AUROR: Defending Against Poisoning Attacks in Collaborative Deep Learning Systems

Shiqi Shen Shruti Tople Prateek Saxena
National University of Singapore
{shiqi04, shruti90, prateeks}@comp.nus.edu.sg

ABSTRACT

Deep learning in a collaborative setting is emerging as a cornerstone of many upcoming applications, wherein untrusted users collaborate to generate more accurate models. From the security perspective, this opens collaborative deep learning to poisoning attacks, wherein adversarial users deliberately alter their inputs to mis-train the model. These attacks are known for machine learning systems in general, but their impact on new deep learning systems is not well-established.

We investigate the setting of indirect collaborative deep learning — a form of practical deep learning wherein users submit masked features rather than direct data. Indirect collaborative deep learning is preferred over direct, because it distributes the cost of computation and can be made privacy-preserving. In this paper, we study the susceptibility of collaborative deep learning systems to adversarial poisoning attacks. Specifically, we obtain the following empirical results on 2 popular datasets for handwritten images (MNIST) and traffic signs (GTSRB) used in auto-driving cars. For collaborative deep learning systems, we demonstrate that the attacks have 99% success rate for misclassifying specific target data while poisoning only 10% of the entire training dataset.

As a defense, we propose AUROR*, a system that detects malicious users and generates an accurate model. The accuracy under the deployed defense on practical datasets is nearly unchanged when operating in the absence of attacks. The accuracy of a model trained using* AUROR *drops by only 3% even when 30% of all the users are adversarial.* AUROR *provides a strong guarantee against evasion; if the attacker tries to evade, its attack effectiveness is bounded.*

1. INTRODUCTION

Deep learning techniques have brought a paradigm shift in data driven applications, from speech recognition (e.g., Apple's Siri [4], Google Now [7], Microsoft's Cortana [6] and recently Facebook M [3]), to image identification (e.g., Google's Photos [5], Facebook's Moments [2]). Several security applications like spam filtering [9], malware detection [23, 25] and others use neural network algorithms as their back-bone. However, to attain reasonable accuracy in these systems, deep learning algorithms require extensive training datasets. This constraint restricts the widespread use of learning techniques in many applications. To address this limitation, collaborative deep learning is emerging as a popular technique to gather data from different sources and combine them to generate larger datasets. For example, asking users to provide movie ratings, tagging photos, crowd sourcing, marking emails as spam are widely adopted ways to generate large and varied datasets [8, 11].

Direct collaborative learning, i.e., training directly on user provided data has two major concerns. First, submitting personal information to external parties exposes users to privacy risks since they cannot control how their data is used after sharing. Even passive inference attacks are a known threat; for example, anonymized movie ratings can reveal enough information to de-anonymize users when combined with public data sources [31]. The second concern is the vast amount of computation time (over few weeks) necessary to train the learning algorithms on large datasets. *Indirect collaborative learning* addresses both these concerns [35, 39, 43]. In this technique, each user computes partially on its data to generate *masked features*. Instead of the original data, all the users submit the masked features to the server. The server performs the remaining computation of the learning algorithm on the masked features to generate a *global* model, thereby guaranteeing each user's data privacy and substantially reducing the computation costs on the centralized server.

Indirect collaborative learning, unlike direct collaboration offers a weaker adversarial setting. The adversary in this setting can only poison its own local data without observing the training data of other users. Moreover, the poisoned data only influences the global model indirectly via the masked features. Although, the training process becomes privacy-preserving and cost-efficient due to distributed computation, as we highlight, it remains susceptible to poisoning attacks. Malicious / adversarial users can poison or tamper the training data to influence the behavior of the global model [42]. However, the severity of poisoning attacks in indirect collaborative deep learning is not yet well-understood. Thus, understanding whether such restricted poisoning via masked features can influence the global model substantially is important. As our first contribution in the paper, we study the efficacy of poisoning attacks to varying levels of poisoning in state-of-the-art indirect collaborative deep learning systems.

To understand the effect of these attacks, we test a state-of-the-art indirect collaborative deep learning system which incorporates differential privacy techniques [39]. We test using two popular datasets and demonstrate targeted poisoning attacks that influence the global model, misclassifying specific target data (e.g., digit 1 as 3). We select the well-known MNIST images for handwritten characters as the first dataset for this system [24]. Using the at-

ACSAC '16, December 05-09, 2016, Los Angeles, CA, USA
 ISBN 978-1-4503-4771-6/16/12...$15.00
DOI: http://dx.doi.org/10.1145/2991079.2991125

tack strategy of mislabeling dataset, we show that when 10% of all the participants are malicious, the attack success rate is upto 99%. Moreover, for a total of 10 output classes, the average accuracy of the global model drops by 24% as compared to the model trained with benign dataset when 30% of users are malicious. As our second dataset, we select German traffic sign images (GTSRB) that has 43 output classes [40]. Our attack exhibits 79% success rate for specific target and the accuracy drops by about 9% when 30% of users are malicious. Our experiments show that the attacker's advantage increases with fewer output classes in the system.

Existing defenses against poisoning attack focus specifically on non-deep learning algorithms and assume access to complete training datasets. Nelson et. al [32] and Mozaffari-Kermani et. al [29] assume the existence of a golden model generated from a trusted dataset and classify the training data as malicious or benign based on its effect on the accuracy of this model. However, in our setting, we do not have access to such a pre-defined trusted model. Muhlenbach et. al [30] propose to identify malicious training data based on the label of their neighboring data values. This method, however, requires access to the entire training dataset beforehand. Hence, previous defenses are insufficient and cannot be re-purposed in a straightforward manner to prevent poisoning attacks in indirect collaborative deep learning systems. This problem raises the question *— is it possible to defend against poisoning attacks without having access to the complete training data?*

In this paper, we propose a defense against poisoning attacks that does not require the availability of entire training data. To this end, we design AUROR— an automated defense that filters out malicious users based only on their masked features. The key insight in designing AUROR is that poisoning of training data strongly influences the distribution of the masked features learnt by the system. Since each masked feature corresponds to a different information in the training data, the main challenge lies in identifying which set of masked features are affected due to poisoning of the dataset. Thus, AUROR involves two key steps a) identifying relevant masked features corresponding to the attack strategy and b) detecting malicious users based on the anomalous distribution of the masked features. Our solution provides a strong accuracy guarantee against arbitrary poisoning of data by all the colluding malicious users.

We implement our defense and evaluate its effectiveness on real datasets for deep learning systems. We employ AUROR to identify malicious users in our attack set up. The detection rate for identifying malicious users in both MNIST and GTSRB datasets is 100% for a fraction of malicious users between 10% to 30%. We measure the accuracy drop in the trained model after removing the malicious users detected using AUROR. The trained model exhibits a small accuracy drop ranging from 0% to 3% as compared to benign setting for both the datasets. Thus, AUROR trained model provides accuracy guarantees comparable to benign datasets. Our experiments demonstrate that even an optimal adversarial strategy cannot do any better in degrading this guarantee.

Contributions. We make the following contributions in this paper:

- ***Measuring Sensitivity of Poisoning Attacks.*** We empirically demonstrate that poisoning attacks are a serious threat to indirect collaborative deep learning systems regardless of masking the essential features of the training data and restricted poisoning capacity of the adversary.
- **AUROR.** We introduce a statistical mechanism called AUROR as a countermeasure to poisoning attacks in indirect collaborative deep learning systems. AUROR automates the process of identifying masked features exhibiting abnormal distribution and detects malicious users in the system based on these features.

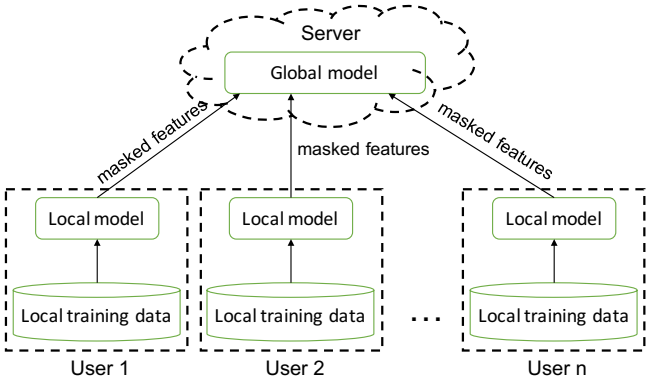

Figure 1: The indirect collaborative learning setting. The users compute a local model and submit masked features to the server. The server computes a global model based on all the abstract features.

- ***Evaluation.*** Our empirical evaluation validates that AUROR can identify malicious users with almost 100% detection rate, while limiting accuracy drop to under 3% for the final model even when 30% users are malicious. AUROR thus enables accurate and robust indirect collaborative learning systems.

2. OVERVIEW

Our goal is to understand the impact of targeted poisoning on deep learning systems and investigate a practical defense that is resistant to evasion.

2.1 Problem Setting

Several enterprises employ collaborative learning to understand their customer's behavior towards their products [8, 26]. They collect data from the users in the form of reviews, feedback or other attributes. In this work, we refer to the entity that collects the data and processes learning algorithms over them as *server*. The server does not necessarily need to be a centralized entity. The tasks of the server can be performed in a distributed manner among the participants. The participants or customers that submit their data to the server are referred as *users*. The users can be individuals, groups or companies with their own dataset. A user does not know or learn any direct information about the training data of other users.

In this paper, we specifically examine the indirect collaborative learning setting (as shown in Figure 1). This setting differs from the direct collaborative learning in the way in which users upload their data. Instead of submitting the raw data directly to the server, users mask some information about their data and send it to the server. This saves both the bandwidth (data costs) and yields better privacy which is essential for practical adoption of any new technique. We refer to the masked information as *masked features*. The users compute a *local* model on their machines that generates the masked features. The server collects these masked features from all the users and performs operations (for e.g., summation) to generate a *global* trained model. This global model captures the features from the entire dataset, thus has a very high accuracy. Lastly, we use the global model to perform the classification of the test dataset.

We select the privacy preserving deep learning (PPDL) system by Shokri et. al [39] as the classification system. The PPDL system performs image recognitions by indirect collaborative learning setting as shown in Figure 1. This system uses differential privacy techniques to mask the features before submitting them to

the server. PPDL uses a distributed selective stochastic gradient descent (SSGD) technique to distribute the computation between the local and global model. Users of the system compute gradients from the input data and upload them as features of the data. Section 3.2 gives a detailed explanation about their system and its working.

2.2 Threat Model

In the above discussed indirect collaborative learning setting, we consider that a constant fraction (f) of the users are *malicious*. These malicious users are incentivized to modify or *poison* the training dataset to affect the accuracy of the global model. For example, spammers can mark genuine emails as spams, thereby reducing the accuracy of the learnt spam filter. Such a spam filter will then assign emails with genuine content as spam resulting in reduced credibility of the filter among its users. In this model, a majority of the users are honest while a small fraction f (such that $f < n/2$) of the users are malicious. This is a rational setting, for example, consider the case where a product company bribes users to give higher ratings to their products on an e-commerce website [1, 10]. Although, some of the users are compromised, the total number of participants is much higher, resulting in benign users forming a majority. We consider that all the adversaries know the learning algorithm used by a particular application and are able to tamper with the actual data accordingly (e.g., either by adding fake data or mislabeling the training data). Moreover, all the malicious users can collude together to poison their training data. However, the adversary cannot learn the global model beforehand as the malicious users have no knowledge about the training data generated from the benign users.

The server in our model is honest-but-curious i.e., it passively observes the data gathered from the users to learn information about it. Our model directly inherits privacy guarantees from the previous work [39]. The server cannot infer information about the original data as all the users submit masked features. Our attack does not break the privacy guarantees of the previous system but only influences the accuracy of the final global model.

2.3 Problem Definition

Targeted Poisoning Attacks. In non-deep learning systems, several known adversarial learning techniques reduce the accuracy of the global model. For example, attackers can modify the test data to avoid the detection of malicious samples on a well-trained model [13, 16, 21, 33, 34, 41, 46] or alter the training data to affect the model accuracy during the learning phase [17, 36, 42]. In this paper, we specifically focus on the latter problem known as causative or poisoning attacks caused by tampering with the training data [14, 22, 27] on deep learning systems. Depending on the attacker's motive, poisoning attacks can be of two kinds a) random attacks and b) targeted attacks [22]. Random attacks are perpetrated with the intention to reduce the accuracy of the model, whereas targeted attacks are performed with a more focused objective. Specifically, in targeted attack, the attacker's goal is, for a given *source* data, the model M outputs *target* data of the attacker's choice. For example, an adversary can poison the training data to classify a source data (e.g., a spam email) as a target data (e.g., a genuine email).

Definition 2.1. *(***Source Data***) Source data is any input value to the global model for which the attacker wants to influence the model's output.*

Definition 2.2. *(***Target Data***) Target data is a value of attacker's choice that the influenced model should output for a given source input.*

The attacker arbitrarily chooses the source and target data values first and then poisons its training data known as the *poison set*. The poisoning strategy depends on the underlying learning algorithm used by the system. The final poisoned model M_P represents the features from the entire training data provided by malicious as well as benign users. We define the success rate of attack as follows:

Definition 2.3. *(***Attack Success Rate***) The attack succeeds if the poisoned model outputs the desired target value T for a source input I, $M_P(I) \rightarrow T$ and sets $S_I = 1$ otherwise the attack fails with $S_I = 0$ where S_I is used to indicate whether the target attack succeeds. The success rate (SR) of the attack for the model M_P is given as:*

$$SR_{M_P} = \frac{(\Sigma_{I \in D} S_I)}{|D|} \times 100$$

where D is the domain of all possible source inputs.

Along with the output of the source data, targeted poisoning effects the overall accuracy of the global model. We measure this effect on the accuracy by calculating the accuracy drop due to poisoning on dataset.

Definition 2.4. *(***Accuracy Drop***) Accuracy drop (AD) in a model M_P due to poisoning is defined as the difference in the overall accuracy of a model trained using benign dataset and malicious dataset, specifically as below:*

$$AD_{(M_B \rightarrow M_P)} = A_{M_B} - A_{M_P}$$

where A_{M_B} and A_{M_P} is the accuracy of benign model M_B and malicious model M_P respectively.

Higher is the value of accuracy drop $AD_{(M_B \rightarrow M_P)}$, larger is the influence of poisoning on the global model. Clearly, performing targeted attacks is more difficult than random attacks, as the attacker has to poison the training data with a specific goal. It is even harder to perform targeted attacks in collaborative learning where each user contributes a small portion of data towards the entire training dataset. Since all the previous poisoning attacks are performed in direct collaborative machine learning setting [42], it is not well-understood if targeted attacks are equally effective when the users submit only their masked features. As a first step towards this direction, we explore the effectiveness of targeted poisoning attacks at various levels of poisoning in indirect collaborative learning and study the efficacy of these attacks on deep learning systems.

Defense Solution. As a preventive measure, we design a defense for poisoning attacks in the indirect collaborative learning setting. To thwart these poisoning attacks, the server should distinguish malicious users among all the participants and exclude the features contributed by these users while training the global model. This detection and elimination strategy ensures that the global model is not influenced due to the poisoned data. Thus, any server that employs this defense before computing the global model can guarantee a robust and accurate model even under attack by a fraction f of malicious users. However, the challenge in designing this defense lies in correctly detecting the malicious users. One method is to observe the labels of samples of all users and detect discrepancies between them, as suggested by previous work [30]. The difficulty of identifying malicious users escalates when the server does not have access to the entire original training data but can only observe the masked features of the data. In this work, we investigate the problem of designing AUROR— a defense that thwarts poisoning attacks in indirect collaborative learning setting without access to training data. A global model trained using AUROR M_A is robust

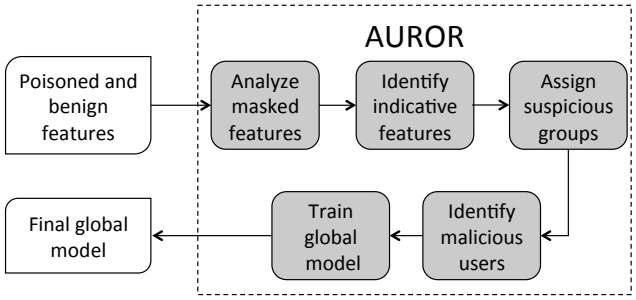

Figure 2: The design details and steps involved in AUROR. AUROR takes the masked features from users as input and produces a the final accurate global model.

and effective if the attack success rate SR_{M_A} and the accuracy drop $AD_{(M_B \to M_A)}$ of M_A are small enough to be acceptable for practical purposes.

Research Questions. In summary, we aim to answer the following research questions:

- RQ1: What is the impact of targeted poisoning attacks in indirect collaborative deep learning?
- RQ2: Can we effectively thwart poisoning attacks on deep learning systems without having access to the entire training dataset?

2.4 Our Approach

We propose AUROR— a defense against poisoning attack in indirect collaborative learning setting that can detect malicious users with high accuracy, thereby building a robust global model. AUROR does not need access to raw dataset and computes only on submitted features. In designing AUROR, we highlight several key observations that allow us to detect the malicious users.

Key Observations. Our first observation is that whatever be the underlying distribution δ of the training data, the distribution of the masked features δ' preserves the distribution in the benign setting, $\delta \sim \delta'$. The second observation is that poisoning on the training data directly affects the distribution pattern of some specific masked features referred to as *indicative* features. Let i be an indicative feature then in an attack setting $\delta(i) \nsim \delta'(i)$ within statistical error bounds. Thus, in an attack scenario, indicative features from majority of honest users will exhibit a similar distribution while those from malicious users will exhibit an anomalous distribution. As long as the fraction of malicious users is within range ($\epsilon < f < n/2$), AUROR performs this detection correctly. Here, ϵ is the lower bound on the number of malicious users for which AUROR can perform accurate detection. The value of ϵ varies with the type of underlying dataset so that we need to determine the value of ϵ for each dataset.

AUROR **Design.** Figure 2 shows the steps involved in designing AUROR. It follows a deterministic algorithm and has no knowledge about the attacker's goal i.e., source or target values. AUROR is applied on the training data before the online testing phase of the model. AUROR takes masked features as input from both benign and malicious users. At first, it analyzes the distribution of the masked features from all the users to identify the indicative features. The key challenge is to extract the indicative features that change distribution in an attack setting. Depending on the type of dataset, different features vary due to poisoning of the input dataset. To decide whether a particular feature is indicative or not, it groups the values for that feature uploaded by all users into different clusters. If the distance between the centers of two clusters is greater than a threshold value α, the feature is marked as indicative. The value of α depends on the original distribution δ of the training dataset.

After this, it detects malicious users based on the anomalous distribution of the indicative features. To identify this anomalous distribution, it clusters the users based on their indicative features. The cluster which has majority participants is marked as honest, whereas the other cluster is marked as suspicious. All the users in the suspicious group are suspected to be malicious, but are unconfirmed. AUROR marks a user in the suspicious group as malicious if it appears for more than τ times. The parameter τ controls the error-rate tolerance of the benign users whose fraction of the indicative features show distribution similar to malicious users due to statistical error. AUROR selects the value of τ automatically based on the input. Note that malicious users will have an anomalous distribution for most of the indicative features and hence will always appear in suspicious groups. Thus, the value of τ depends on the total number of indicative features for that training data and differs for each application.

Finally, AUROR trains the global model excluding the features submitted by malicious users to produce M_A. Thus, AUROR automatically generates a final accurate global model given a set of input features and the learning algorithm. For our case studies we show that both the metrics, i.e., the attack success rate SR and accuracy drop AD of M_A are very small. This demonstrates that AUROR is a promising defense for indirect collaborative learning applications.

Difficulty in Evading AUROR**.** To evade detection, malicious users have to ensure that AUROR cannot distinguish between malicious and benign users. For achieving this, the attacker has to prevent AUROR from finding indicative features among all the masked features i.e., limit the deviation of indicative features due to poisoning of dataset to less than α. The adversary can avoid deviating the distribution by using two strategies.

First, it can limit the fraction of malicious users below the lower bound of ϵ. Due to a small fraction of malicious users, the impact on the distribution of indicative features is small as well. Second, it can limit the poison set for each of the adversarial users i.e., instead of poisoning the entire dataset the attacker can partially poison its dataset to reduce the deviation in distribution of indicative features. Reducing the poison set or fraction of malicious users will keep the distribution of indicative features within statistical error bounds of the benign features thus evading detection from AUROR. However, we claim that for both these strategies the success rate of the attack in influencing the global model drops significantly. We experimentally verify our claim and show that attackers can not successfully perform poisoning attacks while simultaneously evading our defense. Thus, AUROR is robust against evasion attempts and is a strong defense eliminating malicious users.

3. BACKGROUND

Indirect collaborative deep learning is essential in improving the model accuracy while enhancing the performance and privacy via distributed computation. We first provide a background of deep learning algorithms and then a state-of-the-art solution that computes these algorithms in an indirect collaborative setting.

3.1 Deep Learning

Multilayer Perceptron. Multilayer perceptron (MLP) is a direct network where each node is fully connected with the node in the

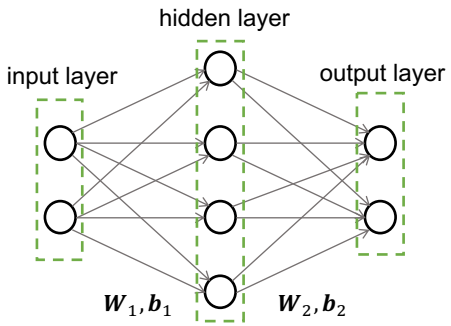

Figure 3: Deep learning architecture. $\mathbf{W_i}$ and $\mathbf{b_i}$ are the weight parameters and bias parameters respectively for layer i.

next layer. Figure 3 shows a simple multilayer perceptron architecture with one hidden layer. Each node is a computing unit having a non-linear activation function and generates an output value, with the exception of the nodes at the input layer. The activation function of a layer multiplies the output vector of previous layer with its weight matrix and then adds the bias vector. For example, the output vector $\mathbf{x_i}$ for layer $i (i > 1)$ is represented as,

$$\mathbf{x_i} = f_i(\mathbf{W_i x_{i-1}} + \mathbf{b_i})$$

where $\mathbf{W_i}$, $\mathbf{b_i}$ and $f_i(\cdot)$ are weight matrix, bias vector and activation function respectively for layer i. The bias vector $\mathbf{b_i}$ enhances the representation capability of the network. In its absence, the model will be restricted in its expressiveness. The weight matrix and bias vector are denoted together as parameter $\mathbf{P}$. This parameter $\mathbf{P}$ represents the contribution from each input value towards the learnt model. Note that, for the first input layer, the output vector is same as the raw input data.

To determine the parameter $\mathbf{P}$, multilayer perceptron uses back-propagation technique with the gradient descent method [37]. The gradient descent technique calculates the gradient value over the entire training data in single iteration. However, this method does not scale to large training datasets. Alternatively, stochastic gradient descent [18] divides the whole training data into small subset of training data called min-batch, and trains the model on each min-batch. A single iteration of stochastic gradient descent operates over the min-batches of the entire dataset. For example, if there 100 samples in the complete dataset and the min-batch size is 10 samples, then one iteration trains on 10 min-batches.

Convolutional Neural Network. The traditional MLP model has very high dimension because of the fully connected network and hence does not scale well in terms of performance. The convolutional neural networks (CNN) model is a type of multilayer perceptron model which requires minimal amount of processing because of its structure. In a CNN model, the layers in the network are not necessarily fully connected. This structure reduces the number of parameters used to train the model thereby making the training process faster. For layers that are not fully connected, each node only connects with a small region of the nodes in previous layer, like a 3×3 region. This region is called as local receptive field for the node.

In addition to decreasing the number of parameters, when sliding the local receptive field across all input values with certain stride, we will apply the node with same weights and bias though all local receptive field, which is called as parameter sharing scheme. This scheme is based on an assumption that if a feature is useful in one spatial region, then it should also be valuable for other regions. The shared parameters are defined as a kernel or filter. To enhance the representation capability of network, we increase the number of nodes in the convolutional layer, which is also called as the depth of the layer. Pooling layer is another periodically existed layer in CNN after successive convolutional layers. Most commonly, we use the MAX pooling layer with filters of size 2×2. Pooling layer can progressively reduce the amount of parameters and computation for training the network.

In this paper, we use both the traditional MLP and CNN model, which are widely used in deep learning, to perform poisoning attacks and apply our defense on them.

3.2 Privacy Preserving Deep Learning

For performing attacks, we select the privacy-preserving model for image recognition using deep learning proposed by Shokri et al. [39], which is a state-of-the-art system that provides indirect collaborative learning. The model uses stochastic gradient descent method to compute gradient values. Each user trains its own model based on its training dataset and generates the gradient value for each iteration. They mask these gradient values using the noise generated from Laplace distribution, making the local output differential private before submitting to the server. We refer to these masked gradient values as masked features.

The server collects the masked gradient values from all the users and aggregates them to update the global model. After updating the global model, the server generates an updated set of parameters from the global model that capture the features from the entire training dataset. The users download these updated parameters and provide them as input to their local model, thus generating different gradient values. This process is repeated several times until the global model stabilizes.

4. TARGETED POISONING ATTACKS

We study the efficacy of targeted poisoning attacks using two image recognition tasks in indirect collaborative learning. The image recognition system uses deep learning algorithm as its underlying technique to train a global model. The final global model classifies the test images into a given set of categories. In this system, poisoning attacks aim to classify a source test image as a target image. We perform our attacks with the following goals:

- To understand the amount of poisoning necessary to achieve a reasonable attack success rate
- To understand the difficulty levels for performing targeted attacks with different goals
- To understand the impact of poisoning attacks on the accuracy of the global model

4.1 Handwritten Digit Images

Dataset. We use the MNIST dataset of handwritten digits [24], a popular benchmark for training and testing deep learning models [38]. The dataset provides 60,000 training samples and 10,000 test samples where around 1000 test samples correspond to each digit. Each image has 28×28 pixels, which is flattened into a vector with 784 features and given as input to the deep learning model. To assign a value for each feature, integer numbers from 0 to 255 that represent different shades of grey are transformed to floating point values. Hence, each feature has a floating point value between 0 and 1.

Network Architecture. We use a simple multilayer perceptron neural network to train the classifier for handwritten digits. There are only two hidden layers using Tanh function as its activation

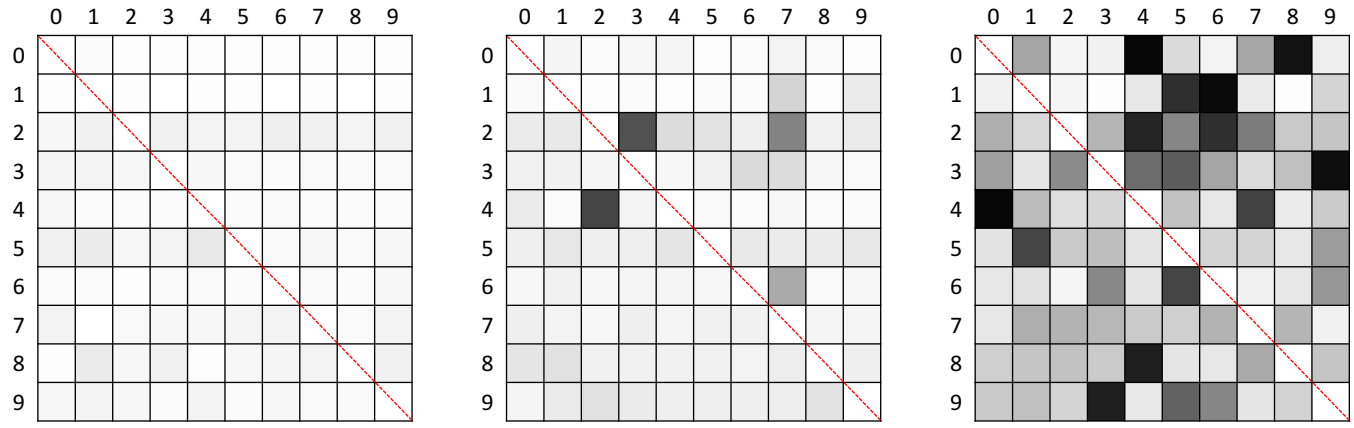

Figure 4: Attack success rate for classifying different pairs of source and target digits when fraction of malicious users is 30%, 20% and 10% respectively. The entry in each row represents the source label while the entry in each column represents the target label. The gray value in each cube represents the success rate. Black represents 0%, while white represents 100% attack success rate. The success rate is the highest when the malicious ratio is 30%.

Fraction of Malicious users (%)	Accuracy Drop (%)		
	Min	Max	Avg
10%	2	15	9
20%	7	27	16
30%	15	68	24

Table 1: Average accuracy drop, maximum accuracy drop and minimum accuracy drop for MNIST due to poisoning attacks as compared to benign model.

function in the network. For the output layer, the activation function is a Softmax function that converts the output of previous layer to a digit between 0 to 9. We divide the 60,000 training samples among 30 users, each of them having 2000 samples. Every user computes the local model based on these 2000 training samples.

Attack Strategy. To measure the severity of targeted poisoning attacks on above outlined system, we perform several experiments by varying the fraction of malicious users f in this setting. The malicious users range from 10% to 30% of all the users as we consider $f < n/2$. In each experiment, the malicious users poison their training data with the goal of influencing the global model such that it classifies a source digit (e.g., 1) as a target digit (e.g., 3). Each malicious user extracts all the samples corresponding to the source digit from its training data and ignores the images for the remaining digits. Further, it replicates the source digit images to occupy the entire training space of the user (2000 samples) and mislabels them as the target digit. The malicious user uses this poisoned training data to compute the gradients from its local model. We run the experiments for all the possible sources (1-10) and target (1-10) digits.

Attack Findings. To measure the attack success rate, we test the global model with the samples for each digit in the test dataset. Figure 4 shows the attack success rate for classifying each of the source digits (e.g., 0) as the other digits (1 to 9) when fraction of malicious users is 30%, 20% and 10%. The graph uses different level of gray value with white for the largest value (90% and above) and black for the lowest value (0%) to represent the attack success rate. We observe that when the fraction of malicious users is 30% the attack success rate for all the combinations vary between 89% to 100% with 96% average attack success rate. For 20% malicious users, we observe that the attack success rate reduces for specific targets but is still high for majority of source to target digits with 91% average attack success rate. The success rate greatly varies for different targets when the malicious fraction is reduced to 10%. For instance, the attack success rate is only 3% for mislabeling 4 as 0, while 94% for mislabeling 0 as 6. Other digits demonstrate a similar behavior. This phenomenon is an extrapolation of the effect due to distance between the hyperplane of source and target digits as explained in [34]. The average attack success rate increases from 65% to 96% with the increase in the fraction of malicious users from 10% to 30%.

Lastly, we test all the 10,000 test samples and measure the accuracy of the global model. In our experiments, the accuracy of the model when trained under completely benign training data is 86%. Table 1 shows the accuracy drop for the global model due to poisoning of the training data as compared to benign dataset for different fraction of malicious users. The maximum accuracy drop is as high as 68% (e.g. mislabeling 7 to 1) and minimum is 15%. In addition, the average accuracy drop for various targeted poisoning attack is 24%, 16% and 9% for 30%, 20% and 10% of malicious users respectively. This shows that the accuracy drop increases with increase in the fraction of malicious users. Hence, our experiments demonstrate that targeted poisoning attacks not only influence the classification of the source data but also affect the overall accuracy of the model.

4.2 German Traffic Sign Benchmarks

Dataset. We use GTSRB dataset of German traffic signs [40], another popular benchmark for deep learning problem. The dataset provides 39,209 training images in 43 classes and 12,630 test images in random order. Unlike MNIST, the images of traffic signs are RGB images and are stored in PPM format. The size of images varies from 15×15 to 250×250. To eliminate the influence of size, we reshape all images into 32×32 format. In addition, we standardize the images with zero mean and unit variance. Hence, each feature has a floating point value between 0 and 1. Among all training set, we randomly choose 39,000 training samples and divide them into 30 users, each of them having 1300 samples. Each user computes its local model based on these 1300 training images.

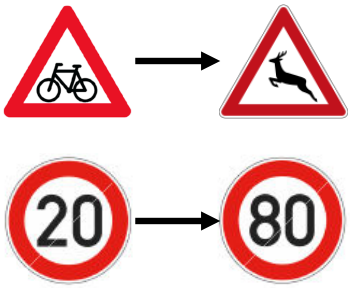

Figure 5: Poisoning the dataset to classify a sign of bicycle crossing as a sign of wild animal crossing (above) and classify a sign of 20 km/h maximum speed limit at 80 km/h (below)

Fraction of Malicious Users (%)	Attack Success Rate (%)		Accuracy Drop (%)	
	Bicycle to Wild Animal	20 km/h to 80 km/h	Bicycle to Wild Animal	20 km/h to 80 km/h
10	31	26	2	2
20	18	17	3	6
30	79	61	9	10

Table 2: Attack success rate and accuracy drop for mislabeling a sign of bicycle crossing as a sign of wild animal crossing and 20 km/h as 80 km/h in GTSRB dataset

Network Architecture. We use convolutional neural network (CNN) to train the classifier for traffic signs. For CNN architecture, there are two convolutional layers, two pooling layers, one fully-connected layer and a final output layer. The first convolutional layer consists of 64 filters with 2×2 and the second one with 16 filters with the same size as the first one. After each convolutional layer, there is one pooling layer followed with a filter of same size 2×2. Before the output layer, there is a fully-connected layer consisting of 64 nodes with ReLU function as their activation function, which is similar with the hidden layer of MLP architecture. For the output layer, the activation function is a Softmax function that converts the output of previous layer to a label between 0 to 42.

Attack Strategy. Since the number of classes for GTSRB is more than 40, we randomly perform two pairwise target poisoning attacks — classifying a cyclists crossing sign as a wild animals crossing sign and classifying the sign of 20 km/h maximum speed limit as 80 km/h (as shown in the Figure 5). The attack strategy is similar to that of MNIST dataset where the malicious users replicate the source image and mislabel them as the target image.

Attack Findings. Table 2 shows the attack success rate and accuracy drop for varying levels of malicious ratio when we mislabel a sign of bicycle crossing as a sign of wild animal crossing (case 1) and a sign of 20 km/h maximum speed limit as 80 km/h (case 2) respectively. The success rate for case 1 and case 2 is 79% and 61% separately for 30% malicious ratio and reduces with the decrease in the malicious ratio, although the attack success rate with 10% of malicious users is a little higher than when fraction of malicious users is 20%. This shows that the attacker can poison the dataset to achieve a targeted attack. However, the success rate differs depending on the source and target images that the attacker selects.

The accuracy drop is small in the final model due to poisoning of dataset as compared to benign model. The model trained under benign dataset exhibits an accuracy of 84%. As the total number of classes is more (43), the accuracy drop is smaller for the GTSRB dataset as compared to the MNIST dataset which as 10 classes. Thus, even though the attacker can achieve significant success for a targeted misclassification, the overall classification for other images is fairly accurate. Specifically, the accuracy drop for case 1 and case 2 is 9% and 10% when the malicious ratio is 30%.

Result 1: Targeted poisoning attacks are significantly effective in indirect collaborative deep learning setting regardless of masking the essential features of the training data and restricted poisoning capacity of the adversary.

5. DESIGN

Our AUROR defense against poisoning attacks is deployable at the training phase before the final global model is generated. AUROR filters the malicious users before creating the final model, using the following steps.

Identifying Indicative Features. In designing AUROR, the first step is to identify the indicative features that show an anomalous distribution under attack setting. It collects all the masked features from users and compares each of these features. In this step, we use KMeans algorithm to divide all users into two clusters for any given masked features for first 10 epochs since features for first 10 epochs have enough difference between malicious users and benign users and decrease the computation. However, any other clustering algorithm can be used instead. We then calculate the distance between the centers of these clusters. If the distance exceeds a certain limit α, we consider the masked features as indicative features. The threshold α determines whether the two clusters are distinct from each other. We choose $\alpha = 0.02$ for MNIST dataset and for GTSRB $\alpha = 0.0045$.

Identifying Malicious Users. The second step in AUROR is to identify the malicious users based on the indicative features. For each indicative feature, the users are divided into different clusters. Every cluster with number of users which is smaller than $n/2$ are marked as suspicious clusters, since the attackers are in the minority. The users that appear in suspicious clusters for more than $\tau = 50\%$ of the total indicative features are confirmed as malicious users. The threshold τ ensures that the benign users that show a similar distribution as the malicious users within statistical error are not labeled as malicious.

Training Global Model. The server excludes the input values from the malicious users identified in the previous step and trains the global model on the remaining masked features. The training procedure varies based on the underlying learning algorithm used in the application. For the cases we illustrate in our paper, we use privacy preserving deep learning architecture (PPDL) to retrain the model.

Implementation. We implement the prototype of AUROR in Python. The implementation contains a total of 5401 lines calculated using CLOC tool available on Linux. We use Theano package [12] to realize the back-propagation procedure in deep learning. To implement the collaborative setting for these applications, we use multiprocessing module in Python to run users' programs asynchronously. We create a server process to aggregate masked features submitted by each user and compute the final global model.

6. AUROR EVALUATION

To evaluate the efficacy of AUROR, we apply our defense to the targeted poisoning attacks performed on image recognition sys-

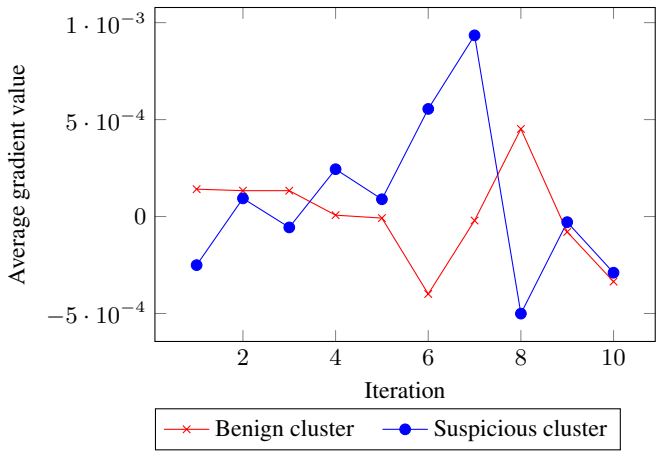

Figure 6: For MNIST dataset, the gradient value for 8^{th} weight parameter of third node in first hidden layer for mislabeling 7 as 8. All the users cannot be distinguished over 10 iterations.

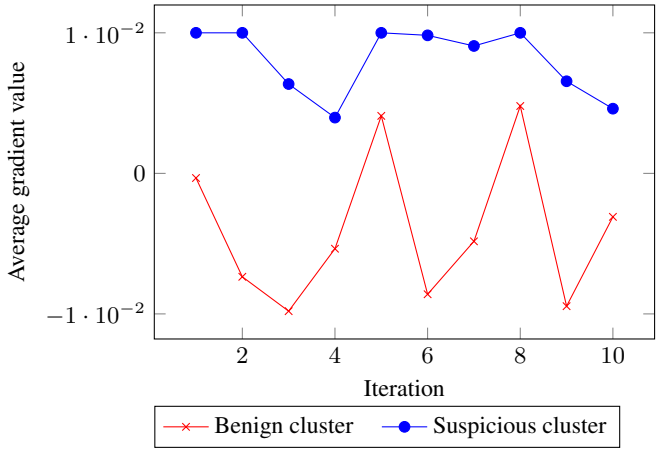

Figure 7: For MNIST dataset, the gradient value for the bias parameter of 8^{th} node in final layer for mislabeling 7 as 8. The gradient shows different distribution for benign and malicious users.

tems. The poison set of malicious user includes mislabeled source images. Both the benign and malicious users submit their masked gradient values of the training data as features to the server for each iteration. Each gradient value corresponds to a particular parameter of the classifier. The server in our setting executes AUROR where the input is the masked features from the users and output is the final global model trained using AUROR. We perform our experiments on a server running on Ubuntu Trusty (Ubuntu 14.04.3 LTS), equipped with 40 CPUs E5-2660 v3 each having a processor speed of 2.6GHz and 64 GB RAM.

Evaluation Goals. We perform the evaluation with the following three goals:

- To measure the detection rate of AUROR for identifying the malicious users.
- To evaluate the attack success rate of global model trained using AUROR.
- To evaluate the accuracy drop of the global model as compared to the benign model.

6.1 Handwritten Digit Images

Identifying Indicative Features. AUROR analyses the distribution of gradient values uploaded by the users for several iterations of the training phase. Figure 6 shows the comparison of the average gradient values for 8^{th} weight parameter of third node in first hidden layer for the first 10 iterations between benign cluster and suspicious cluster. The average gradient value of benign users shows a similar distribution as the malicious users during the iterations. Hence, AUROR discards this feature and does not use it in the future steps. On the contrary, Figure 7 shows the comparison of the average gradient values for the bias parameter of 8^{th} node in final layer. The gradient values exhibit two different kinds of distribution. Based on the anomalous behavior AUROR selects this as a indicative feature. The number of masked features selected as indicative features varies for different experiments. For example, AUROR selects 64 indicative features for mislabeling 7 as 8 while 67 indicative features when mislabeling 1 as 4 with 30% malicious users. In addition, we notice that all the indicative features come from the final layer, because the parameters in final layer have the largest influence towards the final result. Hence, the changes to final layer's parameter are larger than the other parameters.

Malicious Ratio (%)	Accuracy Drop (%)
10	0
20	1
30	3

Table 3: Accuracy drop for MNIST dataset after retraining the model using AUROR for 10% to 30% malicious ratio

Detecting Malicious Users. AUROR uses KMeans clustering algorithm to separate all the users into different groups based on their uploaded indicative features. The groups with fraction of users within 50% are marked as suspicious. For every indicative feature, it creates clusters of benign and suspicious users. The users that appear in the suspicious clusters for more than $\tau = 50\%$ of the indicative features are marked as malicious. We observe that the 12^{th} and 14^{th} benign users appear in suspicious clusters three times of 64 indicative features when we mislabel 7 as 8 with 30% of malicious users. The value of τ shows the tolerance of AUROR towards the minor difference between benign users. We calculate the detection rate based on the users marked as malicious by AUROR and the actual number of malicious users in every experiment. We observe that the detection rate is 100% for 10% to 30% of malicious users.

Evaluating the Final Model. For generating the final global model, AUROR removes the users detected as malicious in the previous step and trains the model. We measure the attack success rate and accuracy drop of this final global model trained using AUROR defense. We observe that the attack success rate reduces largely after training the model without the malicious user, which are all below 5% when malicious ratio is 10%, 20% and 30%.

We measure the accuracy drop of the final global model as compared to the benign model and study the improvement over the poisoned model. Figure 3 shows the accuracy drop for malicious ratios ranging from 10% to 30%. We observe that the accuracy drop is very small as compared to the benign model. It is only 3% when the fraction of malicious users is 30%. This highlights an important finding that the overall accuracy of the image recognition system remains similar to the accuracy of benign model even after excluding the training data from malicious users.

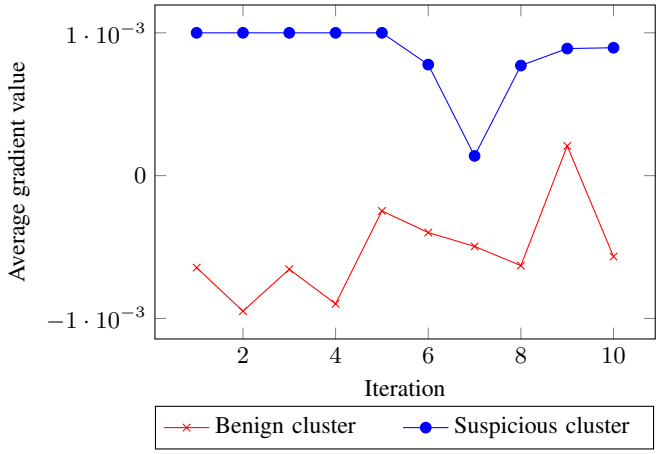

Figure 8: In GTSRB dataset, the gradient value for the 547^{th} weight of final layer for the first 10 iterations. The gradient show different distribution for benign and malicious users.

6.2 German Traffic Sign Benchmarks

The German Traffic Sign Benchmarks (GTSRB) dataset is a collection of images used to generate models for using in auto-driving cars. The trained model classifies a sign into one of the 43 classes.

Identifying Indicative Features. AUROR analyses the distribution of gradient values uploaded by the users for several iterations of the training phase. Figure 8 shows the comparison of the gradient values for 547^{th} weight of final layer for the first 10 iterations. The gradient values exhibit two different kinds of distribution. Based on the anomalous behavior AUROR selects this as a indicative feature. The number of masked features selected as indicative features vary for different experiments. For example, AUROR selects 36 indicative features when we mislabel the sign of bicycle crossing as the sign of wild animal crossing with 30% malicious users, while selects 55 indicative features when we mislabel the traffic sign of 20 km/h maximum speed limit as 80 km/h. In addition, we also observe the distribution of all indicative features. The indicative features are all from the final layer of the model, which confirms the finding that the parameters of final layer are easy to change since they have the largest influence over final result compared with other parameters.

Detecting Malicious Users. For each indicative feature, AUROR creates clusters of benign and suspicious users. The users that occur in the suspicious clusters with the frequency less than $\tau = 50\%$ are marked as malicious. Like MNIST, we have the same observation that the second and third benign users both appear in the suspicious clusters twice of 9 indicative features when we mislabel the sign of 20 km/h maximum speed limit as 80 km/h with 10% of malicious users. We calculate the detection rate based on the users marked as malicious using AUROR and the actual number of malicious users in every experiment. We observe that the detection rate is 100% for 10% to 30% of malicious users.

Evaluating the Final Model. To measure the effectiveness of our solution, we calculate the accuracy drop after retraining the model using AUROR on the GTSRB dataset. Table 4 shows the accuracy drop as compared to the benign setting when mislabeling a sign of bicycle crossing as a sign of wild animal crossing. The accuracy drop is negligible for the malicious ratios from 10% to 30%, indicating that the overall accuracy of the model is not affected drastically by removing the dataset contributed by malicious users. We measure the attack success rate of the retrained model and report that it is below 5% for malicious ratio from 10% to 30%.

Fraction of Malicious Users (%)	Success Rate (%)	Accuracy Drop (%)
10	1	0
20	2	0
30	2	0

Table 4: Attack success rate and accuracy drop for GTSRB dataset after retraining the model using AUROR for 10% to 30% malicious ratio

6.3 Evading AUROR

There are two main approaches to evade AUROR's detection mechanism. The first strategy is to decrease the fraction of malicious users so that the influence of the poisoned data on the global model is reduced. Our experiments demonstrate for MNIST dataset, when the poison set of attackers is 100% malicious data, the detection rate of our method AUROR is 100% even when there is only one malicious user among 30 participants. To mislabel the sign of bicycle crossing as the sign of wild animal crossing in GTSRB dataset, the detection rate is 60% when number of malicious users is reduced to one. Although the detection rate is 60%, we find that the attack success rate is only 3% with 1% accuracy drop. Thus, our experiments confirm that decreasing the number of malicious users can evade detection mechanism of AUROR in some cases but the final result generated by these poisoning attack cannot achieve the attackers' goal of misclassifying data.

The second strategy is to decrease the number of malicious samples in malicious users' training set. Table 5 shows the average detection rate, average attack success rate and average accuracy drop with various combination of fraction of malicious data and fraction of malicious users for MNIST dataset when misclassifying 5 as 3. When the fraction of malicious users is 20%, the average detection rate of AUROR on MNIST is 100% even when only 20% of the training set of each adversary is poisoned, while the average detection rate is 0% when the fraction of malicious data reduces to 14%. Although the average detection rate drops to 0%, the average attack success rate is only 34% with 3% of average accuracy drop. Hence, the adversaries can decrease the number of malicious samples in their training set to decrease the distance of gradients between malicious users and benign users (α). However, while evading AUROR, the average attack success rate and average accuracy drop become relatively low such that the goal of adversaries cannot be achieved. Table 6 shows similar relation of average detection rate, average attack success rate and average accuracy drop for different fraction of malicious data and fraction of malicious users while mislabeling the sign of bicycle crossing as the sign of wild animal crossing in the GTSRB dataset. In addition, we observe that the attack success rate varies between a wide range. For example, the minimum and maximum attack success rate are 1.6% and 57.8% respectively for 90% of malicious data and 20% of malicious user when mislabeling the sign of bicycle crossing as the sign of wild animal crossing in the GTSRB dataset. Even with such a variation, the detection rate is 100%. Thus, AUROR is robust and a promising solution against evasion.

Result 2: A robust and strong defense against targeted poisoning attacks is possible based on the masked features and by exploiting the limited poisoning characteristics of indirect collaborative deep learning systems.

Fraction of Malicious Users (%)	Metrics (%)	Fraction of Malicious Data (%)		
		14	**18**	**20**
10	**DR**	33	100	100
	SR	21	20	21
	AD	1	0	2
20	**DR**	0	100	100
	SR	34	38	45
	AD	3	2	3
30	**DR**	74	100	100
	SR	54	69	68
	AD	4	5	5

Table 5: Average detection rate (DR), average attack success rate (SR) and average accuracy drop (AD) for MNIST when the fraction of malicious data is decreased.

Fraction of Malicious Users (%)	Metrics (%)	Fraction of Malicious Data (%)		
		50	**70**	**90**
20	**DR**	33	70	100
	SR	31	13	22
	AD	2	2	4
30	**DR**	100	96	100
	SR	16	60	32
	AD	2	2	5

Table 6: Average Detection rate (DR), average attack success rate (SR) and average accuracy drop (AD) for GTSRB when the fraction of malicious data is decreased.

7. RELATED WORK

Collaborative Learning. Various researches have shown the benefits of collaborative learning in the area of machine learning algorithms [28, 45]. With the advent of deep learning, researches have proposed collaborative setting for deep learning algorithms. Wang et. al use hierarchical Bayesian model and show that collaborative deep learning significantly improves the state-of-the-art recommendation systems [43]. Xu et. al separate the learning tasks to each user based on the data locality property, which is compatible with various learning algorithms and ensure that only final result is revealed to other user [44]. To make this setting privacy preserving, Pathak et. al aggregate a classifier using classifiers trained locally by separate participants [35]. Their model leverages the technique of differential privacy to hide the information about the training data. However, simple aggregation generates classifiers with very high accuracy. Recently, Shokri et. al use differential privacy to design a deep learning model that supports collaboration among users while preserving the privacy of their training data [39]. Our work is motivated by such indirect collaborative deep learning models.

Adversarial Learning. Adversarial learning is a problem that has been studied by researchers for a long time [14, 20, 22, 27]. Researchers divide this problem into two categories by the influence: one is the causative attacks or poisoning attacks, the other is the exploratory attacks. For the poisoning attacks, the adversaries alter the training procedures by polluting the training data, while for exploratory attacks they modify the samples in order to bypass the trained classifier. Deep learning systems are also open to these attacks. Various researchers have focused on performing exploratory attacks on deep learning models and proposed algorithms and defenses for the same. Papernot et. al provide algorithms to craft adversarial samples such that the final model misclassifies specific target samples [34]. They exploit the mapping between the input feature and output class so that attackers can efficiently evade the classifiers by perturbing the related features. Goodfellow et. al utilize the gradients to update the input value in order to get the adversarial samples [21]. Moreover, in a recent work, Papernot et al. introduce another attack algorithm that creates an alternate model of the input-output pairs and craft adversarial samples based on the auxiliary model [33]. All the previous work focus on exploratory attacks that are perpetrated after the training phase is complete. However, the emerging collaborative learning technique opens these deep learning algorithms to poisoning attacks. Although, poisoning attacks are well understood with respect to machine learning algorithm for example in recommendation systems [17, 36, 42], there impact on deep learning systems is unexplored. According to our knowledge, this is the first work to understand the efficacy of poisoning attacks in deep learning systems and provide a concrete defense against it.

Defense. Defenses against poisoning attacks are known in the area of collaborative recommendation systems using machine learning techniques [15, 19, 36]. Biggio et. al [15] examine the effectiveness of bagging ensembles towards the poisoning attack in spam filter and the intrusion detection system. Previous work has considered data sanitization to measure the negative impact of training dataset towards a trusted model [29, 32]. Muhlenbach et al. filter samples that do not have same class as their neighbors, which needs access to raw training dataset [30]. All the previous work is targeted towards defending poisoning attacks for machine learning algorithms like k-nearest neighbors (KNN), support vector machines (SVM) and others. Thwarting poisoning attacks in deep learning systems, where the final model takes into consideration the parameters from every node in the neural network of the input dataset is not known. In this work, we show that detecting poisoned dataset based on anomalous distribution of the parameters is an effective and promising solution.

8. CONCLUSION

In this paper, we demonstrate the impact of targeted poisoning attacks on deep learning systems for two datasets (MNIST and GTSRB) in indirect collaborative learning setting. Targeted poisoning attacks are effective even if the attackers can poison a limited fraction of training data and the final model is trained using masked features from the training data. To thwart against these attack, we propose AUROR— a statistical defense that exploits the fact that malicious users can only poison their dataset without the knowledge about the data of other users. Our evaluation confirms that AUROR is a promising defense against poisoning attacks in indirect collaborative learning.

9. ACKNOWLEDGEMENTS

We thank the anonymous reviewers of this paper for their helpful feedback. We also thank Shweta Shinde, Viswesh Narayanan and Yaoqi Jia for useful discussion and feedback on an early version of the paper. This work is supported by the Ministry of Education, Singapore under Grant No. R-252-000-560-112 and a university research grant from Intel. All opinions expressed in this work are solely those of the authors.

10. REFERENCES

[1] Batman V Superman caught purchasing fake ratings on IMDB. http://www.bleachbypass.com/batman-v-superman-fake-imdb-ratings/.

[2] Facebook's Moments app uses artifical intelligence. http://money.cnn.com/2015/06/15/technology/facebook-moments-ai/.

[3] Facebook's Virtual Assistant 'M' Is Super Smart. It's Also Probably a Human. http://recode.net/2015/11/03/facebooks-virtual-assistant-m-is-super/-smart-its-also-probably-a-human/.

[4] How 'Deep Learning' Works at Apple, Beyond. https://www.theinformation.com/How-Deep-Learning-Works-at-Apple-Beyond.

[5] Improving Photo Search: A Step Across the Semantic Gap. http://googleresearch.blogspot.sg/2013/06/improving-photo-search-step-across.html.

[6] Making Cortana smarter: how machine learning is becoming more dynamic. http://www.techradar.com/sg/news/.

[7] Meet The Guy Who Helped Google Beat Apple's Siri. http://www.forbes.com/sites/roberthof/2013/05/01/meet-the-guy-who-helped-google-beat-apples-siri/#7c3a2bda56cb.

[8] Personalized Recommendations Frequently Asked Questions. http://www.imdb.com/help/show_leaf?personalrecommendations.

[9] Spam filter. https://gmail.googleblog.com/2015/07/the-mail-you-want-not-spam-you-dont.html.

[10] 'The Interview' Now Has a Perfect 10 Rating on IMDb. http://motherboard.vice.com/read/the-interview-has-a-perfect-10-on-imdb.

[11] The mail you want, not the spam you donâĂŹt. https://gmail.googleblog.com/2015/07/the-mail-you-want-not-spam-you-dont.html.

[12] Theano Package. https://github.com/Theano/Theano.

[13] A. Anjos and S. Marcel. Counter-measures to photo attacks in face recognition: a public database and a baseline. In *International joint conference on Biometrics (IJCB)*, pages 1–7. IEEE, 2011.

[14] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar. Can machine learning be secure? In *Symposium on Information, computer and communications security*, pages 16–25. ACM, 2006.

[15] B. Biggio, I. Corona, G. Fumera, G. Giacinto, and F. Roli. Bagging classifiers for fighting poisoning attacks in adversarial classification tasks. In *Multiple Classifier Systems*, pages 350–359. Springer, 2011.

[16] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

[17] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[18] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics*, pages 177–186. Springer, 2010.

[19] Y. Cao and J. Yang. Towards making systems forget with machine unlearning. In *Symposium on Security and Privacy*, pages 463–480. IEEE, 2015.

[20] N. Dalvi, P. Domingos, S. Sanghai, D. Verma, et al. Adversarial classification. In *Proceedings of the 10th SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108. ACM, 2004.

[21] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the 3th International Conference on Learning Representations*, 2015.

[22] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and Artificial Intelligence*, pages 43–58. ACM, 2011.

[23] W. Jung, S. Kim, and S. Choi. Poster: Deep learning for zero-day flash malware detection. In *36th IEEE Symposium on Security and Privacy*, 2015.

[24] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324. IEEE, 1998.

[25] Y. Li, R. Ma, and R. Jiao. A hybrid malicious code detection method based on deep learning. In *International Journal of Security and Its Applications*, volume 9, pages 205–216, 2015.

[26] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. In *Internet Computing, IEEE*, volume 7, pages 76–80. IEEE, 2003.

[27] D. Lowd and C. Meek. Adversarial learning. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 641–647. ACM, 2005.

[28] L. Melis, G. Danezis, and E. De Cristofaro. Efficient private statistics with succinct sketches. In *Proceedings of the 23rd Network and Distributed System Security Symposium*, 2015.

[29] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha. Systematic poisoning attacks on and defenses for machine learning in healthcare. In *IEEE Journal of Biomedical and Health Informatics*, volume 19, pages 1893–1905. IEEE, 2015.

[30] F. Muhlenbach, S. Lallich, and D. A. Zighed. Identifying and handling mislabelled instances. In *Journal of Intelligent Information Systems*, volume 22, pages 89–109. Springer, 2004.

[31] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *29th IEEE Symposium on Security and Privacy*, pages 111–125. IEEE, 2008.

[32] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. Sutton, J. Tygar, and K. Xia. Misleading learners: Co-opting your spam filter. In *Machine learning in cyber trust*, pages 17–51. Springer, 2009.

[33] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. Berkay Celik, and A. Swami. Practical black-box attacks against deep learning systems using adversarial examples. In *arXiv preprint arXiv:1602.02697*, 2016.

[34] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy*, pages 372–387. IEEE, 2016.

[35] M. Pathak, S. Rane, and B. Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Information Processing Systems*, pages 1876–1884, 2010.

[36] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. Tygar. Antidote: understanding

and defending against poisoning of anomaly detectors. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 1–14. ACM, 2009.

[37] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In *Neurocomputing: foundations of research*, pages 673–695. MIT Press, 1988.

[38] J. Schmidhuber. Deep learning in neural networks: An overview. In *Neural Networks*, volume 61, pages 85–117. Elsevier, 2015.

[39] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1310–1321. ACM, 2015.

[40] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. In *Neural Networks*, volume 32, pages 323–332. Elsevier, 2012.

[41] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations*, 2014.

[42] G. Wang, T. Wang, H. Zheng, and B. Y. Zhao. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *Proceedings of the 23rd USENIX conference on Security symposium*, pages 239–254, 2014.

[43] H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM, 2015.

[44] K. Xu, H. Ding, L. Guo, and Y. Fang. A secure collaborative machine learning framework based on data locality. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–5. IEEE, 2015.

[45] K. Xu, H. Yue, L. Guo, Y. Guo, and Y. Fang. Privacy-preserving machine learning algorithms for big data systems. In *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*, pages 318–327. IEEE, 2015.

[46] W. Xu, Y. Qi, and D. Evans. Automatically evading classifiers. In *Proceedings of the 2016 Network and Distributed Systems Symposium*, 2016.