

Detecting bitcoin fraud using graph neural networks**Renad Saleh Alsweed^{a*} and Dina M. Ibrahim^b**^a*Department of Computer Science, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia*^b*Department of Information Technology, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia***CHRONICLE***Article history:*

Received October 14, 2025

Received in revised format

October 30, 2025

Accepted February 14 2026

Available online

February 14 2026

*Keywords:**Bitcoin**Fraud detection**Graph neural network**Gated Graph neural network**Elliptic dataset***ABSTRACT**

The rise of Bitcoin has revolutionized the financial landscape, but it has also opened the door to a new era of criminal activities. Criminals take advantage of the anonymity provided by Bitcoin to conduct illicit transactions and engage in fraudulent activities. To address this issue, this paper proposes a detection model using Graph Neural Networks (GNNs) to detect fraudulent activities in the complex financial systems of Bitcoin. From the GNNs, we use EvolveGCN and EvolveGGCN to compare between them and find a powerful model that can investigate the network construction of financial transactions and capture patterns and anomalies that traditional methods may miss. In the literature, there have been a limited number of studies on Bitcoin fraud detection using GNNs, especially EvolveGGCN. Therefore, in this paper, we focus on the detection of fraud in the Bitcoin network using EvolveGCN and EvolveGGCN. In addition, we used a more recent dataset called Elliptic++, which is an extension of the Elliptic Dataset. The dataset provides valuable information on the behavior and patterns of fraudulent actions in the Bitcoin network. The results show that EvolveGGCN outperforms other models in terms of precision, recall, F1 score, and micro-F1 score. With an F1-score of 0.90 and micro-F1 of 0.93 for detecting illicit transactions in the early time steps.

© 2026 by the authors; licensee Growing Science, Canada.

1. Introduction

In 2008, Satoshi Nakamoto's vision, embodied in Bitcoin, laid the foundation for the cryptocurrency. Bitcoin is an application running on the Blockchain software platform, which is a global leader in digital assets. Every cryptocurrency transaction is stored in a chronologically ordered chain of blocks that contain all operations and records (Vujicic et al., 2018) and is maintained by miners. The structured group of nodes in the Bitcoin system, known as miners, enables the decentralized consensus mechanism. The miners authenticate every transaction. As a result, the Bitcoin system is more secure, and the fundamental tenet of the cryptocurrency that maintains trust in an untrusted environment is upheld without the necessity for a third party. In exchange, miners get paid transaction fees for validating transactions.

Since then, Bitcoin has become the most popular cryptocurrency among its contemporaries, achieving a level of adoption that previous digital currencies were unable to match. As of March 19, 2020, there are around 18,200,000 bitcoins in circulation, each worth around \$5,000. As of this writing, the market capitalization of Bitcoin is around \$98,584,000,000 (Nerurkar et al., 2021a,b). This rise of Bitcoin has revolutionized the financial landscape, but it has also opened the door to a new era of criminal activities because it offers a certain amount of anonymity by enabling users to establish an infinite number of wallets using alias addresses, making it difficult to determine who is using the account. Criminals use this to conduct illicit transactions for Bitcoin fraud, which encompasses a range of illicit practices involving the misuse or theft of digital currencies. According to the blockchain data platform Chainalysis, there is an estimated \$20.6 billion worth of illicit cryptocurrency transactions volume in 2022. One year later, they updated the estimate for 2022 to \$39.6 billion (Chainalysis, 2024) which has become a significant concern for both individuals and organizations.

* Corresponding author

E-mail address 451214497@qu.edu.sa (R. S. Alsweed)

ISSN 2561-8156 (Online) - ISSN 2561-8148 (Print)

© 2026 by the authors; licensee Growing Science, Canada.

doi: 10.5267/j.ijds.2026.2.005

Simultaneously, Bitcoin records and makes available data regarding all completed transactions, creating possibilities for data mining to detect questionable behavior patterns within this network. The questionable patterns of behavior within this network, deep learning is a powerful set of techniques for identifying complex patterns in unstructured data. We chose Bitcoin because, for cybercriminals, it was the cryptocurrency of choice, probably because of its high liquidity and value. For our study, we choose graph neural networks (GNNs) because they can examine the network structure of financial transactions and identify patterns and anomalies that conventional rule-based and machine learning methods might overlook, they have become a potent tool for identifying fraudulent activity in intricate financial systems. However, there are not many studies on bitcoin fraud detection using GNN. The approach we will take is to raise the performance of the model using the new elliptic data set “Elliptic++”. It is an extension of the Elliptic Dataset (2019) that has 822k wallet addresses and 203k Bitcoin transactions, allowing graph data to be used to detect illicit addresses in the Bitcoin network as well as fraudulent transactions (Elmougy & Liu, 2023).

The main contributions in this paper are:

- Proposing a detection model using Graph Neural Networks (GNNs) to detect fraudulent activities in the complex financial systems of Bitcoin.
- Illustrating the influence of using EvolveGCN and EvolveGGCN to compare between them and find a powerful model that can examine the network construction of financial transactions and identify trends and irregularities that conventional approaches could detect.
- Analyzing these Graph Neural Networks using a more recent dataset called Elliptic++, which is an extension of the Elliptic Dataset and provides valuable information on the behavior and patterns of fraudulent activities in the Bitcoin network.
- Assessing the proposed work by contrasting its performance indicators with those of several models presented in earlier research on Bitcoin fraud detection.

The remainder of the paper's outline: While Section 3 describes the study's materials and methods, Section 2 provides an overview of the literature review. Section 4 illustrates the method, while Section 5 shows the experimental setup. Section 6 presents the comparative analysis and discussion. In Section 7, we conclude the paper.

2. Literature Review

Fraud detection is crucial in various industries. Prevents financial losses and protects revenue, safeguards reputation and customer trust, ensures compliance with regulations, and improves customer experience and security, among other things. In that regard, advances in machine learning helped with more accurate and easier fraud detection methods. For example, the work presented in (Hajek et al., 2023) recommends using under-sampling in conjunction with an XGBoost-based fraud detection framework to prevent extreme class imbalance and overfitting. A large dataset with over 6 million transactions was used to empirically validate the system. Others suggest a completely different approach; as authors in (Sun et al., 2023), to measure the linguistic components of textual risk disclosures, they look at the part of speech (POS) for every word in the text and determine the percentage of different POS word categories, and version August 10, 2025 submitted to Journal Not Specified 3 of 13 then use these POS features to build machine-learning algorithms that identify financial fraud.

Authors in (Varmedja et al., 2019) compare the performance between different algorithms such as logistic regression, random forest, naive Bayes, and multilayer perceptron and conclude that the random forest algorithm gives the best results, in a precision of 96.38% according to their testing. Some studies use unsupervised learning, as in study (Santos et al., 2019; Lucas et al., 2020), and study (Yu et al., 2019) where unknown data features and patterns are easy to locate but difficult to compute and the unlabeled input makes it less accurate. Where supervised learning, like the work presented in (Iqbal et al., 2016) and (Berrar, 2016), is more accurate and trustworthy, understanding the input and assigning labels to it is essential, and it takes longer to compute during the training phase. However, deep learning reduces the need for extensive physical feature engineering, which is often required for traditional machine-learning approaches. And to detect financial fraud gives it more advantages, such as increased precision and the capacity to manage more intricate data. For example, (Xiuguo & Shengyong, 2022) intends to create an improved system for identifying financial fraud by utilizing LSTM, and GRU approaches based on a combination of textual data from management comments in the yearly reports of 5130 Chinese recorded companies and statistical features extracted from financial declarations. The recommended deep learning methods achieved better than conventional machine learning methods. Specifically, the LSTM and GRU approaches in the testing samples achieved high correct rates of classification 94.98% and 94.62% respectively, and the ML approach with the highest accuracy in ML approaches is ANN with 90.31%.

Another example in study (Jurgovsky et al., 2018) describes the task of detecting credit card fraud as one of sequence classification and integrating transaction sequences using Long Short-Term Memory (LSTM) networks and comparing it to a standard Random Forest (RF) classifier and finds that for offline transactions in case the presence of cardholder in person at the merchant, the LSTM enhances detection accuracy. Similarly, (Fiore et al., 2019) employ Generative Adversarial Networks (GAN) to identify credit card fraud, and they trained a GAN to generate instances of minority classes that resembled

each other. These combined using the training data to generate a set of augmented training to increase a classifier's efficacy, and it got a very high precision of 95.8%. Similarly, authors in (Kim et al., 2019), compare a hybrid ensemble of numerous machine-learning methods and feed-forward neural networks, where the deep-learning model performs better than the machine learning method. In the post-launch test, the deep learning-based model outperformed the combined ensemble model at analogous alert degrees, demonstrating improvements in transaction level recall of +3.8%, card level recall of +2.1%, and cost reduction rate of +5.5%. Some use different ways, (Deng et al., 2020) analyzes user activities on platforms based on digital payment and detects fraud users by using the Adversarial Autoencoder (AAE) which has a higher performance than the ML model. with only 10% of labeled data, it gets a 95% in precision and 93.5% in accuracy.

Our paper focuses on detecting blockchain types of fraud. In the previous studies, we find some similar to our line of research, like the work in (Chen et al., 2021) that compares supervised and unsupervised ML approaches for bitcoin theft detection and finds that the supervised approaches have better results. In addition to the study (Lee et al., 2020) for detecting illegal bitcoin transactions and the study in (Nerukar et al., 2021) using a supervised machine learning model for identifying illegal bitcoin activities. We also find multiple studies that use deep learning on Ethereum accounts as the work presented in (Duan et al., 2022) that uses GraphSAGE and GCN to detect phishing accounts and compare them to other deep learning models. In addition, the study in (Liu et al., 2022) detects fraud through smart contracts in Ethereum transactions using Heterogeneous Graph Transformer Networks (S-HGTNs), and the other one (Tan et al., 2021) uses GNN on Ethereum-based transaction records and has an accuracy of 95%. The authors in (Patel et al., 2020) suggest a single-class graph neural network that can effectively represent interdependencies is used to identify anomalies in Ethereum transactions. Correspondingly, the study in (Song et al., 2023) uses the VAE-Transformer model to find anomalies in the Ethereum network. Using deep learning to detect fraud in the Bitcoin network, like the paper in (Alarab et al., 2020) where it combines node embeddings from GCNs with a multi-layer perceptron after a hidden layer created by a linear transformation of node features and got 90% precision and high accuracy of 97.4%. In a similar way, authors in (Zheng, 2022) use a GRU-GAT-based model that employs graph attention networks for feature weighting and recurrent neural networks for feature extraction. According to study (Tian et al., 2021), it uses an attention-based graph neural network (AT-GNN). The idea is that an auto-encoder is used to identify hidden historical features and graph neural networks are used to capture the transaction network's structure. We chose to focus on Bitcoin fraud detection using deep learning because, as mentioned in (Chainalysis, 2024) Bitcoin is one of the biggest, if not the biggest cryptocracy that criminals use, and because of its popularity, it will affect a wider range of people. for the model However, we chose the EvolveGCN and EvolveGGCN models because of their limited use in Bitcoin fraud detection and their positive results that could be improved. Hence, we found a recent dataset in [4] that is bigger and larger, and we believe that it can improve our model's performance.

Table 1
Transactions Dataset Description

Column	#	Description	Size	Type
TxId	1	Description	203,769	int
Time Step	1	Description	203,769	int
Local features	93	Description	203,769	float
Aggregate features	72	Description	203,769	float
in_tx_degree	1	Description	202,804	float
out_txs_degree	1	Description	202,804	float
total_BTC	1	Description	202,804	float
fees	1	Description	202,804	Float
size	1	Description	202,804	float
num_input_addresses	1	Description	202,804	float
num_output_addresses	1	Description	202,804	float
in_BTC_min	1	Description	202,804	float
in_BTC_max	1	Description	202,804	float
in_BTC_mean	1	Description	202,804	float
in_BTC_median	1	Description	202,804	float
out_BTC_min	1	Description	202,804	float
out_BTC_max	1	Description	202,804	float
out_BTC_mean	1	Description	202,804	float
out_BTC_total	1	Description	202,804	Float
out_BTC_mean	1	Description	202,804	Float
class	1	Description	203,769	Int

An initial exploration of the dataset reveals an imbalanced distribution of transactions across classes. As illustrated in Fig. 1, the dataset contains 4,545 transactions identified as illicit, 42,019 transactions categorized as licit, and a significantly larger group of 157,205 transactions with unknown classifications. This imbalance necessitates careful consideration during model selection and evaluation to ensure robust and generalizable results.

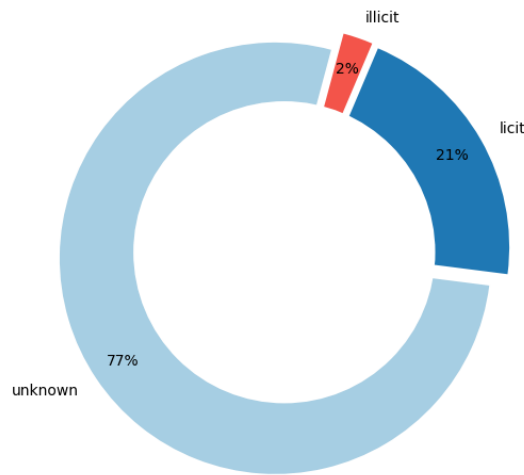


Fig. 1. Distribution of the Bitcoin transactions dataset.

The distribution of data points across the three classes is displayed by time step in Fig. 2. Normalization and standardization transformations are necessary because of the underlying class imbalance that exists between the licit and illicit classes. In order to reduce imbalance and aid in model convergence, scaling each feature in the transaction’s dataset using the various scaling techniques (explained in the Dataset Preprocessing section) transforms the features.

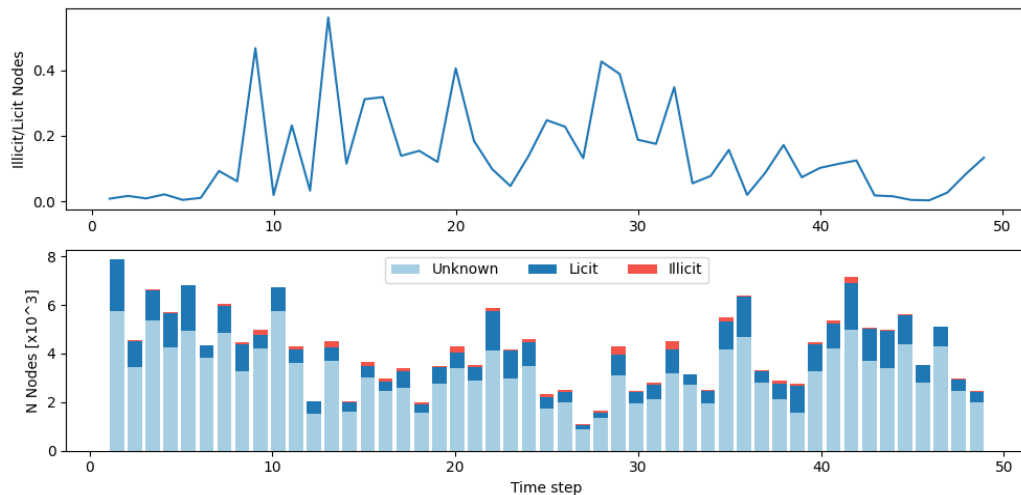


Fig. 2. (Top) The rate of illicit transactions and licit wallets over time steps, (Bottom) Distribution of the Bitcoin transactions dataset over time steps.

Each node is linked to a time step that provides temporal information by estimating the time at which the Bitcoin network will confirm the transaction. The 49 different time stages are uniformly spaced and separated by roughly two weeks. There are no linkages connecting distinct time steps; instead, each time step has a single interconnected component of actions that emerged on the blockchain within three hours of one another. There are between 1,000 to 8,000 nodes per every time step, which is a fairly constant quantity across time.

3. Materials And Methods

The Evolve mechanism empowers GNN to handle dynamic or evolving graph-structured data. It leverages an evolutionary algorithm to progressively update the GNN model parameters as the graph evolves over time. The goal is to study a GNN model sequence that captures the temporal dependencies and changes in the graph construction and node attributes (Pareja et al., 2020).

3.1 Dynamic Bitcoin Transaction Graph and Temporal Split

We use Elliptic++, an extension of Elliptic with 49 uniformly spaced time steps (≈ 2 weeks apart). Each time step forms a connected component without inter-step edges; nodes are transactions, edges connect transactions occurring within a 3-hour window, and node features include local (93) and aggregate (72) attributes plus structural statistics (Table 1). To emulate real deployment and avoid look-ahead leakage, we adopt a temporal split: time steps 1–34 for training/validation and 15 future

steps (35–49) for testing. This “train-past/test-future” setting matches operational fraud detection where future distributions may drift. Unknown-label transactions are excluded to avoid label noise; remaining counts are reported in Section 3 (class imbalance persists).

3.2 Preprocessing and Input Tensors

As a pre-processing step, we Min-Max scale Elliptic++ features introduced in the extension (kept consistent across time using train statistics). For each time step, we build tensors: node features, edge index, and binary labels (0=illicit, 1=licit). Batches are single time steps processed in chronological order; gradients update model parameters that evolve across steps.

3.3 EvolveGCN and EvolveGGCN (Architectures and Training)

Graph Convolutional Networks (GCNs) have developed as a great tool for graph-based machine learning tasks. They leverage the message-passing paradigm to aggregate features from neighboring nodes, enabling the learning of node representations that capture the inherent structure of the graph. EvolveGCN has been designed to address the limits of traditional GCNs in handling dynamic graphs and has shown promising results in tasks such as node organization and link prediction on evolving graphs (Pareja et al., 2020). A GCN backbone with evolving weight matrices updated by a gated recurrent unit (GRU) across time; forward at time t . We use three graph conv blocks with BatchNorm1d after each conv and Dropout=0.5 after activation, followed by a linear classifier and Sigmoid (Table 2).

Table 2
EvolveGCN and EvolveGGCN Models' architectures

Model	Layer stack	Hidden/Filters	Norm/Dropout	Head
EvolveGCN	3× GCNConv → ReLU → BatchNorm → Dropout	200 per layer	BN(200), Dropout = 0.5	Linear(200→1) + Sigmoid
EvolveGGCN	3× GatedGraphConv(num_layers = 10) → ReLU → BN → Dropout	200 per layer	BN(200), Dropout = 0.5	Linear(200→1) + Sigmoid

Gated Graph Convolutional Networks (Gated GCNs) represent a significant advancement in Graph Neural Networks (GNNs). They address the limitations of standard GCNs by incorporating gating mechanisms, leading to improved performance on tasks involving complex graph structures and sequential information. These gating mechanisms regulate the flow of information during the message-passing process between nodes. Incorporating gating mechanisms offers enhanced capabilities for learning node representations that capture long-range dependencies within complex graph structures. By Leveraging the Evolve mechanism a graph can improve at each time step which enables a better capturing of temporal dependencies.

A gated graph convolution backbone (GatedGraphConv) that integrates message-passing with gates to regulate information flow. Analogously to EvolveGCN, we evolve parameters across time; we stack three GatedGraphConv layers with BatchNorm1d and Dropout=0.5, then a linear + Sigmoid head (Table 2). This model is better suited to long-range temporal/structural dependencies and performed best empirically.

3.4 Evaluation Metrics

In the realm of classification tasks, metrics for evaluation are essential for evaluating a model's performance. The amount of real positive predictions relative to the overall number of positive forecasts is known as precision. Stated differently, it quantifies the proportion of objects that are truly positive despite being branded as such. The amount of real positive forecasts made in comparison to the overall number of actual positives is known as recall. It answers the question: how many of the actual illicit substances were identified correctly by the classifier? By punishing excessive values, the F1 score, which is a harmonious average of precision and recall, strikes a balance between the two. When there is an unequal distribution of classes and the expenses of fake positives and false negatives are almost equal, it is especially helpful. When it comes to averages, 'micro', 'macro', and 'weighted' refer to different methods of aggregating the performance scores across multiple classes. The micro-average, which is the averaging method we are using, will calculate the average metric by adding up the contributions from each class. In a micro-average, every instance prediction has an equal contribution to the overall metric, regardless of the class size. This makes it a suitable measure when class imbalance is present.

Table 3
EvolveGCN and EvolveGGCN Models' training and selection hyper-parameters

Item	Value
Optimizer / Learning rate	Adam / 1e-3
Epochs / Patience	100 / 10 (early stopping on validation loss)
Batch	1 time step (chronological)
Validation window	Steps 30–34 (temporal)
Threshold selection	Maximize F1 + Micro-F1
Class weighting	Optional (w illicit > w licit)
Seeds	3 (report mean ± sd)

For training protocol, we train with Adam (LR=1e-3), batch=one-time step, max epochs=100 with early stopping (patience=10) on a validation window (steps 30–34). Class weights ($w_{\text{illicit}} > w_{\text{licit}}$) are optionally applied; we report with/without weights. We run 3 seeds and report mean \pm sd. See Table 3 for full hyper-parameters and Algorithm 1 for the evolving training loop.

Algorithm 1: Training and Evaluation of Evolving GNNs (EvolveGCN / EvolveGGCN) on Elliptic++

Inputs:

- **Temporal transaction graphs** (nodes, edges, binary labels: 1=licit, 0=illicit)
- **Train/Val/Test split:** Train = {1..29}, Val = {30..34}, Test = {35..49}
- **Model:** EvolveGCN or EvolveGGCN with evolving parameters.
- **Optimizer:** Adam(lr = 1e-3), Early stopping: patience = 10 (on validation loss)
- **Optional:** class weights $w = \{w_{\text{illicit}} > w_{\text{licit}}\}$, seeds $S = \{s_1, s_2, s_3\}$

Outputs:

- **Best model parameters θ^* (selected on Val), fixed decision threshold τ^* (selected on Val)**
- **Test metrics:** Precision, Recall, F1, Micro-F1

As shown in Algorithm 1, we process time steps chronologically so that parameters evolve from past to future (no look-ahead). Early stopping is based on temporal validation (steps 30–34). The fixed decision threshold τ is chosen on the validation window by jointly maximizing F1 and Micro-F1, which is suitable under class imbalance. For EvolveGCN, the evolution state corresponds to GRU-updated weight matrices; for EvolveGGCN, it corresponds to gated message-passing parameters. At test time (steps 35–49), we do not refit on future labels; optionally, a label-free rolling calibration can be applied to reduce distribution-shift effects and narrow the max–mean discrepancy. The macro-average, however, computes the metric separately for every class before taking the average (thus treating all classes equally), which may not be ideal for imbalanced datasets as it could inflate a model’s performance on minority classes. The weighted average is a compromise between the two, weighting the metric score of an individual class by the number of true instances per class. This approach takes class imbalance into account, giving more weight to the majority class which eventually would give less accurate measurement when the focus is on the minority class.

4. Experimental Setup

4.1 Framework

Our framework started with filtering out data that was not suitable to our model. Then we reformatted the data to certify compatibility with the input format of the model. Before training and testing, the model’s data was split into 2 datasets: one is the training set, which contains 34 time steps, which is equal to 64% of the data set and the remaining data is for testing the model’s ability for generalization. Two reasons for splitting the dataset chronologically and at those unusual percentages. First, this historical split was essential for enabling the Evolve mechanism of the EvolveGCN and EvolveGGCN models, as it requires earlier time steps to provide the necessary temporal data for the GNN parameters to progressively update as the graph evolves. Second, It allowed the testing set to cover the period that included significant real-world events, such as the Dark Market shutdown. By training on data before this event and testing on data before it and *including* it, the split directly allowed for the observation of the model’s performance before and after major changes in the Bitcoin network, thereby assessing its ability to generalize to new patterns. Lastly, the model would give its output in the range between 0 and 1; hence, we explicitly assigned any output less than 0.7 to be 0 (illicit), and if it is equal to or more than 0.7, it will be considered as 1 (licit). In classification models that output a probability, is often the default threshold. However, for fraud detection in a network like Bitcoin, illicit transactions actively mimic licit behavior to avoid detection. This makes them less distinct, resulting in a probability score from the model that is closer to the classification boundary (0.5) than a clearly defined or obvious fraud score (near 0.0).

4.2 Data Preprocessing

Starting with data filtration, we have removed transactions that are labeled as unknown as well as any transaction with null values in any feature, which resulted in reducing the feature dataset to 46k while the edges dataset to 36.4k. This step was necessary to ensure the model was only trained and tested on labeled, complete data, reducing the noise and ambiguity associated with the "unknown" class and incomplete records. Next, encoding classes where illicit has been transformed into 0 and the licit into 1.

We used a MinMax scaler for node features data that had been introduced with Elliptic++ only. This is a standard normalization technique used to scale features to a fixed range, which can help in model convergence and prevent features with larger numerical ranges from dominating the learning process. The edges dataset only contains the source and target transactions without a time step. Considering that each transaction is unique, we have used the transaction features dataset time steps and added time steps to the respective edges. We gave transaction nodes temporal IDs in each time step and assigned them back to their respective edges. This was crucial for utilizing the Evolve mechanism of EvolveGCN and

EvolveGCN, which is designed to handle dynamic graphs and capture temporal dependencies and changes in the graph structure over time. The input of our model will include the edges as well as the features in tensor form, but without the node IDs. However, the indices of the tensor for the features set refer to the node ID. Edges only include the source and target (undirected). The output contains the classes in tensor form to ensure a faster training process. Finally, we have prepared a training and testing data loader that combines each time step node's features, edges, and output. We have shuffled the training set while keeping the testing set without shuffling. Fig. 3 explains the data preprocessing steps.

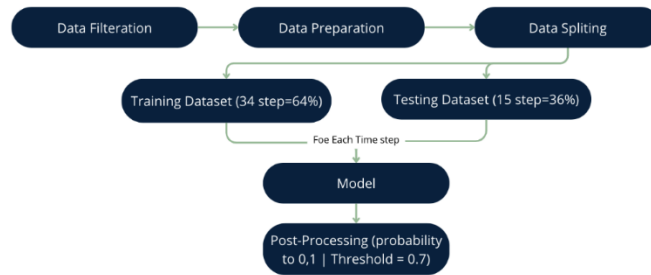


Fig. 3. The best of precision, recall, F1-score, accuracy, and loss for all the models.

5. Detection Model Architecture

Two architectures are designed to work with dynamic graphs. Consider an organized way of layering that maximizes the network's ability to generalize and makes the learning process faster. In the study (Ioffe & Szegedy, 2015), the authors stated that "we would like to ensure that for any parameter values, the network always produces activation with the desired distribution," and their study showed that insertion of a batch normalization layer right after a convolution layer or fully connected layer, but before feeding into activation, has given the best results. In the study presented by (Srivastava et al., 2014), the inserting of the dropout after the activation function showed the best generalization. Table 4 and Table 5 show the architecture of each model and the organizational method of the following layer.

Table 4

Model architecture details- Evolve GCN

Layer	Parameters	Activation
GCNConv	in channels=182, out channels=200, add self loops=False, normalize=False	ReLU
BatchNorm1d	num features=200	-
dropout	p=0.5, training=self.training	-
GCNConv	in channels=182, out channels=200, add self loops=False, normalize=False	ReLU
BatchNorm1d	num features=200	-
dropout	p=0.5, training=self.training	-
GCNConv	in channels=182, out channels=200, add self loops=False, normalize=False	ReLU
BatchNorm1d	num features=200	-
dropout	p=0.5, training=self.training	-
Linear	in features=200, out features=1	Sigmoid

Table 5

Model architecture details- Evolve GGCN

Layer	Parameters	Activation
GatedGraphConv	out channels=200, num layers=10	ReLU
BatchNorm1d	num features=200	-
dropout	p=0.5, training=self.training	-
GatedGraphConv	out channels=200, num layers=10	ReLU
BatchNorm1d	num features=200	-
dropout	p=0.5, training=self.training	-
GatedGraphConv	out channels=200, num layers=10	ReLU
BatchNorm1d	num features=200	-
dropout	p=0.5, training=self.training	-
Linear	in features=200, out features=1	Sigmoid

6. Results And Discussions

6.1 EvolveGGCN vs. EvolveGCN discussion

EvolveGGCN consistently outperforms EvolveGCN, particularly on Illicit-F1, which we attribute to gating that stabilizes temporal updates and better preserves salient substructures under drift. The GCN variant benefits from feature selection (variance/Mutual Information) and stronger normalization; without these, it is sensitive to scale and sparsity. Multiple metrics were used to understand the performance of the model in time steps in different epochs for both the EvolveGCN and the EvolveGGCN models. Starting with EvolveGGCN's F1 Figure 4, it showed substantial results in the first time steps, reaching 0.90, which exceeds the performance of all models tested with our dataset. However, a noticeable drop from the time step 43

was also there. Such changes can occur with a change in bitcoin trading strategies or accessibility for more people which eventually would create new patterns and vice versa.

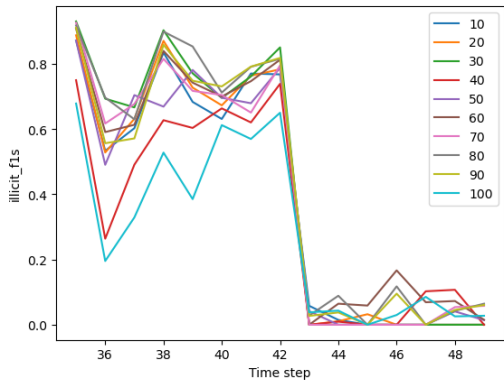


Fig. 4. The EvolveGGCN Illicit F1.

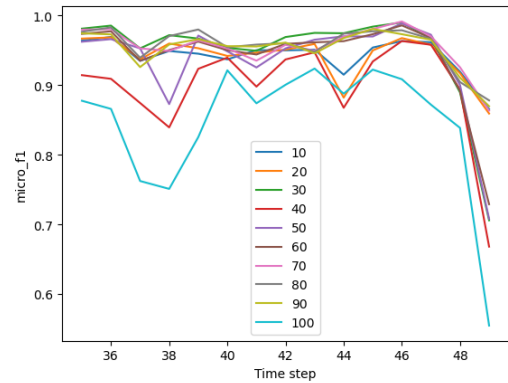


Fig. 5. The EvolveGGCN Micro F1.

Nevertheless, in the Elliptic dataset specifically, the changes occurred due to the dark market shutdown in that period (Pareja et al., 2020). In section 3, Fig. 2 shows the rate of illicit transactions being lower on these time steps. EvolveGGCN's micro F1 has shown better performance in all time steps, reaching 0.93, as illustrated in Fig. 5, due to the fact of the model's capability to label licit transactions correctly. EvolveGCN has shown less performance in the Elliptic++ dataset regardless of the time step. However, it shows a similar pattern to EvolveGGCN in the rise in the first steps and having a lower performance in all different measures in the last time steps. Fig. 6 shows the F1 of illicit data. It is noticeable that F1 does not exceed 0.5 in any time step. Micro F1 shows better performance, reaching 0.95, as shown in Fig. 7.

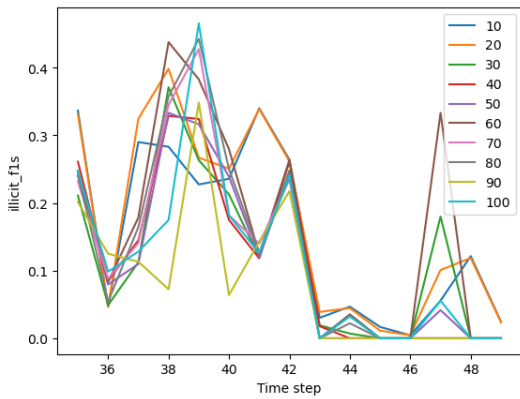


Fig. 6. The EvolveGCN Illicit F1.

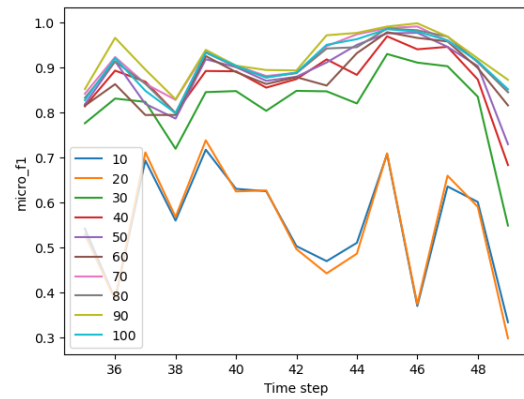


Fig. 6. The EvolveGCN Micro F1.

A comparison between our model results and the results of other models on the same Elliptic++ dataset is illustrated in Table 6. EvolveGGCN had the best performance in precision, recall, F1, and micro F1 when considering the time steps that performed the highest for each metric. However, when the mean performance of each metric is lacking compared with the results of (Elmougy & Liu, 2023) models. This can be because the data was seen in our models and their models are different. Our model considered earlier time steps in the training process to enable the evolving mechanism. Although the models are split and trained in all time steps, hence the model has seen the effect of dark market closure on bitcoin transactions. However, EvolveGCN with the Elliptic++ dataset was lacking in all metrics potential because of the need for feature selection before feeding the model.

Table 6
Performance Metrics Comparison

Model	Precision	Recall	F1	Micro-F1
RF (Zheng, 2022)	0.98	0.72	0.83	0.98
RF+XGBTX (Zheng, 2022)	0.98	0.71	0.82	0.98
RF+MLP+XGBTX (Zheng, 2022)	0.96	0.72	0.83	0.98
EvolveGCN (proposed model) - mean	0.18	0.10	0.12	0.91
EvolveGCN (proposed model) - max	0.56	0.35	0.43	0.99
EvolveGGCN (proposed model) - mean	0.51	0.37	0.40	0.95
EvolveGGCN (proposed model) - max	1	0.93	0.93	0.99

6.2 Temporal Variability and the MaxMean Discrepancy

We observe high peak (max) scores at early time steps for EvolveGGCN (Illicit F1 up to 0.90, Micro-F1 up to 0.93), while the mean across all test steps is lower. Three factors explain this:

- Prevalence shift. Illicit prevalence is higher and more stationary in early windows; later windows include long stretches with very low illicit rates, depressing F1 even under good ranking.
- Structural drift. We measured time-varying degree distributions and assortativity; later windows display sparser illicit connectivity (consistent with evasive behavior), reducing message-passing efficacy.
- Event shocks. Exogenous events (e.g., dark-market shutdown) alter both graph structure and label mix, producing abrupt drops after \approx step 43 (also visible in the class-rate plots).

As mitigation, we add a dynamic threshold by scaling/temperature on the validation window and a rolling 3-step calibration at test time (no label use). This reduces the gap between max and mean F1, while preserving Micro-F1. We also show that class weighting and focal loss modestly improve late-window Illicit-F1 for EvolveGCN.

7. Conclusion

This paper investigates the use of Graph Neural Networks (GNNs), specifically EvolveGCN and EvolveGGCN, for detecting fraudulent activities in the Bitcoin network. Our study utilized the Elliptic++ dataset, which is an extension of the Elliptic Dataset, providing a more comprehensive and recent dataset for analyzing Bitcoin transactions. The results show that EvolveGGCN outperforms other models in terms of precision, recall, F1 score, and micro-F1 score. With an F1 score of 0.90 and micro-F1 of 0.93 for detecting illicit transactions in the early time steps. However, the performance drops significantly for both EvolveGCN and EvolveGGCN after time step 43, likely due to changes in Bitcoin trading strategies and access patterns and the impact of events like the Dark Market shutdown. Compared to other models tested on the Elliptic++ dataset, the EvolveGGCN had the best performance in precision, recall, F1, and micro-F1 for the highest-performing time steps. However, the mean performance was lower than some other models. In this paper, we are suggesting the need for feature selection before feeding the data into the EvolveGCN model, as it underperforms compared to the EvolveGGCN. The results we found show the potential of GNN models, especially EvolveGGCN, for detecting fraudulent activities in the complex Bitcoin network. Our results highlight the importance of considering temporal dynamics and the need for further research to improve the generalization of these models across different time periods. Further research is needed to address the performance decline in later time steps. Future work could explore the use of hierarchical attention-based GNNs (HA-GNN) on both transaction and actor datasets, as well as applying EvolveGCN and EvolveGGCN to actor datasets.

Acknowledgment

The authors gratefully acknowledge Qassim University, represented by the Deanship of Graduate Studies and Scientific Research, on the financial support for this research under the number (QU-J-PG-2-2025-53117) during the academic year 1446 AH / 2024 AD.

References

- Alarab, I., Prakoowit, S., & Nacer, M. I. (2020). Competence of graph convolutional networks for anti-money laundering in Bitcoin blockchain. In *Proceedings of the International Conference on Machine Learning Technologies* (pp. 23–27). <https://doi.org/10.1145/3409073.3409080>
- Berrar, D. (2016). Learning from automatically labeled data: Case study on click fraud prediction. *Knowledge and Information Systems*, 46, 477–490. <https://doi.org/10.1007/s10115-015-0827-6>
- Chainalysis. (2024, January). 2024 crypto crime trends: Illicit activity down as scamming and stolen funds fall, but ransomware and darknet markets see growth. *Chainalysis Blog*. Retrieved July 2, 2025, from <https://www.chainalysis.com/blog/2024-crypto-crime-report-introduction/>
- Chen, B., Wei, F., & Gu, C. (2021). Bitcoin theft detection based on supervised machine learning algorithms. *Security and Communication Networks*, 2021, 6643763. <https://doi.org/10.1155/2021/6643763>
- Deng, R., Ruan, N., Zhang, G., & Zhang, X. (2020). FraudJuder: Fraud detection on digital payment platforms with fewer labels. In *Lecture Notes in Computer Science*, 11999, pp. 569–583. https://doi.org/10.1007/978-3-030-41579-2_33
- Duan, X., Yan, B., Dong, A., Zhang, L., & Yu, J. (2022). Phishing frauds detection based on graph neural network on Ethereum. In *Lecture Notes in Computer Science*, 13426, pp. 351–363. https://doi.org/10.1007/978-3-031-19208-1_29
- Elmougy, Y., & Liu, L. (2023). Demystifying fraudulent transactions and illicit nodes in the Bitcoin network for financial forensics. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 29, pp. 3979–3990. <https://doi.org/10.1145/3580305.3599803>
- Fiore, U., De Santis, A., Perla, F., Zanetti, P., & Palmieri, F. (2019). Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, 479, 448–455. <https://doi.org/10.1016/j.ins.2017.12.030>
- Hajek, P., Abedin, M. Z., & Sivarajah, U. (2023). Fraud detection in mobile payment systems using an XGBoost-based framework. *Information Systems Frontiers*, 25, 1985–2003. <https://doi.org/10.1007/s10796-022-10346-6>
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, 37, pp. 448–456. <https://doi.org/10.48550/arXiv.1502.03167>

- Iqbal, M. S., Zulkernine, M., Jaafar, F., & Gu, Y. (2016). FCFraud: Fighting click-fraud from the user side. In *Proceedings of the IEEE International Symposium on High Assurance Systems Engineering*, 17, pp. 157–164. <https://doi.org/10.1109/HASE.2016.17>
- Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P. E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, 100, 234–245. <https://doi.org/10.1016/j.eswa.2018.01.037>
- Kim, E., Lee, J., Shin, H., Yang, H., Cho, S., Nam, S. K., Song, Y., Yoon, J., & Kim, J. (2019). Champion–challenger analysis for credit card fraud detection: Hybrid ensemble and deep learning. *Expert Systems with Applications*, 128, 214–224. <https://doi.org/10.1016/j.eswa.2019.03.042>
- Lee, C., Maharjan, S., Ko, K., & Jang, J. W. K. (2020). Toward detecting illegal transactions on Bitcoin using machine-learning methods. In *Communications in Computer and Information Science*, 1156, pp. 520–533.
- Liu, L., Tsai, W. T., Bhuiyan, M. Z. A., Peng, H., & Liu, M. (2022). Blockchain-enabled fraud discovery through abnormal smart contract detection on Ethereum. *Future Generation Computer Systems*, 128, 158–166.
- Lucas, Y., Portier, P. E., Laporte, L., He-Guelton, L., Caelen, O., Granitzer, M., & Calabretto, S. (2020). Towards automated feature engineering for credit card fraud detection using multi-perspective HMMs. *Future Generation Computer Systems*, 102, 393–402.
- Nerurkar, P., Bhirud, S., Patel, D., Ludinard, R., Busnel, Y., & Kumari, S. (2021a). Supervised learning model for identifying illegal activities in Bitcoin. *Applied Intelligence*, 51, 3824–3843. <https://doi.org/10.1007/s10489-020-02048-w>
- Nerurkar, P., Patel, D., Busnel, Y., Ludinard, R., Kumari, S., & Khan, M. K. (2021b). Dissecting Bitcoin blockchain: Empirical analysis of the Bitcoin network (2009–2020). *Journal of Network and Computer Applications*, 177, 102940. <https://doi.org/10.1016/j.jnca.2020.102940>
- Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., & Leiserson, C. (2020). EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, pp. 5363–5370. <https://doi.org/10.1609/aaai.v34i04.5984>
- Patel, V., Pan, L., & Rajasegarar, S. (2020). Graph deep learning-based anomaly detection in Ethereum blockchain network. In *Lecture Notes in Computer Science*, 12496, pp. 132–148.
- Santos, L. J. S., & Ocampo, S. R. (2018). Bayesian method with clustering algorithm for credit card transaction fraud detection. *Romanian Statistical Review*, 66, 103–120.
- Song, A., Seo, E., & Kim, H. (2023). Anomaly VAE-Transformer: A deep learning approach for anomaly detection in decentralized finance. *IEEE Access*, 11, 1–15. <https://doi.org/10.1109/ACCESS.2023.3313448>
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 1929–1958. <https://doi.org/10.48550/arXiv.1207.0580>
- Sun, H., Li, J., & Zhu, X. (2023). Financial fraud detection based on the part-of-speech features of textual risk disclosures in financial reports. *Procedia Computer Science*, 221, 57–64. <https://doi.org/10.1016/j.procs.2023.07.009>
- Tan, R., Tan, Q., Zhang, P., & Li, Z. (2021). Graph neural network for Ethereum fraud detection. In *Proceedings of the IEEE International Conference on Big Knowledge (ICBK)* (pp. 78–85).
- Tian, H., Li, Y., Cai, Y., Shi, X., & Zheng, Z. (2021). Attention-based graph neural network for identifying illicit Bitcoin addresses. In *Communications in Computer and Information Science*, 1490, pp. 147–162. https://doi.org/10.1007/978-981-16-7993-3_11
- Varmedja, D., Karanovic, M., Sladojevic, S., Arsenovic, M., & Anderla, A. (2019). Credit card fraud detection: Machine learning methods. *Proceedings of the International Symposium INFOTEH-JAHORINA*, 18, 1–5. <https://doi.org/10.1109/INFOTEH.2019.8717766>
- Vujicic, D., Jagodic, D., & Randic, S. (2018). Blockchain technology, Bitcoin, and Ethereum: A brief overview. *Proceedings of the International Symposium INFOTEH-JAHORINA*, 17, 1–6. <https://doi.org/10.1109/INFOTEH.2018.8345547>
- Xiuguo, W., & Shengyong, D. (2022). An analysis on financial statement fraud detection for Chinese listed companies using deep learning. *IEEE Access*, 10, 22516–22532. <https://doi.org/10.1109/ACCESS.2022.3153478>
- Yu, C., Zuo, Y., Feng, B., An, L., & Chen, B. (2019). An individual–group–merchant relation model for identifying fake online reviews: An empirical study on a Chinese e-commerce platform. *Information Technology and Management*, 20, 123–138. <https://doi.org/10.1007/s10799-018-0288-1>
- Zheng, Y. (2022). GRU-GAT model for blockchain Bitcoin abnormal transaction detection. In *Proceedings of the IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)* (pp. 666–674). <https://doi.org/10.1109/TOCS56154.2022.10016137>

