

RESEARCH ARTICLE

Anomaly-Based Intrusion on IoT Networks Using AIGAN-a Generative Adversarial Network

ZHIPENG LIU¹, (Member, IEEE), JUNYI HU², (Member, IEEE), YANG LIU³, (Member, IEEE), KAUSHIK ROY³, (Senior Member, IEEE), XIAOHONG YUAN³, (Senior Member, IEEE), AND JINSHENG XU³, (Senior Member, IEEE)

¹Fidelity Investments, Boston, MA 02110, USA

²Department of Computer Science, University of Massachusetts (UMASS) Lowell, Lowell, MA 01854, USA

³Department of Computer Science, North Carolina Agricultural and Technical State University, Greensboro, NC 27401, USA

Corresponding author: Zhipeng Liu (zliu2@aggies.ncat.edu)

ABSTRACT Adversarial attacks have threatened the credibility of machine learning models and cast doubts over the integrity of data. The attacks have created much harm in the fields of computer vision, and natural language processing. In this paper, we focus on the adversarial attack, in particular the poisoning attack, against the network intrusion detection system (NIDS), which is often viewed as the first line of defense against cyber threats. We develop a generative adversarial network (GAN) in AIGAN, which uses deep learning techniques to generate adversarial data and to conduct an anomaly attack on IoT networks. To evaluate the effectiveness of our generator, we measure the similarities between real and fake data using the Jaccard similarity index, in addition comparing the F1-scores from four generic algorithms: multilayer perception, logistic regression, decision tree, random forest. We contrast the performance of ten machine learning classifiers experimented on two real IoT datasets and their fake adversarial samples. Our work highlights a vulnerable side of NIDS created by machine learning when attacked with adversarial perturbation.


INDEX TERMS Generative adversarial network, the IoT, machine learning, network intrusion detection system, poisoning attack.

I. INTRODUCTION

The art of deception is used to create a different persona to trick others in order to gain confidence. As the world progresses into the fifth industrial age, scammers no longer need to show their faces to continue to deceive [1]. Digital scams of cellular text messages or email became viable options [2]. Even high-value artworks no longer require paint or brush to complete [3]. With the quick development of the metaverse [4], the virtual world on top of our physical world, each of us has a virtual identity that's connected to our real-world identity. However, our virtual identities face the threat of adversarial attacks [5]. In recent years, adversaries have the ability to synthetically generate falsified data in the form of images [6] and voices [7]. Human beings can no

longer differentiate real and fake using eyes, one of the typical five senses humans rely so heavily on to navigate the physical world. Adversarial attacks share a common ground in adversarial sampling, which is the process of adding noise to input data to cause perturbations in neural networks (NN) to misclassify [8]. Lately, the techniques have been shifted and applied to network data [9], [10], [11]. Adversarial samples undermine data integrity, a foundation any machine learning model relies heavily on [12].

Attackers can approximate a white-box attack by using the notion of "transferability," which means that an input designed to confuse a certain Machine learning (ML) model is able to trigger a similar behavior within a different model [13]. In this work, we model a white-box attack by evaluating our examples against a variety of ML models, thus showing performance over a wide range of possible intrusion detection systems.

The associate editor coordinating the review of this manuscript and approving it for publication was Rajeeb Dey .

Cybersecurity is crucial in nearly all aspects of our lives, including social, economic, and political systems. According to a report from SlashNext,¹ phishing attacks alone are projected to increase by 61% in 2022 significantly. This alarming trend underscores the need for effective measures to counter such threats. AIGAN's capacity to generate realistic synthetic data addresses the challenge of obtaining labeled data for training in this domain, thus enhancing the quality and diversity of available datasets. By training intrusion detection models using AIGAN-generated adversarial instances, the system can develop resilience and effectively detect novel intrusion techniques.

Network Intrusion Detection System (NIDS) monitors network traffic to detect abnormal activities, such as attacks against hosts or servers. Trained on normal and attack traffic, ML classifiers offer the benefit of spotting novel differences in network traffic, and classification outcomes lead to further preventive or mitigation actions. Our previous works have demonstrated the effectiveness of our self-designed deep learning models on anomaly detection [14], [15], [16] and attack detection [17], [18], [19]. However, an increasing trend lately has been applying adversarial attacks on network datasets [20], [21]. In this work, we study the deception of adversarial ML by the following two steps. First, we generate fake data from our existing lab datasets [15], [17] using a deep learning (DL) technique in generative adversarial network (GAN) [22]. Next, we leverage the information at hand to conduct an adversarial attack on our NIDS [15], [17]. We want to demonstrate that our ML models are not immune from perturbations, resulting from different adversarial attacks. Although GAN is often applied as a mitigation solution while presented as a problem [23], [24], [25], for this work, we mainly focus on the sample generation ability of GAN [26]. The results of this work can assist organizations with a vulnerability analysis of their networks and raise awareness of adversarial attacks on cybersecurity. Given our results, a cyber adversary with some insight on a target network NIDS could effectively attack that network with the most effective perturbation algorithm.

Adversarial machine learning can be categorized into four types of attacks, poisoning, evasion, extraction, and inference, as classified by the National Institute of Standards and Technology (NIST) [27].

Poisoning is adversarial contamination of training data [28]. ML systems can be re-trained using data collected during operations. An example of this is when NIDS is frequently retrained with new data. However, an attacker can tamper with this data by inserting malicious samples that can cause problems during retraining [29].

Evasion attacks [30], [31] are the most prevalent type of attack. Spammers and hackers frequently try to avoid being detected by obscuring the content of their spam emails and malware. They conceal footprints to evade identification and appear legitimate without manipulating the

training data. An illustration of this technique is image-based spam, where the spam message is concealed within an attached image to prevent textual analysis by anti-spam filters.

Extraction [32], [33], or model stealing, involves an adversary probing a black box ML system in order to either reconstruct the model or extract the data it was trained on. This can lead to issues when either the training data or the model itself is sensitive and confidential. For example, model stealing could be used to extract a proprietary stock trading model which the adversary could then use for their own financial benefit.

Inference attacks [34] leverage overgeneralization of training data, a common weakness of supervised ML models, to identify data used during model training. Security concerns are raised for models trained on sensitive data such as medical records and personally identifiable information, as attackers can carry out their attacks without requiring knowledge or access to the target model's parameters. The increasing popularity of transfer learning and the availability of state-of-the-art ML models in the public domain enables tech companies to develop models based on public ones, providing attackers with easy access to information about the model's structure and type. However, membership inference relies heavily on overfitting resulting from poor ML practices, meaning a model that generalizes well to the real distribution of data should theoretically be more secure to membership inference attacks.

The attack we conduct is a poisoning attack. A poisoning attack happens when the adversary has access to either the data input or the detection models of the NIDS. Given the access, the adversary has the ability to inject bad data into the model's training pool, and hence the model learns something it shouldn't. For example, Steinhardt [35] reported that, even under strong defenses, a 3% training data set poisoning leads to an 11% drop in accuracy. An article [36] even goes as far as to say that poisoning attack is the biggest threat to AI models. The poison attack takes place before the model is trained. They aim at modifying a part of the data used for training purposes to corrupt the final model. Unlike a black-box attack, in which the adversary has neither information about the learning algorithms nor the training dataset, we proceed with the assumption that the adversary doesn't have access to the model algorithms but can modify the training data, otherwise known as a white-box setting.

The contributions of this work are:

- Perform an adversarial poisoning attack – we examine the effectiveness of adversarial examples, in particular with the focus of Poisoning attack in a white-box testing setting, against generic algorithms, ensemble models and deep learning models
- Create AIGAN, a supervisory-trained perturbation algorithm that pushes two self-improving deep learning models, generator and discriminator, to race against each other achieving adversarial learning

¹<https://slashnext.com/the-state-of-phishing-2022/>

- Generate synthetic data samples based on two existing real datasets, namely the CCD-INID-V1 [15] and CCD-IDSV1 [17], using AIGAN.

The rest of the sections are organized as such: Following the premises of the taxonomy of our work from the current section, Section II reviews the state-of-the-art in adversarial ML. Section III describes the methodology used for generating our adversarial samples. We elaborate further on the selected datasets, the intuition, and design of our GAN model, the experimental setup, as well as the metrics used to benchmark the perturbation effectiveness. Section IV presents the results based on our application of multiple model-based IDS and contrasts their predictive results on the original dataset and the generated samples. Additionally, we examine our findings. Section V concludes our work by summarizing our findings, limitations, challenges and constitute future research directions.

II. RELATED WORKS

The preeminence of adversarial attacks has seen increasing notice in the recent literature [37], [38], [39], [40], [41]. The methods such as the Monte Carlo (MC) simulation method [42], generic algorithms [43], swarm particles [44], variational autoencoder [45], GAN [22] - to create synthetic data, or adversarial samples, from existing data have recently been investigated on NIDS [46], [47], [48]. The common goal is adding noise to data and creating spatial differences between the real dataset and fake dataset, making it difficult for classifiers to recognize the difference. Our previous works [15], [17] have discussed the effectiveness of our deep learning-based models in intrusion detection. In this paper, we mainly focus on the literature overview of the development of GAN through the ages and describe how generating adversarial samples using GAN has been applied in cybersecurity.

As aforementioned, GANs have been applied in anomaly detection [49], [50], [51], [52], [53], as well as data augmentation [54], [55], [56], [57], [58]. To provide a solid foundation for adversarial training [59], which is the process of retraining models once the training sets are injected with new adversarial samples, data augmentation must not be overlooked. The most applied areas when GANs are used for data augmentation are in health [46], and several studies recently started exploring the technique as an alternative for data imbalance [60], [61]. Yang et al. [62] demonstrate that generative method can speed up the poisoned data generation rate by up to 239.38x compared to the direct gradient method. Miao et al. [63] used crowdsourcing to demonstrate the effectiveness of generative poisoning attack especially when the data points/datasets are huge/large. Arora et al. [64] presented a systematic literature review of GANs applications in the cybersecurity domain, including an analysis of specific extended GAN frameworks and currently used stable cybersecurity datasets. Their work compared how security professionals are employing GANs to produce amazing results in

fields, such as Intrusion Detection, Steganography, Password Cracking, and Anomaly Generation.

The work of Zenati et al. [65] uses GANs along with an encoding network to develop a unique anomaly score. This work assists with creating real-time detectors but does not contribute to the field of adversarial ML. Andresini et al. [66] created the MAGNETO system which uses GANs to create training data for IDS. They treat netflow data as 2D images and then use GANs to create training data to resolve the issue of unbalanced training sets. Even though both works gain high attention by attempting to apply transfer learning with GAN from computer vision to cybersecurity, generating adversarial examples is not their primary purpose.

In [67] GANs are used for data augmentation of network traffic data represented in 1D arrays of flow features, while a Random Forest is subsequently trained with the augmented training set. A similar approach is illustrated in [68], with GANs used for data augmentation, while Logistic Regression (LR), support vector machines (SVM) or Feed-forward Deep Neural Networks are trained for the classification. Shin et al. [69] propose the use of Sequence Generative approaches (SeqGAN and Seq2Seq) to generate new data in a sequence of network flows. Finally, Wang et al. [70] use random feature nullification to build an adversary-resistant deep learning model in malware detection.

The common trait of the research mentioned above is that they use GANs to generate rare adversarial samples and achieve a balanced condition before training the classification model. However, to the best of our knowledge, the performances are not as well as what GANs achieved with image data [71].

Piplai et al. [72] conducted an adversarial poisoning attack, which titled NAttack! Adversarial Attacks, by applying Fast Sign Gradient Method (FSGM) [73] to effectively perturb the ‘attack’ samples so that the classifier trained in the previous step, is forced to classify them as ‘non-attack’. The attack was carried out on a dataset from the IEEE BigData 2019 Cup:Suspicious Network Event Recognition challenge [74].

Hu and Tan [75] propose a GAN-based algorithm named MalGAN to generate adversarial malware examples which are able to bypass black-box machine learning detection models. Drawing ideas from genetic programming, Xu et al. [76] propose a generic method to evaluate the robustness of classifiers under attack. Their key idea is to stochastically manipulate a malicious sample to find a variant that preserves the malicious behavior but is classified as benign by the classifier. More recently, Alzantot et al. [77] introduced GenAttack, a gradient-free optimization technique that uses genetic algorithms for synthesizing adversarial examples in the black-box setting. Mosli et al. [78] created AdversarialPSO, a black-box attack that uses fewer queries to create adversarial examples with high success rates. AdversarialPSO is based on particle swarm optimization [79] and is flexible in balancing the number of queries submitted to the target compared to the quality of imperceptible adversarial

examples. However, they do not consider the constraints on their data fields when crafting their examples and thus risk causing the attacks to fail. The authors in [80] use a GAN to generate adversarial values for use in features input into their particle swarm optimizer that generates adversarial examples in IDS environments. While they acknowledge that features have constraints in this domain, they only modify a small number of features and modify features such as the Transport level protocol field which will cause the traffic to fail. Our work does not exclude any features from the original datasets. Additionally, their PSO heuristic attempts to minimize the distance between the original feature and the adversarial feature while our work does not attempt to do this. Devine and Bastian [81] used a ML approach for robust malware classification that integrates an MC simulation for adversarial perturbation with meta-learning using a stacked ensemble-based methodology.

III. METHODOLOGY

In this section, we give a technical description of the features used in the datasets. We then explain the details of the techniques employed for adversarial example generation. This leads to the layout of our computational setting. The section concluded with the listing of classifiers used for testing.

A. DATASET

We used two datasets to assist researchers in detecting cyber anomalies and to fill the void of publicly available IoT datasets [15], [17].

The first dataset is titled the Center for Cyber Defense (CCD) IoT Network Intrusion Dataset V1 (CCD-INID-V1), used in our previous work [15]. On top of creating the dataset, we propose a hybrid lightweight form of IDS—an embedded model (EM) for feature selection and a convolutional neural network (CNN) for attack detection and classification. The proposed method has two models: (a) RCNN: Random Forest (RF) is combined with CNN and (b) XCNN: eXtreme Gradient Boosting (XGBoost) is combined with CNN. RF and XGBoost are the embedded models to reduce less impactful features. We attempt anomaly (binary) classifications and attack-based (multiclass) classifications on CCD-INID-V1 to explore the effectiveness of these learning-based security models.

The second dataset is titled as the CCD-IDSv1. The dataset was created in an OpenStack environment as described in our previous work [17]. The dataset was labeled based on network flows as they were identified using our filter rules. In addition to creating the dataset, we developed two DL-based ensemble models: Ensemble-CNN-10 and Ensemble-CNN-LSTM. Ensemble-CNN-10 combines 10 CNN models developed from 10-fold cross-validation, whereas Ensemble-CNN-LSTM combines base CNN and LSTM models. The work highlights feature importance for both anomaly detection and multi-attack classification.

- ```

Step 1: Develop an application in Android Studio
Step 2: Reformat and setup Android Things OS on Raspberry Pi equipped with Rainbow HAT
Step 3: App starts, and Rainbow HAT starts sensing
Step 4: Sensor readings are sent live to Google cloud database Firebase
Step 5: Wireshark/tcpdump captures benign net traffic
Step 6: Inject various cyber attacks
Step 7: Wireshark/tcpdump captures malicious net traffic
Step 8: Captured packet files are feature engineered using NFStream
Step 9: Files are labeled and concatenated
Step 10: Data files exported

```

FIGURE 1. Data collection process for CCD-INID-V1.

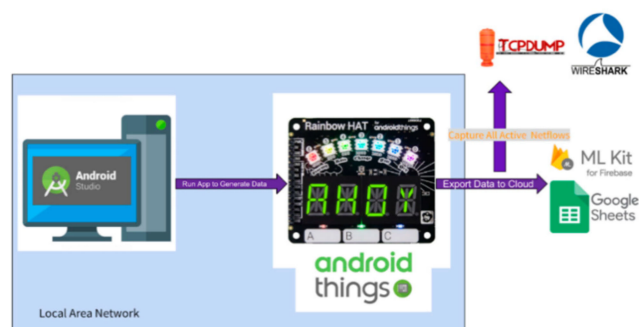


FIGURE 2. CCD-INID-V1's data architecture [15].

#### 1) CCD-INID-V1

This dataset was generated using a hybrid model, which combines physical devices with network virtualization functions. The data originate from our Raspberry Pis. The dataset contains 83 features generated from NFStream [82], which identifies bidirectional flows from netflows. The steps are illustrated in Figure 1. The generation architecture is illustrated in Figure 2. The dataset carries five attack types that were prevalent at the time of creation. The five attacks are ARP Poisoning, ARP DoS, UDP Flood, Hydra Brute-force and SlowLoris. The dataset was created to resemble the characteristics of IoT devices as they transmit data to cloud servers. Some of the features are categorical whereas some are continuous. The full listing with description of the features can be found in [15]. The dataset contains two variations: anomaly and multi-classed. In the anomaly dataset, attack is labeled as 0 and benign labeled as 1.

#### 2) CCD-IDS-V1

This dataset was created in a fully virtualized environment. The network architecture was created using OpenStack. OpenStack is a cloud operating system that controls large pools of computing, storage, and networking resources throughout a data center, all managed and provisioned through APIs with common authentication mechanisms.

The network environment, as shown in Figure 3, is implemented on OpenStack. Two internal networks, internal\_1 and internal\_2, are created. Five instances of Operating System

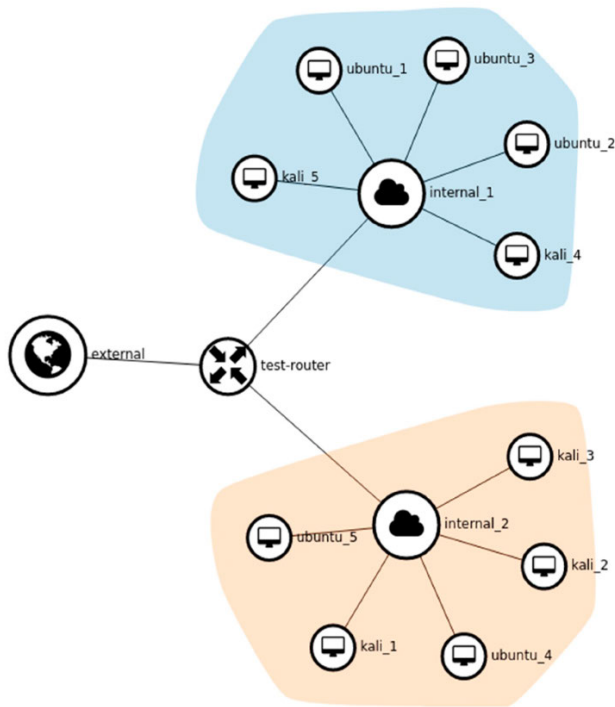


FIGURE 3. CCD-IDS-V1’s network architecture in OpenStack [17].

are created for each internal network, respectively. For this research work, Linux environments, Kali and Ubuntu, are used. Kali Linux is primarily used as a penetration testing environment that contains different attacks by default. Five different Kali systems are used to attack five different Ubuntu systems in parallel. The setting enables scalability for the number of connected devices as well as the attack diversity. The current attacks are carried out by Kali systems on Ubuntu systems at random time intervals. Every Kali system will infiltrate the Ubuntu system.

To extract the features from the raw PCAP files, first we convert the files into the Argus compatible format. Argus is a data network transaction auditing tool that categorizes and tracks network packets that match the libpcap filter expression into a protocol-specific network flow transaction model. Argus reports on the transactions that it discovers, as periodic network flow data, that are suitable for historic and near real-time processing for forensics, trending, and alarm/alerting. In this research, 25 features/attributes, shown in Table 2 of [17], are extracted from both malicious and normal traffic. These attributes consist of network flow information, including their statistical properties as well. The final CCD\_IDSv1 dataset is in CSV format for evaluation. The dataset is labeled in two different ways: for anomaly detection and threat or multi-attack classification. Anomaly detection is binary classification, so the dataset is labeled into two classes: normal and attack. For threat classification, each different attack is labeled, including normal usage for multi-class classification.

TABLE 1. Classification results to real data and fake data.

| Classifier | Runtime      | F1-score | Data |
|------------|--------------|----------|------|
| LR         | 0.197 s      | 0.62     | Real |
| LR         | 0.13 s       | 0.94     | Fake |
| RF         | 3.33 s       | 1.00     | Real |
| RF         | 3.34 s       | 0.97     | Fake |
| DT         | 0.28 s       | 1.00     | Real |
| DT         | 0.38 s       | 0.93     | Fake |
| SVM        | 11 mins 11 s | 0.62     | Real |
| SVM        | 12 mins 23 s | 0.95     | Fake |
| KNN        | 23 s         | 1.00     | Real |
| KNN        | 24 s         | 0.93     | Fake |
| NB         | 0.136 s      | 1.00     | Real |
| NB         | 0.128 s      | 0.78     | Fake |
| Adaboost   | 0.195 s      | 1.00     | Real |
| Adaboost   | 0.226 s      | 0.97     | Fake |
| GBoost     | 16.951 s     | 1.00     | Real |
| GBoost     | 17.46 s      | 0.96     | Fake |
| RCNN       | 0.45 s       | 1.00     | Real |
| RCNN       | 0.37 s       | 0.89     | Fake |

**B. ADVERSARIAL SAMPLES**

Based on the datasets with real data, CCD-INID-V1 and CCD-IDSv1, created in [15], we inject noise using GAN to generate two synthetic datasets. The generation process consists of two parts - adversarial training, adversarial data generation.

GAN is typically composed of two deep neural network models: discriminator (A), and generator (B). The discriminator attempts to discern if its inputs are from the genuine data set or from the adversarial data set. The generator’s task is to learn from the discriminator’s output and thus train so that its output may deceive the discriminator.

The training of GANs is based on a zero-sum or minimax game with two players, each one (A and B) trying to maximize its own benefits. The game converges when both players reach a point where changing their actions (updating the

TABLE 2. Classification results to real data and fake data.

| Classifier | Runtime      | F1-score | Data |
|------------|--------------|----------|------|
| LR         | 1.91 s       | 0.70     | Real |
| LR         | 12.92 s      | 0.91     | Fake |
| RF         | 14.9 s       | 1.00     | Real |
| RF         | 35.02 s      | 0.92     | Fake |
| DT         | 0.34 s       | 1.00     | Real |
| DT         | 2.37 s       | 0.87     | Fake |
| SVM        | 17 mins 8 s  | 0.49     | Real |
| SVM        | 27 mins 14 s | 0.90     | Fake |
| KNN        | 4 mins 58 s  | 1.00     | Real |
| KNN        | 5 mins 01 s  | 0.95     | Fake |
| NB         | 0.139 s      | 1.00     | Real |
| NB         | 0.228 s      | 0.75     | Fake |
| Adaboost   | 0.36 s       | 1.00     | Real |
| Adaboost   | 1.24 s       | 0.89     | Fake |
| GBoost     | 31.28 s      | 1.00     | Real |
| GBoost     | 32.45 s      | 0.93     | Fake |
| CNN        | 175.8 s      | 0.92     | Real |
| CNN        | 175.37 s     | 0.37     | Fake |

weights of neural networks) does not bring more benefits (or the loss functions for A and B cannot be further minimized). This point is the Nash equilibrium for the following equation:

- A(xi) is the discriminator’s estimate of the probability that real data instance xi is real.
- B(zi) is the generator’s output when given noise zi.
- A(B(zi)) is the discriminator’s estimate of the probability that a fake instance is real.
- The formula is derived from cross-entropy between real and generated distributions.

This equation shows that B tries to minimize the loss function while A tries to maximize it.

The Generator is a neural network model responsible for generating realistic samples from the target domain. The input for the generator model is a vector randomly sampled from a uniform or Gaussian distribution. This vector is used as a starting point for the G model to generate synthetic data

$$V(A, B) = \frac{1}{X} \sum_{i=1}^{i=X} \log A(xi) + \frac{1}{Z} \sum_{i=1}^{i=Z} \log(1 - A(B(zi)))$$

FIGURE 4. Formula of GAN.

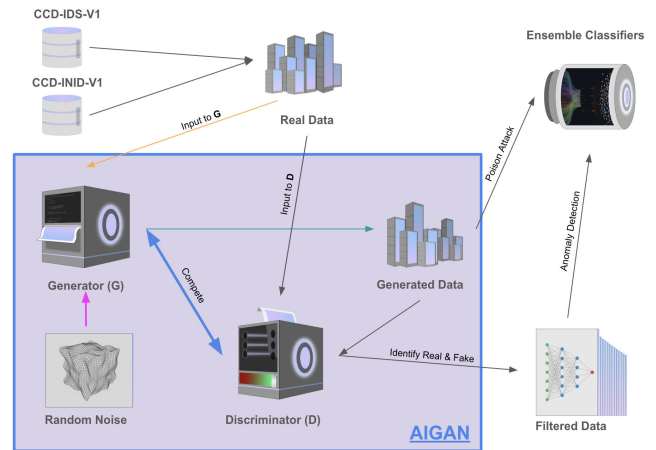


FIGURE 5. AIGAN’s architecture.

| Layer (type)     | Output Shape | Param # |
|------------------|--------------|---------|
| dense_16 (Dense) | (None, 15)   | 60      |
| dense_17 (Dense) | (None, 30)   | 480     |
| dense_18 (Dense) | (None, 50)   | 1550    |
| dense_19 (Dense) | (None, 83)   | 4233    |

FIGURE 6. Summary of AIGAN’s generator.

in the problem domain. This random vector represents a compressed version of features of the outputs referred to as latent features or a latent vector. In fact, during the training process, the Generator converts this random vector to meaningful data points. In this way, each new random vector drawn from the latent space is converted to a new output in the problem domain.

Adversarial data generation does not occur before adversarial training. But the process is sequential yet iterative as it is carried out over and over before reaching an acceptable stage. The process typically consists of five steps. Step 1 is to define the scope of research in order to select the right architecture for GAN. For this work, we apply our own AIGAN to generate IoT data in order to perform an anomaly intrusion attack on NIDS. AIGAN stands for an anomaly-based intrusion using generative adversarial network. The architecture is exhibited in Figure 5.

Step 2 is training the generator. The summary that describes the learning layers of AIGAN’s generator can be found in Figure 6. When the generator is trained, the discriminator is idle. During the generator training through any

| Layer (type)        | Output Shape | Param # |
|---------------------|--------------|---------|
| dense_20 (Dense)    | (None, 25)   | 2100    |
| dropout_6 (Dropout) | (None, 25)   | 0       |
| dense_21 (Dense)    | (None, 50)   | 1300    |
| dropout_7 (Dropout) | (None, 50)   | 0       |
| dense_22 (Dense)    | (None, 50)   | 2550    |
| dropout_8 (Dropout) | (None, 50)   | 0       |
| dense_23 (Dense)    | (None, 1)    | 51      |

FIGURE 7. Summary of AIGAN's discriminator.

random noise as input, the generator tries to transform the input into meaningful data. The generator gets random noise using latent dimensions. Intertwined with the next step, which is training the discriminator, the output from generator is passed into the discriminator to be classified as either real or fake. As the discriminator loss is calculated, backpropagation is performed on both the discriminator and the generator to calculate gradients. The gradients are used to update generator weights.

Step 3 is used for training the discriminator. Figure 7 demonstrates the composition of AIGAN's discriminator. When the discriminator is trained, the generator is idle. The discriminator is trained on the real dataset. Initially started off with only a forward path, no backpropagation needed with the first training of the discriminator. The discriminator classifies both real and fake data. The calculated loss helps improve its performance and penalizes it when it misclassifies real as fake or vice-versa. Then the weights of the discriminator are updated through discriminator loss.

In Step 4, the discriminator is trained on fake data. The samples which are generated by the generator will be passed into the discriminator. As the discriminator attempts to label real and fake, the feedback is presented to the generator repeatedly.

The discriminator is trained using the complete adversarial vectors from the generator along with benign vectors from the real data set. These vectors are then classified by the discriminator and previous classifiers as benign or malicious. The outputs of the two classifiers are compared in the following way: If both classify an input as malicious, then the feedback to the discriminator is labeled malicious. Otherwise, the classification is benign. In the next step, the generator is once again trained based on the feedback given by the discriminator and tries to improve performance.

These steps are repeated iteratively until the discriminator no longer can be fooled by the generator.

### C. EXPERIMENT SETUP

The overall accuracy performance of the proposed methodology is measured by analyzing the F1-score of the intrusion detection models learned. This is the harmonic mean of Pre-

cision and Recall, where Precision measures the ability of an intrusion detection system to identify only the attacks, while Recall can be thought of as the system's ability to find all the attacks. The higher the F1-score, the better the balance between Precision and Recall achieved by the algorithm. On the contrary, the F1-score is not so high when one measure is improved at the expense of the other. In addition, we consider Accuracy (that is measured in the evaluation of various competitors). This is the ratio of flows correctly labeled on all flows tested. All these metrics are computed on the testing set of the considered dataset.

#### 1) ENVIRONMENT

The efficiency performance is evaluated with the computation time spent training the intrusion detection model. The computation time is collected on a Windows machine with an Intel(R) Core (TM) i7-10870H CPU @ 2.21GHz and 64 GB RAM. All the experiments are executed on a single GeForce RTX 3080. The duration of computation is expressed in units of minutes and seconds.

#### 2) EVALUATIONS

We want to assess the effectiveness of adversarial malware samples against the state-of-the-arts ML models. During adversarial training, we look at the progression between generator loss versus discriminator loss. We compare the F1 scores of the models when applied on the original real datasets versus the F1 scores when applied on the synthetic datasets. The NIDS classifiers include random forest (RF), decision trees (DT), logistic regression (LR), k nearest neighbor (KNN), Naïve Bayes (NB), support vector machines (SVM), AdaBoost, Gradient Boosting (GBoost), Convolutional Neural Network (CNN) [17], RCNN (RF+CNN) [15].

Besides comparing the performance of the ten listed classifiers, we provide comparative analysis with the assistance of a toolkit named Table Evaluator [83]. TableEvaluator is a library to evaluate how similar a synthesized dataset is to a real data. In other words, it tries to give an indication into how real your fake data is. The toolkit applies four classifiers to compare the F1 scores of real and fake data and calculates the similarity score using Jaccard.

The F1 score is a machine learning evaluation metric that measures a model's accuracy. It is calculated by combining the precision and recall scores of the model. The precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive. In contrast, recall measures the proportion of correctly predicted positive instances out of all actual positive instances in the dataset. The Jaccard similarity index, sometimes referred to as the Jaccard similarity coefficient, compares members of two sets to see which members are shared and which are distinct. It's a measure of similarity for the two sets of data, with a range from 0 to 1. The higher the percentage, the more similar the two populations. The formula of Jaccard is shown

| Parameters                       | Settings                 |
|----------------------------------|--------------------------|
| Learning Rate                    | 0.001                    |
| Batch Size                       | 512                      |
| Dropout                          | 0.4                      |
| Conv2d_1 filter (filter size)    | 64 (2 × 2)               |
| Conv2d_2 filter (filter size)    | 128 (2 × 2)              |
| Conv2d_3 filter (filter size)    | 256 (2 × 2)              |
| MaxPooling2d_1                   | 2 × 2                    |
| Conv2d_4 filter (filter size)    | 128 (2 × 2)              |
| MaxPooling2d_2                   | 2 × 2                    |
| Dense_1                          | 768                      |
| Dense_2                          | 256                      |
| Dense_3                          | 128                      |
| Dense_4                          | 64                       |
| Output layer activation function | Softmax                  |
| Checkpoint                       | Best validation accuracy |

FIGURE 8. Parameter summary of the pre-trained CNN [17].

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

FIGURE 9. The formula of Jaccard similarity index.

in Figure 9. The four classifiers are DT, LR, multi-layer perceptron (MLP), and RF.

Besides the above metrics, we show the means and standard deviations between real and fake data. For each dataset, we show the cumulative sums and distributions to certain highlighted features. We also demonstrate the correlation matrix plots and principal component analysis (PCA) to both datasets.

### 3) PARAMETERS

We are performing the adversarial attack on our NIDS, which consists of eight generic machine learning algorithms and two pre-trained deep learning models.

The two pre-trained deep learning models are taken from our previous works [15], [17]. The structure and hyperparameters for our pre-trained CNN are shown in Figure 8. The summary to RCNN is as follows:

- An embedding layer of batch size 512
- A convolutional 2D layer of size 64 × 64 using RELU activation function
- A dropout layer with rate of 0.3
- A convolutional 2D layer of size 128 × 128 using RELU activation function
- A maxpooling layer
- A flatten layer
- A dense layer of size 128
- A dense layer of size 64
- A dropout layer with rate of 0.3
- A dense layer of size 16
- An output layer of 2 or n classes using Adam optimizer

The parameters for the eight generic machine learning classification algorithms are:

LR - (penalty = 'l2', \*, dual = False, tol = 0.0001, C = 1.0, fit\_intercept = True, intercept\_scaling = 1, class\_weight = None, random\_state = None, solver = 'lbfgs', max\_iter = 100, multi\_class = 'auto', verbose = 0, warm\_start = False, n\_jobs = None, l1\_ratio = None)

RF - (n\_estimators = 100, \*, criterion = 'gini', max\_depth = None, min\_samples\_split = 2, min\_samples\_leaf = 1, min\_weight\_fraction\_leaf = 0.0, max\_features = 'sqrt', max\_leaf\_nodes = None, min\_impurity\_decrease = 0.0, bootstrap = True, oob\_score = False, n\_jobs = None, random\_state = None, verbose = 0, warm\_start = False, class\_weight = None, ccp\_alpha = 0.0, max\_samples = None)

DT - (\*, criterion = 'gini', splitter = 'best', max\_depth = None, min\_samples\_split = 2, min\_samples\_leaf = 1, min\_weight\_fraction\_leaf = 0.0, max\_features = None, random\_state = None, max\_leaf\_nodes = None, min\_impurity\_decrease = 0.0, class\_weight = None, ccp\_alpha = 0.0)

SVM - (\*, C = 1.0, kernel = 'rbf', degree = 3, gamma = 'scale', coef0 = 0.0, shrinking = True, probability = False, tol = 0.001, cache\_size = 200, class\_weight = None, verbose = False, max\_iter = -1, decision\_function\_shape = 'ovr', break\_ties = False, random\_state = None)

KNN - (n\_neighbors = 3, \*, weights = 'uniform', algorithm = 'auto', leaf\_size = 30, p = 2, metric = 'minkowski', metric\_params = None, n\_jobs = None)

NB - (\*, priors = None, var\_smoothing = 1e-09)

AdaBoost - (base\_estimator = None, \*, n\_estimators = 50, learning\_rate = 1.0, algorithm = 'SAMME.R', random\_state = None)

GBoost - (\*, loss = 'log\_loss', learning\_rate = 0.1, n\_estimators = 100, subsample = 1.0, criterion = 'friedman\_mse', min\_samples\_split = 2, min\_samples\_leaf = 1, min\_weight\_fraction\_leaf = 0.0, max\_depth = 3, min\_impurity\_decrease = 0.0, init = None, random\_state = None, max\_features = None, verbose = 0, max\_leaf\_nodes = None, warm\_start = False, validation\_fraction = 0.1, n\_iter\_no\_change = None, tol = 0.0001, ccp\_alpha = 0.0)

## IV. RESULTS

### A. GENERAL RESULTS

Since the attack types from both datasets overlap, we can see how the adversarial poisoning attack performed when seen in different situations. The results also provide us with insights on potential weaknesses to the datasets for future improvements. The subsequent discussion of the results indicates where useful similarities are found between real and fake data. It should be noted that no features were dropped for the experiments in order to provide the most realistic results.

### B. CCD-INID-V1

In this section we demonstrate the findings for the dataset CCD-INID-V1.



Classifier F1-scores and their Jaccard similarities::

| index                       | f1_real | f1_fake | jaccard_similarity |
|-----------------------------|---------|---------|--------------------|
| DecisionTreeClassifier_fake | 0.5600  | 0.8800  | 0.4085             |
| DecisionTreeClassifier_real | 1.0000  | 0.5000  | 0.3333             |
| LogisticRegression_fake     | 0.5600  | 0.9600  | 0.3889             |
| LogisticRegression_real     | 0.7100  | 0.5000  | 0.3793             |
| MLPClassifier_fake          | 0.5600  | 0.8700  | 0.3072             |
| MLPClassifier_real          | 0.7100  | 0.7100  | 1.0000             |
| RandomForestClassifier_fake | 0.5600  | 0.9000  | 0.4085             |
| RandomForestClassifier_real | 1.0000  | 0.5100  | 0.3423             |

FIGURE 10. Classifier F1 and jaccard scores for CCD-INID-V1.

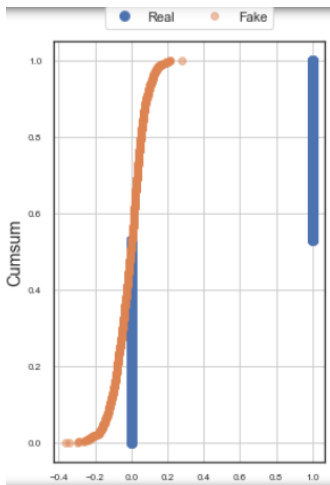


FIGURE 11. Cumulative sum for feature 'Expiration\_ID'.

In the table shown in Figure 10, the row index contains the name of the estimator and the name of the dataset. The second and third columns provide the F1-scores on show how they perform when predicting real and fake data. The last column shows the similarity score according to the Jaccard similarity index. Using row 2 as an example, the DT estimator is applied on the real data. The estimator correctly classifies data as real with a F1-score of 1.00 and classifies data as fake with a 0.50 F1-score. The similarity is consistently approximately around 0.4.

We select a few features that provide us with the best insights to the comparisons between real and fake data out of the 83 features. By looking at the cumulative sums of the selected features, we observe the generator successfully learn certain characteristics of real features, particularly in Figure 11. - Figure 16. The cumulative sums, also known as running totals, are the data sum's progression through time. It displays the total contribution of a certain measure against time. The real data is shown in blue, and the fake data is shown in orange. The more overlap between the two data, the harder to differentiate between real and fake.

In figures 17, 18 and 19, the distributions of selective features are shown. Orange resembles fake data and blue is real data in the histograms.

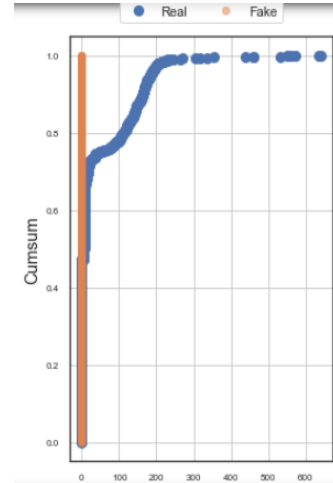


FIGURE 12. Cumulative sum for feature 'dst2src\_stddev\_ps'.

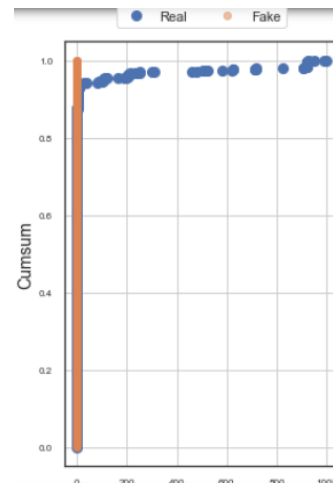


FIGURE 13. Cumulative sum for feature 'bidirectional\_min\_piat\_ms'.

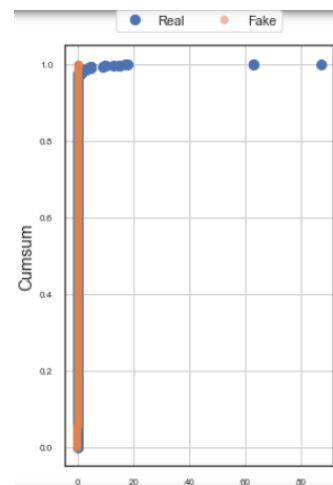


FIGURE 14. Cumulative sum for feature 'dst2src\_min\_piat\_ms'.

By examining the correlation between real and fake data in figures 20, 21, and 22, we observe that marginal differences exist between real and fake data.

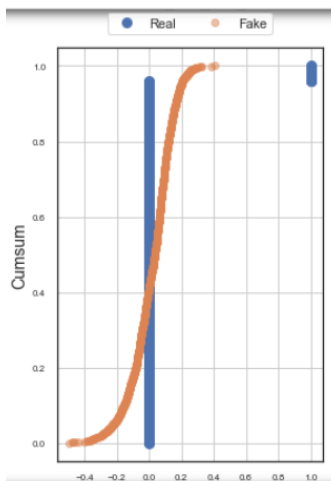


FIGURE 15. Cumulative sum for feature 'bidirectional\_syn\_packets.'

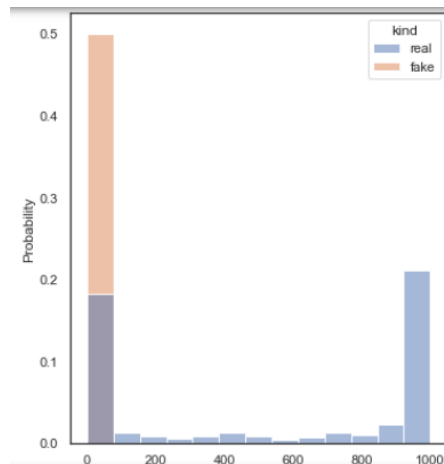


FIGURE 18. Distribution diagram for feature 'bidirectional\_duration\_ms.'

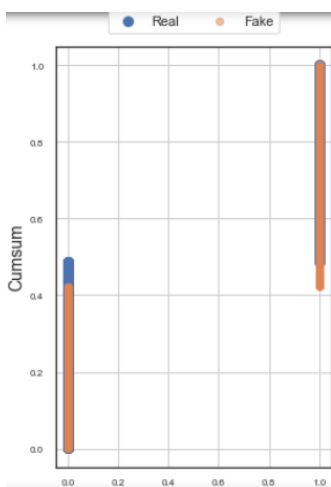


FIGURE 16. Cumulative sum for feature 'traffic\_type.'

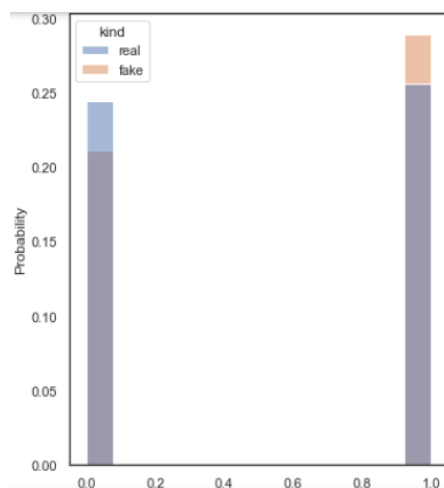


FIGURE 19. Distribution diagram for feature 'traffic\_type.'

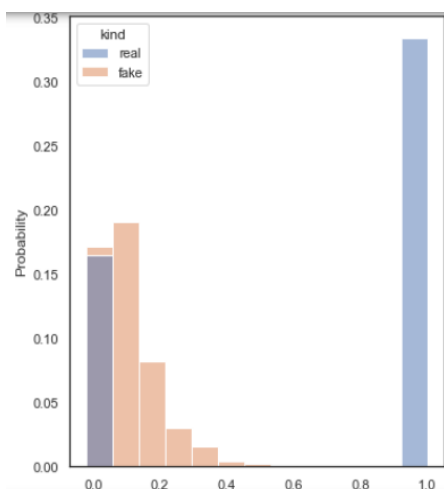


FIGURE 17. Distribution diagram for feature 'dst\_ip\_is\_private.'

In Figure 20, the correlations between features are less obvious, with majority of the figure showing whiteness, which stands for 0.00 or no correlation.

But in Figure 21, the features from the fake dataset are highly correlated. The redder shown means higher positive correlation whereas the bluer shown means higher negative correlation.

Figure 22 shows that real and fake has an evenly spread. The more redness means more difference whereas the whiteness means no difference. Figures 20, 21, 22 illustrate that for human’s set of eyes, it is visually hard to clearly identify a pattern of difference between the real data and the created synthetic fake data. This confusion is exactly how AIGAN is affecting decision making of the classifiers.

In Figure 23, the fake data is more evenly spread across the board compared to the real data.

In Table 1, we observe the perturbations caused by the fake data generated with AIGAN. For most classifiers, the F1-scores appear to suffer a drop in scoring from real data to fake data. Our pretrained RCNN model suffered a 11% drop and NB suffered a 22% drop.

And finally, we see the loss progressions between the competition of generator and discriminator in Figure 25. We can

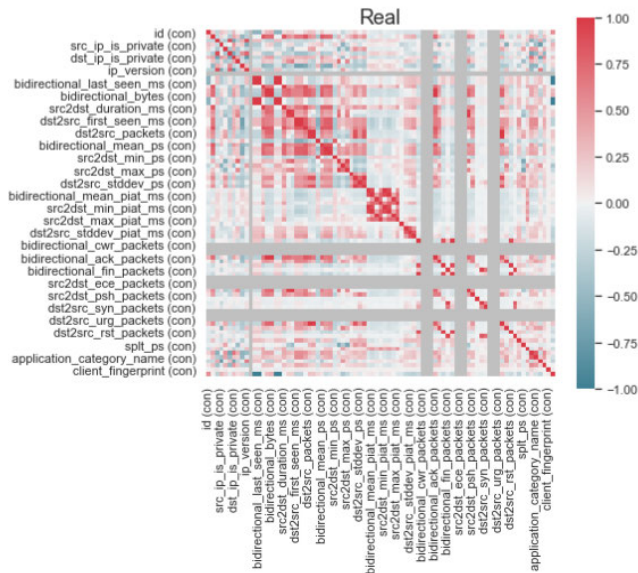


FIGURE 20. Correlation between features for the real dataset.

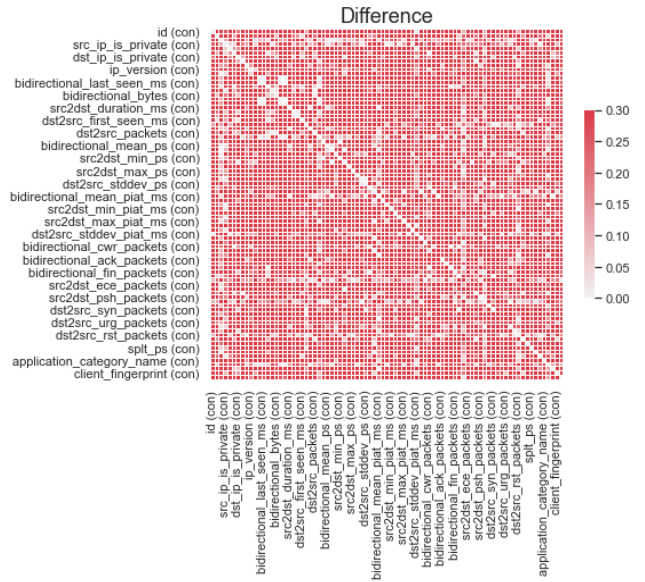


FIGURE 22. Difference between features for the real and fake dataset.

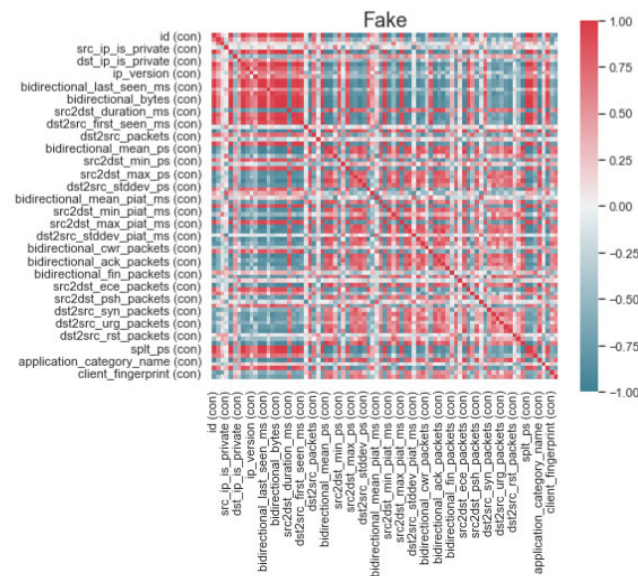


FIGURE 21. Correlation between features for the fake dataset.

see the generator attempts in the first 50 epochs to deceive the discriminator until it cannot do so. Discriminator is able to consistently handle the generator’s attempts.

C. CCD-IDSV1

Similar to Figure 10, using the Jaccard similarity index, the similarity score is consistently approximately around 0.4. The similarity score achieved 1.00 when applied MLP classifier on real dataset and 0.9231 when applied LR on real dataset.

Just as the previous dataset, we selected several features out of the 23 features from CCD-IDSV1 to provide insights.

Looking at Figures 26 to 33, we can see AIGAN was able to create the replicates to these features at certain points.

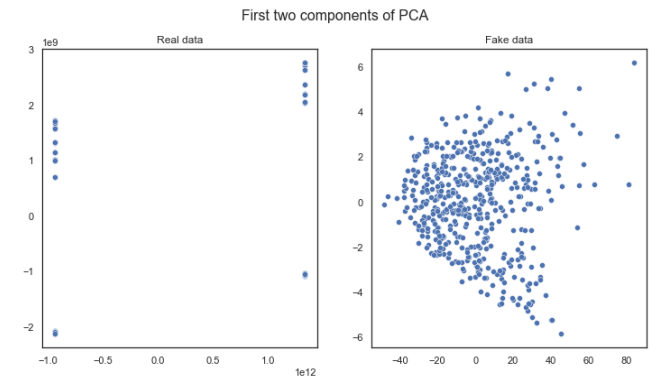


FIGURE 23. PCA between real and fake data.

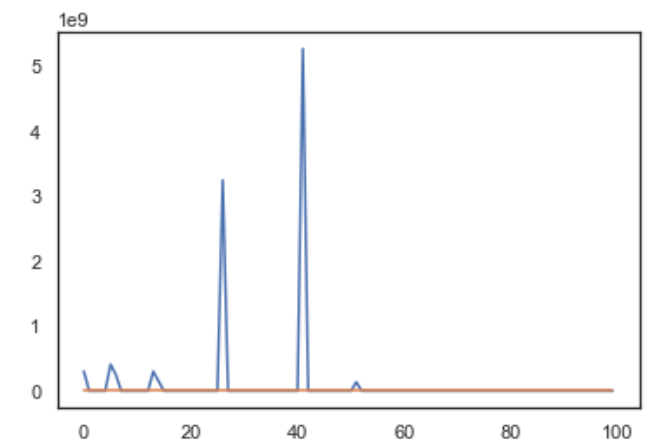


FIGURE 24. Generator (blue) loss and discriminator (orange) loss (100 epochs).

By examining the correlation between real and fake data in figures 34, 35, and 36, we observe extensive differences exist between real and fake data compared to CCD-INID-V1.

Classifier F1-scores and their Jaccard similarities::

| index                       | f1_real | f1_fake | jaccard_similarity |
|-----------------------------|---------|---------|--------------------|
| DecisionTreeClassifier_fake | 0.4300  | 0.8900  | 0.2658             |
| DecisionTreeClassifier_real | 1.0000  | 0.4700  | 0.3072             |
| LogisticRegression_fake     | 0.5700  | 0.9000  | 0.4388             |
| LogisticRegression_real     | 0.9700  | 0.9500  | 0.9231             |
| MLPClassifier_fake          | 0.5700  | 0.9600  | 0.3986             |
| MLPClassifier_real          | 0.9500  | 0.9500  | 1.0000             |
| RandomForestClassifier_fake | 0.5700  | 0.8900  | 0.4286             |
| RandomForestClassifier_real | 1.0000  | 0.4700  | 0.3072             |

FIGURE 25. Classifier F1 and jaccard scores for CCD-IDSV1.

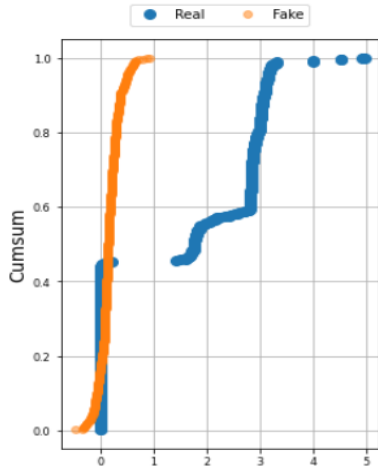


FIGURE 26. Cumulative sum for feature 'runtime'.

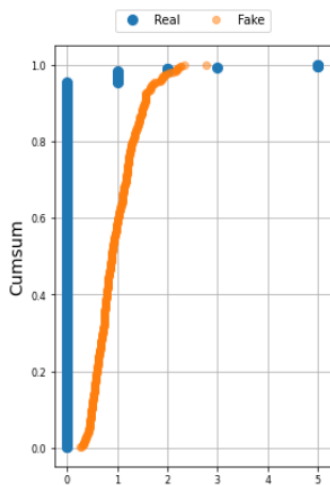


FIGURE 27. Cumulative sum for feature 'Proto'.

In Figure 34, we identify higher positive correlations among several features by spotting the clear redness. Figure 35 demonstrates that features have fewer positive correlations but have slightly more negative correlations. Combine the patterns we see in Figure 34 and Figure 35; Figure 36 confirms that differences between real and fake data is more noticeable since more bright red is shown and less white blocks exist in the figure.

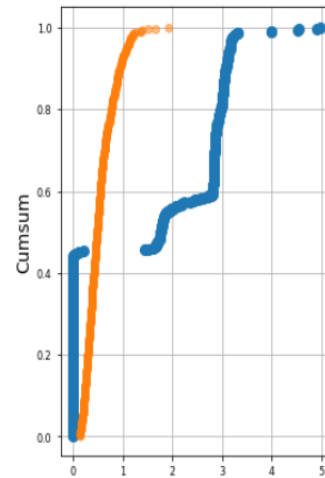


FIGURE 28. Cumulative sum for feature 'Sum'.

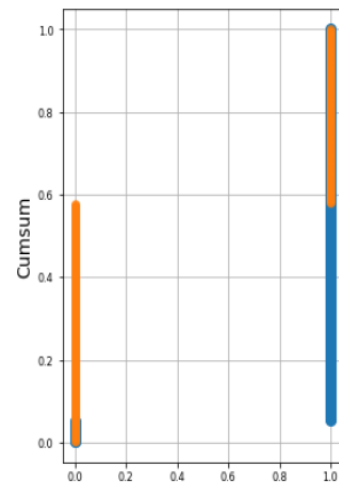


FIGURE 29. Cumulative sum for feature 'type'.

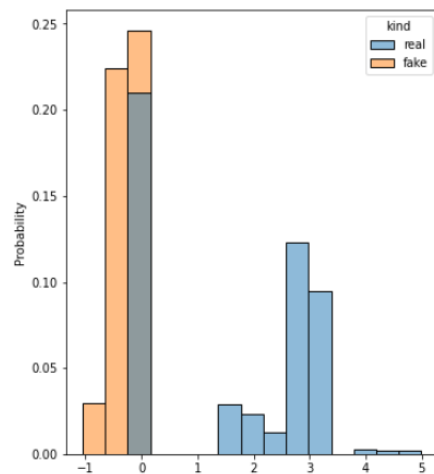


FIGURE 30. Distribution diagram for feature 'Dur'.

Identical to what was shown in Figure 23, the PCA spread for fake data is more random than the straight-line shape for real data.

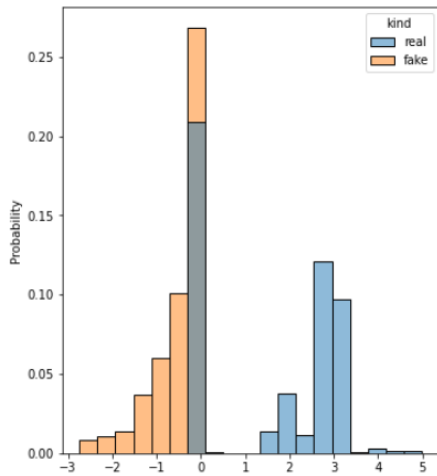


FIGURE 31. Distribution diagram for feature 'Max.'

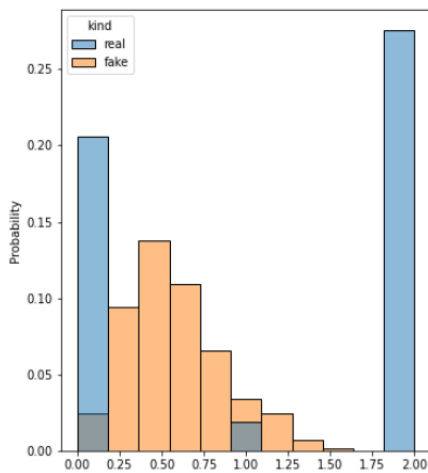


FIGURE 32. Distribution diagram for feature 'Cause.'

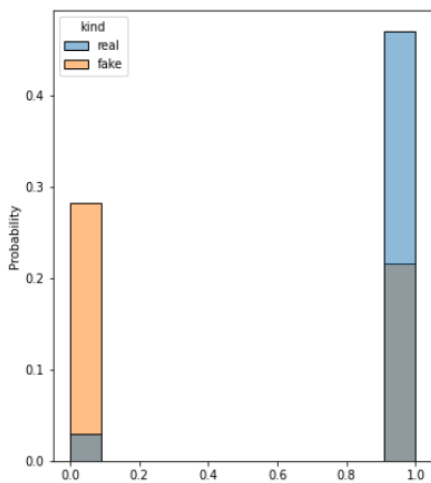


FIGURE 33. Distribution diagram for feature 'Type.'

In Table 2, just as it was the case for CCD-INID-V1, F1-scores prove perturbations exist under the influence of AIGAN. DT has a drop of 13% while NB has a drop of 25%

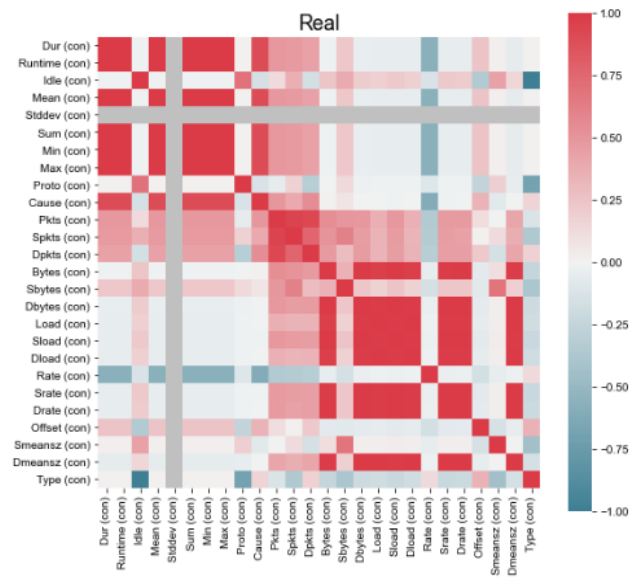


FIGURE 34. Correlation between features for the real dataset.

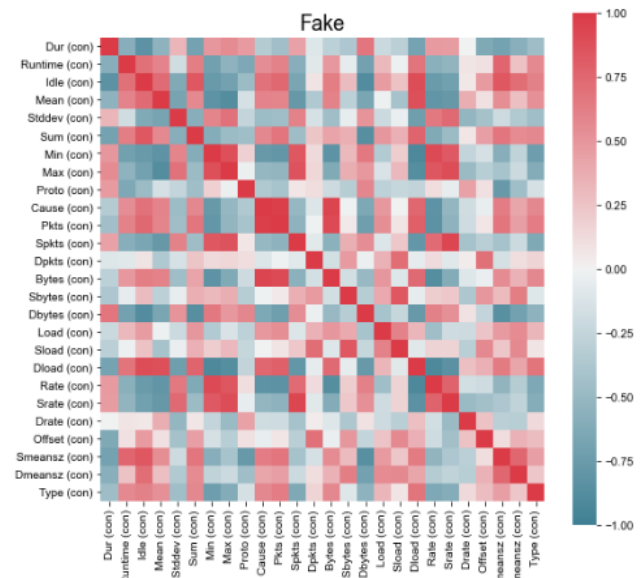


FIGURE 35. Correlation between features for the fake dataset.

in F1- score. Our pretrained CNN model suffered the most with a drop of 55%.

Just as the case of CCD-INID-V1, AIGAN’s discriminator was able to easily beat the generator in adversarial training. The results demonstrated that AIGAN was able to incur perturbations in ML classifiers when applied on the CCD-INID-V1 dataset and the CCD-IDSV1 dataset. AIGAN impacted the most against the pretrained CNN models, which was proven to provide high detection rates without the threat of adversarial attack. However, AIGAN’s generator did not match up well with the current discriminator.

V. CONCLUSION

In this research, we seek to assess the effectiveness of adversarial training, in particular using GAN, towards the generic

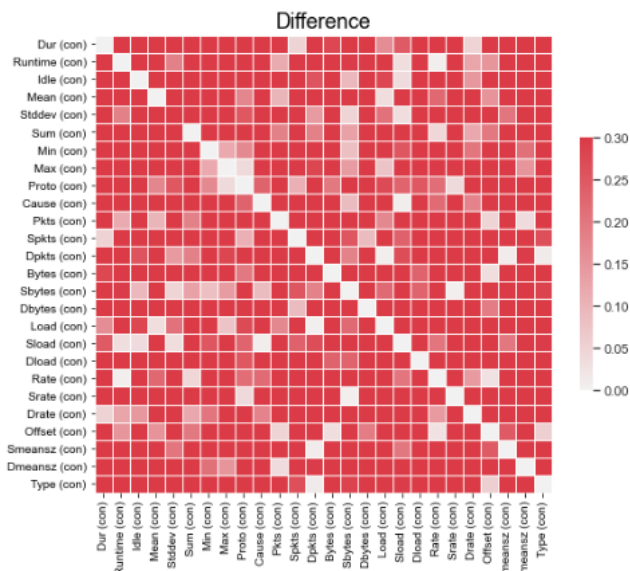


FIGURE 36. Differences between features for the real and fake dataset.

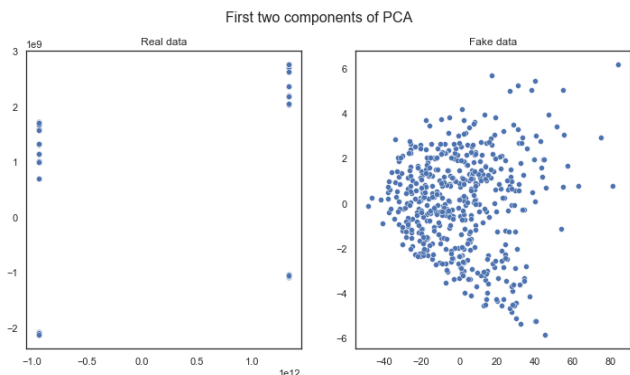


FIGURE 37. PCA between real and fake data.

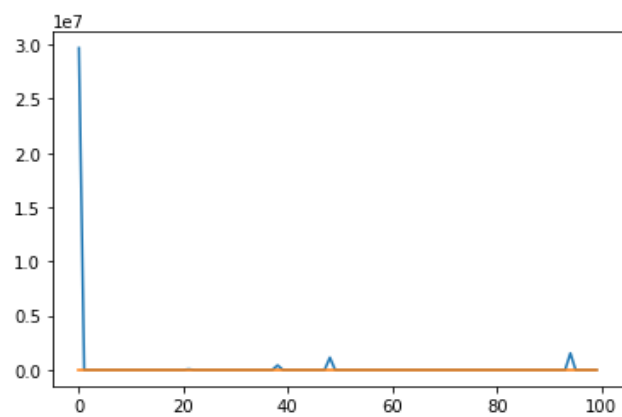


FIGURE 38. Generator (blue) loss and discriminator (orange) loss (100 epochs).

classification models as well as our self-generated models. Our work highlights the vulnerability of ML-based NIDS in the face of adversarial perturbation.

Even though GAN has been used on some publicly available datasets such as NSL-KDD, and USWF-15, we extend the testing on our own datasets. Not only do we want to

see how effective the attack can be, but also to seek insights into possible future directions. We evaluated the effectiveness of proposed methodologies using two benchmark IoT datasets, namely the CCD-INID-V1, and CCD-IDSV1. The experimental results prove the viability of our proposed methodology. The fake network data generated by AIGAN was able to cause perturbations not only in classical generic algorithms but also in our pretrained CNN models. However, our discriminator was able to identify the real and fake data from both datasets.

There are a few limitations in this study. First, the dataset used in this study may have inherent biases, which could impact the performance and accuracy of the proposed method. Awareness of these biases and their potential impact is important. Second, the scalability of the method to larger and more complex IoT networks should be considered. While promising results may be observed on a smaller scale, the performance and efficiency may vary when applied to larger networks. Further investigation is needed to assess scalability.

The challenges associated with the proposed method are notable. Adversarial attacks evolve and become more sophisticated, posing challenges for effectively detecting and mitigating new and unseen attack types. Continuous research and updates to the method are necessary to stay abreast of the evolving threat landscape. Furthermore, generalizing the method’s effectiveness and performance across diverse IoT domains and datasets can be challenging. Adequate testing and evaluation across various scenarios and datasets are crucial to ensure its applicability in different contexts.

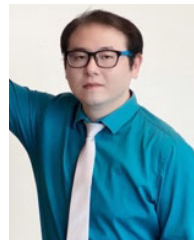
As future work, we plan to explore the effectiveness by looking into further existing methods, such as MAGNeTo, Conditional GAN, Wasserstein distance metric to train the networks (WGAN), particle swarm optimization, to create adversarial malware examples. When applying these techniques, we will consider the challenges faced by current GANs. One challenge is to properly balance generator and discriminator. Discriminator has to be more lenient to allow generator to create better synthetic data. GANs also lack the understanding holistic structure to the entire datasets. Injecting the element of explainable artificial intelligence techniques [84] is a potential solution to strengthen this weakness. Similar to how information is used to explain image classifications, we can explore this to identify an attack signature by putting the traffic characteristics that are most relevant for each attack family in the spotlight. To this end, we plan to extend the current investigation to signature-based classifications.

REFERENCES

- [1] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, “Industry 4.0 and industry 5.0—Inception, conception and perception,” *J. Manuf. Syst.*, vol. 61, pp. 530–535, Oct. 2021.
- [2] R. Soler-Costa, P. Lafarga-Ostáriz, M. Mauri-Medrano, and A.-J. Moreno-Guerrero, “Netiquette: Ethic, education, and behavior on internet—A systematic literature review,” *Int. J. Environ. Res. Public Health*, vol. 18, no. 3, p. 1212, Jan. 2021.
- [3] H. Gangadharbatla, “The role of AI attribution knowledge in the evaluation of artwork,” *Empirical Stud. Arts*, vol. 40, no. 2, pp. 125–142, Jul. 2022.

- [4] M. Stylianos, "Metaverse," *Encyclopedia*, vol. 2, no. 1, pp. 486–497, 2022.
- [5] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, and X. Shen, "A survey on metaverse: Fundamentals, security, and privacy," 2022, *arXiv:2203.02662*.
- [6] E. Wood, T. Baltrušaitis, C. Hewitt, S. Dziadzio, T. J. Cashman, and J. Shotton, "Fake it till you make it: Face analysis in the wild using synthetic data alone," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 3661–3671.
- [7] X. Zheng, Y. Liu, D. Gunceler, and D. Willett, "Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end ASR systems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 5674–5678.
- [8] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.
- [9] O. Ibitoye, R. Abou-Khamis, M. E. Shehaby, A. Matrawy, and M. O. Shafiq, "The threat of adversarial attacks on machine learning in network security—A survey," 2019, *arXiv:1911.02621*.
- [10] N. Martins, J. M. Cruz, T. Cruz, and P. H. Abreu, "Adversarial machine learning applied to intrusion and malware scenarios: A systematic review," *IEEE Access*, vol. 8, pp. 35403–35419, 2020.
- [11] B. Xi, "Adversarial machine learning for cybersecurity and computer vision: Current developments and challenges," *WIREs Comput. Statist.*, vol. 12, no. 5, p. e1511, Sep. 2020.
- [12] A. Oprea, "Machine learning integrity and privacy in adversarial environments," in *Proc. 26th ACM Symp. Access Control Models Technol.*, Jun. 2021, pp. 1–2.
- [13] A. Minagi, H. Hirano, and K. Takemoto, "Natural images allow universal adversarial attacks on medical image classification using deep neural networks with transfer learning," *J. Imag.*, vol. 8, no. 2, p. 38, Feb. 2022.
- [14] Z. Liu, N. Thapa, A. Shaver, K. Roy, X. Yuan, and S. Khorsandroo, "Anomaly detection on IoT network intrusion using machine learning," in *Proc. Int. Conf. Artif. Intell., Big Data, Comput. Data Commun. Syst. (icABCD)*, Aug. 2020, pp. 1–5.
- [15] Z. Liu, N. Thapa, A. Shaver, K. Roy, M. Siddula, X. Yuan, and A. Yu, "Using embedded feature selection and CNN for classification on CCD-INID-V1—A new IoT dataset," *Sensors*, vol. 21, no. 14, p. 4834, Jul. 2021.
- [16] A. Shaver, Z. Liu, N. Thapa, K. Roy, B. Gokaraju, and X. Yuan, "Anomaly based intrusion detection for IoT with machine learning," in *Proc. IEEE Appl. Imag. Pattern Recognit. Workshop (AIPR)*, Oct. 2020, pp. 1–6.
- [17] N. Thapa, Z. Liu, A. Shaver, A. Esterline, B. Gokaraju, and K. Roy, "Secure cyber defense: An analysis of network intrusion-based dataset CCD-IDSv1 with machine learning and deep learning models," *Electronics*, vol. 10, no. 15, p. 1747, Jul. 2021.
- [18] N. Thapa, Z. Liu, D. B. Kc, B. Gokaraju, and K. Roy, "Comparison of machine learning and deep learning models for network intrusion detection systems," *Future Internet*, vol. 12, no. 10, p. 167, Sep. 2020.
- [19] D. J. Gunn, Z. Liu, R. Dave, X. Yuan, and K. Roy, "Touch-based active cloud authentication using traditional machine learning and LSTM on a distributed tensorflow framework," *Int. J. Comput. Intell. Appl.*, vol. 18, no. 4, Dec. 2019, Art. no. 1950022.
- [20] E. Nowroozi, Y. Mekdad, M. H. Berenjestanaki, M. Conti, and A. E. Fergougui, "Demystifying the transferability of adversarial attacks in computer networks," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 3, pp. 3387–3400, Sep. 2022.
- [21] Y. E. Sagduyu, Y. Shi, and T. Erpek, "IoT network security from the perspective of adversarial deep learning," in *Proc. 16th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2019, pp. 1–9.
- [22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–12.
- [23] X. Hu, A. G. Chung, P. Fieguth, F. Khalvati, M. A. Haider, and A. Wong, "ProstateGAN: Mitigating data bias via prostate diffusion imaging synthesis with generative adversarial networks," 2018, *arXiv:1811.05817*.
- [24] A. Abusitta, E. Aïmeur, and O. A. Wahab, "Generative adversarial networks for mitigating biases in machine learning systems," 2019, *arXiv:1905.09972*.
- [25] A. Abusitta, O. A. Wahab, and B. C. M. Fung, "VirtualGAN: Reducing mode collapse in generative adversarial networks using virtual mapping," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–6.
- [26] C. Zhang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna, "Generative adversarial network for synthetic time series data generation in smart grids," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. for Smart Grids (SmartGridComm)*, Oct. 2018, pp. 1–6.
- [27] E. Tabassi, K. J. Burns, M. Hadjimichael, A. D. Molina-Markham, and J. T. Sexton, "A taxonomy and terminology of adversarial machine learning," in *Proc. NIST IR*, 2019, pp. 1–29.
- [28] H. Zhang, T. Zheng, J. Gao, C. Miao, L. Su, Y. Li, and K. Ren, "Data poisoning attack against knowledge graph embedding," 2019, *arXiv:1904.12052*.
- [29] X. Liu, S. Si, X. Zhu, Y. Li, and C.-J. Hsieh, "A unified framework for data poisoning attack to graph-based semi-supervised learning," 2019, *arXiv:1910.14147*.
- [30] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Srndic, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," 2017, *arXiv:1708.06131*.
- [31] L. Chen, Y. Ye, and T. Bourlai, "Adversarial machine learning in malware detection: Arms race between evasion attack and defense," in *Proc. Eur. Intell. Secur. Informat. Conf. (EISIC)*, Sep. 2017, pp. 99–106.
- [32] X. Zhang, C. Fang, and J. Shi, "Thief, beware of what get you there: Towards understanding model extraction attack," 2021, *arXiv:2104.05921*.
- [33] T. Miura, S. Hasegawa, and T. Shibahara, "MEGEX: Data-free model extraction attack against gradient-based explainable AI," 2021, *arXiv:2107.08909*.
- [34] M. Wu, X. Zhang, J. Ding, H. Nguyen, R. Yu, M. Pan, and S. T. Wong, "Evaluation of inference attack models for deep learning on medical data," 2020, *arXiv:2011.00177*.
- [35] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–12.
- [36] J. Shen and M. Xia, "AI data poisoning attack: Manipulating game AI of go," 2020, *arXiv:2007.11820*.
- [37] S. Laatyauoui and M. Saber, "Adversarial attacks on machine learning systems," in *Proc. Int. Conf. Digital Technol. Appl.* Cham, Switzerland: Springer, 2022, pp. 200–208.
- [38] X. Wan, L. Zeng, and M. Sun, "Exploring the vulnerability of deep reinforcement learning-based emergency control for low carbon power systems," in *Proc. 31th Int. Joint Conf. Artif. Intell.*, 2022, pp. 1–13.
- [39] D. Thapar, A. Nigam, and C. Arora, "Merry go round: Rotate a frame and fool a DNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 15034–15043.
- [40] L. Romero and C. Rubio-Medrano, "Defeating adversarial attacks with MTD and genetic algorithm," Tech. Rep., 2022.
- [41] B. M. Garlapati, A. K. Singh, and S. R. Chalamala, "A robust method to protect text classification models against adversarial attacks," in *Proc. Int. FLAIRS Conf.*, vol. 35, May 2022, pp. 1–12.
- [42] C. Z. Mooney, *Monte Carlo Simulation*. Newbury Park, CA, USA: Sage, 1997.
- [43] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann, "Review of classifier combination methods," in *Machine Learning in Document Analysis and Recognition*, 2008, pp. 361–386.
- [44] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proc. Congr. Evol. Comput.*, 2004, pp. 325–331.
- [45] L. Mescheder, S. Nowozin, and A. Geiger, "Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2017, pp. 2391–2400.
- [46] E. Alhajjar, P. Maxwell, and N. Bastian, "Adversarial machine learning in network intrusion detection systems," *Expert Syst. Appl.*, vol. 186, Dec. 2021, Art. no. 115782.
- [47] H. A. Alatwi and A. Aldweesh, "Adversarial black-box attacks against network intrusion detection systems: A survey," in *Proc. IEEE World AI IoT Congr. (AIoT)*, May 2021, pp. 0034–0040.
- [48] I. Idriissi, M. Azizi, and O. Moussaoui, "An unsupervised generative adversarial network based-host intrusion detection system for Internet of Things devices," *Indonesian J. Elect. Eng. Comput. Sci.*, vol. 25, no. 2, pp. 1140–1150, 2022.
- [49] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth, "F-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks," *Med. Image Anal.*, vol. 54, pp. 30–44, May 2019.

- [50] A. Geiger, D. Liu, S. Alnegheimish, A. Cuesta-Infante, and K. Veeramachaneni, "TadGAN: Time series anomaly detection using generative adversarial networks," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2020, pp. 33–43.
- [51] C. Zhang and R. Zuo, "Recognition of multivariate geochemical anomalies associated with mineralization using an improved generative adversarial network," *Ore Geol. Rev.*, vol. 136, Sep. 2021, Art. no. 104264.
- [52] J. Yu, Y. Song, D. Tang, D. Han, and J. Dai, "Telemetry data-based spacecraft anomaly detection with spatial-temporal generative adversarial networks," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–9, 2021.
- [53] S. Ger, Y. Subramanian Jambunath, and D. Klabjan, "Autoencoders and generative adversarial networks for imbalanced sequence classification," 2019, *arXiv:1901.02514*.
- [54] Y. Yoo, U. J. Jung, Y. H. Han, and J. Lee, "Data augmentation-based prediction of system level performance under model and parameter uncertainties: Role of designable generative adversarial networks (DGAN)," *Rel. Eng. Syst. Saf.*, vol. 206, Jan. 2021, Art. no. 107316.
- [55] F. M. M. Mokbal, D. Wang, X. Wang, and L. Fu, "Data augmentation-based conditional Wasserstein generative adversarial network-gradient penalty for XSS attack detection system," *PeerJ Comput. Sci.*, vol. 6, p. e328, Jan. 2020.
- [56] A. Shilandari, H. Marvi, H. Khosravi, and W. Wang, "Speech emotion recognition using data augmentation method by cycle-generative adversarial networks," *Signal, Image Video Process.*, vol. 16, no. 7, pp. 1–8, 2022.
- [57] H. Ohno, "Training data augmentation: An empirical study using generative adversarial net-based approach with normalizing flow models for materials informatics," *Appl. Soft Comput.*, vol. 86, Jan. 2020, Art. no. 105932.
- [58] J. Chen, M. E. Mowlaei, and X. Shi, "Population-scale genomic data augmentation based on conditional generative adversarial networks," in *Proc. 11th ACM Int. Conf. Bioinf., Comput. Biol. Health Informat.*, Sep. 2020, pp. 1–6.
- [59] A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–20.
- [60] S. Afzal, M. Maqsood, F. Nazir, U. Khan, F. Aadil, K. M. Awan, I. Mehmood, and O.-Y. Song, "A data augmentation-based framework to handle class imbalance problem for Alzheimer's stage detection," *IEEE Access*, vol. 7, pp. 115528–115539, 2019.
- [61] F. H. K. dos Santos Tanaka and C. Aranha, "Data augmentation using GANs," 2019, *arXiv:1904.09135*.
- [62] C. Yang, Q. Wu, H. Li, and Y. Chen, "Generative poisoning attack method against neural networks," 2017, *arXiv:1703.01340*.
- [63] C. Miao, Q. Li, L. Su, M. Huai, W. Jiang, and J. Gao, "Attack under disguise: An intelligent data poisoning attack mechanism in crowdsourcing," in *Proc. World Wide Web Conf. World Wide Web*, 2018, pp. 13–22.
- [64] A. Arora, "A review on application of GANs in cybersecurity domain," *IETE Tech. Rev.*, vol. 39, no. 2, pp. 1–9, 2020.
- [65] H. Zenati, M. Romain, C. S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 727–736.
- [66] G. Andresini, A. Appice, L. De Rose, and D. Malerba, "GAN augmentation to deal with imbalance in imaging-based intrusion detection," *Future Gener. Comput. Syst.*, vol. 123, pp. 108–127, Oct. 2021.
- [67] J. Lee and K. Park, "GAN-based imbalanced data intrusion detection system," *Pers. Ubiquitous Comput.*, vol. 25, no. 1, pp. 121–128, 2021.
- [68] H. Zhang, X. Yu, P. Ren, C. Luo, and G. Min, "Deep adversarial learning in intrusion detection: A data augmentation enhanced framework," 2019, *arXiv:1901.07949*.
- [69] S. Shin, I. Lee, and C. Choi, "Anomaly dataset augmentation using the sequence generative models," in *Proc. 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2019, pp. 1143–1148.
- [70] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia, X. Xing, X. Liu, and C. L. Giles, "Adversary resistant deep neural networks with an application to malware detection," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1145–1153.
- [71] C. C. Aggarwal, *Neural Networks and Deep Learning*. Cham, Switzerland: Springer, 2018.
- [72] A. Piplai, S. S. L. Chukkappalli, and A. Joshi, "NAttack! Adversarial attacks to bypass a GAN based classifier trained to detect network intrusion," in *Proc. IEEE 6th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2020, pp. 49–54.
- [73] T. Muncsan and A. Kiss, "Transferability of fast gradient sign method," in *Proc. SAI Intell. Syst. Conf.* Cham, Switzerland: Springer, Sep. 2020, pp. 23–34.
- [74] A. Janusz, D. Kaluza, A. Chadzyska-Krasowska, B. Konarski, J. Holland, and D. Slezak, "IEEE BigData 2019 cup: Suspicious network event recognition," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 5881–5887.
- [75] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, *arXiv:1702.05983*.
- [76] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proc. Netw. Distrib. Syst. Symp.*, vol. 10, Feb. 2016, pp. 1–20.
- [77] M. Alzantot, Y. Sharma, S. Chakraborty, H. Zhang, C.-J. Hsieh, and M. B. Srivastava, "GenAttack: Practical black-box attacks with gradient-free optimization," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2019, pp. 1111–1119.
- [78] R. Mosli, M. Wright, B. Yuan, and Y. Pan, "They might NOT be giants: Crafting black-box adversarial examples with fewer queries using particle swarm optimization," 2019, *arXiv:1909.07490*.
- [79] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, May 1995, pp. 1942–1948.
- [80] I. Aljarah and S. A. Ludwig, "MapReduce intrusion detection system based on a particle swarm optimization clustering algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 955–962.
- [81] S. Devine and N. Bastian, "An adversarial training based machine learning approach to malware classification under adversarial conditions," in *Proc. HICSS*, 2021, pp. 1–10.
- [82] *NFStream*. Accessed: May 2022. [Online]. Available: <https://www.nfstream.org/>
- [83] B. Brenninkmeijer, A. de Vries, E. Marchiori, and Y. Hille, "On the generation and evaluation of tabular data using GANs," Doctoral dissertation, Radboud Univ., Nijmegen, The Netherlands, 2019.
- [84] R. Hughes, C. Edmond, L. Wells, M. Glencross, L. Zhu, and T. Bednarz, "EXplainable AI (XAI): An introduction to the XAI landscape with practical examples," in *Proc. SIGGRAPH Asia Courses*, Nov. 2020, pp. 1–62.



**ZHIPENG LIU** (Member, IEEE) is currently a Senior AI and ML Engineer with Fidelity Investments. He has published several peer-reviewed journals. His has expertise in human-in-the-loop and human-on-the-loop AutoML frameworks, cybersecurity, machine learning, deep learning, and reinforcement learning.



**JUNYI HU** (Member, IEEE) is currently pursuing the Ph.D. degree with the Department of Computer Science, University of Massachusetts (UMASS) Lowell. He is a Senior Software Engineer with Splunk. He started the professional career in the field of information technology, involved in various industries including banking and IT companies. He is a strong advocate for the practical application of cutting-edge research in solving real-world problems and am committed to making a positive impact in the world through the work. His research interest includes natural language processing (NLP) which included text mining, document engineering, data modeling, and related application engineering.





**YANG LIU** (Member, IEEE) is currently pursuing the Ph.D. degree with the Department of Computer Science, North Carolina Agricultural and Technical State University. He is a Graduate Research Assistant and a Teaching Assistant. His research interests include natural language processing (NLP), social media mining, public health surveillance, artificial intelligence and machine learning, and quantum computing.



**KAUSHIK ROY** (Senior Member, IEEE) is currently a Professor and the Interim Chair with the Department of Computer Science, North Carolina Agricultural and Technical State University (NC A&T), and the Director of the Center for Cyber Defense (CCD). He also directs the Cyber Defense and AI Laboratory. His research is funded by the National Science Foundation (NSF), the Department of Defense (DoD), and the Department of Energy (DoE). He has more than 150 publications, including 40 journal articles and a book. His research interests include cybersecurity, cyber identity, biometrics, machine learning (deep learning), data science, cyber-physical systems, and big data analytics.



**XIAOHONG YUAN** (Senior Member, IEEE) is currently a Professor with the Department of Computer Science, North Carolina Agricultural and Technical State University (NC A&T). Her research has been funded by the National Security Agency, the National Centers of Academic Excellence in Cybersecurity (NCAE-C), the National Science Foundation, the Department of Energy, and the Department of Education. Her research interests include AI and machine learning, anomaly detection, software security, cyber identity, and cyber security education. She has served on the editorial board for several journals on cybersecurity.



**JINSHENG XU** (Senior Member, IEEE) received the B.S. degree from Nanjing University, China, the M.S. degree from Peking University, China, and the Ph.D. degree from Michigan State University. He is currently an Associate Professor with the Department of Computer Science, North Carolina Agricultural and Technical State University (NC A&T). He is participated in numerous funded projects in cyber security research and education. He teaches Python for data science, network security, data structures, advanced algorithms, and other courses. He has published many peer-reviewed research papers on the topics of cyber security. His research interests include network security, machine learning, and modeling and simulation.

...