One-Class Adversarial Nets for Fraud Detection

Panpan Zheng
University of Arkansas
pzheng@uark.edu

Shuhan Yuan
University of Arkansas
sy005@uark.edu

Xintao Wu
University of Arkansas
xintaowu@uark.edu

Jun Li
University of Oregon
lijun@cs.uoregon.edu

Aidong Lu
University of North Carolina at Charlotte
aidong.lu@uncc.edu

ABSTRACT

Many online applications, such as online social networks or knowledge bases, are often attacked by malicious users who commit different types of actions such as vandalism on Wikipedia or fraudulent reviews on eBay. Currently, most of the fraud detection approaches require a training dataset that contains records of both benign and malicious users. However, in practice, there are often no or very few records of malicious users. In this paper, we develop one-class adversarial nets (OCAN) for fraud detection using training data with only benign users. OCAN first uses LSTM-Autoencoder to learn the representations of benign users from their sequences of online activities. It then detects malicious users by training a discriminator with a complementary GAN model that is different from the regular GAN model. Experimental results show that our OCAN outperforms the state-of-the-art one-class classification models and achieves comparable performance with the latest multi-source LSTM model that requires both benign and malicious users in the training phase.

KEYWORDS

fraud detection, one-class classification, generative adversarial nets

1 INTRODUCTION

Online platforms such as online social networks (OSNs) and knowledge bases help play a major role in online communication and knowledge sharing. However, there are various malicious users who conduct various fraudulent actions, such as spams, rumors, and vandalism, imposing severe security threats to OSNs and their legitimate participants. For example, the crowdsourcing mechanism of Wikipedia attracts the attention of vandals who involve in various vandalism actions like spreading false or misleading information to Wikipedia users. Meanwhile, fraudsters in the OSNs can also easily register fake accounts, inject fake content, or take fraudulent activities. To protect legitimate users, most Web platforms have tools or mechanisms to block malicious users. For example, Wikipedia adopts ClueBot NG [8] to detect and revert obvious bad edits, thus helping administrators to identify and block vandals.

Detecting malicious users has also attracted increasing attention in the research community [6, 18, 19, 37, 38]. For example, research in [19] focused on predicting whether a Wikipedia user is a vandal based on his edits. The VEWS system adopted a set of behavior features based on user edit-patterns, and used several traditional classifiers (e.g., random forest or SVM) to detect vandals. To improve detection accuracy and avoid manual feature reconstruction, a multi-source long-short term memory network (M-LSTM) was proposed to detect vandals [39]. M-LSTM is able to capture different aspects of user edits and the learned user representations can be further used to analyze user behaviors. However, these detection models are trained over a training dataset that consists of both positive data (benign users) and negative data (malicious users). In practice, there are often no or very few records from malicious users in the collected training data.

In this work, we tackle the problem of identifying malicious users when only benign users are observed in the training dataset. We develop one-class adversarial nets (OCAN) for fraud detection. During training, OCAN contains two phases. First, OCAN adopts the LSTM-Autoencoder [32] to encode the benign users into a hidden space based on their online activities, and the encoded vectors are called benign user representations. Then, OCAN trains a classifier over a combination of the benign user representations from the LSTM-Autoencoder and the complementary samples generated by an improved generative adversarial nets. The regular generative adversarial nets (GAN) consist of two neural networks, a discriminator and a generator [11]. The generator of a regular GAN model is trained to generate fake samples which match the distribution of real data, while the discriminator of a regular GAN is trained to identify whether the samples are real or fake. However, in our scenario, the generated fake samples could be very different from malicious users. Hence, the discriminator of the regular GAN does not have high confidence on separating the benign and malicious users. To address this issue, we adopt the idea of "bad" GAN, where the generator is trained to generate complementary samples instead of matching the original data distribution [9]. In particular, we propose a complementary GAN model, where the generator aims to generate samples that are complementary to the representations of benign users. Meanwhile, the discriminator is trained to separate benign users and complementary samples. Since the behaviors of malicious users and that of benign users are complementary, the discriminator is able to more accurately detect malicious users. By combining the encoder of LSTM-Autoencoder and the discriminator of the complementary GAN, OCAN can accurately predict whether a new user is benign or malicious based on his online activities.

The advantages of OCAN for fraud detection are as follows. First, since OCAN does not require any information about malicious users, we do not need to manually compose a mixed training dataset, thus more adaptive to different types of malicious user identification tasks. Second, different from existing one-class classification models, OCAN generates complementary samples of benign users and trains the discriminator to separate complementary samples from benign users, enabling the trained discriminator to better separate malicious users from benign users. Third, OCAN can capture the sequential information of user activities. After training, the detection model can adaptively update a user representation once the user commits a new action and predict whether the user is a fraud or not dynamically.

2 RELATED WORK

Fraud detection: Due to the openness and anonymity of Internet, the online platforms attract a large number of malicious users, such as vandals, trolls, and sockpuppets. Many fraud detection techniques have been developed in recent years [1, 4, 15, 37], including content-based approaches and graph-based approaches. The content-based approaches extract content features, (i.e., text, URL), to identify malicious users from user activities on social networks [2]. Meanwhile, graph-based approaches identify frauds based on network topologies. Research in [38] proposed two deep neural networks for fraud detection on a signed graph. Often based on unsupervised learning, the graph-based approaches consider fraud as anomalies and extract various graph features associated with nodes, edges, ego-net, or communities from the graph [1, 26].

Fraud detection is also related to the malicious behavior and misinformation detection, including detecting the vandalism edits on Wikipedia or Wikidata, rumor and fake review detection. Research in [13] developed both content and context features of a Wikidata revision to identify the vandalism edit. Research in [20] focused on detecting hoaxes on Wikipedia by finding characteristic in terms of article content and features of the editor who created the hoax. In [25], different types of behavior features were extracted and used to detect the fake reviews on Yelp. Research in [23] have identified several representative behaviors of review spammers. Research in [36] studied the co-anomaly patterns in multiple review-based time series.

Deep neural network: Deep neural networks have achieved promising results in computer vision, natural language processing, and speech recognition [21]. Recurrent neural network as one type of deep neural networks is widely used for modeling time sequence data [12]. However, it is difficult to train standard RNNs over long sequences of text because of gradient vanishing and exploding [3]. Long shot-term Memory (LSTM) [14] was proposed to model temporal sequences and capture their long-range dependencies more accurately than the standard RNNs. LSTM-Autoencoder is a sequence-to-sequence model that has been widely used for paragraph generation [22], video representation [32], etc. GAN is a framework for estimating generative models via an adversarial process [11]. Recently, generative adversarial nets (GAN) have achieved great success in computer vision tasks, including image generation [28, 31] and image classification [5, 31]. Currently, the GAN model is usually applied on two or multi-class datasets instead of one-class.

One-class classification: One-class classification (OCC) algorithms aim to build classification models when only one class of samples are observed and the other class of samples are absent [17], which is also related to the novelty detection [27]. One-class support vector machine (OSVM), as one of widely adopted for one class classification, aims to separate one class of samples from all the others by constructing a hyper-sphere around the observed data samples [24, 34]. Other traditional classification models also extend to the one-class scenario. For example, one-class nearest neighbor (OCNN) [33] predicts the class of a sample based on its distance to its nearest neighbor in the training dataset. One-class Gaussian process (OCGP) chooses a proper GP prior and derives membership scores for one-class classification [16]. However, OCNN and OCGP need to set a threshold to detect another class of data. The threshold is either set by a domain expert or tuned based on a small set of two-class labeled data. In this work, we propose a framework that combines LSTM-Autoencoder and GAN to detect vandals with only knowing benign users. To our best knowledge, this is the first work that examines the use of deep learning models for fraud detection when only one-class training data is available. Meanwhile, comparing to existing one-class algorithms, our model trains a classifier by generating a large number of "novel" data and does not require any labeled data to tune parameters.

3 PRELIMINARY

3.1 Long Short-Term Memory Network

Long short-term memory network (LSTM) is one type of recurrent neural network. Given a sequence $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ where $\mathbf{x}_t \in \mathbb{R}^d$ denotes the input at the t-th step, LSTM maintains a hidden state vector $\mathbf{h}_t \in \mathbb{R}^h$ to keep track the sequence information from the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. The hidden state $\mathbf{h}_t$ is computed by

$$\begin{aligned}
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c), \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i), \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f), \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o), \\
\mathbf{c}_t &= \mathbf{i}_t \odot \tilde{\mathbf{c}}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1}, \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned} \tag{1}$$

where σ is the sigmoid function; $\odot$ represents element-wise product; $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$ indicate the input gate, forget gate, output gate, and cell activation vectors and $\tilde{\mathbf{c}}_t$ denotes the intermediate vector of cell state; $\mathbf{W}$ and $\mathbf{U}$ are the weight parameters; $\mathbf{b}$ is the bias term.

We simplify the update of each LSTM step described in Equation 1 as

$$\mathbf{h}_t = LSTM(\mathbf{x}_t, \mathbf{h}_{t-1}), \tag{2}$$

where $\mathbf{x}_t$ is the input of the current step; $\mathbf{h}_{t-1}$ is the hidden vector of the last step; $\mathbf{h}_t$ indicates the output of the current step.

3.2 Generative Adversarial Nets

Generative adversarial nets (GAN) are generative models that consist of two components: a generator G and a discriminator D. Typically, both G and D are multilayer neural networks. $G(\mathbf{z})$ generates fake samples from a prior $p_{\mathbf{z}}$ on a noise variable $\mathbf{z}$ and learns a generative distribution p_G to match the real data distribution p_{data}. On the contrary, the discriminative model D is a binary classifier that predicts whether an input is a real data $\mathbf{x}$ or a generated fake data from $G(\mathbf{z})$. Hence, the objective function of D is defined as:

$$\max_D \quad \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))], \tag{3}$$

where $D(\cdot)$ outputs the probability that $\cdot$ is from the real data rather than the generated fake data. In order to make the generative distribution p_G close to the real data distribution p_{data}, G is trained by fooling the discriminator not be able to distinguish the generated data from the real data. Thus, the objective function of G is defined as:

$$\min_{G} \quad \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]. \tag{4}$$

Minimizing the Equation 4 is achieved if the discriminator is fooled by generated data $G(\mathbf{z})$ and predicts high probability that $G(\mathbf{z})$ is real data.

Overall, GAN is formalized as a minimax game $\min_{G} \max_{D} V(G, D)$ with the value function:

$$V(G, D) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]. \tag{5}$$

Theoretical analysis shows that GAN aims to minimize the Jensen-Shannon divergence between the data distribution p_{data} and the generative distribution p_G [11]. The minimization of JS divergence is achieved when $p_G = p_{\text{data}}$. Therefore, GAN is trained by distinguishing the real data and generated fake data.

4 OCAN: ONE-CLASS ADVERSARIAL NETS

4.1 Framework Overview

OCAN contains two phases during training. The first phase is to learn user representations. As shown in the left side of Figure 1, LSTM-Autoencoder is adopted to learn the benign user representations from the benign user activity sequences. The LSTM-Autoencoder model is a sequence-to-sequence model that consists of two LSTM models as the encoder and decoder respectively. The encoder computes the hidden representation of an input, and the decoder computes the reconstructed input based on the hidden representation. The objective function of the LSTM-Autoencoder is to make the reconstructed input close to the original input. In our scenario, the LSTM-Autoencoder is trained by only benign users. However, it successfully captures the salient information of users' activity sequences. The encoder of the trained LSTM-Autoencoder, when deployed for fraud detection, expects to be able to map the benign users and malicious users to relatively different regions in the continuous feature space because the activity sequences of benign and malicious users are different.

The second phase is to train a classifier to detect malicious users with only observed benign users. The basic idea of OCAN is to generate potential malicious users and train a classifier in a supervised manner. Especially, we will use the discriminator from the trained complementary GAN as the classifier. In our OCAN, we generate complementary samples that are in the low-density area of benign users, and train the discriminator to separate the real and complementary benign users. Thus, the classification boundary of the discriminator is around the high-density area of real benign users. The discriminator then has the ability to detect malicious users which locate in different regions from benign users. The framework of training complementary GAN for fraud detection is shown in the right side of Figure 1.

The pseudo-code of training OCAN is shown in Algorithm 1. Given a training dataset M_{benign} that contains activity sequence feature vectors of N benign users, we first train the LSTM-Autoencoder model (Lines 3–9). After training the LSTM-Autoencoder, we adopt the encoder in the LSTM-Autoencoder model to compute the benign user representation (Lines 11–14). Finally, we use the benign user representation to train the complementary GAN (Lines 16–20). For simplicity, we write the algorithm with a minibatch size of 1, i.e., iterating each user in the training dataset to train LSTM-Autoencoder and GAN. In practice, we sample m real benign users and use the generator to generate m complementary samples in a minibatch. In our experiments, the size of minibatch is 32.

Our OCAN moves beyond the naive approach of adopting a regular GAN model in the second phase. The generator of a regular GAN aims to generate the representations of fake benign users that are close to the representations of real benign users. The discriminator of a regular GAN is to identify whether an input is a representation of a real benign user or a fake benign user from the generator. However, one potential drawback of the regular GAN is that once the discriminator is converged, the discriminator cannot have high confidence on separating real benign users from real malicious users. We denote the OCAN with the regular GAN as OCAN-r and compare its performance with OCAN in the experiment.

Algorithm 1: Training One-Class Adversarial Nets

```
Inputs  : Training dataset $M_{\text{benign}} = \{\mathcal{X}_1, \cdots, \mathcal{X}_N\}$,
          Training epochs for LSTM-Autoencoder
          $Epoch_{AE}$ and GAN $Epoch_{GAN}$
Outputs : Well-trained LSTM-Autoencoder and GAN
1  initialize parameters in LSTM-Autoencoder and GAN;
2  $j \leftarrow 0$;
3  while $j < Epoch_{AE}$ do
4      foreach user $u$ in $M_{benign}$ do
5          compute the reconstructed sequence of user activities
             by LSTM-Autoencoder (Eq. 6, 8, and 9);
6          optimize the parameters in LSTM-Autoencoder with
             the loss function Eq. 10;
7      end
8      $j \leftarrow j + 1$;
9  end
10 $\mathcal{V} = \emptyset$;
11 foreach user $u$ in $M_{benign}$ do
12     compute the benign user representation $\mathbf{v}_u$ by the encoder
         of LSTM-Autoencoder (Eq. 6, 7);
13     $\mathcal{V} += \mathbf{v}_u$;
14 end
15 $j \leftarrow 0$;
16 while $j < Epoch_{GAN}$ do
17     foreach benign user representation $\mathbf{v}_u$ in $\mathcal{V}$ do
18         optimize the discriminator $D$ and generator $G$ with
             loss functions Eq. 16, 14, respectively;
19     end
20 end
21 return well-trained LSTM-Autoencoder and GAN
```

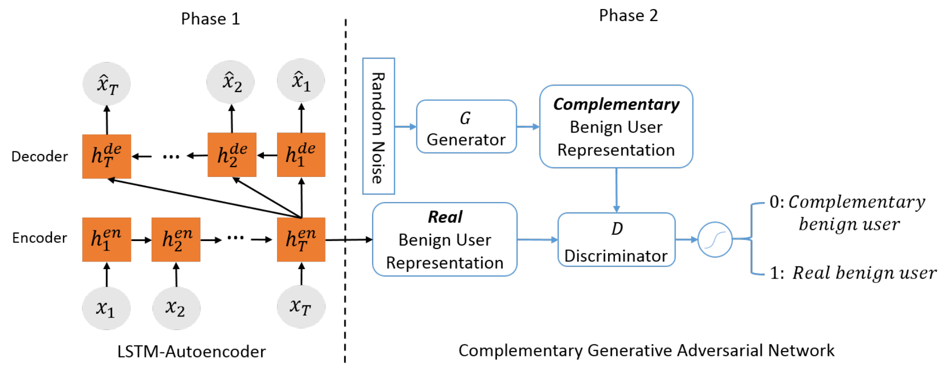


Figure 1: The training framework of OCAN

4.2 LSTM-Autoencoder for User Representation

The first phase of OCAN is to encode users to a continuous hidden space. Since each online user has a sequence of activities (e.g., edit a sequence of pages), we adopt LSTM-Autoencoder to transform a variable-length user activity sequence into a fixed-dimension user representation. Formally, given a user u with T activities, we represent the activity sequence as $\mathcal{X}_u = (\mathbf{x}_1, \dots, \mathbf{x}_t, \dots, \mathbf{x}_T)$ where $\mathbf{x}_t \in \mathbb{R}^d$ is the t-th activity feature vector.

Encoder: The encoder encodes the user activity sequence $\mathcal{X}_u$ to a user representation with an LSTM model:

$$\mathbf{h}_t^{en} = LSTM^{en}(\mathbf{x}_t, \mathbf{h}_{t-1}^{en}), \tag{6}$$

where $\mathbf{x}_t$ is the feature vector of the t-th activity; $\mathbf{h}_t^{en}$ indicates the t-th hidden vector of the encoder.

The last hidden vector $\mathbf{h}_T^{en}$ captures the information of a whole user activity sequence and is considered as the user representation $\mathbf{v}$:

$$\mathbf{v} = \mathbf{h}_T^{en}. \tag{7}$$

Decoder: In our model, the decoder adopts the user representation $\mathbf{v}$ as the input to reconstruct the original user activity sequence $\mathcal{X}$:

$$\mathbf{h}_t^{de} = LSTM^{de}(\mathbf{v}, \mathbf{h}_{t-1}^{de}), \tag{8}$$

$$\hat{\mathbf{x}}_t = f(\mathbf{h}_t^{de}), \tag{9}$$

where $\mathbf{h}_t^{de}$ is the t-th hidden vector of the decoder; $\hat{\mathbf{x}}_t$ indicates the t-th reconstructed activity feature vector; $f(\cdot)$ denotes a neural network to compute the sequence outputs from hidden vectors of the decoder. Note that we adopt $\mathbf{v}$ as input of the whole sequence of the decoder, which has achieved great performance on sequence-to-sequence models [7].

The objective function of LSTM-Autoencoder is:

$$\mathcal{L}_{(AE)}(\hat{\mathbf{x}}_t, \mathbf{x}_t) = \sum_{t=1}^{T} (\hat{\mathbf{x}}_t - \mathbf{x}_t)^2, \tag{10}$$

where $\mathbf{x}_t$ ($\hat{\mathbf{x}}_t$) is the t-th (reconstructed) activity feature vector. After training, the last hidden vector of encoder $\mathbf{h}_T$ can reconstruct the sequence of user feature vectors. Thus, the representation of user $\mathbf{v} = \mathbf{h}_T^{en}$ captures the salient information of user behavior.

4.3 Complementary GAN

The generator G of complementary GAN is a feedforward neural network where its output layer has the same dimension as the user representation $\mathbf{v}$. Formally, we define the generated samples as $\tilde{\mathbf{v}} = G(\mathbf{z})$. Unlike the generator in a regular GAN which is trained to match the distribution of the generated fake benign user representation with that of benign user representation p_{data}, the generator G of complementary GAN learns a generative distribution p_G that is close to the complementary distribution p^* of the benign user representations, i.e., $p_G = p^*$. The complementary distribution p^* is defined as:

$$p^*(\tilde{\mathbf{v}}) = \begin{cases} \frac{1}{\tau} \frac{1}{p_{\text{data}}(\tilde{\mathbf{v}})} & \text{if } p_{\text{data}}(\tilde{\mathbf{v}}) > \epsilon \text{ and } \tilde{\mathbf{v}} \in \mathcal{B}_{\mathbf{v}} \\ C & \text{if } p_{\text{data}}(\tilde{\mathbf{v}}) \le \epsilon \text{ and } \tilde{\mathbf{v}} \in \mathcal{B}_{\mathbf{v}}, \end{cases} \tag{11}$$

where ϵ is a threshold to indicate whether the generated samples are in high-density regions; τ is a normalization term; C is a small constant; $\mathcal{B}_{\mathbf{v}}$ is the space of user representation. To make the generative distribution p_G close to the complementary distribution p^*, the complementary generator G is trained to minimize the KL divergence between p_G and p^*. Based on the definition of KL divergence, we have the following objective function:

$$\begin{aligned} \mathcal{L}_{KL(p_G \| p^*)} &= -\mathcal{H}(p_G) - \mathbb{E}_{\tilde{\mathbf{v}} \sim p_G} \log p^*(\tilde{\mathbf{v}}) \\ &= -\mathcal{H}(p_G) + \mathbb{E}_{\tilde{\mathbf{v}} \sim p_G} \log p_{\text{data}}(\tilde{\mathbf{v}}) \mathbb{1}[p_{\text{data}}(\tilde{\mathbf{v}}) > \epsilon] \\ &\quad + \mathbb{E}_{\tilde{\mathbf{v}} \sim p_G} (\mathbb{1}[p_{\text{data}}(\tilde{\mathbf{v}}) > \epsilon] \log \tau - \mathbb{1}[p_{\text{data}}(\tilde{\mathbf{v}}) \le \epsilon] \log C), \end{aligned} \tag{12}$$

where $\mathcal{H}(\cdot)$ is the entropy, and $\mathbb{1}[]$ is the indicator function. The last term of Equation 12 can be omitted because both τ and C are constant terms and the gradients of the indicator function $\mathbb{1}[\cdot]$ with respect to parameters of the generator are mostly zero.

Meanwhile, the complementary generator G adopts the feature matching loss [29] to ensure that the generated samples are constrained in the space of user representation $\mathcal{B}_{\mathbf{v}}$.

$$\mathcal{L}_{\text{fm}} = \| \mathbb{E}_{\tilde{\mathbf{v}} \sim p_G} f(\tilde{\mathbf{v}}) - \mathbb{E}_{\mathbf{v} \sim p_{\text{data}}} f(\mathbf{v}) \|^2, \tag{13}$$

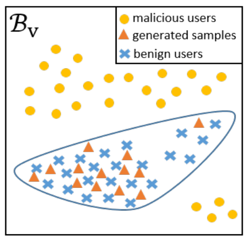
where $f(\cdot)$ denotes the output of an intermediate layer of the discriminator used as a feature representation of $\mathbf{v}$.

Thus, the complete objective function of the generator is defined as:

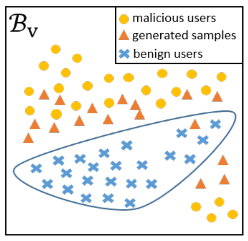
$$\min_G \quad -\mathcal{H}(p_G) + \mathbb{E}_{\tilde{\mathbf{v}}\sim p_G} \log p_{\text{data}}(\tilde{\mathbf{v}}) \mathbb{1}[p_{\text{data}}(\tilde{\mathbf{v}}) > \epsilon] \\ + \| \mathbb{E}_{\tilde{\mathbf{v}}\sim p_G} f(\tilde{\mathbf{v}}) - \mathbb{E}_{\mathbf{v}\sim p_{\text{data}}} f(\mathbf{v}) \|^2 . \tag{14}$$

Overall, the objective function of the complementary generator aims to let the generative distribution p_G close to the complementary samples p^*, i.e., $p_G = p^*$, and make the generated samples from different regions (but in the same space of user representations) than those of the benign users.

Figure 2 illustrates the difference of the generators of regular GAN and complementary GAN. The objective function of the generator of regular GAN in Equation 4 is trained to fool the discriminator by generating fake benign users similar to the real benign users. Hence, as shown in Figure 2a, the generator of regular GAN generates the distribution of fake benign users that have the same distribution of real benign users in the feature space. On the contrary, the objective function of the generator of complementary GAN in Equation 14 is trained to generate complementary samples that are in the low-density regions of benign users (shown in Figure 2b).



(a) Regular GAN



(b) Complementary GAN

Figure 2: Demonstrations of the ideal generators of regular GAN and complementary GAN. The blue line indicates the distribution regions of benign users.

To optimize the objective function of generator, we need to approximate the entropy of generated samples $\mathcal{H}(p_G)$ and the probability distribution of real samples p_{data}. To minimize $-\mathcal{H}(p_G)$, we adopt the pull-away term (PT) proposed by [9, 40] that encourages the generated feature vectors to be orthogonal. The PT term increases the diversity of generated samples and can be considered as a proxy for minimizing $-\mathcal{H}(p_G)$. The PT term is defined as

$$\mathcal{L}_{PT} = \frac{1}{N(N-1)} \sum_{i}^{N} \sum_{j \neq i}^{N} \left(\frac{f(\tilde{\mathbf{v}}_i)^T f(\tilde{\mathbf{v}}_j)}{\| f(\tilde{\mathbf{v}}_i) \|^2 \| f(\tilde{\mathbf{v}}_j) \|^2} \right), \tag{15}$$

where N is the size of a mini-batch. Approximating p_{data} is usually computationally expensive. We adopt the idea proposed by [30] that uses the discriminator of a pre-trained regular GAN as a proxy for evaluating $p_{\text{data}}(\tilde{\mathbf{v}}) > \epsilon$. The regular GAN is trained based on benign user representations. While the generator is trained to generate samples that are close to real benign users, the discriminator should detect whether a sample is from the real data distribution p_{data} or from the generator. Thus, the pre-trained discriminator is sufficient to identify the data points that are above a threshold of p_{data}.

The discriminator D takes the benign user representation $\mathbf{v}$ and generated user representation $\tilde{\mathbf{v}}$ as inputs and tries to distinguish $\mathbf{v}$ from $\tilde{\mathbf{v}}$. As a classifier, D is a standard feedforward neural network with a softmax function as its output layer, and the objective function of D is:

$$\max_D \quad \mathbb{E}_{\mathbf{v}\sim p_{\text{data}}}[\log D(\mathbf{v})] + \mathbb{E}_{\tilde{\mathbf{v}}\sim p_G}[\log(1 - D(\tilde{\mathbf{v}}))] + \\ \mathbb{E}_{\mathbf{v}\sim p_{\text{data}}}[D(\mathbf{v}) \log D(\mathbf{v})]. \tag{16}$$

The first two terms in Equation 16 are the objective function of discriminator in the regular GAN model. Therefore, the discriminator of complementary GAN is trained to separate the benign users and complementary samples. The last term in Equation 16 is a conditional entropy term which encourages the discriminator to detect real benign users with high confidence. Then, the discriminator is able to separate the benign and malicious users clearly.

Although the objective functions of the discriminators of regular GAN and complementary GAN are similar, the capabilities of discriminators of regular GAN and complementary GAN for malicious detection are different. The discriminator of regular GAN aims to separate the benign users and generated fake benign users. However, after training, the generated fake benign users locate in the same regions as the real benign users (shown in Figure 2a). The probabilities of real and generated fake benign users predicted by the discriminator of regular GAN are all close to 0.5. Thus, giving a benign user, the discriminator cannot predict the benign user with high confidence. On the contrary, the discriminator of complementary GAN is trained to separate the benign users and generated complementary samples. Since the generated complementary samples have the same distribution as the malicious users (shown in Figure 2b), the discriminator of complementary GAN can also detect the malicious users.

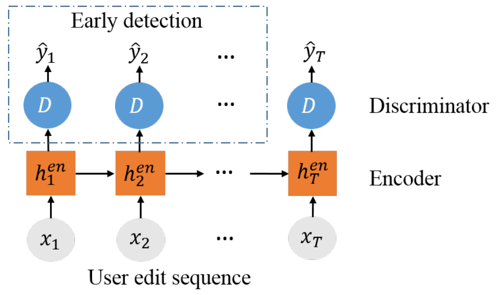


Figure 3: The fraud detection model

5 FRAUD DETECTION MODEL

Although the training procedure of OCAN contains two phases that train LSTM-Autoencoder and complementary GAN successively, the fraud detection model is an end-to-end model. We show the procedure pseudocode of detecting fraud in Algorithm 2 and illustrate its structure in Figure 3. To detect a malicious user, we first compute the user representation $\mathbf{v}_u$ based on the encoder in the LSTM-Autoencoder model (Line 3). Then, we predict the user label based on the discriminator of complementary GAN, i.e., $p(\hat{y}_u|\mathbf{v}_u) = D(\mathbf{v}_u)$.

Algorithm 2: Fraud Detection

```
Inputs  : Testing dataset $M_{\text{test}} = \{\mathcal{X}_1, \cdots, \mathcal{X}_N\}$,
          Well-trained LSTM-Autoencoder and GAN
Outputs : the user labels $\mathcal{Y}$ in $M_{\text{test}}$
1 $\mathcal{Y} = \emptyset$;
2 foreach user $u$ in $M_{test}$ do
3     compute the user representation $\mathbf{v}_u$ by the encoder in
        LSTM-Autoencoder (Eq. 6, 7);
4     predict the label $\hat{y}_u$ of the user by $D(\mathbf{v}_u)$
5     $\mathcal{Y} += \hat{y}_u$
6 end
7 return the user labels $\mathcal{Y}$
```

Early fraud detection: The upper-left region of Figure 3 shows that our OCAN model can also achieve early detection of malicious users. Given a user u, at each step t, the hidden states $\mathbf{h}^{en}_{u_t}$ are updated until the t-th step by taking the current feature vector $\mathbf{x}_{u_t}$ as input and are able to capture the user behavior information until the t-th step. Thus, the user representation at the t-th step is denoted as $\mathbf{v}_{u_t} = \mathbf{h}^{en}_{u_t}$. Finally, we can use the discriminator D to calculate the probability $p(\hat{y}_{u_t}|\mathbf{v}_{u_t}) = D(\mathbf{v}_{u_t})$ of the user to be a malicious user based on the current step user representation $\mathbf{v}_t$.

6 EXPERIMENTS

6.1 Experiment Setup

Dataset: To evaluate OCAN, we focus on one type of malicious users, i.e., vandals on Wikipedia. We conduct our evaluation on UMDWikipedia dataset [19]. This dataset contains information of around 770K edits from Jan 2013 to July 2014 (19 months) with 17105 vandals and 17105 benign users. Each user edits a sequence of Wikipedia pages. We keep those users with the lengths of edit sequence range from 4 to 50. After this preprocessing, the dataset contains 10528 benign users and 11495 vandals.

To compose the feature vector $\mathbf{x}_t$ of the user's t-th edit, we adopt the following edit features: (1) whether or not the user edited on a meta-page; (2) whether or not the user consecutively edited the pages less than 1 minutes; (3) whether or not the user's current edit page had been edited before; (4) whether or not the user's current edit would be reverted.

We further evaluate our model on a credit card transaction dataset in Section 6.5. Although it is not a sequence dataset, it can still be used to compare the performance of OCAN against baselines in the context of one-class fraud detection.

Hyperparameters: For LSTM-Autoencoder, the dimension of the hidden layer is 200, and the training epoch is 20. For the complementary GAN model, both discriminator and generator are feedforward neural networks. Specifically, the discriminator contains 2 hidden layers which are 100 and 50 dimensions. The generator takes the 50 dimensions of noise as input, and there is one hidden layer with 100 dimensions. The output layer of the generator has the same dimension as the user representation which is 200 in our experiments. The training epoch of complementary GAN is 50. The threshold ϵ defined in Equation 14 is set as the 5-quantile probability of real benign users predicted by a pre-trained discriminator. We evaluated several values from 4-quantile to 10-quantile and found the results are not sensitive.

Repeatability: Our software together with the datasets used in this paper are available at https://github.com/PanpanZheng/OCAN

6.2 Comparison with One-Class Classification

Baselines: We compare OCAN with the following widely used one-class classification approaches:

- One-class nearest neighbors (**OCNN**) [33] labels a testing sample based on the distance from the sample to its nearest neighbors in training dataset and the average distance of those nearest neighbors. If the difference between these two distances is larger than a threshold, the testing sample is an anomaly.
- One-class Gaussian Processes (**OCGP**) [16] is a one-class classification model based on Gaussian process regression.
- One-class SVM (**OCSVM**) [34] adopts support vector machine to learn a decision hypersphere around the positive data, and considers samples located outside this hypersphere as anomalies.

Note that both OCNN and OCGP require a small portion (5% in our experiments) of vandals as a validation dataset to tune an appropriate threshold for vandal detection. However, OCAN does not require any vandals for training and validation. Since the baselines are not sequence models, we compare OCAN to baselines in two ways. First, we concatenate all the edit feature vectors of a user to a *raw feature vector* as an input to baselines. Second, the baselines have the same inputs as the discriminator, i.e., the *user representation* $\mathbf{v}$ computed from the encoder of LSTM-Autoencoder. Meanwhile, OCAN cannot adopt the raw feature vectors as inputs to detect vandals. This is because GAN is only suitable for real-valued data [11].

To evaluate the performance of vandal detection, we randomly select 7000 benign users as the training dataset and 3000 benign users and 3000 vandals as the testing dataset. We report the mean value and standard deviation based on 10 different runs. Table 1 shows the means and standard deviations of the precision, recall, F1 score and accuracy for vandal detection. First, OCAN achieves better performances than baselines in terms of F1 score and accuracy in both input settings. It means the discriminator of complementary GAN can be used as a one-class classifier for vandal detection. We can further observe that when the baselines adopt the raw feature vector instead of user representation, the performances of both OCNN and OCGP decrease significantly. It indicates that the user representations computed by the encoder of LSTM-Autoencoder capture the salient information about user behavior and can improve the performance of one-class classifiers. However, we also notice that the standard deviations of OCAN are higher than the baselines with user representations as inputs. We argue that this is because GAN is widely known for difficult to train. Thus, the stability of OCAN is relatively lower than the baselines.

Furthermore, we show the experimental results of OCAN-r, which adopts the regular GAN model instead of the complementary GAN in the second training phase of OCAN, in the last row of Table 1. We can observe that the performance of OCAN is better than OCAN-r. It indicates that the discriminator of complementary GAN

Table 1: Vandal detection results (mean±std.) on precision, recall, F1 and accuracy

Input	Algorithm	Precision	Recall	F1	Accuracy
Raw feature vector	OCNN	0.5680 ± 0.0129	0.8646 ± 0.0599	0.6845 ± 0.0184	0.6027 ± 0.0161
	OCGP	0.5767 ± 0.0087	0.9000 ± 0.0560	0.7023 ± 0.0193	0.6196 ± 0.0142
	OCSVM	0.6631 ± 0.0057	0.9829 ± 0.0011	0.7919 ± 0.0040	0.7417 ± 0.0064
User representation	OCNN	0.8314 ± 0.0351	0.8028 ± 0.0476	0.8150 ± 0.0163	0.8181 ± 0.0153
	OCGP	0.8381±, 0.0225	0.8289 ± 0.0374	0.8326 ± 0.0158	0.8337 ± 0.0139
	OCSVM	0.6558 ± 0.0058	**0.9590 ± 0.0096**	0.7789 ± 0.0064	0.7278 ± 0.0080
	OCAN	**0.9067 ± 0.0615**	0.9292 ± 0.0348	**0.9010 ± 0.0228**	**0.8973 ± 0.0244**
User representation	OCAN-r	0.8673 ± 0.0355	0.8759 ± 0.0529	0.8701 ± 0.0267	0.8697 ± 0.0244

which is trained on real and complementary samples can more accurately separate the benign users and vandals.

Table 2: Early Vandal detection results on precision, recall, F1, and the average number of edits before the vandals are blocked

	Vandals	Precision	Recall	F1	Edits
M-LSTM	7000	0.8416	0.9637	0.8985	7.21
	1000	0.9189	0.8910	0.9047	5.98
	400	0.9639	0.6767	0.7951	3.64
	300	0.0000	0.0000	0.0000	0.00
	100	0.0000	0.0000	0.0000	0.00
OCAN	0	**0.8014**	**0.9081**	**0.8459**	**7.23**
OCAN-r	0	0.7228	0.8968	0.7874	7.18

6.3 Comparison with M-LSTM for Early Vandal Detection

We further compare the performance of OCAN in terms of early vandal detection with one latest deep learning based vandal detection model, M-LSTM, developed in [39]. Note that M-LSTM assumes a training dataset that contains both vandals and benign users. In our experiments, we train our OCAN with the training data consisting of 7000 benign users and no vandals and train M-LSTM with a training data consisting the same 7000 benign users and a varying number of vandals (from 7000 to 100). For OCAN and M-LSTM, we use the same testing dataset that contains 3000 benign users and 3000 vandals. Note that in OCAN and M-LSTM, the hidden state $\mathbf{h}_t^{en}$ of the LSTM model captures the up-to-date user behavior information and hence we can achieve early vandal detection. The difference is that the M-LSTM model uses $\mathbf{h}_t^{en}$ as the input of a classifier directly whereas OCAN further trains complementary GAN and uses its discriminator as a classifier to make the early vandal detection. In this experiment, instead of applying the classifier on the final user representation $\mathbf{v} = \mathbf{h}_T^{en}$, the classifiers of M-LSTM and OCAN are applied on each step of LSTM hidden state $\mathbf{h}_t^{en}$ and predict whether a user is a vandal after the user commits the t-th action.

Table 2 shows comparison results in terms of the precision, recall, F1 of early vandal detection, and the average number of edits before the vandals were truly blocked. We can observe that OCAN achieves a comparable performance as the M-LSTM when the number of vandals in the training dataset is large (1000, 4000, and 7000). However, M-LSTM has very poor accuracy when the number of vandals in the training dataset is small. In fact, we observe that M-LSTM could not detect any vandal when the training dataset contains less than 400 vandals. On the contrary, OCAN does not need any vandal in the training data.

The experimental results of OCAN-r for early vandal detection are shown in the last row of Table 2. OCAN-r outperforms M-LSTM when M-LSTM is trained on a small number of the training dataset. However, the OCAN-r is not as good as OCAN. It indicates that generating complementary samples to train the discriminator can improve the performance of the discriminator for vandal detection.

6.4 OCAN Framework Analysis

LSTM-Autoencoder: The reason of adopting LSTM-Autoencoder is that it transforms edit sequences to user representation. It also helps encode benign users and vandals to relatively different locations in the hidden space, although the LSTM-Autoencoder is only trained by benign users. To validate our intuition, we obtain user representations of the testing dataset by the encoder in the LSTM-Autoencoder. Then, we map those user representations to a two-dimensional space based on the Isomap approach [35]. Figure 4 shows the visualization of user representations. We observe that the benign users and vandals are relatively separated in the two-dimensional space, indicating the capability of LSTM-Autoencoder.

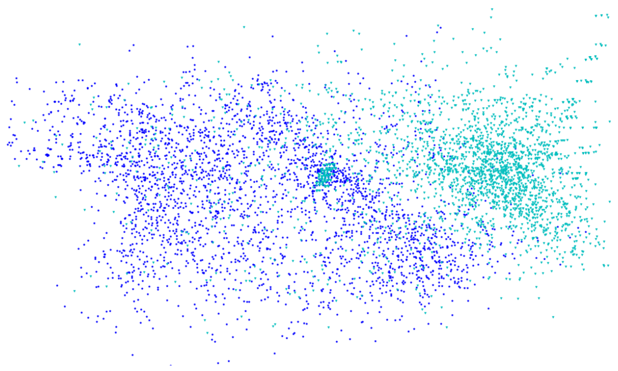
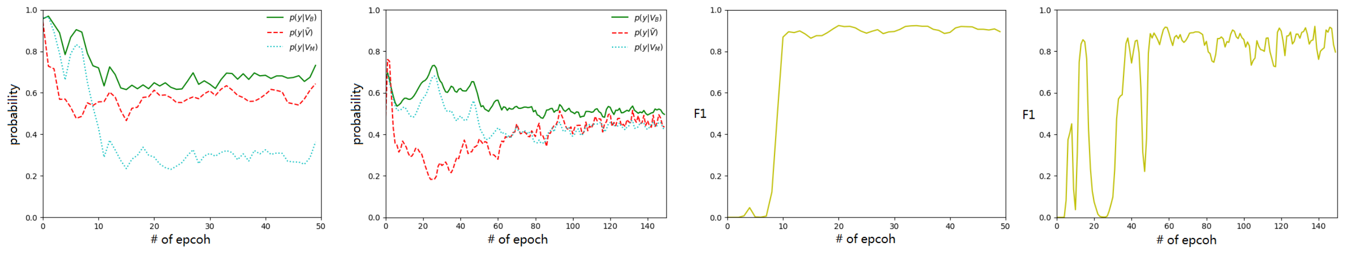


Figure 4: Visualization of 3000 benign users (blue star) and 3000 vandals (cyan triangle) based on user representation.

Complementary GAN vs. Regular GAN:



(a) Prob. predicted by OCAN **(b) Prob. predicted by OCAN-r** **(c) F1 score of OCAN** **(d) F1 score of OCAN-r**

Figure 5: Training progresses of OCAN (5a,5c) and OCAN-r(5b,5d). Three lines in Figures 5a and 5b indicate the probabilities of being benign users predicted by the discriminator: real benign users $p(y|v_B)$ (green line) vs. generated samples $p(y|\tilde{v})$ (red broken line) vs. real malicious users $p(y|v_M)$ (blue dotted line). Figures 5c and 5d show the F1 scores of OCAN and OCAN-r during training.

In our OCAN model, the generator of complementary GAN aims to generate complementary samples that lie in the low-density region of real samples, and the discriminator is trained to detect the real and complementary samples. We examine the training progress of OCAN in terms of predication accuracy. We calculate probabilities of real benign users $p(y|\mathbf{v}_B)$ (shown as green line in Figure 5a), malicious users $p(y|\mathbf{v}_M)$ (blue dotted line) and generated samples $p(y|\tilde{\mathbf{v}})$ (read broken line) being benign users predicted by the discriminator of complementary GAN on the testing dataset after each training epoch. We can observe that after OCAN is converged, the probabilities of malicious users predicted by the discriminator of complementary GAN are much lower than that of benign users. For example, at the epoch 40, the average probability of real benign users $p(y|\mathbf{v}_B)$ predicted by OCAN is around 70%, while the average probability of malicious users $p(y|\mathbf{v}_M)$ is only around 30%. Meanwhile, the average probability of generated complementary samples $p(y|\tilde{\mathbf{v}})$ lies between the probabilities of benign and malicious users.

On the contrary, the generator of a regular GAN in the OCAN-r model aims to generate fake samples that are close to real samples, and the discriminator of GAN focuses on distinguishing the real and generated fake samples. As shown in Figure 5b, the probabilities of real benign users and probabilities of malicious users predicted by the discriminator of regular GAN become close to each other during training. After the OCAN-r is converged, say epoch 120, both the probabilities of real benign users and malicious users are close to 0.5. Meanwhile, the probability of generated samples is similar to the probabilities of real benign users and malicious users.

We also show the F1 scores of OCAN and OCAN-r on the testing dataset after each training epoch in Figure 5c and 5d. We can observe that the F1 score of OCAN-r is not as stable as (and also a bit lower than) OCAN. This is because the outputs of the discriminator for real and fake samples are close to 0.5 after the regular GAN is converged. If the probabilities of real benign users predicted by the discriminator of the regular GAN swing around 0.5, the accuracy of vandal detection will fluctuate accordingly.

We can observe from Figure 5 another nice property of OCAN compared with OCAN-r for fraud detection, i.e., OCAN is converged faster than OCAN-r. We can observe that OCAN is converged with only training 20 epochs while the OCAN-r requires nearly 100 epochs to keep stable. This is because the complementary GAN is trained to separate the benign and malicious users while the regular GAN mainly aims to generate fake samples that match the real samples. In general, matching two distributions requires more training epochs than separating two distributions. Meanwhile, the feature matching term adopted in the generator of complementary GAN is also able to improve the training process [29].

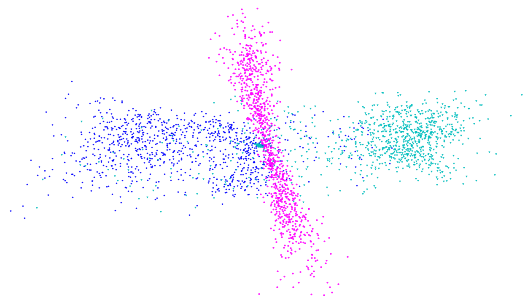


Figure 6: 2D visualization of three types of users: real benign (blue star), vandal (cyan triangle), and complementary benign (red dot)

Visualization of three types of users: We project the user representations of the three types of users (i.e., benign, vandal and complementary benign generated by OCAN) to a two-dimensional space by Isomap [35]. Figure 6 visualizes the three types of users. We observe that the generated complementary users lie in the low-density regions of real benign users. Meanwhile, the generated samples are also between the benign users and vandals. Since the discriminator is trained to separate the benign and complementary benign users, the discriminator is able to separate benign users and vandals.

User clustering: To further analyze the complementary GAN model, we adopt the classic DBSCAN algorithm [10] to cluster 3000 benign users, 3000 vandals from the testing dataset, and 3000 generated complementary benign users. Table 3 shows clustering results including cluster size and class distributions of each cluster. We set the maximum radius of the neighborhood $\epsilon = 1.4305$ (the average distances among the user representations) and the minimum

Table 3: Clustering results of DBSCAN

	Cluster Size	Benign	Vandal	Complement
C1	2537	2448	89	0
C2	2420	93	2327	0
C3	2840	0	0	2840
Isolated	1203	459	584	160

number of points $minPts = 180$. We observe three clusters where C1 is benign users, C2 is vandal, and C3 is complementary samples in addition to 1203 isolated users that could not form any cluster. We emphasize that our OCAN can still make good predictions of those isolated points accurately (with 89% accuracy).

We further calculate the centroid of C1, C2 and C3 based on their user representations and adopt the centroids to calculate distances among each type of users. The distance between the centroids of real benign users and complementary benign users is 3.6346, while the distance between the centroids of real benign users and vandals is 3.888. Since the discriminator is trained to identify real benign users and complementary benign users, the discriminator can detect vandals which have larger distances to real benign users than that of complementary benign users.

6.5 Case Study on Credit Card Fraud Detection

We further evaluate our model on a credit card fraud detection dataset [1]. The dataset records credit card transactions in two days and has 492 frauds out of 284,807 transactions. Each transaction contains 28 features. We adopt 700 genuine transactions as a training dataset and 490 fraud and 490 genuine transactions as a testing dataset. Since the transaction features of the dataset are numerical data derived from PCA, OCAN is able to detect frauds by using raw features as inputs. Meanwhile, we also evaluate the performance of OCAN in the hidden feature space. Because the transaction in credit card dataset is not a sequence data, we adopt the regular autoencoder model instead of LSTM-autoencoder to obtain the transaction representations. In our experiments, the dimension of transaction representation is 50.

Table 4 shows the classification results of credit card fraud detection. Overall, the performance of OCAN and baselines are similar to the results of vandal detection shown in Table 1. OCAN achieves the best accuracy and F1 with both input settings. Meanwhile, the performance of OCAN using transaction representations as inputs is better than using raw features. It shows that OCAN can outperform the existing one-class classifiers in different datasets and can be applied to detect different types of malicious users.

7 CONCLUSION

In this paper, we have developed OCAN that consists of LSTM-Autoencoder and complementary GAN for fraud detection when only benign users are observed during the training phase. During training, OCAN adopts LSTM-Autoencoder to learn benign user representations, and then uses the benign user representations to train a complementary GAN model. The generator of complementary GAN can generate complementary benign user representations that are in the low-density regions of real benign user representations, while the discriminator is trained to distinguish the real and complementary benign users. After training, the discriminator is able to detect malicious users which are outside the regions of benign users. We have conducted theoretical and empirical analysis to demonstrate the advantages of complementary GAN over regular GAN. We conducted experiments over two real world datasets and showed OCAN outperforms the state-of-the-art one-class classification models. Moreover, our OCAN model can also achieve early vandal detection since OCAN takes the user edit sequences as inputs and achieved comparable accuracy with the latest M-LSTM model that needs both benign and vandal users in the training data. In our future work, we plan to extend the techniques for fraud detection in the semi-supervised learning scenario.

[1]https://www.kaggle.com/dalpozz/creditcardfraud

ACKNOWLEDGMENTS

The authors acknowledge the support from National Science Foundation to Panpan Zheng and Xintao Wu (1564250), Jun Li (1564348) and Aidong Lu (1564039).

REFERENCES

[1] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *DMKD* 29, 3 (2015), 626–688.
[2] Fabrício Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgílio Almeida. 2010. Detecting spammers on twitter. In *CEAS*.
[3] Y. Bengio, P. Simard, and P. Frasconi. 1997. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on* 5, 2 (1997), 157–166.
[4] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. 2014. Uncovering Large Groups of Active Malicious Accounts in Online Social Networks. In *CCS*.
[5] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. *arXiv:1606.03657 [cs, stat]* (2016).
[6] Justin Cheng, Michael Bernstein, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2017. Anyone Can Become a Troll: Causes of Trolling Behavior in Online Discussions. In *CSCW*. 1217–1230.
[7] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]* (2014).
[8] Wikipedia contributors. 2010. ClueBot NG. (2010). https://en.wikipedia.org/wiki/User:ClueBot_NG
[9] Zihang Dai, Zhilin Yang, Fan Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Good Semi-supervised Learning that Requires a Bad GAN. In *NIPS*.
[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *KDD*. 226–231.
[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. In *NIPS*.
[12] Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *arXiv:1308.0850 [cs]* (2013).
[13] Stefan Heindorf, Martin Potthast, Benno Stein, and Gregor Engels. 2016. Vandalism Detection in Wikidata. In *CIKM*. 327–336.
[14] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[15] Meng Jiang, Peng Cui, Alex Beutel, Christos Faloutsos, and Shiqiang Yang. 2014. CatchSync: Catching Synchronized Behavior in Large Directed Graphs. In *KDD*.
[16] Michael Kemmler, Erik Rodner, Esther-Sabrina Wacker, and Joachim Denzler. 2013. One-class classification with Gaussian processes. *Pattern Recognition* 46, 12 (2013), 3507–3518.
[17] Shehroz S Khan and Michael G Madden. 2014. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review* 29, 3 (2014), 345–374.
[18] Srijan Kumar, Justin Cheng, Jure Leskovec, and V. S. Subrahmanian. 2017. An Army of Me: Sockpuppets in Online Discussion Communities. In *WWW*. 857–866.
[19] Srijan Kumar, Francesca Spezzano, and V.S. Subrahmanian. 2015. VEWS: A Wikipedia Vandal Early Warning System. In *KDD*. 607–616.

Table 4: Fraud detection results (mean±std.) on precision, recall, F1 and accuracy of credit card fraud detection

Input	Algorithm	Precision	Recall	F1	Accuracy
Raw feature vector	OCNN	0.8029 ± 0.1497	0.9075 ± 0.0596	0.8383 ± 0.0660	0.8132 ± 0.1082
	OCGP	0.9700 ± 0.0222	0.7308 ± 0.0812	0.8302 ± 0.0450	0.8531 ± 0.0324
	OCSVM	0.6590 ± 0.0100	**0.9404 ± 0.0017**	0.7749 ± 0.0068	0.7267 ± 0.0107
	OCAN	**0.9755 ± 0.0110**	0.7416 ± 0.0498	**0.8416 ± 0.0330**	**0.8613 ± 0.0243**
Transaction representation	OCNN	0.7058 ± 0.1396	0.9390 ± 0.0786	0.7910 ± 0.0608	0.7401 ± 0.1079
	OCGP	0.8813 ± 0.1177	0.8566 ± 0.0822	0.8576 ± 0.0417	0.8536 ± 0.0623
	OCSVM	0.6547 ± 0.0151	**0.9509 ± 0.0101**	0.7755 ± 0.0127	0.7245 ± 0.0185
	OCAN	**0.9067 ± 0.0614**	0.8320 ± 0.0319	**0.8656 ± 0.0220**	**0.8701 ± 0.0264**

[20] Srijan Kumar, Robert West, and Jure Leskovec. 2016. Disinformation on the Web: Impact, Characteristics, and Detection of Wikipedia Hoaxes. In *WWW*.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[22] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A Hierarchical Neural Autoencoder for Paragraphs and Documents. In *ACL*.

[23] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. 2010. Detecting Product Review Spammers Using Rating Behaviors. In *CIKM*. 939–948.

[24] Larry M. Manevitz and Malik Yousef. 2001. One-Class SVMs for Document Classification. *JMLR* 2, Dec (2001), 139–154.

[25] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie S. Glance. 2013. What Yelp Fake Review Filter Might Be Doing?. In *ICWSM*. 409–418.

[26] Caleb C. Noble and Diane J. Cook. 2003. Graph-based Anomaly Detection. In *KDD*.

[27] Marco A. F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. 2014. A review of novelty detection. *Signal Processing* 99 (2014), 215–249.

[28] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434 [cs]* (2015).

[29] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved Techniques for Training GANs. *arXiv:1606.03498 [cs]* (2016).

[30] Liam Schoneveld. 2017. *Semi-Supervised Learning with Generative Adversarial Networks.* Ph.D. Dissertation.

[31] Jost Tobias Springenberg. 2016. Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. In *ICLR*.

[32] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. 2015. Unsupervised Learning of Video Representations using LSTMs. In *ICML*.

[33] David M. J. Tax and Robert P. W. Duin. 2001. Uniform Object Generation for Optimizing One-class Classifiers. *JMLR* 2, Dec (2001), 155–173.

[34] David M. J. Tax and Robert P. W. Duin. 2004. Support Vector Data Description. *Machine Learning* 54, 1 (2004), 45–66.

[35] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. 2000. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science* 290, 5500 (2000), 2319–2323.

[36] Sihong Xie, Guan Wang, Shuyang Lin, and Philip S. Yu. 2012. Review Spam Detection via Temporal Pattern Discovery. In *KDD*. 823–831.

[37] X. Ying, X. Wu, and D. Barbará. 2011. Spectrum based fraud detection in social networks. In *ICDE*. 912–923.

[38] Shuhan Yuan, Xintao Wu, Jun Li, and Aidong Lu. 2017. Spectrum-based deep neural networks for fraud detection. In *CIKM*.

[39] Shuhan Yuan, Panpan Zheng, Xintao Wu, and Yang Xiang. 2017. Wikipedia Vandal Early Detection: from User Behavior to User Embedding. In *ECML/PKDD*.

[40] Junbo Zhao, Michael Mathieu, and Yann LeCun. 2016. Energy-based Generative Adversarial Network. *arXiv:1609.03126 [cs, stat]* (2016).