# One-Class Adversarial Nets for Fraud Detection

**Panpan Zheng,[1]* Shuhan Yuan,[1]* Xintao Wu,[1] Jun Li,[2] Aidong Lu[3]**
[1]University of Arkansas, {pzheng,sy005,xintaowu}@uark.edu
[2]University of Oregon, lijun@cs.uoregon.edu
[3]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 with only benign users as training data. 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 of 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.

## Introduction

Online platforms such as online social networks (OSNs) and knowledge bases 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. To protect legitimate users, most Web platforms have tools or mechanisms to block malicious users. For example, Wikipedia adopts ClueBot NG [1] 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 (Cheng et al. 2017; Kumar et al. 2017; Yuan et al. 2017a; Kumar, Spezzano, and Subrahmanian 2015; Yuan et al. 2017b). 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. Manually labeling a large number of malicious users is tedious.

In this work, we tackle the problem of identifying malicious users when only benign users are observed. The basic idea is to adopt a generative model to generate malicious users with only given benign users. Generative adversarial networks (GAN) as generative models have demonstrated impressive performance in modeling the real data distribution and generating high quality synthetic data that is similar to real data (Goodfellow et al. 2014; Radford, Metz, and Chintala 2015). However, given benign users, a regular GAN model is unable to generate malicious users.

We develop one-class adversarial nets (OCAN) for fraud detection. During training, OCAN contains two phases. First, OCAN adopts the LSTM-Autoencoder (Srivastava, Mansimov, and Salakhutdinov 2015) 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 improved generative adversarial nets in which the discriminator is trained to be a classifier for distinguishing benign users and malicious users with the generator producing potential malicious users. To this end, we adopt the idea of bad GAN (Dai et al. 2017) that the generator is trained to generate complementary samples instead of matching the original data distribution. The generator of the complementary GAN aims to generate samples that are complementary to the representations of benign users, i.e., the potential malicious users. We revise the objective function of the discriminator in the regular GAN to achieve one-class classification. The discriminator is trained to separate benign users and complementary samples. Since the behaviors of malicious users and that of benign users are complementary, we expect the discriminator can distinguish benign users and 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

*Equal contribution.

[1]https://en.wikipedia.org/wiki/User:ClueBot_NG

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.

## Related Work

**Fraud detection:** Many fraud detection techniques have been developed in recent years (Akoglu, Tong, and Koutra 2015; Jiang et al. 2014; Cao et al. 2014; Ying, Wu, and Barbará 2011; Kumar and Shah 2018), 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 (Benevenuto et al. 2010). Research in (Kumar, Spezzano, and Subrahmanian 2015) focused on predicting whether a Wikipedia user is a vandal by identifying a set of behavior features based on user edit-patterns. To improve detection accuracy and avoid manual feature construction, a multi-source long-short term memory network (M-LSTM) was proposed to detect vandals (Yuan et al. 2017b). Meanwhile, graph-based approaches identify frauds based on network topologies. 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 (Akoglu, Tong, and Koutra 2015; Manzoor et al. 2016; Ying, Wu, and Barbará 2011; Wu et al. 2013).

**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 (Khan and Madden 2014), which is also related to the novelty detection (Pimentel et al. 2014). One-class support vector machine (OCSVM), 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 (Tax and Duin 2004; Manevitz and Yousef 2001). Other traditional classification models also extend to the one-class scenario. For example, one-class nearest neighbor (OCNN) (Tax and Duin 2001) 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 (Kemmler et al. 2013). 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.

## Preliminary: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 $p_{\text{data}}$. 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:

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

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 $p_{\text{data}}$, $G$ 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:

$$\min_G \quad \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]. \quad (2)$$

Minimizing the Equation 2 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:

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

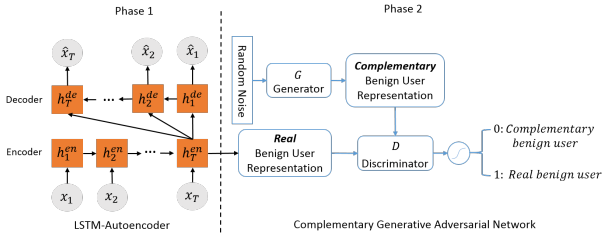



Figure 1: The training framework of OCAN

## OCAN: One-Class Adversarial Nets

### 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 benign user representations from 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 hidden representations of inputs, and the decoder computes the reconstructed inputs based on the hidden representations. The trained LSTM-Autoencoder can capture the salient information of users' activity sequences because the objective function is to make the reconstructed input close to the original input. The encoder of the trained LSTM-Autoencoder, when

deployed for fraud detection, is expected to map the benign users and malicious users to relatively separate regions in the continuous feature space because the activity sequences of benign and malicious users are different.

Given the user representations, the second phase is to train a complementary GAN with a discriminator that can clearly distinguish the benign and malicious users. The generator of the complementary GAN aims to generate complementary samples that are in the low-density area of benign users, and the discriminator aims to separate the real and complementary benign users. The discriminator then has the ability to detect malicious users which locate in separate regions from benign users. The framework of training complementary GAN is shown in the right side of Figure 1.

The pseudo-code of training OCAN is shown in Algorithm 1. Given a training dataset $M_{\text{benign}}$ 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 complementary GAN

1 initialize parameters in LSTM-Autoencoder and complementary 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. 4, 6, and 7);
6  optimize the parameters in LSTM-Autoencoder with the loss function Eq. 8;
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. 4, 5);
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. 14, 12, respectively;
19  **end**
20 **end**
21 **return** well-trained LSTM-Autoencoder and complementary GAN

## 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 $\mathcal{X}_u = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ 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:

$$\mathbf{h}_t^{en} = LSTM^{en}(\mathbf{x}_t, \mathbf{h}_{t-1}^{en}), \tag{4}$$

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 $\mathbf{v}$:

$$\mathbf{v} = \mathbf{h}_T^{en}. \tag{5}$$

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

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

$$\hat{\mathbf{x}}_t = f(\mathbf{h}_t^{de}), \tag{7}$$

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 (Cho et al. 2014).

The objective function of LSTM-Autoencoder is:

$$\mathcal{L}_{(AE)}(\hat{\mathbf{x}}_t, \mathbf{x}_t) = \sum_{t=1}^{T} (\hat{\mathbf{x}}_t - \mathbf{x}_t)^2, \tag{8}$$

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.

## Complementary GAN

The generator $G$ of complementary GAN is the same as that of the bad GAN in (Dai et al. 2017). Basically, it is a feed-forward 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 $p_{\text{data}}$, 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., $p_G = p^*$.

Following (Dai et al. 2017), we define the complementary distribution p* as:

$$p^*(\tilde{\mathbf{v}}) = \begin{cases} \frac{1}{\tau}\frac{1}{p_{\text{data}}(\tilde{\mathbf{v}})} & \text{if } p_{\text{data}}(\tilde{\mathbf{v}}) > \epsilon \text{ and } \tilde{\mathbf{v}} \in \mathcal{B}_{\mathbf{v}} \\ C & \text{if } p_{\text{data}}(\tilde{\mathbf{v}}) \leq \epsilon \text{ and } \tilde{\mathbf{v}} \in \mathcal{B}_{\mathbf{v}}, \end{cases} \quad (9)$$

where $\epsilon$ is a threshold to indicate whether the generated samples are in high-density regions; $\tau$ 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, the objective function is:

$$\begin{aligned} &\mathcal{L}_{KL(p_G \| p^*)} = -\mathcal{H}(p_G) - \mathbb{E}_{\tilde{\mathbf{v}}\sim p_G} \log p^*(\tilde{\mathbf{v}}) \\ &= -\mathcal{H}(p_G) + \mathbb{E}_{\tilde{\mathbf{v}}\sim p_G} \log p_{\text{data}}(\tilde{\mathbf{v}}) \mathbb{1}[p_{\text{data}}(\tilde{\mathbf{v}}) > \epsilon] \\ &+ \mathbb{E}_{\tilde{\mathbf{v}}\sim p_G}(\mathbb{1}[p_{\text{data}}(\tilde{\mathbf{v}}) > \epsilon] \log \tau - \mathbb{1}[p_{\text{data}}(\tilde{\mathbf{v}}) \leq \epsilon] \log C), \end{aligned} \quad (10)$$

where $\mathcal{H}(\cdot)$ is the entropy, and $\mathbb{1}[\cdot]$ is the indicator function. The last term of Equation 10 can be omitted because both $\tau$ 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, following (Dai et al. 2017), the complementary generator $G$ adopts the feature matching loss (Salimans et al. 2016) to ensure that the generated samples are constrained in the space of user representation $\mathcal{B}_{\mathbf{v}}$.

$$\mathcal{L}_{\text{fm}} = \| \mathbb{E}_{\tilde{\mathbf{v}}\sim p_G} f(\tilde{\mathbf{v}}) - \mathbb{E}_{\mathbf{v}\sim p_{\text{data}}} f(\mathbf{v}) \|_2^2, \quad (11)$$

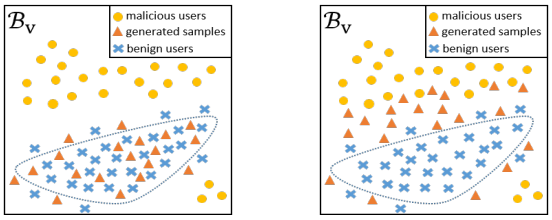
where $f(\cdot)$ denotes the output of an intermediate layer of the discriminator used as a feature representation of $\mathbf{v}$.

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

$$\begin{aligned} \min_G \quad & -\mathcal{H}(p_G) + \mathbb{E}_{\tilde{\mathbf{v}}\sim p_G} \log p_{\text{data}}(\tilde{\mathbf{v}}) \mathbb{1}[p_{\text{data}}(\tilde{\mathbf{v}}) > \epsilon] \\ & + \| \mathbb{E}_{\tilde{\mathbf{v}}\sim p_G} f(\tilde{\mathbf{v}}) - \mathbb{E}_{\mathbf{v}\sim p_{\text{data}}} f(\mathbf{v}) \|_2^2 . \end{aligned} \quad (12)$$

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 2 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 similar distribution of real benign users in the feature space. On the contrary, the objective function of the generator of complementary GAN in Equation 12 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 dot line indicates the high density 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 $p_{\text{data}}$. To minimize $-\mathcal{H}(p_G)$, following (Dai et al. 2017), we adopt the pull-away term (PT) proposed by (Zhao, Mathieu, and LeCun 2016) 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

$$\mathcal{L}_{PT} = \frac{1}{N(N-1)} \sum_{i}^{N} \sum_{j\neq i}^{N} \left(\frac{f(\tilde{\mathbf{v}}_i)^T f(\tilde{\mathbf{v}}_j)}{\| f(\tilde{\mathbf{v}}_i) \| \| f(\tilde{\mathbf{v}}_j) \|}\right)^2, \quad (13)$$

where $N$ is the size of a mini-batch.

The probability distribution of real samples $p_{\text{data}}$ is usually unavailable, and approximating $p_{\text{data}}$ is computationally expensive. In this paper, we adopt the approach proposed by (Schoneveld 2017) that a discriminator from a regular GAN can detect whether the data from the real data distribution $p_{\text{data}}$ or from the generator's distribution. The basic idea is that the discriminator is able to detect whether a sample is from the real data distribution $p_{\text{data}}$ or from the generator when the generator is trained to generate samples that are close to real benign users. Hence, the discriminator is sufficient to identify the data points that are above a threshold of $p_{\text{data}}$ during training. We separately train a regular GAN model based on benign user representations and use the discriminator of the regular GAN as a proxy to evaluate $p_{\text{data}}(\tilde{\mathbf{v}}) > \epsilon$.

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 feed-forward neural network with a softmax function as its output

layer, and we define the objective function of $D$ as:

$$\max_{D} \quad \mathbb{E}_{\mathbf{v}\sim p_{\text{data}}}[\log D(\mathbf{v})] + \mathbb{E}_{\tilde{\mathbf{v}}\sim p_G}[\log(1 - D(\tilde{\mathbf{v}}))] + \mathbb{E}_{\mathbf{v}\sim p_{\text{data}}}[D(\mathbf{v}) \log D(\mathbf{v})]. \tag{14}$$

Different from the objective function of the discriminator introduced in the bad GAN for the purpose of semi-supervised learning, we revise the objective function of D in our complementary GAN based on the regular GAN. The first two terms in Equation 14 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 14 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.

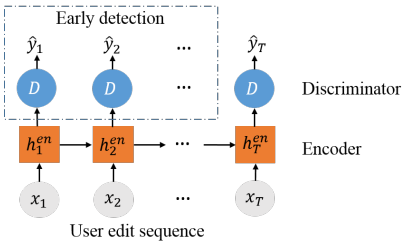



Figure 3: The fraud detection model

## 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 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 (Equations 4 and 5). Then, we predict the user label based on the discriminator of complementary GAN, i.e., $p(\hat{y}_u|\mathbf{v}_u) = D(\mathbf{v}_u)$.

**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 $\mathbf{h}_{u_t}^{en}$ 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 $\mathbf{v}_{u_t} = \mathbf{h}_{u_t}^{en}$. 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 $\mathbf{v}_t$.

## Experiments

### 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 (Kumar, Spezzano, and Subrahmanian 2015). 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.

**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 $\epsilon$ defined in Equation 12 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 are available at https://github.com/PanpanZheng/OCAN.

### Comparison with One-Class Classification

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

- One-class nearest neighbors (**OCNN**) (Tax and Duin 2001) 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.
- One-class Gaussian process (**OCGP**) (Kemmler et al. 2013) is a one-class classification model based on Gaussian process regression.

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

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

- One-class SVM (**OCSVM**) (Tax and Duin 2004) adopts support vector machine to learn a decision hypersphere around the positive data, and considers samples located outside this hypersphere as anomalies.

For baslines, we use the implementation provided in NDtool [2]. The hyperparameters of baselines set as default values in NDtool. 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 (Goodfellow et al. 2014).

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

[2] http://www.robots.ox.ac.uk/~davidc/publications_NDtool.php

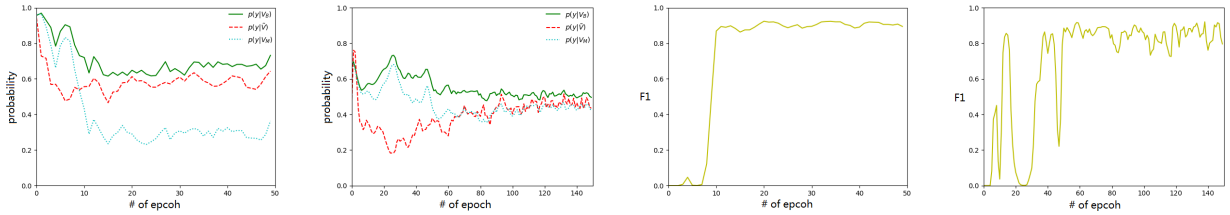
Table 2: Early detection results on precision, recall, F1, and the average number of edits before the vandals are blocked

| | 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 |

## 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 (Yuan et al. 2017b). 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




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

Figure 4: Training progresses of OCAN (4a,4c) and OCAN-r(4b,4d). Three lines in Figures 4a and 4b indicate the probabilities of benign users predicted by the discriminator: real benign users $p(y|\mathbf{v}_B)$ (green line) vs. generated samples $p(y|\tilde{\mathbf{v}})$ (red broken line) vs. real malicious users $p(y|\mathbf{v}_M)$ (blue dotted line). Figures 4c and 4d show the F1 of OCAN and OCAN-r during training.

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.

## OCAN Framework Analysis

**Complementary GAN vs. Regular GAN:** 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 4a), 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 that 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 generates 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 4b, 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, 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 4c and 4d. 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 4 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 (Salimans et al. 2016).

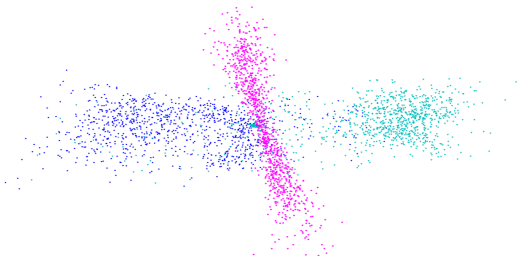


Figure 5: 2D visualization of three types of users: real benign (blue star), vandal (cyan triangle), and complementary benign (red dot)

**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 (Tenenbaum, Silva, and

Langford 2000) and show the projection in Figure 5. 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.

## Conclusion

In this paper, we have developed OCAN for fraud detection when only benign users are available 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 on a real world dataset and showed that OCAN outperforms the state-of-the-art one-class classification models.

## Acknowledgments

This work was supported in part by NSF 1564250, 1564348, 1564039 and 1841119.

## References

Akoglu, L.; Tong, H.; and Koutra, D. 2015. Graph based anomaly detection and description: a survey. *DMKD* 29(3):626–688.

Benevenuto, F.; Magno, G.; Rodrigues, T.; and Almeida, V. 2010. Detecting spammers on twitter. In *CEAS*.

Cao, Q.; Yang, X.; Yu, J.; and Palow, C. 2014. Uncovering large groups of active malicious accounts in online social networks. In *CCS*.

Cheng, J.; Bernstein, M.; Danescu-Niculescu-Mizil, C.; and Leskovec, J. 2017. Anyone can become a troll: Causes of trolling behavior in online discussions. In *CSCW*, 1217–1230.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078 [cs, stat]*.

Dai, Z.; Yang, Z.; Yang, F.; Cohen, W. W.; and Salakhutdinov, R. 2017. Good semi-supervised learning that requires a bad gan. In *NIPS*.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial networks. In *NIPS*.

Jiang, M.; Cui, P.; Beutel, A.; Faloutsos, C.; and Yang, S. 2014. Catchsync: Catching synchronized behavior in large directed graphs. In *KDD*.

Kemmler, M.; Rodner, E.; Wacker, E.-S.; and Denzler, J. 2013. One-class classification with gaussian processes. *Pattern Recognition* 46(12):3507–3518.

Khan, S. S., and Madden, M. G. 2014. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review* 29(3):345–374.

Kumar, S., and Shah, N. 2018. False information on web and social media: A survey. *arXiv:1804.08559 [cs]*.

Kumar, S.; Cheng, J.; Leskovec, J.; and Subrahmanian, V. S. 2017. An army of me: Sockpuppets in online discussion communities. In *WWW*, 857–866.

Kumar, S.; Spezzano, F.; and Subrahmanian, V. 2015. Vews: A wikipedia vandal early warning system. In *KDD*, 607–616.

Manevitz, L. M., and Yousef, M. 2001. One-class svms for document classification. *JMLR* 2(Dec):139–154.

Manzoor, E. A.; Momeni, S.; Venkatakrishnan, V. N.; and Akoglu, L. 2016. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *KDD*.

Pimentel, M. A. F.; Clifton, D. A.; Clifton, L.; and Tarassenko, L. 2014. A review of novelty detection. *Signal Processing* 99:215–249.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434 [cs]*.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A.; and Chen, X. 2016. Improved techniques for training gans. *arXiv:1606.03498 [cs]*.

Schoneveld, L. 2017. *Semi-Supervised Learning with Generative Adversarial Networks*. Ph.D. Dissertation.

Srivastava, N.; Mansimov, E.; and Salakhutdinov, R. 2015. Unsupervised learning of video representations using lstms. In *ICML*.

Tax, D. M. J., and Duin, R. P. W. 2001. Uniform object generation for optimizing one-class classifiers. *JMLR* 2(Dec):155–173.

Tax, D. M. J., and Duin, R. P. W. 2004. Support vector data description. *Machine Learning* 54(1):45–66.

Tenenbaum, J. B.; Silva, V. d.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290(5500):2319–2323.

Wu, L.; Wu, X.; Lu, A.; and Zhou, Z. 2013. A spectral approach to detecting subtle anomalies in graphs. *J. Intell. Inf. Syst.* 41(2):313–337.

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

Yuan, S.; Wu, X.; Li, J.; and Lu, A. 2017a. Spectrum-based deep neural networks for fraud detection. In *CIKM*.

Yuan, S.; Zheng, P.; Wu, X.; and Xiang, Y. 2017b. Wikipedia vandal early detection: from user behavior to user embedding. In *ECML/PKDD*.

Zhao, J.; Mathieu, M.; and LeCun, Y. 2016. Energy-based generative adversarial network. *arXiv:1609.03126 [cs, stat]*.