Early Anomaly Detection by Learning and Forecasting Behavior

Tong Zhao*, Bo Ni*, Wenhao Yu, Meng Jiang
Department of Computer Science and Engineering, University of Notre Dame, USA
{tzhao2,bni,wyu1,mjiang2}@nd.edu

ABSTRACT

Graph anomaly detection systems aim at identifying suspicious accounts or behaviors on social networking sites and e-commercial platforms. Detecting anomalous users at an early stage is crucial to minimize financial loss. When a great amount of observed behavior data are available, existing methods perform effectively though it may have been too late to avoid the loss. However, their performance would become unsatisfactory when the observed data are quite limited at the early stage. In this work, we propose ELAND, a novel framework that uses behavior data augmentation for early anomaly detection. It has a Seq2Seq-based behavior predictor that predicts (i) whether a user will adopt a new item or an item that has been historically adopted and (ii) which item will be adopted. ELAND exploits the mutual enhancement between behavior prediction and graph anomaly detection. The behavior graph is augmented with the predicted behaviors such that the graph-based anomaly detection methods can achieve better performance, and the detection results can support the behavior predictor in return. Experiments show that ELAND improves the performance of a variety of graph-based anomaly detection methods. With the augmented methods in ELAND, the performance of anomaly detection at an earlier stage is comparable with or better than non-augmented methods on a greater amount of observation.

ACM Reference Format:
Tong Zhao*, Bo Ni*, Wenhao Yu, Meng Jiang. 2021. Early Anomaly Detection by Learning and Forecasting Behavior. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

During the last two decades, we have witnessed a boom in social networks and many web services. While it certainly makes people's life easier and more convenient, it indirectly creates a market for malicious incentives. One can make huge profits by posting fake reviews on Yelp and Amazon, or boosting specific topics to trending topics/hashtags on Twitter and Weibo. Such behaviors have serious negative impact on our society [29]. For example, the fake trending topics have tremendous effects on political outcomes [3, 49]. The fake reviews constantly undermine customers' ability to make fair judgements. The fake followers will cause inflated popularity, give false credentials, and ruin user experiences. In the past decade, there has been a line of research detecting anomalous behaviors on social networks [1, 19, 24]. Most of the work focused on detecting the anomalies after they have done the bad things. However, detecting the anomalous users after they have achieved their goals is a sub-optimal solution. Their malicious behaviors have already affected the vast majority of normal users [50]. For example, the boosted topic might have already been on the trending list on Twitter for hours. Millions of users may have already seen and believed it. The fake negative reviews on Yelp could have already affected plentiful users' impression on the restaurant. Therefore, we argue that anomaly detection would be much more useful when it could be done early to stop the malicious behaviors before they achieve their targets. In this work, we study the problem of early anomaly detection on social networking sites.

Performing anomaly detection at an early stage is challenging due to the scarcity of available observations. Despite the effectiveness of existing graph anomaly detection methods, we nevertheless witness a decline in performance when data are insufficient or incomplete. Empirically we observed that popular anomaly detection methods such as FRAUDAR [19] would have a decrease of *15%* on Area under the ROC curve (AUC) when only the earliest 10% of the data were available. Similar cutback also occurred for other types of graph mining-based anomaly detection methods as well as graph representation learning methods. Can the methods achieve an comparable performance at an early stage when data were incomplete?

Our idea is to learn and forecast behaviors to "create" data to boost early anomaly detection. Although one anomalous user might not have sufficient behaviors to be detected, the detection methods could still identify him with high confidence if his possible future behaviors are provided. That is, we predict his future behaviors by finding behavioral patterns from the entire dataset and generating a sequence of items that the user may adopt in the future. Even though the observation on this specific user is limited, the big data of the whole population where his associates might already show their suspicious behavior patterns can help. With the predicted behaviors of a large number of users, the "user-adopts-item" graph data can be augmented and containing much richer information than before. Thus the detection methods can more accurately detect anomalies from the graph data with augmented behaviors.

Present work. We propose ELAND (Early Anomaly Detection) , a novel early graph anomaly detection framework that has two components: (1) an anomaly detection module that detects anomalous users from graph data and (2) a Seq2Seq user behavior forecasting module that augments the graph. We present two methods of training for the proposed framework: (1) ELAND-ITR where the two components are inter-dependent on each other and hence trained iteratively in a bootstrapping style and (2) ELAND-E2E where we

* Equal contribution.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

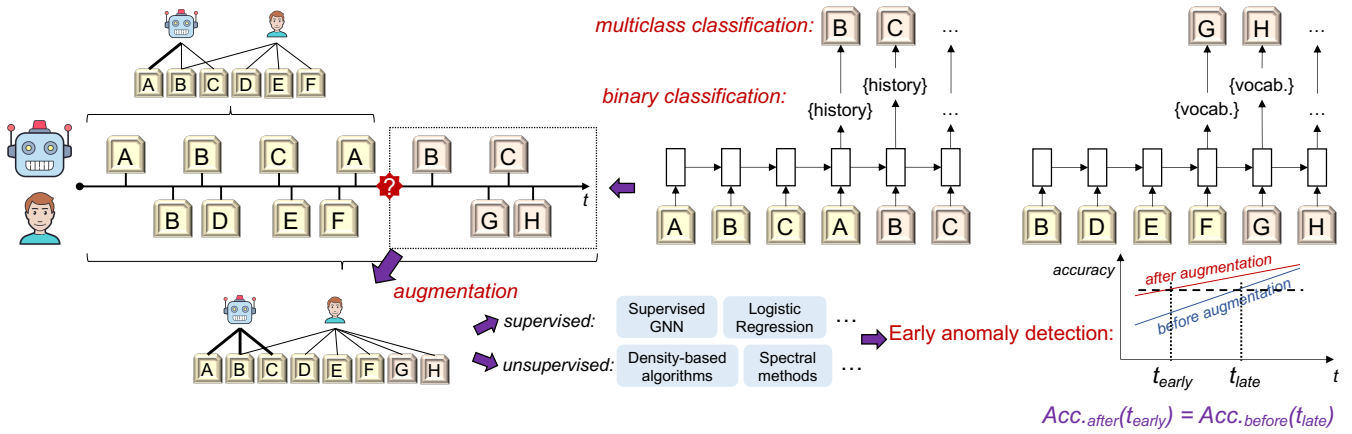


Figure 1: The early anomaly detection framework: The Seq2Seq model predicts the items that a user will adopt in the future based on his/her behavior history in two steps: The first step predicts whether the item has been in the user's behavior history or is a new item in the global "vocabulary." The second step predicts the item from the historical items or from the vocabulary. Then, the framework uses the predicted items to augment user-item bipartite behavior graphs. The predicted items and links will improve the performance of a variety of graph anomaly detection methods including semi-supervised learning (e.g., GCN [25]) and unsupervised learning (e.g., Fraudar [19], LockInfer [23]). Our framework's performance of anomaly detection at an earlier stage is comparable with non-augmented methods with more observed data.

jointly train the two modules as an end-to-end model. Shown in Figure 1 is the illustration of the two modules. The behavior forecasting module consists of a two-step decoder to (1) predict whether the next item a user will adopt comes from his/her behavior history and (2) predict which item will be adopted through similarity matching. The framework takes advantage of the patterns of benign and malicious users that the detection module captures to enhance the behavior prediction.

The contributions of this work are summarized as follows:

- We propose a novel idea that is to achieve early-stage graph anomaly detection by learning and forecasting behaviors. Considering behavior data as sequences, Seq2Seq models can be employed to forecast the behaviors.
- We design a novel framework of two components, behavior forecasting and augmented graph anomaly detection, that achieves early anomaly detection (see Figure 1).
- We conduct extensive experiments on Weibo data and achieve better performance on both unsupervised and supervised anomaly detection methods when data were incomplete.

2 RELATED WORK

Our work addresses the early-stage graph anomaly detection problem using a two-step sequence predictor for data augmentation, so in this section, we will review related work on graph anomaly detection, sequence prediction and graph neural networks.

2.1 Graph Anomaly Detection

This topic has received a great amount of academic interest in the past decade [9, 13, 24, 62, 64]. Several methods [1, 44] were proposed following the graph outlier detection strategy. Similar approaches have been developed for bipartite graphs [51]. Flock [46] utilized dense subgraph detection algorithms to find the suspicious dense blocks in the graph. Fraudar [19] and Mzoom [48] showed that the edge density-based suspiciousness of subgraph or subtensor can be maximized with approximation guarantee. SpokEn [42] found the "spokes" pattern on pairs of eigenvectors of graphs. LockInfer [23] identified pattern of communities based on singular vectors of graphs. CatchSync [22] and CrossSpot [21] found the lockstep behaviors made by anomalous users. fBox [47] located mini-scale attacks missed by spectral techniques. SpotLight [12] detected outlier users in streaming graphs. Rev2 [28] was an iterative algorithm that calculated reviewer fairness scores. Dominant [10] was an attributed graph auto-encoder that detects anomalous nodes. NetProbe [39] inferred suspicious behaviors by looking at each user's neighbors and using belief propagation. Tian et al. [54] studied the early detection of rumors on Twitter.

2.2 Sequence Prediction

Sequential data prediction is considered by many as a key problem in machine learning and artificial intelligence [26]. For example, language models in natural language processing (NLP) aims to predict the next word in textual data based on the given context [2, 52]; action prediction in computer vision (CV) is to infer the action after a series of actions that has been observed [27, 33]. Recurrent neural network (RNN) has enjoyed considerable success in various sequence prediction tasks due to their powerful ability of understanding the structure of data dynamically over time and producing high prediction capacity. Long short term memory (LSTM) network, as an extension of RNN, can learn long term dependencies of the input data [18]. In this work, we view user's adopted items as a sequence and focus on predicting the items that the user is likely to adopt in the future. Some previous works in Recommender Systems [5, 20, 43] also used sequence prediction to address the cold-start problem. Here we take one step further to exploit the

mutual enhancement of sequence prediction and graph anomaly detection methods.

2.3 Graph Neural Networks

In the past few years, following the initial idea of convolution based on spectral graph theory [4], many spectral GNNs have since been developed and improved by [8, 17, 25, 30, 32]. As spectral GNNs generally operate (expensively) on the full adjacency, spatial-based methods which perform graph convolution with neighborhood aggregation became prominent [14, 16, 37, 38, 55, 63], owing to their scalability and flexibility [58]. Moreover, several works proposed more advanced architectures which add residual connections to facilitate deep GNN training [31, 57]. More recently, dynamic graph learning methods [34, 36] have been proposed to explore the idea of combining GNNs with recurrent neural networks for learning with dynamic graphs. GCRN [45] proposed a modified LSTM [18] by replacing fully connected layers with GCN layers [25]. EvolveGCN [40] proposed to use GRU [6] to learn the parameter changes in GCN [25] instead of node representation changes.

3 PROBLEM DEFINITION

Consider a bipartite graph $\mathcal{G}_t = (\mathcal{U}, \mathcal{V}, \mathcal{E}_t)$ at timestamp t, where $\mathcal{U}$ is the set of m users, $\mathcal{V}$ is the set of n items and $\mathcal{E}_t$ is the set of edges at time t. Let $\mathbf{A}_t \in \mathbb{Z}^{m\times n}$ and $\mathbf{X}_t \in \mathbb{R}^{(m+n)\times k}$ be the adjacency matrix and feature matrix at time t where k is the dimensional of raw features. In the case that user features and item features have different dimensions, one of them can be projected to the same space as the other via linear transformation. We also denote $\mathbf{y} \in \{0,1\}^m$ as labels for users where anomalies get 1 and others get 0. We follow the widely accepted definition of anomalous users in graphs by previous works [19, 22, 23]. In social networks such as micro-blogging platforms, the anomalous user accounts are the botnets or the ones that frequently post advertisements or malicious links. Following the above notations and definitions, we first define the task of anomalous user detection, and then proceed to give a formal definition of early-stage anomaly detection.

Definition 1. *(Anomalous User Detection) Given the bipartite graph $\mathcal{G}$ and feature matrix $X \in \mathbb{R}^{(m+n)\times k}$, find a function $g : \mathcal{G}, \mathbf{X} \rightarrow \hat{\mathbf{y}}$ that returns a vector of prediction logits $\hat{\mathbf{y}} \in [0,1]^m$.*

Next, we define early-stage anomaly detection. Let T be a late time when the observed data is sufficient for existing graph anomaly detection methods to perform well. We aim to design a framework that can achieve comparable or better performance at an early time t when the observations are incomplete. Formally, our goal is to find a data augmentation framework satisfying the following criteria:

Definition 2. *(Early Stage Anomalous User Detection) Let $h : (\mathbb{R}^m, \mathbb{R}^m) \rightarrow \mathbb{R}$ be an evaluation metric (e.g., f-measure) such that a larger value is more desirable holding other conditions the same. Let g be an anomaly detection method. Design a function $f : \mathcal{G}_t, \mathbf{X}_t \rightarrow \mathcal{G}'_t, \mathbf{X}'_t$ that satisfies $h(g(f(\mathcal{G}_t, \mathbf{X}_t)), \mathbf{y}) \geq h(g(\mathcal{G}_t, \mathbf{X}_t), \mathbf{y})$. If we assume the performance to be monotonically increasing as more data become available, the above criteria is also equivalent to $\exists T > t$ s.t.*

$$h(g(f(\mathcal{G}_t, \mathbf{X}_t)), \mathbf{y}) \geq h(g(\mathcal{G}_T, \mathbf{X}_T), \mathbf{y}) \tag{1}$$

When applying graph anomaly detection methods on real-world data, which usually has high complexity and uncertainty, it's hard to theoretically guarantee that the performance would be monotonically increasing as more data become available. In the following two sections, we first introduce our proposed framework Eland which approximates $g(f(\mathcal{G}_t, \mathbf{X}_t))$; then we show that Eland *empirically* satisfies the above desired property.

4 ELAND: THE FRAMEWORK

In this section, we first present the two major components of our proposed Eland approach towards the above problem. Then we introduce two ways of training Eland: a bootstrapping style iterative training Eland-itr that can be used on any existing graph anomaly detection methods; and an end-to-end model Eland-e2e that combines the proposed behavior forecasting module with neural based graph anomaly detection methods into one model.

4.1 Graph Anomaly Detection Module

The first component of our proposed Eland framework is a graph anomaly detection module. It is worth pointing out that this part of Eland is general and model-agnostic, in the sense that it can be directly applied to any anomaly detection or node classification model (e.g., Fraudar [19], CatchSync [22], GCN [25], GraphSAGE [16]) suffices to be the component of the framework and could have its performance improved by the proposed Eland. Without the loss of generality, let the anomaly detection model g_{fd} be defined as:

$$\hat{\mathbf{y}} = g_{fd}(\mathbf{A}, \mathbf{X}; \Theta), \tag{2}$$

where $\hat{\mathbf{y}} \in [0,1]^m$ is the predicted suspiciousness of the user nodes of being anomalies, $\mathbf{A}$ is the adjacency matrix, $\mathbf{X}$ is the node feature matrix, and Θ stands for the trainable parameters.

Neural-based graph representation learning methods (e.g., graph neural networks) are capable of learning low-dimensional node representations as well as making predictions. We take advantage of the learned representations of user nodes. Without the loss of generality, here we take the widely used Graph Convolutional Network (GCN) [25] as an anomalous user detection method as it is capable of semi-supervised node classification. The graph convolution operation of each GCN layer is defined as:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \tag{3}$$

where l indicates the layer, $\mathbf{H}^l$ is the node embedding matrix generated by l-th layer, $\mathbf{W}$ is the weight matrix of the co-responding layer, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with added self-loops, $\tilde{\mathbf{D}}$ is the diagonal degree matrix $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and $\sigma(\cdot)$ denotes a nonlinear activation such as the Rectified Linear Unit (ReLU).

Thus the graph neural networks (GNNs) used for graph anomaly detection can be notated as:

$$\hat{\mathbf{y}}, \mathbf{Z} = g_{fd-gnn}(\mathbf{A}, \mathbf{X}; \Theta), \tag{4}$$

where g_{fd-gnn} is a multi-layer GNN model, $\mathbf{Z}$ is the node embedding matrix generated by the second last layer, and $\hat{\mathbf{y}}$ is the predicted user suspiciousness by the last layer. We use a standard binary cross entropy loss when training the GNNs individually.

4.2 Behavior Forecasting Module

The behavior forecasting module provides data augmentation on the user-item bipartite graph with a sequence-to-sequence neural model. It is an LSTM-based encoder-decoder network [53]. This

module is designed to capture behavior patterns from the sequential data (i.e., item adoption history) and use them for graph augmentation. In general, it takes the following functional form:

$$\mathbf{A}' = g_{bf}(\mathbf{A}, \mathbf{X}, \hat{\mathbf{y}}; \Theta), \tag{5}$$

where $\mathbf{A}'$ stands for the adjacency matrix with augmented predicted behaviors, $\mathbf{A}$ is the original adjacency matrix, $\mathbf{X}$ is the feature matrix, $\hat{\mathbf{y}}$ is the predicted user suspiciousness by g_{fd} or g_{fd-gnn}, and Θ stands for the trainable parameters.

Encoder. Here we regard each user's item history as a sequence of features that are constructed as prior knowledge. For example, for content-based item (e.g., trending topics), the features can be the embedded representations of the texts and/or in-degree and out-degree of the item in the graph. Suppose user u_i has a behavior history of j items. We denote the sequence of the behaviors as $x^{(i)} = \{\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_j^{(i)}\}$, in which each item $\mathbf{x}_j^{(i)} \in \mathbb{R}^k$ stands for the feature vector of the corresponding item node. If a GNN-based anomaly detection module is used to learn node representations, say $\mathbf{z}_i$ for user u_i, then the behavior history is represented as $x^{(i)} = \{\mathbf{x}_1^{(i)} \oplus \mathbf{z}_i, \ldots, \mathbf{x}_j^{(i)} \oplus \mathbf{z}_i\}$, where $\oplus$ stands for vector concatenation. For the ease of reading, we omit user index i, denoting $\mathbf{x}_j^{(i)}$ as $\mathbf{x}_j$ and $\mathbf{x}^{(i)}$ as $\mathbf{x}$ in the following of this section, and the following use of i will be mainly for indication of time stamps.

We adopt LSTM encoder to capture contextualized representations of each item in the sequence. The encoder contains a forward LSTM to make predictions based on the historical items. Hence the hidden state of each item is:

$$\mathbf{h}_i, \mathbf{c}_i = \overrightarrow{\text{LSTM}}(\mathbf{x}_i, \mathbf{h}_{i-1}, \mathbf{c}_{i-1}), \tag{6}$$

where $\mathbf{h}_i$ and $\mathbf{c}_i$ refer to the hidden state and cell state in the i-th step, respectively.

Decoder. We use a two-step decoder for predicting the items from the hidden states. First, the decoder decides whether the user is going to select an item that has already been adopted. Based on the results of the first step, the decoder further predicts the item by finding the most similar item in the "vocabulary." The two-step design is motivated from our observation that a great number of anomalous entities tend to repeatedly adopt things from the same set of items, compared with benign users.

We first decode the hidden states and cell states for each item, the readout operation of which is defined as follows:

$$\hat{p}_i = \phi(\mathbf{W}_{rh} \cdot \mathbf{h}_i + \mathbf{W}_{rc} \cdot \mathbf{c}_i + \mathbf{b}_r), \tag{7}$$

where $\mathbf{W}_{rh}, \mathbf{W}_{rc} \in \mathbb{R}^{k \times k}$ and $\mathbf{b}_r \in \mathbb{R}^k$ are trainable parameters, $\phi(\cdot)$ is the sigmoid function, and $\hat{p}_i$ is the probability of repeating item. Bigger $\hat{p}_i$ indicates that the user is more likely to perform a repeating behavior (e.g., re-post a trending topic that the user has posted in the past).

Moreover, we use another readout function to decode the prediction of next item by

$$\hat{\mathbf{x}}_{i+1} = \mathbf{W}_p \cdot \mathbf{x}_i + \mathbf{b}_p, \tag{8}$$

where $\mathbf{W}_p \in \mathbb{R}^{|\mathbf{h}| \times k}$, $\mathbf{b}_p \in \mathbb{R}^k$ are trainable parameters and $\hat{\mathbf{x}}_{i+1}$ is the predicted feature vector for the next item.

Algorithm 1: ELAND-ITR

Input : Adjacency matrix $\mathbf{A}$; node feature matrix $\mathbf{X}$; number of iterations I; anomaly detection module g_{fd} or g_{fd-gnn}; behavior forecasting module g_{bf}.
Output : User prediction results $\hat{\mathbf{y}}$.

```
if GNN-based anomaly detection module then
    ŷ, Z = g_fd-gnn(A, X) ;          // Defined in Eq.(4)
    X = concat(X, Z) ;
else
    ŷ = g_fd(A, X) ;                 // Defined in Eq.(2)
end
for i in range(I) do
    Re-initialize the parameters Θ in g_bf ;
    A' = g_bf(A, X, ŷ) ;             // Defined in Eq.(5)
    if GNN-based anomaly detection module then
        Re-initialize the parameters Θ in g_fd-gnn ;
        ŷ, Z = g_fd-gnn(A', X) ;
        X = concat(X, Z) ;
    else
        Re-initialize the parameters Θ in g_fd ;
        ŷ = g_fd(A', X) ;
    end
end
return ŷ ;
```

Graph Augmentation. Finally, we augment the graph by explicitly predicting the future items for users and adding the predicted behaviors as edges into the graph. When $\hat{\mathbf{x}}_{i+1}$ is predicted by Eq.(8), the next input item is determined by cosine similarity:

$$\mathbf{x}_{i+1} = \underset{\mathbf{x} \in \mathbf{X}_i}{\text{argmin}} \left(\frac{\hat{\mathbf{x}}_{i+1} \cdot \mathbf{x}}{||\hat{\mathbf{x}}_{i+1}|| \times ||\mathbf{x}||} \right). \tag{9}$$

where $\mathbf{X}_i$ is the current vocabulary calculated by

$$\mathbf{X}_i = \begin{cases} \mathbf{X}, & \hat{p}_i \leq 0.5; \\ \{\mathbf{x}_1, ..., \mathbf{x}_i\}, & \hat{p}_i > 0.5. \end{cases} \tag{10}$$

Thus, we have new edges and get $\mathbf{A}'$ by adding them into $\mathbf{A}$.

4.3 ELAND-ITR

The graph anomaly detection module benefits from the enriched graph structure generated by behavior forecasting, and the forecasting module benefits from the detection module. Both modules are interdependent and mutually enhancing each other. Hence we use an bootstrapping training strategy to iteratively train both models and jointly optimize their performances.

Algorithm 1 shows the process of ELAND-ITR. We start with the anomaly detection module on the original graph. If the anomaly detection module is GNN-based, the features are updated by concatenating the learned node representations with the original node features. Then each iteration proceeds as follows: it trains a new forecasting module g_{bf} with the graph and the results given by the detection module g_{fd} or g_{fd-gnn}; then it trains and makes inference with a new graph anomaly detection module on the updated graph structure $\mathbf{A}'$. The final prediction result $\hat{\mathbf{y}}$ is reported.

During the inference stage, we utilize the predicted suspiciousness scores $\hat{y}_i$ for each user given by the anomaly detection module to decide the number of predictions we make for user u_i:

$$n_i = \lfloor \kappa \cdot \hat{y}_i \rfloor, \tag{11}$$

where $\kappa \in \mathbb{Z}^+$ is a hyperparameter to control the maximum number of predictions. The intuition is that anomalous users tend to perform more behaviors to achieve their goal (e.g., fake trending topic boosting), so we generate more predicted items for the users that are more likely to be anomalies (by the anomaly detection module).

Training ELAND-ITR. The GNN-based anomaly detection module is trained with the standard binary cross entropy:

$$\mathcal{L}_{fd-gnn} = -\sum_{u=1}^{|\mathcal{U}|} \big(y_u \log(\hat{y}_u) + (1-y_u)\log(1-\hat{y}_u)\big). \tag{12}$$

The behavior forecasting module is trained with two distinct losses. One loss function supervises the first readout which calculates the probability of repeating item defined in Eq.(7):

$$\mathcal{L}_{repeat} = -\sum_{u=1}^{|\mathcal{U}|}\sum_{i=1}^{j} \big(p_i^{(u)} \log(\hat{p}_i^{(u)}) + (1-p_i^{(u)})\log(1-\hat{p}_i^{(u)})\big), \tag{13}$$

and the second supervises the prediction of future items:

$$\mathcal{L}_{pred} = \frac{1}{j}\sum_{u=1}^{|\mathcal{U}|}\sum_{i=1}^{j} \frac{\hat{\mathbf{x}}_i^{(u)} \cdot \mathbf{x}_i^{(u)}}{||\mathbf{x}_i^{(u)}|| \times ||\mathbf{x}_i^{(u)}||}. \tag{14}$$

Hence the behavior forecasting module g_{bf} can be trained with

$$\mathcal{L}_{bf} = \mathcal{L}_{repeat} + \alpha \cdot \mathcal{L}_{pred}, \tag{15}$$

where α is a hyperparameter to control the weights.

4.4 ELAND-E2E

In addition to ELAND-ITR, we propose an end-to-end model ELAND-E2E that does not require the iterative training process. Hence, it avoids the potential error propagation issue in bootstrapping. The anomaly detection module in ELAND-E2E is a neural model that allows training with back-propagation in order to be trained together with the rest of the model. So, we use GNN models as the detection module. In the decoder of the behavior forecasting module, we omit the first readout layer which decides whether the next item is repeated to give the model more generalizability. The forecasting module can be defined as

$$\mathbf{A}' = g_{bf-e2e}(\mathbf{A}, \mathbf{X}), \tag{16}$$

which does not require $\mathbf{y}'$ as input anymore.

Moreover, when augmenting the graph with g_{bf}, instead of discretely predicting the next item as shown in Eq. 9, we use the cosine similarity of the predictions $\pi = [\pi_1, ..., \pi_{|\mathcal{V}|}]$. Then we apply Gumbel-softmax [11, 35] to obtain π' by:

$$\pi_i' = \frac{\exp\left(\log\left(\pi_i\right) + g_i\right)/\tau)}{\sum_{j=1}^{k}\exp\left(\log\left(\pi_j\right) + g_j\right)/\tau)}, \tag{17}$$

where $g_i \sim Gumbel(0,1)$ is a random variate sampled from the Gumbel distribution and τ is a temperature hyperparameter controlling the distribution of the results. Smaller τ results in more difference between classes, so we use a low temperature τ. Then we use the Straight-Through trick [11] to further discretize π_i' into one-hot vectors and add it into the adjacency matrix without affecting its sparsity. Gradients are directly passed backward to the probabilities before Straight-Through during training.

Algorithm 2: ELAND-E2E

Input : Adjacency matrix $\mathbf{A}$; node feature matrix $\mathbf{X}$; behavior forecasting module g_{bf-e2e}; anomaly detection module g_{fd-gnn}; number of training epochs *n_epochs*.
Output : User prediction results $\hat{\mathbf{y}}$.

```
/* model training                                          */
```
Initialize Θ_{bf-e2e} in g_{bf-e2e} and Θ_{df-gnn} in g_{fd-gnn} ;
for *epoch in range(n_epochs)* **do**
  $\mathbf{A}' = g_{bf-e2e}(\mathbf{A}, \mathbf{X})$; // Defined in Eq.(16)
  $\hat{\mathbf{y}}, \mathbf{Z} = g_{fd-gnn}(\mathbf{A}', \mathbf{X})$; // Defined in Eq.(4)
  Calculate $\mathcal{L}_{e2e}$ with Eq.(19);
  Update Θ_{bf-e2e} and Θ_{df-gnn} with $\mathcal{L}_{e2e}$;
end
```
/* model inferencing                                       */
```
$\mathbf{A}' = g_{bf-e2e}(\mathbf{A}, \mathbf{X})$;
$\hat{\mathbf{y}}, \mathbf{Z} = g_{fd-gnn}(\mathbf{A}', \mathbf{X})$;
return $\hat{\mathbf{y}}$;

We use preferential attachment to calculate the number of predictions. Zang et al. [59] showed that the dynamics of a random variable x of exponential distribution $\frac{dx(t)}{dt}$ is proportional to $x(t)$ (i.e. $\frac{dx(t)}{dt} \propto x(t)$). Since user behavior patterns are generally considered to be consistent [15], here we assume that the item selection process follows a Poisson process, whose frequency distribution could be further generalized to an exponential distribution [7]. Thus we can calculate the number of prediction for user u_i using preferential attachment:

$$n_i = \left\lfloor \beta \cdot \frac{x_i}{\sum x} \right\rfloor, \tag{18}$$

where x_i, x are u_i's degree and all users' degree in the graph, respectively. β is a hyperparameter that controls the number of edges to augment the graph. n_i edges will be predicted for u_i, taking the current distribution of edges into consideration.

Training ELAND-E2E. Two loss functions are used to update the parameters: a standard binary cross entropy loss $\mathcal{L}_{fd-gnn}$ for the predicted labels as defined in Eq.(12) and a Cosine similarity loss function $\mathcal{L}_{pred}$ as defined in Eq.(14). Thus ELAND-E2E is trained with a multi-task loss function defined as

$$\mathcal{L}_{e2e} = \mathcal{L}_{fd-gnn} + \alpha \cdot \mathcal{L}_{pred}, \tag{19}$$

where α is the weight of item-prediction loss.

Algorithm 2 shows the process of ELAND-E2E. In each epoch, the graph structure is augmented by the output from the behavior forecasting module with the Gumbel-softmax trick. The GNN-based anomaly detection module takes the augmented graph to predict labels $\mathbf{y}'$. $\mathcal{L}_{e2e}$ defined by Eq.(19) is used to supervise the model.

Table 1: The proposed ELAND improves the performance of a variety of graph-based mining and learning methods, with early, partial amount of data. Smaller percentage indicates less observed data and earlier graph anomaly detection.

Percentage of available data (%)			10	20	30	40	50	60	70	80	90	100
FRAUDAR [19]	Original	AUC	51.6	53.8	54.6	56.0	57.1	57.1	58.1	59.2	59.5	60.4
		AP	13.9	16.1	16.2	17.4	18.4	18.4	19.0	18.9	19.4	20.0
	+ELAND-ITR	AUC	52.0	54.3	55.3	57.1	58.2	57.6	58.8	60.3	60.3	61.2
		AP	14.3	16.9	17.2	18.5	19.2	19.2	15.4	20.2	20.2	21.3
LOCKINFER [23]	Original	AUC	53.7	56.6	59.2	54.7	55.5	54.1	55.4	55.9	56.1	56.4
		AP	8.5	8.8	9.9	9.4	10.3	10.0	10.0	9.9	9.8	10.4
	+ELAND-ITR	AUC	57.0	59.8	60.4	61.3	61.3	62.6	61.6	61.7	62.3	64.5
		AP	9.5	10.4	12.3	12.4	12.4	17.3	15.2	12.9	12.3	13.7
GCN [25]	Original	AUC	81.7	82.9	83.2	83.4	83.7	83.9	83.9	84.1	84.1	84.1
		AP	40.8	42.8	45.5	47.7	50.1	50.4	50.0	50.1	48.8	49.0
	+ELAND-ITR	AUC	82.5	83.0	83.6	83.5	83.8	84.0	83.9	84.2	84.2	84.1
		AP	39.5	45.0	42.0	46.1	43.8	43.4	45.1	43.6	45.5	46.4
	+ELAND-E2E	AUC	82.7	83.0	83.5	84.6	85.0	85.8	86.0	**86.3**	86.3	86.4
		AP	46.5	46.6	45.9	51.4	50.3	51.9	51.8	51.6	54.2	54.2
GRAPHSAGE [16]	Original	AUC	81.7	81.8	82.2	82.7	82.9	83.0	83.2	83.6	83.4	83.6
		AP	42.5	42.4	42.8	43.0	43.0	42.7	43.2	45.0	47.7	50.9
	+ELAND-ITR	AUC	82.9	83.0	82.7	82.8	83.1	83.6	83.3	83.6	83.5	83.6
		AP	42.8	45.0	43.7	43.8	44.3	43.7	44.0	43.8	48.7	50.8
	+ELAND-E2E	AUC	82.7	84.0	84.6	**86.1**	**86.7**	**86.9**	**86.9**	85.9	**87.0**	**87.0**
		AP	46.0	44.5	44.6	45.3	46.5	47.4	46.4	47.2	46.9	48.9
HETGNN [60]	Original	AUC	82.4	82.8	83.1	83.1	83.6	83.1	82.8	82.9	82.8	83.0
		AP	50.1	51.3	50.8	51.1	52.1	51.0	50.9	50.2	50.8	51.2
	+ELAND-ITR	AUC	83.0	83.1	83.3	83.5	83.7	83.2	83.0	83.0	83.0	83.2
		AP	50.6	52.2	51.2	53.1	52.9	51.6	52.0	53.0	50.7	51.4
	+ELAND-E2E	AUC	**84.2**	**84.9**	**85.2**	84.9	85.5	85.4	85.7	86.1	86.5	86.8
		AP	**55.2**	**56.9**	**57.9**	**58.8**	**56.2**	**56.2**	**57.0**	**57.2**	**56.3**	**55.9**

5 EXPERIMENTS

In this section, we evaluate ELAND for anomaly detection on a real-world dataset to answer the following research questions (**RQ**):

- **RQ1:** Does ELAND improve the performance of graph anomaly detection methods?
- **RQ2:** Can ELAND achieve early anomaly detection with incomplete behavior data as desired in Section 3?
- **RQ3:** Is ELAND able to enhance graph sequence-based evolutionary learning methods for anomaly detection?

5.1 Experimental Settings

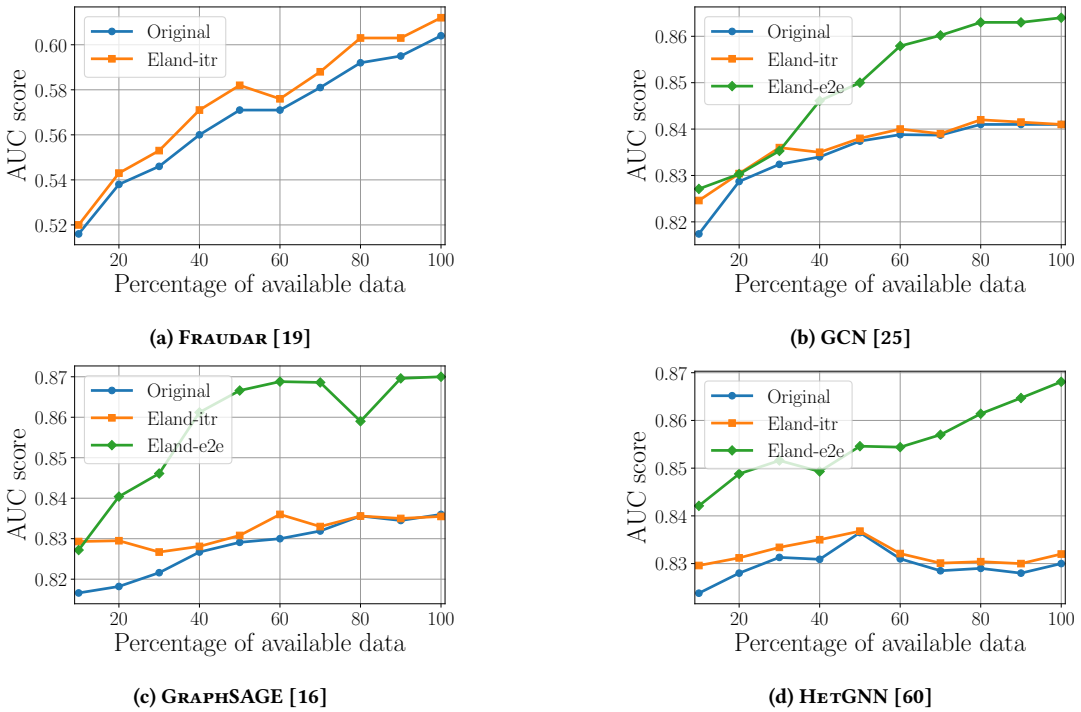
Here we introduce the datasets and baseline methods.

5.1.1 *Datasets.*

We use a micro-blogging dataset from Tencent Weibo that consists of posts and public user profiles. We build "who-post-what" graphs where the user nodes are registered users and item nodes are micro-blog posts. An edges between a users and an item indicate that the user posted or re-posted the item. The features of post items were generated from their text using GLOVE [41] embeddings. The features of users are the aggregated features of their adopted posts.

Graph statistics. The full graph has 40,235 users, 3,288 unique tweets, and 75,285 edges (including tweets and retweets). We divide the dataset based on the percentage of user's posts in the entire dataset. Each 10% has about 7,000 edges.

Positive labels. As the dataset is a "who-post-what" micro-blogging graph, we define the anomalous users in this dataset as the social-spam users accounts which continuously post advertisements or malicious links [19]. Due to the large scale (40K+ users), the dataset does not have manually annotated golden labels. So we labelled the users by their post text, profile information, and their behavior time. Specifically, we use the following criteria to label anomalies:



(a) FRAUDAR [19] (b) GCN [25]

(c) GRAPHSAGE [16] (d) HETGNN [60]

Figure 2: ELAND-ITR and ELAND-E2E perform better than the original version of (a) unsupervised graph mining method and (b–d) semi-supervised graph learning methods in terms of AUC score.

(1) *Social spambot accounts:* The major conspicuous characteristic of suspicious users is their bot-controlled behavior. As we observed, most suspicious users posted in a fixed set of time intervals. For example, a user is identified as an anomaly if more than 2/3 of the time intervals of his posts are within 30 ± 2 seconds. (2) *Accounts with suspicious posts:* We checked the post text and spotted the accounts whose posts are mostly advertisements or content with malicious links. (3) *Deactivated accounts:* We identified the accounts that had certain suspicious posts, did not have associated WeChat accounts [56], and also had been deactivated (which were very likely deactivated by the platform). To further validate the above rule-based labels, we randomly sampled 1,000 user accounts with their post history for hand labeling by experts. Over 97% of the rule-based labels and labels by two experts are consistent.

With these criteria, 3,283 users are identified as anomalies (positive labels) and 36,952 users are identified as benign users. An anonymized version of this dataset will be released for future research in the community.

5.1.2 *Baselines.*

We apply ELAND on the following graph anomaly detection methods that did not have the behavior forecasting module:

- **FRAUDAR** [19]: An unsupervised graph anomaly detection method with efficient graph mining that aims to address the issue of camouflage behaviors.
- **LOCKINFER** [23]: An unsupervised graph anomaly detection method that uses the spectral graph patterns to discover lockstep behaviors of fraudsters as a clue for detection.
- **GCN** [25]: An semi-supervised graph neural network. The neural structure has been empirically demonstrated to be useful in the area of graph anomaly detection [61].
- **GRAPHSAGE** [16]: An inductive graph neural network model that can also be used on semi-supervised node classification.
- **HETGNN** [60]: A graph neural network model that handles the heterogeneous graphs of multiple types of nodes.
- **EVOLVEGCN** [40]: A graph neural network model that takes a sequence of graphs as input and models the dynamics of the parameters in GCN [25] via GRU [6] along time.

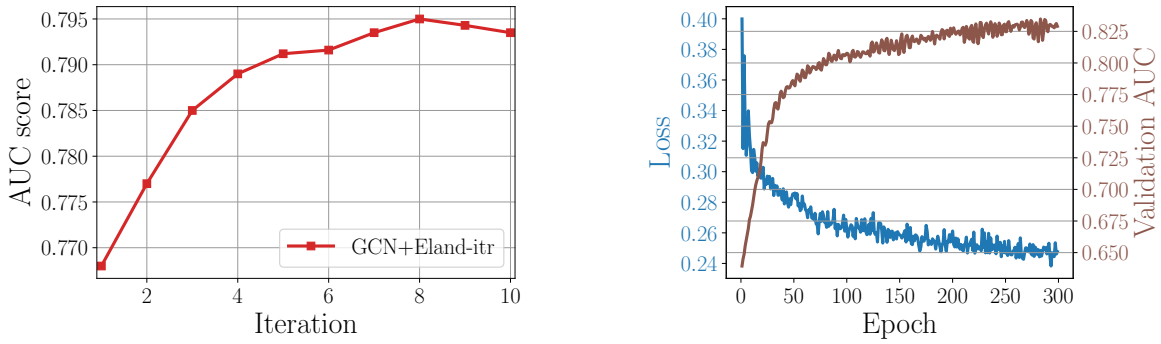
5.2 Experiment Results

Here we present and analyze experimental results to answer the three questions RQ1–3.

5.2.1 *RQ1: Enhancing graph mining and learning methods.*

Table 1 summarizes the performance of ELAND over the baseline methods. We report Area under the ROC Curve (AUC) and Average Precision (AP) as our evaluation metrics. FRAUDAR and LOCKINFER are not neural methods and hence can work only with ELAND-ITR and cannot work with ELAND-E2E. The best performances at each percentage of available data are in bold fonts. In short, our ELAND-ITR and ELAND-E2E consistently improve the basic models on early anomaly detection.

For graph mining methods, ELAND-ITR improves 1.1% on FRAUDAR and 6.0% on LOCKINFER in terms of the average of the AUC score, respectively. However, the results of FRAUDAR and LOCKINFER are generally not as good as the GNN-based methods, because



(a) ELAND-ITR with GCN

(b) ELAND-E2E with GCN

Figure 3: Convergence process of the proposed ELAND-ITR and ELAND-E2E. When it came to the 8th iteration (in ELAND-ITR) or about 200 epochs (in ELAND-E2E), the AUC score of the two methods converge.

Table 2: Given the earliest 20%, 30%, ... 60% data, EVOLVEGCN learns the graph sequence to detect anomalous users. Every 10% data forms a graph snapshot. The proposed methods improve the AUC for early anomaly detection.

Early graph sequence	Original	+ELAND-ITR	+ELAND-E2E
$\{\mathcal{G}_{0-10\%}, \mathcal{G}_{10\%-20\%}\}$	78.2	78.9	**80.9**
$\{\mathcal{G}_{0-10\%}, \ldots, \mathcal{G}_{20\%-30\%}\}$	76.6	79.2	**81.0**
$\{\mathcal{G}_{0-10\%}, \ldots, \mathcal{G}_{30\%-40\%}\}$	78.6	79.6	**79.7**
$\{\mathcal{G}_{0-10\%}, \ldots, \mathcal{G}_{40\%-50\%}\}$	78.7	**80.1**	80.0
$\{\mathcal{G}_{0-10\%}, \ldots, \mathcal{G}_{50\%-60\%}\}$	79.4	79.4	**79.5**

they are unsupervised methods while the neural methods utilize supervisory information.

In general, ELAND-E2E with HETGNN as the detection module achieved the best performance, especially when the data were the most insufficient (i.e., the earliest/sparsest). As more data became gradually available, ELAND-E2E with GRAPHSAGE as the detection module obtained better AUC score with small margins. GRAPHSAGE may have better precision so it sometime achieved the highest AP. Nevertheless, ELAND-E2E with HETGNN keeps the large surplus on AP over all other methods.

5.2.2 *RQ2: Achieving early anomaly detection.*

To better show the improvements of the proposed ELAND framework on early anomaly detection, we present Figure 2, in which we show the trends of AUC scores of original graph anomaly detection methods and our proposed framework change as more observed data is available. We can observe that in general, both ELAND-ITR and ELAND-E2E are able to accomplish the early-stage anomaly detection task as we defined in Section 3; ELAND-E2E generally accomplish more improvements over the original anomaly detection methods than ELAND-ITR. Particularly, for FRAUDAR, we achieve roughly the same performance with 50% of data with ELAND-ITR as with 70% of data without ELAND-ITR; for GNN-based detection methods, ELAND-ITR and ELAND-E2E usually start with similar improvements with achieving the same level of performance with 10% of available data as the baselines with 20% (GCN) / 60% (GRAPHSAGE) / 50% (HETGNN) of data. However, as more observation are available, ELAND-E2E starts to show more ability of improving performance than ELAND-ITR. For example, with only 40% (GCN) / 20% (GRAPHSAGE) / 10% (HETGNN) of available observation, ELAND-E2E is capable of achieving better performance than the original model with all available data.

Although for methods like HETGNN, the baseline is not monotonically increasing, it is also worth mentioning that for all of the baseline detection methods, we are able to observe substantial improvements in absolute performance on average, which are evidence for ELAND's powerful ability to model the evolution of the graph structure and user-item interactions.

Model convergences. Figure 3 presents the convergence progress of the proposed framework ELAND. Figure 3a first shows the change of performance over iterations of ELAND-ITR with GCN. We can observe a smooth yet fast increase of the AUC score at the first few iterations, which then reaches a steady state. The curve shows the bootstrapping iterative training design of ELAND-ITR is meaningful and demonstrates the mutual beneficial relationship between the two modules. Figure 3b shows the convergence of loss and validation AUC during the training stage of ELAND-E2E.

5.2.3 *RQ3: Improving evolutionary graph learning EVOLVEGCN.*

Here we further test the ability of the proposed framework ELAND with EVOLVEGCN [40], which is one the the state-of-the-arts GNN model for modeling dynamic or evolution of graphs. Different from other GNN models that takes a static graph as input, EVOLVEGCN takes a graph sequence as input. To construct the graph sequences required for EVOLVEGCN, we several graphs defined above as the graph sequence. For example, when giving EVOLVEGCN 30% of available data, we give it three graph snapshots each containing 10%/20%/30% of available data as the graph sequence. Moreover, the last graph snapshot after augmented by our proposed forecasting module will be the 4th graph during training.

Table 2 presents the results of EVOLVEGCN and with EVOLVEGCN with both ELAND-ITR and ELAND-E2E on five different graph sequences that gradually contain more information. We observe that

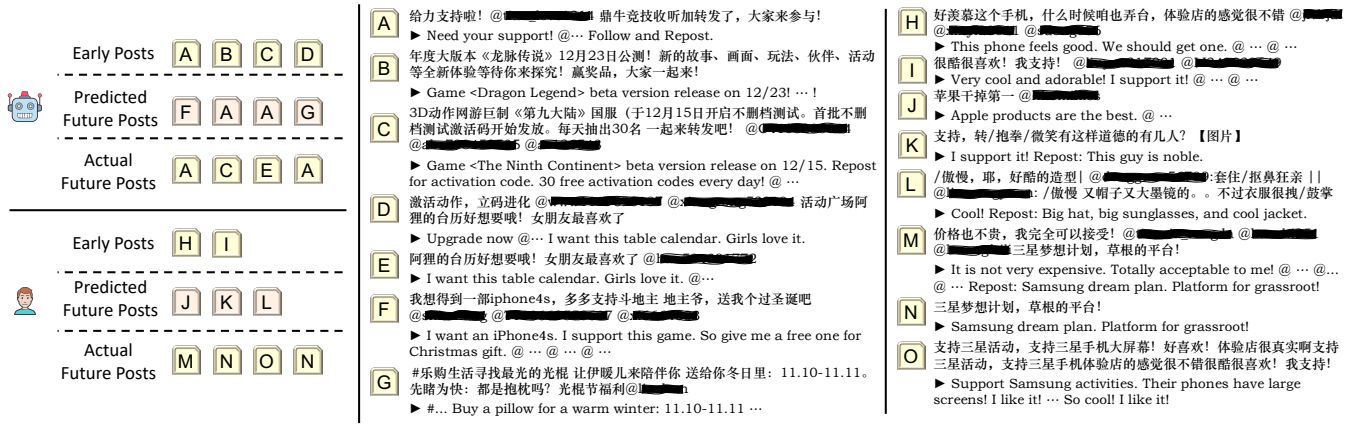


Figure 4: Case studying showing the historical posts, predicted posts (by ELAND-E2E) and actual future posts sequences of two users, as well as the actual post content each followed by a brief translation (marked with solid triangles). This case study shows the previously wrong classified users can be correctly classified with the predicted posts by ELAND-E2E.

the proposed methods can still consistently achieve same or better performance than the original EVOLVEGCN method. Similar as previous experiments, the proposed ELAND framework with EVOLVEGCN outperforms the original model and ELAND-E2E with EVOLVEGCN has the best performances. Both EVOLVEGCN and ELAND-ITR with EVOLVEGCN show improvement in performance as more data is available. However, it is interesting that ELAND-E2E with EVOLVEGCN achieves the best performance on the third graph sequence, which contains only three graph snapshots. As more information (graph snapshots) is given to the model, although ELAND-E2E can achieve a better performance than original EVOLVEGCN and ELAND-ITR, we find a decline on the performance. One possible reason is that EVOLVEGCN only utilizes the node representations from the last GRU-cell [6], which is the GCN [25] on the final snapshot. The model may lose contextual information learned from the early graphs and hence hurt the performance.

5.3 Case Studies

To further demonstrate the effectiveness of our proposed ELAND framework, we conduct case studies in the Weibo data. We study two user cases: (1) an anomalous user who was falsely classified as a benign user at early stage when data were not complete, and (2) a benign user who was falsely classified as an anomaly at early stage. Figure 4 presents their historical posts, predicted posts (given by ELAND-E2E), and actual future posts.

For the anomalous user, GRAPHSAGE originally classified him as a benign user with confidence of 0.73. ELAND-E2E successfully classified him as an anomaly with confidence of 0.58. From the posts of this user we observe that the user was posting advertisements in early posts. However, the posts did not repeat or show any strong pattern of an anomaly. ELAND-E2E correctly predicted that he will repeat posting the message *A* and hence enforced his suspicious pattern. The ground truth showed that this user actually did continue to repeat the advertisement posts (e.g., repeat posting message *A*). We verified that this user was an anomaly by checking the complete history of his posts and we found the history had a large number of advertisements.

For the other user, GRAPHSAGE falsely predicted this benign user as an anomaly with confidence of 0.73. With ELAND-E2E, this user was correctly classified as a benign user with confidence of 0.74. One possible reason is that this user mentioned about cell phones in his early posts and there were a large number of advertisement posts about phones. ELAND-E2E predicted that he would be posting some unrelated posts, one of which was related with iPhone. Although the predicted future posts were not exactly the same as the actual future posts, the predicted posts showed that the behavior pattern of this user would not be like that of an anomalous user. We verified this user was benign. Most of his posts were about personal ideas. His posts which supported Samsung phones caused the false positive classified by GRAPHSAGE.

6 CONCLUSION

In this work, we proposed a novel early anomaly detection framework ELAND that consistently improved the existing graph anomaly detection methods in the task of early anomaly detection when graph is *temporally* incomplete. In summary, our work managed to model both the node representation and graph topology evolution through behavior forecasting via behavioral sequence modelling and prediction. The experiment results based on the real world dataset demonstrated the effectiveness of proposed ELAND framework on improving the performance of existing graph anomaly detection methods and solving the task of early anomaly detection.

REFERENCES

[1] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion fraud detection in online reviews by network effects. In *AAAI*.
[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* (2003).
[3] Alexandre Bovet and Hernán A Makse. 2019. Influence of fake news in Twitter during the 2016 US presidential election. *Nature communications* 10, 1 (2019).
[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[5] Lesly Alejandra Gonzalez Camacho and Solange Nice Alves-Souza. 2018. Social network data to alleviate cold-start in recommender system: A systematic review. *Information Processing & Management* (2018).

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[7] John CB Cooper. 2005. The poisson and exponential distributions. *Mathematical Spectrum* 37, 3 (2005), 123–125.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*. 3844–3852.

[9] Kaize Ding, Jundong Li, Nitin Agarwal, and Huan Liu. 2020. Inductive Anomaly Detection on Attributed Networks. In *IJCAI*.

[10] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep anomaly detection on attributed networks. In *SDM*. 594–602.

[11] Ben Poole Eric Jang, Shixiang Gu. 2017. Categorical Reparameterization with Gumbel Softmax. In *ICLR*.

[12] Dhivya Eswaran, Christos Faloutsos, Sudipto Guha, and Nina Mishra. 2018. Spotlight: Detecting anomalies in streaming graphs. In *KDD*. ACM, 1378–1386.

[13] Shaohua Fan, Chuan Shi, and Xiao Wang. 2018. Abnormal event detection via heterogeneous information network embedding. In *CIKM*.

[14] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *KDD*. 1416–1424.

[15] Ellen Giebels. 2008. Influence: science and practice. *Tijdschrift conflicthantering* (2008), 14–15.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.

[17] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[19] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *KDD*. 895–904.

[20] Ruo Huang, Shelby McIntyre, Meina Song, Zhonghong Ou, et al. 2018. An attention-based recommender system to predict contextual intent based on choice histories across and within sessions. *Applied Sciences* 8, 12 (2018), 2426.

[21] Meng Jiang, Alex Beutel, Peng Cui, Bryan Hooi, Shiqiang Yang, and Christos Faloutsos. 2016. Spotting suspicious behaviors in multimodal data: A general metric and algorithms. *IEEE Transactions on Knowledge and Data Engineering* 28, 8 (2016), 2187–2200.

[22] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014. Catchsync: catching synchronized behavior in large directed graphs. In *KDD*.

[23] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2016. Inferring lockstep behavior from connectivity pattern in large graphs. *Knowledge and Information Systems* 48, 2 (2016), 399–428.

[24] Meng Jiang, Peng Cui, and Christos Faloutsos. 2016. Suspicious Behavior Detection: Current Trend and Future Directions. *IEEE Computer Society* (2016).

[25] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[26] Stefan Kombrink, Tomáš Mikolov, Martin Karafiát, and Lukáš Burget. 2011. Recurrent neural network based language modeling in meeting recognition. In *Twelfth annual conference of the international speech communication association*.

[27] Yu Kong, Zhiqiang Tao, and Yun Fu. 2017. Deep sequential context networks for action prediction. In *CVPR*. 1473–1481.

[28] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: Fraudulent user prediction in rating platforms. In *WSDM*. 333–341.

[29] Srijan Kumar and Neil Shah. 2018. False information on web and social media: A survey. *arXiv preprint arXiv:1804.08559* (2018).

[30] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing* 67, 1 (2018), 97–109.

[31] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In *ICCV*. 9267–9276.

[32] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive graph convolutional neural networks. In *AAAI*.

[33] Jun Liu, Amir Shahroudy, Gang Wang, Ling-Yu Duan, and Alex C Kot. 2018. SSNet: scale selection network for online 3D action prediction. In *CVPR*. 8349–8358.

[34] Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming graph neural networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 719–728.

[35] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*.

[36] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognition* 97 (2020), 107000.

[37] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*. 5115–5124.

[38] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*. 2014–2023.

[39] Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. 2007. NetProbe: A Fast and Scalable System for Fraud Detection in Online Auction Networks. *International World Wide Web Conference (WWW)* (2007).

[40] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B Schardl, and Charles E Leiserson. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs.. In *AAAI*. 5363–5370.

[41] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

[42] B Aditya Prakash, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. 2010. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 435–448.

[43] Kiran Rama, Pradeep Kumar, and Bharat Bhasker. 2019. Deep Learning to Address Candidate Generation and Cold Start Challenges in Recommender Systems: A Research Survey. *arXiv preprint arXiv:1907.08674* (2019).

[44] Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *KDD*. 985–994.

[45] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*. Springer.

[46] Neil Shah. 2017. Flock: Combating astroturfing on livestreaming platforms. In *Proceedings of the 26th International Conference on World Wide Web*. 1083–1091.

[47] Neil Shah, Alex Beutel, Brian Gallagher, and Christos Faloutsos. 2014. Spotting suspicious link behavior with fbox: An adversarial perspective. In *ICDM*.

[48] Kijung Shin, Bryan Hooi, and Christos Faloutsos. 2016. M-zoom: Fast dense-block detection in tensors with quality guarantees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 264–280.

[49] Kai Shu and Huan Liu. 2019. Detecting fake news on social media. *Synthesis Lectures on Data Mining and Knowledge Discovery* (2019).

[50] Kai Shu, Guoqing Zheng, Yichuan Li, Subhabrata Mukherjee, Ahmed Hassan Awadallah, Scott Ruston, and Huan Liu. 2020. Leveraging Multi-Source Weak Social Supervision for Early Detection of Fake News. *arXiv preprint arXiv:2004.01732* (2020).

[51] Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. 2005. Neighborhood formation and anomaly detection in bipartite graphs. In *ICDM*.

[52] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

[53] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NeurIPS*.

[54] Lin Tian, Xiuzhen Zhang, Yan Wang, and Huan Liu. 2020. Early detection of rumours on Twitter via stance transfer learning. In *European Conference on Information Retrieval*.

[55] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[56] Wikipedia contributors. 2020. Wechat— Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/WeChat [Online; accessed 14-June-2020].

[57] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536* (2018).

[58] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*. 974–983.

[59] Chengxi Zang, Peng Cui, Wenwu Zhu, and Fei Wang. 2019. Dynamical Origins of Distribution Functions. In *KDD*. 469–478.

[60] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD*.

[61] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Nguyen Hung, Zi Huang, and Lizhen Cui. 2020. GCN-Based User Representation Learning for Unifying Robust Recommendation and Fraudster Detection. *SIGIR* (2020).

[62] Tong Zhao, Chuchen Deng, Kaifeng Yu, Tianwen Jiang, Daheng Wang, and Meng Jiang. 2020. Error-bounded Graph Anomaly Loss for GNNs. In *CIKM*.

[63] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2020. Data Augmentation for Graph Neural Networks. *arXiv preprint arXiv:2006.06830* (2020).

[64] Tong Zhao, Matthew Malir, and Meng Jiang. 2018. Actionable objective optimization for suspicious behavior detection on large bipartite graphs. In *IEEE International Conference on Big Data*.