Self-adaptive Norm Update for Faster Gradient-based L_2 Adversarial Attacks and Defenses

Yanhong Liu and Fengming Cao
Pingan International Smart City, China

Keywords: Adversarial Attacks and Defenses, Computer Vision, Neural Networks, Deep Learning.

Abstract: *Adversarial training* has been shown as one of the most effective defense techniques against adversarial attacks. However, it is based on generating strong adversarial examples by attacks in each iteration of its training process. Research efforts have always been paid to reduce the time overhead of attacks, without impacting their efficiency. The recent work of *Decoupled Direction and Norm* (DDN) pushed forward the progress on the gradient-based L_2 attack with low norm, by adjusting the norm of the noise in each iteration based on whether the last perturbed image is adversarial or not. In this paper, we propose a *self-adaptive* way of adjusting the L_2 norm, by considering whether the perturbed images in the last two iterations are both adversarial or not. Experiments conducted on the MNIST, CIFAR-10 and ImageNet datasets show that our proposed attack achieves comparable or even better performance than DDN with up to 30% less number of iterations. Models trained with our attack achieve comparable robustness to those trained with the DDN attack on the MNIST and CIFAR-10 datasets, by taking around 20% less training time, when the attacks are limited to a maximum norm.

1 INTRODUCTION

After the emergence of deep learning techniques, it has received considerable attraction and applied in a wide range of computer vision tasks, such as classification (Szegedy et al., 2015; Simonyan and Zisserman, 2015; He et al., 2016) and object detection (Girshick, 2015; Liu et al., 2016; Redmonand et al., 2016), leading to state-of-the art performance. However, it has been shown that deep neural network models are vulnerable to adversarial examples (Biggio et al., 2013; Szegedy et al., 2013). For example, an image with small perturbations added is perceptually very similar to the original one, but misclassified with high confidence by the classifier.

Research in the literature usually measures the added noise by L_p norm ($p = 0, 1, 2, \infty$, etc), which gives a type of distance measure between the original image and the adversarial example. Each form of the L_p norm has its own preference of the distortion on the image to be attacked. Various attack techniques (Papernot et al., 2016; Modas et al., 2019; Chen et al., 2018; Goodfellow et al., 2015; Kurakin et al., 2016; Madry et al., 2018; Szegedy et al., 2013; Carlini and Wagner, 2017; Rony et al., 2019) have been devised to generate adversarial examples, with each one typically specific to a particular form of L_p norm. Among of these, the *projected gradient descent* (PGD) (Madry et al., 2018) method is well known for its effectiveness on L_∞ attack. The state-of-the-art for L_2 attack was proposed by Carlini and Wagner (Carlini and Wagner, 2017), known as the C&W attack.

As for the optimization strategy used, the attacks can be classified into two main branches. One is to find the perturbation within a specified norm ball around the original sample, while the perturbed example maximizes the loss function. PGD is the representative work along this line. The other is to search for the perturbation with the lowest norm, among all the possible distortions that mislead the classifier. The typical work along this line refers to C&W (Carlini and Wagner, 2017).

It has always been a widely active research area for devising *defense* methods (Prakash et al., 2018; Gu et al., 2019; Machado et al., 2019) to get robust models that can correctly label the adversarially perturbed images. One of the most effective defenses (Athalye et al., 2018) until now is the *adversarial training* approach, which arguments each minibatch of training data with adversarial examples. Usually, iterative attacks (Kurakin et al., 2017; Madry et al., 2018) are used to generate stronger adversarial exam-

Liu, Y. and Cao, F.
Self-adaptive Norm Update for Faster Gradient-based L2 Adversarial Attacks and Defenses.
DOI: 10.5220/0010186100150024
In *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2021)*, pages 15-24
ISBN: 978-989-758-486-2

ples for training, in order to get more robust models. The more attack iterations are used, the stronger examples are generated. However, it is prohibitively time-consuming since it needs to compute the gradient on the input in each attack iteration. Some research efforts (Shafahi et al., 2019; Wong et al., 2020; Zheng et al., 2020) have been paid to speed up the adversarial training process, which mainly consider about the widely adopted L_∞ attack PGD (Madry et al., 2018).

There is far rarely work on adversarial training based on L_2 attacks, especially for those searching for the perturbation with the minimum norm. The main reason is due to the lack of an efficient attack method. C&W requires a line-search for one of the optimization terms and thus needs thousands of iterations. The extremely high computational cost makes it impossible to apply it in adversarial training. To tackle with the difficulty of C&W attack and reduce the time cost, Rony et al. (Rony et al., 2019) proposed to *decouple the direction and the norm* (DDN) of the generated perturbation in each gradient-based iteration. Along the direction of the gradient, the norm of the perturbation is constrained by projecting it onto an ε-sphere around the original image. The value of ε is adjusted by a binary decision, based on whether the distorted image is adversarial or not. In this way, DDN generates adversarial examples closest to the decision boundary, and thus opens the possibility of using an efficient L_2 attack for adversarial training.

In this paper, we follow this line of work on speeding-up the L_2 attack. We observe that DDN algorithm adjusts the L_2 norm ε by a fixed update factor, which may slow down the progress of approaching the decision boundary. We propose to adjust the norm in a self-adaptive way in each gradient-based iteration, by tracing whether the perturbed images were adversarial or not in the previous two steps. We can change the norm to a greater degree, if the previous two perturbations led the same predictions. Otherwise, we change the norm less since the two perturbed images have crossed the decision boundary. We design a *self-adaptive norm update* (ANU) scheme for L_2 attack with low norm, which can approach to the minimum adversarial perturbation much faster than DDN. This improvement in time reduction helps to further speed up the adversarial training process using L_2 attack.

Extensive experiments on the MNIST, CIFAR-10 and ImageNet datasets show that the ANU algorithm generally has better performance than DDN when taking the same number of iterations, with comparable execution time. It was also observed that the ANU attack can achieve comparable or even better performance than DDN with up to 30% less number of iterations, which is especially useful under the scenario of adversarial training. Experiments show that the defense models trained with our attack may achieve comparable robustness with those trained with DDN on MNIST and CIFAR-10, with around 20% reduced training time.

2 RELATED WORK

In this section, we review the basic concept of adversarial examples, related work on attack and defense methods, and then give the background of our work.

2.1 Threat Model

In this paper, we consider the *white-box* attack scenarios, where the model architecture and parameters are known to the adversary. The *white-box* attacks can be used to evaluate the worst-case robustness of the model, or generate adversarial examples for adversarial training.

2.2 L_p Norm

We study the problem of image classification. Given an image sample x from the input space $\mathbb{R}^n$ (i.e. $x \in \mathbb{R}^n$), it has a true label y (the index of the label) from the predefined set of m possible labels $\mathbb{R}^m$. For example, a grey-scale image with $h \times w$ pixels defines a two-dimensional vector $x \in \mathbb{R}^{hw}$, where x_i denotes the intensity of the i-th pixel. The value of each pixel is scaled to be in the range $[0,1]$. Similarly, a color RGB image defines a three-dimensional vector $x \in \mathbb{R}^{3hw}$.

The image classifier, modeled by a neural network F parameterized by θ, outputs the probability of x belonging to y_i for each $y_i \in \mathbb{R}^m$, i.e. $F(x)$ is a list of probability $P(y_i|x,\theta)$. The sample x is then labelled as $\mathbb{C}(x) = \arg\max_i P(y_i|x,\theta)$.

An adversary attacks x with small perturbations added to the intensity of the image pixels, such that the resulted image is misclassified. Suppose that the distorted image is denoted by x', a successful attack will lead the classifier to have $\mathbb{C}(x')$ not equal to its true label y. To quantify the difference of x' to its original image x, the widely used distance metrics in the literature are L_p ($p = 0, 1, 2, \infty$, etc.) norms, which give a reasonable approximation of human perceptual similarity. Let δ denote the perturbation added to x, i.e. $\delta = x' - x$. The L_p distance between x and x',

denoted by $\|\delta\|_p$, is defined as follows:

$$\|\delta\|_p = \left(\sum_{i=1}^{n} |\delta|^p\right)^{\frac{1}{p}} \quad (1)$$

Intuitively, L_0 distance measures the number of pixels that are altered in x'. L_1 attacks lead to sparsity in the perturbation, with only a few pixels adjusted. L_2 distance measures the standard Euclidean distance between x and x'. L_2 attacks lead to noise that are more localized in the image, since it can trade off a large perturbation in some pixel for less perturbations in other pixels. L_∞ distance measures the maximum change for all the pixel values, and thus L_∞ attacks lead to small noise everywhere in the image.

2.3 Optimization Objective

The attacks can be grouped into *non-targeted* and *targeted* ones. For the non-targeted ones, an adversarial example x' leads to a misclassified label $\mathbb{C}(x') \neq y$. The targeted attacks assume a predefined label $y_{target} \neq y$, while leading the classifier to predict x' to be y_{target}.

Taking the non-targeted attacks as an example, we show two main approaches of optimization over generating the adversarial examples. The first approach is to minimize the norm of the distortion δ under the constraint that $x+\delta$ misleads the classifier, which can be formulated as:

$$\begin{aligned} &\min_{\delta} \|\delta\|_p \\ \text{such that} \quad &\mathbb{C}(x+\delta) \neq y, \\ \text{and} \quad &x+\delta \in [0,1]^n \end{aligned} \quad (2)$$

For targeted attacks, a similar formulation can be obtained by modifying the constraint to be $\mathbb{C}(x+\delta) = y_{target}$

The other approach is to maximize the loss function $J(x+\delta, y, \theta)$ (equivalent to minimize the probability of being correctly classified), given that the norm of the perturbation is bounded by ε, which can be formulated as:

$$\begin{aligned} &\max_{\delta} J(x+\delta, y, \theta) \\ \text{such that} \quad &\|\delta\|_p \leq \varepsilon, \\ \text{and} \quad &x+\delta \in [0,1]^n \end{aligned} \quad (3)$$

A similar formulation can be derived for targeted attacks, by modifying the objective to be $\min_{\delta} J(x+\delta, y_{target}, \theta)$ (equivalent to $\max_{\delta} P(y_{target}|x+\delta, \theta)$).

2.4 Attacks

Since the discovery of the phenomena of adversarial examples, various gradient-based attack techniques have been proposed to generate examples to evaluate the robustness of deep neural network models, based on different L_p norm specifications.

L_0/L_1: The JSMA (Papernot et al., 2016) L_0 attack constructs adversarial saliency maps through computing *forward derivatives*, and thus identifies the input pixels with the highest potential to change the decision of the classifier. The SparseFool (Modas et al., 2019) method exploits the low mean curvature of the decision boundary, and proposes a geometry-inspired sparse L_0/L_1 attack that controls the sparsity of the perturbations. The EAD (Chen et al., 2018) work formulates the process of attacking the neural networks as an elastic-net regularized optimization problem and features L_1-oriented adversarial examples.

L_∞: The *fast gradient sign method* (FGSM) (Goodfellow et al., 2015) generates adversarial examples with a single gradient step, which maximizes the loss function with the norm of the perturbations upper bounded, following Eqn. (3). The basic iterative method (Kurakin et al., 2016) applies multiple and smaller FGSM steps, which was further strengthened by adding multiple random restarts. These enhancements were later well known as the PGD attack (Madry et al., 2018), which is the state of the art method for L_∞ attack. There have been some improvements on PGD that aim at making it more effective and/or more query-efficient by changing its update rule to Adam (Uesato et al., 2018) or momentum (Dong et al., 2018).

L_2: FGSM and PGD can also be extended to generate L_2 attacks, with the objective of optimizing Eqn. (3). Following the constrained minimization objective as stated in Eqn. (2), Szegedy et al. (Szegedy et al., 2013) reformulated the problem in the following form that is better suited for optimization:

$$\begin{aligned} &\min_{\delta} c \cdot \|\delta\|_2^2 + \log P(y|x+\delta, \theta) \\ \text{such that} \quad &x+\delta \in [0,1]^n \end{aligned} \quad (4)$$

where c is a constant. A box-constrained optimizer (*L-BFGS*) is used to address the constraint of $x+\delta \in [0,1]^n$ and line search is performed to find an appropriate value of c.

DeepFool (Moosavi-Dezfooli et al., 2016) iteratively constructs untargeted adversarial examples by assuming a linear approximation of the neural network models. C&W attack (Carlini and Wagner, 2017) reformulates the optimization problem in Eqn. (2) in a similar way to L-BFGS, as shown in Eqn. (5). Instead of using the box-constrained optimization, it proposes to change the variables using the tanh function and thus address the constraint of $x+\delta \in [0,1]^n$ naturally. It also tries to optimize the difference between *logits* (the model output before the softmax ac-

tivation), instead of the loss function.

$$\begin{aligned} &\min_{\delta} \|x' - x\|_2^2 + c \cdot f(x') \\ \text{where} \quad &f(x') = \max(Z(x')_y - \\ &\max\{Z(x')_i : i \neq y\}, -\kappa) \end{aligned} \tag{5}$$

where x' is equal to $\frac{1}{2}\tanh(arctanh(2x-1)+\delta)+1)$. $Z(x')_i$ denotes the *logit* corresponding to the i-th class. κ denotes a confidence parameter, a higher value of which results in higher confidence of misclassifying the adversarial sample x'.

C&W shows the state-of-the-art performance for L_2 attack. However, it has to iteratively search for the appropriate value of constant c. The prohibitively high computational cost makes it not suitable for generating adversarial examples for training defense models.

The DDN attack (Rony et al., 2019) makes a great progress along the way of being time efficient (as few as 100 iterations), with comparable results to C&W. We will detail this algorithm in the next section.

2.5 Defenses

Adversarial training has been shown as one of the most effective defense methods to train robust models against adversarial examples (Athalye et al., 2018). It mixes the clean train data with adversarial examples, where the adversarial loss function becomes:

$$J^*(x,y,\theta) = \beta J(x,y,\theta) + (1-\beta)J(x',y,\theta) \tag{6}$$

where β specifies a constant ratio. $J(x,y,\theta)$ defines the normal loss function of the model, such as the cross-entropy loss. The adversarial example x' of the original image x may be generated by the attacks as stated in the last subsection. When it is generated following the objective of Eqn. (3), the training process becomes a *min-max* optimization problem as formulated by Madry's defense. Most defenses focus on this line of work due to the availability of efficient attack mechanisms such as FGSM (Goodfellow et al., 2015) or PGD (Madry et al., 2018) etc, where the attacks aim to maximize the worst-case loss given that the perturbations are bounded by a maximum norm ball.

However, the research on attacks with the objective of Eqn. (2) lacks behind on the aspect of time efficiency. The recently proposed DDN (Rony et al., 2019) attack moves a big step forward, which makes it possible to train defense models following this line.

3 THE ANU ATTACK

In this section, we first analyze the DDN algorithm (Rony et al., 2019) in detail, and then propose a self-adaptive norm update scheme to further improve the time efficiency of L_2 attack.

3.1 Decoupled Direction and Norm

As shown in Alg. 1 (step 4 and 5), the DDN (Rony et al., 2019) attack iteratively refines the noise δ_k, based on the gradient (denoted by g) of the loss function $J(x'_{k-1}, y, \theta)$ relative to the noise δ_{k-1} computed in the last iteration. The noise δ_k is updated along the direction of g, with the goal of increasing the loss for the untargeted attack while decreasing the loss for the targeted attack.

In each iteration, the DDN algorithm constrains the norm of the perturbation by projecting δ_k on an ε-sphere around the original image x (step 11). The PGD attack also projects the noise on a pre-specified ε-sphere, where ε is the maximal norm ball allowed (see Eqn. (3)). However, DDN adapts the value of ε in each iteration to make the perturbed image closer to the decision boundary.

In each iteration k, as shown from step 6 to 10, the ε-sphere is updated based on if x'_{k-1} is adversarial or not. If x'_{k-1} is not adversarial, i.e. $\mathbb{C}(x'_{k-1}) = y$, the norm ε_k is increased to be $(1+\gamma)\varepsilon_{k-1}$. Otherwise, if x'_{k-1} is adversarial, i.e. $\mathbb{C}(x'_{k-1}) \neq y$, the norm ε_k is decreased to be $(1-\gamma)\varepsilon_{k-1}$. It can be seen that ε_k is changed to improve the probability of making x'_k cross the decision boundary.

Note that DDN takes a fixed factor of γ for updating the norm in each iteration. We argue that this fixed ratio of scaling for ε slow down the convergence of the adversarial norm to the decision boundary. As illustrated in Figure 1, the noise δ_k is updated along the direction of g, and then projected back onto the ε_k-sphere around the original image x. Figure 1 (a) shows that the norm is scaled up with a fixed ratio of $1+\gamma$, when both x'_{k-2} and x'_{k-1} are not adversarial. In the case that x'_{k-2} is not adversarial and x'_{k-1} is adversarial, Figure 1 (b) shows that the norm is scaled up to $(1+\gamma)\varepsilon_{k-2}$ and then reduced back to $(1-\gamma)\varepsilon_{k-1} = (1-\gamma^2)\varepsilon_{k-2}$ that is even smaller than ε_{k-2}.

3.2 Self-adaptive Norm Update

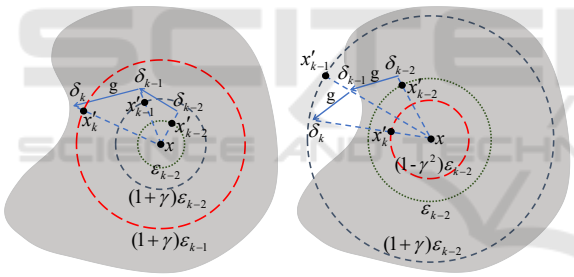
Instead of using a fixed factor of γ, in this paper we propose a self-adaptive scheme of adjusting the norm update factor γ_k in each iteration, as shown in Alg. 2.

With an initial input of γ_0, we update the value of γ_k by observing whether the past two perturbed images are adversarial or not. As shown from step 7 to 13, if x'_{k-2} and x'_{k-1} are both adversarial or not, we can say that these two distorted images did not cross

Algorithm 1: Algorithm of DDN Attack.

Input: x: original image to be attacked
Input: y: true label (untargeted) or targeted label (targeted)
Input: K: number of iterations
Input: α: step size in the direction of g
Input: γ: factor to update the norm
Output: x': adversarial image

1 Initialize $\delta_0 \leftarrow 0$, $x'_0 \leftarrow x$, $\varepsilon_0 \leftarrow 1$;
2 If targeted attack: $m \leftarrow -1$ else $m \leftarrow +1$;
3 **for** $k \leftarrow 1$ *to* K **do**
4 $\quad g \leftarrow m\nabla_{\delta_{k-1}} J(x'_{k-1}, y, \theta)$;
5 $\quad \delta_k \leftarrow \delta_{k-1} + \alpha \frac{g}{\|g\|_2}$;
6 $\quad$ **if** x'_{k-1} *is adversarial* **then**
7 $\quad\quad \varepsilon_k \leftarrow (1-\gamma)\varepsilon_{k-1}$ // decrease norm
8 $\quad$ **else**
9 $\quad\quad \varepsilon_k \leftarrow (1+\gamma)\varepsilon_{k-1}$ // increase norm
10 $\quad$ **end**
11 $\quad x'_k \leftarrow x + \varepsilon_k \frac{\delta_k}{\|\delta_k\|_2}$ // projection
12 $\quad x'_k \leftarrow clip(x'_k, 0, 1)$ // make $x'_k \in [0,1]^n$
13 **end**
14 Return x'_k that has lowest norm $\|x'_k - x\|_2$ and is adversarial;



(a) (b)

Figure 1: Update of ε_k by DDN in the case that (a) both x'_{k-2} and x'_{k-1} are not adversarial, (b) x'_{k-2} is not adversarial and x'_{k-1} is adversarial.

the decision boundary, since the variation of the norm ε_{k-1} was less than expected. Hence, we propose to update the norm ε_k with a larger factor γ_k, in order to make x'_k cross the decision boundary with a higher probability. We do this by scaling γ_{k-1} with a ratio of $1+\alpha_\gamma$ (see step 10).

On the other hand, if x'_{k-2} is non-adversarial and x'_{k-1} is adversarial or vice visa, we reduce the variation of the norm ε_k by adjusting the factor γ_k to a smaller value (see step 12). In this way, we let x'_k approach closer to the decision boundary with a higher probability.

In the rest of the algorithm, the norm ε_k is modified with the self-adaptive factor γ_k. With the proposed algorithm, we expect that the adversarial image with a comparable minimum L_2 norm can be obtained with less number of iterations than DDN. Note that our algorithm degrades to DDN when α_γ is set to be zero.

Algorithm 2: Algorithm of Self-Adaptive Norm Update Attack.

Input: x: original image to be attacked
Input: y: true label (untargeted) or targeted label (targeted)
Input: K: number of iterations
Input: α: step size in the direction of g
Input: γ_0: initial norm update factor
Input: α_γ: step size for updating γ_k with cosine annealing scheduling
Output: x': adversarial image

1 Initialize $\delta_0 \leftarrow 0$, $x'_0 \leftarrow x$, $\varepsilon_0 \leftarrow 1$;
2 Initialize $\mathcal{A}_{-1} \leftarrow 0$ // [ANU]
3 If targeted attack: $m \leftarrow -1$ else $m \leftarrow +1$;
4 **for** $k \leftarrow 1$ *to* K **do**
5 $\quad g \leftarrow m\nabla_{\delta_{k-1}} J(x'_{k-1}, y, \theta)$;
6 $\quad \delta_k \leftarrow \delta_{k-1} + \alpha \frac{g}{\|g\|_2}$;
$\quad$ /* start for updating γ_k */
7 $\quad \mathcal{A}_2 \leftarrow \mathcal{A}_1$;
8 $\quad \mathcal{A}_1 \leftarrow$ Bool($x'_{k-1}$is adversarial);
9 $\quad$ **if** $\mathcal{A}_1 = \mathcal{A}_2$ **then**
10 $\quad\quad \gamma_k \leftarrow (1+\alpha_\gamma)\gamma_{k-1}$ // increase γ_k
11 $\quad$ **else**
12 $\quad\quad \gamma_k \leftarrow (1-\alpha_\gamma)\gamma_{k-1}$ // decrease γ_k
13 $\quad$ **end**
$\quad$ /* end for updating γ_k */
14 $\quad$ **if** x'_{k-1} *is adversarial* **then**
15 $\quad\quad \varepsilon_k \leftarrow (1-\gamma_k)\varepsilon_{k-1}$ // [ANU]
16 $\quad$ **else**
17 $\quad\quad \varepsilon_k \leftarrow (1+\gamma_k)\varepsilon_{k-1}$ // [ANU]
18 $\quad$ **end**
19 $\quad x'_k \leftarrow x + \varepsilon_k \frac{\delta_k}{\|\delta_k\|_2}$ // projection
20 $\quad x'_k \leftarrow clip(x'_k, 0, 1)$ // make $x'_k \in [0,1]^n$
21 **end**
22 Return x'_k that has lowest norm $\|x'_k - x\|_2$ and is adversarial;

Suppose that γ_{k-1} is equal to γ, Figure 2 (a) shows that ANU scales the norm ε_k to a larger value (compared to Figure 1 (a)) with a greater factor γ_k than γ, based on the observation that both ε_{k-2} and ε_{k-1} are non-adversarial. In other words, x'_k has a higher probability of being adversarial. In the case that x'_{k-2} is not adversarial and x'_{k-1} is adversarial, Figure 2 (b) shows that the norm ε_k reduces back to $(1-\gamma_k)(1+\gamma_{k-1})\varepsilon_{k-2}$, which is supposed to be greater than that computed by DDN (as shown in Figure 1 (b)), since γ_k is smaller than γ_{k-1}. It is equal to say that the newly

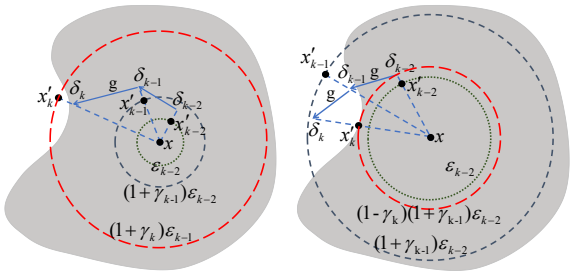


Figure 2: Update of ε_k by ANU in the case that (a) both x'_{k-2} and x'_{k-1} are not adversarial, (b) x'_{k-2} is not adversarial and x'_{k-1} is adversarial.

computed x'_k has a higher probability of being adversarial under the condition that ε_k is less than ε_{k-1}.

4 ADVERSARIAL TRAINING WITH ANU

As having been described in Section 2.5, adversarial examples generated by the attacks can be used to augment the training set. The trained model is thus expected to be robust to adversarial attacks. In this paper, we mainly focus on the robustness of the models against the attacks, hence we simplify the Eqn. (6) to consider only the adversarial examples:

$$J^*(x,y,\theta) = J(x',y,\theta) \tag{7}$$

where x' is an adversarial example of x produced by the ANU algorithm, which is projected to be within an ε-sphere around x. In other words, adversarial examples used for training are constrained to have a maximum norm of ε.

Due to the computational efficiency of ANU, we expect that the robust models based on the ANU attacks can be trained with considerably less time, with comparable robustness to that based on the DDN attacks.

5 EXPERIMENTAL EVALUATION

We have conducted extensive experiments on the MNIST, CIFAR-10 and ImageNet datasets, showing the effectiveness of our proposed ANU algorithm compared to the state-of-the-art L_2 attack DDN. We used the same model architectures for training as in (Rony et al., 2019). The base image classifiers, trained with 50 epochs, have 99.41% and 85.64% accuracy on the test sets of MNIST and CIFAR-10, respectively. A pre-trained Inception V3 model (Szegedy et al., 2016) was used for the ImageNet experiments, which takes cropped images of size 299×299. All the images were normalized in the $[0,1]$ range.

For the experiments on DDN, the hyper-parameters were initialized in the same way as (Rony et al., 2019), i.e. $\varepsilon_0 = 1$ and $\gamma = 0.05$. The initial step size α (in the direction of gradient g) was set to 1, which was reduced with cosine annealing to 0.01 in the last iteration. The norm update factor γ was chosen to be 0.05, since the norm of the adversarial perturbation at the best case can be reduced to $\varepsilon_0(1-\gamma)^K$ after K iterations. For $K >= 100$ iterations, $\gamma = 0.05$ is enough for the algorithm to find an adversarial example with the smallest possible perturbation (change one pixel by 1/255 for images encoded in 8 bit values) if it exists. The details of the derivation can be referred to (Rony et al., 2019). For the experiments on ANU, the step size α was set in the same way as DDN. The step size α_γ (for updating the norm update factor γ_k) was also set with cosine annealing.

In this section, we first verify the basic idea of ANU with a small example, illustrating its improvement on the process of norm update. Then we compare the performance results of ANU and DDN. Finally, we evaluate the robustness of the defense models trained with these two algorithms respectively.

5.1 Illustration

We first conducted the two gradient-based attacks with 100 iterations on the first image from the MNIST test dataset, using the base classifier for MNIST. Figure 3 shows the minimum norm ε_k that has been achieved ever for those iterations where the perturbed image x'_k is adversarial, as it iterates from 1 to 100. Suppose that $\mathcal{S}$ denote the set of all iterations where the perturbed images are adversarial, i.e. $\mathcal{S} = \{i | x'_i \text{ is adversarial}, 1 \le i \le K\}$. Given the index k of such an iteration, we have $\varepsilon_k = \min\{\varepsilon_i | i \in \mathcal{S} \text{ and } i \le k\}$. It can be observed that the DDN attack gradually reduces the norm of the adversarial noise at a steady step after learning the first adversarial image. The ANU attack first continues to increase the norm update factor γ_i and learns the first adversarial image much earlier than DDN, and then changes γ_i at a self-adjusted pace. We observe that ANU can arrive at a steady solution of the lowest adversarial norm ε_k much faster than DDN.

We also conducted a similar experiment on the first 1000 images from the MNIST test dataset, with 1000 iterations of attacks. Figure 4 also shows that in the beginning the ANU attack aggressively increases the norm update factors to find adversarial images,

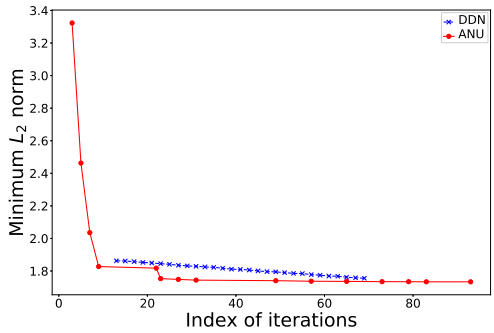


Figure 3: Change of the minimum L_2 norm achieved for the adversarial images during the attack of one MNIST sample for 100 iterations.

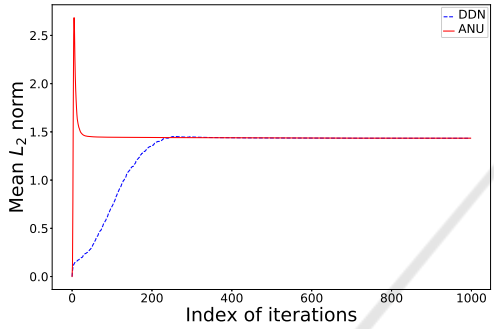


Figure 4: Change of the mean L_2 norm for the perturbed images during the attack of 1000 MNIST samples for 1000 iterations.

and then adaptively adjusts the update factors with reduced mean L_2 norm. It further verifies that the ANU algorithm approaches to the neighbor of the minimum adversarial norm much faster than DDN.

5.2 Attack Evaluation

For a complete comparison of our proposed ANU algorithm with DDN, we conducted two sets of experiments: untargeted attacks and targeted attacks. We generated attacks for the first 1000 images from the test datasets of MNIST and CIFAR-10 respectively, using the base image classifiers as described earlier. For ImageNet, we randomly chose one image for each of the 1000 classes from the validation set.

Table 1 shows the results for the untargeted attacks, including the number of iterations (budget) taken, the mean and median L_2 norms of all attacks as well as the execution times that were reported on an NVIDIA GTX 2080 Ti with 11GB of memory. Note that all the images were successfully attacked. For simplicity, we used a common set of hyperparameters for ANU. We set γ_0 to be 0.4, 0.2, 0.1 and initialized α_γ to be 0.15, 0.1, 0.08 for MNIST, CIFAR-10 and ImageNet respectively. Note that it is possible to get

Table 1: Performance comparison of ANU and DDN for the untargeted attacks on MNIST, CIFAR-10 and ImageNet.

	Attack	Budget	Mean L_2	Median L_2	time (s)
MNIST	DDN	50	1.5095	1.4738	1.0
		70	1.4610	1.4428	1.1
		100	1.4470	1.4400	1.4
		200	1.4411	1.4358	2.5
		500	1.4362	1.4334	6.2
		1000	1.4346	1.4328	12.2
	ANU	50	1.4572	1.4429	0.8
		70	**1.4457**	**1.4336**	**1.0**
		100	1.4425	1.4311	1.4
		200	1.4363	1.4310	2.8
		500	1.4362	1.4297	6.3
		1000	1.4347	1.4297	12.4
CIFAR-10	DDN	50	0.1900	0.1657	1.9
		70	0.1709	0.1536	2.6
		100	0.1676	0.1521	3.5
		200	0.1665	0.1511	6.9
		500	0.1659	0.1510	17.6
		1000	0.1655	0.1502	35.1
	ANU	50	0.1679	0.1519	1.8
		70	**0.1669**	**0.1511**	**2.5**
		100	0.1662	0.1509	3.5
		200	0.1658	0.1505	6.8
		500	0.1656	0.1500	17.1
		1000	0.1655	0.1498	35.1
ImageNet	DDN	50	0.5116	0.4937	220.9
		70	0.4862	0.4853	311.4
		100	0.4769	0.4805	435.3
		200	0.4674	0.4738	872.2
		500	0.4576	0.4704	2187.9
		1000	0.4497	0.4686	4365.1
	ANU	50	0.4809	0.4836	219.6
		70	**0.4717**	**0.4801**	**307.3**
		100	0.4667	0.4772	445.6
		200	0.4597	0.4743	876.5
		500	0.4529	0.4717	2197.6
		1000	0.4469	0.4700	4384.5

better performance for ANU by tuning these hyperparameters.

It can be observed that the mean/media norm achieved for both DDN and ANU attacks decreases as the budget increases, with more execution time. With the same budget, the ANU attack generally achieves less mean/median norm, with comparable run time. More interestingly, the ANU attack with less number of iterations can achieve comparable or better performance than DDN. For example, the ANU attack with 70 iterations achieves less norms than DDN with 100 iterations, with around 30% reduced time.

We also conducted the targeted attacks using both the ANU and DDN algorithms. We set both γ_0 and the initial value of α_γ to be 0.05 for MNIST and Im-

ageNet. For CIFAR-10, we set γ_0 to be 0.15 and initialized α_γ to be 0.06.

Using all 9 possible classes (except for the true label) as the target, we generated 9 attacks for each test image of MNIST and CIFAR-10. We generated attacks against 20 randomly chosen classes for each test image of ImageNet. Results are reported in Table 2 for the totally 9000 attacks on MNIST and CIFAR-10 respectively, and 20000 attacks on ImageNet. Similar conclusions can be obtained as the untargeted attacks. Note that the ANU attack has no absolute advantage over DDN for the case of 1000 iterations. It may be explained by the reason that with large number of iterations (e.g., greater than or equal to 1000), it provides enough time for DDN to cover the neighborhood of the decision boundary, as shown in Figure 4. However, in this paper we are more interested in the attacks with less number of iterations. Notably, the ANU attack with less number of iterations (85, 80 and 87 on MNIST, CIFAR-10 and ImageNet respectively) achieves better results than the DDN attack with 100 iterations, with reduced execution time (by about 15%, 20% and 12% on MNIST, CIFAR-10 and ImageNet respectively).

5.3 Defense Evaluation

To conduct a fair comparison to DDN, we used the same architectures as (Rony et al., 2019) for training robust models. A small CNN architecture was used for MNIST (Rony et al., 2019; Carlini and Wagner, 2017). A Wide ResNet (Zagoruyko and Komodakis, 2016) with 28 layers and widening factor of 10 (WRN-28-10) was used for CIFAR-10. For each step of adversarial training, we attacked the image with DDN with a budget of 100 iterations. When trained with ANU, we attacked the image with less number of iterations. The norm of the perturbations is limited to a maximum $\varepsilon = 2.4$ on the MNIST experiments, and $\varepsilon = 1$ on the CIFAR-10 experiments. The detailed settings can be referred to (Rony et al., 2019).

We trained a robust MNIST model with ANU and set the number of iterations to 70 for generating each adversarial example, which has a test accuracy of 98.88% on the clean samples, with the total training time of 3857.8 seconds. We also trained a robust MNIST model with DDN as the internal attack for generating the adversarial examples, which has an accuracy of 98.96%, with the total training time of 4840.4 seconds.

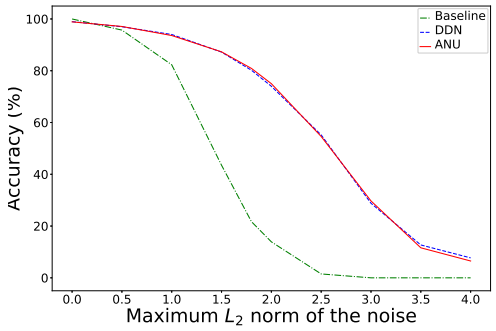
[1]We report the times on ImageNet with the original ones divided by 20.

Table 2: Performance comparison of ANU and DDN for the targeted attacks on MNIST, CIFAR-10 and ImageNet.

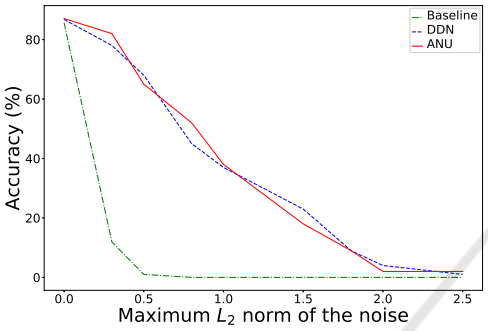
	Attack	Budget	Mean L_2	Median L_2	time (s)
MNIST	DDN	50	2.1069	2.0952	5.6
		85	2.0527	2.0317	9.5
		100	2.0453	2.0311	11.2
		300	2.0283	2.0191	34.8
		500	2.0258	2.0174	60.2
		1000	2.0229	2.0149	117.3
	ANU	50	2.0847	2.0705	5.6
		85	**2.0447**	**2.0264**	**9.6**
		100	2.0391	2.0255	11.2
		300	2.0273	2.0179	36.4
		500	2.0258	2.0172	60.8
		1000	2.0239	2.0156	118.9
CIFAR-10	DDN	50	0.3510	0.3290	15.0
		80	0.3409	0.3233	24.0
		100	0.3394	0.3209	30.5
		300	0.3359	0.3187	95.3
		500	0.3350	0.3176	157.6
		1000	0.3340	0.3163	324.5
	ANU	50	0.3433	0.3257	15.1
		80	**0.3392**	0.3214	**24.7**
		100	0.3381	0.3209	31.3
		300	0.3357	0.3179	95.6
		500	0.3349	0.3170	158.5
		1000	0.3344	0.3172	329.5
ImageNet	DDN	80	0.8194	0.7756	245.5[1]
		87	0.7986	0.7556	266.8
		100	0.7737	0.7337	305.1
		300	0.6677	0.6305	921.9
		500	0.6446	0.6066	1532.2
		1000	0.6234	0.5897	3059.5
	ANU	80	0.7881	0.7424	245.8
		87	**0.7718**	**0.7257**	**268.3**
		100	0.7545	0.7107	306.0
		300	0.6651	0.6264	924.3
		500	0.6439	0.6056	1537.1
		1000	0.6229	0.5888	3071.1

We trained the robust CIFAR-10 models with these two attacks. The model trained with ANU by setting the number of iterations to 80, has a test accuracy of 87.08%, with the training time of 36.1 hours. The CIFAR-10 model trained with DDN has an accuracy of 86.76%, with the training time of 44.2 hours. It can be concluded that the ANU attack-based robust models with less budget can achieve comparable test accuracy on the clean samples as the models trained with the DDN attack, with around 20% less training time.

We also ran multiple attacks to the three types of models on MNIST and CIFAR-10 respectively: the baseline (the base image classifier as described in the beginning of this section), the DDN-attack based and



(a)



(b)

Figure 5: Models robustness on (a) MNIST and (b) CIFAR-10, as we increase the maximum allowed L_2 norm for the attack.

the ANU-attack based models, with different values of the maximum allowed L_2 norm for the perturbations. As shown in Figure 5, the accuracy of the three models decreases as the maximum allowed norm ε increases. However, the robustly trained models are much slower in decrease of accuracy. It is worth noting that the models trained with ANU achieve comparable robustness to those trained with DDN, although it takes less budget and reduces the training time by around 20% for them.

Figure 6 shows some adversarial examples generated by the ANU attack with 1000 iterations for the above different models. We chose such an attack since it is stronger for evaluating the model robustness compared to those with less number of iterations. It can be observed that adversarial examples for the baseline model have the smallest L_2 norm, while the largest perturbations are needed to successfully attack the model trained with ANU.

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented the *Self-Adaptive Norm Update* gradient-based L_2 attack, which learns to

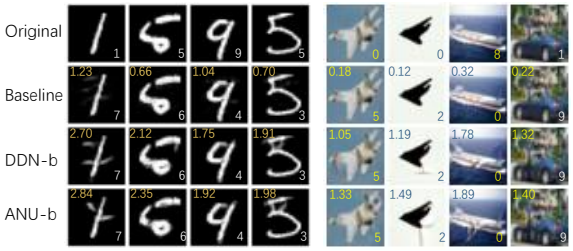


Figure 6: Adversarial examples against three models: baseline, DDN-based, ANU-based defenses. Text on top of each image indicates L_2 norm of the noise $\|\delta\|_2$. Text on bottom indicates the predicted class[2].

scale the norm update factor in a self-adaptive way in each iteration. In comparison to the state-of-the-art *Decoupled Direction and Norm* L_2 attack, our algorithm achieves comparable or even better performance with fewer iterations. The proposed attack can then be used to speed up the process of adversarial training, where the attack is used to generate adversarial examples close to the decision boundary. The experiments with the MNIST and CIFAR-10 datasets show that the robust models trained with our attack have comparable robustness to those based on DDN, while taking about 20% less training time.

Some techniques have been proposed in the literature (Shafahi et al., 2019; Wong et al., 2020; Zheng et al., 2020) to speed up the adversarial training, by alleviating the time overhead of attacks during the training. These techniques are complementary to our work. In the future, we may explore to combine our attack algorithm with them to further accelerate adversarial training based on L_2 attacks. On the other way around, we may study the problem of improving the accuracy of robust models for both clean and L_2 norm-based adversarial examples, by differentiating the distributions of these two class of images (Xie et al., 2020).

REFERENCES

Athalye, A., Carlini, N., and Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning (ICML)*.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndi´c, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion attacks against machine learning at test time. In *ECML-PKDD*.

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*.

[2]For CIFAR-10, 0: airplane, 1: automobile, 2: bird, 5: dog, 8: ship, 9: truck.

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. (2018). Ead: Elastic-net attacks to deep neural networks via adversarial examples. In *AAAI Conference on Artificial Intelligence*.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., and Li, J. (2018). Boosting adversarial attacks with momentum. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Girshick, R. (2015). Fast r-cnn. In *IEEE International Conference on Computer Vision*.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Gu, S., Yi, P., Zhu, T., Yao, Y., and Wang, W. (2019). Detecting adversarial examples in deep neural networks using normalizing filters. In *International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial examples in the physical world. In *arXiv preprint arXiv:1607.02533*.

Kurakin, A., Goodfellow, I., and Bengio, S. (2017). Adversarial machine learning at scale. In *International Conference on Learning Representations (ICLR)*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., C.-Y.Fu, and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European Conference on Computer Vision*.

Machado, G., Goldschmidt, R., and Silva, E. (2019). Multimagnet: A non-deterministic approach based on the formation of ensembles for defending against adversarial images. In *International Conference on Enterprise Information Systems - Volume 1: ICEIS*.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Modas, A., Moosavi-Dezfooli, S.-M., and Frossard, P. (2019). Sparsefool: a few pixels make a big difference. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Papernot, N., Mcdaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016). The limitations of deep learning in adversarial settings. In *IEEE Symposium on Security and Privacy*.

Prakash, A., Moran, N., Garber, S., DiLillo, A., and Storer, J. (2018). Protecting jpeg images against adversarial attacks. In *Data Compression Conference*.

Redmonand, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Rony, J., Hafemann, L. G., Oliveira, L. S., Ayed, I. B., Sabourin, R., and Granger, E. (2019). Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4317–4325.

Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. (2019). Adversarial training for free! In *Neural Information Processing Systems (NeurIPS)*.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Szegedy, H., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Uesato, J., O'Donoghue, B., van den Oord, A., and Kohli, P. (2018). Adversarial risk and the dangers of evaluating against weak attacks. In *arXiv preprint arXiv:1802.05666*.

Wong, E., Rice, L., and Kolter, J. (2020). Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*.

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A. L., and Le, Q. V. (2020). Adversarial examples improve image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *Proceedings of the British Machine Vision Conference*.

Zheng, H., Zhang, Z., Gu, J., Lee, H., and Prakash, A. (2020). Efficient adversarial training with transferable adversarial examples. In *IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*.