Hindawi
International Transactions on Electrical Energy Systems
Volume 2022, Article ID 2890761, 11 pages
https://doi.org/10.1155/2022/2890761

Research Article

Multiple Adversarial Domains Adaptation Approach for Mitigating Adversarial Attacks Effects

Bader Rasheed,[1] **Adil Khan**,[1] **Muhammad Ahmad**,[2] **Manuel Mazzara**,[3] **and S. M. Ahsan Kazmi**[4]

[1]*Institute of Data Science and Artificial Intelligence, Innopolis University, Innopolis, Russia*
[2]*Department of Computer Science, National University of Computer and Emerging Sciences, Islamabad, Pakistan*
[3]*Institute of Software Development and Engineering, Innopolis University, Innopolis, Russia*
[4]*Faculty of Computer Science and Creative Technologies, University of the West of England, Bristol, UK*

Correspondence should be addressed to Adil Khan; a.khan@innopolis.ru

Received 6 July 2022; Accepted 19 September 2022; Published 10 October 2022

Academic Editor: R. Sitharthan

Although neural networks are near achieving performance similar to humans in many tasks, they are susceptible to adversarial attacks in the form of a small, intentionally designed perturbation, which could lead to misclassifications. The best defense against these attacks, so far, is adversarial training (AT), which improves a model's robustness by augmenting the training data with adversarial examples. However, AT usually decreases the model's accuracy on clean samples and could overfit to a specific attack, inhibiting its ability to generalize to new attacks. In this paper, we investigate the usage of domain adaptation to enhance AT's performance. We propose a novel multiple adversarial domain adaptation (MADA) method, which looks at this problem as a domain adaptation task to discover robust features. Specifically, we use adversarial learning to learn features that are domain-invariant between multiple adversarial domains and the clean domain. We evaluated MADA on MNIST and CIFAR-10 datasets with multiple adversarial attacks during training and testing. The results of our experiments show that MADA is superior to AT on adversarial samples by about 4% on average and on clean samples by about 1% on average.

1. Introduction

Machine learning (ML) and deep learning (DL) have achieved remarkable performance in providing intelligent solutions in different domains [1, 2]. Nevertheless, ML and DL systems have shown susceptibility to adversarial attacks in the form of small purposely created perturbations leading to misclassifications, which could render ML models useless, especially in security applications [3–6]. Moreover, the generated adversarial examples for one model can be transferred to attack other models [7]. The field of adversarial machine learning got popular over the past few years, and many defense methods were proposed to protect models against adversarial attacks. Among these methods, adversarial training (AT) [8] is the most popular defense, which works by training the model not only on clean samples but on generated adversarial samples as well.

Although AT has been shown to help increase the robustness of deep learning models, it has some drawbacks. More specifically, AT might lead to overfitting on the used attacks, making the model robust only against the seen attacks and failing to generalize against unseen attacks or new methods [9]. Furthermore, AT increases the model's robustness at the expense of decreasing its accuracy on clean data. To mitigate these drawbacks, our goal is to use methods of domain adaptation to reduce the gap between the adversarial distribution and the clean distribution. We aim at learning robust features that are domain-invariant between the clean domain and the adversarial domain. In this regard, we refer to clean samples as the source domain and adversarial samples as the target domain.

Generally, domain adaptation methods take advantage of the immense development in adversarial learning techniques to learn domain-invariant features by minimizing the statistical distance between the source and target domains [10]. These methods essentially work by aligning the global distributions of source and target domains, without exploring the underlying complicated multimodal nature of these distributions [11]. This could lead to an irrelevant alignment of the domain distributions [12], especially in diverse domain adaptation scenarios such as adversarial samples. Even if the global adversarial and clean domains have been aligned correctly, the adversarial and clean samples with the same label could still be mapped far from each other in the feature space. Therefore, the main points to consider when employing domain adaptation techniques for solving the problem of adversarial attacks are

(1) Learning domain-invariant robust features by maximally matching the multimodal domain distributions
(2) Preventing the incorrect modes' alignment in the adversarial and clean distributions
(3) Minimizing the intraclass distance by aligning the clean and adversarial samples from the same class as closely as possible
(4) Maximizing the interclass distance by aligning the clean and adversarial samples from different classes as far as possible leads to a reduced effect of adversarial attacks since the required perturbation to construct an adversarial example would have to be large

Accordingly, in this paper, we propose a novel multiple adversarial domain adaptation (MADA) method, which uses adversarial domain adaptation for learning robust domain-invariant features. Instead of simply considering the classification loss on adversarial and clean samples, as in AT, we consider finding an optimum alignment of the adversarial and clean domains. This helps in decreasing the sample space for adversarial examples. The overall architecture consists of three components: a feature generator, a domain critic, and a classifier. The domain critic is trained to play a min-max game with the feature generator [13] by maximizing the Wasserstein distance between the adversarial and the clean samples. On the other hand, the feature generator is trained to produce robust features by minimizing the Wasserstein distance between the adversarial and the clean samples. The feature generator also considers a classification loss to prevent any incorrect alignment of modes in the adversarial and clean distributions and considers a triplet loss to minimize the intraclass distance and maximize the interclass distance.

In short, the contributions of this paper are directed at improving the generalization of AT on both adversarial and clean samples by formulating the problem as a multiple-domain adaptation task where adversarial domains represent target domains. Specifically, we introduce a novel domain adaptation approach and employ it to minimize the distance between the clean and adversarial domains. We evaluate our MADA method using MNIST and CIFAR-10 datasets on FGSM, PGD, and BIM adversarial attacks. Experimental results show that MADA generalizes better than AT on all conducted tests.

2. Background

This section covers the necessary background materials, organized into the following three sections.

2.1. Adversarial Attacks. ML in general and DL in specific have achieved good performance in many areas such as computer vision, audio recognition, natural language processing, and many other domains [14, 15]. Nevertheless, recent studies showed that these systems have unpleasant susceptibility to adversarial examples [3], where a small unrecognizable perturbation is added to the input sample, causing the model to misclassify this sample. Attackers can exploit this gap in models' behavior, making the models useless in real-life scenarios.

Formally, suppose a multiclass classification model $h: X^{m*n} \longrightarrow \widehat{Y}^{m*k}$, where m is the number of samples, n is the dimension of the input space, and k is the number of classes. We can generally define an adversarial attack on an input sample $x \in X$ as

$$x^{\mathrm{adv}}: h\left(x^{\mathrm{adv}}\right) \neq h(x) s.t: \left\|x^{\mathrm{adv}} - x\right\|_p \leq \Delta, \tag{1}$$

where Δ is the allowed perturbation, which determines the amount of change that we can add to the input. The amount of perturbation is measured using the l_p norm. Different attacks use different l_p norms, and the most popular ones are l_0, l_2, and l_∞.

Adversarial attacks are classified according to their knowledge of the target model into white-box, gray-box, and black-box attacks [16]. In the case of white-box attacks, the adversaries have full knowledge of the target model, including model architecture and weights, which makes crafting such attacks easier. In gray-box attacks, the adversaries have limited knowledge of the target model, such as the predicted probability of each class. In black-box attacks, the attacker can only query the model to get the final prediction, and the restriction could include the number of allowed queries. However, the existing white-box attacks are transferable to many gray-box and black-box settings [7].

Several attack methods were introduced in the literature to find x^{adv}. The first method was the fast gradient sign method (FGSM) [8]. It tries to maximize the loss function by finding the gradients of the loss with respect to the input sample and updating the sample along the direction of the gradient with a restriction on the L_∞ norm of perturbation so that the difference between adversarial and clean samples is imperceptible. Mathematically

$$x^{\mathrm{adv}} = x + \varepsilon * \operatorname{sign}\left(\nabla_x J\left(h(x), y_{\mathrm{true}}\right)\right) s.t: \left\|x^{\mathrm{adv}} - x\right\|_\infty \leq \Delta. \tag{2}$$

Further works introduced stronger iterative attacks. One example is the basic iterative method (BIM) [17], which is similar to FGSM but runs for multiple iterations. It creates iterative perturbations as

$$\begin{aligned} x_0^{\text{adv}} &= x, \\ x_{t+1}^{\text{adv}} &= x_t^{\text{adv}} + \alpha * \text{sign}\left(\nabla_{x_t^{\text{adv}}} J\left(h\left(x_t^{\text{adv}}\right), y_{\text{true}}\right)\right), \\ x_{t+1}^{\text{adv}} &= \text{clip}\left(x_{t+1}^{adv}, x_{t+1}^{adv} - \varepsilon, x_{t+1}^{adv} + \varepsilon\right), \end{aligned} \tag{3}$$

where t determines the iteration number, ϵ is the attack step or the attack learning rate, and $\text{clip}(\text{input}, a, b)$ restricts the adversarial sample to reside in the range $[a, b]$. The attack starts from the original point itself, iteratively adds perturbations in a direction that maximizes the loss, and then clips the final result to a feasible area. This area could be the pixels range $[0, 255]$ in the image domain.

Similarly, the projected gradient descent (PGD) [17] attack works iteratively and restricts the maximal perturbation by projecting the perturbed sample into a feasible area. Different from BIM, which initializes the first point as the original sample, PGD initializes the first point randomly within the area around the original sample. The noisy initialization of PGD leads to a stronger attack that converges better.

Carlini and Wagner (C and W) [18] proposed another popular attack that reformulates the optimization problem by minimizing the distance between the adversarial samples and clean samples and changes the perturbation variable so that the adversarial sample always resides in the allowed range of the optimization process. Jacobian-based saliency map approach (JSMA) [19] attack computes the Jacobian matrix of the logit layer with respect to the input and then defines the adversarial saliency map to find the best input features to change to obtain the attack. Universal adversarial patch attack finds the patch perturbation (such as an eyeglass frame) [20] in a restricted region of the input sample by optimizing the perturbation overall benign samples. Some of these adversarial attacks are summarized in Table 1.

To mitigate the effect of adversarial attacks, different adversarial defense methods were introduced, including heuristic and certificated defenses [16]. Heuristic defense methods defend against a particular attack with no guarantee of the same performance on other attacks. On the other hand, certified defense approaches give robustness certifications for the lowest performance under any adversarial attacks with well-defined constraints.

The most reliable heuristic defense method is AT [23], which improves the model's robustness by augmenting adversarial examples to the training dataset. Randomization-based defenses [24] try to introduce some randomization to the input or the model architecture during inference. This random transformation is expected to eliminate the effect of adversarial perturbation. Other works [25] tried to use denoising on the input or high-level features to mitigate the effect of adversarial perturbation.

As mentioned earlier, certified defense methods tend to give a certificate on the model accuracy under specific situations, regardless of the used attack method to fool the model. These methods try to prove the upper bounds of the model's robustness. For example [26], we considered an optimization procedure as a linear program that minimizes the worst case over convex relaxation of the set of activations reachable through a norm-bounded perturbation. However, scalability is still a common issue for this kind of defense.

2.2. *Adversarial Domain Adaptation (ADA).*

In unsupervised domain adaptation, we are given a labelled source domain $D_s = (x_i^s, y_i^s)_{i=1}^{m_s}$ of m_s source samples and an unlabeled target domain $D_t = (x_j^t)_{j=1}^{m_t}$ of m_t target samples [27]. The source domain and target domain are sampled from distributions $\mathbb{P}(X_s, Y_s)$ and $\mathbb{Q}(X_t)$, respectively, where $\mathbb{P} \neq \mathbb{Q}$. ADA aims to design and implement an adversarial learning approach for generating robust features $z = G_z(x)$ by reducing the distance between the target and source distributions with an adaptive multiclass classifier $y = G_y(z)$ such that the expected classification risk is minimized with a cross-entropy loss as follows:

$$L_{\text{cls}}\left(X, Y, G_z, G_y\right) = -\mathbb{E}_{(x,y)\sim(X,Y)} \sum_{k=1}^{K} \mathbb{I}[k = y] \log G_y\left(G_z(x)\right). \tag{4}$$

The adversarial learning approach is implemented as a two-player game between a domain discriminator G_d and a feature extractor G_z. G_d is trained to differentiate between the source domain and the target domain, whereas G_z is trained to fool G_d.

G_z extracts domain-invariant features z by learning the parameters θ_z that maximize the L_d loss of domain discriminator G_d. The domain discriminator G_d tries to distinguish samples from the two domains by learning the parameters θ_d that minimize the loss L_d. Moreover, the parameters θ_y of the label predictor G_y are learned to predict the class label of the input sample. Thus, the objective function of the domain adversarial network is

$$\begin{aligned} C_0\left(\theta_z, \theta_y, \theta_d\right) = {} & \frac{1}{m_c} \sum_{x_i \in D_s} L_{\text{cls}}\left(G_y\left(G_z(x_i)\right), y_i\right) \\ & - \frac{\lambda}{m} \sum_{x_i \in \left(D_s \cup D_t\right)} L_d\left(G_d\left(G_z(x_i)\right), d_i\right), \end{aligned} \tag{5}$$

where $m = m_c + m_a$ and λ is a trade-off parameter. At the point of convergence, we get the parameters $\hat{\theta}_z, \hat{\theta}_y, \hat{\theta}_d$ as the optimal solution for equation (5) as

$$\begin{aligned} \left(\hat{\theta}_z, \hat{\theta}_y\right) &= \text{argmin}_{\theta_z, \theta_y} C_0\left(\theta_z, \theta_y, \theta_d\right), \\ \left(\hat{\theta}_d\right) &= \text{argmax}_{\theta_d} C_0\left(\theta_z, \theta_y\right). \end{aligned} \tag{6}$$

2.3. *Wasserstein Distance.*

The interesting approach of adversarial learning consisting of a discriminator and a generator trying to compete against each other motivates both to improve their functionalities and eventually converge. Basically, the loss function of the generator evaluates how close the synthetic data distribution is and the real data distribution, which is measured using Jensen–Shannon (JS) and Kullback–Leibler (KL) divergence [28]. However, using the JS metric for simultaneous adversarial learning between

Table 1: Summary of adversarial attacks on neural networks.

Attack	Attack frequency	Perturbation Norm	Attack strategy
L-BFGS [21]	One-shot	l_∞	Constrained optimization
FGSM [8]	One-shot	l_∞	Gradient optimization
I-FGSM [22]	Iterative	l_∞	Gradient optimization
PGD [17]	Iterative	l_∞	Gradient optimization
C and W [18]	Iterative	l_∞, l_0, l_2	Constrained optimization

the discriminator and the generator cannot guarantee a convergence, especially, when the two distributions are disjoint [29]. When the discriminator is perfect, we get $G_d(x) = 1$, $\forall x \in \mathbb{P}_r$ and $G_d(x) = 0$, $\forall x \in \mathbb{P}_g$, where $\mathbb{P}_r$ is the real distribution and $\mathbb{P}_g$ is the generated distribution. In this case, the loss function is equal to zero, and we have no gradients to update the parameters of the generator and the discriminator during learning.

To solve these issues, the Wasserstein metric was used instead of JS divergence since it has a much smoother value space [30]. Wasserstein Distance, also known as Earth mover's distance, is another metric for measuring the distance between distributions. It is interpreted as the minimum energy required to transform one distribution to look like another. The formula of the Wasserstein distance is

$$W\left(\mathbb{P}_r, \mathbb{P}_g\right) = \inf_{\gamma \sim \prod\left(\mathbb{P}_r, \mathbb{P}_g\right)} \mathbb{E}_{(x,y)\in\gamma} \|x - y\|_p, \tag{7}$$

where γ is the transport plane, and it is a joint distribution, from $\prod(\mathbb{P}_r, \mathbb{P}_g)$ which is the set of all possible joint probability distributions between $\mathbb{P}_r$ and $\mathbb{P}_g$. Specifically, $\gamma(x, y)$ is the percentage of mass that should be moved from point x to y so that x comes from the same probability distribution of y. Once the amount of mass required to be moved from x to y is moved, the marginal distribution over x should add up to $\mathbb{P}_g$: $\sum_x \gamma(x, y) = \mathbb{P}_g(y)$ and similarly $\sum_y \gamma(x, y) = \mathbb{P}_r(x)$. For finding the cost using EM, $\gamma(x, y)$ is treated as the amount of mass to be moved, and $\|x - y\|$ is the mass traveling distance. The greatest lower bound indicates the minimum cost among all visible ones. Then, the expected cost averaged across all the (x, y) pairs are

$$\begin{aligned} W\left(\mathbb{P}_r, \mathbb{P}_g\right) &= [t] \inf_{\pi \sim \prod\left(\mathbb{P}_r, \mathbb{P}_g\right)} \mathbb{E}_{(x,y)\in\pi} \|x - y\|_p, \\ \sum_y \pi(x, y) &= [t]\mathbb{P}_r(x), \\ \sum_x \pi(x, y) &= [t]\mathbb{P}_g(y). \end{aligned} \tag{8}$$

However, finding all the possible joint distributions in $\prod(\mathbb{P}_r, \mathbb{P}_g)$ is an intractable problem. Thus, the authors of [30] proposed to solve the dual problem using the Kantorovich–Rubinstein duality, which is expressed as follows:

$$W\left(\mathbb{P}_r, \mathbb{P}_g\right) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)]. \tag{9}$$

In the above equation, the lower upper bound is made over all the K-Lipschitz functions [31] $f\colon X \longrightarrow \mathbb{R}$ and K is the Lipschitz constant for the function $f(.)$. f comes from a family of K-Lipschitz continuous functions, $\{f_w\}_{w\in W}$, parameterized by w. Intuitively, the Lipschitz constraint makes the function $f(.)$ smoother and prevents it from fast changes. Two common approaches exist for enforcing the constraint of 1-Lipschitz in the above equation: gradient penalty and weight clipping. Gradient clipping restricts the weight w value in f, into a certain range controlled by the hyperparameters c. However, this method may undergo a gradient vanishing problem. On the other hand, gradient penalty works by enforcing the gradients to have a norm at most 1 everywhere:

$$L_{\text{grad}}(\hat{x}) = \left(\left\|\nabla_{\hat{x}} f_w(\hat{x})\right\|_2 - 1\right), \tag{10}$$

where $\hat{x}$ points are defined not only in real and generated samples but at all points between them, and all $\hat{x}$ points should have a gradient norm of 1 for f.

To utilize the Wasserstein distance in AT, the discriminator's loss function is configured as measuring the Wasserstein distance between the two distributions $\mathbb{P}_r$ and $\mathbb{P}_g$. Thus, the discriminator function is not to directly differentiate between fake samples and real ones. Instead, it is trained to find the Wasserstein distance between the two distributions by learning a K-Lipschitz continuous function. That is why, it is called a critic. As the generator g_θ generates more realistic samples that are similar to the original ones, its loss function decreases during training, and the Wasserstein distance gets smaller. Therefore, the training function should find the optimal value for w parameters of the function $f(.)$ from the following formula:

$$\begin{aligned} L\left(\mathbb{P}_r, \mathbb{P}_g\right) = W\left(\mathbb{P}_r, \mathbb{P}_g\right) &= \max_{w\in W} \mathbb{E}_{x\sim\mathbb{P}_r}[f_w(x)] \\ &\quad - \mathbb{E}_{x\sim\mathbb{P}_g}[f_w(g_\theta(x))]. \end{aligned} \tag{11}$$

3. Multiple Adversarial Domain Adaptation

In this section, we describe our proposed approach (MADA). We first formulate the problem as a domain adaption problem and then explain how MADA achieves the global domain alignment and class-level alignment.

3.1. Formulation. As mentioned earlier, in MADA, we formulate the defense against adversarial attacks as a domain adaptation problem. We are given one clean domain $D_c = (x_{c_{(i)}}, y_{c_{(i)}})_{i=1}^{m_c}$ of m_c clean samples and adversarial domains $D_a^d = (x_{a_{(j)}}^d, y_{a_{(j)}}^d)_{j=1}^{m_a^d}$ of m_a^d adversarial samples of each adversarial distribution corresponding to each adversarial attack and $d\colon 1 \longrightarrow n$ where n is the number of adversarial distributions or considered adversarial attacks. The clean domain and adversarial domains are sampled from distributions $\mathbb{P}(X_c, Y_c)$ and $\mathbb{Q}_a^d(X_a^d, Y_a^d)$, respectively, where $\mathbb{P} \neq \mathbb{Q}_a^d$. MADA aims at designing and implementing an adversarial learning approach for learning robust features $z = G_z(x)$ and adaptive multiclass classifier $y = G_y(z)$ while reducing the distance between the adversarial and clean distributions, such that the expected risk on the adversarial domains is minimized.

However, in the adversarial domain adaptation problem, the class boundaries in clean and adversarial distributions could have complicated multimodal structures. Thus, the formulation of the problem in (5) might not maximally match the distributions, or they could be incorrectly aligned. To solve this problem, some studies design multiple class-wise domain discriminators. The idea is to use one separate discriminator for aligning each semantic class, which helps in mitigating the incorrect alignment of domain distributions. However, allocating separate discriminators does not consider the interclass relationship and basically forces all classes to be orthogonal with each other [32]. Employing class structural information from the label space could help in capturing the multimodal structure, especially, in the problem of the adversarial attack where the class relationships should remain consistent across the adversarial and clean domains.

Thus, we are targeting a multiadversarial domain adaptation method for solving the problem of adversarial attacks. The extracted features should guarantee that a clean sample x and adversarial samples generated from it should be as close as possible in the embedding space (intraclass distance minimization). On the other hand, the clean samples from other classes and adversarial samples generated from them should be as far as possible from x. For this purpose, MADA automatically and adaptively searches for robust generalized features shared by clean and adversarial domains.

In other words, MADA is a new domain adaptation-inspired method that jointly aligns the clean and adversarial distributions at both class level and data level. To minimize the statistical distribution distance at the data level, we use Wasserstein distance, whereas we adapt a triplet loss to align the adversarial and clean distributions at the class level. Adversarial learning is used to reduce the domain shift between the distributions by performing an adversarial game between a feature extractor G_z from one side and a domain critic G_D from the other side. Through the data-level and class-level alignment approach, discriminative and robust domain-invariant features could be learned. Therefore, the feature space shared by all domains can be automatically discovered after the feature generator fools the domain critic successfully. The full architecture of MADA components is shown in Figure 1.

3.2. Global Domain Alignment.

3.2. Global Domain Alignment. Three elements are involved to globally align the distributions using Wasserstein distance in this stage, namely the feature extractor G_z, classifier G_y, and domain critic G_D. Global domain alignment is achieved after finishing adversarial learning between the feature extractor G_z and other components, and domain-invariant robust features are obtained.

As was mentioned in (11), the loss function of the critic is adapted as finding the best $f_w(.)$ that minimizes the Wasserstein distance between source and target domains. In our problem, we have the clean domain $\mathbb{P}_c$ as the source domain and the family of adversarial attack domains $\mathbb{Q}_a^d$ as the target domains. By generalization to n domains, equation (11) becomes

$$\begin{aligned} L_{gb} &= L\left(\mathbb{P}_c, \mathbb{Q}_a^d\right) = W\left(\mathbb{P}_c, \mathbb{Q}_a^d\right) \\ &= \sum_{d=1}^{d=n} \sum_{x^c \in X^c} \frac{1}{m_c} \sum_{x^c \in X^c} \left[f_w\left(f_g\left(x^c\right)\right)\right] \\ &\quad - \frac{1}{m_a^d} \sum_{x_a^d \in X_a^d} \left[f_w\left(f_g\left(x_a^d\right)\right)\right]. \end{aligned} \tag{12}$$

To enforce the Lipschitz constrain, we use gradient penalty as follows:

$$L_{\text{grad}}(\hat{x}) = \left(\left\|\nabla_{\hat{x}} f_w(\hat{x})\right\|_2 - 1\right). \tag{13}$$

Therefore, the objective function of domain critic D becomes

$$\max_{\theta_w}\left(L_{gb} - \lambda_1 L_{\text{grad}}\right), \tag{14}$$

where λ_1 is a balancing parameter. The other component of adversarial learning is the feature extractor G_z, and its goal is to generate domain-invariant features by minimizing the Wasserstein distance between the clear distribution from one side and the adversarial distributions from the other side with respect to parameter θ_g while keeping the parameters θ_w of the critic G_d fixed as follows:

$$\min_{\theta_g} \max_{\theta_w}\left(L_{gb} - \lambda_1 L_{\text{grad}}\right). \tag{15}$$

The goal of training the classifier G_y is to find the optimal θ_y for classifying the samples from the clean and adversarial domains. The classifier depends on the features generated by G_z as input and contains many fully connected layers. The objective function for optimizing G_y is

$$\min_{\theta_y} L_{\text{cls}} = \sum_{(x_i, y_i) \in \left(D_c \cup D_a^d\right)} H\left(G_y\left(G_z\left(x_i\right)\right), y_i\right), \tag{16}$$

where $H(.)$ here is the cross-entropy loss and (x, y) is the available-labeled samples in the clean and adversarial domains. The objective function becomes the following equation:

$$\min_{\theta_g, \theta_c}\left\{L_c + \lambda_2 \max_{\theta_w}\left(L_{gb} - \lambda_1 L_{\text{grad}}\right)\right\}. \tag{17}$$

In the original WGAN [30] paper, the authors suggest performing five training updates for the discriminator for each update of the generator (critic training step $n = 5$). This number is not fixed and should be changed according to the network architecture complexity of the generator and discriminator. However, in our setting, we observe that when we increase the network complexity, the generator could

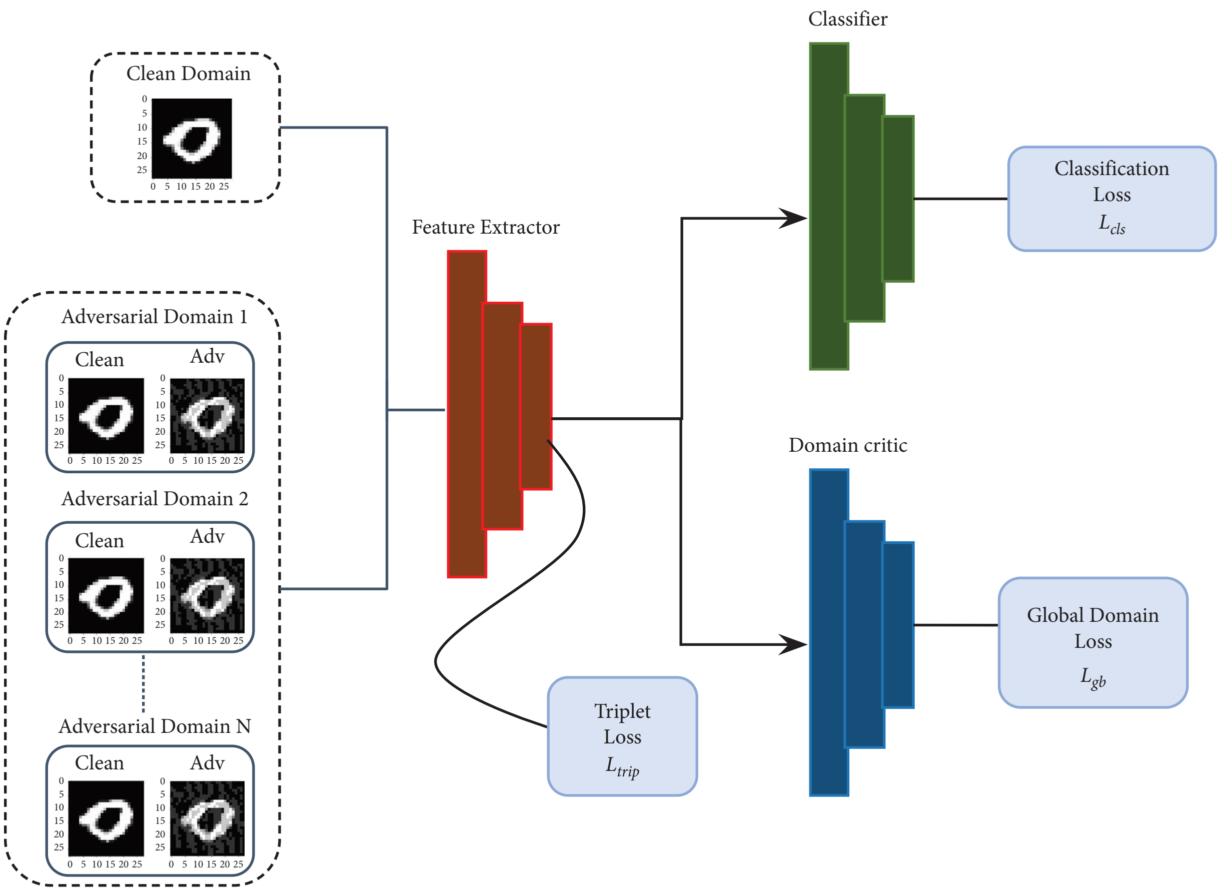

Figure 1: The architecture of the proposed method.

easily overcome the discriminator after a relatively small number of epochs. In this case, the required critic training step becomes large (more than 15). This makes the training very expensive since we need to perform gradient penalty estimation in every step. That is why, we slightly modify the training process so that we do not perform any unnecessary updates for the domain critic but at the same time do not allow the generator to overcome the discriminator by adaptively changing the critic training step n. We check the Wasserstein distance for the discriminator W_d and generator W_g and keep updating the discriminator until W_d is larger than W_g. Our final algorithm is described in Algorithm 1.

3.3. Class-Level Alignment. Reducing the global distribution discrepancy, without considering the class-level association among the source and target samples, could lead to semantic misalignment. To solve this problem, we add a class similarity-preserving constraint to our objective function. As a result, samples with the same labels should be pulled closer to the feature space, and samples with different labels should be pushed far from each other. This class-level alignment can be implemented by minimizing a triplet loss so that clean and adversarial features embedding maintains intraclass closeness and interclass separability [33]. Triplet loss operates on three samples as input: an anchor x_a that is any arbitrary sample, a positive sample x_p that has the same class as the anchor, and a negative sample x_n that has a different class from the anchor. Triplet loss works by minimizing the distance in the feature space between the anchor sample and the positive sample and maximizing the distance between the anchor sample x^a and the negative sample. Mathematically, the triplet loss function is defined as follows:

$$L_{\text{trip}} = \sum_{x^a,x^p,x^n} \max\left(D\left(x^a, x^p\right) - D\left(x^a, x^n\right) + m, 0\right), \tag{18}$$

where m is the margin by which the distance between the anchor and the positive sample is at least larger than the distance between the anchor and the negative sample.

In our setting, the anchor x_c^a is the clean sample classified by a classifier f_1 with a true label y^a, the positive sample x_{adv}^p is an adversarially perturbed sample classified with label y^p, but it should be classified as y^a, and the negative sample x_{adv}^n is an adversarially perturbed sample classified with label y^a, but it should be classified as $y^n \neq y^a$. f_1 is a classifier trained on the clean domain only, so it considers accuracy as the only measure of performance and does not consider the robustness of the model. Hence, the triplets set in our settings are

$$\begin{aligned}\tau = \Big\{\left(x_c^a, x_{\text{adv}}^p, x_{\text{adv}}^n\right) | f_1\left(x_c^a\right) \\ \neq f_1\left(x_{\text{adv}}^p\right), \text{and } f_1\left(x_c^a\right) = f_1\left(x_{\text{adv}}^n\right)\Big\}.\end{aligned} \tag{19}$$

For each of the target adversarial domains, we compose triplet training samples by attacking the clean source dataset with an adversarial attack of each of the adversarial target domains. We use the batch hard strategy for choosing proper

```
Require: A clean dataset $(X^c, Y^c)$, minibatch size $m$, learning rates $(\alpha_{ftr}, \alpha_{cr}, \alpha_{\text{cls}})$, critic training step $n$, adversarial domains d
1 Initialize feature generator $G_z$, domain critic $G_d$, and classifier $G_y$, with weights $\theta_z, \theta_d, \theta_y$
2 Repeat
3 Sample clean minibatch $\{x_i^c, y_i^c\}_{i=1}^m$
4 for $j = 1, \ldots, d$ do
5 Use the current state of $G_z, G_y$ to generate adversarial examples and create adversarial minibatch $\left\{x_i^{\text{adv}_j}, y_i^{\text{adv}_j}\right\}_{i=1}^m$
6 end for
7 while $W_d < Wg$ do
8 Find $L_{gb}, L_{\text{grad}}, W_d$
9 $\theta_d \leftarrow \theta_d + \alpha_{ftr} * \nabla_{\theta_d}[L_{gb} - \lambda L_{\text{grad}}]$
10 end while
11 Find $L_{\text{cls}}, L_{\text{trip}}, W_g$
12 $\theta_c \leftarrow \theta_c - \alpha_{\text{cls}} * \nabla_{\theta_c}[L_{\text{cls}}]$
13 $\theta_z \leftarrow \theta_z - \alpha_{ftr} * \nabla_{\theta_z}[L_{\text{cls}} + \lambda_2 \max_{\theta_w}(L_{gb} - \lambda_1 L_{\text{grad}}) + \lambda_3 L_{\text{trip}}]$
```

Algorithm 1: MADA algorithm.

positive and negative samples. For each clean anchor sample x_c^a in the batch, the hardest positive sample, or the farthest positive sample, is $\text{argmax}_{x_{\text{adv}}^p}(\|f(x_c^a) - f(x_{\text{adv}}^p)\|_2^2)$ and the hardest negative sample, or the closest negative sample, is $\text{argmin}_{x_{\text{adv}}^n}(\|f(x_c^a) - f(x_{\text{adv}}^n)\|_2^2)$. Our final objective function becomes as follows:

$$\min_{\theta_g, \theta_c}\left\{L_c + \lambda_2 \max_{\theta_w}\left(L_{gb} - \lambda_1 L_{\text{grad}}\right) + \lambda_3 L_{\text{trip}}\right\}. \quad (20)$$

4. Experiments

In principle, our method can be applied to any dataset and any adversarial attack. For comparison against adversarial training, we focus on two datasets on MNIST [34] and CIFAR10 [35] and three adversarial attacks PGD, BIM, and FGSM. For all experiments, we normalize the pixel values to the range [0, 1].

4.1. Experiment Setup. In every training iteration, we use FGSM, PGD, and BIM to generate three adversarial targets on the fly. To evaluate the effectiveness of our method, we compare our MADA method with

(1) normal training (NT) with cross-entropy loss [36] on the clean training data
(2) adversarial training (AT) with the cross-entropy loss on the clean training data and the adversarial examples from the FGSM, PGD, and BIM
(3) MADA without triplet loss, where we remove the triplet loss to measure the effect of class-level alignment
(4) MADA without triplet loss and classification loss, where we keep only the global domain alignment loss

For each dataset, we train a vanilla model (NT), MADA model, and three above-explained adversarial models with perturbation ϵ for comparison and evaluate these models on FGSM, PGD, and BIM attacks bounded by the same ϵ. We consider L_∞ as a measure of perturbation in all attacks. The experiments were implemented on a single GeForce GTX 1080 Ti.

The network architecture and training parameters are chosen so that they work, but they could be optimized to have better performance. In principle, any conventional image classification model can be used. The features' generator consists of a stack of convolutional layers, while the critic and classifier consist of a stack of fully connected layers. While any optimization method can be used for training, we choose Adam optimization [37] for training all the components with a batch size of 64, 200 epochs, and $(\beta_1 = 0.9, \beta_2 = 0.99)$. The learning rate starts at 0.005 and is decayed by 2 every 30 epochs. After training, the domain critic can be removed, and the robust feature generator and the classifier can be used instead of the conventional image classifier.

4.2. Experimental Results

4.2.1. Results on MNIST. Since it is not hard to classify MNIST, we use simple network architectures for the different components shown in Table 2. The allowed adversarial perturbation ϵ, in this case, is 0.3, and the maximum number of iterations for BIM and PGD is 30. The accuracy results are reported in Table 3. NT has the best accuracy on clean data but has the worst robustness or accuracy on adversarial samples. The accuracy on clean samples is almost the same between AT and MADA, but MADA increases robustness significantly on adversarial samples. We also notice the importance of classification and triplet loss, where the accuracy of MADA decreases significantly on both clean and adversarial samples when we remove them. The results in Table 3 show that MADA efficiently exploits the three proposed losses to find the best alignment between adversarial and clean domains without sacrificing the accuracy on clean samples.

4.2.2. Results on CIFAR10. Here, we use the VGG architecture since classifying CIFAR10 is harder than MNIST. The convolution layers compose the feature extractor, and the last fully connected layers form the classifier and the

Table 2: Component architecture for MNIST.

Feature Extractor	Classifier	Domain Critic
Conv2d (3, 64, 5)	Linear (32*4*4, 100)	Linear (32*4*4, 128) +ReLu
BatchNorm2d (64)	BatchNorm1d (100) + ReLu	Dropout ()
MaxPool2d (2) + LeakyReLU	Dropout ()	Linear (128, 64) +ReLu
Conv2d (64, 32, 5)	Linear (100, 50)	Dropout ()
BatchNorm2d (32)	BatchNorm1d (100) + ReLu	Linear (64, 64) +ReLu
MaxPool2d (2) + LeakyReLU	Linear (50, 10)	Linear (64, 1)

Table 3: Accuracy results on MNIST.

Defense	Clean%	FGSM	BIM	PGD
Vanilla	**98.63**	11.33	10.56	10.14
AT	97.53	91.37	89.28	87.10
MADA without L_{trip}	97.18	92.32	90.84	88.32
MADA without L_{trip} and L_{cls}	93.67	90.11	87.97	84.01
MADA	98.07	**96.81**	**96.62**	**96.01**

Table 4: Component architecture for CIFAR10.

Feature Extractor	Classifier	Domain Critic
Conv2d(3, 64, 3)	Linear (32*4*4, 100)	Linear (32*4*4, 128) +ReLu
BatchNorm2d(64) +ReLu	BatchNorm1d(100) + ReLu	Dropout()
Conv2d(64, 64, 3)	Dropout()	Linear (128, 64) +ReLu
BatchNorm2d(64) +ReLu	Linear (100, 50)	Dropout()
MaxPool2d(2)	BatchNorm1d(100) + ReLu	Linear(64, 64) +ReLu
Dropout()	Linear (50, 10)	Linear (64, 1)
Conv2d (64, 128, 3)		
BatchNorm2d(128) +ReLu		
Conv2d(128, 128, 3)		
BatchNorm2d(128) +ReLu		
MaxPool2d(2) + dropout()		
Conv2d(128, 256, 3)		
BatchNorm2d(256) +ReLu		
Conv2d(256, 256, 3)		
BatchNorm2d(256) +ReLu		
MaxPool2d(2) + dropout()		
Conv2d (256, 512, 3)		
BatchNorm2d (512) +ReLu		
Conv2d (512, 512, 3)		
BatchNorm2d (512) +ReLu		
MaxPool2d(2) + dropout()		

Table 5: Accuracy results on CIFAR10.

Defense	Clean%	FGSM	BIM	PGD
Vanilla	**83.35**	14.99	15.62	11.07
AT	82.92	64.22	61.12	49.35
MADA without L_{trip}	82.73	64.12	62.02	49.69
MADA without L_{trip} and L_{cls}	79.56	62.18	59.90	47.86
MADA	83.28	**66.16**	**63.43**	**52.61**

domain critic which do not change from MNIST as shown in Table 4.

Here, the maximum allowed perturbation is $\epsilon = 0.031$, and the number of iterations for BIM and PGD is 30. We can observe from the results in Table 5 that MADA leads to a very small drop in the clean accuracy while increasing the robustness of adversarial samples compared to AT. These results on CIFAR correspond to previously noticed observations on MNIST and show that MADA surpasses AT on both clean and adversarial samples.

4.2.3. Feature Visualization. We further investigate the difference in the distribution of the extracted features between NT and AT for the MNIST dataset. We use t-SNE (*t*-distributed stochastic neighbor embedding) to plot the

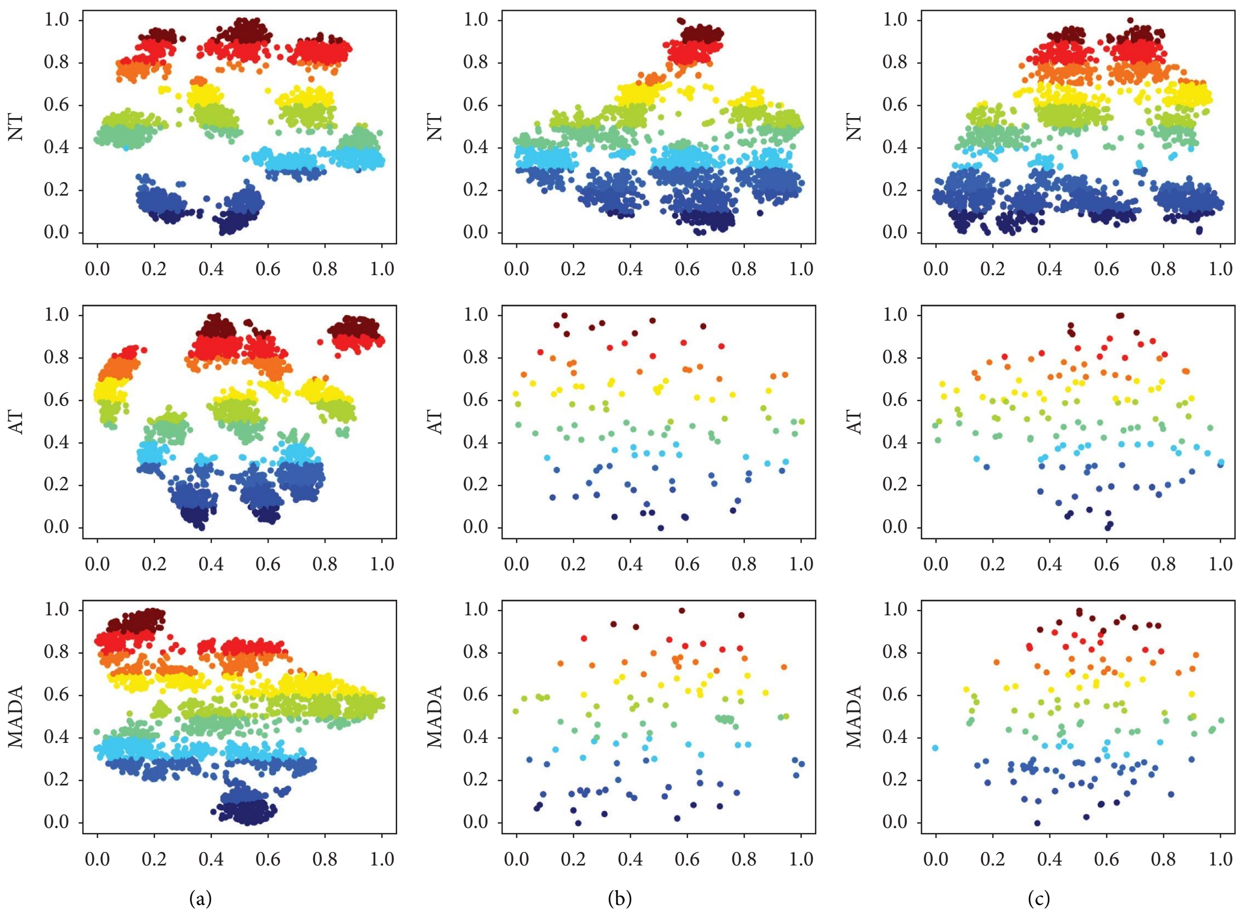

FIGURE 2: *t*-SNE visualizations for the exacted features of testing data and adversarial testing data from FGSM and PGD for the MNIST dataset. (a) Clean. (b) FGSM. (c) PGD.

embedded features in two-dimensional space. Figure 2 shows feature visualizations of testing data and adversarial data from FGSM and PGD for the NT, AT, and MADA models. The color in Figure 2 corresponds to different classes. The figures show that our method forces the model to make the data from the same class to be as close as possible to each other and as far as possible from samples from different classes. We notice also that the constructed adversarial samples in MADA are farther from the center of the class compared to NT and AT, which means that the adversarial methods need to add stronger perturbations in order to fool the model.

5. Conclusion

In this paper, we design a domain adaptation-based approach to boost the performance of adversarial training on adversarial samples. The proposed approach reduces the effect of adversarial attacks by aligning the adversarial domain distributions near the clean distribution in the feature embedding space. The experimental results show that our approach increases the generalization of the model in the adversarial domain and gives a better interpretation of the features in the embedding space. Our approach can be further developed by studying different ways to align different distributions rather than Wasserstein distance, which we keep for further research.

Data Availability

Previously reported datasets were used to support this study and are available at DOI: 10.1109/MSP.2012.2211477 and DOI: 10.1.1.222.9220. These prior studies and datasets are cited at relevant places within the text as references [28, 29].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] D. A. Neu, J. Lahann, and P. Fettke, “A systematic literature review on state-of-the-art deep learning methods for process prediction,” *Artificial Intelligence Review*, vol. 55, pp. 1–27, 2021.

[2] M. Ahmad, S. Shabbir, S. K. Roy et al., “Hyperspectral image classification—traditional to deep models: a survey for future prospects,” *Ieee Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 15, pp. 968–999, 2022.

[3] B. Rasheed, A. Khan, S. Ahsan Kazmi, R. Hussain, M. Jalil Piran, and D. Young Suh, “Adversarial attacks on featureless

deep learning malicious URLs detection," *Computers, Materials & Continua*, vol. 68, no. 1, pp. 921–939, 2021.

[4] A. R. Javed, M. Usman, S. U. Rehman, M. U. Khan, and M. S. Haghighi, "Anomaly detection in automated vehicles using multistage attention-based convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4291–4300, 2021.

[5] T. Khan, K. Singh, M. H. Hasan et al., "ETERS: a comprehensive energy aware trust-based efficient routing scheme for adversarial WSNs," *Future Generation Computer Systems*, vol. 125, pp. 921–943, 2021.

[6] A. R. Javed, S. U. Rehman, M. U. Khan, M. Alazab, and T. G. Reddy, "CANintelliIDS: detecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1456–1466, 2021.

[7] A. Demontis, M. Melis, M. Pintor et al., "Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks," in *Proceedings of the 28th USENIX Security Symposium*, pp. 321–338, USENIX Association, Berkeley, CA, USA, September 2019.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of the 3rd International Conference On Learning Representations, ICLR 2015*, San Diego, CA, USA, December 2015.

[9] L. Rice, E. Wong, and J. Z. Kolter, "Overfitting in adversarially robust deep learning," in *Proceedings of the 37th International Conference on Machine Learning ICML 2020*, Vienna, Austria, 2020.

[10] R. Shao, X. Lan, J. Li, and P. C. Yuen, "Multi-adversarial discriminative deep domain generalization for face presentation attack detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June 2019.

[11] I. E. I. Bekkouch, D. C. Nicolae, A. Khan, S. M. A. Kazmi, A. M. Khattak, and B. Ibragimov, "Adversarial reconstruction loss for domain generalization," *IEEE Access*, vol. 9, pp. 42 424–442 437, 2021.

[12] L. F. Alvarenga e Silva, D. C. Guimarães Pedronette, F. A. Faria, J. P. Papa, and J. Almeida, "Improving transferability of domain adaptation networks through domain alignment layers," 2021, https://ui.adsabs.harvard.edu/abs/2021arXiv210902693A/abstract.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds., vol. 27, Curran Associates, Inc, Red Hook, Ny, USA, 2014https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[14] L. Deng and D. Yu, *Deep Learning: Methods and Applications*, Now Publishers, Norwell, MA, USA, 2013.

[15] B. Rasheed and A. Y. Popov, "Network graph datastore using DiSc processor," in *Proceedings of the 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering*, pp. 1582–1587, Institute of Electrical and Electronics Engineers Inc, Moscow, Russia, January 2019.

[16] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, vol. 6, no. 3, pp. 346–360, 2020.

[17] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *Artificial Intelligence Safety and Security*, CRC Press, Boca Raton, FL, USA, 2019.

[18] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the IEEE Symposium on Security and Privacy*, San Jose, CA, USA, May 2017.

[19] N. Papernot, P. Mcdaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proceedings of the 2016 IEEE European Symposium on Security and Privacy*, pp. 372–387, Saarbruecken, Germany, March 2016.

[20] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 1528–1540, Association for Computing Machinery, New York, NY, USA, October 2016.

[21] C. Szegedy, W. Zaremba, I. Sutskever et al., "Intriguing properties of neural networks," in *Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, Banff, Canada, April 2014.

[22] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *Proceedings of the ICLR 2017 - Workshop Track Proceedings. International Conference on Learning Representations*, Toulon, France, July 2019.

[23] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, https://arxiv.org/abs/1706.06083.

[24] C. Xie, Z. Zhang, A. L. Yuille, J. Wang, and Z. Ren, "Mitigating adversarial effects through randomization," 2017, https://arxiv.org/abs/1711.01991.

[25] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: detecting adversarial examples in deep neural networks," 2018, https://arxiv.org/abs/1704.01155.

[26] E. Wong and J. Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *Proceedings of the 35th International Conference on Machine Learning*, pp. 8405–8423, International Machine Learning Society (IMLS), Stockholm, Sweden, November 2018.

[27] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pp. 2962–2971, Honolulu, HI, USA, January 2017.

[28] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep generative modelling: a comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[29] S. A. Barnett, "Convergence problems with generative adversarial networks (GANs)," 2018, https://arxiv.org/abs/1806.11382.

[30] M. Arjovsky, S. Chintala, and L. Bottou, "WGAN," 2017, https://arxiv.org/abs/1701.07875v3.

[31] J. Heinonen, "Lectures on lipschitz analysis," 2005, https://papers2://publication/uuid/44B1237F-0C8C-4207-B0EB-42AA9ED3A96D.

[32] Z. Wang, B. Jing, Y. Ni, N. Dong, P. Xie, and E. Xing, "Adversarial domain adaptation being aware of class relationships," in *Proceedings of the 24th European Conference on Artificial Intelligence*, vol. 325, pp. 1579–1586, Santiago de Compostela, Spain, 2019.

[33] X. Wang and F. Liu, "Triplet loss guided adversarial domain adaptation for bearing fault diagnosis," *Sensors*, vol. 20, no. 1, p. 320, 2020.

[34] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141-142, 2012.

[35] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features from Tiny Images*, pp. 1–58, Citeseerx, Princeton, NJ, USA, 2009.

[36] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Advances in Neural Information Processing Systems*, vol. 2018, pp. 8778–8788, 2018.

[37] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *Proceedings of the 3rd International Conference On Learning Representations, ICLR 2015 - Conference Track Proceedings*, San Diego, CA, USA, May 2015.