Generation of Low Distortion Adversarial Attacks via Convex Programming

Tianyun Zhang
EECS Department
Syracuse University
tzhan120@syr.edu

Sijia Liu
MIT-IBM Watson AI Lab
IBM Research
sijia.liu@ibm.com

Yanzhi Wang
ECE Department
Northeastern University
yanzhi.wang@northeastern.edu

Makan Fardad
EECS Department
Syracuse University
makan@syr.edu

***Abstract*—As deep neural networks (DNNs) achieve extraordinary performance in a wide range of tasks, testing their robustness under adversarial attacks becomes paramount. Adversarial attacks, also known as adversarial examples, are used to measure the robustness of DNNs and are generated by incorporating imperceptible perturbations into the input data with the intention of altering a DNN's classification. In prior work in this area, most of the proposed optimization based methods employ gradient descent to find adversarial examples. In this paper, we present an innovative method which generates adversarial examples via convex programming. Our experiment results demonstrate that we can generate adversarial examples with lower distortion and higher transferability than the C&W attack, which is the current state-of-the-art adversarial attack method for DNNs. We achieve 100% attack success rate on both the original undefended models and the adversarially-trained models. Our distortions of the L_∞ attack are respectively 31% and 18% lower than the C&W attack for the best case and average case on the CIFAR-10 data set.**

***Index Terms*—Deep neural networks, adversarial attack, convex programming.**

I. Introduction

Deep neural networks (DNNs) continue to show extraordinary performance in a variety of tasks, such as image recognition [1]–[3], speech recognition [4], [5], and natural language processing [6]. However, recent research shows that DNNs are vulnerable to adversarial attacks [7], [8]. Adversarial attacks, also known as adversarial examples, are generated by incorporating imperceptible perturbations into the original input data in order to mislead the prediction of DNNs [9], [10].

Research on the robustness of DNNs follows two directions in general. The first is to enhance the robustness of DNNs, which increases the degree of difficulty for adversarial attacks to fool DNNs [11]–[14]. The second is to design adversarial attack methods to test the robustness of DNNs [7], [15]–[18]. These two aspects reciprocally benefit each other towards hardening DNNs, and our research in this paper belongs to the latter one.

Adversarial attacks can be either untargeted or targeted. In untargeted attacks, adversarial examples are generated to fool DNNs' prediction towards a label other than the correct one [19]. In targeted attacks, adversarial examples are designed to force the DNNs to classify the data with a desired incorrect target label [7]. In this paper, we focus on the problem of targeted attack generation, as such attacks are commonly regarded as being stronger [17].

Despite the fact that the loss functions of DNNs are non-convex, most adversarial attack generation problems in the literature are solved by gradient descent; for example [20] solves the C&W attack problem via ADAM. Recent papers on certifying the robustness of DNNs employ relaxations to formulate convex optimization problems [21], [22].

In contrast to these methods, in this paper we first formulate the adversarial attack generation problem as one with a convex objective function but non-convex constraints. We then design an algorithm which iteratively solves a related convex problem. We prove that upon convergence of our iterative algorithm, the obtained solution is feasible for the original (non-convex) problem. We achieve 100% attack success rate on both the original undefended models and the adversarially-trained models. Our distortions of the L_∞ attack are respectively 31% and 18% lower than the C&W attack for the best case and average case on the CIFAR-10 data set.

II. Related Literature

A. Gradient Descent Based Attack Methods

L-BFGS Attack [7]: The L-BFGS attack is the first attack based on optimization. It aims to minimize the cross-entropy loss of the adversarial example and the target label, while minimizing the L_2 distortion of the adversarial example and original data.

FGM Attack [15] & IFGM Attack [8]: The fast gradient method (FGM) attack uses the gradient of the loss function to find the direction in which the intensity of pixels should be changed. It is an attack that is designed to be fast rather than to pursue low distortion in the original data. The iterative fast gradient method (IFGM) attack is a refinement of the FGM attack which takes multiple smaller steps instead of a single step on gradient descent.

C&W Attack [17]: Based on the basic ideas of L-BFGS attack, C&W attack design their own objective functions instead of cross-entropy loss, which help them achieve 100% attack successful rate. Besides on L_2 attack, C&W also design iterative methods for the L_0 and L_∞ attack, in which the objective functions are non-differentiable. C&W attack is state-of-the-art in the adversarial attacks on DNNs.

B. Related Work on Convex Programming and Mixed Integer Linear Programming

Robustness Certification of DNNs: Recently, convex optimization methods have been used to certify the robustness of DNNs rather than to generate adversarial attacks. Examples include the use of linear programming in [21], quadratic programming in [22], and semidefinite programming in [23].

Binarized Neural Networks Attack: The paper [24] presents a new method based on mixed integer linear programming to attack binarized neural networks. The generation of low distortion attacks on binarized neural networks is a non-convex problem, where the binary nature of the activation functions is responsible for the lack of convexity. The authors use the property that the output of every layer is composed of zeros and ones to translate the lack of convexity into binary constraints. This presents a special case in which the non-convex problem can be solved by mixed integer linear programming.

C. Representative Defense Method

Defensive Distillation [16]: Defensive distillation uses distillation for the purpose of improving the robustness of a neural network. In the defensive distillation method, we need to train a teacher network model at "high temperature" at first and then employ the teacher network to produce soft labels for the training data set. Later, the created soft labels are used to train a distilled model. Finally, we reduce the temperature to low values when we test the accuracy of the distilled model.

Adversarial Training [25]: In adversarial training, adversarial examples with correct labels are mixed into the training data set. The neural network is then retrained to increase its robustness.

III. Proposed Convex Programming Based Attack

A. Notation and Definitions

1) Distortion on Different Cases: In general, three different cases are considered in the measurement of distortion for targeted adversarial attacks:

- Best case: Select the target class that is the easiest to attack, which means distortion of the adversarial examples corresponding to this class is the lowest among all the incorrect classes.
- Average case: Calculate the average distortion of the adversarial examples on all the incorrect target classes.
- Worst case: Select the target class that is the most difficult to attack, which means distortion of the adversarial examples corresponding to this class is the highest among all the incorrect classes.

2) Box Constraints and Discretization: In an actual image, the intensity of each pixel is indicated by an integer between 0 and 255. In an optimization framework, however, we assume that pixel values belong to the convex interval [0,1]. This is known as a box constraint. Once the optimization problem is solved and a solution for the adversarial example is found, we revert back to an actual image by multiplying the entries of the solution by 255 and rounding to the closest integer. This process is known as discretization. It has been demonstrated in the literature that discretization rarely affects the success rate of an attack [17].

B. Problem Formulation

Consider an N-layer DNN, where the collection of weights and biases in the i-th layer are respectively denoted by $\mathbf{W}_i$ and $\mathbf{b}_i$, and that all the layers in the DNN are fully connected. Assume that $\mathbf{x}_0$ is a vector representation of an image in the test set, and that $\mathbf{x}$ is an adversarial example that we wish to generate. The example $\mathbf{x}$ has the property that it is a small perturbation of $\mathbf{x}_0$ but is classified as belonging to the incorrect target class t by the DNN.

The output of the first layer of the DNN to input $\mathbf{x}$ is

$$\mathbf{y}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1).$$

Here, $\mathbf{y}_1$ and $\mathbf{b}_1$ are vectors, and $\sigma(\cdot)$ is the non-linear activation function which acts elementwise on its vector argument. This function is generally chosen to be the ReLU function [26] in state-of-the-art DNNs, which is defined as

$$\sigma(\tau) = \begin{cases} \tau & \text{if } \tau \geq 0, \\ 0 & \text{if } \tau < 0. \end{cases}$$

In a DNN the output of one layer is the input to the next, and thus the output of the i-th layer for $i = 2, \ldots, N-1$ is

$$\mathbf{y}_i = \sigma(\mathbf{W}_i\mathbf{y}_{i-1} + \mathbf{b}_i).$$

The output before the softmax function (the collection of logits) is

$$\mathbf{z} = \mathbf{W}_N\mathbf{y}_{N-1} + \mathbf{b}_N.$$

The logits are input into the softmax function to calculate the scores of different classes. The class with the highest score will determine the classification made by the DNN. Since the softmax function is an increasing function, the class with the highest logit will achieve the highest score and become the classification result. For a targeted adversarial attack, the target class t should have the highest logit, which means

$$(\mathbf{z})_t = \max(\mathbf{z}),$$

where $(\mathbf{z})_t$ is the t-th element in the vector $\mathbf{z}$. The above equation can be equivalently rewritten as

$$\mathbf{z} \leq (\mathbf{z})_t \mathbb{1},$$

where $\mathbb{1}$ is the column vector of all ones. The above inequality ensures the success of the targeted attack. To ensure that $\mathbf{x}$ is an imperceptible perturbation of $\mathbf{x}_0$ we minimize the L_p distortion between the adversarial example and the original data. Namely, we minimize

$$\|\mathbf{x} - \mathbf{x}_0\|_p,$$

which is a convex function of $\mathbf{x}$ for $p \geq 1$. Also, to ensure the adversarial example yields a valid image we impose the constraint

$$0 \leq \mathbf{x} \leq \mathbb{1}.$$

We can now formulate the adversarial attack problem as

$$\begin{aligned} \underset{\mathbf{x},\ \mathbf{y}_i,\ \mathbf{z}}{\text{minimize}} \quad & \|\mathbf{x}-\mathbf{x}_0\|_p \\ \text{subject to} \quad & \mathbf{y}_1 = \sigma(\mathbf{W}_1\mathbf{x}+\mathbf{b}_1) \\ & \mathbf{y}_i = \sigma(\mathbf{W}_i\mathbf{y}_{i-1}+\mathbf{b}_i),\ i=2,\dots,N-1 \\ & \mathbf{z} = \mathbf{W}_N\mathbf{y}_{N-1}+\mathbf{b}_N \\ & \mathbf{z} \le \mathbf{z}_t\mathbb{1},\ 0 \le \mathbf{x} \le \mathbb{1}. \end{aligned} \tag{1}$$

This optimization problem has a convex objective and convex inequality constraints. However, $\sigma(\cdot)$ is a nonlinear function which renders the equality constraints, and therefore the optimization problem as a whole, non-convex.

C. Solving a Convex Relaxation of (1)

In this section, we propose an algorithm which iteratively solves a convex relaxation of (1) to obtain an approximate solution. This approximate solution is feasible in the sense that it satisfies all the constraints in (1).

Since $\sigma(\cdot)$ acts elementwise on its argument, we can consider the effect of $\sigma(\cdot)$ as an elementwise multiplication of the input vector with a binary vector $\mathbf{a}_i$ whose elements are zero/one based on the sign of the elements of the vectors $\mathbf{W}_1\mathbf{x}+\mathbf{b}_1$ and $\mathbf{W}_i\mathbf{y}_{i-1}+\mathbf{b}_i$,

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{a}_1 \circ (\mathbf{W}_1\mathbf{x}+\mathbf{b}_1), \\ \mathbf{y}_i &= \mathbf{a}_i \circ (\mathbf{W}_i\mathbf{y}_{i-1}+\mathbf{b}_i),\ i=2,\dots,N-1, \end{aligned}$$

where $\circ$ denotes elementwise vector multiplication.

Due to the dependence of $\mathbf{a}_i$ on the sign of $\mathbf{W}_i\mathbf{y}_{i-1}+\mathbf{b}_i$, the equality constraint $\mathbf{y}_i = \mathbf{a}_i \circ (\mathbf{W}_i\mathbf{y}_{i-1}+\mathbf{b}_i)$ is still non-convex. We break this dependence by using an iterative procedure in which the sign of $\mathbf{W}_i\mathbf{y}_{i-1}+\mathbf{b}_i$, computed from the solution of the previous iteration, is used to form $\mathbf{a}_i$ in the current iteration. Concretely, rather than solve the non-convex problem (1), we solve for $k=0,1,\dots,T$ the convex problem

$$\begin{aligned} \underset{\mathbf{x},\ \mathbf{y}_i,\ \mathbf{z}}{\text{minimize}} \quad & \|\mathbf{x}-\mathbf{x}_0\|_p + \lambda\|\mathbf{x}-\mathbf{x}^{(k)}\|_2 \\ \text{subject to} \quad & \mathbf{y}_1 = \mathbf{a}_1^{(k)} \circ (\mathbf{W}_1\mathbf{x}+\mathbf{b}_1) \\ & \mathbf{y}_i = \mathbf{a}_i^{(k)} \circ (\mathbf{W}_i\mathbf{y}_{i-1}+\mathbf{b}_i),\ i=2,\dots,N-1 \\ & \mathbf{z} = \mathbf{W}_N\mathbf{y}_{N-1}+\mathbf{b}_N \\ & \mathbf{z} \le \mathbf{z}_t\mathbb{1},\ 0 \le \mathbf{x} \le \mathbb{1}. \end{aligned} \tag{2}$$

We denote by $\mathbf{x}^{(k+1)}, \mathbf{y}_i^{(k+1)}, \mathbf{z}^{(k+1)}$ the solution of problem (2) at iteration k, and let $\mathbf{x}^{(0)} = \mathbf{x}_0$. We set the value of $\mathbf{a}_1^{(k)}$ according to

$$(\mathbf{a}_1^{(k)})_j = \begin{cases} 1 & \text{if } (\mathbf{W}_1\mathbf{x}^{(k)}+\mathbf{b}_1)_j \ge 0, \\ 0 & \text{if } (\mathbf{W}_1\mathbf{x}^{(k)}+\mathbf{b}_1)_j < 0, \end{cases} \tag{3}$$

where $(\mathbf{v})_j$ denotes the jth element of the vector $\mathbf{v}$ and j takes all values between one and the dimension of the vector $\mathbf{a}_1$. We employ a special procedure to compute the values of $\mathbf{a}_i^{(k)}$. Rather than use $\mathbf{y}_i^{(k)}$ from the previous iteration, we propagate forward through the layers the value $\mathbf{x}^{(k)}$ of $\mathbf{x}$ from the previous iteration and denote the resulting values by $\mathbf{y}_i^{[k]}$. To make this precise, we find $\mathbf{a}_1^{(k)}$ from (3) and set $\mathbf{y}_1^{[k]} = \mathbf{a}_1^{(k)} \circ (\mathbf{W}_1\mathbf{x}^{(k)}+\mathbf{b}_1)$. We then find $\mathbf{a}_2^{(k)}$ from

$$(\mathbf{a}_2^{(k)})_j = \begin{cases} 1 & \text{if } (\mathbf{W}_2\mathbf{y}_1^{[k]}+\mathbf{b}_2)_j \ge 0, \\ 0 & \text{if } (\mathbf{W}_2\mathbf{y}_1^{[k]}+\mathbf{b}_2)_j < 0, \end{cases}$$

and set $\mathbf{y}_2^{[k]} = \mathbf{a}_2^{(k)} \circ (\mathbf{W}_2\mathbf{y}_1^{[k]}+\mathbf{b}_2)$. We continue this procedure so that for $i=2,\dots,N-1$,

$$(\mathbf{a}_i^{(k)})_j = \begin{cases} 1 & \text{if } (\mathbf{W}_i\mathbf{y}_{i-1}^{[k]}+\mathbf{b}_i)_j \ge 0, \\ 0 & \text{if } (\mathbf{W}_i\mathbf{y}_{i-1}^{[k]}+\mathbf{b}_i)_j < 0. \end{cases} \tag{4}$$

We emphasize that problem (2) is convex and can therefore be solved efficiently using convex optimization tools.

This motivates Algorithm 1: We iteratively solve (2), using (3) and (4) to update $\mathbf{a}_i^{(k+1)}$, until the condition $\mathbf{a}_i^{(k+1)} = \mathbf{a}_i^{(k)}$, $i=1,\dots,N-1$ is satisfied. We initialize the algorithm by setting $\mathbf{x}^{(0)} = \mathbf{x}_0$.

Proposition III.1. *If for some k we have*

$$\mathbf{a}_i^{(k+1)} = \mathbf{a}_i^{(k)}, \quad i=1,\dots,N-1, \tag{5}$$

then the solution $\mathbf{x}$ of (2), denoted by $\mathbf{x}^{(k+1)}$, is a feasible solution of problem (1).

Proof. We need to demonstrate that when condition (5) holds, the optimal solution of the convex problem (2) satisfies all the constraints in problem (1).

Recall that $\mathbf{x}^{(k+1)}$, $\mathbf{y}_i^{(k+1)}$, $\mathbf{z}^{(k+1)}$ denote the solution of (2) at iteration k. Since the solution satisfies the constraints, in particular we have

$$\begin{aligned} \mathbf{y}_1^{(k+1)} &= \mathbf{a}_1^{(k)} \circ (\mathbf{W}_1\mathbf{x}^{(k+1)}+\mathbf{b}_1), \\ \mathbf{y}_i^{(k+1)} &= \mathbf{a}_i^{(k)} \circ (\mathbf{W}_i\mathbf{y}_{i-1}^{(k+1)}+\mathbf{b}_i),\ i=2,\dots,N-1. \end{aligned}$$

From $\mathbf{a}_i^{(k+1)} = \mathbf{a}_i^{(k)}$, we conclude that

$$\mathbf{y}_1^{(k+1)} = \mathbf{a}_1^{(k+1)} \circ (\mathbf{W}_1\mathbf{x}^{(k+1)}+\mathbf{b}_1), \tag{6}$$

$$\mathbf{y}_i^{(k+1)} = \mathbf{a}_i^{(k+1)} \circ (\mathbf{W}_i\mathbf{y}_{i-1}^{(k+1)}+\mathbf{b}_i),\ i=2,\dots,N-1. \tag{7}$$

According to (6) and the definition of $\mathbf{y}_1^{[k]}$ we can derive that $\mathbf{y}_1^{(k+1)} = \mathbf{y}_1^{[k+1]}$. Similarly, from (7), the definition of $\mathbf{y}_2^{[k]}$, and $\mathbf{y}_1^{(k+1)} = \mathbf{y}_1^{[k+1]}$, we obtain $\mathbf{y}_2^{(k+1)} = \mathbf{y}_2^{[k+1]}$. This procedure can be continued to show that $\mathbf{y}_i^{(k+1)} = \mathbf{y}_i^{[k+1]}$ for $i=1,2,\dots,N-1$. Therefore, equation (7) is equivalent to

$$\mathbf{y}_i^{(k+1)} = \mathbf{a}_i^{(k+1)} \circ (\mathbf{W}_i\mathbf{y}_{i-1}^{[k+1]}+\mathbf{b}_i),\ i=2,\dots,N-1. \tag{8}$$

Now, replacing k with $k+1$ in the definition of $\mathbf{a}_i^{(k)}$, equations (6) and (8) are respectively equivalent to

$$\begin{aligned} \mathbf{y}_1^{(k+1)} &= \sigma(\mathbf{W}_1\mathbf{x}^{(k+1)}+\mathbf{b}_1), \\ \mathbf{y}_i^{(k+1)} &= \sigma(\mathbf{W}_i\mathbf{y}_{i-1}^{(k+1)}+\mathbf{b}_i),\ i=2,\dots,N-1. \end{aligned}$$

Recalling that the constraints involving $\mathbf{z}$ are the same in (1) and (2), the above argument implies that $\mathbf{x}^{(k+1)}$, $\mathbf{y}_i^{(k+1)}$, $\mathbf{z}^{(k+1)}$ satisfy the constraints in (1) and therefore characterize a feasible point. This completes the proof of the proposition.

Algorithm 1 Find approximate solution of (1) by iteratively solving (2)

Input: image $\mathbf{x}_0$, weights $\mathbf{W}_i$, biases $\mathbf{b}_i$, parameter λ
Set $\mathbf{x}^{(0)} = \mathbf{x}_0$
Calculate $\mathbf{a}_i^{(0)}$ according to (3) and (4)
Set $k = 0$
for $k \leq T$ **do**
  Solve problem (2) to obtain $\mathbf{x}^{(k+1)}$
  Update $\mathbf{a}_i^{(k+1)}$ according to (3) and (4)
  if Condition (5) is satisfied **then**
    Break for loop
  end if
  Set $k = k + 1$
end for

Algorithm 2 Iterative method to guarantee convergence of Algorithm 1

Input: image $\mathbf{x}_0$, weights $\mathbf{W}_i$, biases $\mathbf{b}_i$
Set parameter λ
repeat
  Apply Algorithm 1
  if Condition (5) is not satisfied **then**
    Increase value of λ
  end if
until Condition (5) is satisfied
Set $\hat{\mathbf{x}}$ to solution of Algorithm 1
Set $\hat{\lambda} = \lambda$

The parameter λ characterizes the relative importance of the two terms in the objective function of (2): A small value of λ de-emphasizes the second norm, which results in a solution with lower distortion and therefore better performance; a large value of λ emphasizes the second norm, which helps achieve convergence (at the expense of performance) when (2) is solved iteratively by penalizing the difference of the optimal $\mathbf{x}$ between two consecutive iterations.

This motivates Algorithm 2: We choose a small value of λ and check the convergence of Algorithm 1. If convergence, as determined by the satisfaction of condition (5), is not achieved then we increase the value of λ and apply Algorithm 1 again; if convergence is achieved then we have found a value of λ that results in a feasible solution. The advantage of this process is that when λ is small, the optimization problem (2) is allowed to explore the $\mathbf{x}$-space for a solution with small distortion. Therefore, our aim is to find the smallest value of λ that results in convergence (in our experiments such a value of λ could always be found); we refer to this value as $\hat{\lambda}$, and refer to the solution of Algorithm 1 with $\lambda = \hat{\lambda}$ as $\hat{\mathbf{x}}$.

Remark: Once $\hat{\lambda}$ is obtained, we may explore whether solutions with lower distortion than $\hat{\mathbf{x}}$ can be found as follows: We start from $\lambda = \hat{\lambda}$ and apply Algorithm 1 with the important difference that rather than setting $\mathbf{x}^{(0)} = \mathbf{x}_0$ we take $\mathbf{x}^{(0)} = \hat{\mathbf{x}}$. We then iteratively reapply Algorithm 1, we decrease λ if convergence is achieved and otherwise increase λ, each time setting $\mathbf{x}^{(0)}$ to be the solution of Algorithm 1 from the previous iteration. In our experiments we find that applying this methods for several iterations usually helps us find a solution with lower distortion than just applying Algorithms 1 and 2. Moreover, the runtime for adjusting λ is acceptable and we will further discuss this in section IV-G.

IV. Performance Evaluation

We compare our proposed method with the IFGM attack [8] and the C&W attack [8], in which the C&W attack is state-of-the-art adversarial attack on DNNs. In the C&W attack, the authors proposed their method for L_0, L_2 and L_∞ attacks, since L_0 norm is non-convex, it is not applicable for convex programming. Thus we compare our L_2 and L_∞ attacks with other two works. Our experimental results demonstrate that the adversarial examples generated by our method have lower distortion than the IFGM attack and the C&W attack on the MNIST [1] and CIFAR-10 [27] data sets.

A. Experiment Setup

We evaluate the performance of different attack methods on the LeNet-300-100 [1]. In this network, the number of neurons in the two hidden layers are 300 and 100, respectively. The activation functions after the hidden layers are chosen to be ReLU. The test accuracy of LeNet-300-100 on the MNIST and CIFAR-10 data sets are around 98% and 57%, respectively. In our proposed algorithm, we solve the convex problem by CVXPY [28], [29], which is a tool for convex programming in Python.

B. Success Rate and Distortion for L_2 Attack

We test the L_2 attack of our proposed method, the IFGM attack method and the C&W attack method on the first 500 images in the test sets of the MNIST and CIFAR-10 data sets. For every image we implement targeted attacks on its 9 incorrect labels. In the 4500 adversarial attacks in each data set, both of the methods achieve 100% attack success rate (ASR), and the L_2 distortion of different attack methods on CIFAR-10 are shown in Table 1.

In both of the data sets, the performance of our method and the C&W attack are much better than the IFGM attack. In the MNIST data set, our results are close to the C&W attack. While in the larger data set CIFAR-10, we achieve lower distortion than the C&W attack on both of the three cases.

C. Success Rate and Distortion for L_∞ Attack

The data sets setup for the L_∞ attack test is the same as the L_2 attack. The results of different L_∞ attack methods are shown in Table 2.

On the L_∞ attack, we achieve a notable improvement compared with the results of the C&W attack. In the CIFAR-10 data set, we respectively reduce the L_∞ distortion by 31% and 18% for the best case and average case compared with the C&W attack on the L_∞ attack.

TABLE I
COMPARISON OF DIFFERENT L_2 ATTACKS FOR CIFAR-10 DATA SETS

Data Set	Attack Method	Best Case		Average Case		Worst Case	
		ASR	L_2	ASR	L_2	ASR	L_2
CIFAR-10	IFGM (L_2)	100	0.168	100	0.740	100	1.339
	C&W (L_2)	100	0.158	100	0.648	100	1.114
	Convex Programming (L_2)	**100**	**0.154**	**100**	**0.645**	**100**	**1.112**

TABLE II
COMPARISON OF DIFFERENT L_∞ ATTACKS FOR MNIST AND CIFAR-10 DATA SETS

Data Set	Attack Method	Best Case		Average Case		Worst Case	
		ASR	L_∞	ASR	L_∞	ASR	L_∞
MNIST	IFGM (L_∞)	100	0.081	100	0.134	100	0.197
	C&W (L_∞)	100	0.076	100	0.117	100	0.156
	Convex Programming (L_∞)	**100**	**0.074**	**100**	**0.114**	**100**	**0.152**
CIFAR-10	IFGM (L_∞)	100	0.0046	100	0.0198	100	0.0379
	C&W (L_∞)	100	0.0051	100	0.0181	100	0.0299
	Convex Programming (L_∞)	**100**	**0.0035**	**100**	**0.0149**	**100**	**0.0256**

D. Attack Against Defensive Distillation

We test our L_2 attack and L_∞ attack against the defensive distillation [16] on MNIST and CIFAR-10 data sets. In each data set, we use the first 500 images to implement 4500 attacks on the DNNs trained by the defensive distillation on different temperature. Both of our L_2 attack and L_∞ attack achieve 100% attack success rate on both of the data sets for all the different temperature set in the defensive distillation.

E. Attack Against Adversarial Training

We test the performance of our method under adversarial training [25] using data augmentation, where we add 4500 adversarial examples with correct labels into the training data set. For these adversarially-trained models, we find that the adversarial examples generated by our L_∞ method consistently have lower distortion compared with the adversarial examples generated by the C&W method. We elaborate on these results for the L_∞ case below.

In our first group of experiments, we generate adversarial examples, perform adversarial training, and attack adversarially-trained models, all using *the same* method (i.e., our convex programming method versus the C&W method), and then compare their distortions against each other. On MNIST, the average distortion of our method is 0.124, which is lower than 0.129 for the C&W method. On CIFAR-10, the average distortion of our method is 0.0231, which is 12% lower than 0.0262 for the C&W method.

In our second group of experiments, we implement adversarial training using adversarial examples generated by IFGM method, and then attack the adversarially-trained model using our method and the C&W method. On MNIST, the average distortion of our method is 0.126, which is lower than 0.132 for the C&W method. On CIFAR-10, the average distortion of our method is 0.0262, which is 11% lower than 0.0295 for the C&W method.

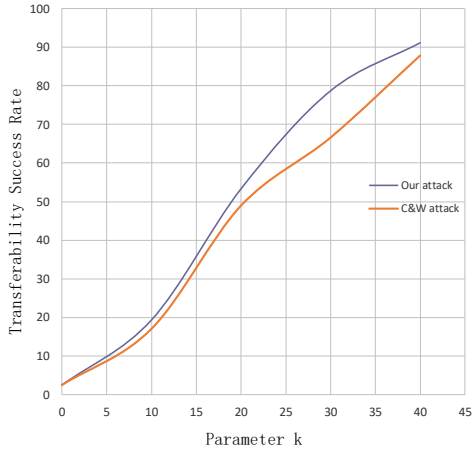

Fig. 1. Comparison of the transferability success rate for our L_2 attack and the C&W L_2 attack on MNIST data set

F. Transferability of Our Proposed Attack Method

To test the transferability of our proposed attack method, we change the constraint $\mathbf{z} \leq \mathbf{z}_t \mathbb{1}$ in problem (2) to

$$\mathbf{z} + k\mathbb{1} - k\mathbf{e}_t \leq \mathbf{z}_t \mathbb{1},$$

where $\mathbf{e}_t$ is a vector in which the tth element is one and the other elements are zeros. Here, we incorporate a nonnegative parameter k to control the confidence of the adversarial examples. In this case, the logit of the targeted class is higher than the logits of other classes by k or more. As k increases, the adversarial examples have higher transferability.

Transferability means that the adversarial examples of one DNN model could be transferred to be the adversarial examples of another DNN model [30]. In the transferability success rate test, we generate 900 adversarial examples using the first 100 images on the test set of MNIST by our L_2 attack and the C&W L_2 attack individually. And then we use the adversarial examples generated by each method to attack the model trained by the defensive distillation at temperature

$T = 100$. Transferability success rate is calculated by the sum of adversarial examples which can successfully achieve targeted attack on the model trained by the defensive distillation divided by 900 (the total number adversarial examples).

The result of the transferability success rate is shown in Figure 1, which demonstrates that our method achieves higher transferability success rate than the C&W attack. When the confidence $k = 0$, the transferability success rate is low for both of the methods, but it increases as k increases, the transferability success rate of our method exceeds 90% when $k = 40$.

G. Comparison of Runtime and Distortion

To generate adversarial examples with comparable distortion to the C&W attack, our method requires only a few iterations and the convex problem in every iteration can be efficiently solved.

Specifically, for L_∞ attacks on the CIFAR-10 data set, the generation of adversarial examples with comparable distortion to the C&W attacks consumed only one third the runtime of the C&W method. If we use the same runtime, our average distortion is 13% lower than that of the C&W method. Moreover, if we are allowed twice the runtime of C&W to adjust λ, we can generate adversarial examples with 18% lower distortion than the C&W attacks. We performed our experiments on a machine with an Intel I7-7700K CPU, 16 GB RAM and an NVIDIA GTX 1080 TI GPU.

V. Conclusions

In this paper we propose an innovative method for generating adversarial examples via convex programming. Our method achieves a 100% attack success rate on both the original undefended models and the adversarially-trained models. We also decrease the distortion (on both original undefended models and adversarially-trained models) and increase the transferability of adversarial examples compared with state-of-the-art attack methods.

VI. Acknowledgments

This research was supported by the National Science Foundation under awards CAREER CMMI-1750531, ECCS-1609916, CNS-1739748, and CNS-1704662.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[5] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 30–42, 2012.

[6] D. Andor, C. Alberti, D. Weiss, A. Severyn, A. Presta, K. Ganchev, S. Petrov, and M. Collins, "Globally normalized transition-based neural networks," *arXiv preprint arXiv:1603.06042*, 2016.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *ICLR*, 2013.

[8] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[9] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 427–436.

[10] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou, "Hidden voice commands." in *USENIX Security Symposium*, 2016, pp. 513–530.

[11] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[12] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of neural nets through robust optimization," *arXiv preprint arXiv:1511.05432*, 2015.

[13] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[14] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," *arXiv preprint arXiv:1803.01442*, 2018.

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *stat*, vol. 1050, p. 20, 2015.

[16] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.

[17] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[18] K. Xu, S. Liu, P. Zhao, P.-Y. Chen, H. Zhang, Q. Fan, D. Erdogmus, Y. Wang, and X. Lin, "Structured adversarial attack: Towards general implementation and better interpretability," in *International Conference on Learning Representations*, 2019.

[19] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] E. Wong, F. Schmidt, J. H. Metzen, and J. Z. Kolter, "Scaling provable adversarial defenses," in *Advances in Neural Information Processing Systems*, 2018, pp. 8400–8409.

[22] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in Neural Information Processing Systems*, 2018, pp. 4939–4948.

[23] A. Raghunathan, J. Steinhardt, and P. Liang, "Certified defenses against adversarial examples," in *International Conference on Learning Representations*, 2018.

[24] E. B. Khalil, A. Gupta, and B. Dilkina, "Combinatorial attacks on binarized neural networks," in *International Conference on Learning Representations*, 2019.

[25] F. Tramr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *International Conference on Learning Representations*, 2018.

[26] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[27] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[28] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[29] S. D. Akshay Agrawal, Robin Verschueren and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[30] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.