This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3234806

Adversarial Autoencoder Data Synthesis for Enhancing Machine Learning-based Phishing Detection Algorithms

Hossein Shirazi
Management Information Systems
San Diego State University
San Diego, CA, USA
hshirazi@sdsu.edu

Shashika R. Muramudalige
Department of Electrical & Computer Engineering
Colorado State University
Fort Collins, CO, USA
Shashika.Muramudalige@colostate.edu

Indrakshi Ray
Department of Computer Science
Colorado State University
Fort Collins, CO, USA
iray@colostate.edu

Anura P. Jayasumana
Department of Electrical & Computer Engineering
Colorado State University
Fort Collins, CO, USA
Anura.Jayasumana@colostate.edu

Haonan Wang
Department of Statistics
Colorado State University
Fort Collins, CO, USA
Haonan.Wang@colostate.edu

Supervised machine learning is often used to detect phishing websites. However, the scarcity of phishing data for training purposes limits the classifier's performance. Further, machine learning algorithms are prone to adversarial attacks: small perturbations on attack data can bypass the classifier. These problems make machine learning less effective for phishing detection. We propose two Generative Adversarial Network (GAN) based approaches that synthesize phishing and legitimate samples to mimic real-world websites. Information about real-world datasets is obtained from ten publicly available phishing datasets which are used by the AAE (Adversarial Autoencoder) and WGAN (Wasserstein GAN) for generating synthetic data. Using both real and synthesized data, we demonstrate how to implement classifiers with higher performance and more resistance to adversarial attacks. We propose a set of hypotheses and validate them through experiments to demonstrate: (i) indistinguishability of synthesized samples from actual ones, (ii) susceptibility of classifiers to adversarial attacks, (iii) mitigating adversarial attacks by training on larger datasets that include correctly labeled synthesized samples, and (iv) better performance of classifiers trained on large datasets. Our AAE and WGAN have been trained on a wide range of datasets, making us optimistic about its widespread applicability.

***Index Terms*—Phishing Detection, Adversarial Attacks, Adversarial Auto-encoder**

I. Introduction

Phishing attacks, even with sophisticated detection algorithms, still dominate the cyber-crime landscape. FBI's Internet Crime Complaint Center (IC3) reports phishing (including its various forms such as vishing, smishing, and pharming) to be the most prevalent crime type by number in 2019, with an estimated 12.5 billion USD in financial losses worldwide between 2013-2018 [1, 2]. Adversaries learn from their previous attempts to (i) improve attacks and lure more victims and (ii) bypass existing detecting algorithms to obtain sensitive users' information [3, 4] for nefarious purposes.

Social engineering attacks in general, and phishing attacks specifically, are problematic not because of the vulnerability in systems but due to the misjudgment of humans in distinguishing legitimate entities from fake ones. Consequently, several counter-measures have been studied in the literature, differing with respect to the methodologies and the types of attacks they protect against. Machine learning algorithms have shown promising results [5, 6, 7, 8]. Machine learning requires large volumes of labeled data for training the classifiers [9], which are then deployed for detecting phishing websites. Issues related to *unavailability* of data in the phishing context are well-known [10]. Machine learning algorithms are also prone to attacks, such as techniques devised by attackers for bypassing the classifiers. Our preliminary works on applying machine learning for phishing detection have demonstrated the following challenges.

Data Gathering. Complexities involved in gathering attack data and reluctance of parties owning datasets to share them due to concerns such as privacy, confidentiality, and liability [11] are barriers that have prevented high volume phishing datasets from becoming available. There exists repositories that collect links of phishing websites; examples include `PhishTank.com` and `OpenPhish.com`. However, such websites only provide a list of links. Creating a labeled phishing dataset involves accessing the links, visiting the malicious websites, extracting the features, and performing classification. These extra steps are complex tasks requiring expertise.

Data Volume. While the volume of the training dataset is critical for obtaining a high accuracy of the detection model, obtaining such a dataset is not an easy task. The low number of existing phishing datasets [10] does not allow the learning classifier to converge, and we get inconsistent values for accuracy.

Adversarial Attack. Malicious users often attempt to poison the training set or bypass the detection algorithms. In the context of phishing detection, the adversary creates new phishing websites, *e.g.*, by manipulating feature set to bypass the model and evade being caught by the classifiers.

In this paper, we focus on two goals: (i) improving the F1

score of classification algorithms by augmenting the dataset, and (ii) making the detection algorithm robust against adversarial sampling attacks. With respect to the first goal, we propose a deep-learning approach to synthesize new samples that preserve existing data properties. This obviates the need for actual data collection. These samples are added to the training datasets. This is needed when data is unavailable, or the collection process is laborious or infeasible. We use synthesized samples to demonstrate adversarial attacks on the classifier model for the second goal. We can make the classifiers significantly more attack-resistant by injecting labeled synthesized samples into the training set and re-training the models.

A. Our Approach

We advocate the use of two synthesized data generation algorithms, namely, Adversarial Autoencoder (AAE) and Wasserstein Generative Adversarial Network (WGAN), to mimic websites that match the ones generated by actual attackers. We generate both phishing and legitimate samples, which are used to augment real-world datasets.

The use of synthesized samples solves multiple problems. First, it addresses the scarcity of phishing data. Second, constructing a dataset from phishing sites requires visiting the website and processing the data to make it amenable for analysis. This is labor-intensive and requires visiting malicious, often illegal, websites. Samples gathered from such sites are often very limited in size and, therefore, insufficient for machine learning algorithms. The synthesized data from multiple sites can have novel malicious combinations not present in the individual sites. Third, some synthesized samples can be correctly labeled and inserted into the training dataset to make the algorithms more robust against adversarial attacks.

We formulate several hypotheses to demonstrate different aspects of our work. Our first task is to evaluate if a learning algorithm can distinguish synthesized samples from original ones. We next evaluate if the synthesized phishing samples can circumvent the trained model. This demonstrates the propensity of learning models to exploratory attacks [12] by which an attacker perturbs some features to test if the sample will bypass the classifier. Subsequently, we check if injecting a small portion of synthesized samples, labeled correctly, into the training set results in learning algorithms more resistant to exploratory attacks. Finally, we evaluate if using a smaller number of samples negatively affects the learning scores and whether adding synthesized samples (both legitimate and phishing) improves the F1 score of models, even for original datasets not generated by our AAE and WGAN models.

We conducted experiments on ten phishing datasets [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24] and seven classification algorithms namely *Decision Tree* (DT), *Gradient Boosting* (GB), *k-Nearest Neighbors* (KNN), *Random Forest* (RF), *Support Vector Machine* with two kernels: *Linear* (SVM(l)) and *Gaussian* (SVM(r)) kernel, and a *Deep Neural Networks* (DNN). Results demonstrate that our proposed deep-learning generative approaches proved all of the defined hypotheses.

B. Key Contributions

Key contributions of this paper are given below.

- We develop two generative models of AAE and WGAN that can synthesize phishing and legitimate websites that mimic original ones to augment the training dataset. AAE outperforms WGAN in different ways.
- We exemplify the widespread applicability of our approach on ten publicly available phishing datasets and seven different classification algorithms.
- We define a measure to evaluate the closeness of original measured data to synthetic data generated by proposed algorithms. We show that synthesized data are less likely to be distinguished from original data.
- We quantify the improvement of the F1 score of models by using synthesized data. We demonstrate that adding labeled synthesized samples to the training increases the performance of classification algorithms.
- We discuss how to design robust classifiers using synthetic data that can withstand adversarial attacks.

The rest of the paper is organized as follows. In Section II, we describe related work on machine learning based phishing detection and provide some background on generative networks. In Section III, we model the attacker and describe how to produce synthesized samples using two different generative networks. In Section IV, we create the experimental configuration and discuss our results. In Section V, we conclude the paper and mention some future work.

II. Related work

Machine learning algorithms are well-suited for detecting whether a given website is phishing. Earlier machine learning approaches [5, 6, 7, 8, 25, 26] used features from diverse perspectives using public datasets or their own ones. The models were trained on phishing and legitimate datasets to predict whether unknown instances are genuine or phishing.

A. Phishing Detection By Machine Learning

Content-based features. Zhou *et al.* [27] extracted 154 features based on the content of a webpage using four time-based, two search-based, and 11 heuristic features to create a labeled dataset. They created a balanced dataset with 8180 instances. Zhou *et al.* concluded that Random Tree was the best classifier, achieving a precision of 99.4% and 0.1% false positive rate.

Niakanlahiji *et al.* [5] introduced PhishMon, that uses features derived from HTTP responses, SSL certificates, HTML documents, and JavaScript files. It does not rely on third-party services to extract features, it is language agnostic, and detects phishing instances in real-time. The authors reported accuracy of 95% on their datasets.

Subasi *et al.* [28] used one of the existing datasets that we also have used in our experiments to compare performance of Adaboost algorithm and multi boosting on phishing detection. They demonstrated that Adaboost outperformed Multiboosting, achieving accuracy scores as high as 97.61%.

Visual similarity. Mao *et al.* [7] studied visual similarity of phishing and legitimate websites by automatically comparing

the respective Cascading Style Sheets (CSS). The authors proposed a learning-based aggregation analysis mechanism for distinguish phishing websites from legitimate ones.

Shirazi *et al.* [29] defined fingerprint of a legitimate website. The fingerprint uniquely represents a legitimate website by considering both its visual and textual characteristics. Machine learning techniques were used to detect the similarity of suspicious websites with fingerprints of genuine websites. The approach fingerprinted 14 legitimate websites and tested against 1446 unique phishing samples. Authors reported an accuracy of at least 98% for all legitimate websites. The goal of this work was detecting whether a given website is being targeted by phishers. The goal of our current paper, like most other works on phishing detection, tries to check whether a given website is phishing or genuine.

URL-based detection. Phishing instance detection by analyzing the URL of phishing websites have also been proposed. Hong *et al.* [25] consider only the URL of the website and collect a handful of lexical features that have been proposed by other researchers and combine them with features obtained from the blacklisted domains. The results show F-1 scores of 84%.

Sahinguz *et al.* [6] used a set of natural language processing based features of URLs of websites. They ran seven different classifiers for detecting phishing websites and achieved a 97.98% accuracy rate. This study is language independent and can detect phishing websites in real-time without needing third-party services.

Al-Ahmadi *et al.* [30] trained a Long Short-Term Memory (LSTM) as Generative Adversarial Network (GAN) to synthesize new URL. They also trained a Convolutional Neural Network (CNN) as a discriminator to decide whether the URLs are phishing or not. These two components are working together to improve the overall performance. Authors reported an accuracy of 97.5%, which is significantly high.

Kamran *et al.* [31] proposed a conditional GAN for synthesizing adversarial examples and also detecting phishing URLs. Authors used a game-theory perspective to understand the rationale for the decision-making processes of the attacker and the defender.

Haynes *et al.* [32] concluded that using deep learning neural networks on URL-based features alone failed to achieve high accuracy in detecting phishing websites. In a separate experiment, the authors used language transformers that represent context-dependent text sequences for detecting phishing websites. These transformers were able to learn directly from the text in URLs and were able to distinguish between legitimate and malicious websites without feature definition and extraction. Transformer-based approaches outperformed other approaches and achieved accuracy of more than 95% in all cases.

Hybrid approaches. Jain *et al.* [8] extracted 19 different features from the URL and the source code of websites to distinguish phishing websites from legitimate ones. The features are extracted from the client-side and do not rely on third-party services. They achieved a 99.39% true positive rate and the overall accuracy was 99.09%.

B. Data Generation Approaches

Dataset quality and quantity also effect the performance of machine learning algorithms. Shirazi *et al.* [13] observed that datasets used by researchers are often biased with respect to the features based on the URL or content. Moreover, some of the features become obsolete with time or as new attacks emerge. Sometimes the authors extract features for the first page of legitimate websites, but not the other pages. Machine learning algorithms must be trained on enough data samples, but there is not a simple way to estimate the needed dataset size. The right size often depends on the complexity of the problem and that of the learning algorithm and falls under the sample size determination problem.

Figueroa *et al.* described a sample size prediction algorithm that conducted weighted fitting of learning curves in an active learning algorithm [33]. Active learning systems attempt to minimize the number of required labeled data and maximize the accuracy of the model by asking queries in the form of unlabeled instances to be marked by another agent such as the domain expert [34].

Small datasets often create inaccurate learning models, so the right size dataset is critically important. Data gathering and labeling are challenging and often times expensive operations. Since getting enough attack data may be infeasible, many data augmentation techniques have been proposed and used in the literature [35, 36, 37, 38]. However, our approach focuses exclusively on phishing samples, and has been tested on a large set of datasets for evaluation.

Shirazi *et al.* [35] used an adversarial algorithm to generate new synthesized samples for increasing the dataset size. This work further showed how these synthesized phishing samples can evade the classifier. The authors used a heuristic algorithm for feature manipulation in order to generate samples. Our current paper extends the AAE network so that it is capable of generating more sophisticated samples with a well-studied algorithm which ensures that the sample matches real-world data. In addition, we also demonstrate the use of WGAN for generating synthetic samples in this work. Our previous work [35] also does not provide any solution that protects against exploratory attacks. Our current work demonstrates how to train the model to make it resilient to exploratory attacks.

Other domains including social analytics [39, 40, 41], privacy [42], health informatics [43], video traffic classification [44] also face the issue of limited data availability and data incompleteness. In many cases, data collection and maintenance require effort and poses challenges due to data privacy, confidentiality, and liability issues. Behavioral and social network data are inherently sparse and incomplete because sometimes the behavioral indicators are not shown or recorded [45]. Muramudalige *et al.* [46] showed an adversarial data generation technique with novel feature mapping techniques to synthesize sparse, incomplete, and small datasets while mapping into complex objects. The proposed method was validated via three real-world datasets, which were small and incomplete.

Adversarial deep-learning approaches for data generation have been used in several domains due to their accuracy and

efficacy. Goodfellow *et al.* [47] presented the use of GAN for data generation without requiring comprehensive problem-specific theoretical basis or empirical verification [48]. The first such GAN architecture [47] is only capable of capturing the precise distribution of continuous and complete data but cannot be used for learning the distribution of discrete variables [49].

Since there is a critical need to capture data distribution with discrete features for diverse application domains such as phishing, medical, crime data, *etc.*, Makhzani *et al.* [50] proposed the AAE, which is a probabilistic autoencoder that uses the GAN framework as a variational inference algorithm for both discrete and continuous latent variables. Choi *et al.* [49] focused on learning the distribution of discrete features, such as diagnosis or medication codes, using a combination of an autoencoder and the adversarial framework. Wasserstein GAN (WGAN) [51] is another well-known technique used in various domains for both continuous and discrete distributions such as image generations [52] and Internet traffic generation [53]. WGAN has a unique loss function (Earth-Mover (EM) distance or Wasserstein-1) to calculate the difference between actual and generated data distributions. Kattadige *et al.* [44] applied the same feature mapping techniques proposed in [46] with WGAN to synthesize video traffic data for more accurate video types classification. In this work, we use AAE and WGAN to generate more realistic phishing and legitimate website samples.

Chen *et al.* [54] presented an attack-agnostic defense mechanism for detecting poisoning attacks, which means it is not designed to detect specific types of attack. In this work, authors proposed two novel designs. First, a synthetic data generation that uses conditional GAN (cGAN). In the next step, a WGAN is set up to learn the distribution present in the predictions related to the synthesized data. By defining a detection boundary, attack samples can be distinguished from original samples.

III. Our Approach

In this section, we first present our threat model. We then discuss the synthetic data generation techniques using AAE and WGAN which we use to produce phishing and legitimate samples. We then discuss our experimental methodology. Finally, we briefly explain the ML classifiers that we used in our experiments.

A. Threat Model

We define the threat model by stating our assumptions in terms of *goal*, *knowledge*, and *influence* of an attacker [35].

Attacker's Goal. We assume an attacker will aim to attack the integrity of the system by making the system label a phishing instance as legitimate.

Attacker's Knowledge. We assume an attacker only knows about the features of the phishing instances but not the learning model parameters. This is a realistic assumption as an attacker may have access to the definition of existing datasets but not the specific implementation of a classifier. The attacker does not have any information about other system parameters like the algorithms that have been used, dataset instances, or learning parameters.

Shirazi *et al.* [35] demonstrated the vulnerabilities of learning models against adversarial sampling attacks using a feature manipulation approach. However, in this current work, we focus on evaluating adversarial samples and their effect on phishing detection, similar to that of [35], and also address the problem of inadequate volume of phishing dataset.

Attacker's Influence. Ling *et al.* [55] discuss two types of attacks: (a) *Causative Attacks* and (b) *Exploratory Attacks*. In *Causative Attacks*, the attacker mislabels a portion or the entire training data to affect the algorithm. In other words, the attacker poisons the training data. In *Exploratory Attacks*, the attacker crafts samples so as to evade the classifier without direct influence. In this study, we assume the adversary carries out exploratory attacks and targets the integrity of the system; he cannot inject adversarial samples into the training set to carry out causative attacks.

B. Adversarial Autoencoder (AAE) for Synthesized Data Generation

We utilize the AAE for synthesizing both phishing and legitimate samples. Since AAE can generate both continuous and discrete data distributions, it is very suitable for generating discrete feature sets of datasets described in Subsection IV-A. The high-level architecture of the AAE is depicted in Figure 1. The autoencoder derives a compressed knowledge representation of the original input, which reconstructs the same data distribution.

$$q(z) = \int_x q(z|x)p_d(x)dx \tag{1}$$

An aggregated posterior distribution of $q(z)$ on the latent code is defined with the encoding function $q(z|x)$ and the data distribution $p_d(x)$ as shown in Eq. 1 where x denotes real phishing dataset. In this work, we synthesize phishing and legitimate samples separately, where we train two different AAEs for each dataset.

The operating principle of AAE is that the autoencoder seeks to minimize the reconstruction error while the adversarial network attempts to minimize the adversarial cost. *Reconstruction phase* and *regularization phase* are two simultaneous phases that arise during training. In the reconstruction phase, the autoencoder's data reconstruction error, often referred to as the loss, is minimized. The regularization phase relates to the adversarial component of the network. It minimizes the adversarial cost to fool the discriminator by maximally regularizing an aggregated posterior distribution $q(z)$ to the prior $p(z)$ distribution.

The simultaneous training process leads the discriminative adversarial network into believing that the samples from hidden code $q(z)$ come from the prior distribution $p(z)$ [50]. A normal distribution is exploited as the arbitrary previous $p(z)$ in this work. After the training process, the adversarial network synthesizes samples similar to the actual samples through the prior distribution $p(z)$.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3234806

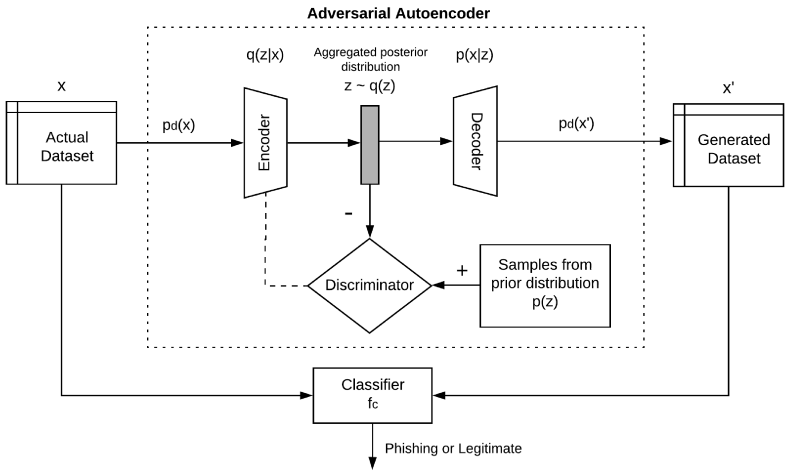


Fig. 1: The high-level architecture of synthetic data generation approach. The adversarial autoencoder (AAE) generates both phishing and legitimate samples. The top row represents the standard autoencoder that reconstructs the data from the latent code z. The next row shows the discriminator network that predicts whether the samples emerge from the hidden code of the autoencoder $q(z)$ or the user-defined prior distribution $p(z)$ [50]. $p_d(x)$ denotes the data distribution. $q(z|x)$ and $p(x|z)$ denote the encoding and decoding distributions respectively. After the data generation, a machine learning classifier (f_c) described in Subsection III-E is applied for different classification tasks.

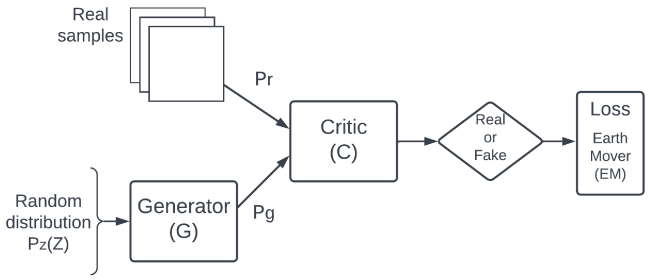


Fig. 2: The high-level architecture of Wasserstein Generative Adversarial Network (WGAN)

The synthesized dataset has the characteristics of phishing samples generated by real-world attackers and the characteristics of legitimate samples collected from real websites. We feed them into a classification algorithm that can distinguish phishing samples from legitimate ones. This classifier is unaware of whether the samples are synthesized or actual. The instances are labeled as legitimate or phishing, and the classifier will predict them subsequently.

C. Wasserstein Generative Adversarial Network (WGAN)

Similar to AAE, we use the WGAN for synthesizing both phishing and legitimate samples. Since WGAN is fluent in generating both continuous and discrete data distributions, it is very suitable for generating discrete feature sets of datasets described in Subsection IV-A. The high-level architecture of the WGAN is depicted in Figure 2.

Let $\mathbb{P}_r$ and $\mathbb{P}_g$ be the actual and generated distributions respectively. Typically in GAN architecture, instead of evaluating the density of the distribution ($\mathbb{P}_r$), we can define a random variable Z with a noise (known) distribution $p_z(z)$ and send it through a parametric function $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$ that is capable of synthesizing samples from a certain distribution $\mathbb{P}_\theta$ [51]. To achieve the objective in WGAN, two deep-neural networks, i.e., generator (G) and discriminator compete each other in the training phase. In WGAN, the discriminator is called the critic (C). C determines the real and fake samples from $p_z(z)$ and G confuses C by convincing that synthesized samples reach from the real distribution ($\mathbb{P}_r$). Eventually, the G will be capable of generating data samples that are similar to real samples by mapping its distribution ($\mathbb{P}_g$) to $\mathbb{P}_\theta$. The contest between G and C is a two-player minimax game with value function $V(C, G)$ [47]:

$$\min_G \max_C V(C,G) = \mathbb{E}_{x\sim p_r(x)}[logC(x)] + \mathbb{E}_{z\sim p_z(z)}[log(1 - C(G(z)))] \quad (2)$$

Further, WGAN has a distinct loss function to compute the difference between actual ($\mathbb{P}_r$) and generated ($\mathbb{P}_g$) data distributions compared to the ordinary GAN. WGAN uses the Earth-Mover (EM) distance or Wasserstein-1 as the loss function [51] while GAN calculates the loss via the standard cross-entropy [47]. Earth-Mover (EM) distance can be defined as follows.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \prod(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \quad (3)$$

where $\mathbb{P}_r$, $\mathbb{P}_g$ is the set of all joint distributions $\gamma(x, y)$, whose marginal distributions are $\mathbb{P}_r$ and $\mathbb{P}_g$. In other words, the similarity between actual and generated data is calculated by finding the infimum of the expected values of distances between data points from the distributions of actual and generated data [53].

Further, training of WGANs does not require maintaining a careful balance of the G and the C, and also does not require a cautious design of the network architecture. The Wasserstein

loss function is also capable of providing a continuous and usable gradient compared to many other loss functions [51]. Therefore, WGAN is useful for these types of contexts where only small and discrete training data is available.

In both AAE and WGAN, we train separate generative models for phishing and legitimate samples in each dataset, which enables capturing different underlying trends of phishing and legitimate samples with different sets of distinct features. The feature values are varied in many value ranges. Thus, the values are normalized between -1 and 1 before feeding to the AAE or WGAN and are denormalized after data generation. The synthesized phishing and legitimate samples are subsequently integrated into a single dataset. Now we have two datasets:

1) *Original Dataset* that has both phishing and legitimate samples and was obtained from publicly available datasets. The original dataset is used to generate synthesized samples.
2) *Synthesized Dataset* that consists of new synthesized phishing and legitimate samples generated by the AAE or WGAN.

After we synthesize data, we apply both original and synthesized datasets in our experiments in Section IV.

D. Experimental Methodology

We define five hypotheses based on the goals introduced in Section I. We define five scores, corresponding to the five hypotheses. Each score empirically evaluates the corresponding hypothesis.

Hypothesis-1. The classification algorithm has acceptable performance on the dataset without considering any synthesized sample, *i.e.*, the performance is acceptable on the dataset that contains only original samples. This hypothesis serves to demonstrate that we are using the most appropriate classification algorithm for the datasets. Hypothesis-1 states that the classification algorithms reproduce the accuracy close to that reported by original authors of the respective datasets. In other words, we need to evaluate that our classification algorithms outperform, or are at least as good as, the performance reported by authors of those datasets. For cases where the authors do not report any results, the accuracy needs to be in an acceptable range. We train and test our classifiers without considering synthesized samples and compare results with the authors' results to prove this hypothesis. Δ^1 that evaluates Hypothesis-1 is defined as follows.

Δ^1 is the difference between the accuracy reported by original authors of the dataset and the accuracy we got in our experiments. Positive values or close to zero are desired as it proves that the accuracy of our model is better or close to what the original authors reported.

Hypothesis-2. Synthesized samples are indistinguishable from actual data with regards to the machine learning classification algorithm. Hypothesis-2 demonstrates that a machine learning algorithm will not be able to distinguish synthesized samples from actual ones. For this purpose, a classification algorithm is designed to distinguish synthesized samples from original ones. We construct a dataset including original samples and synthesized samples. Label of samples indicate if a sample is original or synthesized, and we do not care if the samples are phishing or legitimate. We then train a classifier to distinguish original samples from synthesized ones. Δ^2 is the F1 score of this experiment. We denote the best F1 as Δ^2_{Max}. We denote the average F1 over all classifiers as Δ^2_{Avg}. Average F1 on different classifiers is important because a classifier may perform well on few datasets, but do poorly on others.

Hypothesis-3. Synthesized samples, generated by the AAE and WGAN, will evade the classifier and be mislabeled more than original samples. Mislabeling may happen for both phishing and legitimate samples. In other words, synthesized phishing samples will be incorrectly labeled as legitimate more than original phishing samples. Note that, if this is indeed the case, then attackers can create such synthesized samples that will easily evade the classifiers.

We first train a classifier with only the original samples in the dataset. This guarantees that the algorithms do not have information about synthesized samples. We then test classifiers with two sets of samples: once with original dataset and then with synthesized samples generated by each of synthesizer algorithms. The difference of these two sets of results is defined as Δ^3 score.

Δ^3 specifies the difference in F1 score of a model when it is tested on original samples and synthesized samples. The lower values for Δ^3 shows that the F1 of model against synthesized samples is lower than F1 of model against original samples. In other words, it indicates that the pair of classifier and dataset are more vulnerable against synthesized samples. It should be said that this vulnerability is not because of the classifier we have used in this experiment but it is related to the pair of classifier and dataset together. In other words, different classifier or an extended dataset may mitigate this vulnerability.

Hypothesis-4. Re-training classifiers on datasets that have been injected with synthesized samples will improve the F1 score of the models with regards to synthesized samples. This hypothesis considers the mitigation vulnerability. Injecting synthesized samples in training set will improve the F1 score to the level when there was no synthesized samples in the training or testing datasets.

Δ^4 calculates the difference between the F1 of a classifier when it is tested with synthesized samples: once it is trained only with original samples, and once it is trained with both original and synthesized samples. Higher values on Δ^4 are desired.

Hypothesis-5. Augmenting dataset with synthesized samples improves F1 score with respect to the original samples. A useful application of the proposed approach is to increase the size of the training set without the need to gather real data. Δ^5 is defined for this purpose.

Δ^5 defines the improvement of the F1 of the original samples, *i.e.*, we calculate the difference between the F1 of two classifiers when it is tested with only original samples: once it is trained on only original samples and, once it is trained on both original and synthesized samples. This score helps to understand if adding synthesized samples can improve the accuracy of the classifier with regards to the original samples.

Table I summarises hypotheses and scores we defined in this section.

E. Machine Learning Classifier

We consider six statistical classification algorithms available in the Scikit-learn tool [56] to train and evaluate our model. These are *Decision Tree* (DT), *Gradient Boosting* (GB), *k-Nearest Neighbors* (KNN), *Random Forest* (RF), *Gaussian Naive Bayes* (GNB), and *Support Vector Machine* with two kernels: *Linear* (SVM(l)) and *Gaussian* (SVM(r)) kernel. These algorithms are widely used in the literature for phishing detection and achieved promising results. In addition to these statistical algorithms, we have also implemented a deep neural network model to compare the results. This can demonstrate vulnerability of Deep Neural Network (DNN) algorithms to evaluate against poisoning attack in the context of phishing detection. For each experiment, we optimize specific parameters of each classifier to obtain the best results and prevent overfitting. We fine-tuned each algorithm with wide ranges of hyper-parameters to get the best performing model.

For DT, we varied the maximum depth from 2 to 20. For GB algorithm, we checked learning rates from 0.05 to 1 with 20 estimators and a maximum depth of 10. We checked the RF algorithm with different numbers of estimators, varying between 10 to 200 estimators. For KNN algorithm, we varied the number of neighbors parameters from 3 to 25. For SVM with linear kernel, we checked C parameters with the following values: $\{0.001, 0.01, 0.1, 1, 10\}$. For SVM with Gaussian kernel, in addition to C parameter, we checked Gamma parameters with $\{0.01, 0.1, 1\}$ values.

For DNN, we used a network with one input layer, two hidden layers, and one output layer. The first hidden layer includes 32 hidden nodes, and the second hidden layer has 16 hidden nodes and the ReLu activation function. For the output layer, we used the sigmoid activation function. We used dropout regularization with 0.2 rate and binary cross-entropy loss function. We trained each network with 500 epochs and an early stopping method with a patience of 200.

IV. Experiments and Evaluation

We conduct a set of experiments to empirically prove the hypotheses that are defined in Section III. We start with introducing the datasets used, followed by experiments we conducted for each hypothesis and their results.

A. Summary of Phishing Datasets

We use ten phishing datasets publicly available on the Internet. We list the total number of instances for each dataset, and also the number of phishing and legitimate instances. In addition, we explain the number and types of features in each dataset. Types of features are essential as they explain the characteristics of datasets on which our algorithms can be executed. In addition, we mention the highest accuracy reported by the original authors of the dataset, if it was available, for comparison purposes.

DS1: Shirazi *et al.* [13] phishing dataset focuses on a subset of domain-name-based features without requiring third-party services. This dataset includes 7 features with 1K legitimate samples and 1.2K phishing samples; total of 2.2K samples. The reported accuracy varies between 97 to 98 on the validation set and is unknown for live phishing URLs.

DS2: Rami *et al.* [14] dataset was shared through the UCI machine learning repository [57]. Authors detected characteristics that help discern phishing websites from legitimate ones, including long URL, IP address in URL, adding prefixes and suffixes to the domain, and request URL. The authors defined 30 features that can be categorized as follows: *URL-based*, *abnormal-based*, *HTML-based*, *JavaScript-based*, and *domain name-based* features. *abnormal-based* are features extracted based on abnormality on URLs, *e.g.* DNS records did not find in WHOIS database. The authors also analyzed the most significant features of the detection algorithm. The authors reported 92.2 for accuracy on this dataset. We used all 30 features in our experiments, regardless of their relative importance.

DS3: Abdelhamid *et al.* [15] dataset is listed in the UCI machine learning repository [57]. The essential features include HTML content-based features and some that require third-party service inquiries, such as DNS servers that perform domain-name age lookup. The best accuracy reported for this dataset is 97.

DS4: Tan *et al.* [16] published their dataset on Mendeley dataset library. It includes 48 URL-based and HTML-based features. The authors integrated a feature selection phase with a training phase and chose the ten best features with a random forest classifier. We used all 48 features in our experiments. The best accuracy reported is 96.

DS5: Hannousse *et al.* [17] dataset includes more than 11,000 phishing and legitimate URLs with 87 extracted features from three different categories: 56 extracted from the structure and syntax of URLs, 24 extracted from the page contents, and seven are extracted by querying external services. The dataset consists of 50% of phishing and 50% of legitimate instances. The best accuracy reported by the authors is 96.6.

DS6: Vrbancic *et al.* [18, 19] dataset has 111 URL-based features without considering the contents of webpages or using third-party services. This dataset includes more than 146,000 instances of phishing and legitimate websites. The best accuracy reported for this dataset 94.39.

DS7: Moruf *et al.* [20, 21] dataset consists of 7,200 phishing and 5,800 legitimate websites with 35 features. The authors added image identity, page style, layout identity, and text identity features. The authors reported an accuracy of 98.3.

DS8: Mahmodi *et al.* [22, 23] dataset of 8,000 legitimate and phishing instances had 75 URL and content-based features. The authors reported an accuracy of 97.0.

DS9: Muhammad *et al.* [24] dataset of 15,000 instances of phishing and legitimate websites had 79 URL-based features. The best reported accuracy for this dataset is 96.5.

DS10: Marchal *et al.* [58, 59] released a URL-based phishing dataset with more than 96 thousands instances. Authors argued that phishing URLs usually have few relationships between the URL part that must be registered (low-level domain) and the remaining part of the URL (upper-level domain, path, query). They defined the concept of intra-URL relatedness and, to

TABLE I: **Summary of Hypotheses**. For each hypothesis, we defined two experiments (Exp.1 and Exp.2) and calculated difference. Org means only original data was used for training or testing and Syn indicates use of synthesized data. Syn. ∪ Org. indicates use of both original and synthesized data. Target specifies result of classification labels for experiment. Phi vs. Leg indicates classifier distinguished phishing samples from legitimate ones and Syn vs. Org indicates distinguishing synthesized samples from original ones.

#	Hypothesis Description	Training Exp1	Training Exp2	Testing Exp1	Testing Exp2	Target
Hyp-1	Reproducing datasets authors accuracy	Org	Org	Org	Org	Leg vs. Phi
Hyp-2	Distinguishing synthesized and original samples	Syn. ∪ Org.	–	Syn. ∪ Org.	–	Syn vs. Org
Hyp-3	Mislabeling of synthesized samples	Org	Org	Org	Syn	Leg vs. Phi
Hyp-4	Recovery performance for synthesized samples	Org	Syn. ∪ Org.	Syn	Syn	Leg vs. Phi
Hyp-5	Increased performance for original samples	Syn ∪ Org	Org	Org	Org	Leg vs. Phi

TABLE II: **Summary of datasets**. This table summarises datasets we used in our experiments, including **N**ame we used in this paper, released **Y**ear, author's reported **A**ccuracy, dataset instance sizes (number of **L**egitimate, **P**hishing, and **T**otal instances), number of **F**eatures and types of features (**URL**-based, page-content-based (**Pg.**), and inquiring **3rd.** party services).

Dataset			Size (K)			Features			
N	Y	A	L	P	T	F	URL	Pg.	3rd.
DS1	2018	98.0	1	1.2	2.2	7	✓	✓	
DS2	2012	92.2	6.2	4.9	11.1	30	✓	✓	
DS3	2014	97	0.6	0.7	1.3	9		✓	✓
DS4	2018	96	5.0	5.0	10.0	48	✓	✓	
DS5	2020	96.6	5.7	5.7	11.4	87	✓	✓	✓
DS6	2020	-	58.0	30.7	88.7	111	✓		
DS7	2020	98.3	5.9	7.2	13.1	35		✓	
DS8	2020	97.0	2.2	1.8	4.0	76	✓	✓	
DS9	2020	-	7.8	7.6	15.8	79	✓		
DS9	2014	-	48.0	47.9	95.9	12	✓		✓

evaluate it, extracted 12 features from words that compose a URL based on query data from Google and Yahoo search engines. For this dataset, the best-reported accuracy is 96.28.

Table II summarizes the number of instances, features, and the portion of legitimate vs. phishing instances in each dataset. We also specify whether these datasets have URL-based, page-content-based, and third-party-based features.

B. Hyp-1: Reproducing results cited by original authors

We evaluate Hypothesis-1 as it questions whether our proposed method can reproduce the accuracy close to the accuracy reported by the original authors of datasets without considering any synthesized samples. We use 80% of data for training purposes and 20% for testing in five-fold cross-validation. We run all experiments ten times and report the mean accuracy.

Table III summarises the accuracy scores we achieved for all ten datasets and seven classification algorithms. It also expresses reported accuracy by authors. For calculating Δ^1_{Acc}, we selected the maximum accuracy we got among seven classifiers (declared in bold font) and then subtracted the reported accuracy by authors.

Positive values for Δ^1_{Acc} indicate our model outperformed the accuracy of original authors. For 6 out of 10 datasets, we

TABLE III: **Evaluation for Hyp-1.** This table reports the accuracy of all classifiers and datasets when trained to detect phishing samples from legitimate samples without considering synthesized samples. The best performance among different classifiers is in bold. When tested on different datasets, the average performance for each classifier is also reported. It shows reported performance by original authors of each dataset. Also, it shows Δ^1_{Acc} metric or the difference between best reported performance and reported performance by original authors.

	DT	GB	KNN	RF	SVC(l)	SVM(r)	DNN	Auth.	Δ^1_{Acc}
DS1	95.9	96.8	96.2	**97.3**	95	95.2	96.2	98	-0.7
DS2	96.7	97	94.8	**97.3**	93.2	96.3	96.7	92.2	5.1
DS3	91.6	93.6	93.2	92.8	90	**94.0**	90.8	97	-3
DS4	97.6	98.2	87.3	**98.6**	94.6	91.8	97.4	96	2.6
DS5	94.2	95.8	93	**96.5**	95.1	95.3	95.5	96.6	-0.1
DS6	95.6	95.2	88.1	**97.1**	93	94.7	96	94.39	2.71
DS7	99.1	99.1	99	**99.2**	93.9	94.2	99.1	98.3	0.9
DS8	99.1	**99.9**	97.9	99.6	98.8	80.6	99	97	2.9
DS9	97.2	98.3	97	**98.5**	96.3	96.9	97.9	96.5	2
DS10	93.6	92.9	89.4	**95.5**	83.1	87.5	89.6	96.28	-1.33
Avg.	96.06	96.68	93.59	**97.24**	93.3	92.65	95.82	96.44	1.11

are reporting positive Δ^1_{Acc} values. For the four datasets of DS1, DS3, DS5, and DS10 that we report negative values for Δ^1_{Acc}, the results are not statistically significant. In average, the accuracy we are reporting is 1.11% better than the accuracy reported by original authors of this dataset. In addition, the best average accuracy we got among classifiers belonging to RF classifiers is higher than the average accuracy reported by the original authors. These results prove Hypothesis-1.

C. Hyp-2: Distinguishing synthesized samples

We evaluate Hypothesis 2 to see if synthesized samples are distinguishable from original samples. Both AAE and WGAN algorithms generated 10K phishing and 10K legitimate samples for each dataset. We train our set of classifiers on synthesized samples (positive labels) and original samples (negative labels). We test each classifier to evaluate the performance to predict label of each given sample. We use 80% of data for training purposes and 20% for testing in five-fold cross-validation. Although some datasets are imbalanced, we report the best performing F1 scores in Table IV, separately for both AAE and WGAN algorithms.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3234806

DS	DT		GB		KNN		RF		SVC(l)		SVC(r)		DNN		Max	
	A	W	A	W	A	W	A	W	A	W	A	W	A	W	A	W
DS1	56	54.5	54.3	51.1	46.2	39.2	51	48.8	0	0	47	32.8	38.3	29.2	56	54.5
DS2	82.9	84	88.1	88.4	72.5	77.3	90.7	91.5	9.8	9.7	89.9	88.3	82.8	84.8	90.7	91.5
DS3	12.7	20.8	14.2	27	14.4	24.8	15.3	26.6	0	0	13.9	20.5	11	27.2	15.3	27.2
DS4	94.2	92.3	96.2	96	89	92.1	97.5	97.4	54.2	11	93.7	93.8	96.7	97.6	97.5	97.6
DS5	99.9	99.5	99.8	100	74.5	85.2	99.9	100	74.4	20.6	92.9	96.4	96.7	99	99.9	100
DS6	99.3	100	99.1	100	90.3	96.6	99.8	100	87.7	98.4	96.7	99.9	99.6	100	99.8	100
DS7	90.2	95.3	93.1	96.7	85.7	88.5	97.2	98.2	17.1	0	78.7	72.4	91.6	95.3	97.2	98.2
DS8	99.1	98.9	99.2	99.5	73.8	86.3	99.6	99.9	28.4	40.1	75.3	41.4	89.9	97.4	99.6	99.9
DS9	99.9	99.8	100	100	93.3	96	100	100	88.7	90	98.7	92.8	99.5	99.8	100	100
DS10	97.9	99.7	98.9	99.8	91.8	94.1	98.9	99.9	70.4	73.1	93.3	95.3	97.1	99.7	98.9	99.9
Avg.	83.21	84.48	84.29	85.85	73.15	78.01	84.99	86.23	43.07	34.29	78.01	73.36	80.32	83	84.99	86.23

TABLE IV: **F1 score for Hyp-2.** Table reports results of F1 scores for different classifiers for two synthesizer algorithms of **A**AE and **W**GAN for all datasets. It also reports **Max**imum F1 score for each dataset and **Ave**rage for each classifiers.

We fine-tune all classifiers over datasets in this experiment. The lower $\Delta^2_{Max\_F1}$ scores demonstrate the lack of ability to distinguish synthesized samples from legitimate ones and support our Hypothesis-2. Table IV summarises $\Delta^2_{Max\_F1}$ scores as we defined in Section III. For each triple of classifier, dataset, and synthesizer algorithm (AAE or WGAN), we report the F1 score. We also report the best F1 score for each pair of datasets and algorithms and the average F1 score for each pair of classifier and algorithm. As Table IV shows, the best F1 scores for DS1 and DS3, declared by $\Delta^2_{Max\_F1}$, are very low. For other datasets, the $\Delta^2_{Max\_F1}$ are reasonably high, with the lowest values for DS2 are 90.7 for AAE, and 91.5 for WGAN. While Δ^2 values are significant, the average of different classifies over our datasets is very low. The highest average score belongs to the RF classifier for both AAE and WGAN synthesizers, 84.99 and 86.23. In other words, on average, the RF classifier is able to detect synthesized samples, no matter what algorithm was used, better than any other classification algorithm we tested. These results show that while classifiers may successfully discriminate synthesized samples from original ones in some datasets, the average results are low and prove our hypothesis that synthesized samples are difficult to distinguish from original ones on average.

D. Hyp-3: Performance of synthesized samples

To prove Hypothesis-3, we checked the F1 score of classifiers against synthesized samples, with two generators of AAE and WGAN, with models trained exclusively on original samples. This will demonstrate whether synthesized samples can bypass the models and go undetected and samples of what algorithms are more likely to bypass. We calculated Δ^3, as the difference F1 score of classifiers trained exclusively on original samples and tested once on synthesized samples and once only on original samples. For reporting results, we averaged the difference between the F1 scores of models trained on original samples when models were tested once against original samples and once against synthesized samples on different datasets for both algorithms. That number shows how F1 between these two testing sets changes, either increasing or decreasing on average. We have used 2000 synthesized phishing samples and 2000 legitimate samples for testing purposes.

Figure 3 depicts Δ^3 scores for different classifiers, datasets, and algorithms. Different classifiers faced a decrease in Δ^3 at least 5%, and for 3 of those, around 10%. SVM with Gaussian kernel is the worst with 10% and 20% for AAE and WGAN algorithms, respectively. This is a clear sign that all tested algorithms in this experiment are vulnerable to synthesized samples. In addition, AAE was able to synthesize samples that are evading classifiers more than WGAN, a sign that indicates AAE is more successful.

In addition, Figure 3 shows the Δ^3 for different datasets. On average, the Δ^3 score has been decreased by around 5% for all datasets, some datasets up to 15%; a clear sign that synthesized samples can evade the classifier. Among different datasets, DS4, DS5, DS9, and DS10 have more decrease in learning scores.

These results demonstrate that our synthesized phishing and legitimate samples are able to evade trained classifiers, a clear sign of vulnerability for models for both our classification algorithms and datasets. Our experiments used ten public phishing datasets, seven conventional machine learning classifiers, and two synthesizing algorithms. Our experiments were carried out on a wide range of datasets and classification algorithms which demonstrates the problem of evading classifiers.

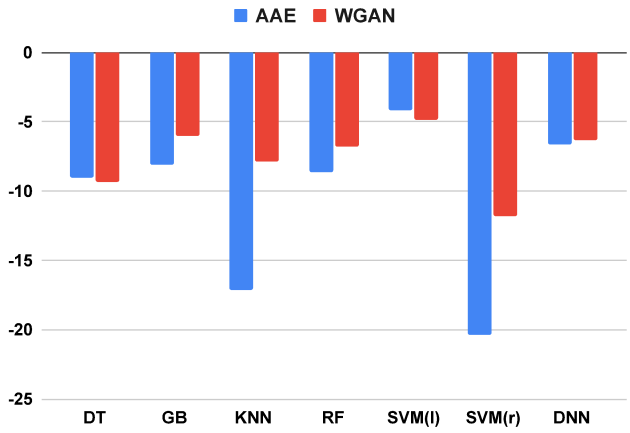
E. Hyp-4: Mitigating against adversarial samples

In order to prove Hypothesis-4, we injected synthesized samples to measure if the F1 score increased. We define Δ^4, which is the difference F1 score between when models were trained only with original samples and when they were trained with both original and synthesized samples. For each dataset, we injected 80% synthesized samples into the training set and reserved 20% for testing. Figure 4 explains the results of experiments for Δ^4 based on a classifier, datasets, and synthesizing algorithms.

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2023.3234806

Fig. 3: Percentage decrease of accuracy and F1 score for models trained exclusively on original samples but tested on original samples over those tested only on original samples.

(a) Classifiers performance



(b) Datasets Performance

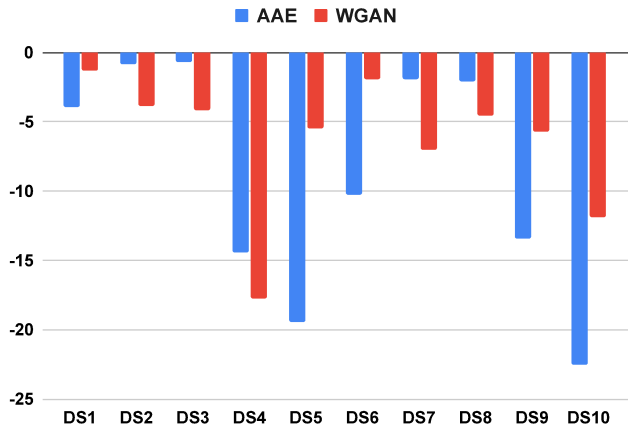
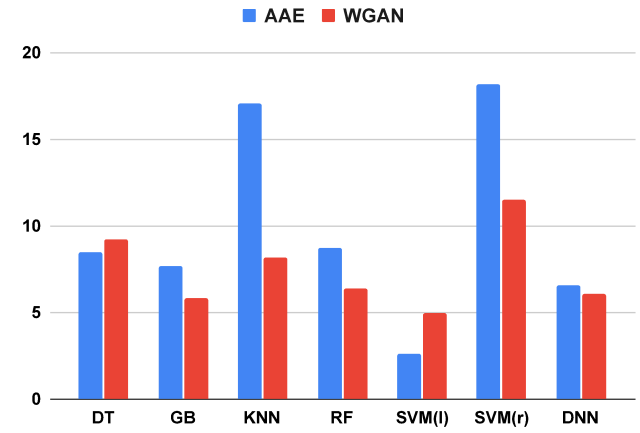


Fig. 4: This figure depicts the percentage of increase in F1 score when the model is trained with both original and synthesized samples or it is trained exclusively with original samples without considering synthesized samples. In both cases, models are tested with synthesized samples.

(a) Classifiers performances



(b) Datasets performances

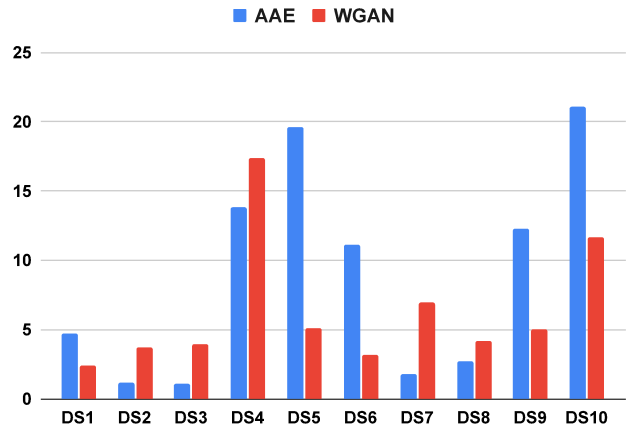


Figure 4 depicts improvements over F1 score that happened after injecting synthesized samples into the training set. Note that Δ^4 is significantly high when we consider the average on datasets and classifiers. Δ^4 scores, for all cases, have improved. Among different classifiers, KNN and SVM with Gaussian kernel.

The average of Δ^4 is 7.4% and 6.1% respectively, while the decrease in F1 score in the previous experiment was 7.1% and 5.6%. This shows that the F1 score has improved by the same amount degraded in the previous experiment. These results prove that Hypothesis-4, as the proposed approach, was able to recover the model to the level that we had when we were using only original samples.

F. Hyp-5: Improving F1 score for original dataset.

In previous experiments, we showed injecting synthesized samples into the training set mitigates the vulnerability against synthesized samples. However, this is not the only benefit of using our approach. We want to analyze if augmenting the training dataset with synthesized samples improves the F1 score of the models over those trained only on original samples. We denote Δ^5 as this difference. Table V summarizes results for two algorithms of AAE and WGAN.

Table V demonstrates F1 score has been slightly improved for different datasets and algorithms. As original scores for these models are significantly high, even a small improvement is considerable. Besides, the improvement for specific algorithms is significant. For example, the F1 score has been increased for the DS1 dataset in all cases. DS1 is one of the

DS	DT		GB		KNN		RF		SVC(l)		SVC(r)		DNN		Max	
	A	W	A	W	A	W	A	W	A	W	A	W	A	W	A	W
DS1	1	1.6	0.4	1.6	1.4	1.6	0.2	1	0.8	0.4	1	0.7	1.6	0.8	0.85	1.09
DS2	-1	-0.4	-1.2	-1.3	1	0.8	0.3	0	0.1	0.1	0.3	-0.4	-0.8	-1.3	0.4	-0.16
DS3	0.9	-0.9	0.4	0.6	-0.7	-0.9	-0.7	-3.7	0.9	4.1	-1.1	-1.1	2.7	-1.5	0.28	-0.2
DS4	-1.5	-0.9	-1.2	-0.3	1	1.9	-0.3	-0.7	-0.8	-1.1	-0.3	2.3	-0.5	0.1	-0.48	-0.36
DS5	0	0.2	-0.5	-0.8	-1.5	1.1	0.3	-0.4	-0.4	-1.6	3.2	-0.5	-0.3	-1.2	0.3	-0.44
DS6	0.3	-0.1	-1.3	0.2	-0.1	0.5	0.1	0.2	-0.3	-0.3	10	0	0.2	0.1	2.42	1.24
DS7	0	0.2	-0.2	-0.6	0	0.1	-0.1	0	-0.2	-0.8	-0.3	0.1	-0.1	-0.7	-0.19	-0.11
DS8	0.1	-1.1	-0.4	-0.7	-0.1	-1	0.2	-0.3	1	-0.4	1.5	0	1.3	-0.1	0.65	-0.49
DS9	0.2	-0.5	-0.2	-0.5	-0.6	-1	-0.4	0.2	-1.5	-0.7	0	-2.6	-0.7	0.1	-0.53	-0.7
DS10	-0.5	-0.3	-0.1	-0.5	-0.9	0	0.7	-0.2	-5.3	0.5	-2.1	-1.2	-2.7	-0.1	-1.53	-0.15
Avg.	-0.05	-0.22	-0.43	-0.23	-0.05	0.31	0.03	-0.39	-0.57	0.02	1.22	-0.27	0.07	-0.38		

TABLE V: **F1 score for Hyp-5.** Table reports results of F1 scores for different classifiers for two synthesizer algorithms of **A**AE and **W**GAN for all datasets. It also reports **Max**imum F1 score for each dataset and **Ave**rage for each classifiers.

datasets with the lowest number of instances. This shows that our approach can be useful to enhance the size of the dataset and have higher performance. On average, results of SVM with the linear kernel (SVM(l)) have been increased by more than 1.2% with AAE.

The improvement in F1 score that Δ^5 scores have demonstrated proves Hypothesis-5. In other words, extending the dataset with synthesized samples helped to improve the F1 score of the system for both synthesized and original samples.

V. Conclusion and Future Work

Supervised machine learning is a promising approach for phishing detection. Adequate amount of data about phishing websites are often infeasible to obtain for reasons of privacy, confidentiality, and liability. In order to address this problem, we develop AAE and WGAN based technique for generating data that mimic phishing and genuine websites. We ensure that the features of the synthesized phishing samples can be realistically produced by an attacker. We use 10 publicly available datasets (with different feature sets) for our experiments. Our experiments ensure that injecting synthesized data in the training set improved the F1 score of the learning algorithms. Moreover, including some correctly labeled synthesized data in the training set produced algorithms that were significantly more robust to exploratory attacks. Our future work involves the use the AAE and WGAN for other security related domains where it is hard to obtain attack data, *e.g.* generating attack data for Internet of Things or Cyber Physical Systems. In this study, we evaluated only statistical learning models. In future, we plan to explore the vulnerability of neural networks against adversarial attacks. We also plan to explore different attack types and make learning models more robust against wider range of attacks.

Acknowledgment

We thank Sid Sutton, who helped us with implementation. This work was supported in part by NSF with award numbers DMS 1923142, DMS 2123761, CNS 1932413, IIS 2027750, CNS 1715458, and CNS 1822118 and by American Megatrends, Statnett, Cyber Risk Research, ARL, NIST, and The NewPush. This work was also supported by the U.S. Department of Justice, Office of Justice Programs/National Institute of Justice under Award 2017-ZA-CX-0002. Opinions or points of view expressed in this article are those of the authors and do not necessarily reflect the official position of policies of the funding agencies.

References

[1] Grant Ho, Asaf Cidon, Lior Gavish, Marco Schweighauser, Vern Paxson, Stefan Savage, Geoffrey M Voelker, and David Wagner. Detecting and characterizing lateral phishing at scale. In USENIX Security Symposium, 2019.

[2] Federal Bureau of Investigation (FBI). Business e-mail compromise 12 billion dollar scam. https://www.ic3.gov/media/2018/180712.aspx, (accessed July 2, 2020).

[3] Shivender Singh, Anil K Sarje, and Manoj Misra. Client-side counter phishing application using adaptive neuro-fuzzy inference system. In International Conference on Computational Intelligence and Communication Networks, 2012.

[4] Rachna Dhamija, J Doug Tygar, and Marti Hearst. Why phishing works. In Proceedings of the SIGCHI conference on Human Factors in computing systems, pages 581–590, 2006.

[5] Amirreza Niakanlahiji, Bei-Tseng Chu, and Ehab Al-Shaer. Phishmon: A machine learning framework for detecting phishing webpages. In Intelligence and Security Informatics, 2018.

[6] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. Machine learning based phishing detection from urls. Expert Systems with Applications, 2019.

[7] Jian Mao, Jingdong Bian, Wenqian Tian, Shishi Zhu, Tao Wei, Aili Li, and Zhenkai Liang. Phishing page detection via learning classifiers from page layout feature. EURASIP Journal on Wireless Communications and Networking, 2019.

[8] Ankit Kumar Jain and Brij B Gupta. Towards detection of phishing websites on client-side using machine learning based approach. Telecommunication Systems, 2018.

[9] Jens Kirchner, Andreas Heberle, and Welf Löwe. Classification vs. regression-machine learning approaches for service recommendation based on measured consumer experiences. In IEEE World Congress on Services, 2015.

[10] Z. Dou, I. Khalil, A. Khreishah, A. Al-Fuqaha, and M. Guizani. Systematization of knowledge (sok): A systematic review of software-based web phishing detection. IEEE Communications Surveys Tutorials, 2017.

[11] Sebastian Abt and Harald Baier. Are we missing labels? a study of the availability of ground-truth in network security research. In 2014 Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, pages 40–55, 2014.

[12] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In ACM workshop on Security and artificial intelligence, 2011.

[13] Hossein Shirazi, Bruhadeshwar Bezawada, and Indrakshi Ray. "kn0w thy doma1n name": Unbiased phishing detection using domain name based features. In Access Control Models and Technologies, 2018.

[14] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. An assessment of features related to phishing websites using an automated technique. In Internet Technology And Secured Transactions, 2012.

[15] Neda Abdelhamid, Aladdin Ayesh, and Fadi Thabtah. Phishing detection based associative classification data mining. Expert Systems with Applications, 2014.

[16] Choon Lin Tan. Phishing dataset for machine learning: Feature evaluation, 2018. https://data.mendeley.com/datasets/h3cgnj8hft/1 (Accessed 2019-05-12).

[17] Salima Hannousse, Abdelhakim; Yahiouche. Web page phishing detection, 2020. https://data.mendeley.com/datasets/c2gw7fy2j4/2 (Accessed 2020-09-12) .

[18] Grega Vrbančič. Phishing dataset for machine learning: Feature evaluation, 2020. http://dx.doi.org/10.17632/72ptz43s9v.1 (Accessed 2020-10-12).

[19] Grega Vrbančič, Iztok Fister, and Vili Podgorelec. Datasets for phishing websites detection. Data in Brief, 33:106438, 2020.

[20] Moruf Adebowale. Phishing dataset for machine learning: Feature evaluation, 2019. https://data.mendeley.com/datasets/gt7xdbs3kt/2 (Accessed 2020-09-12) .

[21] M.A. Adebowale, K.T. Lwin, E. Sánchez, and M.A. Hossain. Intelligent web-phishing detection and protection scheme using integrated features of images, frames and text. Expert Systems with Applications, 115:300 – 313, 2019.

[22] Hadi Sadoghi Yazdi, emad mahmodi, and Abbas Ghaemi Bafghi. Data for: An online minimal uncertainty drift-aware method for anomaly detection in social networking, 2020. https://data.mendeley.com/datasets/zw7knrxpy5/1 (Accessed 2020-09-12) .

[23] Hadi Sadoghi Yazdi, Abbas Ghaemi Bafghi, et al. A drift aware adaptive method based on minimum uncertainty for anomaly detection in social networking. Expert Systems with Applications, 162:113881, 2020.

[24] MUHAMMAD HANIF. Malware webpages data, 2020. https://data.mendeley.com/datasets/zsj5pgrsg9/1 (Accessed 2020-09-12) .

[25] Jiwon Hong, Taeri Kim, Jing Liu, Noseong Park, and Sang-Wook Kim. Phishing url detection with lexical features and blacklisted domains. In Adaptive Autonomous Secure Cyber Systems, pages 253–267. Springer, 2020.

[26] Vaibhav Patil, Pritesh Thakkar, Chirag Shah, Tushar Bhat, and SP Godse. Detection and prevention of phishing websites using machine learning approach. In International Conference on Computing Communication Control and Automation, 2018.

[27] Xin Zhou and Rakesh Verma. Phishing sites detection from a web developer's perspective using machine learning. In 53rd International Conference on System Sciences, 2020.

[28] Abdulhamit Subasi and Emir Kremic. Comparison of adaboost with multiboosting for phishing website detection. Procedia Computer Science, 168:272–278, 2020.

[29] Hossein Shirazi, Landon Zweigle, and Indrakshi Ray. A machine-learning based unbiased phishing detection approach. In Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, pages 423–430. SciTePress, 2020.

[30] Saad Al-Ahmadi, Afrah Alotaibi, and Omar Alsaleh. Pdgan: Phishing detection with generative adversarial networks. IEEE Access, 10:42459–42468, 2022.

[31] Sharif Amit Kamran, Shamik Sengupta, and Alireza Tavakkoli. Semi-supervised conditional gan for simultaneous generation and detection of phishing urls: A game theoretic perspective. arXiv preprint arXiv:2108.01852, 2021.

[32] Katherine Haynes, Hossein Shirazi, and Indrakshi Ray. A machine-learning based unbiased phishing detection approach. In Proceedings of the 18th International Conference on Mobile Systems and Pervasive Computing, 2021.

[33] Rosa L Figueroa, Qing Zeng-Treitler, Sasikiran Kandula, and Long H Ngo. Predicting sample size required for classification performance. BMC Medical Informatics and Decision Making, 2012.

[34] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[35] Hossein Shirazi, Bruhadeshwar Bezawada, Indrakshi Ray, and Charles Anderson. Adversarial sampling attacks against phishing detection. In IFIP Annual Conference on Data and Applications Security and Privacy, 2019.

[36] Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando Perez-Cruz. Passgan: A deep learning approach for password guessing. In Applied Cryptography and

Network Security, 2019.

[37] Weiwei Hu and Ying Tan. Generating adversarial malware examples for black-box attacks based on gan, 2017.

[38] Vahid Mirjalili, Sebastian Raschka, Anoop Namboodiri, and Arun Ross. Semi-adversarial networks: Convolutional autoencoders for imparting privacy to face images. In Conference on Biometrics, 2018.

[39] Benjamin WK Hung, Anura P Jayasumana, and Vidarshana W Bandara. INSiGHT: A system to detect violent extremist radicalization trajectories in dynamic graphs. Data & Knowledge Engineering, 2018.

[40] Benjamin WK Hung, Anura P Jayasumana, and Vidarshana W Bandara. Finding emergent patterns of behaviors in dynamic heterogeneous social networks. IEEE Transactions on Computational Social Systems, 2019.

[41] S. R. Muramudalige, B. W. K. Hung, A. P. Jayasumana, and I. Ray. Investigative graph search using graph databases. In 2019 First International Conference on Graph Computing (GC), pages 60–67, 2019.

[42] Brett K. Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P. Bhavnani, James Brian Byrd, and Casey S. Greene. Privacy-preserving generative deep neural networks support clinical data sharing. bioRxiv, 2018.

[43] Choong Ho Lee and Hyung-Jin Yoon. Medical big data: promise and challenges. Kidney research and clinical practice, 36(1):3, 2017.

[44] C. M. Kattadige, S. R. Muramudalige, K. N. Choi, G. Jourjon, H. Wang, A. P. Jayasumana, and K. Thilakarathna. Videotrain: A generative adversarial framework for synthetic video traffic generation. In IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2021.

[45] Jytte Klausen, Rosanne Libretti, Benjamin W. K. Hung, and Anura P. Jayasumana. Radicalization trajectories: An evidence-based computational approach to dynamic risk assessment of homegrown jihadists. Studies in Conflict & Terrorism, 2018.

[46] Shashika R. Muramudalige, Anura P. Jayasumana, and Haonan Wang. A comparative study of complex data object generation with likelihood and deep generative approaches. 2021. In Progress.

[47] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. International Conference on Neural Information Processing Systems, 2014.

[48] Junchi Yan. Recent advance in temporal point process: from machine learning perspective. SJTU Technical Report, 2019.

[49] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. arXiv preprint arXiv:1703.06490, 2017.

[50] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian Goodfellow. Adversarial autoencoders. In International Conference on Learning Representations, 2016.

[51] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In International conference on machine learning, pages 214–223, 2017.

[52] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. Advances in neural information processing systems, 30:5767–5777, 2017.

[53] Sina Fathi-Kazerooni and Roberto Rojas-Cessa. Gan tunnel: Network traffic steganography by using gans to counter internet traffic classifiers. IEEE Access, 8:125345–125359, 2020.

[54] Jian Chen, Xuxin Zhang, Rui Zhang, Chen Wang, and Ling Liu. De-pois: An attack-agnostic defense against data poisoning attacks. IEEE Transactions on Information Forensics and Security, 16:3412–3425, 2021.

[55] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial machine learning. In ACM Workshop on Security and Artificial Intelligence, 2011.

[56] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. Journal of Machine Learning Research, 2011.

[57] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository. http://archive.ics.uci.edu/ml (Accessed 2019-05-12).

[58] Samuel Marchal. Phishstorm - phishing / legitimate url dataset. aalto university, 2014. https://data.mendeley.com/datasets/c2gw7fy2j4/2 (Accessed 2020-09-12) .

[59] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. Phishstorm: Detecting phishing with streaming analytics. IEEE Transactions on Network and Service Management, 11(4):458–471, 2014.