Securing the Deep Fraud Detector in Large-Scale E-Commerce Platform via Adversarial Machine Learning Approach

Qingyu Guo
qyguo@ntu.edu.sg
Nanyang Technological University

Zhao Li
lizhao.lz@alibaba-inc.com
Alibaba Group

Bo An
boan@ntu.edu.sg
Nanyang Technological University

Pengrui Hui, Jiaming Huang
pengrui.hpr,jimmy.hjm@alibaba-inc.com
Alibaba Group

Long Zhang
james.zl@alibaba-inc.com
Alibaba Group

Mengchen Zhao*
zhao0204@e.ntu.edu.sg
Nanyang Technological University

ABSTRACT

Fraud transactions are one of the major threats faced by online e-commerce platforms. Recently, deep learning based classifiers have been deployed to detect fraud transactions. Inspired by the findings on adversarial examples, this paper is the first one to analyze the vulnerability of deep fraud detector to the slight perturbation on input transactions. However, the sparsity and discretization of transaction data result in the non-convex discrete optimization, which is notoriously challenging, and existing attacks fail to address this issue with L_1 and L_2 distortion bounds. Despite this, in this paper, we make several contributions to (i) address such an optimization problem, (ii) investigate the vulnerability of deployed fraud detector to the practical attacks, and (iii) improve the robustness of the detection model. First, inspired by the iterative Fast Gradient Sign Method (FGSM) for the L_∞ attack, we propose the *Iterative Fast Coordinate Method (IFCM)* for the discrete L_1 and L_2 attacks which is efficient to generate enough instances with satisfactory effectiveness for adversarial training. Second, we provide two novel and strong attack algorithms to address the discrete optimization and evaluate the robustness of the fraud detector. The first one is called *Augmented Iterative Search (AIS)* algorithm, which repeatedly searches for effective "simple" perturbation. The second one is named by *Rounded Relaxation with Reparameterization (R3)*, which rounds the solution obtained by solving a relaxed and unconstrained optimization with reparameterization trick. Third, we conduct the extensive experimental evaluation on millions of real-world transactions and a deployed fraud detector in TaoBao, one of the largest e-commerce platforms in the world. The results show that (i) The deployed detector is highly vulnerable to attacks as the average precision is decreased from nearly 90% to as low as 20% with little perturbation; (ii) Our algorithms significantly outperform the extensive adaptions of the state-of-the-art attacks. (iii) The defense measure of adversarial training significantly improves the robustness of the model as the average precision of 85.9% is achieved under our strong attacks, and meanwhile, the performance on the unperturbed data is preserved at 89%.

*Corresponding author.

WWW'19, May 13-17, San Francisco, USA
2019. ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583312

ACM Reference Format:
Qingyu Guo, Zhao Li, Bo An, Pengrui Hui, Jiaming Huang, Long Zhang, and Mengchen Zhao. 2019. Securing the Deep Fraud Detector in Large-Scale E-Commerce Platform via Adversarial Machine Learning Approach. In *Proceedings of the ACM Web Conference 2019 (WWW '19), May 13-17, 2019, San Francisco, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583312

1 INTRODUCTION

With the rapid growth of information technologies, e-commerce has become prevalent nowadays. Large e-commerce platforms serve billions of users and connect them to factories, stores and third-party merchants with numerous products available. The big success of e-commerce also motivates the emergence of fraud transactions to illegally promote items and stores. In order to increase sales, the fraudulent sellers turn to the third party malicious service platforms where they can hire human labors to create fake purchases and visits [26, 29], and thus provide a fake impression that the target item is popular. This results in the much higher ranking of the items from fraudulent sellers, and causes huge losses to the platforms as it hurts the user experience. The online fraud activities have caused the loss in billions of dollars[1], and it is reported that just in China, the online fraudulent industries involve over 1.6 million human labors and create billions of dollars of illegal incomes[2].

Fraud detection is critical to the development of e-commerce platforms. Recently, with billions of transaction data available, deep learning based models are proposed and deployed to detect fraud transactions on a real-time basis [6, 32, 33]. Inspired by recent findings in the domain of computer vision that deep learning based image classifier could be fooled by imperceptible noises during test phase [27], we are interested in analyzing the vulnerability of deep fraud detector to slight perturbations of the input data. This study is of high practical values for e-commerce platforms as the existence of such adversarial perturbations could indicate the risk of deploying deep fraud detectors. As we show in our motivating example, the fraudulent sellers and malicious service providers are capable of exploiting such perturbations to bypass the detector with little cost.

In this paper, we aim to investigate the vulnerability of the deep fraud detector being fooled by the slight and feasible perturbations. However, the e-commerce domain is different from computer vision,

[1]https://www.signifyd.com/blog/2017/10/26/ecommerce-fraud-eight-industries/
[2]http://cn.chinadaily.com.cn/2015-11/05/content_22379527.htm

as the inputs are mostly discrete and sparse. Few works considering discrete adversarial perturbation either propose the straightforward extension of the existing Fast Gradient Method [9], or take into account only the L_∞ attack [2], and cannot handle more complex L_1 and L_2 attacks, which are of higher practical interests in the domain of fraud detection.

This paper makes the following contributions.

- First, inspired by the Fast Gradient Sign Method (FGSM), to handle the L_1 and L_2 norm bound constraints, we propose the Iterative Fast Coordinate Method (IFCM) which repeatedly conducts a single unit descent on a single coordinate. IFCM is shown to efficiently generate adversarial perturbations with satisfactory effectiveness.
- Second, to address the discrete optimization of adversarial perturbations and further exploit the vulnerability of the deep fraud detector, we provide two novel attacks. The first attack is *Augmented Iterative Search (AIS)*. In each iteration, AIS proceeds to search for "simple" perturbations instructed by a priority policy to trade-off the goal of decreasing fraudulent score and satisfying the distortion bound. AIS also augments the searching process via restricting the space of tuple-unit perturbation based on the *Value of Perturbation (VoP)*. The second attack is *Rounded Relaxation with Reparameterization (R3)*, an optimization-based attack. R3 first solves the relaxation of the discrete optimization with reparameterization trick and rounds the continuous solution to obtain the discrete perturbation satisfying the distortion constraint.
- Third, we conduct an extensive experimental evaluation with the real-world transaction data and deployed fraud detector from TaoBao, one of the largest e-commerce platforms in the world. The experimental results show that (i) the deployed fraud detector is severely threatened by crafted adversarial perturbations, as our AIS and R3 attacks successfully decrease the detector's average precision from nearly 90% to as low as 20% with little perturbation. Meanwhile, through case studies, we verify that the created adversarial perturbations are feasible for the malicious sellers or service providers to implement. (ii) Our proposed AIS and R3 attacks significantly outperform several state-of-the-art attacks adapted to the domain of fraud detection, including C&W attack [4], Logit-Space attack [2], and EAD attack [5] in all testing settings. (iii) We propose the adversarial training with adversarial examples generated by IFCM, and results show that the robustness of the detection model is significantly improved as we achieve an average precision above 85.9% under all tested attacks. Meanwhile, the average precision on unperturbed data remains nearly 90%, suggesting the necessity of adversarial training to secure the deep fraud detector.

2 RELATED WORK

In this section, we review the related works in crafting adversarial examples to attack DNN models and various defense techniques.

2.1 Attacks on DNNs

FGSM [8] and its variant R+FGSM [28] are single step attacks, which perturb the input towards the gradient direction of the loss function. The extensions of FGSM iteratively perform single step attacks with smaller step size, such as BIM [13] and PGD [15]. Other iterative attacks include DeepFool [17], which approximates the decision boundary near the target instance with the linear hyperplane, and UAP [16] to produce a single perturbation which is capable of attacking multiple images. FGSM and R+FGSM are further extended to perturb towards the direction which maximizes the confidence score on the least likely labels for the target image [8, 28].

Besides gradient directional methods, Papernot et al. [21] propose the JSMA attack which uses the Jacobian matrix to get the salient map and modifies the features with high significance to confuse the classifier. C&W attack [3] relaxes the hard constraint on the distortion, and adds a penalty on the L_p norm ($p \in \{0, 2, \infty\}$) of perturbation. EAD [5] improves the transferability of adversarial examples through elastic-net regularization. Several works study the adversarial examples to evade DNN models in real-world applications, including the face recognition system [25], objection detection [11], and malware detection [9].

Most of the works focus on image recognition tasks and regard the perturbation as continuous which is not the case in fraud detection domain. While Grosse et al. consider the binary input [9] and adapt the fast gradient algorithms to solve their problem in a similar way as our IFCM$_1$, they only consider the L_1 attack. Buckman et al. propose the Gumbel-softmax trick to approximate the one-hot encoding of the discrete feature. However, their attack is suitable for L_∞ distortion bound and fails to consider the constraints that correlate the input features such as L_1 and L_2 norm constraints in our problem.

Our R3 attack is inspired by optimization-based L_2 attacks in the domain of computer vision [4, 5, 27]. These attacks all handle the L_2 norm constraint by imposing a penalty in the objective function and conducting a binary search to find a suitable constant coefficient of penalty term. This is a reasonable way to find the minimal distortion to successfully craft an adversarial example. However, in our problem, we are asked to find the most effective adversarial instance given the maximal distortion. While one could conduct a binary search to find the suitable constant coefficient that the distortion constraint is satisfied, the neural networks are non-convex in general and empirical attempts show that the binary search is highly unstable to generate a distortion satisfying the given constraint [27]. On the other hand, our R3 attack utilizes the reparameterization trick to transform the constrained optimization to an equivalent unconstrained problem easier to solve.

2.2 Defenses

There are several defense measures to improve the robustness of deep learning based classifiers, such as defensive distillation [20, 22], which adopts the distillation technique [10] to retrain the same network with the confidence score outputted by the original model as the soft label, feature squeezing [30], detection approaches for adversarial instances [4], and adversarial (re)training [8], a general process based on robust optimization that augments the training data with adversarial examples [8, 15, 34]. Some recent works propose to regularize or mask the gradient of the loss function with respect to the inputs [24]. Almost all defenses are shown to be effective only for some attacks [31].

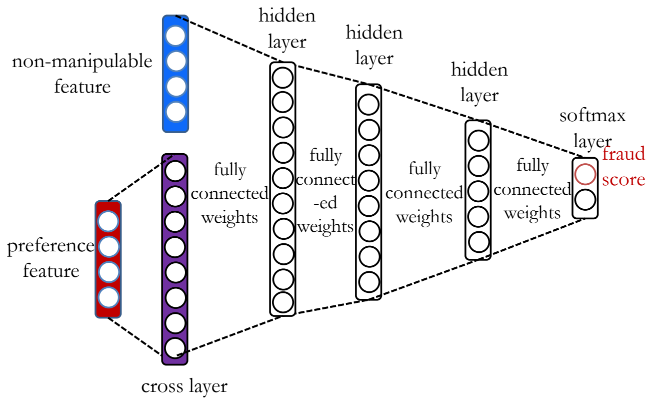


Figure 1: Abstracted architecture of deployed deep fraud detector

Recently, Athalye et al. pointed out the defenses based on obfuscated gradients are threatened by their characteristic behaviors which can be exploited to design attacks circumventing them [1]. It is believed that adversarial training does not cause obfuscated gradient [1]. Thus, in this paper, we use adversarial training to achieve robust fraud detection and our experimental evaluation verifies the effectiveness of the strengthened detector.

3 DEEP FRAUD DETECTION AND ATTACKS

In this section, we first briefly introduce the deep learning model for fraud detection which is used by one of the largest e-commerce platforms in the world. We then present concrete examples to show the possible attacks by malicious sellers and service providers.

3.1 Deep Fraud Detection

Figure 1 shows the architecture of deep neural networks in deployed fraud detection system of TaoBao. The goal of the detector is to predict whether an online transaction is fraudulent. For this aim, each input transaction is represented as a feature vector $\mathbf{x}$ which concatenates two classes of features. The first is the non-manipulable features. For instance, there are several fraudulent indices associated with buyers and items provided by offline methods, such as label propagation on the user-item transaction graphs, to measure the offline maliciousness based on past records. Other non-manipulable features are prices[3] In this paper, we focus on manipulatable features whose adjustment could have a significant effect on the detector's prediction. For this aim, in order to ease the reading of the paper, we denote a transaction by the vector of only perturbable features. In the context of fraud detection we investigate, the perturbable features are those characterizing the user's preference, defined as follows.

$\mathbf{x} \in \mathbb{N}^{|C|}$ denotes the user's preferences and interests over the global set of item categories C. Let $n = |C|$. x_i is the number of records that the user interacts with items in i-th category of C in past fixed period, normally 30 days, such as adding an item in shopping cart, adding to "favorite", etc. $\mathbf{x}$ represents the user's global interest in online shopping. Fraudulent buyers tend to show different preference patterns compared with benign users. Each transaction is associated with a label $y \in \{0, 1\}$, such that $y = 1$ if the transaction is fraudulent and $y = 0$ otherwise.

All features are first normalized to be within $[0, 1]^n$ with a "smooth" normalization function to tackle the extreme values. Here, the feature value $\mathbf{x}$ is normalized to be $\frac{\log(\mathbf{x}+1)}{\log M}$ in element-wise manner where M is the largest number of interactions observed in training data. The preference features are first passed to a cross layer [14] to exploit the interactions between different categories. The output of cross layer is concatenated with non-manipulable features and passed to a multi-layer feed-forward neural network. Each hidden-layer consists of rectified linear units as hidden neurons [18]:

$$relu(x) = \max(0, x)$$

The output of the last hidden-layer $\mathbf{z} \in \mathbb{R}^2$, also called the *logit*, is then passed to a softmax layer to produce the confidence score on the transaction $\mathbf{x}$ being fraudulent

$$f^{\mathbf{w}}(\mathbf{x}) = \frac{e^{z_1}}{e^{z_0} + e^{z_1}}$$

where $\mathbf{w}$ denotes the parameters of the model, including the weights and biases on hidden neurons. The model is trained over a large-scale data set $\mathcal{D} = \{\mathbf{x}^i, y^i\}_{i=1}^N$ by minimizing the regularized total loss function $L^{\mathbf{w}}(\mathbf{x}, y)$ over $\mathcal{D}$

$$\min_{\mathbf{w}} \underset{(\mathbf{x},y)\sim\mathcal{D}}{\mathbb{E}} [L^{\mathbf{w}}(\mathbf{x}, y)] + \lambda\|\mathbf{w}\|^2 \tag{1}$$

where λ is the regularization coefficient. The commonly used loss function in classification tasks is the cross-entropy loss

$$L^{\mathbf{w}}(\mathbf{x}, y) = -\mathbb{1}_{\{y=1\}} \log f^{\mathbf{w}}(\mathbf{x}) - \mathbb{1}_{\{y=0\}} \log (1 - f^{\mathbf{w}}(\mathbf{x}))$$

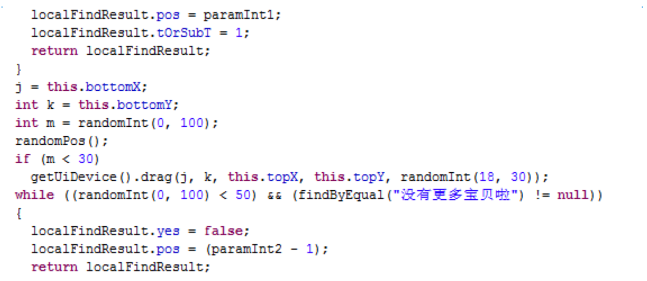
and the stochastic gradient descent is widely applied to optimize the parameter $\mathbf{w}$ [7, 23].

3.2 Attacks to Fraud Detector

With billions of customers served on e-commerce platforms, the profit of conducting fraud transactions could be huge for the malicious sellers as they significantly promote the target items in search rankings. Driven by this huge profit, a lot of malicious service platforms provide sophisticated tactics to bypass the strict detection of platforms and conduct fraud transactions for the fraudulent stores. One common tactic is to mimic benign users' behavior such as using automatic visit script to visit different items before purchasing the target items and hiring human labors to chat with the merchants before making a transaction. Figure 2 shows an example of automatic scripts to bypass the fraud detection, which is a piece of Java code as part of a mobile malicious service App which creates random perturbation to a target transaction such as dragging and browsing to behave like humans, intentionally visiting items from certain categories, etc. According to the domain experts, malicious platforms can create around 200 thousand visits in total per day with those Apps and promote over one thousand items from malicious sellers. If vulnerabilities exist for a fraud detector, the malicious platforms could exploit those threats to adapt the fraud transaction data in order to bypass the detection.

Formally, let δ be the vector of perturbation on fraud instance $\mathbf{x}$ where δ_i is the amount of changes on x_i, and δ is discrete, i.e.,

[3]Prices are hard to perturb in practice and the empirical study shows that slight perturbations on prices cause negligible changes



```
    localFindResult.pos = paramInt1;
    localFindResult.tOrSubT = 1;
    return localFindResult;
  }
  j = this.bottomX;
  int k = this.bottomY;
  int m = randomInt(0, 100);
  randomPos();
  if (m < 30)
    getUiDevice().drag(j, k, this.topX, this.topY, randomInt(18, 30));
  while ((randomInt(0, 100) < 50) && (findByEqual("没有更多宝贝啦") != null))
  {
    localFindResult.yes = false;
    localFindResult.pos = (paramInt2 - 1);
    return localFindResult;
```

Figure 2: Example of fraudulent automatic scripts

$\delta \in \mathbb{Z}^n$. Assume $\delta \geq \mathbf{0}$ since it is more practical for realistic adversaries to add additional records of interactions via measures such as aforementioned automatic scripts, rather than to retrieve existing footprints of interactions[4]. In the domain of computer vision, the adversarial perturbations need to satisfy some constraints which essentially indicate that the perturbations are small enough, so that the ground truths of the adversarial instances remain the same as the original ones [8]. Here, we also impose similar restrictions on δ, in order to capture the *feasible* perturbations that adversaries can conduct with little effort. Such restrictions are normally expressed by the L_p norm bound on perturbations, where p is commonly drawn from $\{0, 1, 2, \infty\}$. Thus, the set of feasible δ bounded by L_p norm is defined as follows.

$$\Delta_q^p := \{\delta \mid \delta \in \mathbb{Z}^n, \|\delta\|_p \leq q, \delta \geq \mathbf{0}\} \tag{2}$$

where q is the maximal distortion in L_p norm, and

$$\|\delta\|_0 = \sum_{i=1}^{n} \mathbb{1}_{\{\delta_i > 0\}} \qquad \|\delta\|_1 = \sum_{i=1}^{n} |\delta_i|$$

$$\|\delta\|_2 = \sqrt{\sum_{i=1}^{n} \delta_i^2} \qquad \|\delta\|_\infty = \max_i |\delta_i|.$$

In this paper, we focus on attacks bounded by L_1 and L_2 norms, which are realistic against the discrete and sparse preference features. L_0 attack is not practical as it allows the adversary to add infinitely many records on one category. L_∞ attack is not reasonable as the preference features are sparse. While there are around 400 categories, in average the total number of interactions for a user is limited to only two or three hundred. However, even an adversarial perturbation with L_∞ norm of 1 can add one interaction at each categories, violating the sparse nature of preference features.

After choosing a suitable distortion metric, for a deep learning classifier with model parameter $\mathbf{w}$, given a fraud transaction instance $\mathbf{x}$ in testing or deploying phase, we aim to find the adversarial example, an instance $\mathbf{x}^{adv}$ crafted with a feasible perturbation $\delta \in \Delta_q^p$, i.e., $\mathbf{x}^{adv} = \mathbf{x} + \delta$, such that the classifier makes a different prediction on $\mathbf{x}^{adv}$. This can be done by solving the following optimization problem

$$\begin{aligned} \min_{\delta} \quad & f^{\mathbf{w}}(\mathbf{x} + \delta) \\ \text{s.t.} \quad & \delta \in \Delta_q^p. \end{aligned} \tag{3}$$

[4]Note that, our attacks can be easily adapted to allow negative perturbations, and the conclusions remain the same as verified by additional experimental evaluation.

Discrete optimization is known to be challenging, and few works have considered crafting effective L_1 and L_2 attacks with discrete perturbations. In the following sections, we will present our three novel L_1 and L_2 attack algorithms. We first provide an *Iterative Fast Coordinate Method (IFCM)* which is efficient enough to generate a large number of adversarial examples with satisfactory attacking effectiveness. Therefore, IFCM is utilized in the defense measure of adversarial training. To evaluate the robustness of the detector more accurately, we provide two stronger attacks, one is the search-based attack called *Augmented Iterative Search (AIS)*, and the other is optimization-based attack named *Rounded Relaxation with Reparameterization (R3)*. Though AIS and R3 cannot generate millions of adversarial examples for adversarial training, their scalability is sufficient to evaluate the robustness of the model, and both algorithms outperform the adaption of the state-of-the-art methods significantly in attacking effectiveness. Empirical evaluation shows that with adversarial training on samples generated by IFCM, the model is significantly more robust against stronger attacks such as AIS and R3, which shows the significance of our work in securing the deep fraud detector.

4 FAST ATTACK: IFCM

In this section, we propose an efficient and effective attack algorithm called *Iterative Fast Coordinate Method* (IFCM), which is inspired by the iterative gradient sign method [13]. Given the sparsity and discretilization of input data, IFCM conducts multiple gradient descents, each with step size 1 along a single coordinate. The descending coordinate is selected in different manners when generating L_1 and L_2 attacks respectively. In IFCM for L_1 attack (IFCM$_1$), the coordinate with the smallest partial derivative is chosen. While in IFCM for L_2 attack (IFCM$_1$), the selection of descending coordinate not only considers the value of partial derivative along that coordinate, but also takes into account the increment of L_2 norm caused by one unit descent along the coordinate. Thus, following the greedy algorithm for the knapsack problem, IFCM$_2$ selects the coordinate with the highest *Value of Perturbation* (VoP) which is defined later. The two variants of IFCM are illustrated as follows.

4.1 IFCM$_1$

As illustrated by Algorithm 1, at each iteration, IFCM$_1$ computes the gradient $\nabla_{\mathbf{x}} f^{\mathbf{w}}(\mathbf{x}^{adv})$ of the fraudulent score with the current adversarial example (Line 3), and selects the coordinate i_{min} with the minimal partial derivative (Line 4). If the partial derivative of $f^{\mathbf{w}}(\mathbf{x}^{adv})$ with $x_{i_{min}}$ is negative, a step of coordinate descent along the coordinate i_{min} is conducted with step size 1; Otherwise, when the gradient $\nabla_{\mathbf{x}} f^{\mathbf{w}}(\mathbf{x}^{adv})$ is positive along every coordinate, or the maximal distortion q is reached, the algorithm terminates and returns $\mathbf{x}^{adv}$.

4.2 IFCM$_2$

The main difference between IFCM$_2$ and IFCM$_1$ lies in the choice of descending coordinate. IFCM$_1$ selects the coordinate simply by comparing the partial derivative, while the coordinate chosen in IFCM$_2$ is decided by *Value of Perturbation* (VoP). VoP is inspired by the greedy approach for knapsack problem. For each coordinate i, a unit descent on such coordinate decreases the fraudulent

Algorithm 1: Iterative Fast Coordinate Method for L_1 attack (IFCM$_1$)

```
input  : Fraud instance $\mathbf{x}$, $q$, detection model parameter $\mathbf{w}$
output : Adversarial example $\mathbf{x}^{adv}$ with $L_1$ norm bounded distortion
1 $\mathbf{x}^{adv} \leftarrow \mathbf{x}, \delta \leftarrow \mathbf{0}_n$;
2 while $\sum_{i=1}^{n} \delta_i < q$ do
3     Compute the gradient $\nabla_{\mathbf{x}} f^{\mathbf{w}}(\mathbf{x}^{adv})$ with back-propagation;
4     $i_{min} = \arg\min_i \frac{\partial f^{\mathbf{w}}(\mathbf{x}^{adv})}{\partial x_i}$;
5     if $\frac{\partial f^{\mathbf{w}}(\mathbf{x}^{adv})}{\partial x_{i_{min}}} > 0$ then
6         return $\mathbf{x}^{adv}$;
7     $x^{adv}_{i_{min}} \leftarrow x^{adv}_{i_{min}} + 1, \delta_{i_{min}} \leftarrow \delta_{i_{min}} + 1$;
8 return $\mathbf{x}^{adv}$.
```

Algorithm 2: Iterative Fast Coordinate Method for L_2 attack (IFCM$_2$)

```
input  : Fraud instance $\mathbf{x}$, $q$, detection model parameter $\mathbf{w}$
output : Adversarial example $\mathbf{x}^{adv}$ with $L_1$ norm bounded distortion
1 $\mathbf{x}^{adv} \leftarrow \mathbf{x}, \delta \leftarrow \mathbf{0}_n$;
2 while $\sqrt{\sum_{i=1}^{n} \delta_i^2} < q$ do
3     Compute the gradient $\nabla_{\mathbf{x}} f^{\mathbf{w}}(\mathbf{x}^{adv})$ with back-propagation;
4     Compute $VoP_i$ for $i \in [n]$;
5     $i_{max} = \arg\max_i VoP_i$;
6     if $VoP_{i_{max}} \leq 0$ then
7         return $\mathbf{x}^{adv}$;
8     $x^{adv}_{i_{max}} \leftarrow x^{adv}_{i_{max}} + 1, \delta_{i_{max}} \leftarrow \delta_{i_{max}} + 1$;
9 return $\mathbf{x}^{adv}$.
```

score $f^{\mathbf{x}^{adv}}$ by an estimated amount of $-\frac{\partial f^{\mathbf{w}}(\mathbf{x}^{adv})}{\partial x_i}$ (the *value*), and meanwhile increases the L_2 norm of current perturbation δ by $\sqrt{\|\delta\|_2^2 + 2\delta_i + 1} - \|\delta\|_2$ (the *weight*). VoP_i is defined as the estimated decrease on the fraudulent score per unit increase of L_2 norm via increasing δ_i

$$VoP_i = \begin{cases} \frac{-\frac{\partial f^{\mathbf{w}}(\mathbf{x}^{adv})}{\partial x_i}}{\sqrt{\|\delta\|_2^2+2\delta_i+1}-\|\delta\|_2} & \text{if } \sqrt{\|\delta\|_2^2 + 2\delta_i + 1} \leq q \\ -\infty & \text{otherwise.} \end{cases}$$

5 SEARCH-BASED ATTACK: AIS

In order to further evaluate the vulnerability of deep fraud detector to adversarial perturbation, we propose a strong attack method called *Augmented Iterative Search* (AIS) which shows superior performance in fooling the detector and outperforms all benchmarks significantly in our experimental evaluation. Since the size of perturbation space is the combinatorial number, for example, $|\Delta_q^1| = C_{n+q-1}^q$, the exhaustive search is intractable. Thus, AIS repeatedly searches for a “simple" perturbation which brings the largest decrease on the objective value of optimization (3) and accumulates those “simple" perturbations to form the final solution. The simplest perturbation is the *single-unit* perturbation which has values zeros on all but one entries, with value 1 or -1 on the only non-zero entry. AIS goes further by taking into account the *tuple-unit* perturbations which have two entries with non-zero values, one with 1 and one with -1[5]. However, the number of all possible tuple-unit perturbations is of quadratic order $O(n^2)$ with n, which makes it impractical to exhaustively search all of them. Thus, AIS augments the search process of the tuple-unit perturbations by effectively restricts the space of such perturbations, as we will explain later.

5.1 AIS$_1$

Similar with IFCM, we implement two variants of AIS, AIS$_1$ and AIS$_2$, for L_1 and L_2 attacks respectively. AIS$_1$ is illustrated in Algorithm 3. δ^t denotes the accumulative perturbations until iteration t and $\delta^0 = \mathbf{0}$. At each iteration t, the set of all single-unit perturbations is denoted by $\Delta_t^- \cup \Delta_t^+$ where Δ_t^- contains the single-unit perturbations which have value -1 on the only non-zero entry and the adoption of $\delta^- \in \Delta_t^-$ decreases the L_1 norm of accumulative perturbation δ^{t-1} by 1. Similarly, $\delta^+ \in \Delta_t^+$ has value 1 on the only non-zero entry. Formally,

$$\begin{aligned} \Delta_t^- &= \{\delta \in \mathbb{Z}^n \mid \|\delta\|_1 = 1, \|\delta^{t-1} + \delta\|_1 = \|\delta^{t-1}\|_1 - 1, \delta^{t-1} + \delta \geq \mathbf{0}\} \\ \Delta_t^+ &= \{\delta \in \mathbb{Z}^n \mid \|\delta\|_1 = 1, \|\delta^{t-1} + \delta\|_1 = \|\delta^{t-1}\|_1 + 1\}. \end{aligned} \tag{4}$$

The set of all tuple-unit perturbations is denoted by $\Delta_t^\pm$

$$\Delta_t^\pm = \{\delta^+ + \delta^- \mid \delta^+ \in \Delta_t^+, \delta^- \in \Delta_t^-\}$$

However, the size of $\Delta_t^\pm$ is $mn - m$ where m is the number of non-zero entries in δ^{t-1}, and in the worst case, $\|\Delta_t^\pm\|$ can be quadratic order of n. To augment the search of tuple-unit perturbation, AIS$_1$ replaces $\Delta_t^\pm$ with its restricted subset $\tilde{\Delta}_{t,k}^\pm$ with much smaller size. In order to do so, we first define the value of a “simple" perturbation δ with respect to current solution $\mathbf{x}^{adv}$ as the marginal decrease on the objective value of (3) due to the adoption of δ

$$V(\mathbf{x}^{adv}, \delta) = f^{\mathbf{w}}(\mathbf{x}^{adv}) - f^{\mathbf{w}}(\mathbf{x}^{adv} + \delta)$$

The restricted set of tuple-unit perturbations $\tilde{\Delta}_{t,k}^\pm$ is defined as follows

$$\tilde{\Delta}_{t,k}^\pm = \{\delta^+ + \delta^- \mid \delta^+ \in \tilde{\Delta}_{t,k}^+, \delta^- \in \tilde{\Delta}_{t,k}^-\} \tag{5}$$

where $\tilde{\Delta}_{t,k}^+ \subset \Delta_t^+$ and $\tilde{\Delta}_{t,k}^- \subset \Delta_t^-$ contain the k single-unit perturbations with top-k values among Δ_t^+ and Δ_t^- respectively. The intuition behind $\tilde{\Delta}_{t,k}^\pm$ is that a tuple-unit perturbation is more likely to has higher marginal decrease on $f^{\mathbf{w}}(\mathbf{x}^{adv})$ if it consists of the single-unit perturbations with high values. The size of $\tilde{\Delta}_{t,k}^\pm$ is of order $O(k^2)$, and the choice of k balances the scalability and solution quality.

With Δ_t^-, Δ_t^+ and $\tilde{\Delta}_{t,k}^\pm$ defined in equations (4) and (5), we are ready to introduce the search operation in AIS$_1$. Given the L_1 norm bound of distortion, a reasonable policy is to proceed the spaces of perturbations Δ_t^-, Δ_t^+ and $\tilde{\Delta}_{t,k}^\pm$ separately with different priorities.

[5]Notice that the value of -1 does not necessarily violate the non-negative requirement on δ, which is the accumulative perturbation of “simple" perturbations adopted so far.

Δ_t^- is assigned with the highest priority as the perturbation within it can decrease the L_1 norm of the accumulative perturbation. Thus, if there exists one single-unit perturbation $\delta^- \in \Delta_t^-$ with positive value, it is returned and added to δ^{t-1}, the accumulative perturbation so far (Lines 4–6). If Δ_t^- contains no single-unit perturbation with positive value, $\tilde{\Delta}_{t,k}^{\pm}$ is proceeded. Similarly, the tuple-unit perturbation $\delta^{\pm}$ with positive value is returned and δ^t is updated (Lines 8–10). Finally, if both Δ_t^- and $\tilde{\Delta}_{t,k}^{\pm}$ cannot help to decrease the objective $f^{\mathbf{w}}(\mathbf{w}(\mathbf{x}^{adv})$, AIS$_1$ sacrifices one-unit increase of L_1 norm on accumulative perturbation and searches for δ^+ with positive value from Δ_t^+ (Lines 14–16).

Algorithm 3: Augmented Iterative Search (AIS) for L_1 attack

```
input : Fraud instance $\mathbf{x}, q$, maximum iteration $K$, detection model parameter $\mathbf{w}$, $k$
output: Adversarial example $\mathbf{x}^{adv}$ with $L_1$ norm bounded distortion
1  $\mathbf{x}^{adv} \leftarrow \mathbf{x}, \delta^0 \leftarrow \mathbf{0}_n$;
2  for $t \leftarrow 1$ to $K$ do
3      Compute $\Delta_t^-$, $\Delta_t^+$ and $\tilde{\Delta}_{t,k}^{\pm}$ with (4) and (5);
4      Solve $\min_{\delta \in \Delta_t^-} V(\mathbf{x}^{adv}, \delta)$ and get $\delta^-$;
5      if $V(\mathbf{x}^{adv}, \delta^-) > 0$ then
6          $\delta^t \leftarrow \delta^{t-1} + \delta^-$, $\mathbf{x}^{adv} \leftarrow \mathbf{x}^{adv} + \delta^-$;
7      else
8          Solve $\min_{\delta \in \tilde{\Delta}_t^{\pm}} V(\mathbf{x}^{adv}, \delta)$ and get $\delta^{\pm}$;
9          if $V(\mathbf{x}^{adv}, \delta^{\pm}) > 0$ then
10             $\delta^t \leftarrow \delta^{t-1} + \delta^{\pm}$, $\mathbf{x}^{adv} \leftarrow \mathbf{x}^{adv} + \delta^{\pm}$;
11         else
12             if $\|\mathbf{x}^{adv} - \mathbf{x}\|_1 = q$ then
13                 return $\mathbf{x}^{adv}$
14             Solve $\min_{\delta \in \Delta_t^+} V(\mathbf{x}^{adv}, \delta)$ and get $\delta^+$;
15             if $V(\mathbf{x}^{adv}, \delta) > 0$ then
16                 $\delta^t \leftarrow \delta^{t-1} + \delta^+$, $\mathbf{x}^{adv} \leftarrow \mathbf{x}^{adv} + \delta^+$;
17             else
18                 return $\mathbf{x}^{adv}$;
19 return $\mathbf{x}^{adv}$
```

5.2 AIS$_2$

The adoption of AIS to generate L_2 attack follows the similar manner as AIS$_1$. The key difference is that, in AIS$_1$, a single-unit perturbation would always change the L_1 norm of the accumulative perturbation by 1, and the L_1 norm of the accumulative perturbation is unchanged when a tuple-unit perturbation is applied. However, this is not always the case when producing L_2 attacks. Therefore, the value of a "simple" perturbation needs to take into account not only the decrease of fraudulent score $f^{\mathbf{w}}(\mathbf{x}^{adv})$, but also the change of the L_2 norm of accumulative perturbation. Given the current solution $\mathbf{x}^{adv}$ of optimization (3) and a feasible "simple" perturbation δ, we define the *Value of Perturbation (VoP)* as follows.

$$VoP(\mathbf{x}^{adv}, \delta) =$$
$$\begin{cases} \frac{f^{\mathbf{w}}(\mathbf{x}^{adv}) - f^{\mathbf{w}}(\mathbf{x}^{adv}+\delta)}{\|\mathbf{x}^{adv}+\delta\|_2 - \|\mathbf{x}^{adv}\|_2 + \epsilon} & \text{if } \|\mathbf{x}^{adv}+\delta\|_2 \geq \|\mathbf{x}^{adv}\|_2 \\ (\|\mathbf{x}^{adv}\|_2 - \|\mathbf{x}^{adv}+\delta\|_2) \cdot (f^{\mathbf{w}}(\mathbf{x}^{adv}) - f^{\mathbf{w}}(\mathbf{x}^{adv}+\delta)) & \text{o.w.} \end{cases}$$

Similar with AIS$_1$, we start from the definition the single-unit perturbations, which form Δ_t^- and Δ_t^+. Δ_t^- is the same as defined in (4) while Δ_t^+ is a bit different, considering the L_2 norm constraint.

$$\Delta_t^+ = \{\delta \in \mathbb{Z}^n \mid \|\delta\|_1 = 1, \|\delta^{t-1}+\delta\|_1 = \|\delta^{t-1}\|_1+1, \|\delta^{t-1}+\delta\|_2 \leq q\}.$$

The restricted set $\tilde{\Delta}_{t,k}^{\pm}$ naturally follows the definition in (5) where $\tilde{\Delta}_{t,k}^{+} \subset \Delta_t^+$ and $\tilde{\Delta}_{t,k}^{-} \subset \Delta_t^-$ contain the k single-unit perturbations with top-k *VoP*s among Δ_t^+ and Δ_t^- respectively. Notice that in AIS, the proceeding of "simple" perturbation space is instructed by a priority policy that those perturbations which can decrease the L_2 norm of accumulative perturbation are evaluated first. For this aim, we divide $\tilde{\Delta}_{t,k}^{\pm}$ into two sets, $\tilde{\Delta}_{t,k}^{\pm,m}$ which consists of perturbations whose adoption can decrease the L_2 norm of accumulative perturbation, and $\tilde{\Delta}_{t,k}^{\pm,p}$, the complementary set of $\tilde{\Delta}_{t,k}^{\pm,m}$.

$$\tilde{\Delta}_{t,k}^{\pm,m} = \{\delta \in \tilde{\Delta}_{t,k}^{\pm} \mid \|\delta^{t-1}+\delta\|_2 < \|\delta^{t-1}\|_2\} \quad \tilde{\Delta}_{t,k}^{\pm,p} = \tilde{\Delta}_{t,k}^{\pm} \backslash \tilde{\Delta}_{t,k}^{\pm,m}.$$

Algorithm 4 illustrates the details of AIS$_2$. The first priority of AIS$_2$ is to proceed $\Delta_t^- \cup \tilde{\Delta}_{t,k}^{\pm,m}$, with perturbations decreasing the L_2 norm of the accumulative perturbation. If there exists one "simple" perturbation $\delta^- \in \Delta_t^- \cup \tilde{\Delta}_{t,k}^{\pm,m}$ with positive *VoP*, it is returned and added to δ^{t-1}, the accumulative perturbation so far (Lines 4–6). Otherwise, $\Delta_t^+ \cup \tilde{\Delta}_{t,k}^{\pm,p}$ is proceeded (Lines 8–10). The algorithm terminates until no "simple" perturbation within $\Delta_t^- \cup \Delta_t^+ \cup \tilde{\Delta}_{t,k}^{\pm}$ has positive *VoP* (Line 12) or the maximum iteration K is reached.

Algorithm 4: Augmented Iterative Search (AIS) for L_2 attack

```
input : Fraud instance $\mathbf{x}, q$, maximum iteration $K$, detection model parameter $\mathbf{w}$, $k$
output: Adversarial example $\mathbf{x}^{adv}$ with $L_2$ norm bounded distortion
1  $\mathbf{x}^{adv} \leftarrow \mathbf{x}, \delta^0 \leftarrow \mathbf{0}_n$;
2  for $t \leftarrow 1$ to $K$ do
3      Compute $\Delta_t^-$, $\Delta_t^+$ and $\tilde{\Delta}_{t,k}^{\pm}$;
4      Solve $\min_{\delta \in \Delta_t^- \cup \tilde{\Delta}_{t,k}^{\pm,m}} VoP(\mathbf{x}^{adv}, \delta)$ and get $\delta^-$;
5      if $VoP(\mathbf{x}^{adv}, \delta^-) > 0$ then
6          $\delta^t \leftarrow \delta^{t-1} + \delta^-$, $\mathbf{x}^{adv} \leftarrow \mathbf{x}^{adv} + \delta^-$;
7      else
8          Solve $\min_{\delta \in \Delta_t^+ \cup \tilde{\Delta}_{t,k}^{\pm,p}} VoP(\mathbf{x}^{adv}, \delta)$ and get $\delta^+$;
9          if $VoP(\mathbf{x}^{adv}, \delta^+) > 0$ then
10             $\delta^t \leftarrow \delta^{t-1} + \delta^+$, $\mathbf{x}^{adv} \leftarrow \mathbf{x}^{adv} + \delta^+$;
11         else
12             return $\mathbf{x}^{adv}$
13 return $\mathbf{x}^{adv}$
```

6 OPTIMIZATION-BASED ATTACK: R3

In this section, we propose a novel attack framework called *Rounded Relaxation with Reparameterization (R3).* R3 solves the following relaxed optimization problem of (3) and round off the continuous solution to get the discrete perturbation.

$$\begin{aligned} \min_{\delta \in \mathbb{R}^n_{+,0}} \quad & f^{\mathbf{w}}(\mathbf{x}+\delta) \\ \text{s.t.} \quad & \|\delta\|_p \leq q. \end{aligned} \tag{6}$$

The main challenge here is that the constrained non-convex optimization is hard to solve. Though one can apply the projected gradient descent to ensure the solution is always feasible, unlike the box-constraint in the common L_∞ attacks, the projection operation is difficult under the complex L_1 and L_2 norm constraints. Thus, we propose the reparameterization tricks to transform the constrained optimization (6) to the equivalent unconstrained optimization by exploiting the property of L_1 and L_2 norms.

For the L_1 attack, we notice that the unit L_1 norm ball $\mathcal{B}_1 = \{\delta \in \mathbb{R}^n_{+,0} \mid \|\delta\|_1 = 1\}$ is the set of all n-dimensional probability distribution vectors. Thus, we can resolve the unit L_1 norm constraint in (6) with Softmax reparameterization as follows. Let $\mathbf{z} \in \mathbb{R}^n$ be the variable to reparameterize δ.

$$\delta_i = \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} \quad \forall i \in [n]$$

To generalize it to the L_1 norm within $[0,q]$, we define $\mathbf{r} \in \mathbf{R}^n$ and reparameterize δ as follows

$$\delta_i = q \cdot \frac{(tanh(r_i)+1)\cdot e^{z_i}}{2\sum_{j=1}^n e^{z_j}} \quad \forall i \in [n] \tag{7}$$

One can easily verify that δ defined in this way satisfies the L_1 norm constraint and every feasible δ can be reparameterized with suitable $\mathbf{r}$ and $\mathbf{z}$.

The L_2 norm constraint of δ can also be resolved with similar reparameterization trick using $\mathbf{r}$ and $\mathbf{z}$.

$$\delta_i = q \cdot \frac{(tanh(r_i)+1)\cdot e^{z_i}}{2\sqrt{\sum_{j=1}^n e^{2z_j}}} \quad \forall i \in [n] \tag{8}$$

With reparameterization, we solve the following unconstrained optimization which is equivalent with (6).

$$\min_{\mathbf{r},\mathbf{z} \in \mathbb{R}^n} \quad f^{\mathbf{w}}(\mathbf{x}+\delta) \tag{9}$$

where δ is in the forms of (7) and (8) for L_1 and L_2 attacks respectively.

We use the Adam optimizer to conduct gradient descent to solve (9). After the continuous solution δ is obtained, we round it into a discrete solution satisfying the L_p norm bound of q. The rounding process works as follows. We pick a threshold value η from $[0,1]$. Let $\tilde{\delta}$ be the solution of (9). For each entry $\tilde{\delta}_i$, if the residual value $\tilde{\delta}_i - \lfloor \tilde{\delta}_i \rfloor$ is larger than η, we set δ_i be $\lfloor \tilde{\delta}_i \rfloor + 1$ and otherwise, δ_i is assigned with $\lfloor \tilde{\delta}_i \rfloor$. The binary search is conducted on possible threshold values and output the final solution δ with $\|\delta\|_p$ closest to q. As there are at most n different residual values, the binary search can be done within $\log(n)$ rounds.

7 ADVERSARIAL TRAINING FOR IMPROVING ROBUSTNESS

In order to improve the robustness of the deep fraud detector, we incorporate adversarial examples into the training process (Algorithm 5), and solve the following robust optimization

$$\min_{\mathbf{w}} \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} [\max_{\delta \in \Delta^1_{q_1} \cup \Delta^2_{q_2}} L^{\mathbf{w}}(\mathbf{x}+\delta, y)] \tag{10}$$

where we omit the regularization term. Adversarial training (Algorithm 5) is a common practice to solve (10). The idea is to incorporate the adversarial examples into the training process and to approximate the optimization of the inner problem in (10). Noticing that both L_1 and L_2 attacks are considered in this paper, at each iteration, for each fraud instance in minibatch $\mathcal{B}$ of training data, we uniformly pick p from $\{1,2\}$ and generate one adversarial example with IFCM$_p$. All the generated adversarial examples are grouped by $\mathcal{B}^{adv}$, which is concatenated with $\mathcal{B}$ to form the new minibatch training data $\mathcal{B}'$. A training step is then conducted on $\mathcal{B}'$ with the following loss function

Algorithm 5: Adversarial training with IFCM

```
1 Randomly initialize the network for fraud detection;
2 repeat
3     Read minibatch $\mathcal{B} \subset \mathcal{D}$ from training set $\mathcal{D}$;
4     For each fraud instance in $\mathcal{B}$, randomly selects $p \in \{1,2\}$
        and create an adversarial example with IFCM$_p$, grouped
        into $\mathcal{B}^{adv}$;
5     Concatenate $\mathcal{B}^{adv}$ with $\mathcal{B}$ as $\mathcal{B}'$;
6     Conduct one training step over $\mathcal{B}'$ with loss $\tilde{L}$ in (11);
7 until training converged;
```

$$\begin{aligned} \tilde{L}^{\mathbf{w}}(\mathbf{x},y) =& \mathbb{1}_{\{y=0\}} L^{\mathbf{w}}(\mathbf{x},y) + (1-\gamma)\mathbb{1}_{\{y=1\}} L^{\mathbf{w}}(\mathbf{x},y) \\ &+ \gamma \mathbb{1}_{\{y=1\}} L^{\mathbf{w}}(\mathbf{x}^{adv},y) \end{aligned} \tag{11}$$

where γ is the parameter balancing the loss of the original data and adversarial examples.

8 EXPERIMENTAL EVALUATION

In this section, we present extensive experimental evaluation to illustrate the performance of our methods.

8.1 Experimental Settings

8.1.1 Dataset and Data Analysis. Our dataset is provided by TaoBao, one of the largest e-commerce platforms in the world. The dataset contains all transaction data in a period of 30 days. Each transaction is encoded as a feature vector to represent its statistics and characteristics, including various prices, the number of historical purchases on an item, the offline metrics on fraudulent suspicious of the buyer and item, and the statistics on a user's preference, depicted by the records of interactions on all categories in the past one month. We randomly sample 1.5 million fraud transactions and 1.5 million benign instances from the data in the first 25 days to form a balanced training set. We uniformly pick 300 thousand transactions from the records of the last 5 days as the test data, where the proportion of fraud instances to benign ones is around 1 : 30.

We provide some statistics on the preference features to give a better impression on the fraud detection problem. One key statistic is the L_1 norm of a transaction instance $\mathbf{x}$, that is, the total number of interactions recorded in the past month. We notice that in the fraud transactions, 66.8% have zero interaction, i.e., $\|\mathbf{x}\|_1 = 0$[6], while the percentage is only 3.9% in the benign transactions. According to the explanation from domain experts in TaoBao, those fraud instances with zero L_1 norms are likely to be generated by large amount of simple and cheap robots or scripts by the platforms, which are also easier to detect. The remaining fraud instances with non-zero L_1 norms correspond to the sophisticated scripts, such as the one in Figure 2, which make complicate operations to behave like humans. Table 1 presents the distribution of $\|\mathbf{x}\|_1$ of instances with non-zero L_1 norms, as well as the mean value of $\|\mathbf{x}\|_1$.

There are several points we would like to make regarding these statistics. First, though it seems reasonable to have two separate classifiers, one for instances with zero L_1 norm and the other for the remaining instances, empirical attempts show that it provides little improvement and brings extra vulnerability to adversarial attack. Thus, we use a single classifier as the detector for all instances. Second, despite the high amount of fraud instances with zero L_1 norms, the perturbation on those instances is meaningful as it indicates the potential that the adversary improves their simple bots with extra functions. Third, we also evaluate our attacks on fraud instances with non-zero L_1 norms separately, and the slight perturbation can still bypass the detector with high chance. For example, with $q = 20$ in L_1 attack, which corresponds to the distortion ratio of only around 5% given that the mean value of $\|\mathbf{x}\|_1$ is 377 for fraud instances with non-zero L_1 norms, the adversarial examples generated by AIS$_1$ can fool the detector with probability over 50%.

8.1.2 Model Training and Attack Settings. The deep fraud detector model is trained with the following parameters. The training batch size is 500 and the model is trained over 30000 batches. Adam [12] optimizer is used to conduct the stochastic gradient descent and the learning rate is set to be 0.0002. The regularization parameter α is set to be 0.01. The adversarial training is conducted with adversarial examples generated by IFCM$_1$ and IFCM$_2$, with $q = 30$ and $q = 6$ respectively. To test the attacking effectiveness of various algorithms, we consider $q \in \{10, 20, 30\}$ for the L_1 attack and $q \in \{4, 5, 6\}$ for the L_2 attack.

8.1.3 Baselines. We adapt several state-of-the-art continuous attacks to generate discrete perturbations, including FGSM [8], C&W attack [4], Logit-Space attack [2], and EAD attack [5].

FGM$_1$: FGM$_1$ is a single step L_1 attack method which selects the q coordinates with the minimal partial derivatives and adds one unit perturbation on each of them.

C&W attack: C&W attack is an optimization-based method primarily for continuous L_0, L_2, and L_∞ attacks on image classifiers. We adapt it to generate L_1 attack and use rounding techniques to obtain the discrete perturbation satisfying the norm constraint. C&W uses the Change of Variables (CoV) to replace δ with $\frac{q}{2}(tanh(\mathbf{w})+1)$ (element-wise) and solves the following continuous optimization problem

$$\min_{\mathbf{w}\in\mathbb{R}^n} c\cdot f(\mathbf{x}+\frac{q}{2}(tanh(\mathbf{w})+1)) + \|\frac{q}{2}(tanh(\mathbf{w})+1)\|_p$$

where $f(\mathbf{x}^{adv})$ is suggested to be defined as the difference of the logits, i.e., $f(\mathbf{x}^{adv}) = \max(z_1^{adv} - z_0^{adv}, 0)$ [4] with $\mathbf{z}^{adv}$ being the output of the last hidden-layer when the adversarial example $\mathbf{x}^{adv} = \mathbf{x}+\frac{q}{2}(tanh(\mathbf{w})+1)$ is the input. We replace it with $f(\mathbf{x}^{adv}) = z_1^{adv} - z_0^{adv}$ as z_1 tends to be larger than z_0 in our tested detector and the suitable threshold is around 0.9, rather than 0.5. Constant c is selected with binary search to ensure that $f(\mathbf{x}^{adv})$ is minimized to the extend while $\|\frac{q}{2}(tanh(\mathbf{w})+1)\|_p \leq q$ is still satisfied[7]. The optimization is solved with Adam optimizer. We pick the hyper-parameters that perform the best in solving the optimization in our tests, including the learning rate of 0.1 with 1000 gradient descent steps. 16 binary searches are conducted on $c \in [0, 100]$. The continuous solution δ is rounded off in a similar way as R3 attack. After the binary search returns the optimal c, we solve the optimization with 5 random starting-points of $\mathbf{w}$ and pick the best rounded discrete solution.

C&W-LS attack: Logit-Space attack [2] uses one-hot encoding to represent the discrete feature and applies the Softmax reparameterization to approximate the one-hot encoding. Formally, δ_i is represented by $Softmax(\mathbf{w}_i)\cdot(0, 1, ..., q)$ and the optimized variable is changed to $W = [\mathbf{w}_1, ..., \mathbf{w}_n]$. The optimization is similar to C&W attack, except that an entropy penalty is imposed to penalize δ_i being fractional. That is, C&W-LS solves the following optimization

$$\min_{W\in\mathbb{R}^{n\times(q+1)}} c\cdot f(\mathbf{x}+U(0,1,...,q)^T)+\|U(0,1,...,q)^T\|_p+\beta\cdot\sum_{i\in[n]} H(\mathbf{u}_i)$$

where $U = [\mathbf{u}_1, ..., \mathbf{u}_n]$, $\mathbf{u}_i = Softmax(\mathbf{w}_i)$, i.e., $u_{ij} = \frac{e^{w_{ij}}}{\sum_{j'=0}^{q} e^{w_{ij'}}}$ for $j \in \{0, 1, ..., q\}$, and $H(\mathbf{u}_i) = -\sum_{j=0}^{q} u_{ij}\log(U_{ij})$ is the entropy for $\mathbf{u}_i$ regarded as a distribution, which is minimized when $\mathbf{u}_i$ is one-hot. The implementation follows the same settings as C&W attack. We try different values of β and choose the best one in evaluation, which is 0.1 for L_1 attack and 0.5 for L_2 attack.

EAD attack: EAD attack [5] is a continuous L_1 attack on image classifier. It is also optimization-based attack which solves the following optimization problem with elastic-net regularization

$$\min_{\delta\in\mathbb{R}^n_{0,+}} c\cdot f(\mathbf{x}+\delta)+\beta\cdot\|\delta\|_1+\|\delta\|_2^2$$

We try different values of β and choose the best one in evaluation, which is 5.0.

We also try to round other attack algorithms such as DeepFool [17] for discrete L_2 attack. However, since DeepFool aims to find the successful attack with smallest L_2 distortion and thus imposes no norm restriction on perturbation. The rounding solution of DeepFool performs poorly. We also implement the random attack, which has little effect on degrading the performance of the

[6]We would like to remind the reader that for simplicity, we only consider the preference features to encode an instance, while in the deployed model, there are other non-zero features.

[7]Though in [4], the binary search on c aims to ensure the minimal value of c such that δ successfully fools the detector, it performs worse than our implementation when rounding the solution to the discrete one with L_p norm bound q. Thus, we implement C&W attack in the way explained here.

Type	$\|\mathbf{x}\|_1 \le 100$	$100 < \|\mathbf{x}\|_1 \le 200$	$200 < \|\mathbf{x}\|_1 \le 300$	$\|\mathbf{x}\|_1 > 300$	mean($\|\mathbf{x}\|_1$)
fraud	33.87%	16.43%	10.85%	38.85%	**377**
benign	45.33%	23.56%	12.8%	18.31%	**208**

Table 1: Statistics on fraud and benign instances with non-zero L_1 norms

classifier. Therefore, the results of these attempts are not presented in the paper.

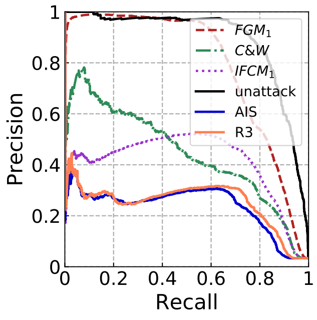
8.1.4 Evaluation Metrics. We evaluate various attacks and adversarial training based on two metrics, *success rate* and *Average Precision (AP)*. Success rate measures the proportion of fraud instances which are correctly recognized by the detector with pre-defined threshold η and mis-classified under the attack. AP is commonly adopted to measure the performance of a classifier, which is defined by the enclosing area of the Precision-Recall curve (PR-curve). For a threshold η, the precision and recall are defined as follows [19]

$$P(\eta) = \frac{tp}{tp + fp} \quad R(\eta) = \frac{tp}{tp + fn},$$

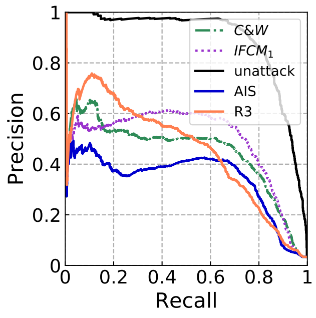
where tp, fp and fn denote the true positive rate, false positive rate and false negative rate respectively. A higher value of AP intuitively means that the detector can recall a large proportion of fraud instances and meanwhile mis-classifies a few benign instances. A stronger attack should be able to degrade the AP to a lower value. The effectiveness of adversarial training can also be measured by AP by verifying whether the obtained model can pertain the AP under strong attacks. PR-curve is also useful for the selection of threshold in deployment. In practice, we choose the threshold η^* which equals the precision and recall, and we use η^* when evaluating the success rate of attacks. One thing to notice is that, an attack degrading the AP to a lower value doesn't mean it has higher success rate at η^*, since AP is an average measure of performance over all threshold values. We now present our key experimental results.

8.2 Average Precision

Tables 2 and 3 show the AP of the original detector and the one obtained via adversarial training under various L_1 and L_2 attacks respectively. Figure 3 shows the PR-curves of the detector under L_1 attacks with $q = 30$ and L_2 attacks with $q = 5$. We denote by *unattack* the trails without adversarial perturbation, corresponding to the performance on the original test data. The results show that (i) The deployed detector is extremely vulnerable to adversarial perturbations, as with $q = 20$ and $q = 5$ for AIS$_1$ and AIS$_2$ respectively, the AP is decreased from nearly 0.9 to 0.434 and 0.344. The numbers further decrease to 0.238 and 0.207 for AIS$_1$ attack with $q = 30$ and AIS$_2$ attack with $q = 6$ respectively. (ii) Our attack methods always achieve the largest degrade of AP in all settings, and significantly outperform baselines, especially AIS and R3, which show the similar performance. (iii) With adversarial training, the detector becomes significantly more robust to adversarial perturbations, as we consider larger q values 40, 50 for L_1 attack and 7 for L_2 attack, and the model can still achieve an AP above 0.859 in all settings. Besides, the performance on the original test data is not degraded due to adversarial training, which shows the necessity of adversarial training.



(a) L_1 attack, $q = 30$



(b) L_2 attack, $q = 5$

Figure 3: PR-curves.

8.3 Success Rate

Tables 4 and 5 show the success rates of tested L_1 and L_2 attacks respectively on the original model with threshold value η^*. To better understand the effectiveness of attacks, we separate the fraud instances based on the L_1 norm and randomly pick 1000 fraud instances with zero L_1 norm and 1000 fraud instances with non-zero L_1 norm. The results show that (i) Both types of fraud instances are highly vulnerable to adversarial perturbations. For instances with zero L_1 norm, with distortion bounds of 20 in L_1 norm and 5 in L_2 norm, our *AIS* attack can craft the instance to bypass the detector with probabilities over 84%. This shows the potential threat for the deployed system as the real-world adversaries and platforms may exploit this property and design more sophisticated scripts and bots with only tens of additional operations to bypass the detector with high changes. While for instances with non-zero L_1 norm, with distortion bounds of 20 in L_1 norm, which corresponds to an average distortion ratio of only 5% given the average L_1 norm for this type of instance being 377, our AIS$_1$ attack can fool the detector with probability 54.1%. (ii) Our attack methods achieve the highest success rates in all settings, and outperform baselines more significantly for instances with zero L_1 norm.

8.4 Feasibility of Attacks

8.4.1 Feasibility of Adversarial Perturbations. The perturbations generated here are not artificial noises on the input. Instead, there do exist such cases where the malicious service platform intentionally creates such perturbations. Recall that in the motivating scenario, we show an automatic script (Figure 2) by a malicious service provider. The script can conduct operations to pretend it as a human. One interesting pattern noticed by domain experts is that, when the script scrolls the item list on the website or mobile Apps, it will not directly visit the target item to create fake impression. Instead, it is designated with certain probability of visiting

Attack	Original model			Model with adversarial training				
	$q = 10$	$q = 20$	$q = 30$	$q = 10$	$q = 20$	$q = 30$	$q = 40$	$q = 50$
unattack	0.892	0.892	0.892	0.89	0.89	0.89	0.89	0.89
FGM$_1$	0.817	0.798	0.787	0.882	0.885	0.892	0.902	0.901
C&W	0.763	0.516	0.442	0.889	0.892	0.897	0.897	0.9
C&W-LS	0.765	0.572	0.575	0.889	0.889	0.889	0.885	0.887
EAD	0.767	0.573	0.393	0.887	0.891	0.895	0.896	
IFCM$_1$	0.767	0.578	0.409	0.879	**0.876**	**0.875**	0.876	0.876
R3	0.745	0.463	0.251	0.881	0.877	**0.875**	**0.873**	**0.859**
AIS$_1$	**0.726**	**0.434**	**0.238**	**0.88**	**0.876**	**0.875**	0.874	0.873

Table 2: *Average Precision (AP)* of the original model and the model with adversarial training under L_1 attacks

Attack	Original model			Model with adversarial training			
	$q = 4$	$q = 5$	$q = 6$	$q = 4$	$q = 5$	$q = 6$	$q = 7$
unattack	0.892	0.892	0.892	0.89	0.89	0.89	0.89
C&W	0.71	0.492	0.238	0.882	0.889	0.893	0.887
C&W-LS	0.66	0.447	0.256	0.888	0.893	0.897	
IFCM$_2$	0.66	0.485	0.343	**0.877**	0.876	0.876	0.877
R3	0.68	0.463	**0.205**	0.894	0.895	**0.875**	0.877
AIS$_2$	**0.57**	**0.344**	0.207	0.878	0.877	0.876	0.875

Table 3: *Average Precision (AP)* of the original model and the model with adversarial training under L_2 attacks

Attack	$\|\|\mathbf{x}\|\| = 0$			$\|\|\mathbf{x}\|\| > 0$		
	$q = 10$	$q = 20$	$q = 30$	$q = 10$	$q = 20$	$q = 30$
FGM$_1$	5.4%	11.1%	13.4%	22.4%	34.1%	41%
C&W	10.1%	73.4%	75.4%	25.9%	47.4%	53.6%
C&W-LS	9.6%	45.5%	58.7%	25.7%	43.6%	44.7%
EAD	9.7%	58.5%	76.5%	26.6%	49%	60.7%
IFCM$_1$	9.7%	58.4%	90.4%	26.3%	49.4%	64.6%
R3	14.1%	81.4%	97.6%	27.9%	53.4%	**66.5%**
AIS$_1$	**19.3%**	**84.9%**	**97.8%**	**28.7%**	**54.1%**	66.4%

Table 4: *Success rate* of L_1 attacks

Attack	L_1 attack			L_2 attack		
	$q = 10$	$q = 20$	$q = 30$	$q = 4$	$q = 5$	$q = 6$
FGM$_1$	0.005	0.005	0.005	-	-	-
IFCM	0.037	0.072	0.103	0.057	0.088	0.125
C&W	372.5	399.8	480.1	285.4	317.9	329.2
C&W-LS	290.1	313.6	455.9	299.5	301.3	333.2
EAD	71.8		89.5	-	-	-
R3		137.8		119.2	119.7	185.2
AIS	39.9	125.6	129.1	74.4	173.7	69.9%

Table 6: Average runtime of generating one adversarial example (in seconds)

Attack	$\|\|\mathbf{x}\|\| = 0$			$\|\|\mathbf{x}\|\| > 0$		
	$q = 4$	$q = 5$	$q = 6$	$q = 4$	$q = 5$	$q = 6$
C&W	21.7%	64.4%	94.3%	37%	55.3%	68.7%
C&W-LS	28.2%	82.7%	95.3%	39.3%	58.7%	68.3%
IFCM$_2$	23.7%	78.4%	96.5%	39.2%	58.4%	69.8%
R3	28.8%	72%	94.5%	37.3%	56%	69.9%
AIS$_2$	**59.7%**	**89.8%**	**98.7%**	**45%**	**61.4%**	**71%**

Table 5: *Success rate* of L_2 attacks

another item from certain categories and return to the item list. Imagine that fraudulent sellers and providers recognize the adversarial perturbations to bypass the detector. They can easily design effective scripts or even hire human labors to conduct the perturbation. Furthermore, we also sample adversarial perturbations and the domain experts verify that they are doable by the fraudulent sellers or malicious service providers.

8.4.2 Scalability of Attacks. Table 6 lists the average runtime of generating one adversarial example with various attacks. Notice that different attacks have varying proportions of computations able to be parallelized. Thus, the runtime of crafting a single instance cannot reflect the runtime of processing a batch of samples. From the adversary's perspective, our attacks are feasible as they can generate an adversarial examples within several minutes. Besides, one extra note on AIS is that it belongs to the *black-box* attacks as no information of the architecture of the detector is required. Thus, we argue that AIS could generate adversarial instances that are practical for real-world adversaries to craft.

9 CONCLUSION

This paper reveals the potential risk of deploying deep learning in fraud detection. First, we propose a novel attack called *Augmented Iterative Search* to handle the discrete perturbation. AIS exploits the structure of perturbation space to effectively search for adversarial example with L_1 norm bounded distortion. Second, we strengthen the deep fraud detector via adversarial training. Third, we conduct extensive evaluations on our attack method and the adversarial training process on the deep fraud detector deployed by one of the largest e-commerce platforms in the world. The results show that (i) the deployed fraud detector is vulnerable to the feasible slight perturbations, as our AIS attack successfully degrades its average precision from over 90% to as low as 40%, (ii) our AIS attack significantly outperforms baselines which generates the adversarial examples with lower prediction scores and higher success rates under all tested settings, and (iii) via adversarial training, the robustness of the model is significantly improved and can achieve an average precision above 89% under strong attacks.

REFERENCES

[1] Anish Athalye, Nicholas Carlini, and David A. Wagner. 2018. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*. 274–283.

[2] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. 2018. Thermometer encoding: One hot way to resist adversarial examples. (2018).

[3] Nicholas Carlini, Guy Katz, Clark Barrett, and David L Dill. 2017. Provably minimally-distorted adversarial examples. *arXiv* 1709 (2017).

[4] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP'17)*. 39–57.

[5] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 2018. EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI'18)*. 10–17.

[6] Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang. 2016. Credit Card Fraud Detection Using Convolutional Neural Networks. In *Neural Information Processing - 23rd International Conference (ICONIP'16)*. 483–490.

[7] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep Learning*. Vol. 1.

[8] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *CoRR* abs/1412.6572 (2014).

[9] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick D. McDaniel. 2016. Adversarial Perturbations Against Deep Neural Networks for Malware Classification. *CoRR* abs/1606.04435 (2016).

[10] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015).

[11] Earlence Fernandes Tadayoshi Kohno Bo Li Atul Prakash Amir Rahmati Ivan Evtimov, Kevin Eykholt and Dawn Song. 2017. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945* (2017).

[12] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[13] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *CoRR* abs/1607.02533 (2016).

[14] Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. 2015. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*. 4749–4757.

[15] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. *CoRR* abs/1706.06083 (2017).

[16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal Adversarial Perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*. 86–94.

[17] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 2574–2582.

[18] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*. 807–814.

[19] David L Olson and Dursun Delen. 2008. *Advanced Data Mining Techniques*.

[20] Nicolas Papernot and Patrick D. McDaniel. 2016. On the Effectiveness of Defensive Distillation. *CoRR* abs/1607.05113 (2016).

[21] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *IEEE European Symposium on Security and Privacy (EuroS&P'16)*. 372–387.

[22] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *IEEE Symposium on Security and Privacy (SP'16)*. 582–597.

[23] Herbert Robbins and Sutton Monro. 1985. A stochastic approximation method. In *Herbert Robbins Selected Papers*. 102–109.

[24] Andrew Slavin Ross and Finale Doshi-Velez. 2018. Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing Their Input Gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI'18)*. 1660–1669.

[25] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*. 1528–1540.

[26] Ning Su, Yiqun Liu, Zhao Li, Yuli Liu, Min Zhang, and Shaoping Ma. 2018. Detecting Crowdturfing "Add to Favorites" Activities in Online Shopping. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web (WWW'18)*. 1673–1682.

[27] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *CoRR* abs/1312.6199 (2013).

[28] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick D. McDaniel. 2017. Ensemble Adversarial Training: Attacks and Defenses. *CoRR* abs/1705.07204 (2017).

[29] Haiqin Weng, Zhao Li, Shouling Ji, Chen Chu, Haifeng Lu, Tianyu Du, and Qinming He. 2018. Online e-commerce fraud: A large-scale detection and analysis. In *34th IEEE International Conference on Data Engineering (ICDE'18)*.

[30] Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *25th Annual Network and Distributed System Security Symposium (NDSS'18)*.

[31] Xiaoyong Yuan, Pan He, Qile Zhu, Rajendra Rana Bhat, and Xiaolin Li. 2017. Adversarial Examples: Attacks and Defenses for Deep Learning. *CoRR* abs/1712.07107 (2017).

[32] Ashkan Zakaryazad and Ekrem Duman. 2016. A profit-driven Artificial Neural Network (ANN) with applications to fraud detection and direct marketing. *Neurocomputing* 175 (2016), 121–131.

[33] Panpan Zheng, Shuhan Yuan, Xintao Wu, Jun Li, and Aidong Lu. 2018. One-Class Adversarial Nets for Fraud Detection. *CoRR* abs/1803.01798 (2018).

[34] Stephan Zheng, Yang Song, Thomas Leung, and Ian J. Goodfellow. 2016. Improving the Robustness of Deep Neural Networks via Stability Training. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 4480–4488.