address, a transaction with a very distant merchant would be suspicious. The distribution of the transaction amounts with respect to the distance between the customer and the merchants, shown in Fig. 8, reveals that customers tend to spend more when the merchant is close to them. Consequently, we add a new feature that deduces the distance between the merchant and the customer for each transaction.

Furthermore, the customer maintains some habits like performing most of his purchases during the day or during the night, during the business days, or during the weekends. A further investigation of the data shows that the transactions conducted during the night represent only 29% of the total number of transactions as shown in Fig. 9 Moreover, the fraudulent transactions mostly occur during the night as depicted in Fig. 10. Therefore, adding a feature that determines whether the transaction is performed during the night seems to be relevant for fraud detection. The same investigation is carried out on the transactions performed during business days *vs.* the weekends. Figs. 11 and 12 reveal that most transactions occur during business days but the proportion of fraudulent transactions is higher during the weekends. Therefore, a new feature that indicates if a transaction occurs during the weekends or business days is added in order to reveal possible fraudulent behavior. The average amount and the number of transactions within a specific duration (a day, a week, a month) are also features that are correlated to the customer's behavior and thus help in improving fraud detection performances. All the new added features are depicted in Table 4.

5. Implementation and performance analysis

In this section, we will discuss the steps taken to implement the proposed fraud detection system, the evaluation metrics used to assess its performance, and the testing methodology employed to validate its effectiveness. For this, we first compared our model with another deep learning method namely autoencoders. Then, we compared to related works using the same dataset. The implementation and testing process will provide valuable insights into the capabilities and limitations of

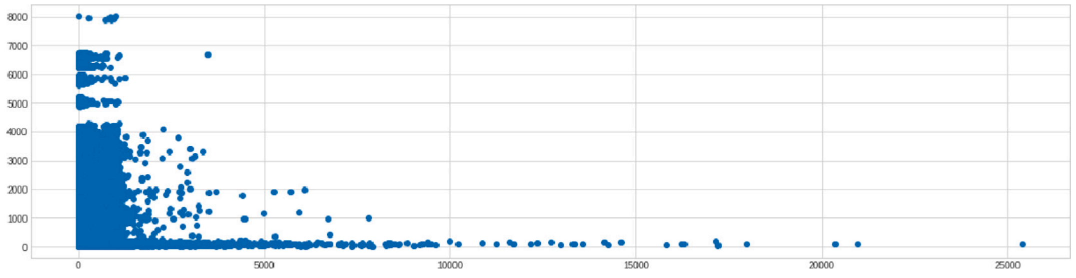


Fig. 8. Distribution of the transaction amounts with respect to the distance between the customer and the merchants.

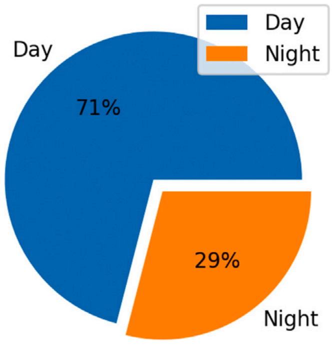


Fig. 9. Transaction proportion during the day *vs.* the night.

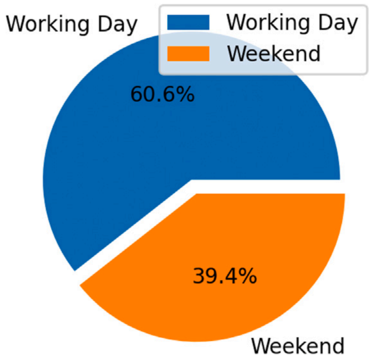


Fig. 11. Transaction proportion during working days *vs.* weekends.

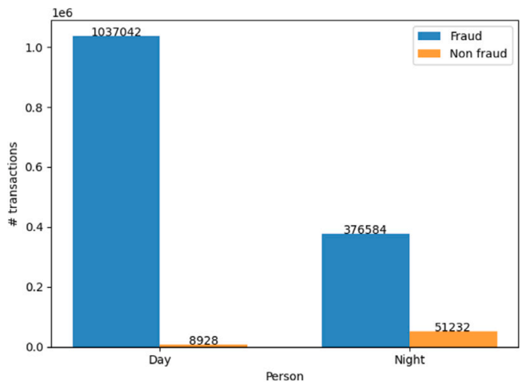


Fig. 10. Fraudulent transaction proportion during the day *vs.* the night.

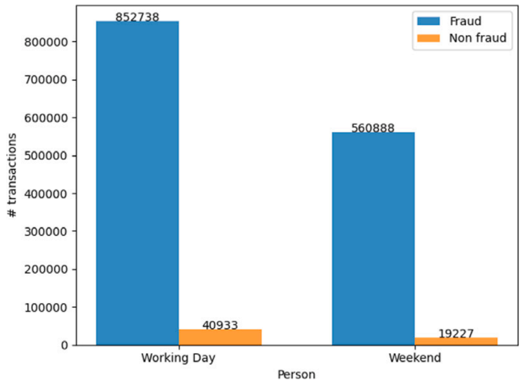


Fig. 12. Fraudulent transaction proportion during business days *vs.* weekends.

the proposed system, guiding future research and development efforts in the field of credit card fraud detection.

5.1. Model parameters

To find the best parameters for our model, we used the Tuner. A tuner refers to an algorithm that helps optimize the hyperparameters of a machine learning model. The tuner's goal is to find the best combination of hyperparameter values that maximizes the model's performance on a given dataset.

Tuners typically work by systematically exploring different hyperparameter values and evaluating the model's performance using a defined metric, such as accuracy or loss. They can use various search strategies, such as grid search, random search, or more advanced techniques like Bayesian optimization or genetic algorithms.

During the tuning process, the tuner iteratively adjusts the hyperparameters and evaluates the resulting models until it converges on the best set of hyperparameters. This helps find the optimal balance between underfitting and overfitting, as well as other trade-offs in the model's performance. The best values retained by the tuner are presented in Table 5.

Class imbalance. An imbalance in class distribution is a prevalent challenge in credit card fraud detection, given the rarity of fraudulent cases compared to legitimate ones. Methods for balancing, such as oversampling and undersampling, were extensively investigated to address this challenge (Cherif et al., 2023). However, oversampling methods can result in the model learning excessively, potentially leading to overfitting, whereas undersampling techniques may result in the loss of information (Stando et al., 2023). Additionally, a classifier developed through the use of sampling methods to artificially equalize data distribution may not be appropriate for a dataset with markedly distinct prevalence rates, as the classifier is trained to perform well specifically on balanced data (Fernando and Tsokos, 2022).

Another alternative to solve the class imbalance while maintaining a realistic distribution involves assigning class weights that are inversely related to the class frequencies in the training data, thereby serving to prevent overfitting. In this research, the weighting approach is preferred over sampling techniques as it does not manipulate the dataset, thus avoiding biased results caused by artificially generated samples of the minority class. In order to deal with the class imbalance problem, the weighted binary cross-entropy loss function is used in this study.

Table 5
The best model parameters.

Parameter	Value
Batch size	64
Learning rate	0.001
Model attention heads	2
Model dense neurons	64
Model dropout rate	0.5
Model embedding size	128
Model layers	2
Model output	16
Scheduler gamma	0.9
sgd momentum	0.9
Weight decay	0.001

It aims to assign a higher weight to the fraudulent transactions since the 'Fraud' class represents a minority and, it assigns lower weight to 'non-fraud' transactions. This loss function is defined by:

$$\delta = -\frac{1}{N}\sum_{i=1}^{N}[w_1 \cdot x_i \log(\hat{p}_i) + w_0 \cdot (1-x_i)\log(1-\hat{p}_i)] \tag{6}$$

where N is the number of samples, x_i is the true label (1 for 'fraud' and 0 for 'non-fraud'), and $\hat{p}_i$ is the predicted probability for the current sample being a fraud. w_1 and w_0 weigh respectively fraudulent and non-fraudulent transactions.

5.2. Experimental settings

Evaluation. To evaluate our model, we report the precision, recall and F1-score. The selection of these metrics is due to their fit for the class imbalance issue.

Precision measures the proportion of correctly predicted positive instances (fraudulent cases) out of all instances predicted as positive. It helps to assess the model's ability to avoid false alarms and accurately identify fraud cases.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{7}$$

Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive instances correctly identified by the model. It evaluates the model's ability to capture all fraudulent cases, minimizing false negatives.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{8}$$

The F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics. It considers both false positives and negatives, making it a suitable metric for class imbalance problems like fraud detection. It rewards models that achieve high precision and recall simultaneously.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

These metrics are crucial for fraud detection because they help evaluate the model's performance in identifying fraudulent cases accurately while minimizing false alarms and missed fraud cases.

In addition to the basic metrics, we also used the ROC curve. The ROC curve is a graphical representation that evaluates the performance of a binary classification model. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various classification thresholds.

$$\text{TPR} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{10}$$

$$\text{FPR} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}} \tag{11}$$

The ROC curve provides a visual summary of the trade-off between the model's ability to correctly identify positive cases (fraudulent transactions) and its tendency to incorrectly classify negative cases (legitimate transactions) as positive. The ROC curve is particularly useful in fraud detection due to class imbalance. In such case, accuracy alone can be misleading, as a model that predicts all cases as negative can still achieve a high accuracy if the negative class is dominant.

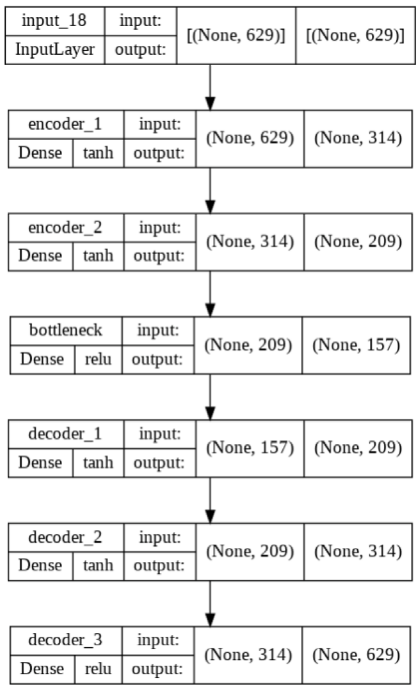

input_18 | input: | [(None, 629)] | [(None, 629)]
InputLayer | output:
encoder_1 | Dense | tanh | input: (None, 629) | output: (None, 314)
encoder_2 | Dense | tanh | input: (None, 314) | output: (None, 209)
bottleneck | Dense | relu | input: (None, 209) | output: (None, 157)
decoder_1 | Dense | tanh | input: (None, 157) | output: (None, 209)
decoder_2 | Dense | tanh | input: (None, 209) | output: (None, 314)
decoder_3 | Dense | relu | input: (None, 314) | output: (None, 629)

Fig. 13. Auto-encoder architecture.

Finally, we used the precision–recall metric. The precision–recall curve is a good metric to evaluate the performance of a classification model when the data is imbalanced or when the cost of false positives and false negatives is not equal. It is particularly useful in scenarios where the focus is on the positive class, such as detecting rare events or in fraud detection. The precision–recall curve provides insights into the trade-off between precision and recall at different classification thresholds, allowing practitioners to choose an appropriate threshold based on their specific requirements. The precision–recall curve is essential for effectively detecting fraud with high confidence. This is crucial for financial services, as it ensures minimal disruption to normal cases. Balancing service experiences and the coverage ratio of malicious nodes is of utmost importance.

Comparison methods. To assess the performance of our model, a comparative study is conducted. First, we built another deep learning model on the same dataset using the autoencoder deep learning approach. In this approach, we modeled the credit card fraud detection problem as an anomaly detection. We used a deep learning architecture to build a representation of the model using non fraudulent transactions only. Consequently, the model fails to reconstruct fraudulent transactions which facilitate their detection as abnormal behavior. The architecture of the deep learning network used is captured in Fig. 13.

Second, we compared the performance of our model to the recent studies that used the same dataset.

5.3. Results

We developed our model using Google Colab framework, a powerful cloud-based platform providing GPU support. The model was trained using 500,000 samples that were divided into training and testing sets, with 80% allocated for training and 20% for testing.

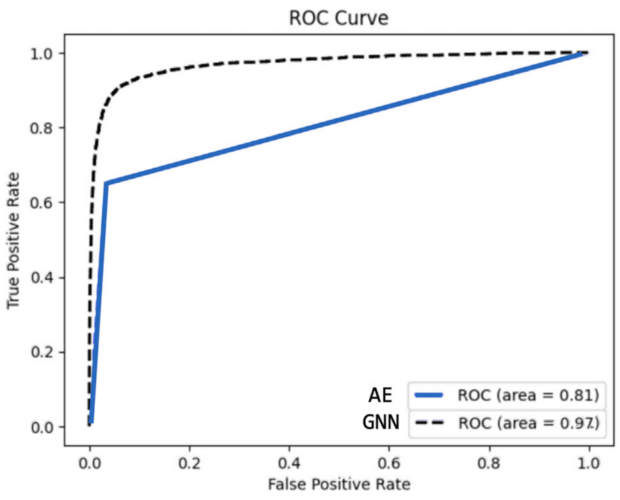


Fig. 14. ROC for AE (solid line) and GNN (dashed line).

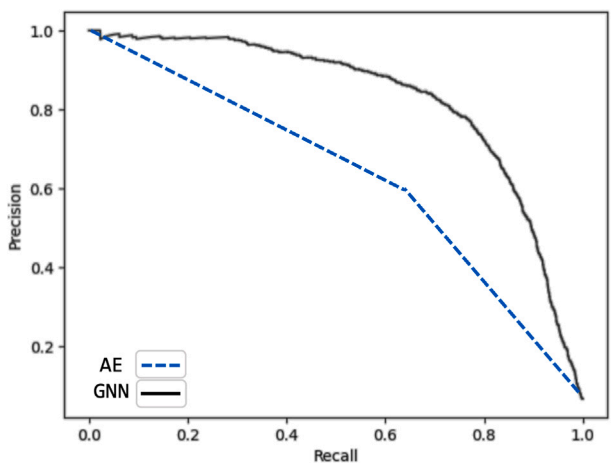


Fig. 15. Precision recall for AE (dashed line) and GNN (solid line).

Table 6
Encoder–decoder GNN vs. Autoencoder.

Category	Class	Precision	Recall	F1-score
AE	0	0.97	**0.97**	0.97
AE	1	0.65	0.64	0.65
Encoder–decoder GNN	0	**0.99**	**0.97**	**0.98**
Encoder–decoder GNN	1	**0.66**	**0.88**	**0.75**

Table 6 displays the results of both models using the three metrics precision, recall and F1-score. We reported the results for positive and negative classes to better analyze the system performance. From the results, we can observe that the AE has comparable results in terms of recall and F1-score, for class 0, while the encoder–decoder GNN has better results for class 1 (fraud). These results indicate that the GNN method is more effective in detecting fraudulent transactions.

The area under the ROC curve (AUROC) is a commonly used metric to quantify the overall performance of a classification model. A higher AUROC indicates better discrimination between positive and negative cases. Fig. 14 displays the AUROC for the improved AE and our GNN model wherein the AE method yielded an AUROC of 0.81 while the GNN method yielded an AUROC of 0.97. It can be inferred that the GNN method outperforms the AE method in terms of classification accuracy. Therefore, the GNN method is more effective than the AE method in differentiating between the fraudulent and non-fraudulent transactions.

Fig. 15 depicts the precision–recall curves for AE and GNN models.

The area under the precision–recall curve (AUC-PR) is a commonly used metric to assess the overall performance of a fraud detection model. The higher the AUC-PR score, the better the model performs in identifying fraudulent cases while minimizing false alarms. Therefore, precision–recall curves are a valuable tool for evaluating the performance of fraud detection models in the presence of class imbalance, where true fraud cases are typically rare compared to non-fraud cases. Fig. 15 indicates that our GNN model highly outperforms the autoencoder approach.

Table 7 displays the comparative analysis of our work against works using the same dataset or a GNN architecture. It indicates that our approach outperformed both traditional methods and previous works using GNN. The GNN method achieved a high precision of 0.82, recall of 0.92, F1-score of 0.86, and F1- AUROC of 0.92. Comparatively, the traditional methods did not report recall, precision, or F1-score, but achieved an accuracy range of 0.96 to 0.99. However, the accuracy is not an appropriate metric for fraud detection problem due to the imbalanced dataset. It is worth noting that Afriyie et al. (2023) achieved higher AUROC. However, the main limitation of this method is the use of undersampling technique which reduces the dataset size. Most of the related work did not report the recall which is an important metric to consider in fraud detection. Besides, our model achieves similar accuracy to Afriyie et al. (2023) and is outperformed by Jose et al. (2023). However, Jose et al. (2023) have used oversampling techniques that lead to overfitting and did not report the most valuable metrics to the fraud detection problem namely, recall, F1-score and AUROC.

6. Conclusion

In conclusion, the encoder–decoder based GNN model employed for credit card fraud detection using the large Sparkov dataset has showcased promising results. By comparing its performance to an encoder–decoder model and existing works, we have observed significant improvements in precision, recall, F1 score, and ROC metrics.

The architecture of the GNN network effectively captures the complex relationships and dependencies within the credit card transaction data. The encoder component extracts meaningful representations from the input data, while the decoder component reconstructs the input based on the learned representations. This process enables the model to identify fraudulent patterns and anomalies with high accuracy.

Furthermore, the GNN model benefits from the utilization of graph converter to efficiently represent and process graph data and batch normalization in the training process. This technique helps in reducing internal covariate shift and accelerates the convergence of the network. It also helps in stabilizing the gradients during the backpropagation process, leading to improved overall performance.

By leveraging the encoder–decoder framework and incorporating batch normalization, the GNN model achieves remarkable precision, recall, F1 score, and ROC values. This demonstrates its potential to effectively detect credit card fraud on a large scale, providing enhanced security for financial transactions. The model's performance surpasses existing works, making it a valuable addition to the field of fraud detection and prevention.

Our solution relies on integrating the distance between the merchant and the customer as a pivotal feature in guiding the fraud detection model. While acquiring this information may present challenges for certain systems, we are optimistic that the increasing prevalence of smart cities and IoT will facilitate easier access to this data for banking systems. The growing popularity of contactless payments enables banks to obtain real-time customer location data, especially when customers use their phones to conduct transactions. Additionally, some banking applications prompt the activation of GPS for this purpose.

In future works, we intend to develop techniques to scale the suggested model to handle large-scale graphs with millions or billions

Table 7
Comparative study.

Method	Algorithm	Best performance				
		Accuracy	Precision	Recall	F1-score	AUROC
Liu et al. (2018)	GNN	–	0.79	–	–	0.92
Liu et al. (2021a)	GNN	–	0.57	–	–	0,81
Afriyie et al. (2023)	Traditional	0.96	–	–	–	**0.98**
Jose et al. (2023)	Traditional	**0.99**	–	–	–	–
Ours	Encoder–decoder GNN	0.97	**0.82**	**0.92**	**0.86**	0.92

of nodes and edges. This could involve exploring distributed computing frameworks and efficient graph sampling strategies to enable efficient training and inference on massive graphs.

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.

Acknowledgments

This research work was funded by Institutional Fund Projects under Grant No. (IFPRC-032-612-2020). Therefore, authors gratefully acknowledge technical and financial support from the Ministry of Education and King Abdulaziz University, Jeddah, Saudi Arabia.

References

Afriyie, J.K., Tawiah, K., Pels, W.A., Addai-Henne, S., Dwamena, H.A., Owiredu, E.O., Ayeh, S.A., Eshun, J., 2023. A supervised machine learning algorithm for detecting and predicting fraud in credit card transactions. Decis. Anal. J. 6, 100163. http://dx.doi.org/10.1016/j.dajour.2023.100163.

Bruna, J., Zaremba, W., Szlam, A., LeCun, Y., 2014. Spectral networks and locally connected networks on graphs. arXiv:1312.6203.

Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., Murphy, K., 2022. Machine learning on graphs: A model and comprehensive taxonomy. arXiv:2005.03675.

Chaudhary, A., Mittal, H., Arora, A., 2019. Anomaly detection using graph neural networks. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing. COMITCon, pp. 346–350. http://dx.doi.org/10.1109/COMITCon.2019.8862186.

Cherif, A., Badhib, A., Ammar, H., Alshehri, S., Kalkatawi, M., Imine, A., 2023. Credit card fraud detection in the era of disruptive technologies: A systematic review. J. King Saud Univ. - Comput. Inf. Sci. 35 (1), 145–174. http://dx.doi.org/10.1016/j.jksuci.2022.11.008.

Chung, J., Lee, K., 2023. Credit card fraud detection: An improved strategy for high recall using KNN, LDA, and linear regression. Sensors 23 (18), 7788. http://dx.doi.org/10.3390/s23187788.

Deng, A., Hooi, B., 2021. Graph neural network-based anomaly detection in multivariate time series. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35, pp. 4027–4035, arXiv:2106.06947.

Fernando, K.R.M., Tsokos, C.P., 2022. Dynamically weighted balanced loss: Class imbalanced learning and confidence calibration of deep neural networks. IEEE Trans. Neural Netw. Learn. Syst. 33 (7), 2940–2951. http://dx.doi.org/10.1109/TNNLS.2020.3047335.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.

Jose, S., Devassy, D., Antony, A.M., 2023. Detection of credit card fraud using resampling and boosting technique. In: 2023 Advanced Computing and Communication Technologies for High Performance Applications. ACCTHPA, pp. 1–8. http://dx.doi.org/10.1109/ACCTHPA57160.2023.10083376.

Keramatfar, A., Rafiee, M., Amirkhani, H., 2022. Graph Neural Networks: A bibliometrics overview. Mach. Learn. Appl. 10, 100401. http://dx.doi.org/10.1016/j.mlwa.2022.100401.

Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, URL: https://openreview.net/forum?id=SJU4ayYgl.

Li, P., Wang, J., Qiao, Y., Chen, H., Yu, Y., Yao, X., Gao, P., Xie, G., Song, S., 2021. An effective self-supervised framework for learning expressive molecular global representations to drug discovery. Brief. Bioinform. 22 (6), bbab109. http://dx.doi.org/10.1093/bib/bbab109.

Liu, Y., Ao, X., Qin, Z., Chi, J., Feng, J., Yang, H., He, Q., 2021a. Pick and choose: A GNN-based imbalanced learning approach for fraud detection. In: Proceedings of the Web Conference 2021. ACM, New York, NY, USA, http://dx.doi.org/10.1145/3442381.3449989.

Liu, Z., Chen, C., Yang, X., Zhou, J., Li, X., Song, L., 2018. Heterogeneous graph neural networks for malicious account detection. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. CIKM '18, Association for Computing Machinery, New York, NY, USA, pp. 2077–2085. http://dx.doi.org/10.1145/3269206.3272010.

Liu, Y., Zeng, K., Wang, H., Song, X., Zhou, B., 2021b. Content matters: A GNN-based model combined with text semantics for social network cascade prediction. In: Karlapalem, K., Cheng, H., Ramakrishnan, N., Agrawal, R.K., Reddy, P.K., Srivastava, J., Chakraborty, T. (Eds.), Advances in Knowledge Discovery and Data Mining. Springer International Publishing, Cham, pp. 728–740. http://dx.doi.org/10.1007/978-3-030-75762-5_57.

Ma, Y., Guo, Z., Ren, Z., Tang, J., Yin, D., 2020. Streaming graph neural networks. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '20, Association for Computing Machinery, New York, NY, USA, pp. 719–728. http://dx.doi.org/10.1145/3397271.3401092.

Rb, A., Kr, S.K., 2021. Credit card fraud detection using artificial neural network. Glob. Transit. Proc. 2 (1), 35–41. http://dx.doi.org/10.1016/j.gltp.2021.01.006, 1st International Conference on Advances in Information, Computing and Trends in Data Engineering (AICDE - 2020).

Sáez, J.A., Krawczyk, B., Woźniak, M., 2016. Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. Pattern Recognit. 57, 164–178. http://dx.doi.org/10.1016/j.patcog.2016.03.012.

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., Sun, Y., 2021. Masked label prediction: Unified message Passing model for semi-supervised classification. arXiv:2009.03509.

Singh, I., Aditya, N., Srivastava, P., Mittal, S., Mittal, T., Surin, N.V., 2023. Credit card fraud detection using neural embeddings and radial basis network with a novel hybrid fruitfly-fireworks algorithm. In: 2023 3rd International Conference on Intelligent Technologies. CONIT, pp. 1–7. http://dx.doi.org/10.1109/CONIT59222.2023.10205378.

Stando, A., Cavus, M., Biecek, P., 2023. The effect of balancing methods on model behavior in imbalanced classification problems. arXiv:2307.00157.

Thomas, J.M., Moallemy-Oureh, A., Beddar-Wiesing, S., Holzhüter, C., 2023. Graph neural networks designed for different graph types: A survey. arXiv:2204.03080.

Tukey, J., 1977. Exploratory Data Analysis. In: Addison-Wesley Series in Behavioral Science, Vol. 2, Addison-Wesley Publishing Company, number, URL: https://books.google.com.sa/books?id=UT9dAAAAIAAJ.

Wu, S., Sun, F., Zhang, W., Xie, X., Cui, B., 2022. Graph neural networks in recommender systems: A survey. arXiv:2011.02260.

Ye, Z., Kumar, Y.J., Sing, G.O., Song, F., Wang, J., 2022. A comprehensive survey of graph neural networks for knowledge graphs. IEEE Access 10, 75729–75741. http://dx.doi.org/10.1109/ACCESS.2022.3191784.

Yeo, H., 2022. Sparkov dataset for credit card fraud detection. https://github.com/atoti/notebooks/tree/master/notebooks/credit-card-fraud-detection, [Online; accessed 19-December-2022].

Zhu, T., Lin, Y., Liu, Y., 2017. Synthetic minority oversampling technique for multiclass imbalance problems. Pattern Recognit. 72, 327–340. http://dx.doi.org/10.1016/j.patcog.2017.07.024.

Zhu, Y., Lyu, F., Hu, C., Chen, X., Liu, X., 2022. Encoder-decoder architecture for supervised dynamic graph learning: A survey. CoRR abs/2203.10480.