Learning Robust Neural Networks using Wasserstein Adversarial GAN

Shashank Goel*
Department of Computer Science
University of California Los Angeles
Los Angeles, CA 90095
shashankgoel@cs.ucla.edu

Parth Shah*
Department of Computer Science
University of California Los Angeles
Los Angeles, CA 90095
parthushah8@cs.ucla.edu

Harini Suresh
Department of Computer Science
University of California Los Angeles
Los Angeles, CA 90095
sharini16@cs.ucla.edu

Abstract

Deep neural networks are highly expressive and powerful learning models that achieve close to human-level performance on many different tasks. However, previous studies have shown that they learn relatively discontinuous functions and are uninterpretable to a significant extent. Many studies have focused on a better understanding of adversarial examples, specially designed inputs to a model perceptually similar to a given input but with a different classification label. Furthermore, algorithms that generate these adversarial examples rely on L_p-norms as a measure for the perceptual similarity between the input and the adversary. However, some recent studies have shown that L_p-norm is an unnecessary and inadequate measure to indicate the perceptual distance between two images. In this work, we propose a novel method of generating complex adversarial examples that may have large L_p-norms but are perceptually more similar relative to the benign input and establish state-of-the-art results in robustness.

1 Introduction

In previous research paradigms (Szegedy et al. [21], Hein and Andriushchenko [9], Goodfellow et al. [8], Madry et al. [15], Papernot et al. [16], Tramèr et al. [22], Carlini and Wagner [2]), adversarial attack and defense mechanisms have measured the degree to which an adversarial example is indistinguishable from its benign original using L_p-norms. For instance, as proposed in Sharif et al. [20], L_0-norm measures the number of different pixels, L_2-norm measures the euclidean distance, and L_{inf}-norm measures the largest distance between corresponding pixels of the two images. However, it has been implicitly assumed that L_p-norm difference is indicative of the perceptual or visual similarity of humans. Notwithstanding, it has been shown that images can be perceptually similar even with a higher L_p-norm difference and experimental approaches to show that L_p distance may be inappropriate for generating adversarial examples (Sharif et al. [20], Jacobsen et al. [11]).

Formally, we will devise an estimate of the perceptual distance D between two images. One can trivially show that L_p-norms don't generate perfect estimates by applying standard pre-processing

*equal contribution

Preprint. Under review.

transformations like cropping, rotation, etc. that cause the benign image to remain perceptually similar to the transformed image but have a rather high L_p distance. It's not trivial to check whether two images are perceptually similar without human supervision. A potential way around this hurdle is learning a discriminator network that outputs a perceptual similarity score for two input images. The discriminator network training uses data augmentation techniques like rotation, recolorization, cropping, flipping, etc. Further, we can train a generator network to fool the discriminator proposed in Goodfellow et al. [6], and generate adversarial examples. However, a drawback of this naive approach is that these transformations may not cover the entire space of perturbations necessary for achieving robustness, and the generator can potentially learn an identity mapping. A novel idea of generating adversarial examples, AdvGANs, has been proposed in Xiao et al. [26] to accommodate these problems. The loss function consists of three different loss objectives 1) to penalize higher norm of the noise in the generated image, 2) the min-max loss of the GAN architecture, and 3) to penalize if the same class label is generated for the output as the initial input when passed through a trained classifier network. The adversary generated for an input x is $x + G(x)$, where G is the generator network and $G(x)$ represents the adversarial noise. The loss function uses a hinge objective on the norm of $G(x)$ given by $L_{hinge} = \mathbb{E}_x \left[\max(0, \|G(x)\|_2 - c)\right]$. We aim to design a network that creates adversarial examples with a high value of $\|G(x)\|_2$. Moreover, we modify the network architecture to accommodate for variations in output adversary by passing an additional input sampled from a normal distribution. Further, we use the Wasserstein distance as a critic to generate perceptually similar adversaries. By using these techniques, we can generate multiple adversaries with a possibly higher L_p norm distance from the single input image that our classifier can use to learn a robust parameter space.

We plan to evaluate our attack mechanism by comparing the accuracy scores of the generated adversarial examples using PGD, AdvGANS and our WAGAN model. Furthermore, we calculate the improved accuracy of the model trained on data augmented using the different attack mechanisms and show it to be more accurate and robust. We use the MNIST (Deng [5]) data set while performing all of our experiments and plan to develop models using Pytorch (Paszke et al. [18]). We compare our attack mechanisms to other attack mechanisms that find adversarial examples having smaller L_p distance by comparing the model accuracy when trained on data augmented by various attack mechanisms. Also, we will show that model trained using data augmented from our attack mechanism to be more generalizable than other models over randomly generated adversarial examples using different attack mechanisms.

2 Related Work

Adversarial Examples Recently, substantial effort has gone towards generating adversarial examples for a target model, which are specially crafted inputs that are perceptually similar to original inputs from real datasets but are classified incorrectly. There has been a number of papers studying threat model based on L_p distance to identify perceptually similar inputs, ranging from improved attacks, heuristic and certified defenses, and verifiers. Some popular mechanisms to generate adversarial examples include the Fast gradient sign method (FGSM) by Goodfellow et al. [7], which applies a first-order approximation of the loss function to construct adversarial samples. Optimization-based methods (Opt) have also been proposed to optimize adversarial perturbation for targeted attacks while satisfying certain constraints by Carlini and Wagner [2] and Liu et al. [12]. Also, many commonly used methods for generating adversarial examples use a form of projected gradient descent over the region of allowable perturbations, originally referred to as the Basic Iterative Method proposed by Madry et al. [14]. Parallelly, new defense mechanisms have also developed like distillation by Papernot et al. [17] and one of the most practically successful one by Madry et al. [13] which uses the adversarial examples in adversarial training.

Neural Networks to generate adversarial examples Also, neural networks have been applied to generate adversarial perturbation; some examples include Baluja and Fischer [1] which combines the re-ranking loss and an L_2 norm loss, encouraging the generated adversarial instance to be close to the original one in terms of L_2 distance. Although the most interesting is the one using Generative adversarial networks(GAN) (Goodfellow et al. [6]) known as AdvGAN [Xiao et al. [26]], where they model a generator that crafts those adversarial examples and a discriminator that encourages the generator to make perceptually similar inputs and additional loss functions are present which

encourages the generator to have low magnitude perturbations that fools the target model. This was the state-of-the-art attack mechanism achieving 88.93% accuracy in the semi-Whitebox setting and 92.7% in the black-box setting on MNIST challenge (Madry et al. [13]) and won the top position.

Wasserstein distance threat model To measure perceptual similarity between two images, most prior work has used the L_p distance metric in attacks that craft adversarial examples, usually using L_0, L_2 or $L_{\inf}$ to minimize the change in the original image. But recent work by Sharif et al. [20] suggests that the nearness according to L_p-norm is unnecessary as well as insufficient for perceptual similarity. This motivated us to look into other distance metrics like the Wasserstein distance, which is a better way to represent the perceptual distance between two images. Wasserstein distances measure the cost of moving pixel mass, which naturally cover "standard" image manipulations such as scaling, rotation, translation, and distortion. Wong et al. [25] were the first to introduce a new Wasserstein distance-based threat model for adversarial attacks. Our goal through this paper is to extend the previous state-of-the-art Adversarial GAN attack framework over the Wasserstein distance threat model, which proves to be a better adversary, and fine-tuning the target model on it results in more robust models.

3 Preliminaries

3.1 PGD based attack

The most prevalent method for constructing adversarial instances is using a variation of projected gradient descent (Madry et al. [14]). Let $\mathcal{X}$ be the set of instances, and $\mathcal{Y}$ be the set of labels. The projection of an instance $x \in \mathcal{X}$ onto a set $\mathcal{S} \subseteq \mathcal{X}$ is defined as

$$\underset{\mathcal{S}}{\text{proj}}(x) = \underset{p \in \mathcal{S}}{\text{argmin}} \, \|x - p\|_2^2$$

representing a point inside the set S, closest to the given point x in the Euclidean space. Let $\mathcal{B}(x, \delta)$ denote a ball of radius δ centered around the point x in the instance space $\mathcal{X}$, reflecting the adversary's threat model, defined by

$$\mathcal{B}(x, \delta) = \{p \in \mathcal{X} : \|x - p\|_2^2 \leq \ \delta\}$$

The projection of an instance $x \in \mathcal{X}$ onto the ball $\mathcal{B}(x, \delta)$ can be expressed as

$$\underset{\mathcal{B}(x,\delta)}{\text{proj}} \, (x) = \underset{p \in \mathcal{B}(x,\delta)}{\text{argmin}} \, \|x - p\|_2^2$$

Iterative Projected Gradient Descent: Given an input example $(x, y) \in \mathcal{X} \times \mathcal{Y}$, let $x_{\text{adv}}^{(t)}$ be the generated adversarial example at time step t. Further, let α be the step size, f_θ be the target model, and $\mathcal{L}$ be the loss function. Then, the projected gradient descent algorithm consists of the following iteration at time step $t + 1$,

$$x_{\text{adv}}^{(t+1)} = \underset{\mathcal{B}(x,\delta)}{\text{proj}} \left[x_{\text{adv}}^{(t)} + \underset{\|w\| < \alpha}{\text{argmax}} \, w^T \left(\frac{\partial}{\partial x_{\text{adv}}^{(t)}} \mathcal{L} \left(f_\theta \left(x_{\text{adv}}^{(t)} \right), y \right) \right) \right] \tag{1}$$

where $x_{\text{adv}}^{(0)}$ is initialized randomly such that $x_{\text{adv}}^{(0)} \in \mathcal{B}(x, \delta)$. The gradient step in the equation forces the adversary to move in the direction of steepest ascent of the loss $\mathcal{L}$ of the target model while still staying in the ball of radius δ around the input x at every iteration.

3.2 Wasserstein based attack

The Wasserstein distance (Villani [23]) is defined as the optimal cost of the standard transport problem in the fractional space that can be interpreted in the context of distributions as the most inexpensive way to change one distribution into another by shifting the probability mass using a linear transformation. When applied to images, it can be thought of as the cost of moving pixel mass

from one pixel to another pixel such that the shifting cost per unit mass increases with the distance between the source and the destination pixels.

Formally, let $x, y \in \mathbb{R}_+^n$ be two non-negative vectors such that $\|x\|_1 = \|y\|_1 = 1$. This condition on the norm of the vectors is not strict in the theoretical sense. It is used for stability purposes and maintains a distance scale for comparison. Let $d(i, j)$ be some given distance function between the elements x_i and y_j. Further, let $C \in \mathbb{R}_+^{n \times n}$ be a given non-negative cost matrix such that C_{ij} represents the cost of moving a unit mass from x_i to y_j. In a typical scenario, $C_{i'j'} > C_{ij}$ if and only if $d(i', j') > d(i, j)$ (e.g. $C_{ij} = d(i, j)$). Then, the Wasserstein distance $d_{\mathcal{W}}$ between x and y is defined as

$$\begin{aligned} d_{\mathcal{W}}(x, y) &= \min_{\Pi \in \mathbb{R}_+^{n \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} \Pi_{ij} C_{ij} \\ \text{subject to } \forall i \in [n], \sum_{j=1}^{n} \Pi_{ij} &= x_i \text{ and } \forall j \in [n], \sum_{i=1}^{n} \Pi_{ij} = y_j \end{aligned} \tag{2}$$

The minimization is carried over the transport policy plan Π, where Π_{ij} denotes the amount of mass that moves from x_i to y_j. Further, we can define the Wasserstein ball (Wong et al. [24]) with radius δ as

$$\mathcal{B}_{\mathcal{W}}(x, \delta) = \{x + \Delta : d_{\mathcal{W}}(x, x + \Delta) \leq \delta\}$$

Projected Gradient Descent (Equation 1) using the Wasserstein ball is not-trivial because calculating the Wasserstein distance itself requires us to solve an optimization problem.

Note: In the conventional literature, the p-Wasserstein distance formulation uses the cost matrix C such that for every $i, j \in [n]$, we have

$$C_{ij} = \left((i_x - j_x)^2 + (i_y - j_y)^2 \right)^{p/2}$$

where (i_x, i_y) (respectively (j_x, j_y)) is the pixel coordinate in the original image corresponding to the index i (respectively j) in the flattened vector of that image.

The following section describes an efficient tractable approach to estimate the Wasserstein distance.

3.3 Efficient Tractable Approximation of Wasserstein distance using Sinkhorn Iterations

Solving for the Wasserstein distance using the dual formulation makes it intractable in learning with the adversarial GAN architecture that we use in this paper. Therefore, to employ an iterative approach for calculating the Wasserstein distance, instead of just optimizing for the cost of moving the pixel mass, we add an entropy term on the transport plan (Peyré and Cuturi [19]) and modify the problem as follows

$$\begin{aligned} \mathcal{H}(\Pi) &= -\sum_{i=1}^{n} \sum_{j=1}^{n} \Pi_{ij} \left(\log \left(\Pi_{ij} \right) - 1 \right) \\ d_{\mathcal{W}}(x, y) &= \min_{\Pi \in \mathbb{R}_+^{n \times n}} \left[\sum_{i=1}^{n} \sum_{j=1}^{n} \Pi_{ij} C_{ij} - \epsilon \mathcal{H}(\Pi) \right] \\ \text{subject to } \forall i \in [n], \sum_{j=1}^{n} \Pi_{ij} &= x_i \text{ and } \forall j \in [n], \sum_{i=1}^{n} \Pi_{ij} = y_j \end{aligned} \tag{3}$$

Since the objective is an ϵ-strongly convex function (Peyré and Cuturi [19]), the above problem has a unique optimal solution. The convergence of the solution of that regularized problem toward an

optimal solution of the original linear program has been studied with precise asymptotics (Cominetti and San Martin [3]). We state without proof that the unique solution for the entropy regularized formulation (3) of the original problem converges to the optimal solution with maximal entropy within the set of all optimal solutions of the actual problem (2).

Introducing two dual vectors $\alpha, \beta \in \mathbb{R}^n$, corresponding to each set of marginal constraints, the Lagrangian $\mathcal{L}(\alpha, \beta, \Pi)$ of the above optimization problem can be expressed as

$$\begin{aligned}
\mathcal{C}(\Pi) &= \sum_{i=1}^{n}\sum_{j=1}^{n} \Pi_{ij} C_{ij} \\
\mathcal{H}(\Pi) &= -\sum_{i=1}^{n}\sum_{j=1}^{n} \Pi_{ij} \left(\log\left(\Pi_{ij}\right) - 1\right) \\
\mathcal{L}(\alpha, \beta, \Pi) &= \mathcal{C}(\Pi) - \epsilon\mathcal{H}(\Pi) - \sum_{i=1}^{n} \alpha_i \left(\sum_{j=1}^{n} \Pi_{ij} - x_i\right) - \sum_{j=1}^{n} \beta_j \left(\sum_{i=1}^{n} \Pi_{ij} - y_j\right)
\end{aligned} \tag{4}$$

Using the first order condition, at optimality, we have

$$\frac{\partial \mathcal{L}(\alpha, \beta, \Pi)}{\partial \Pi_{ij}} = C_{ij} + \epsilon \log \Pi^*_{ij} - \alpha^*_i - \beta^*_j = 0$$

Therefore, we have

$$\Pi^*_{ij} = \exp\left(\alpha^*_i/\epsilon\right)\ \exp\left(-C_{ij}/\epsilon\right)\ \exp\left(\beta^*_j/\epsilon\right) \tag{5}$$

Hence, the unique solution to the regularized formulation can be expressed, for every $i, j \in [n]$, as

$$\begin{aligned}
&\Pi^*_{ij} = u_i K_{ij} v_j \\
&\text{subject to } K_{ij} = \exp\left(-C_{ij}/\epsilon\right) \text{ and } u_i, v_j \geq 0
\end{aligned} \tag{6}$$

Further, the optimal transport plan Π^* must satisfy the following non-linear equations, which correspond to the mass conservation constraints,

$$\begin{aligned}
\sum_{j=1}^{n} \Pi^*_{ij} &= u_i \sum_{j=1}^{n} K_{ij} v_j = x_i \quad , \forall i \in [n] \\
\sum_{i=1}^{n} \Pi^*_{ij} &= v_j \sum_{i=1}^{n} u_i K_{ij} = y_j \quad , \forall j \in [n]
\end{aligned} \tag{7}$$

The set of equations (7) can be solved iteratively using the Sinkhorn's algorithm (Cuturi [4]) with the following set of update equations

$$\begin{aligned}
u_i^{(t+1)} &= \frac{x_i}{\sum_{j=1}^{n} K_{ij} v_j^{(t)}} \quad , \forall i \in [n] \\
v_i^{(t+1)} &= \frac{y_j}{\sum_{i=1}^{n} u_i^{(t+1)} K_{ij}} \quad , \forall j \in [n]
\end{aligned} \tag{8}$$

initialized with an arbitrary positive values $v_j^0 = 1, \forall j \in [n]$. Further, the optimal transport plan Π^* and the optimal cost i.e., the Wasserstein distance $d_{\mathcal{W}}(x, y)$, can be calculated using the following set of equations,

$$\Pi_{ij}^* = u_i^T K_{ij} v_j^T$$

$$d_{\mathcal{W}}(x, y) = \min_{\Pi \in \mathbb{R}_+^{n \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} \Pi_{ij}^* C_{ij} \tag{9}$$

It may be useful to note that a different initialization of v_j might lead to a different solution for u_i and v_j, since if u_i, v_j are solutions to the set of equations 7 then so are $\lambda u_i, v_j/\lambda$ for any $\lambda > 0$. It turns out, however, that these iterations converge and result in the same optimal policy (Wong et al. [24]) defined in 9. An efficient vectorized implementation of the above iterative algorithm is given below:

Efficient Approximation for Wasserstein Distance

Input: Feature vector $\boldsymbol{X} \in \mathbb{R}_+^{n \times d}$, Feature Vector $\boldsymbol{Y} \in \mathbb{R}_+^{n \times d}$, Cost Matrix $C \in \mathbb{R}^{d \times d}$
Algorithm parameters: Entropy factor $\epsilon \in (0, \infty]$, number of iterations T

Define dual variables $u, v \in \mathbb{R}_+^n$
Define kernel matrix $K \in \mathbb{R}_+^{d \times d}$

Initialize delta $\delta = 1e - 8$
Initialize dual variable $v = \mathbb{1}^{n \times d}$
Initialize kernel matrix: $K = \exp(-C/\epsilon)$

foreach *t: 1* to *T* **do**
  $u \leftarrow x/\big(vK^T + \delta\big)$
  $v \leftarrow y/(Ku + \delta)$
end foreach

$\Pi^* \leftarrow \text{diag}(u) K \text{diag}(v)$
$d_{\mathcal{W}} = \Pi^* \odot C$

return $d_{\mathcal{W}}$

4 WAGAN - Wasserstein Adversarial GAN

4.1 Problem Definition

Let $\mathcal{D}$ be a dataset comprising of m datapoints, (x_i, y_i) being the i^{th} instance of the training set, where $x_i \in \mathcal{X} \subseteq \mathcal{R}^n$ generated according to an unknown distribution $x_i \sim P_{\text{data}}$ and $y_i \in \mathcal{Y}$ the corresponding true class labels and $f : \mathcal{X} \to \mathcal{Y}$ as a classifier learned using the training set from $\mathcal{D}$. Given any instance (x, y) predicted correctly by the learned classifier $f(x) = y$, our goal is to generate an adversarial example $x_{\mathcal{A}}$ s.t. $d_{\mathcal{W}}(x, x_{\mathcal{A}}) \leq \epsilon$ so that it is perceptually very similar but it's predicted label is not the same as the true label of the original, $f(x) = y \neq f(x_{\mathcal{A}})$.

4.2 WAGAN Framework

Figure 1 illustrates the overall architecture of WAGAN Framework, which consists four major components: a Generator $\mathcal{G}$, a Discriminator $\mathcal{D}$, a Wasserstein Approximator $\mathcal{W}$ and the target neural network f. Given an instance x it passes through the generator $\mathcal{G}$ which generates a perturbation $\mathcal{G}(x)$ which gives us the corresponding adversary $x_{\mathcal{A}} = x + \mathcal{G}(x)$. Then x and $x_{\mathcal{A}}$ are sent to the discriminator $\mathcal{D}$ whose role is to distinguish the generated data $x_{\mathcal{A}}$ and the original instance x and encourage the generator to create indistinguishable data.

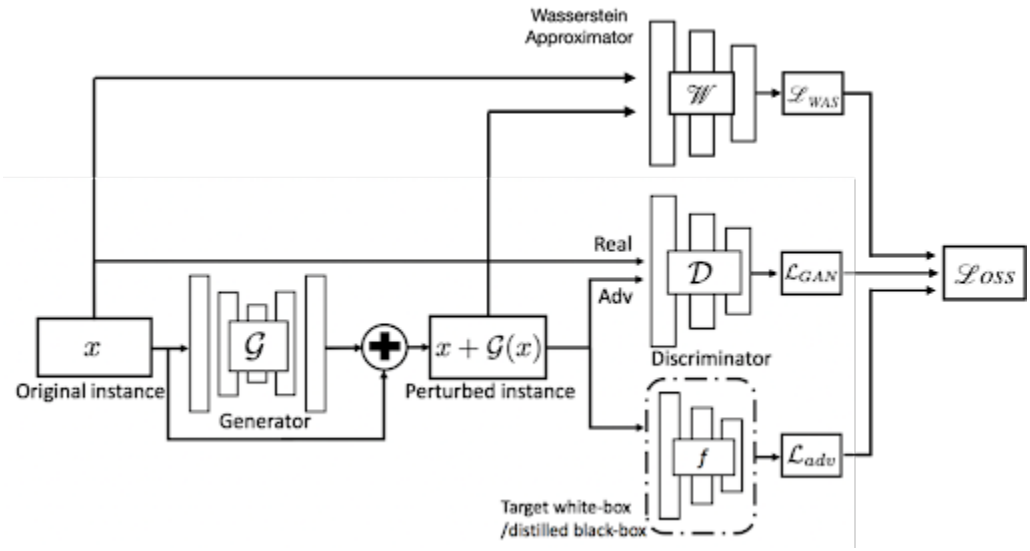

Figure 1: WAGAN Architecture

To fulfill the the goal of fooling the target model we pass $x_{\mathcal{A}}$ to the target model f to calculate it's loss which represents the opposite of the distance between the predicted class and the ground truth class. Also we want our adversary to be close to the original instance and so we pass x and $x_{\mathcal{A}}$ to the wasserstein approximator $\mathcal{W}$ to calculate a loss which represents how close they are as per the algorithm described above 9.

The final loss function used to train the generator $\mathcal{G}$ and the discriminator $\mathcal{D}$ can be broken down into four components which encourages different aspects of our adversarial model.

Adversarial Loss:

$$\mathcal{L}_{\text{gan}} = \mathbb{E}_x \log \mathcal{D}(x) + \mathbb{E}_x \log(1 - \mathcal{D}(x + \mathcal{G}(x)))$$

The discriminator $\mathcal{D}$ aims to maximize this loss which in turn encourages it to give higher score to original instances x and lower scores to perturbed data $x + \mathcal{G}(x) = x_{\mathcal{A}}$. Whereas the generator $\mathcal{G}$ aims to minimize this loss which encourages it to generate perturbed data that is indistinguishable from the original instance.

Untargeted Attack Loss:

$$\mathcal{L}_{\text{adv}}^{f} = -\mathbb{E}_x \ell_f(x + \mathcal{G}(x), y)$$

This loss encourages the generator $\mathcal{G}$ to generated perturbed data $x + \mathcal{G}(x) = x_{\mathcal{A}}$ that is classified as the true label y with low probability and therefore fooling the model under attack f.

Hinge Loss:

$$\mathcal{L}_{\text{hinge}} = \mathbb{E}_x \max(0, \|\mathcal{G}(x)\|_2 - c)$$

As previous work focused on the L_p-norm distance metric to calculate closeness of two inputs we include the hinge loss for experimental purposes. Depending on whether it directly or inversely affects the final objective it produces adversaries which have low or high perturbation. Our motivation was to originally create adversaries which are perceptually very similar to the original instance(low wasserstein distance) but are very far apart in terms of L_p distance which can be realized when the final objective inversely depends on $\mathcal{L}_{\text{hinge}}$.

Wasserstein Loss:

$$\mathcal{L}_{\text{was}} = \mathbb{E}_x \max(0, d_{\mathcal{W}}(x, x + \mathcal{G}(x)) - c)$$

This loss function encourages the generator to generate adversaries which have lower wasserstein distance from the original image it is approximated by algorithm 9 which makes the adversary

perceptually similar to the original instance. Note that c here denotes a user-specified bound which stabilizes GAN's training as shown in Isola et al. [10].

Finally our full objective can be expressed as,

$$\mathcal{L} = \mathcal{L}^{f}_{\text{adv}} + \alpha\mathcal{L}_{\text{gan}} + \beta\mathcal{L}_{\text{hinge}} + \gamma\mathcal{L}_{\text{was}}$$

where α, β and γ control the relative importance of each objective. The generator $\mathcal{G}$ and discriminator $\mathcal{D}$ model are obtained by solving the minmax game $\arg\min_{\mathcal{D}} \max_{\mathcal{G}} \mathcal{L}$. Once $\mathcal{G}$ is trained on the training data and target model, it can produce $x_{\mathcal{A}} = x + \mathcal{G}(x)$ for any given instance x.

We trained three variations of this adversarial model, WAGAN, WAGAN $\beta-$ and WAGAN $\beta+$ where $\beta = 0, \beta < 0$, and $\beta > 0$ respectively and compared the robustness of model trained over adversarial examples generated by each of these models in the following section.

5 Experimental Results

In this section, we generate adversarial examples from all the six different attack mechanisms discussed above - baseline attack mechanism, Projected Gradient Descent (PGD) - a popular baseline technique for generating adversaries, AdvGAN - the state-of-the-art model in the literature of adversarial examples generation, WAGAN, WAGAN $\beta-$ and WAGAN $\beta+$ and compare them. Table 1 summarizes the accuracy scores of the original target model against adversarial examples generated by the six different settings and Table 2 summarizes the robustness of the target model additionally trained on the six corresponding set of adversarial training examples. The experiments can be replicated on a single GPU.

Implementation Details: All our experiments were performed on the popular MNIST (Deng [5]) dataset having 32×32 images of handwritten digits. It has 60,000 training images and 10,000 test images. For the target model, we kept the batch size as 256 and trained it for 20 epochs with a 0.001 learning rate and for the next 30 epochs with a 0.0001 learning rate. For the PGD method, we use 200 iterations of gradient ascent and a maximum difference of 0.003 between the input and the projected adversarial image. For the AdvGAN model we kept the value of $\alpha = 10$ and $\beta = 1$ (as used in previous work) and for each of the variation of WAGAN model we kept the value of $\alpha = 10$, $\beta = -1, 0, +1$ for WAGAN $\beta-$, WAGAN and WAGAN $\beta+$, and $\gamma = 1$. For all the adversarial models, AdvGAN, WAGAN $\beta-$, WAGAN and WAGAN $\beta+$, we kept the batch size as 128 and trained for a total of 60 epochs with a learning rate of 0.001 for the first 50 epochs and 0.0001 for the last 10.

Evaluating Goodness of Adversarial Examples: In this experiment, we use the PGD method, the AdvGAN model and the WAGAN model with three different β settings for generating the adversarial examples. The Generative models are trained on the MNIST train dataset. We report the train and test accuracies of the target model trained using the original dataset on the different classes of adversaries. The results of the experiments are summarized in Table 1. Adversaries generated by WAGAN $\beta-$ gave the worst accuracies on the target model, also note that the adversary generated by AdvGAN are not as good for the unseen testing data, showing weak generalizability.

Dataset	Train	Test
Original	99.96	99.30
PGD.	19.87	20.35
AdvGAN	0.67	17.39
WAGAN	0.53	0.58
WAGAN $\beta-$	**0.32**	**0.44**
WAGAN $\beta+$	0.69	0.64

Table 1: Train and Test Accuracies of the trained target model on different classes of adversaries

Evaluating Robustness of Adversarial fine-tuned Target Model: In this experiment, the target model is fine-tuned with different classes of adversarial examples generated in the previous experiment. We report the test accuracies of the fine-tuned target models on the original dataset and the different classes of adversaries. The results of the experiments are summarized in Table 2. We have highlighted the fine-tuned models proving to the most robust for a particular adversarial testing set, leaving aside

the fine-tuned model trained on the adversarial training set coming from the same set of adversarial examples. Each row in the table represents the model fine-tuned using the respective adversarial training examples and each column represents the accuracies of each of those fine-tuned models on the respective adversarial testing examples. Here the WAGAN $\beta-$ fine-tuned model is emperically most robust in comparison to the other models and the adverserial test examples generated by WAGAN are the most difficult to predict correctly, very close to the adversaries generated by WAGAN $\beta-$.

Model/Dataset	**Base**	**PGD**	**AdvGAN**	**WAGAN**	**WAGAN** $\beta-$	**WAGAN** $\beta+$	**Avg**
Base	99.30	20.35	17.39	0.58	0.44	0.64	23.12
PGD	90.42	99.70	88.47	86.01	89.65	92.77	91.17
AdvGAN	91.67	93.87	99.02	88.59	91.33	96.50	93.50
WAGAN	95.22	94.69	92.22	98.24	**95.23**	94.77	95.06
WAGAN $\beta-$	**96.62**	**98.35**	**96.95**	**96.72**	98.27	**97.32**	**97.37**
WAGAN $\beta+$	95.30	97.20	96.80	92.40	93.48	98.91	95.68
Avg	94.76	84.03	81.81	**77.09**	78.07	80.15	

Table 2: Test accuracy of fine-tuned target model on different classes of adversaries

6 Conclusion

This paper proposed the Wasserstein Adversarial GAN (WAGAN) attacking framework to generate adversarial examples using generative adversarial networks (GAN) based architecture coupled with the recently introduced Wasserstein distance threat model. Once trained, the generator from the WAGAN model can be used independently to generate adversarial examples for any given instance. The generated adversarial examples are presumably perceptually similar to the original sample but misclassified by the target model. Furthermore, we showed that adversarial examples generated by WAGAN have a higher attack success rate than those generated by other competing methods, which can be seen in Table 1. We observe that its attack success rate generalizes better to new unseen instances from the testing set than a contemporary attacking framework, AdvGAN. Experiments suggest that amongst the various paradigms of WAGAN, the $\beta-$ attacking framework proves to be the most challenging adversary geerator. This analysis makes the WAGAN attacking framework a promising candidate for testing new defense mechanisms. In addition, through Table 2 we empirically show that models robust against WAGAN $\beta-$ adversarial examples are robust against other attacking frameworks, which leads us to the hypothesis that a trivial defense mechanism of additionally training the target model on adversarial training examples generated by WAGAN $\beta-$ will defend well against another attack mechanism. However, this claim needs more empirical evidence(by testing it against other attack mechanisms) and theoretical reasoning, both in the L_p-norm and Wasserstein distance paradigm, which can be worked upon in the future. Additionally, the WAGAN framework is easily mutable to any other distance metric. One exciting application could include another neural model that captures the perceptual similarity between the newly generated and the original instance. One can also create a better and more efficient Wasserstein approximator, keeping in mind that its gradient still needs to be tractable for the backpropagation step to work.

References

[1] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. In *Proceedings of AAAI-2018*, 2018. URL `http://www.esprockets.com/papers/aaai2018.pdf`.

[2] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

[3] Roberto Cominetti and Jaime San Martin. Asymptotic analysis of the exponential penalty trajectory in linear programming. *Mathematical Programming*, 67:169–187, 10 1994. doi: 10.1007/BF01582220.

[4] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transportation distances. *arXiv: Machine Learning*, 2013.

[5] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[7] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

[8] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

[9] Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 2263–2273, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. doi: 10.1109/CVPR.2017.632.

[11] Jörn-Henrik Jacobsen, Jens Behrmann, Nicholas Carlini, Florian Tramèr, and Nicolas Papernot. Exploiting excessive invariance caused by norm-bounded adversarial robustness. *ArXiv*, abs/1903.10484, 2019.

[12] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *Proceedings of 5th International Conference on Learning Representations*, 2017.

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *ArXiv*, abs/1706.06083, 2018.

[15] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

[16] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597, 2016. doi: 10.1109/SP.2016.41.

[17] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning, 2017.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[19] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Found. Trends Mach. Learn.*, 11:355–607, 2019.

[20] Mahmood Sharif, Lujo Bauer, and Michael Reiter. On the suitability of lp-norms for creating and preventing adversarial examples. pages 1686–16868, 06 2018. doi: 10.1109/CVPRW.2018.00211.

[21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199.

[22] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. Ensemble adversarial training: Attacks and defenses. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rkZvSe-RZ.

[23] Cédric Villani. *The Wasserstein distances*, pages 93–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-71050-9. doi: 10.1007/978-3-540-71050-9_6. URL https://doi.org/10.1007/978-3-540-71050-9_6.

[24] Eric Wong, Frank R. Schmidt, and J. Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations. In *ICML*, 2019.

[25] Eric Wong, Frank R. Schmidt, and J. Zico Kolter. Wasserstein adversarial examples via projected sinkhorn iterations, 2020.

[26] Chaowei Xiao, Bo Li, Jun yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3905–3911. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/543. URL https://doi.org/10.24963/ijcai.2018/543.