

Poisoning Behavioral-based Worker Selection in Mobile Crowdsensing using Generative Adversarial Networks

Ruba Nasser^{a,b}, Ahmed Alagha^c, Shakti Singh^{a,b}, Rabeb Mizouni^{a,b,*}, Hadi Otrok^{a,b}, Jamal Bentahar^{a,c}

^a*Computer Science Department, Khalifa University, Abu Dhabi*

^b*Center of Cyber-Physical Systems (C2PS), Khalifa University, Abu Dhabi*

^c*CIISE, Concordia University, Montreal, Canada*

Abstract

With the widespread adoption of Artificial intelligence (AI), AI-based tools and components are becoming omnipresent in today's solutions. However, these components and tools are posing a significant threat when it comes to adversarial attacks. Mobile Crowdsensing (MCS) is a sensing paradigm that leverages the collective participation of workers and their smart devices to collect data. One of the key challenges faced at the selection stage is ensuring task completion due to workers' varying behavior. AI has been utilized to tackle this challenge by building unique models for each worker to predict their behavior. However, the integration of AI into the system introduces vulnerabilities that can be exploited by malicious insiders to reduce the revenue obtained by victim workers. This work proposes an adversarial attack targeting behavioral-based selection models in MCS. The proposed attack leverages Generative Adversarial Networks (GANs) to generate poisoning points that can mislead the models during the training stage without being detected. This way, the potential damage introduced by GANs on worker selection in MCS can be anticipated. Simulation results using a real-life dataset show the effectiveness of the proposed attack in compromising the victim workers' model and evading detection by an outlier detector, compared to a benchmark. In addition, the impact of the attack on

*I am the corresponding author

Email address: rabeb.mizouni@ku.ac.ae (Rabeb Mizouni)

reducing the payment obtained by victim workers is evaluated.

Keywords: Mobile Crowdsensing, Generative Adversarial Networks,
Adversarial Machine learning

1. Introduction

Mobile Crowdsensing (MCS) is a sensing approach that utilizes the collective involvement of mobile workers and their smart devices to collect data. The management platform first receives sensing tasks from task requesters and subsequently performs worker selection so that the quality of service (QoS) is maximized. The workers collect the requested data and send it back to the platform, which then pays the workers back for their contributed data. Artificial Intelligence (AI) based solutions have been widely adopted to optimize the performance of MCS. A notable real-world example is Uber, which leverages deep learning methods to optimize the Quality of Service (QoS) by predicting workers' estimated arrival times [1]. Waze is another well-known MCS application that utilizes data contributed by workers to enhance the driving experience by using AI methods to predict traffic and crash locations and timings [2], [3].

One of the key challenges faced in MCS systems is selecting workers who are more likely to complete the tasks assigned. While they may initially accept these tasks, their behavior can vary significantly, leading to potential cancellations influenced by various contextual factors, such as the day of the task and the weather conditions. AI techniques have shown great potential in addressing this challenge. Using historical task data, supervised learning methods can be adopted to build behavioral models for each worker. These models can then be leveraged to predict the willingness of the workers to perform the task [4]. Despite the effectiveness of AI in optimizing MCS worker selection, their adoption introduces vulnerabilities that can be exploited by insider adversaries during the training phase [5], [6]. Insider adversaries are individuals who have access to the management platform and exploit their trusted positions to alter the training process of the AI models by injecting malicious data [7]. This causes

the models to behave incorrectly when deployed, ultimately undermining the system’s fairness.

This work proposes an adversarial attack aimed at compromising behavior-based worker selection in MCS systems. The proposed attack bridges the theoretical advancements in adversarial machine learning by providing a novel application in practical settings like MCS worker selection and further highlights its real-world impact on the workers. In addition, one of the primary motivations for developing this attack model is to explore the risks presented by insider adversaries who have trusted access to the MCS platform and leverage it to manipulate the training process of the AI models. The proposed attack also highlights critical concerns about the reliability of AI-based selection systems due to their inherent vulnerabilities to malicious exploitation. These concerns are not merely theoretical; real-world cases have demonstrated the consequences of untrustworthy AI decisions. For example, Amazon had to discontinue its AI-based selection algorithm after discovering that it favored male over female candidates [8].

To this end, we propose a novel attack on behavioral-based MCS where malicious insiders leverage Generative Adversarial Networks (GANs) to generate poisoning points that can degrade the performance of victim workers’ models by identifying vulnerable regions in the feature space of their data. GANs are powerful tools for data generation due to their unique adversarial training mechanism. A typical GAN architecture comprises two neural networks: the generator and the discriminator, trained simultaneously with opposing objectives in a minimax framework. The generator aims to produce synthetic data that closely resembles real data, while the discriminator learns to differentiate between genuine and generated samples [9]. Numerous GAN variants have been developed by incorporating additional components, and by modifying the loss functions to guide the generation process toward specific features or desired characteristics. In this study, we extrapolate on the method proposed in [10] and build upon its framework to tailor the Poisoning GAN (PGAN) approach specifically for insider attacks on behavioral-based selection in MCS. The PGAN

utilized in this study consists of a generator that tries to minimize the losses of both a discriminator and the victim worker’s behavioral model during training [10]. The insider then utilizes the trained PGAN to generate poisoning points and replaces a portion of the victim workers’ data with the generated data. Subsequently, the platform uses the poisoned models to perform the selection, resulting in fewer task assignments to victim workers and a significant reduction in their overall revenue. While the attack can degrade the performance of the targeted models, it also considers the detectability constraints by generating poisoning points that can bypass outlier detectors. In addition, it does not compromise the QoS achieved for the tasks. Using a GAN-based approach makes the attack effective in terms of stealthiness, as it regulates the generation of attack samples in a way that makes them unnoticeable. This allows the generated poisoning points to bypass anomaly detection models that can be adopted in the MCS system before the training stage. Overall, the main contributions of this work are summarized below:

- A novel adversarial attack on behavioral-based MCS worker selection is proposed. In this attack, insider adversaries use PGANs to generate poisoning data to compromise victim workers’ behavioral models during the training phase.
- Propose a targeted attack on MCS that specifically aims to lower the overall revenue of victim workers by decreasing the number of tasks they get assigned by the platform.
- Propose a novel framework that demonstrates how PGANs can be deployed at the worker selection stage to poison victim workers’ models in MCS systems. The framework can be adopted to assess the efficacy of adversarial attacks against a resilient selection system that incorporates outlier detection and a QoS-based selection mechanism.

This work demonstrates how GAN-based poisoning attacks can be effectively adapted and leveraged against behavioral-based worker selection models in MCS

systems, a scenario that, to the best of our knowledge, has not been previously explored. Furthermore, by integrating and extrapolating PGAN in the MCS pipeline, this work reveals unique vulnerabilities in real-world worker selection systems, which have received limited attention in adversarial machine learning literature. Few studies investigated adversarial attacks on AI-based MCS, focusing mainly on the models adopted to improve the system’s performance at the data aggregation stage and enhance its security [11],[12]. To the best of our knowledge, this is the first work that proposes an adversarial attack specifically targeting behavioral-based selection models in MCS. Simulation results using a real-life dataset show the effectiveness of the proposed attack in compromising the victim workers’ models and evading detection by an outlier detector, compared to a benchmark.

2. Background and Related Work

This section introduces GANs and discusses the related literature encompassing works on behavioral-based worker selection in MCS and adversarial attacks on AI-based MCS and IoT systems.

2.1. Background: Generative Adversarial Networks

GANs are a class of deep learning frameworks designed to generate new data samples that resemble a given training dataset. The typical GAN model consists of two neural networks: a generator G and a discriminator D , trained in a competitive setting. The generator produces synthetic data samples, while the discriminator tries to distinguish between genuine and synthetic data. The training process continues until the generator produces samples that are indistinguishable from real data, effectively capturing the underlying data distribution.

The interaction between G and D can be modeled as a 2-player min-max game as shown in (1), where $V_{GAN}(D, G)$ is the objective function for D that also depends on G . As defined in (2), $V_{GAN}(D, G)$ is the sum of the expected log-likelihood that the discriminator correctly identifies real and generated data,

where x is sampled from the real data distribution $p_x(x)$ and z is sampled from a prior normal distribution $p_z(z)$ [13].

$$\min_G \max_D V_{GAN}(D, G) \quad (1)$$

$$V_{GAN}(D, G) = \mathbb{E}_{x \sim p_x(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (2)$$

In GANs, the generator uses random noise as input, which can limit the model's performance and the quality of the generated data. To overcome this limitation, researchers have introduced auxiliary information to the input noise, enabling the generator to produce higher-quality data. Conditional GAN (CGAN) is one of the widely used types of GANs, and it uses class labels as input to both the generator and discriminator. Similar to GANs, the min-max game can be formulated as shown in (3), where the objective function for CGAN $V_{CGAN}(D, G)$ is defined in (4). The main difference between GAN and CGAN, is that in CGAN, x is sampled from the real data distribution conditioned on class c $p_x(x|c)$ and z is sampled from a prior normal distribution conditioned on the same class $p_z(z|c)$ [14].

$$\min_G \max_D V_{CGAN}(D, G) \quad (3)$$

$$V_{CGAN}(D, G) = \mathbb{E}_{x \sim p_x(x|c)} [\log D(x|c)] + \mathbb{E}_{z \sim p_z(z|c)} [\log(1 - D(G(z|c)))] \quad (4)$$

2.2. GAN-based adversarial attacks

The unique capabilities of GANs can be leveraged to manipulate and deceive machine learning models at various stages, including the pre-deployment or the post-deployment stage of the targeted models. In the pre-deployment stage, GANs are used to poison the training datasets by generating fake samples and injecting them into the datasets with flipped labels. Such techniques have been widely adopted in the computer vision domains [15] and in federated learning

systems [16], [17], [18]. On the other hand, in the post-deployment stage, GANs can be utilized in multiple ways. The first is adversarial example generation, where the generator creates adversarial samples specifically designed to fool the trained classifier [19], [20], [21], [22]. The other approach includes perturbation addition, where the generator creates subtle noise that, when added to the real data sample, results in increased classification error [23], [24]. These methods are predominantly explored within the computer vision domain, where minimal alterations to test images often go unnoticed by human observers, yet can drastically impact model performance.

2.3. Behavioral-based Worker Selection in MCS

Several studies in the literature proposed AI techniques to perform behavioral-based worker selection in MCS systems. In [25], unique models were trained for each worker to predict their willingness to perform the tasks. The models were trained using task-related data, such as the task start time, and worker-related data, such as the number of tasks completed per day. In [26], an auctioning technique that adopts AI to predict workers' ability to complete sensing tasks was proposed. A Long Short-Term Memory (LSTM) model was leveraged to predict the battery levels and internet connectivity status of workers' devices throughout the sensing period. In [27], a biometrics-based selection framework was proposed, where a unique model for each MCS worker is built based on their unique interaction patterns with the smartphone's touching screen. By leveraging machine learning techniques, these behavioral traits were used in order to detect impersonators in the system.

Moreover, in [28], [29], and [30], historical mobility traces were used to predict workers' future location to improve worker selection in MCS. In [28], a deep learning-based approach was adopted to predict the future location values; then a greedy algorithm was used to perform the worker selection, such that the sensing coverage is maximized. In [29], deep learning was also used to perform the location prediction, and a weighted utility-based worker selection algorithm was proposed to perform the worker selection. Finally, in [30], a machine learning-

based approach was proposed to predict workers' future locations, which were then subsequently utilized in a continuous worker selection process based on a genetic algorithm.

Overall, the main advantages of the works discussed in this section include the use of AI-based methods that leverage historical worker data to predict their future behavior, thus optimizing the performance of MCS worker selection. However, these methods are vulnerable to insider threats who can manipulate training data. This introduces risks that could significantly diminish the performance and reliability of the worker selection process.

2.4. Adversarial Attacks on AI-based MCS and IoT Systems

Adversarial machine learning is the study of how machine learning models can be manipulated by carefully crafted input. These inputs are intentionally designed by malicious adversaries to exploit the vulnerabilities of learning algorithms while causing models to make incorrect predictions. Adversarial attacks can be classified based on the phase of the machine learning pipeline in which they occur. In poisoning attacks, the adversary manipulates the training data to corrupt the learning process and degrade model performance. In contrast, evasion attacks involve crafting inputs that deceive a trained model during deployment without altering the training data [31].

Several studies explored insider attacks on AI models adopted in Internet of Things (IoT) systems, either before or after the model deployment. For instance, the attack proposed by [32] targeted ML-based intrusion detection systems in smart home networks. The authors proposed an approach to generate adversarial attack samples targeting the model after deployment, with the aim of misclassifying malicious network packets as normal. The techniques used to generate the adversarial samples include the Fast Gradient Sign Method (FGSM) and Jacobian Saliency Map Attack methods (JSMA). Another study, [33], proposed an adversarial attack on a machine learning-based malware detection system. The attack specifically targeted the model after deployment with the goal of successfully delivering the malware to smart devices. Tech-

niques such as FGSM, JSMA, Carlini and Wagner (C&W), and DeepFool were leveraged.

Moreover, in [34], an insider attack on AI models deployed for medical diagnostic applications was proposed. The attack was conducted during the testing phase, and methods such as FGSM, DeepFool, C&W, and JSMA were employed. The main goal of the attack is to increase the classification error of identifying Covid-19 cases. Finally, the attack proposed in [35] targeted models during the training phase, focusing on air quality monitoring applications. The utilized attack method is label-flipping, where the main goal is to degrade the performance of the ML model deployed to predict the impact of certain chemicals on the overall air quality.

A limited number of studies have proposed adversarial attacks on AI-based MCS systems. These works either aim to bypass security defense measures or degrade the quality of sensing achieved. In [12], an unsupervised learning approach to generate poisoning data that degrades the performance of human activity recognition classifiers was proposed. The attack targets the models during the training phase and uses Self-Organizing Maps (SOM) to generate the poisoning data [12]. Furthermore, in [11], GANs were used to generate fake tasks that can successfully bypass machine learning-based fake task detection models. These models were trained to classify tasks into real and fake based on certain features such as task duration, battery requirement, and start time. Since the features that characterize the fake tasks follow a certain distribution, the machine learning models can successfully identify them. However, the authors argue that adversaries could utilize CGANs to generate another type of fake tasks that are similar to real tasks, making them undetectable by the machine learning models and potentially overwhelming workers' devices.

The works discussed in this section emphasize the significant risks posed by insider threats within IoT and MCS environments. They proposed various attacks on AI models in various domains, including smart home networks, malware detection, medical diagnostics, environmental monitoring, and security. However, none of these studies specifically address the vulnerabilities related to

insider adversaries targeting MCS worker selection and the subsequent implications for the overall revenue generated by workers from MCS tasks.

Table 1 compares the works proposing adversarial attacks on machine learning models in MCS and IoT systems. As illustrated in the table, none of the existing works focused on developing an attack targeting MCS behavioral prediction models. This paper proposes an insider attack targeting victim workers to lower their revenue. The proposed attack uses PGANs, where the generator tries to increase the error of both the discriminator and the victim workers’ behavioral models. Consequently, the trained generator can be used to craft poisoning data that are then injected by the insider into the victims’ training datasets without being detected.

Table 1: Summary of adversarial attacks on machine learning models in MCS and IoT systems

Reference	Application	Phase	Method
[11]	MCS fake task detection	Post-deployment	CGAN
[12]	Activity recognition	Pre-deployment	SOM
[32]	Intrusion detection	Post-deployment	FGSM and JSMA
[33]	Malware detection	Post-deployment	FGSM, JSMA, C&W, DeepFool
[34]	Medical diagnosis	Post-deployment	FGSM, JSMA, C&W, DeepFool
[35]	Air quality monitoring	Pre-deployment	Label flipping
Proposed work	Behavioral-based worker selection	Pre-deployment	PGAN

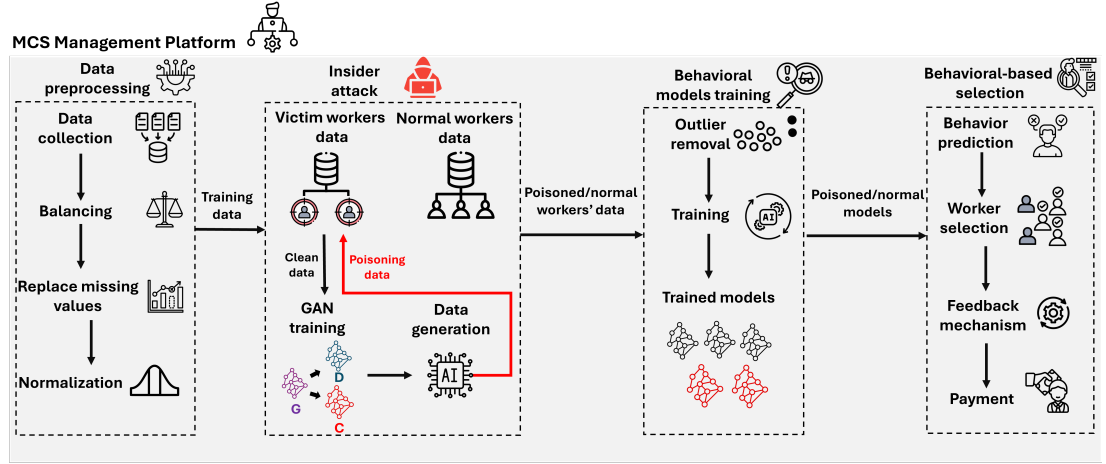


Figure 1: Overview of the proposed attack in the MCS platform, including 4 different modules: Data preprocessing, insider attack, behavioral model training, and behavioral-based selection module.

3. MCS Insider Attack Overview

One of the main reasons AI is adopted at the selection stage is to understand the behavior of the workers, which is done by collecting workers' data from previous tasks and training models to predict their willingness to perform future tasks. However, in this paper, we argue that malicious insiders can generate poisoning data to bias the worker selection using GANs. By replacing a portion of the historical data of the workers with the generated data, insiders can degrade the performance of victim workers' behavioral models by introducing misclassifications specifically related to the target class, without being detected. An overview of the proposed insider attack in the MCS platform is illustrated in Figure 1. Overall, the system includes the following modules:

- **Data preprocessing:** This module is responsible for collecting data and preparing it for training. Initially, contextual data obtained from historical tasks is collected and stored in the platform. This data provides insights into various factors influencing task completion by the workers, and can be further categorized into worker-related features and task-related fea-

tures. Each training point is labeled based on whether or not the tasks were canceled by the workers. Then, the training data undergoes pre-processing steps, which include data balancing using Synthetic Minority Over-sampling Technique (SMOTE), replacing missing values using Principal Component Analysis (PCA), and normalization. These steps are essential to ensure that the dataset is ready for training and the models yield optimal performance [25].

- **Insider attack:** In this module, an insider generates and injects poisoning points into the victim workers' training data to reduce their chances of being selected by the platform. First, a PGAN model is trained to generate poisoning points that are hard to detect by an outlier detector for each victim worker. The dataset used to train the PGAN model includes worker-related and task-related features, which provide insights into the workers' past performance and the context that can influence the workers' participation. The insider then uses the trained PGAN models to generate poisoning data and injects them into the victim workers' training datasets before their behavioral models are trained.
- **Behavioral model training:** This module is responsible for training workers' behavioral models, taking into consideration that malicious adversaries could inject poisoning data to skew the models' learning. Therefore, to ensure that the training data remains reliable and unbiased, the platform employs an Autoencoder-based outlier detection model to identify and remove anomalies from the workers' training datasets. Then, a deep learning model is built and trained for each worker to predict their behavior.
- **Behavioral-based selection:** This module uses the previously trained models to predict the workers' likelihood of canceling the task. Subsequently, a selection algorithm is executed to select a subset of workers such that the QoS of the task is maximized. After the task is completed, a feedback mechanism is deployed to update the selection metrics used

based on the workers' performance. The feedback mechanism adopted also evaluates the QoS achieved by the selected workers to ensure it meets a satisfactory value for the task publisher. Moreover, the platform evaluates the selected workers' contributions and performs the payment accordingly.

4. The Proposed Poisoning Attack

This section presents the proposed attack model, demonstrating the vulnerability of the behavioral models in MCS systems. It also describes the features used, the preprocessing steps performed, and the training process of the PGAN model, as described in the following subsections.

4.1. Threat Model

Let $W = \{w_1, w_2, \dots, w_k\}$ be a set of workers in the MCS system, where each worker i has a training dataset $D_i = \{(x_j, y_j) \mid x_j \in \mathbb{R}^n, y_j \in \{0, 1\}, j = 1, \dots, m\}$ that captures their behavior in terms of the willingness to complete the tasks assigned. The dataset consists of n -dimensional feature vectors x_j and their corresponding class labels y_j , where $y_j = 1$ indicates that the task was canceled by the worker and $y_j = 0$ indicates otherwise. The platform trains a behavioral model M_i for each worker, using their dataset D_i . In the proposed attack, an insider adversary seeks to compromise the models of a set of victim workers V , where $V \subset W$, by injecting poisoning points \mathcal{P}_i to the training dataset \mathcal{D}_i . The objective is to classify data points belonging to class $y_j = 0$ as $y_j = 1$, i.e. predict that a worker will cancel the task when they actually are more likely to accept it.

4.2. Features Description and Data Preprocessing

For each victim worker i , the insider trains a PGAN model using the original dataset \mathcal{D}_i . The features used in each dataset \mathcal{D}_i can be classified into worker-related features and task-related features, both of which can be used for effectively predicting the likelihood of task cancellation by the worker i . The

former provides insights into the workers' past performance and ability to complete the tasks, such as the cumulative rating and assigned workload on the day of the task. On the other hand, the latter includes features related to the tasks assigned and provides context that can influence the likelihood of successful completion, such as the weather conditions and the starting time of the task. In this work, the dataset, including the behavioral data of all workers utilized for training the PGAN models, is explained in Section 6.

Every dataset D_i stored in the platform undergoes the following preprocessing steps. Firstly, PCA is employed to address the issue of missing values in the dataset. Initially, the mean is used to replace the missing values, and then PCA is applied to transform the data to a lower dimensional space. Missing values are estimated after the data is transformed back to the original space based on the relationship identified between the variables when PCA was applied. Secondly, SMOTE is used to balance the dataset. This technique generates synthetic samples for the minority class through interpolation between existing instances and their nearest neighbors. Finally, Min-Max normalization is applied to scale all feature values within the range $[0, 1]$ [25].

4.3. Poisoning Points Generation using GANs

The proposed attack targets a deep learning behavioral model M_i that outputs the probability of task cancellation PC_i by a victim worker $i \in V$. The model takes a feature vector x as input and returns the probability of cancellation, as defined in (5).

$$M_i(x) = PC_i \quad (5)$$

The predicted class label \hat{y}_i is determined by applying a threshold to the model's output, as given by (6).

$$\hat{y}_i = \begin{cases} 1 & , \text{if } PC_i \geq 0.5, \\ 0 & , \text{otherwise} \end{cases} \quad (6)$$

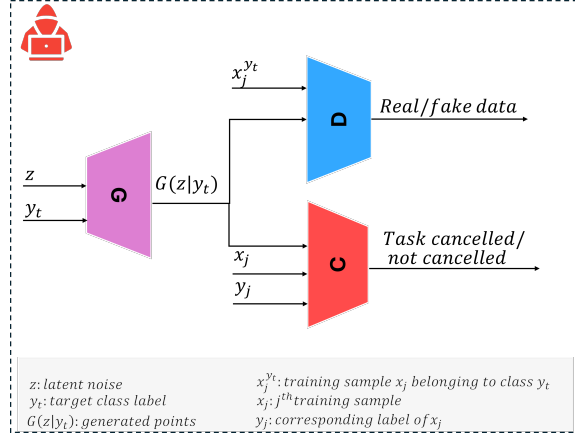


Figure 2: PGAN model representation

Before M_i is trained in the MCS platform, an insider uses \mathcal{D}_i to train a PGAN model and leverages it to generate a set of poisoning points \mathcal{P}_i . The main goal of the attack is to degrade the performance of M_i when trained using the poisoned dataset by increasing the error rate of misclassifying points from class 0 into the target class $y_t = 1$. The attacker injects the points \mathcal{P}_i into the training dataset \mathcal{D}_i by replacing a portion of the data points belonging to target class $y_t = 1$ with \mathcal{P}_i .

As discussed in Section 2.1, GANs and CGANs can be used to generate realistic data. This functionality can be exploited for malicious purposes by adding a third component to the GAN model, enabling the generation of adversarial data. The PGAN model used in this work comprises 3 main components: a generator G_i , a discriminator D_i , and a classifier C_i , as shown in Figure 2. The PGAN model is trained adversarially, where G_i competes against both C_i and D_i . During training, G_i aims to create points that increase the losses of both C_i and D_i . As a result, D_i exhibits a higher error at distinguishing between original and generated points, and C_i 's error in predicting workers' behavior increases. On the other hand, D_i and C_i aim to minimize their respective losses, despite the G_i 's attempts to disrupt their performance.

The interaction between G_i , D_i , and C_i can be modeled as min-max game

as shown in (7), where $V_{PGAN}(D_i, G_i)$ is the objective function for D_i and $W_{PGAN}(C_i, G_i)$ is the objective function for C_i .

$$\min_{G_i} \max_{D_i, C_i} \alpha V_{PGAN}(D_i, G_i) + (1 - \alpha) W_{PGAN}(C_i, G_i) \quad (7)$$

According to the formula in (7), D_i tries to maximize its objective function $V_{PGAN}(D_i, G_i)$ and C_i tries to maximize its objective function $W_{PGAN}(C_i, G_i)$. Note that both objective functions depend on G_i . Moreover, G_i tries to minimize the weighted sum of the objective functions $V_{PGAN}(D_i, G_i)$ and $W_{PGAN}(C_i, G_i)$, where α is the weighting factor that determines the relative contribution of both. By minimizing the combined objectives, G_i generates points that increase the error of D_i , and C_i , making D_i less effective at distinguishing genuine and generated points, while degrading C_i 's ability to predict the worker behavior correctly.

The parameter α plays a crucial role on the attack effectiveness and in shaping the distribution of the generated points. At $\alpha = 1$, the PGAN model behaves as a CGAN and its generator generates points that are highly similar to the target class y_t . However, this comes at the cost of reduced attack effectiveness as the generated data points fail to degrade the behavioral model's performance. On the other hand, when $\alpha = 0$, the PGAN's generator generates points that are further from the target class y_t and exhibit greater similarity to the other class, thus making the attack more effective. By carefully selecting the optimal value of α , which lies between 0 and 1, the attacker can train a PGAN model to generate subtle poisoning points that compromise the worker's behavioral model. Moreover, the attacker can identify the α value, which effectively exploits the decision boundary of the behavioral model, enabling the poisoning points to successfully execute the targeted attack by misclassifying data points from class 0 as 1.

As defined in (8), $V_{PGAN}(D_i, G_i)$ is the sum of the expected log-likelihood that the discriminator correctly identifies real and generated data, where x is sampled from the real data distribution $p_x(x|y_t)$ conditioned on the target

class y_t and z is sampled from a prior normal distribution $p_z(z|y_t)$ conditioned on the same class y_t . The first term represents the log-likelihood that the discriminator correctly classifies real points, where $D_i(x|y_t)$ is the probability that the x is real. On the other hand, the second term represents the log-likelihood that the discriminator identifies the generated data $G_i(z|y_t)$ as fake, where $1 - D(G_i(z|y_t))$ is the probability that the generated data are fake.

$$V_{PGAN}(D_i, G_i) = \mathbb{E}_{x \sim p_{\text{data}}(x|y_t)}[\log D_i(x|y_t)] + \mathbb{E}_{z \sim p_z(z|y_t)}[\log(1 - D(G_i(z|y_t)))]. \quad (8)$$

In addition, $W_{PGAN}(C_i, G_i)$, is expressed in (9), where L_{C_i} is the loss function used to train the classifier C_i and $\lambda \in [0, 1]$ is a weighting factor that balances the classifier's performance between classifying the poisoning points and the original training samples. During PGAN training, the classifier C_i is exposed to both poisoning points and original training points. The term $\mathbb{E}_{z \sim p_z(z|y_t)}[L_{C_i}(G_i(z|y_t))]$ evaluates the expected loss of the classifier on the poisoning points generated by the generator. In addition, the term $\mathbb{E}_{x \sim p_x(x)}[L_{C_i}(x)]$ represents the expected loss of the classifier on the original points. The best value of λ should be chosen such that the classifier's goal is to perform well on both original and poisoning points.

$$W_{PGAN}(C_i, G_i) = -(\lambda \mathbb{E}_{z \sim p_z(z|y_t)}[L_{C_i}(G_i(z|y_t))] + (1 - \lambda) \mathbb{E}_{x \sim p_x(x)}[L_{C_i}(x)]) \quad (9)$$

Algorithm 1 shows the pseudocode of the training process of the PGAN model, where the models G_i , D_i , and C_i are iteratively trained. In each iteration, a mini-batch of random noise samples z is sampled, which is then used by G_i to generate the points \tilde{x} . The discriminator then uses the generated data \tilde{x} and a mini-batch of training samples x_{y_t} , which belong to class y_t , and learns to differentiate between real and generated data by minimizing a loss function L_{D_i} . L_{D_i} is formulated as shown in (10), where y_f denotes the labels of the fake data generated by G , set to 0, y_r denotes the labels of the real data, set to 1, and L_{CE} represents the binary cross-entropy loss function.

Algorithm 1: Training Process for PGAN

Input: target class y_t , mini-batch size m , λ , α , number of iterations M

Output: Trained generator G_i , discriminator D_i , and classifier C_i

Build G_i , D_i , and C_i models

for iteration $j = 1, 2, \dots, M$ **do**

 Sample a mini-batch of m random noise samples z

 Use G_i to generate points \tilde{x} , where $\tilde{x} = G_i(z|y_t)$

 Select a mini-batch of m samples x_{y_t} belonging to class y_t

 Use D_i to make predictions on x_{y_t} and \tilde{x}

 Compute discriminator loss L_{D_i} using (10)

 Update D_i by minimizing L_{D_i}

 Select a mini-batch of m training features x and labels y

 Use C_i to predict the behavior of worker i using \tilde{x} and x

 Compute the classifier loss L_{C_i} using (11)

 Update C_i by minimizing L_{C_i}

 Compute generator loss using (12)

 Update G_i by minimizing L_G

Return trained G_i , D_i , and C_i

$$L_{D_i} = L_{CE}(D_i(\tilde{x}), y_f) + L_{CE}(D_i(x_{y_t}), y_r) \quad (10)$$

Subsequently, the classifier uses a mini-batch of training features x , their corresponding class labels y , and the generated data \tilde{x} , to learn the behavior of the victim worker. This is achieved by minimizing a loss function L_{C_i} , which is formulated in (11).

$$L_{C_i} = \lambda L_{CE}(C_i(\tilde{x}, y_t)) + (1 - \lambda) L_{CE}(C_i(x), y) \quad (11)$$

Finally, the generator model parameters are updated by minimizing the loss function L_{G_i} , defined in (12), where $\alpha \in [0, 1]$ controls the shape of the distribution of the generated points [10].

$$L_{G_i} = \alpha L_{CE}(D_i(\tilde{x}), y_r) + (1 - \alpha) L_{CE}(C_i(\tilde{x}), (1 - y_t)) \quad (12)$$

4.4. Behavioral models training and outlier detection

After training a PGAN model for each victim worker, the insider utilizes the trained generator G_i to produce poisoning data \mathcal{P}_i . This generated data is then injected into the training dataset D_i by replacing the original data points belonging to class y_t with \mathcal{P}_i . Before training the workers' behavioral models, the platform attempts to detect and remove outliers from each class in D_i to prevent biased model decisions.

To achieve this, an autoencoder was trained on the target class data, where an encoder learns a lower dimensional representation of the input, and then a decoder reconstructs the original data from the encoded vector. The target class data is then passed to the trained autoencoder model, and the reconstruction error for each data point is computed. Data points with high reconstruction errors are identified as outliers, based on a predefined threshold [36].

After removing outliers, every dataset $D_i = \{(x_j, y_j) \mid x_j \in \mathbb{R}^{12}, y_j \in \{0, 1\}, j = 1, \dots, n_i\}$ is used to train a deep learning model M_i to predict the behavior of the corresponding worker w_i . The training process involves minimizing a loss function L_i , which evaluates the classification error, given the predicted output \hat{y}_j and the true output y_j . The loss function used in this work is Binary Cross Entropy (BCE), which is also shown in (13). As a result, the parameters of the model M_i are adjusted so that the model can be used to make accurate predictions on new data.

$$L_i = -\frac{1}{n_i} \sum_{j=1}^{n_i} y_j \log \hat{y}_j + (1 - y_j) \log(1 - \hat{y}_j) \quad (13)$$

After training, each model is used at the selection stage to predict the task cancellation probability pc_i . The platform uses this probability along with other parameters to select a group of workers, as described in the following section.

5. Behavioral-based selection

Let $W = \{w_1, w_2, \dots, w_n\}$ denote the set of workers in the MCS system, where i can be mathematically represented as a tuple: $i = (id_i, r_i, l_i, M_i)$. Every worker is characterized by the following attributes: a unique identifier id_i , a reputation score $r_i \in [0, 1]$, the geographic location l_i , and the trained behavioral model M_i . The platform aims to select a subset of workers g , where $g \subset W$, to perform a given task k , such that the QoS is maximized. Each task k is represented as a tuple $k = (id_k, t_k, td_k, l_k)$, where id_k is the task id, t_k is the task's starting time, td_k is the task deadline, and l_k is the task's location.

5.1. The selection approach

The main contribution of this work is to propose a novel framework that shows how existing behavioral-based worker selection can be compromised through adversarial attacks using GANs. Therefore, a greedy selection algorithm is employed to find the subset of selected workers g based on their QoS values, as described in Algorithm 2. The QoS score for each worker QoS_i considers multiple factors, including latency τ_i , reputation r_i , and completion confidence $conf_i$, as defined in (14).

$$QoS_i = \tau_i \times r_i \times conf_i \quad (14)$$

The parameter τ_i represents how quickly the worker can arrive at the task location and has a decreasing value with the increase in the time required to reach the task. It can be evaluated as shown in (15), where tt_i^k is the worker's traveling time to the task estimated based on the current location l_i and td_k is the deadline by which task k should be completed.

$$\tau_i = [1 - \max(0, \min(\log_{td_k}(tt_i^k), 1))] \quad (15)$$

Additionally, r_i denotes the reputation value, reflecting the workers' current and historical performance, as shown in (16).

$$r_i = \gamma r_i + (1 - \gamma)\Omega_i \quad (16)$$

r_i denotes the reputation value before the completion of the task, whereas Ω_i represents the worker's most recent performance and is calculated as the percentage of successfully completed tasks, as shown in (17). The reputation score is updated by taking the weighted sum of these 2 parameters, where γ is the weight value given to the old reputation value.

$$\Omega_i = \frac{\text{Num of successfully completed tasks by worker } i}{\text{Num of assigned tasks to worker } i} \quad (17)$$

The third parameter used in the calculation of the QoS is $conf_i$, which represents the worker's confidence in completing the task. It can be evaluated as shown in (18) [25].

$$conf_i = 1 - pc_i \quad (18)$$

Algorithm 2: Greedy Worker Selection

Input: Set of workers W , task k , group size $GroupSize$

Output: Selected subset of workers S to perform task k

Initialize $g \leftarrow \emptyset$; // Start with an empty set of selected
workers

for each worker $i \in W$ **do**

Evaluate worker latency τ_i using Equation (15);
Calculate worker reputation r_i using Equation (17);
Compute QoS for worker i using Equation (14);

end

Sort workers in W by their QoS_i values in descending order;

while $|S| < GroupSize$ and W is not empty **do**

Select the worker i with the highest QoS_i from W ;
Add i to g ;
Remove i from W ;

end

5.2. Feedback mechanism and payment evaluation

After completing a task, the system updates the reputation values based on the workers' performance, as shown in (16). In addition, the feedback mechanism also includes evaluating and monitoring the QoS of the selected group of workers QoS_g , to identify potential security breaches that may have occurred [37]. The QoS metric is evaluated for a group of workers rather than for individual workers because MCS tasks are typically carried out collaboratively. Therefore, the service quality and reliability of the sensing outcome depend on the combined efforts and collective contributions of all members within the group. This step is critical to ensure the reliability of the system and support early detection of insider threats. Therefore, the value of QoS_g can serve as feedback, triggering an alarm for potential manipulation in the platform. However, the proposed attack does not affect the QoS values, demonstrating its ability to bypass this additional layer of detection. QoS_g can be evaluated as illustrated in (19), where r_g , τ_g , and $conf_g$ are the reputation, time score, and confidence of the selected group, respectively.

$$QoS_g = w_1 r_g + w_2 \tau_g + w_3 conf_g \quad (19)$$

r_g and $conf_g$ can be evaluated as shown in (20) and (21), respectively, by taking the minimum value of the reputation and confidence of the workers in the group.

$$r_g = \min_{i \in g} \{r_i\} \quad (20)$$

$$conf_g = \min_{i \in g} \{conf_i\} \quad (21)$$

In addition, the latency of the group is evaluated as shown in (22), where $|g|$ is the group size. It is calculated by taking the product of the average latency and the exponential of its negative standard deviation. This approach ensures that the overall value decreases if there is more variability in the individual scores [25].

$$\tau_g = \left(\frac{1}{|g|} \times \sum_{i \in g} \tau_i\right) \times e^{-\sigma(\tau_i)} \quad (22)$$

After the evaluation of QoS_g , workers are paid based on their contributions to the sensing task. The payment for each user is calculated as in (23).

$$\text{Payment} = \mu + \left(\frac{QoS_g - QoS_i}{QoS_g} \times \text{BP}\right) \quad (23)$$

The payment value is determined by multiplying a base payment BP by the worker’s contribution to the task, which is calculated as the ratio of the difference between QoS_g and QoS_i to QoS_g . This amount is then added to a fee μ that adjusts the price based on traffic conditions to encourage workers to participate [30].

6. Results

This section presents the simulation results showing the effectiveness of the proposed poisoning attack. Firstly, the Ride Austin dataset was filtered by worker ID, and each subset was treated as an individual training dataset. Subsequently, 86 PGAN models were individually trained, one for each worker, to generate the poisoning samples, which were then injected into the corresponding worker’s training dataset. This is done to ensure that the robustness of the attack is validated across a diverse set of 86 workers, and to prove that it is generalizable across different behavioral patterns.

Four experiments were conducted, each designed to evaluate a distinct aspect. The first experiment examines the effect of α on the models’ performance and on the distribution of the generated poisoning points. The second experiment evaluates the impact of varying poisoning percentages on the models’ performance. The third experiment compares the proposed approach with two existing benchmarks by assessing the detectability of the attack by an outlier detector and evaluating the models’ performance for varying poisoning percentages. Lastly, the fourth experiment analyzes how the attack affects worker selection.

6.1. Model architecture and training setup

The PGAN model used in our experiments consists of three main components: a generator, a discriminator, and a classifier. The generator network comprises four fully connected layers. The first layer combines the input noise and conditional label information, mapping them to a 100-dimensional vector. The subsequent layers are dense fully connected layers with 784 and 1024 neurons, respectively. The final layer maps the output to 12 dimensions, matching the dataset’s dimensionality. Each layer uses Leaky ReLU activations, and the final layer applies a sigmoid activation.

The discriminator and classifier architectures are similar, each consisting of four fully connected layers. The first layer combines the input data and conditional label, mapping them to a 784-dimensional vector. The next two layers are dense layers with 1024 and 512 neurons, respectively. The final layer outputs a scalar value representing the probability that the input is real or fake. Each layer uses Leaky ReLU activations, and the final output is passed through a sigmoid activation to produce a probability between 0 and 1.

The total number of trainable parameters in the proposed GAN-based model is approximately 4.29 million. This includes parameters from the Generator, Discriminator, and Classifier networks. These values demonstrate the complexity of the model, reflecting the number of learnable parameters that are optimized during the training process. Therefore, the model used is considered significantly less complex, compared to more advanced models, such as ResNet, which includes up to 11 million parameters. To train the PGAN models, an NVIDIA Tesla V100 GPU with memory of 32 GB was used. Moreover, the time taken to train a PGAN model for one worker is around 30 minutes.

Each PGAN model was trained for a total of 2000 epochs and with $\lambda = 0.8$. By setting the value of λ to 0.8, the classifier places more importance on minimizing the loss obtained from the generated data points while still maintaining some importance on the performance of the classifier on real data points. So as the training progresses, it becomes better at classifying the poisoning points correctly, which in turn drives the generator to enhance its ability to generate

better poisoning points that can mislead the classifier.

6.2. Dataset

The experimental results presented in this section utilize a real-life dataset to ensure that they effectively illustrate the impact of the attack on the models of workers exhibiting diverse behaviors in real-life scenarios. The dataset used in this work is the Ride Austin dataset [25]. It contains rides completed and canceled by a total of 86 workers over a period of 8 months in Austin, Texas, USA. This dataset also includes features that can be classified into worker-related or task-related features, as shown in Table 2. The worker-related features include the number of assigned and completed tasks on the day of the task, the worker rating, and the car rating. The worker rating represents the cumulative evaluation of the workers by the task requestors based on their historical performance. Similarly, the car rating is the cumulative rating of the worker’s vehicle over previous tasks. Furthermore, task-related features include the task starting time, requestor’s rating, price adjustment fee, and features representing the weather conditions on the day of the task, such as the precipitation, maximum and minimum temperature, wind speed, and wind gust. The requestor’s rating comprises the cumulative evaluation of the task requestor by the workers, whereas the price adjustment fee represents the additional amount of money added to the workers’ payment in response to real-time conditions, such as traffic, to motivate them to perform the task.

Table 2: Features used in the dataset

Category	Feature	Description
Worker-related	Number of assigned tasks	The total number of assigned tasks on the day of the task.
	Number of completed tasks	The total number of completed tasks on the day of the task.
	Worker rating	The cumulative evaluation of the workers by the task requestors.

Category	Feature	Description
	Car rating	The cumulative evaluation of the worker’s vehicle condition by the task requestors.
Task-related	Start time	The tasks’ starting time.
	Requestor rating	The cumulative evaluation of the task requestor by the workers.
	Price adjustment fee	An additional amount added to the workers’ payment in response to real-time conditions like traffic to encourage them to participate.
	Precipitation	A measure of the amount of rain falling at the time of the task.
	Maximum temperature	The maximum temperature on the day of the task.
	Minimum temperature	The minimum temperature on the day of the task.
	Wind speed	The speed of the wind at the time of the task.
	Wind gust	A sudden brief increase in wind speed beyond the average speed at the time of the task.

6.3. Performance evaluation metrics

To evaluate the effectiveness of the proposed attack on the performance of the models, the following metrics were used: False Positive Rate (FPR), False Negative Rate (FNR), Accuracy, Precision, Recall, and F1 Score. FPR measures the proportion of accepted tasks incorrectly predicted as canceled, while FNR is the proportion of canceled tasks incorrectly predicted as accepted (not canceled). Moreover, accuracy and F1 score measure the overall correctness of the model’s predictions, where accuracy is the percentage of times the model correctly predicted the workers’ behavior, and the F1 score is the harmonic mean of the precision and recall metrics. Precision is the proportion of tasks correctly predicted as canceled to the total number of tasks predicted as canceled. On the other hand, recall is the proportion of tasks predicted as canceled to the total number of canceled tasks.

Besides evaluating the behavioral models’ performance, the effectiveness of

the attack in evading detection is assessed by finding the average number of poisoning points detected as outliers by the autoencoder-based outlier detector [38]. Furthermore, the impact of the attack on worker selection is evaluated using the following metrics, as explained in section 5: the probability of task cancellation by workers derived from the behavioral model, the total payments made to victim workers, and the QoS achieved for the task.

6.4. Impact of α on the effectiveness of the attack

The main goal of the attack is to increase the FPR of the workers’ behavioral models, causing them to mislabel more samples from class 0 as 1. To determine the optimal value of alpha, a unique PGAN model was trained for each of the 86 workers in the dataset, across a range of $\alpha \in [0, 1]$ with increments of 0.1. These models were used to generate poisoning points, which were then injected into each worker’s training dataset by replacing features of the target class with the generated data. The resulting poisoned datasets were used to train workers’ behavioral models, and the effectiveness of the attack was evaluated on each worker’s test dataset.

As illustrated in Figure 3, the FPR of the behavioral models varies significantly with α . For instance, at $\alpha = 1$, the injection of the generated points to the workers’ training datasets results in the lowest FPR. This is because the generated points closely resemble the target class. However, lower alpha values resulted in higher FPRs, with the peak value of 0.12 achieved at $\alpha = 0.1$. Therefore, this proves that this is the optimal value for α , as it most effectively exploits the decision boundary of the behavioral models, thereby maximizing the success of the attack. Figure 4 further illustrates the impact of the attack on the performance of the behavioral models by presenting the F1 scores. It can be observed that, at $\alpha = 0.1$, the F1 reaches its lowest point, with a value of 0.92, which further proves the effectiveness of the attack at this α value.

The value of alpha also plays a crucial role in shaping the distribution of the generated points. To further demonstrate its effect, the t-Distributed Stochastic Neighbor Embedding (t-SNE) dimensionality reduction technique was applied

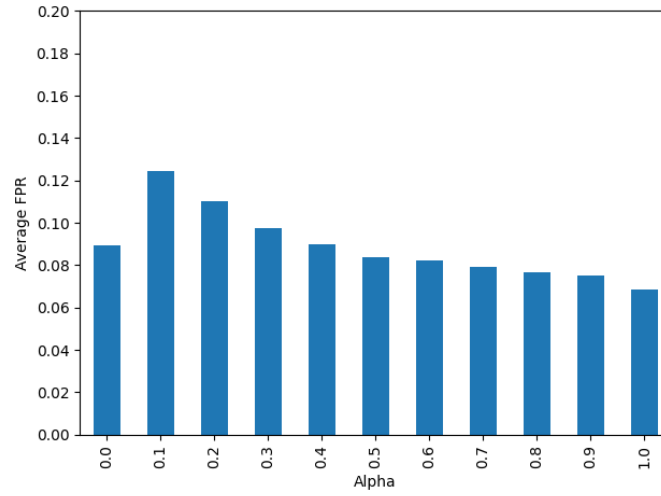


Figure 3: Average FPR of the workers' poisoned models for varying values of alpha

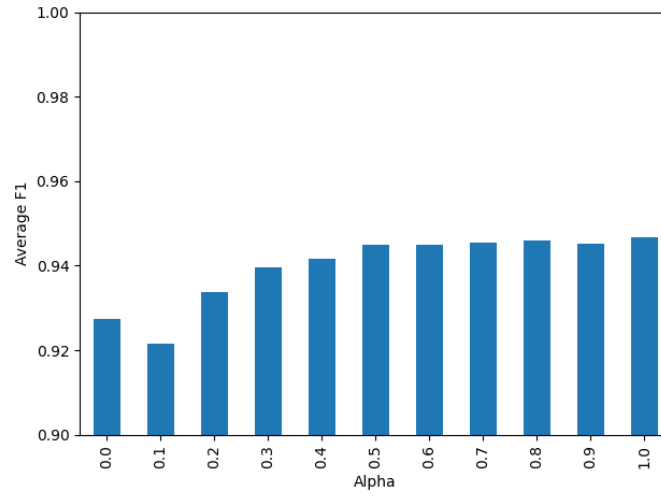


Figure 4: Average F1 scores of the workers' poisoned models for varying values of alpha

to one of the workers’ poisoned datasets. As shown in Figure 5a, when $\alpha = 0$, the generated poisoning points are positioned further from the target class (class 1) and exhibit more similarity to the other class (class 0). This is because, at this α value, no importance is given to generating points that are similar to the target class to evade detection. Instead, the model prioritizes generating points that compromise the model’s performance. Figure 5b illustrates the distribution of the original and poisoning points generated using $\alpha = 0.1$. It can be observed that the poisoning points are spread between class 0 and class 1, reflecting the model’s goal is to generate points that both compromise the model’s performance and partially resemble the target class to evade detection. Additionally, the points generated with $\alpha = 1$, as shown in Figure 5b, closely resemble the target class. In this case, the PGAN behaves as a CGAN to generate points similar to class 1 without compromising the model’s performance.

To further show the impact of alpha across different behavioral patterns, t-SNE was applied to each worker’s poisoned dataset. After reducing the data to two dimensions, the centroid of class 1 data was evaluated. Subsequently, the Euclidean distance was calculated between each class 1 point and its centroid. Finally, a threshold of 5% is applied to these distances to identify the most dissimilar points to the target class.

Figure 6 shows the percentage of the poisoning points deviating from the target class distribution, where each data point is the outcome of averaging the results for all workers. As shown in the figure, the highest fraction is observed at $\alpha = 0$. This is because the generator prioritizes increasing the error of the target classifier without any constraints related to the detectability of the generated poisoning points. As the value of α increases, less weight is given to reducing the error of the classifier, and more weight is given to increasing the error of the discriminator. Therefore, the percentage of poisoning points detected starts decreasing until it reaches its lowest value at $\alpha = 1$. At this value, the generator’s goal is to generate points that increase the error of the discriminator only. Consequently, the generated points become more similar to the points of the target class and the PGAN effectively behaves like a CGAN.

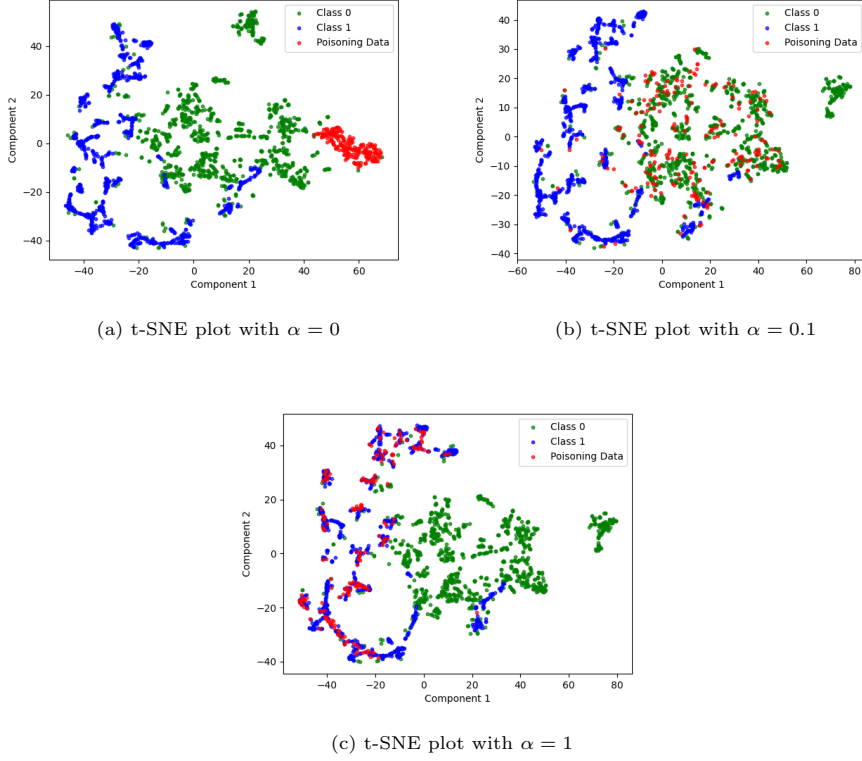


Figure 5: T-sne plots showing the distribution of poisoning points for different α values: (a) $\alpha = 0$, (b) $\alpha = 0.1$, and (c) $\alpha = 1$.

6.5. Impact of varying poisoning percentages on the model performance

Next, the workers' training datasets were poisoned with varying percentages of the target class data. The poisoning points were generated by GAN models trained using $\alpha = 0.1$ and $\lambda = 0.8$. Figure 7 shows the average FPR and FNR of the workers' models for varying poisoning percentages. A clear increase in the FPR is observed, starting at 7% with no poisoning points injected and rising to 33% at 80% poisoning. This highlights the effectiveness of the poisoning strategy in degrading the model's performance, resulting in the misclassification of inputs belonging to class 0 as 1. Additionally, the FNR remains low and relatively unaffected, ensuring the attack remains focused on increasing the FPR.

Additionally, Figure 8 shows that accuracy, F1 score, and precision decrease

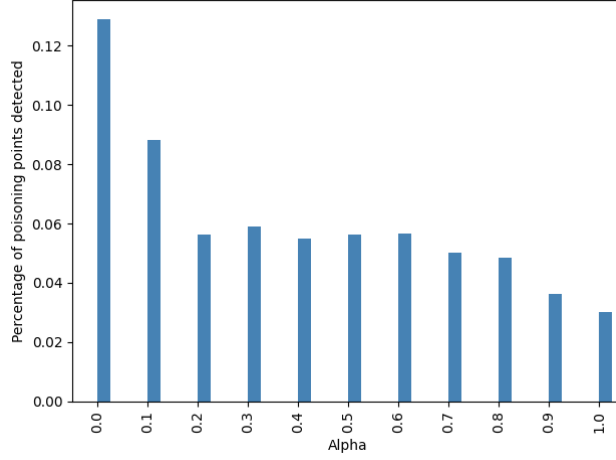


Figure 6: Fraction of poisoning points detected averaged for all workers’ models

significantly as a result of the attack. Compared to the model trained on clean data, accuracy precision and F1 drop by 15%, 20%, and 13%, respectively, at 80% poisoning. However, recall remains relatively stable, decreasing by only 3% at 80% poisoning. This indicates that while the attack significantly affects the model’s overall predictive performance, its ability to identify true positives is only slightly impacted. This outcome aligns with the objective of the attack, which focuses on flipping negative class labels to positive while minimizing interference with the predictions of the positive class.

Besides looking at the class predictions, it is also important to consider the impact of the attack on the task cancellation probabilities. Figure 9 shows the cancellation probabilities averaged for all workers for varying poisoning percentages. As illustrated, these probabilities increase as the poisoning percentage increases. This supports the attack’s objective of reducing the victim workers’ chances of selection. In addition, this also shows that even if the predicted labels don’t change, the attack can still affect the selection process, as the algorithm relies on probabilities rather than solely on labels.

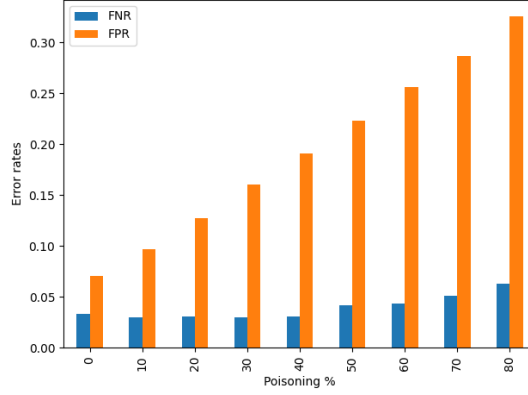


Figure 7: Models’ average error rates for varying poisoning percentages

6.6. Comparison with Benchmark: Detectability and Model Performance

In this section, the proposed attack is compared to [35], which utilized the label-flipping attack method. Label flipping is a commonly used technique to poison AI models during the training phase. The label-flipping attack was implemented by randomly selecting a fraction of data points from class 0 and flipping their labels to 1. The proposed attack is also compared against the feature manipulation attack, where noise is added to randomly selected data points from class 0 to shift them closer to the distribution of class 1. This attack does not alter the original class labels [39].

The performance of the attacks is compared based on two aspects: their detectability by an autoencoder-based outlier detector and their impact on the models’ performance. A consistent approach in training the outlier detector models was adopted for all attack methods to ensure a fair comparison of their detectability. The training of all autoencoder models was conducted for the same number of epochs and using the same architecture. The encoder network includes a fully connected dense layer with a ReLU activation function, while the decoder consists of another fully connected layer followed by a Sigmoid activation function. Each autoencoder model was trained for a total of 50 epochs. Additionally, to assess the impact of the attacks on the performance of

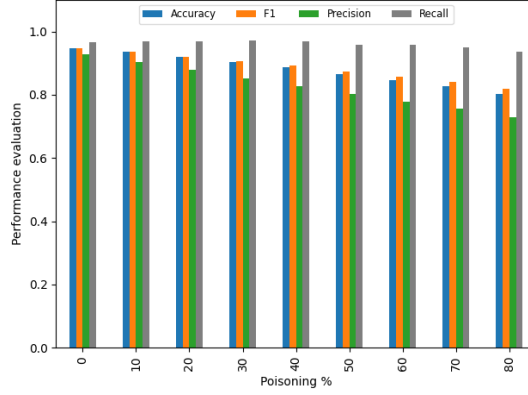


Figure 8: Models performance evaluation averaged for all workers for varying poisoning percentages

the workers’ poisoned models, all classifiers were trained for the same number of epochs and using the same model architecture. The number of epochs used in the simulations is 2000.

Figures 10a and 10b show the average number of poisoning points detected by an autoencoder-based outlier detector using 5% and 10% thresholds, respectively. This aligns with commonly used threshold values in the literature for anomaly detection, as in [36]. As illustrated, the proposed PGAN-based attack yields a lower average number of poisoning points detected at both threshold levels, which demonstrates its effectiveness in generating subtle and less detectable poisoning points, thus making it a more stealthy attack.

Moreover, Figures 11 and 12 show the FPR and FNR of the poisoned behavioral models using the proposed, label flipping, and feature manipulation attacks, respectively. As illustrated in the figures, the proposed attack demonstrates superior performance compared to both benchmark approaches. Firstly, it achieves higher FPRs than the feature manipulation attack. In addition, although the label flipping attack achieves higher FPRs compared to the PGAN-based attack, the proposed method maintains the FNRs closer to the original value obtained without any poisoning, thus achieving its intended goal more effectively.

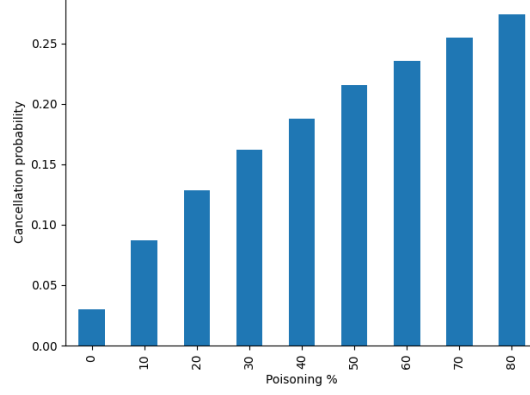


Figure 9: Task cancellation probabilities averaged for all workers

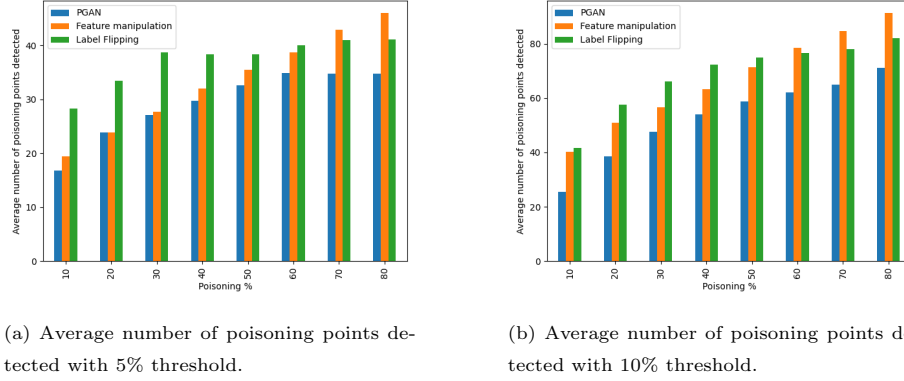


Figure 10: Average number of poisoning points detected by an autoencoder

6.7. Impact of the attack on the selection

The experimental results presented in this section assess the impact of the attack on worker selection across a range of tasks, thereby demonstrating that a one-time poisoning of victim workers’ models can lead to significant long-term effects. To assess the effectiveness of the attack, the total payment made to the victim workers after completing 100 tasks was evaluated, thereby effectively reflecting the diverse nature of tasks from the real-world. Additionally, 20% of the total workers in the dataset were randomly selected as the victims.

The results were then compared with the payment values obtained under

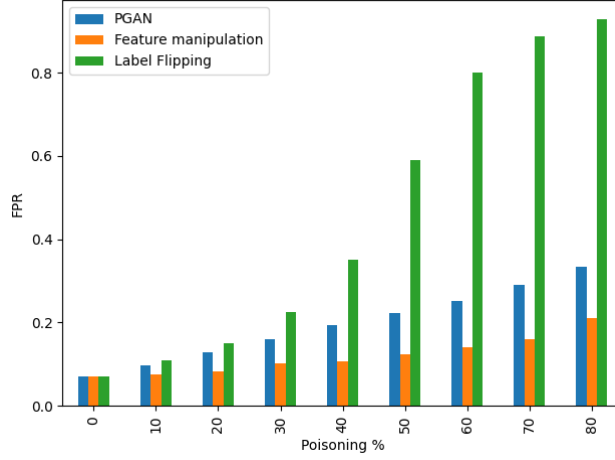


Figure 11: FPR of the proposed method and the benchmarks for varying poisoning percentages

the label flipping and feature manipulation attacks. Figure 13 presents the total payment received by the victim workers for 100 tasks, averaged across all workers. As illustrated, in a normal scenario with no poisoning points injected into the training dataset, the average payment received by the victim workers is 120. As the poisoning percentage increases, the proposed attack significantly reduces the total payment received due to being selected for fewer tasks. In contrast, the feature manipulation attack does not achieve the same effect. Additionally, although the PGAN-based attack results in higher payment values than the label-flipping attack, it is less detectable because of its more gradual reduction in payment as the poisoning percentage increases. The payment reduction can reach up to 47% at 80% poisoning, using the proposed PGAN-based attack. On the other hand, the reduction in payment achieved by label flipping at the same poisoning percentage can reach up to 93%. Consequently, the proposed attack is considered to be more successful, as it reduces the payment of victim workers while evading detection more effectively.

Figure 14 shows QoS_g averaged for all tasks for varying poisoning percentages. It can be seen that although more poisoning points are introduced, the PGAN-based attack maintains relatively stable QoS values. On the other hand,

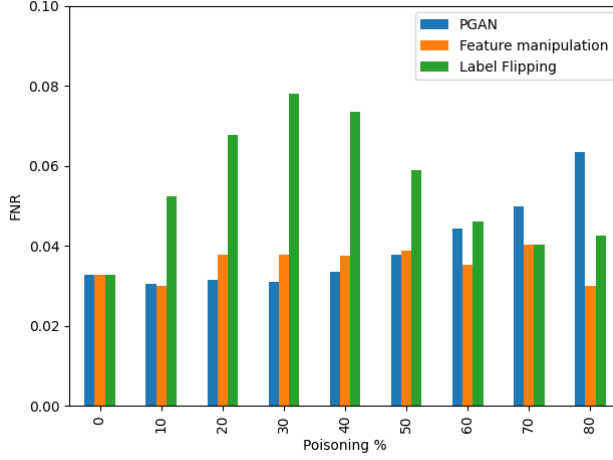


Figure 12: FNR of the proposed method and the benchmarks for varying poisoning percentages

the decrease in the QoS values obtained with the label-flipping attack is more significant. This proves the effectiveness of the proposed attack in bypassing other layers of detection.

6.8. Discussion

6.8.1. Summary of key findings

Based on the results presented in sections 6.4-6.7, it was shown that the proposed PGAN-based attack demonstrates superior performance compared to existing benchmark methods in multiple aspects. Firstly, it generates more subtle and less detectable poisoning points by an autoencoder outlier detector. Secondly, unlike label flipping attack, the proposed method is able to meet the objective of the attack more effectively by predicting that the victim workers will cancel the task when they actually are more likely to accept it. This is done while ensuring that the poisoned model does not erroneously predict task acceptance when the victim worker is, in fact, more likely to cancel. In other words, the proposed method effectively increases the FPR while preserving the FNR.

Thirdly, the PGAN-based attack successfully reduces the total payment re-

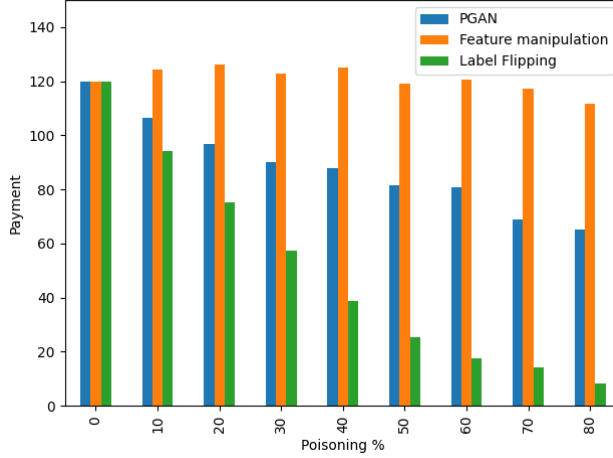


Figure 13: Payment received by the victim workers for 100 tasks, averaged across all workers

ceived by victim workers. However, unlike the benchmark label flipping attack, which results in a sharp and easily detectable decline in payment, the proposed method introduces this reduction more gradually. This further proves the superior performance of the proposed attack in terms of stealthiness, since a gradual reduction in payment makes it less likely for system administrators to suspect that an insider attack took place and can be attributed to normal fluctuations in worker performance. For instance, consider a scenario where a malicious insider poisons the victim workers’ models with 40% poisoning percentage. As shown in Figure 13, using label flipping attack, the average payment received by the victim workers after completing 100 tasks drops sharply to 40, compared to 120 in a normal scenario. In contrast, the PGAN-based attack reduces the payment to 90, a decline that is less drastic and is, therefore, less likely to raise suspicion from the victim workers or the management platform.

6.8.2. Deployment challenges and implications of the proposed attack

One of the key challenges in executing the proposed attack lies in tuning PGAN parameters, such as α , to ensure the effectiveness and stealthiness of the attack. Moreover, the proposed attack in this paper has broader implications

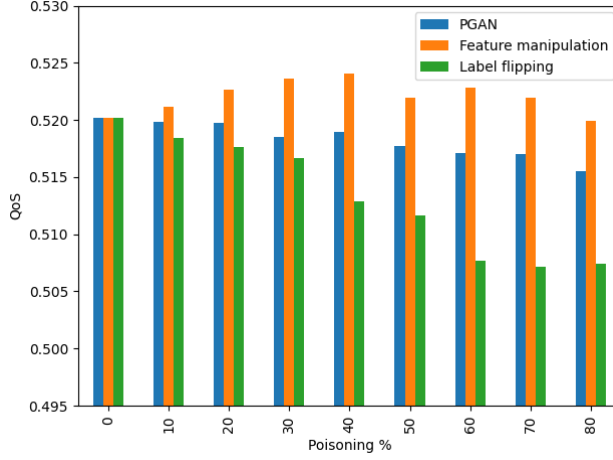


Figure 14: QoS_g averaged for all tasks

on the MCS system, beyond just the reduction in the revenue of the victim workers. It also poses a serious threat to the platform as it can, over time, lead to losing trust in the system’s reliability and trustworthiness. In fact, real-world MCS systems already face challenges in earning and maintaining workers’ trust, as workers may suspect that malicious insiders with access to sensitive resources are manipulating the system’s decisions [40]. Prior studies further highlighted this concern. For instance, in [41] and [42], workers in one of the ride-sharing platforms have reported experiences suggesting possible bias in the selection algorithm. For instance, some drivers shared that after completing nearly all the trips needed to earn a \$100 bonus, they faced an unusually long wait for the final ride, despite being in a busy area. As a result, this lead them to question the fairness of the system and caused them to start losing trust in the platform.

6.8.3. Defense mechanisms

To mitigate potential GAN-based attacks, several defense mechanisms can be adopted. Some of them serve as generic countermeasures, including conventional techniques like cryptographic methods and differential privacy approaches, which help protect sensitive data within the system. By implement-

ing robust privacy-preserving measures, such as those used in federated learning systems, the risks of GAN-based attacks can be mitigated, thus improving the platform’s trustworthiness [43]. However, these approaches come with trade-offs, which include increased computational cost and degradation in the performance of the machine learning models. The platform can also utilize methods that enhance the robustness of the system, such as ensemble learning and adversarial training. In ensemble learning, multiple models are trained, and their aggregated predictions are utilized after deployment, thus reducing the impact of poisoning attacks. In addition, in adversarial training, GAN-generated data are incorporated during the training stage to enhance the model’s robustness against adversarial threats [14].

7. Conclusion

In this paper, a novel adversarial attack on behavioral-based MCS worker selection is proposed, where insider adversaries inject stealthy poisoning points into victim workers’ datasets to reduce their revenue. By leveraging GANs, the attack targets vulnerable regions in the feature space to degrade model performance while ensuring the poisoning points remain undetected by outlier detectors. Simulation results using a real-life dataset demonstrate the effectiveness of the attack. First, the impact of α on the detection rate of poisoning points and model performance was examined. Second, the impact of varying poisoning percentages on the model performance was assessed. Finally, the attack’s effectiveness in reducing victim workers’ payment was evaluated for different poisoning percentages.

References

- [1] Uber, DeepETA: How Uber predicts arrival times using deep learning, available: https://www.uber.com/en-AE/blog/deepeta-how-uber-predicts-arrival-times/?uclid_id=6c27d554-32d3-48de-a931-7bc781617ce0, (accessed: 2025-04-24).

- [2] Waze, Using data to prevent crashes, available: <https://www.waze.com/wazeformcities/casestudies/using-data-to-prevent-crashes/>, (accessed: 2025-04-24).
- [3] R. Nasser, R. Mizouni, S. Singh, H. Otrók, Systematic survey on artificial intelligence based mobile crowd sensing and sourcing solutions: Applications and security challenges, *Ad Hoc Networks* 164 (2024) 103634.
- [4] A. Hussein, M. Raed, A. Al-Shaikhi, M. Mohandes, B. Liu, Crowd anomaly estimation and detection: A review, *Franklin Open* (2024) 100169.
- [5] F. A. Yerlikaya, Ş. Bahtiyar, Data poisoning attacks against machine learning algorithms, *Expert Systems with Applications* 208 (2022) 118101.
- [6] V.-T. Hoang, Y. A. Ergu, V.-L. Nguyen, R.-G. Chang, Security risks and countermeasures of adversarial attacks on ai-driven applications in 6g networks: A survey, *Journal of Network and Computer Applications* (2024) 104031.
- [7] W. Li, W. Meng, L.-F. Kwok, H. Horace, Enhancing collaborative intrusion detection networks against insider attacks using supervised intrusion sensitivity-based trust management model, *Journal of Network and Computer Applications* 77 (2017) 135–145.
- [8] IBM, Shedding Light on AI Bias with Real-World Examples, available: <https://www.ibm.com/think/topics/shedding-light-on-ai-bias-with-real-world-examples>, (accessed: 2025-04-25).
- [9] G.-Q. Zeng, Y.-W. Yang, K.-D. Lu, G.-G. Geng, J. Weng, Evolutionary adversarial autoencoder for unsupervised anomaly detection of industrial internet of things, *IEEE Transactions on Reliability*.
- [10] L. Muñoz-González, B. Pfizner, M. Russo, J. Carnerero-Cano, E. C. Lupu, Poisoning attacks with generative adversarial nets, *arXiv preprint arXiv:1906.07773*.

- [11] Z. Chen, B. Kantarci, Adversarial machine learning-driven fake task anticipation in mobile crowdsensing systems, in: 2021 IEEE International Conference on Service-Oriented System Engineering (SOSE), IEEE, 2021, pp. 57–63.
- [12] A. Prud’Homme, B. Kantarci, Poisoning attack anticipation in mobile crowdsensing: A competitive learning-based study, in: Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning, 2021, pp. 73–78.
- [13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, *Advances in neural information processing systems* 27.
- [14] C. Zhang, S. Yu, Z. Tian, J. J. Yu, Generative adversarial networks: A survey on attack and defense perspective, *ACM Computing Surveys* 56 (4) (2023) 1–35.
- [15] P. Singkorapoom, S. Phoomvuthisarn, Pre-trained model robustness against gan-based poisoning attack in medical imaging analysis, in: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, 2023, pp. 302–313.
- [16] K. Psychogyios, T.-H. Velivassaki, S. Bourou, A. Voulkidis, D. Skias, T. Zahariadis, Gan-driven data poisoning attacks and their mitigation in federated learning systems, *Electronics* 12 (8) (2023) 1805.
- [17] M. Abdullah, V. Bapu, K. Akash, A. M. Khan, M. Bhargavi, Label flipping attacks on federated learning: Gan-based poisoning and countermeasures, in: 2024 International Conference on Data Science and Network Security (ICDSNS), IEEE, 2024, pp. 1–6.
- [18] J. Zhang, J. Chen, D. Wu, B. Chen, S. Yu, Poisoning attack in federated learning using generative adversarial nets, in: 2019 18th IEEE international conference on trust, security and privacy in computing and com-

- munications/13th IEEE international conference on big data science and engineering (TrustCom/BigDataSE), IEEE, 2019, pp. 374–380.
- [19] S. Baluja, I. Fischer, Adversarial transformation networks: Learning to generate adversarial examples, arXiv preprint arXiv:1703.09387.
 - [20] Y. Song, R. Shu, N. Kushman, S. Ermon, Constructing unrestricted adversarial examples with generative models, *Advances in neural information processing systems* 31.
 - [21] X. Wang, K. He, J. E. Hopcroft, At-gan: A generative attack model for adversarial transferring on generative adversarial nets, arXiv preprint arXiv:1904.07793 3 (4) (2019) 3.
 - [22] Z. Zhao, D. Dua, S. Singh, Generating natural adversarial examples, arXiv preprint arXiv:1710.11342.
 - [23] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, D. Song, Robust physical-world attacks on deep learning visual classification, in: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1625–1634.
 - [24] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572.
 - [25] M. Abououf, S. Singh, H. Otrok, R. Mizouni, E. Damiani, Machine learning in mobile crowd sourcing: A behavior-based recruitment model, *ACM Transactions on Internet Technology (TOIT)* 22 (1) (2021) 1–28.
 - [26] M. E. G. Gendy, A. Al-Kabbany, E. F. Badran, Green crowdsensing with comprehensive reputation awareness and predictive device-application matching using a new real-life dataset, *IEEE Access* 8 (2020) 225757–225776.
 - [27] R. Nasser, R. Mizouni, H. Otrok, S. Singh, M. Abououf, M. Kadadha, A biometrics-based behavioral trust framework for continuous mobile crowd sensing recruitment, *IEEE Access* 10 (2022) 68582–68597.

- [28] X. Zhu, Y. Luo, A. Liu, W. Tang, M. Z. A. Bhuiyan, A deep learning-based mobile crowdsensing scheme by predicting vehicle mobility, *IEEE Transactions on Intelligent Transportation Systems* 22 (7) (2020) 4648–4659.
- [29] Q. T. Ngo, S. Yoon, Context-aware worker recruitment for mobile crowd sensing based on mobility prediction, *IEEE Access* 11 (2023) 92353–92364. doi:10.1109/ACCESS.2023.3308202.
- [30] R. Nasser, Z. Aboulhosn, R. Mizouni, S. Singh, H. Otrok, A machine learning-based framework for user recruitment in continuous mobile crowd-sensing, *Ad Hoc Networks* 145 (2023) 103175.
- [31] H.-N. Wei, G.-Q. Zeng, K.-D. Lu, G.-G. Geng, J. Weng, Moar-cnn: Multi-objective adversarially robust convolutional neural network for sar image classification, *IEEE Transactions on Emerging Topics in Computational Intelligence*.
- [32] E. Anthi, L. Williams, A. Javed, P. Burnap, Hardening machine learning denial of service (dos) defences against adversarial attacks in iot smart home networks, *computers & security* 108 (2021) 102352.
- [33] A. Abusnaina, A. Khormali, H. Alasmary, J. Park, A. Anwar, A. Mohaisen, Adversarial learning attacks on graph-based iot malware detection systems, in: 2019 IEEE 39th international conference on distributed computing systems (ICDCS), IEEE, 2019, pp. 1296–1305.
- [34] A. Rahman, M. S. Hossain, N. A. Alrajeh, F. Alsolami, Adversarial examples—security threats to covid-19 deep learning systems in medical iot devices, *IEEE Internet of Things Journal* 8 (12) (2020) 9603–9610.
- [35] N. Baracaldo, B. Chen, H. Ludwig, A. Safavi, R. Zhang, Detecting poisoning attacks on machine learning in iot environments, in: 2018 IEEE international congress on internet of things (ICIOT), IEEE, 2018, pp. 57–64.

- [36] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, L. Benini, A semisupervised autoencoder-based approach for anomaly detection in high performance computing systems, *Engineering Applications of Artificial Intelligence* 85 (2019) 634–644.
- [37] H. Li, J. Tong, S. Weng, X. Dong, T. He, Detecting a business anomaly based on qos benchmarks of resource-service chains for collaborative tasks in the iot, *IEEE Access* 7 (2019) 165509–165519.
- [38] M. Sakurada, T. Yairi, Anomaly detection using autoencoders with non-linear dimensionality reduction, in: *Proceedings of the 2nd Workshop on Machine Learning for Sensory Data Analysis*, ACM, 2014, pp. 4–11. doi:10.1145/2689746.2689747.
- [39] C. Zhang, Z. Tang, K. Li, Clean-label poisoning attack with perturbation causing dominant features, *Information Sciences* 644 (2023) 118899.
- [40] F. Aloraini, A. Javed, O. Rana, P. Burnap, Adversarial machine learning in iot from an insider point of view, *Journal of Information Security and Applications* 70 (2022) 103341.
- [41] B. V. Hanrahan, N. F. Ma, C. W. Yuan, The roots of bias on uber, *arXiv preprint arXiv:1803.08579*.
- [42] CBS News, Algorithmic wage discrimination: Ai’s role in wage disparities, available: <https://www.cbsnews.com/news/algorithmic-wage-discrimination-artificial-intelligence>, (accessed: 2024-11-11).
- [43] M. Yang, X. Wang, H. Qian, Y. Zhu, H. Zhu, M. Guizani, V. Chang, An improved federated learning algorithm for privacy preserving in cybertwin-driven 6g system, *IEEE Transactions on Industrial Informatics* 18 (10) (2022) 6733–6742.