Neural Processing Letters (2023) 55:689–707
https://doi.org/10.1007/s11063-022-10904-8

Graph-Based LSTM for Anti-money Laundering: Experimenting Temporal Graph Convolutional Network with Bitcoin Data

Ismail Alarab[1] · **Simant Prakoonwit**[1]

Accepted: 25 May 2022 / Published online: 16 June 2022

Abstract
Elliptic data—one of the largest Bitcoin transaction graphs—has admitted promising results in many studies using classical supervised learning and graph convolutional network models for anti-money laundering. Despite the promising results provided by these studies, only few have considered the temporal information of this dataset, wherein the results were not very satisfactory. Moreover, there is very sparse existing literature that applies active learning to this type of blockchain dataset. In this paper, we develop a classification model that combines long-short-term memory with GCN—referred to as temporal-GCN—that classifies the illicit transactions of Elliptic data using its transaction's features only. Subsequently, we present an active learning framework applied to the large-scale Bitcoin transaction graph dataset, unlike previous studies on this dataset. Uncertainties for active learning are obtained using Monte-Carlo dropout (MC-dropout) and Monte-Carlo based adversarial attack (MC-AA) which are Bayesian approximations. Active learning frameworks with these methods are compared using various acquisition functions that appeared in the literature. To the best of our knowledge, MC-AA method is the first time to be examined in the context of active learning. Our main finding is that temporal-GCN model has attained significant success in comparison to the previous studies with the same experimental settings on the same dataset. Moreover, we evaluate the performance of the provided acquisition functions using MC-AA and MC-dropout and compare the result against the baseline random sampling model.

Keywords Temporal GCN · Uncertainty estimation · Active learning · Bitcoin data · Anti-money laundering

✉ Ismail Alarab
ialarab@bournemouth.ac.uk

Simant Prakoonwit
sprakoonwit@bournemouth.ac.uk

[1] Bournemouth, UK

1 Introduction

Blockchain intelligence and forensics company CipherTrace have reported a global amount of $US 4.5 billion in 2019 of Bitcoin crime related to illicit services [1]. Money launderers exploit the pseudonym of Bitcoin ledgers by transforming the illegally obtained money from serious crimes into legitimate funds via Bitcoin network. On the other hand, Bitcoin blockchain has attracted intelligence companies and financial regulators who transact on the blockchain to be aware of its risks, such as technical developments in and societal adoption of the cryptocurrency Bitcoin [2]. The arising of illegal services and the public availability of Bitcoin data have urged the need to develop intelligent methods that exploit the transparency of the blockchain records [3]. Such methods can boost anti-money laundering (AML) in Bitcoin and enhance safeguarding cryptocurrency ecosystems. In the past few years, Elliptic company—a cryptocurrency intelligence company focusing on safeguarding cryptocurrency systems—has released a graph network of Bitcoin transactions, known as Elliptic data. This data has been a great support to the research and AML community in order to develop machine learning methods. Elliptic data acquires a graph of Bitcoin transactions that spans handcrafted local features (associated with transactions itself) and aggregated features (associated with neighbouring transactions) with partially labelled nodes. Furthermore, the labelled nodes denote licitly-transacted payments (e.g. miners) and illicit transactions (e.g. theft, scams). Previous researchers have attempted to apply this dataset to classical supervised learning methods [3, 4], graph convolutional network (GCN) [3, 5], EvolveGCN for dynamic graphs [6], signature vectors in blockchain transactions (SigTran) model [7] and uncertainty estimation with multi-layer perceptron (MLP) model [8]. Despite the promising results achieved by these studies, the highest accuracy achieved considering only the set of local features is about 97.4% and f_1-score of 77.3%. Furthermore, only a few have considered the temporal information of this dataset. On the other hand, Bitcoin blockchain is a large-scale data in which the labelling process of this data is very hard and time-consuming. Active learning (AL) approach tackles this problem by querying the labels of the most informative data points that attain high performance with less labelled examples. Using Elliptic data, the only work by Lorenz et al. [9] has presented an active learning solution which has shown the capability of matching the performance of a fully supervised model by using 5% of the labelled data. However, the preceded framework has been presented with classical supervised learning methods which do not consider the graph topology or temporal sequence of Elliptic data. In this paper, we aim:

- To present a model in a novel way that considers the graph structure and temporal sequence of Elliptic data to predict illicit transactions that belong to illegal services in Bitcoin blockchain network.
- To perform active learning on Bitcoin blockchain data that mimics a real-life situation, since Bitcoin blockchain is a massively growing technology and its data is time-consuming to label.

The presented classification model comprises long short-term memory (LSTM) and GCN models, wherein the overall model attains an accuracy of 97.7% and f_1-score of 80% which outperform previous studies with the same experimental settings. On the other hand, the presented active learning framework requires an acquisition function that relies on model's uncertainty to query the most informative data. In this paper, the model's uncertainty estimates are obtained using two comparable methods based Bayesian approximations which are named Monte-Carlo dropout (MC-dropout) [10] and Monte-Carlo adversarial attack

(MC-AA) [11]. We examine these two uncertainty methods due to their simplicity and efficiency where MC-AA method is the first time to be applied in the context of active learning. Hence, we use a variety of acquisition functions to test the performance of the active learning framework using Elliptic data. For each acquisition function, we evaluate the active learning performance that relies on each of MC-AA and MC-dropout uncertainty estimates. We compare the performance of the presented active learning framework against the random sampling acquisition as a baseline model.

This paper is structured as follows: Section 2 describes the related work. Section 3 demonstrates the uncertainty estimation methods used by the active learning framework. Section 4 provides various acquisition functions to be examined in the experiments. Section 5 provides the methods used to perform the classification task. Experiments are detailed in Sect. 6 followed by the results and discussions in Sect. 7. An ablation study of the proposed model is given in Sect. 8. Section 9 states the conclusion to wrap up the whole methodology.

2 Overview of Related Work

With the appearance of illicit services in the public blockchain systems, intelligent methods have undoubtedly become a necessary need for AML regulations with the rapidly increasing amount of blockchain data. Many studies have adopted the machine learning approach in detecting illicit activities in the public blockchain. Harlev et al. [2] have tested the performance of classical supervised learning methods to predict the type of the unidentified entity in Bitcoin. Farrugia et al. [12] have applied XGBoost classifier to detect fraudulent accounts using the Ethereum dataset. Weber et al. [3] have introduced Elliptic data—a large-scale graph-structured dataset of a Bitcoin transaction graph with partially labelled nodes—to predict licit and illicit Bitcoin transactions. This dataset has been introduced by Weber et al. [3] who have discussed the outperformance of the random forest model against graph convolutional network (GCN) in classifying the licit and illicit transactions derived from the Bitcoin blockchain. Subsequently, the classification results using ensemble learning model in [4] have revealed a significant success over other benchmark methods to classify illicit transactions of Elliptic data. Also, Pareja et al. [6] have introduced EvolveGCN which is formed of GCN with a recurrent neural network such as Gated-Recurrent-Unit (GRU) and LSTM. This study has revealed the outperformance of EvolveGCN over the GCN model used by Weber et al. [3] on the same dataset. Another work in [5] has considered the neighbouring information of the Bitcoin transaction graph of Elliptic data using GCN accompanied by linear hidden layers. Without utilising any temporal information from this dataset, the latter reference has achieved an accuracy of 97.4% outperforming the GCN based models that were presented in [3, 6].

Active learning, a subfield of machine learning, is a way to make the learning algorithm choose the data to be trained on [13]. Active learning mitigates the bottleneck of the manual labelling process, such that the learning model queries the labels of the most informative data. Since it is so expensive to obtain labels, active learning has witnessed a resurgence with the appearance of big data where large-scale datasets exist [14]. Lorenz et al. [9] have presented an active learning framework in an attempt to reduce the labelling process of the large-scale Elliptic data of Bitcoin. The presented active learning solution has shown its capability in matching the performance of a fully supervised model with only 5% of the labels. The authors have focused on querying strategies based on uncertainty sampling [13, 15] and expected model change [13, 16]. For instance, the used uncertainty sampling strategy is based on

the predicted probabilities provided by the random forest in [9]. Yet, no study presents an active learning framework that utilises the recent advances in Bayesian methods on Bitcoin data. On the other hand, Gal et al. [17] have presented active learning frameworks on image data where the authors have combined the recent advances in Bayesian methods into the active learning framework. This study has performed MC-dropout to produce the model's uncertainty which is utilised by a given acquisition function to choose the most informative queries for labelling. Concisely, the authors in [18] have applied the entropy [19], mutual information [20], variation ratios [21], and mean standard deviation (Mean STD) [22, 23] acquisition functions which are compared against the random acquisition.

In this study, we conduct experiments using a classification model that exploits the graph structure and the temporal sequence of Elliptic data derived from the Bitcoin blockchain. Motivated by the studies in [9, 17], we perform the active learning frameworks, using pool based-based scenario [13] in which the classifier iteratively samples the most informative instances for labelling from an initially unlabelled pool. For each iteration, the classifier samples a batch of unlabelled data points according to their uncertainty estimates from Bayesian models using the sampling acquisition function.

3 Model Uncertainty: MC-Dropout Versus MC-AA

The two major types of uncertainty in a machine learning model are epistemic and aleatoric uncertainties [24]. Epistemic, also known as model uncertainty [10], is induced from the uncertainty in the parameters of the trained model. This uncertainty is reducible by training the model on enough data. Aleatoric uncertainty is the uncertainty tied with the noisy instances that lie on the decision boundary or in the overlapping region for class distributions, and therefore it is irreducible. MC-dropout has gained popularity as a prominent method in producing the two types of uncertainties [10]. Although MC-dropout is easy to perform and efficient, this method has failed, to some extent, to capture data points lying in the overlapping region of different classes where noisy instances reside [11]. The latter reference has provided an uncertainty method that is capable to reach noisy instances with high uncertainty estimates. This method is so-called MC-AA which targets mainly the instances that fall in the neighbourhood of a decision boundary. Although MC-dropout and MC-AA are both simple and promising methods, MC-AA has provided more reliable uncertainty estimates in [11]. In the light of these studies, we utilise these uncertainty methods as a part of the active learning process.

3.1 MC-Dropout: Monte-Carlo Dropout

Initially, dropout has been provided as a simple regularisation technique that reduces the overfitting of the model [25]. The work in [10] has MC-dropout as a probabilistic approach based on Bayesian approximation to produce uncertainty estimates. MC-dropout uses dropout after every weight layer in a neural network. Uncertainty estimates are produced by activating dropout during the testing phase by multiple stochastic forward passes wherein uncertainty measurement (e.g., mutual information) is computed.

Let $\hat{y}$ be an output of an input x mapped by a neural network, trained on set $\mathcal{D}_{train}$, with layers L and learnable weights $w = \{W_i\}_{i=1}^{L}$. Consider y as an observed output associated with x. Then, we can express the predictive distribution as:

$$p(y|x, \mathcal{D}_{train}) = \int p(y|x, w)p(w|\mathcal{D}_{train})dw, \tag{1}$$

where $p(y|x, w)$ is the model's likelihood and $p(w|\mathcal{D}_{train})$ is the posterior over the weights. Since the posterior distribution is analytically intractable [10], the posterior is replaced by $q(w)$, an approximation of variational distribution. $q(w)$ is obtained from the minimisation of Kullback–Leibler divergence (KL) to approximately match $p(w|\mathcal{D}_{train})$ as follows referring to:

$$KL(q(w)|p(w|\mathcal{D}_{train})$$

Hence, the variational inference leads to an approximated predictive distribution as:

$$q(y|x) = \int p(y|x, w)q(w)dw \tag{2}$$

The work in [10] has chosen $q(w)$ to be the distribution over the matrices whose columns are randomly set to zero for posterior approximation. Then, $q(w)$ can be defined as:

$$W_i = M_i \cdot diag\left(\left[z_{i,j}\right]_{j=1}^{K_i}\right) \tag{3}$$

where $z_{i,j} \sim$ Bernoulli(p_i), as realisation from Bernoulli distribution, for $i = 1, \ldots, L$ and $j = 1, \ldots K_{i-1}$, with K_ixK_{i-1} the dimension of matrix W_i.

Thus, drawing T samples from Bernoulli distribution produces $\left\{W_1^t, ..., W_L^t\right\}_{t=1}^T$. These are obtained from T stochastic forward passes with active dropout during the testing phase of the input data. Then, the predictive mean can be expressed as:

$$E_{q(y|x)(y)} \approx \frac{1}{T}\sum_{t=1}^{T} \hat{y}\left(x, W_1^t, \ldots, W_L^t\right) = p_{MC}(y|x) \tag{4}$$

To obtain uncertainty, mutual information (MI) identifies the information gain of the outputs derived from Monte-Carlo samples over the predictions. Data points that reside near the decision boundary are more likely to acquire high mutual information referring to [8]. We can express mutual information as follows, referring to [10]:

$$\hat{I}(y|x, \mathcal{D}_{train}) = \hat{H}(y|x, \mathcal{D}_{train}) + \sum_{\mathrm{c}} \frac{1}{\mathrm{T}} \sum_{\mathrm{t}=1}^{\mathrm{T}} \mathrm{p(y = c|x,w)logp(y = c|x, w)} \tag{5}$$

where c is the class label, and

$$\hat{H}(y|x, \mathcal{D}_{train}) = -\sum_{c} p_{MC}(y = c|x, w) \log p_{MC}(y = c|x, w) \tag{6}$$

MC-dropout method can be viewed as an ensemble of multiple decision functions derived from the multiple stochastic forward passes. Precisely, it is an ensemble of multiple perturbed decision boundaries. As this method captures data points between different class distributions, a noisy point that falls in the wrong class cannot be captured by MC-dropout, since the latter method influences only the points with weak confidence. This is tackled in MC-AA method that is stated in the next part.

3.2 MC-AA: Monte-Carlo Based Adversarial Attack

Initially, adversarial attacks are introduced as crafted perturbations of the input in order to produce incorrect predictions [26], which affect the integrity of the model by the attackers. These attacks fall are categorised between white-box and black-box attacks. The former is when the attacker has access to the model's parameters, wherein the latter type accounts for using the model as a black box. White-box attacks are designed by adding perturbations to the inputs in the direction of the decision boundary formed by the model. These guided perturbations are the gradients of the loss function with respect to the input such that the input is assumed to belong to different class distribution. One of the methods used to compute the perturbations is known as FGSM (Fast Gradient Sign Method). Primarily, FGSM is proposed in [27] for attacking deep neural networks. This method is based on maximising a loss function $J\ (x,\ y)$ in a neural network model with respect to a given input x and its label y. The aim of this method is to make the classifier perform poorly on the perturbed inputs as worse as possible. The perturbation of input by FGSM can be reformulated as follows:

$$x_\varepsilon = x + \delta x_\varepsilon, \tag{7}$$

with

$$\delta x_\varepsilon = \varepsilon \cdot sign(\nabla_x J(x, y)),$$

where x_ε is the adversarial example, ε is a small number and ∇_x is the gradient with respect to the input x. This method perturbs the given input in the opposite direction of the initial class towards the decision boundary. MC-AA is based on the idea of FGSM by computing multiple perturbed versions of an input in a small range [11]. This leads to multiple outputs that allow obtaining uncertainty. MC-AA can be viewed as ensemble learning of multiple decisions derived from the perturbed versions of an input in a back-and-forth manner in the direction of the decision boundary. Thus, any point falling on the decision boundary will reflect a high uncertainty. In MC-AA, the noisy labels are triggered to move in a small range, so that they are more likely to escape from their wrong class. Thus, the noisy labels will be assigned with some uncertainty. Moreover, this will further increase the number of correctly classified data points to be uncertain, which does not affect the model performance. More formally, consider a discrete interval I that is evenly spaced by β and symmetric at zero, then it can be expressed as follows:

$$I = \left\{ \varepsilon_i | \left(\varepsilon_{i+1} - \varepsilon_i = \beta \right) \wedge \left(\beta = \frac{2\varepsilon_T}{T} \right) \right\}_{t=1}^{T} \tag{8}$$

where $\varepsilon_T = \varepsilon_{max}$ that is the maximum value in the interval I as a tunable hyper-parameter to perturb an input by FGSM. T is a pre-chosen interval size, and it is also the number of ensembles to be performed via MC-AA. Consider a neural network of weights w with function approximation as $f : x \rightarrow \hat{y}$. Let y be the associated observation of x. Since the perturbations by MC-AA over x are applied on a very small range, we can use Taylor expansion up to order 1 to make approximations as follows:

$$f(x_\varepsilon) = f(x + \delta x_\varepsilon) \approx (x) + \frac{f'(x)\delta x_\varepsilon}{1!}$$

To compute the predictive mean $p_{MC-AA}(y|x)$, we find the average of the predictions of a given input as follows:

$$p_{MCAA}(y|x) \approx \frac{1}{T}\sum_{t=1}^{T} f(x_i) \approx \frac{1}{T}\sum_{t=1}^{T} f(x) + \frac{f'(x)\delta x_\varepsilon}{1!} \tag{9}$$

This equation boils down to:

$$p_{MC-AA}(y|x) \approx f(x) = \hat{y} \tag{10}$$

Hence, we obtain an unbiased predictive mean by MC-AA, whereas several perturbations can be used to compute mutual information as the predictive uncertainty. Similarly, to Eq. 5, we estimate uncertainty of x using mutual information as follows:

$$\hat{I}(y|x, \mathcal{D}_{train}) = \hat{H}(y|x, \mathcal{D}_{train}) + \sum_{c} \frac{1}{T}\sum_{t=1}^{T} p(y = c|x_\varepsilon) \log p(y = c|x_\varepsilon), \tag{11}$$

where c is the class label, and

$$\hat{H}(y|x, \mathcal{D}_{train}) = -\sum_{c} p_{MC-AA}(y = c|x) \log p_{MC-AA}(y = c|x). \tag{12}$$

4 Acquisition Functions for Active Learning

Pool-based active learning is a prominent scenario [13, 28] that assumes a set of labelled data available for initial training $\mathcal{D}_{train}$ and a set of unlabelled pool $\mathcal{D}_{pool}$ in a Bayesian model M with model parameters $w \sim p(w|\mathcal{D}_{train})$ and output predictions $p(y|w, \mathcal{D}_{train})$ for $y \in \{0, 1\}$ in binary classification tasks. Then, the Bayesian model M that is already trained on $\mathcal{D}_{train}$ queries the labels—from the unlabelled set $\mathcal{D}_{pool}$—of an informative batch with size b by an oracle in order to obtain an acceptable performance with less training data. Consider an acquisition function $a(x, \mathrm{M})$ that measures the score of a batch of unlabelled data $\{x_i\}_{i=1}^{b} \in \mathcal{D}_{pool}$. Let $\{x^*\}_{i=1}^{b}$ be the informative batch by the acquisition function which can be expressed as follows [29]:

$$\{x^*\}_{i=1}^{b} = \mathrm{argmax}_{\{\mathrm{x_i}\}_{\mathrm{i=1}}^{\mathrm{b}} \subseteq \mathcal{D}_{pool}} a(\{x_i\}, p(w|\mathcal{D}_{train})) \tag{13}$$

In what follows, we demonstrate various acquisition functions which are detailed in [24].

4.1 BALD: Bayesian Active Learning by Disagreement

Bayesian Active Learning by Disagreement (BALD) [20] is an acquisition method that utilises the uncertainty estimates via mutual information between the model predictions and model parameters. Hence, the learning algorithm queries the data points with the highest mutual information measurements. The highest mutual information measurements are produced when the predictions by Monte-Carlo samples are assigned with the highest probabilities where the samples are associated with different classes.

In this paper, we desire to acquire a batch of size b at each sampling iteration.

Using BALD, this can be expressed as:

$$a\Big(\{x_i\}_{i=1}^{b}, p(w|\mathcal{D}_{train})\Big) \approx \sum_{i=1}^{b} \hat{I}(y_i, w|x_i, \mathcal{D}_{train}),$$

where $\hat{I}$ is derived from Eq. 5 for MC-dropout and Eq. 11 for MC-AA. Furthermore, the optimal batch is the one with b-highest scoring data points to reduce the bottleneck of acquiring a single data point at each acquisition step.

4.2 Entropy

This acquisition method computes the entropy using the uncertainty estimates from Eqs. 6 and 12. During the active learning process, we choose the batch size with the maximum predictive entropy [19] which can be written as:

$$a_{Entropy}\Big(\{x_i\}_{i=1}^{b}, p(w|\mathcal{D}_{train})\Big) \approx \sum_{i=1}^{b} \hat{H}(y_i; w|x_i, \mathcal{D}_{train})$$

The maximum entropy explains the lack of confidence within the obtained predictions which are typically near 0.5.

4.3 Variation Ratios

Similarly, we choose the batch with the maximum variation ratios [21] where the variation ratio is expressed as:

$$\text{variation} - \text{ratio[x]} = 1 - \max_{\text{y}} p(y|x, \mathcal{D}_{train})$$

The maximum variation ratios correspond to the lack of confidence in the samples' predictions.

4.4 Mean Standard Deviation

Likewise, we sample a batch that maximise the mean standard deviation (Mean STD) [22, 23]. The predictive standard deviation can be computed as:

$$\sigma_c = \sqrt{E\big[p(y = c|x, w)^2\big] - E[p(y = c|x, w)]^2},$$

where E corresponds to the expected mean. The σ_c measurement computes the standard deviation between the predictions obtained by Monte-Carlo samples on every data point. Consequently, the mean standard deviation is averaged over all c classes which can be derived from:

$$\sigma = \frac{1}{C} \sum_{c} \sigma_c$$

4.5 Random Sampling: Baseline Model

This acquisition function uniformly draws data points from the unlabelled pool at random.

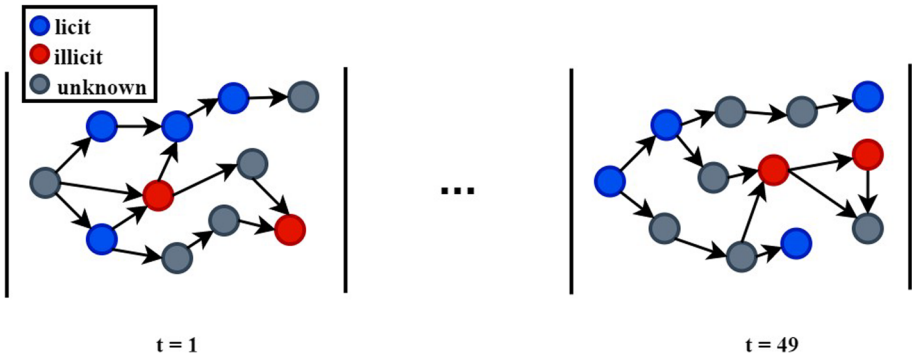

Fig. 1 Representative graph structure of Elliptic data. This dataset incorporates 49 directic acyclic graphs. Each graph is associated with a timestep t. The colours of the nodes denote the labels provided by this dataset

5 Methods

In this section, we provide a detailed description of Elliptic data. Then we demonstrate temporal-GCN which is the proposed classification model to classify the illicit transactions in this dataset.

5.1 Dataset Description

We use the Bitcoin dataset launched by Elliptic company that is renowned for detecting illicit services in cryptocurrencies [3]. This dataset is formed of 49 directed acyclic graphs wherein each is extracted on a specific period of time represented as time-step t, referring to Fig. 1. Each fully connected graph network incorporates nodes as transactions and edges as the flow of payments. In total, this dataset is formed of 203,769 partially labelled transactions, where 21% are labelled as licit (e.g., wallet providers, miners) and 2% are labelled as illicit (e.g. scams, malware, PonziSchemes, …). Each transaction node acquires 166 features such that the first 94 belongs to local features and the remaining as global features. Local features are derived from the transactions' information on each node (e.g. time-step, number of outputs/inputs addresses, number of outputs/inputs unique addresses …). Meanwhile, global features are extracted from the graph network structure between each node and its neighbourhood by using the information of the one-hop backward/forward step for each transaction. In this study, we use the local features which count to 93 features (i.e. excluding time-step) without any graph-related features.

5.2 Temporal Modelling

We refer to the presented model by temporal-GCN. This model is a combination of LSTM and GCN models which are detailed in what follows.

5.2.1 Long Short-Term Memory (LSTM)

Initially, LSTM is proposed by [30] as a special category of recurrent neural networks (RNNs) in order to prevent the vanishing gradient problem. LSTM has proven its efficacy in many general-purpose sequence modelling applications [31–33].

Given a graph network of Bitcoin transactions as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with its adjacency matrix $\mathcal{A} \in \mathbb{R}^{n \times n}$, degree matrix $D \in \mathbb{R}^{n \times n}$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of nodes as Bitcoin transactions and edges as payments flow, respectively, with $|\mathcal{V}| = n$ being the total number of nodes. Consider $x_t \in \mathbb{R}^{d_x}$ as the node feature vector with d_x-dimensional features and layer output $h_t \in [-1, 1]^{d_h}$ as and states $c_t \in \mathbb{R}^{d_h} \in \mathbb{R}dh$ with d_h-dimensional embedding features.

Then, the fully connected LSTM layer, referring to [34], can be expressed as:

$$\begin{aligned} i_t &= \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + w_{ci} \odot c_{t-1} + b_i), \\ f_t &= \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + w_{cf} \odot c_{t-1} + b_f), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc} * x_t + W_{hc} * h_{t-1} + b_c), \\ o_t &= \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + w_{co} \odot c_t + b_o), \\ h_t &= o_t \tanh(c_t), \end{aligned} \tag{14}$$

where $\odot$ is the Hadamard product, $\sigma(.)$ is the sigmoid function and tanh(.) is the hyperbolic tangent function. The remaining notations refer to LSTM layer parameters as follows: i_t, f_t, $o_t \in [0, 1]^{d_h}$ are the input, forget, and output gates, respectively. The weights $W_{x.} \in \mathbb{R}^{d_h, d_x}$, $W_{h.} \in \mathbb{R}^{d_h, d_x}$, $w_{c.} \in \mathbb{R}^{d_h}$ and biases b_i, b_f, b_c, b_o express the parameters of the LSTM model.

5.2.2 Topology Adaptive Graph Convolutional Network: TAGCN

In @@this paper, we use a graph learning algorithm called TAGCN as introduced in [35] which stems from the GCN model. Generally, GCNs are neural networks that are fed with graph-structured data, wherein the node features with a learnable kernel undergo convolutional computation to induce new node embeddings. The kernel can be viewed as a filter of the graph signal (node), wherein the work in [36] suggested the localisation of kernel parameters using Chebyshev polynomials to approximate the graph spectra. Also, the study in [37] has introduced an efficient algorithm for node classification using first-order localised kernel approximations of the graph convolutions.

$$\mathcal{H}^{(l+1)} = \sigma\left(\hat{\mathcal{A}}\mathcal{H}^{(l)} W^{(l)}\right),$$

where $\hat{\mathcal{A}}$ is the normalization of $\mathcal{A}$ defined by:

$$\hat{\mathcal{A}} = \tilde{D}^{-\frac{1}{2}} \tilde{\mathcal{A}} \tilde{D}^{-\frac{1}{2}}, \tilde{\mathcal{A}} = \mathcal{A} + I, \tilde{D} = \text{diag}\left(\sum_j \tilde{A}_{ij}\right),$$

$\tilde{\mathcal{A}}$ is the adjacency matrix of the graph $\mathcal{G}$ with the added self-loops. σ denotes the typical activation function such as $ReLU\ (.) = max(0,.)$. $\mathcal{H}^{(l)}$ is the input node embedding matrix at l^{th} layer. $\mathrm{W}^{(l)}$ is a trainable weight matrix used to update the output embeddings $\mathcal{H}^{(l+1)}$.

On the other hand, the work in [35] has introduced TAGCN which is based on GCN but with fixed-size learnable filters and adaptive to the topology of the graph to perform

convolutions in the vertex domain. Consequently, TAGCN can be expressed as follows:

$$\mathcal{H}^{(l+1)} = \sum_{k=0}^{K} (D^{-\frac{1}{2}} \mathcal{A} D^{-\frac{1}{2}})^k \mathcal{H}^{(l)} \Theta_k, \tag{15}$$

where Θ_k is a learnable weight matrix at k-hop from the node of interest.

5.2.3 Overall Model: Temporal-GCN

Since TAGCN in [35] requires no approximations in comparison to GCN by [37], we exploit the performance of TAGCN in Bitcoin data. Motivated by the work in [38] that has suggested feeding LSTM inputs with GCN node embeddings, temporal-GCN seeks to perform LSTM that learns the temporal sequence in which after is forwarded non-linearly to 2-TAGCN layers to exploit the graph structure of Bitcoin transaction graph. The temporal-GCN model can be expressed as:

$$\begin{aligned} \mathcal{H}^{(1)} &= \text{ReLU}(\text{LSTM}(\mathcal{X})), \\ \mathcal{H}^{(2)} &= \text{ReLU}\Big(\text{TAGCN}\Big(\mathcal{H}^{(1)}, \mathcal{E}\Big)\Big), \\ \mathcal{H}^{(3)} &= \text{Softmax}\Big(\text{TAGCN}\Big(\mathcal{H}^{(2)}, \mathcal{E}\Big)\Big), \end{aligned} \tag{16}$$

where $\mathcal{X}$ is the node feature matrix. LSTM(.) and TAGCN(.,.) are layers mapping a given input to an output from Eqs. 14 and 15, respectively. Softmax function is defined as $\text{Softmax}(x) = \frac{1}{} \exp(x_i)$, where $Z = Z = \sum_i \exp(x_i)$.

6 Experiments

6.1 Experimental Setup

Using the Elliptic data, we split the data following the temporal split as in [3], so that the first 35 graphs (i.e., $t = 1 \rightarrow t = 35$) account for the train set and the remaining are kept for testing. Since this dataset comprises partially labelled nodes, we only use the labelled nodes which add up for 29,894/16,670 transactions in the train/test sets, respectively. To train temporal-GCN, we use Pytorch Geometric (PyG) package [39] which is built on the top of Pytorch (version 1.11.0) enabled-CUDA (version 11.3) in Python programming language. At each time-step t, we feed the relevant graph network with its node feature matrix (i.e., local features excluding timestep) to the temporal-GCN model that is summarised in Eq. 16. LSTM layer uses only the nodes features without any graph-structural information to provide the output matrix $\mathcal{H}^{(1)}$. This matrix is then forwarded to 2-TAGCN layers (in $\mathcal{H}^{(2)}$ and $\mathcal{H}^{(3)}$) that consider the graph-structured data of the top-K influential nodes in the graph, where K is kept by default equal to 3. Hence, a *Softmax* function provides the final class predictions as licit/illicit transactions. We choose *NLLLoss* function and Adam optimiser in order to compute the loss and update the model's parameters. Using the same hyper-parameters from [5], the widths of the hidden layers are set to 50, the number of epochs is set to 50 and the learning rate is fixed at 0.001. Furthermore, we empirically opt 0.7 for the dropout ratio to avoid overfitting. The classification results of the temporal-GCN model are provided in Table 1.

Table 1 Classification results of Elliptic data using local features

Model	% Accuracy	% Precision	% Recall	% F_1 Score
Temporal-GCN	97.7	92.7	71.3	80.6
GCN + MLP [5]	97.4	89.9	67.8	77.3
Evolve-GCN [3]	96.8	85.0	62.4	72.0
Skip-GCN [3]	96.6	81.2	62.3	70.5
Random Forest [3]	96.6	80.3	61.1	69.4
GCN [3]	96.1	81.2	51.2	62.8
MLP [3]	95.8	63.7	66.2	64.9
Logistic regression [3]	92.0	34.8	66.8	45.7

This table shows comparison between the presented model "Temporal-GCN" and previous studies using same features and train/test split

6.2 Active Learning

Active learning has a significant impact to alleviate the bottleneck of labelling especially with this type of data. The main goal of active learning is to use less-training data with achieving acceptable performance. Here, we initially assume the train set as a pool of unlabelled data $\mathcal{D}_{pool}$ and we consider $\mathcal{D}_{train}$ as an empty set to be appended after the querying process. First, the process starts by randomly querying the first batch size for manual labelling, which is arbitrarily assigned to 2000 instances. Afterwards, we append the selected queries to $\mathcal{D}_{train}$ from $\mathcal{D}_{pool}$ to train the temporal-GCN model that is evaluated using the test set at each iteration. Subsequently, the same process is repeated using the uncertainty sampling strategy until we reach an adequate accuracy. However, we query for all instances in $\mathcal{D}_{pool}$. The uncertainty sampling is performed by using one of the acquisition functions demonstrated earlier. These acquisition functions require as input the uncertainty estimates derived by the uncertainty estimation methods. To imitate manual labelling, we append the labels to the queried instances. This experiment is performed using MC-dropout and MC-AA. We compare the performance of the active learning frameworks that use various acquisition functions on the two distinct uncertainty estimation methods. Regarding the hyper-parameters for producing uncertainty estimates, we arbitrarily set $T = 50$ for multiple stochastic forward passes on the unlabelled pool for MC-dropout. With MC-AA, we arbitrarily choose $\varepsilon_T = 0.1$ and $T = 10$.

In addition, we perform random sampling as a baseline which uniformly queries data points at random from the pool. The process of performing active learning with the temporal-GCN model is schematised in Fig. 2. The required time to perform the active learning process in an end-to-end fashion using parallel processing, referring to Fig. 2, is provided in Table 2 using various acquisition functions under the given uncertainty methods.

7 Results and Discussion

We discuss the results of the temporal-GCN model in the light of the previous studies using the same dataset. Subsequently, we provide and discuss the results provided by various active learning frameworks. Then we apply a non-parametric statistical method to discuss

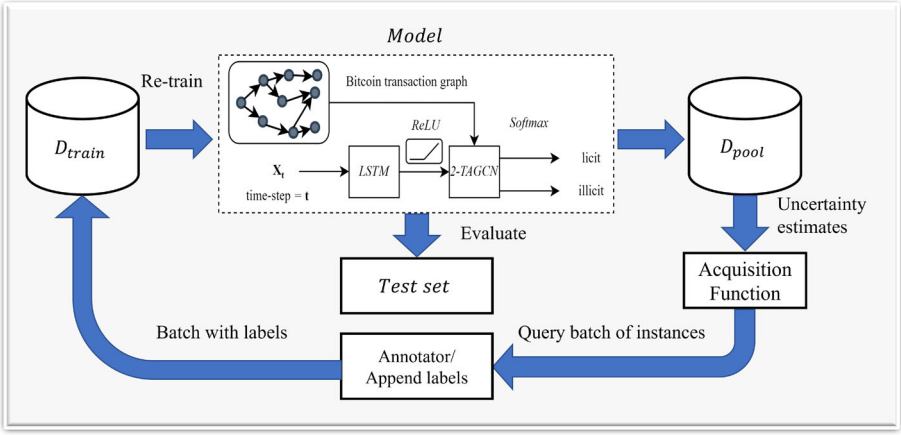

Fig. 2 Schematic representation of the active learning framework using the proposed temporal-GCN model. This frame is a pool-based scenario where annotator queries the data points labels from a pool of unlabelled instances, $\mathcal{D}_{pool}$ using an acquisition function. For each iteration, the queried batch is appended to the train set $\mathcal{D}_{train}$

Table 2 Time required to perform the active learning process in an end-to-end fashion using the proposed temporal-GCN model

Acquisition function	Runtime (mins) using MC-AA	Runtime (mins) using MC-dropout
BALD	$t_{MC-AA} = 28.07$	$tMC - dropout = 27.1$
Entropy	$t_{MC-AA} = 28.9$	$tMC - dropout = 27.3$
Variation ratio	$t_{MC-AA} = 28.9$	$tMC - dropout = 28.3$
Mean STD	$t_{MC-AA} = 28.68$	$tMC - dropout = 27.09$

The time is provided for each experiment that uses the corresponding acquisition function which relies on a given uncertainty estimation method

the significant difference between MC-AA and MC-dropout in performing active learning in comparison to the random sampling strategy.

7.1 Performance of Temporal-GCN

Temporal-GCN has outperformed all previous studies on this dataset that uses local features under the same train/test split settings. The presented model has leveraged temporal sequence and the graph structure of the Bitcoin transaction graph, wherein the classification model can detect illicit transactions with accuracy and f_1-score equal to 97.77% and 80.60%, respectively. In previous studies, Evolve-GCN has attained an accuracy of 96.8%. The latter model has exploited the dynamicity of the graph by performing LSTM on the weights of the GCN layer, which outperformed GCN and skip-GCN without any temporal information. Whereas

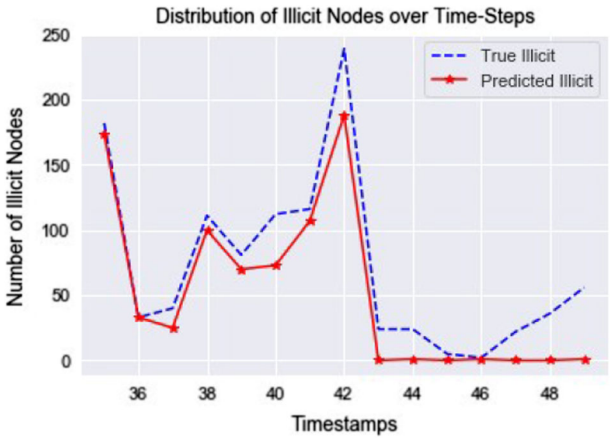

Fig. 3 Illicit transactions distribution in test set over the time-steps. The blue curve represents the actual illicit labels, whereas the red curve represents the illicit predictions that are actually illicit labels by temporal-GCN

in temporal-GCN, LSTM has exploited the temporal sequence of Elliptic data itself before using any graph information in which the new transformed features are mapped non-linearly into the graph-based approach to perform graph convolutions in the vertex domain. Thus, the obtained input features of TAGCN model are enriched with the relevant temporal information. Similarly to [3], we also realise that the presented model performs poorly with the black market shutdown at time-step 43 as shown in Fig. 3. Regarding the time-complexity, the complexity of LSTM is about $\mathcal{O}(4nd_h(d_x + 3 + d_h)$, whereas 2-TAGCN layers have a linear complexity which is $\mathcal{O}(n)$. Consequently, the time-complexity of the temporal-GCN model becomes $\mathcal{O}(n(4\ d_h\ d_x + 3\ d_h + d_h^2 + 1))$ at every epoch.

7.2 Evaluation of Active Learning Frameworks

Referring to Fig. 4, we plot the results of various active learning frameworks using various acquisition functions (BALD, Entropy, Mean STD, Variation Ratio) which in turn utilise MC-dropout and MC-AA uncertainty estimation methods. Moreover, we plot the performance of the baseline model using a random sampling strategy. In the first subplot, BALD has revealed a significant success under MC-AA and MC-dropout uncertainty estimates which active learning is effectively better than the random sampling model. With the remaining acquisition functions, MC-dropout has remarkably achieved a significant outperformance over MC-dropout and the random sampling model.

MC-AA that is utilised in entropy and variation ratio acquisition function has not performed better than random sampling. Furthermore, the active learning framework using the BALD acquisition function in Fig. 4 is capable of matching the performance of a fully supervised model after using 20% of the queried data. This amount of queried data belongs to the second iteration. In our experiments, MC-AA has been revealed to be a viable method as an uncertainty sampling strategy in an active learning approach with BALD and Mean STD acquisition functions. This is reasonable since the latter two methods estimate the uncertainty based on the severe fluctuations of the model's predictions on a given input wherein MC-AA suits this type of uncertainty.

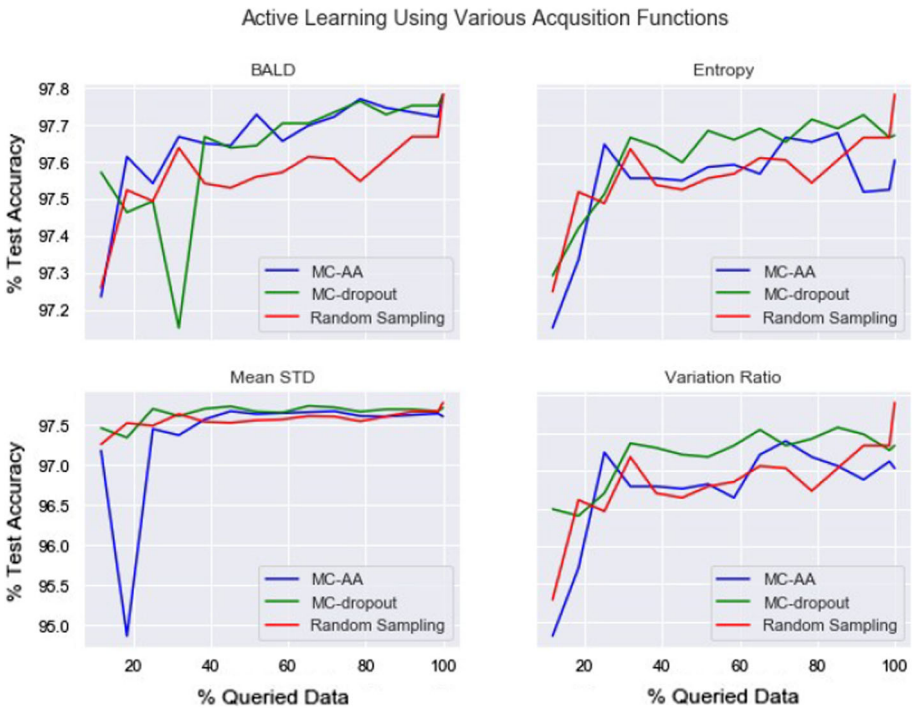

Fig. 4 Results of active learning using BALD via MC-dropout in comparison to BALD via MC-AA

Referring to Table 2, BALD acquisition has recorded the shortest time among other acquisition functions using MC-AA, where this framework has been processed in 28.07 minutes using parallel processing. Whereas the longest time by MC-AA is recorded by the entropy and variation ratio. For MC-dropout, the shortest time is recorded by Mean STD acquisition which is 27.09 min. Whereas the framework using variation ratio has revealed the longest time which is 28.3 min. We also note that the frameworks using MC-AA require more time than the ones using the MC-dropout method. This is due to the adversaries computed by MC-AA which requires more time.

7.3 Wilcoxon Hypothesis Test

To show the statistical significance of the various acquisition functions that appeared in Fig. 4, we perform the non-parametric Wilcoxon signed-rank test [40]. It is used to test the null hypothesis between two paired samples based on the difference between their scores. Given two paired samples $P = \{p_1, \ldots, p_m\}$ and $Q = \{q_1, \ldots, q_m\}$, then the absolute value of the difference between the samples.

This can be expressed as:

$$M = |P - Q| = \{|p_1 - q_1|, \ldots, |p_m - q_m|\},$$

where m is the number of samples of each set. In summary, this test accounts for the statistical difference between the sets P and Q. The Wilcoxon test compares a test statistic T to *Student's t-distribution*. To perform this test, we use the Wilcoxon function in sklearn [41]. Referring to

Table 3 Wilcoxon statistical test between the accuracy derived by MC-AA/MC-dropout with respect to the random sampling model using various acquisition functions

Acquisition function	Wilcoxon (MC-AA, random sampling)	Wilcoxon (MC-dropout, random sampling)
BALD	0.0009	0.0217
Entropy	0.2552	0.309
Variation ratio	0.266	0.0071
Mean STD	0.5507	0.0112

The values correspond to the p values by the test statistic

Fig. 4, we apply the Wilcoxon test for each subplot twice. The first one studies the difference between MC-AA and random sampling curves. Likewise, the second one accounts for the differences between MC-dropout and random sampling curves. For instance, we will refer to the Wilcoxon test applied between MC-AA and random sampling pairs as follows:

$$\text{p - value} = \text{Wilcoxon}(\text{MC} - \text{AA}, \text{Random Sampling}),$$

where p *value* is a statistical measurement that is used to validate how likely the difference of the paired samples is given by the null hypothesis. Let H_0 be the null hypothesis and H_1 be the alternative one. Then they can be written as:

$$H_0 = \text{No difference between the two paired samples}$$
$$H_1 = \text{There is a difference between the two paired samples}$$

We opt for 0.05 for the significance level α where we test against the null hypothesis. The smaller the p value by the Wilcoxon function output, the evidence we have against the null hypothesis. Precisely, the test statistic is statistically significant if p value is less than the significance level. Subsequently, we provide the statistical results (p values) of the various acquisition functions against the baseline random sampling. The results are provided in Table 3. The p values—which are lower than the significance level α—from BALD acquisition function are statistically significant against the null hypothesis in which there is statistical evidence about the difference between each of MC-AA and MC-dropout in comparison to random sampling acquisition. Moreover, MC-dropout against random sampling has shown a statistical significance against the null hypothesis using the variation ratio and Mean STD acquisitions where the p values are 0.071 and 0.0112, respectively. There is no evidence against the null hypothesis for the entropy where the p values are 0.2552 and 0.309 are greater than α.

8 Ablation Study

In this section, we present the ablation studies performed on the proposed temporal-GCN model. Referring to Table 4, we have studied the importance of using LSTM and TAGCN layers. The Model-0 corresponds to the model performed in the experiments. Subsequently, we have applied a combination of replacing each of the given layers with a linear layer (Model-1, Model2, Model-3). We have also studied the case in which we remove one of the first two layers from the original model (Model-6, Model-7). Furthermore, we have shown the performance of the models using either LSTM (Model-4) with linear layers. In Model-2, the replacement of the second layer with a linear layer has attained the highest

Table 4 Ablation study over the layers of the proposed model

Model number	Layer-1	Layer-2	Layer-3	% Accuracy
Model-0 (ours)	LSTM	TAGConv	TAGConv	97.77
Model-1	Linear	TAGConv	TAGConv	97.66
Model-2	LSTM	Linear	TAGConv	97.76
Model-3	LSTM	TAGConv	Linear	97.57
Model-4	LSTM	Linear	Linear	97.44
Model-5	Linear	Linear	Linear	97.51
Model-6	LSTM	TAGConv	None	97.63
Model-7	TAGConv	TAGConv	None	96.65

Each model number is provided by its architecture by changing the layers into linear layer or removing one of the layers. The term'None' in the cells correspond to the model having one of its layers removed

Table 5 Ablation study by changing the number of K-hops in TAGCN layers of the temporal-GCN model as given in Eq. 16

TAGCN K-hops	% Accuracy
$K = 3$	97.77
$K = 2$	97.71
$K = 1$	97.75

accuracy in comparison to Model-0. The removal of LSTM in all cases has provided a drop in the model's performance, especially in Model-7 which reveals the lowest accuracy. On the other hand, using LSTM without the graph learning algorithms in Model-2 has recorded the second-lowest accuracy in this study. We have also tweaked the K hyper-parameter that appeared in TAGCN referring to Eq. 15. The original model uses, by default, K=3 which means that neighbourhood information is aggregated up to 3-hops. Then we have checked the performance for $K \in \{1, 2\}$ as provided in Table 5. The highest accuracy has been recorded for using $K = 3$ and the second one for $K=1$. Surprisingly, the drop in accuracy is not consistent between the different K-values. This might be due to the informative features derived from neighbouring nodes up to 1-hop and 3-hops.

9 Conclusion

For anti-money laundering in Bitcoin, we have presented temporal-GCN, as a combination of LSTM and GCN models, to detect illicit transactions in the Bitcoin transaction graph known as Elliptic data. Also, we have provided active learning using two promising methods to compute uncertainties called MC-dropout and MC-AA. For the active learning frameworks, we have studied various acquisition functions to query the labels from the pool of unlabelled data points. The main finding is that the proposed model has revealed a significant outperformance in comparison to the previous studies with an accuracy of 97.77% under the same experimental settings. LSTM takes into consideration the temporal sequence of Bitcoin transaction graphs, whereas TAGCN considers the graph-structured data of the top-K influential nodes in the graph. Regarding active learning, we are able to achieve an acceptable performance by only

considering 20% of the labelled data with the BALD acquisition function. Moreover, a non-parametric statistical method, the so-called Wilcoxon test, is performed to test whether there is a difference between the type of uncertainty estimation method used in the active learning frameworks under the same acquisition function. Furthermore, an ablation study is provided to highlight the effectiveness of the proposed temporal-GCN.

In future work, we foresee performing different active learning frameworks which utilise different acquisition functions. Furthermore, we seek to extend the temporal-GCN model to other graph-structured datasets for anti-money laundering in blockchain.

References

1. Ciphertrace (2020) spring 2020 cryptocurrency crime and anti-money laundering report. https://ciphertrace.com/spring-2020-cryptocurrency-anti-money-laundering-report/. Accessed 2021-09-01
2. Harlev MA, Sun Yin H, Langenheldt KC, Mukkamala R, Vatrapu R (2018) Breaking bad: de-anonymising entity types on the bitcoin blockchain using supervised machine learning. In: Proceedings of the 51st Hawaii International Conference on System Sciences
3. Weber M, Domeniconi G, Chen J, Weidele DKI, Bellei C, Robinson T, Leiserson CE (2019) Anti-money laundering in bitcoin: experimenting with graph convolutional networks for financial forensics. arXiv preprint arXiv:1908.02591
4. Alarab I, Prakoonwit S, Nacer MI (2020) Comparative analysis using supervised learning methods for anti-money laundering in bitcoin. In: Proceedings of the 2020 5th international conference on machine learning technologies, pp 11–17
5. Alarab I, Prakoonwit S, Nacer MI (2020) Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In: Proceedings of the 2020 5th international conference on machine learning technologies, pp 23–27
6. Pareja A, Domeniconi G, Chen J, Ma J, Suzumura T, Kanezashi H, Kaler T, Schardl T, Leiserson C (2020) Evolvegcn: evolving graph convolu- tional networks for dynamic graphs. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 5363–5370
7. Poursafaei F, Rabbany R, Zilic Z (20201) Sigtran: signature vectors for detecting illicit activities in blockchain transaction networks. In: PAKDD (1). Springer, pp 27–39
8. Alarab I, Prakoonwit S, Nacer MI (2021) Illustrative discussion of mc-dropout in general dataset: uncertainty estimation in bitcoin. Neural Process Lett 53(2):1001–1011
9. Lorenz J, Silva MI, Aparˊ ıcio D, Ascens˜ao JT, Bizarro P (2020) Machine learning methods to detect money laundering in the bitcoin blockchain in the presence of label scarcity. arXiv preprint arXiv:2005.14635
10. Gal Y, Ghahramani Z (2016) Dropout as a bayesian approximation: represent ing model uncertainty in deep learning. In: international conference on machine learning. PMLR, pp 1050–1059
11. Alarab I, Prakoonwit S (2021) Adversarial attack for uncertainty estimation: identifying critical regions in neural networks. arXiv:2107.07618
12. Farrugia S, Ellul J, Azzopardi G (2020) Detection of illicit accounts over the ethereum blockchain. Expert Syst Appl 150:113318
13. Settles B (2009) Active learning literature survey
14. Qiu J, Wu Q, Ding G, Xu Y, Feng S (2016) A survey of machine learning for big data processing. EURASIP J Adv Signal Process 2016(1):1–16
15. Lewis DD, Catlett J (1994) Heterogeneous uncertainty sampling for supervised learning. In: Machine learning proceedings 1994. Elsevier, pp 148–156

16. Settles B, Craven M, Ray S (2007) Multiple-instance active learning. In: Advances in neural information processing systems, vol 20
17. Gal Y, Islam R, Ghahramani Z (2017) Deep Bayesian active learning with image data. In: International conference on machine learning. PMLR, pp 1183–1192
18. Gal Y, Hron J, Kendall A (2017) Concrete dropout. arXiv preprint arXiv:1705.07832
19. Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27(3):379–423
20. Houlsby N, Husz´ar F, Ghahramani Z, Lengyel M (2011) Bayesian active learning for classification and preference learning. arXiv preprint arXiv:1112.5745
21. Johnson EH (1966) Freeman: elementary applied statistics: for students in behavioral science (book review). Soc Forces 44(3):455
22. Kendall A, Badrinarayanan V, Cipolla R (2015) Bayesian segnet: model uncer- tainty in deep convolutional encoder-decoder architectures for scene under- standing. arXiv preprint arXiv:1511.02680
23. Kampffmeyer M, Salberg A-B, Jenssen R (2016) Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 1–9
24. Gal Y (2016) Uncertainty in deep learning. Univ Camb 1(3):4
25. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
26. Chakraborty A, Alam M, Dey V, Chattopadhyay A, Mukhopadhyay D (2018) Adversarial attacks and defences: a survey. arXiv preprint arXiv:1810.00069
27. Goodfellow IJ, Shlens J, Szegedy C (2014) Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572
28. Lewis DD, Gale WA (1994) A sequential algorithm for training text classifiers. CoRR abs/cmp-lg/9407020. arXiv:cmp-lg/9407020. http://arxiv.org/abs/cmp-lg/9407020
29. Kirsch A, Van Amersfoort J, Gal Y (2019) Batchbald: Efficient and diverse batch acquisition for deep Bayesian active learning. Adv Neural Inf Process Syst 32:7026–7037
30. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
31. Graves A, Mohamed A, Hinton G (2013) Speech recognition with deep recur- rent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, pp 6645–6649
32. Srivastava N, Mansimov E, Salakhudinov R (2015) Unsupervised learning of video representations using lstms. In: International conference on machine learning. PMLR, pp 843–852
33. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp 3104–3112
34. Graves A (2013) Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850
35. Du J, Zhang S, Wu G, Moura JM, Kar S (2017) Topology adaptive graph convolutional networks. arXiv preprint arXiv:1710.10370
36. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. Adv Neural Inf Process Syst 29:3844–3852
37. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolu tional networks. arXiv preprint arXiv:1609.02907
38. Seo Y, Defferrard M, Vandergheynst P, Bresson X (2018) Structured sequence modeling with graph convolutional recurrent networks. In: International conference on neural information processing. Springer, pp 362–373
39. Fey M, Lenssen JE (2019) Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428
40. Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83
41. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. J MACH LEARN RES 12:2825–2830