entropy

MDPI

Article

Minimum Adversarial Examples

Zhenyu Du *, Fangzheng Liu and Xuehu Yan

College of Electronic Engineering, National University of Defense Technology, Hefei 230037, China; liufangzheng17@nudt.edu.cn (F.L.); yanxh17@nudt.edu.cn (X.Y.)
* Correspondence: dzy17@nudt.edu.cn

Abstract: Deep neural networks in the area of information security are facing a severe threat from adversarial examples (AEs). Existing methods of AE generation use two optimization models: (1) taking the successful attack as the objective function and limiting perturbations as the constraint; (2) taking the minimum of adversarial perturbations as the target and the successful attack as the constraint. These all involve two fundamental problems of AEs: the minimum boundary of constructing the AEs and whether that boundary is reachable. The reachability means whether the AEs of successful attack models exist equal to that boundary. Previous optimization models have no complete answer to the problems. Therefore, in this paper, for the first problem, we propose the definition of the minimum AEs and give the theoretical lower bound of the amplitude of the minimum AEs. For the second problem, we prove that solving the generation of the minimum AEs is an NPC problem, and then based on its computational inaccessibility, we establish a new third optimization model. This model is general and can adapt to any constraint. To verify the model, we devise two specific methods for generating controllable AEs under the widely used distance evaluation standard of adversarial perturbations, namely L_p constraint and $SSIM$ constraint (structural similarity). This model limits the amplitude of the AEs, reduces the solution space's search cost, and is further improved in efficiency. In theory, those AEs generated by the new model which are closer to the actual minimum adversarial boundary overcome the blindness of the adversarial amplitude setting of the existing methods and further improve the attack success rate. In addition, this model can generate accurate AEs with controllable amplitude under different constraints, which is suitable for different application scenarios. In addition, through extensive experiments, they demonstrate a better attack ability under the same constraints as other baseline attacks. For all the datasets we test in the experiment, compared with other baseline methods, the attack success rate of our method is improved by approximately 10%.

Keywords: information security; minimum adversarial examples (AEs); controllable optimization of AEs; L_p constraint; $SSIM$ constraint

Citation: Du, Z.; Liu, F.; Yan, X. Minimum Adversarial Examples. *Entropy* **2022**, *24*, 396. https://doi.org/10.3390/e24030396

Academic Editor: Eduard Jorswieck

Received: 27 January 2022
Accepted: 10 March 2022
Published: 12 March 2022

1. Introduction

With the wide applications of a system based on DNNs, the concerns of their security become a focus. Recently, researchers have found that adding subtle perturbations to the input of deep neural networks causes models to give a wrong output with high confidence. Furthermore, they call the deliberately constructed inputs adversarial examples (AEs). The attack of DNNs by AEs is called adversarial attacks. These low-cost adversarial attacks can severely damage applications based on DNNs. Adding adversarial patches onto traffic signs can lead to auto-driving system error [1]. Adding adversarial logos to the surface of goods can impede automatic check-out in automated retail [2]. Generating adversarial master prints can destroy deep fingerprint identification models [3]. In any of the aforementioned scenarios, AEs can cause great inconvenience and harm people's lives. Therefore, AEs become an urgent issue in the area of AI security.

In the research on generating AEs, two fundamental problems exist: (1) What is the minimum boundary of the amplitude of adversarial perturbations? All the models try to

generate AEs with smaller adversarial perturbations. It is their objective to add as few adversarial perturbations as necessary to the clean example to achieve the attack; (2) Is the minimum boundary of adversarial amplitude reachable? The reachability refers to whether examples with adversarial perturbations that are under a minimum bound of adversarial amplitude can successfully attack as well as whether AEs exist under that boundary.

In order to answer those two problems, traditional AE generation can be devised into two main optimization models: (1) Taking the successful attack as the objective function and the limitation of perturbations as the constraint. This limitation is usually limited as less than or equal to a value, as shown in Equation (1). For a neural network F, input distribution $\aleph \subset \mathbb{R}^n$, a point $X_0 \in \aleph$, $X \in \mathbb{R}^n$, X is the adversarial example of X_0 under the v constraint. D is the distance metric function:

$$\begin{aligned} &F(X) \neq F(X_0) \\ &\text{s.t. } D(X, X_0) \leq v \end{aligned} \tag{1}$$

(2) Taking the minimum of adversarial perturbations as the target and the success of the attack as the constraint:

$$\begin{aligned} &\min D(X, X_0) \\ &\text{s.t. } F(X) \neq F(X_0) \end{aligned} \tag{2}$$

However, the above two models do not solve the two problems well: (1) for the first model, when setting the limitation of AEs in the constraint, whether the model has a solution depends on the limit value v. The model may have no solution when the limit value v is too small. However, when the limit value is larger, the constraint on the AEs is too relaxed, and thus the gap between the solution and the minimum AEs is larger; (2) For the second model, when the limitation of adversarial perturbations is in the objective function, the perturbations will decrease in the whole optimization process until it drops in the local optimum of the whole objective function. This optimization model can easily fall into local optimization so that the solution is not the minimum adversarial example. At the same time, this paper also proves that finding the minimum AEs is an NPC problem, so it cannot find the real minimum AEs.

Therefore, in this paper, we focus on answering the problems mentioned above. For the first problem, we propose the concept of minimum AEs and give the theoretical lower bound of the amplitude of minimum adversarial perturbations. For the second problem, we prove that generating the minimum adversarial example is an NPC problem, which means that the minimum boundary of adversarial amplitude is computationally unreachable. Therefore, we generate the controllable approximation of the minimum AEs. We use the certified lower bound of minimum adversarial distortion to constrain the adversarial perturbations and transform the traditional optimization problem into another new model. (3) Taking the successful attack as a target and the adversarial perturbations are equal to the lower bound of the minimum adversarial distortion plus a controllable approximation, as shown in Equation (3). ε_{NNS} is the lower bound of the minimum adversarial distortion and δ_ε is a constant of controllable approximation:

$$\begin{aligned} &F(X) \neq F(X_0) \\ &\text{s.t. } D(X, X_0) = \varepsilon_{NNS} + \delta_\varepsilon \end{aligned} \tag{3}$$

This model has two advantages compared with the existing methods: (1) Better attack success rate under the same amplitude of adversarial perturbations. Based on the theoretical lower bound of the amplitude of the minimum perturbations, the AEs overcome the blindness of the existing methods by controlling the increment in that amplitude and improve the attack success rate of the AEs. (2) More precisely controlled amplitude of adversarial perturbations under different constraints. The amplitude of the adversarial perturbations will affect the visual quality of AEs. To go a step further, for different scenarios of applications of the AEs, the requirements of visual quality are different. In

some scenarios, they are very strict, while others are relaxed. There are two common scenarios as follows: (1) collaborative evaluation of humans and machines. In that case, AEs need to deceive both human oracles and the classifiers based on DNNs. For example, in the scenario of auto-driving, if the patches too easily draw humans' attention, these adversarial signs would be moved and they would lose their adversarial effect. (2) Single evaluation of machines. In that case, only the classifiers and models based on DNNs need to be bypassed. In the scenario of massive electronic data filtering, they have a low probability of human involvement. When filtering and testing the harmful data involving violence and terrorism, it may heavily depend on the machines so that it has lower requirements for visual quality. Therefore, in order to adapt the two entirely different scenarios, we need to be able to controllably generate AEs.

Meanwhile, generating controllable AEs also brings additional benefits. There are two different views with different implications: (1) Attackers can adaptively and dynamically adjust the amplitude of perturbations. As the described above, the defense technologies against adversarial attacks are mainly detection methods. From the attackers' point of view, when their target is a combined network or system with detectors in front of the target classifier, as Figure 1 shows, they will expect to evaluate the successful probability of attacking the combined network before implementing the attack. For example, supposing that they know the probability of AEs with fixed perturbation bypassing the detector in advance according to prior knowledge, then they can purposefully generate AEs with bigger perturbations or more minor perturbations with a better visual quality to human eyes. (2) Defenders can actively defend against the attacks with the help of the outputs of controllable AEs. From the defenders' point of view, controllable AEs can help evaluate defenders' abilities against the AEs of different modification amplitude. When inputting different AEs with fixed adversarial perturbations to models, the defenders can evaluate their anti-attack capabilities according to the outputs against the unclean examples and then decide whether to add additional defense strategies with an emphasis on the current setting. For the example mentioned in the last point, if the defender has prior knowledge about the attackers' average perturbation amplitude, they can select whether additional defensive measures are necessary.

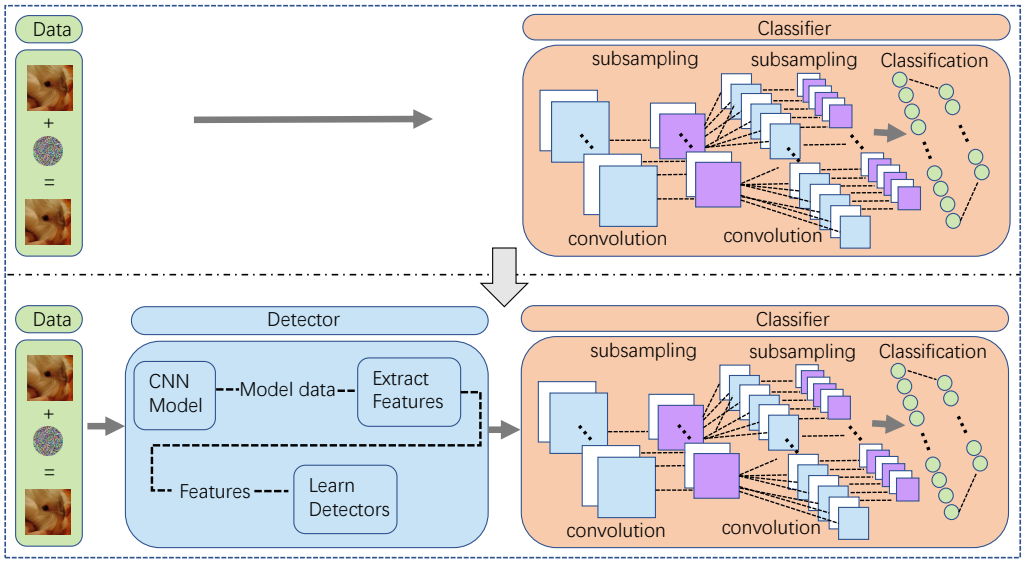

Figure 1. Figure representing the framework of the settings. The top framework is the traditional attack setting and the bottom is our attack setting. In the top setting, the target of the adversarial attack is a single target classifier while our setting is a combined network including a target classifier and a detector.

In this paper, we first give the definitions of minimum adversarial perturbations and AEs and the theorem of generating minimum AEs as an NPC problem and then propose a new model of generating adversarial examples. Furthermore, we give two algorithms for generating an approximation of AEs under L_p and $SSIM$ constraints. We perform experiments under widely used datasets and models for all the datasets tested in the experiment; compared with other baseline methods, the attack success rate of our method is improved by approximately 10%.

Our contributions are as follows:

- We first prove that generating minimum AEs is an NPC problem. We then analyze the existence of AEs with the help of the definition of the lower bound of the minimum adversarial perturbations. According to the analysis, we propose a general framework to generate an approximation of the minimum AEs.
- We propose the methods of generating AEs with a controllable amplitude of AEs under the L_2 and $SSIM$ constraints. Additionally, we further improve the visual quality in case of greater perturbations.
- The experiments demonstrate that our method has a better performance in terms of attack success rate than other widely used methods at baseline under the same constraint. Meanwhile, its performance of precisely controlled amplitude of adversarial perturbations under different constraints is also better.

The rest of this paper is organized as follows. In Section 2, we briefly review the related work. In Section 3, we describe the basic definition, theorem and model of our algorithm in detail and prove the theorem. In Section 4, we give the transformed model of the basic model under two constraints and provide the efficient solution algorithm of the two models, respectively, in the two subsections. In Section 6, we present our experimental results and compare them with other baseline methods. Finally, we conclude our paper in Section 7.

2. Related Work

2.1. Adversarial Attack

There are two main pursuits of AEs: one is the smaller perturbations of the AEs; and the other is the successful attack. Previous works transform the two pursuits into two main optimization models. One takes the successful attack as the objective function and the limitation of perturbations as the constraint. These works include L-BFGS [4], C&W [5], DF [6] and HCA [7]. The other takes the successful attack as the objective function and the limitation of perturbations as the constraint. Such works include UAP [8], BPDA [9] and SA [10]. Other works, including FGSM [11], JSMA [12], BIM [13] and PGD [14] do not directly use the model of the optimization problem. However, these methods convert the successful attack into a loss function, move it along the direction of the decrease or increase in the loss function to find the AEs, and use a value at each step to constrain the perturbations. They can be classified as the second optimization model from the point of method-based view.

However, these works cannot really find the minimum AEs with the minimum amplitude of adversarial perturbations. For the first model, the model may have no solution when the value is set as too small. Furthermore, for the second model, it is easy to fall into local optimization.

Meanwhile, considering the constraint function of adversarial perturbations, the works of adversarial example generation can be divided into two main classes. One AEs generation under L_p constraint, including L_0 constraint [14,15], L_2 constraint [14] and L_∞ constraint [11,13,14], which is widely used. Furthermore, in addition to that L_p constraint, there were other constraints in previous studies. In [16], the authors proposed that the commonly used L_p constraint failed to completely capture the perceptual quality of AEs in the field of image classification. This used the structural similarity index SSIM [17] measure to replace that constraint. Moreover, the other two works [18,19] also used perceptual dis-

tance measures to generate AEs. The work [18] used SSIM while [19] used the perceptual color distance to achieve the same purpose.

However, the constraint of those works is not strict. For the AEs generation under the L_p constraint, it is hard to control the amplitude of perturbations and there is a deviation of AEs generated by those works. For the other constraints, they cannot strictly control the perceptual visual quality: neither the SSIM value nor perceptual color distance.

Therefore, in this paper, we search for the minimum AEs with the minimum amplitude of perturbations. Moreover, we prove that generating the minimum AEs is an NPC problem. Furthermore, we transform that problem into the new optimization model that generates the controllable approximation of the minimum AEs. We generate AEs with a controllable amplitude of adversarial perturbations under the L_p constraint and $SSIM$ constraint, respectively.

2.2. Certified Robustness

The robustness of neural networks focuses on searching the lower bound and upper bound of the robustness of neural networks. The lower bound of the robustness is that there are no AEs when adding adversarial perturbations that are less than or equal to that boundary. Moreover, the upper bound of the robustness adding AEs that are larger than or equal to that bound can always acquire the AEs. The work CLEVER [20] and CLEVER++ [21] were the first neural network robustness evaluation scores. They use extreme value theory to estimate the Lipschitz constant based on sampling. However, that estimation requires many samples to have a better value of estimation. Therefore, the two methods only estimate the lower bound of the robustness of neural networks and cannot provide certification. As follows, the works Fast-Lin and Fast-Lip [22], CROWN [23] and CNN-Cert [24] are methods of certifying the robustness of the neural networks. The Fast-Lin and Fast-Lip [22] can only be used for neural networks with the activation function of ReLu. CROWN [23] can be further used for the networks with all general activation functions. Furthermore, the CNN-Cert [24] can be used for the general convolutional neural networks (CNNs). The basic idea is constructing linear functions to constrain the input and then using the upper and lower bounds of the functions as the upper and lower bounds of input, respectively. After that, it can constrain the whole network layer by layer. The whole process is iterative.

However, the above algorithm does not indicate how to calculate the AEs according to the calculated lower bound, and the reachability of AEs based on the lower bound remains a problem. Therefore, in this paper, we calculate the approximation of the minimum AEs based on the lower bound.

3. Basic Definition, Theorem and Modeling

Definition 1. *(AEs, Adversarial Perturbations). Given a neural network F, a distribution $\aleph \subset \mathbb{R}^n$, a distance measurement $D : R^n \times R^n \to R$ between X and X_0, a point $X_0 \in \aleph$ and a point $X \in \mathbb{R}^n$, we say that X is an adversarial example of X_0 under constraint ε_0 if $F(X) \neq F(X_0)$ and $D(X, X_0) = \varepsilon_0$.*

Definition 2. *(Minimum AEs, Minimum Adversarial Perturbations). Given a neural network F, a distribution $\aleph \subset \mathbb{R}^n$, a distance measurement $D : R^n \times R^n \to R$ between X and X_0, and a point $X_0 \in \aleph$, we say that $X^* \in R^n$ is a minimum adversarial example of X_0 if X^* is an adversarial example of X_0 under constraint ε^* and $\varepsilon^* = \min_X \varepsilon_0$ such that there exists an adversarial example of X_0 under constraint ε_0. ε^* is the minimum adversarial perturbations of X_0 under D constraint.*

Theorem 1. *Given a neural network F, a distribution $\aleph \subset \mathbb{R}^n$, a distance measurement $D : R^n \times R^n \to R$ between X and X_0 and a point $X_0 \in \aleph$, searching for a minimum adversarial example of X_0 is an NPC problem.*

Proof. The proof of Theorem 1 is shown in Appendix A. □

Although it is an NPC problem, researchers calculate the non-trivial upper bounds of the robustness of the neural network [23–25]. We can thus calculate the non-trivial lower bounds of the minimum adversarial perturbations ε_{NNS} of X_0 based on the exact meaning of the two bounds.

We thus model the problem of calculating the non-trivial lower bounds of the minimum adversarial perturbations ε_{NNS} of X_0. For input distribution $\aleph \subset \mathbb{R}^n$, a clean input X_0, perturbed input X of X_0 under the ε constraint, $X \in \mathbb{B}(X_0, \varepsilon)$, $\mathbb{B} = \{X : D(X, X_0) \leq \varepsilon\}$, a neural network $F : \mathbb{R}^n \to \mathbb{R}^k$, original label y of X_0, $F(X_0) = y$, target label y^*, $y^* \neq y$, and we define the non-trivial lower bounds of the minimum adversarial perturbations as ε_{NNS} of X_0, as shown in Equation (4):

$$\varepsilon_{NNS} = \max_{y^* \neq y} \varepsilon^*_{y^*} \tag{4}$$

and:

$$\begin{aligned} \varepsilon^*_{y^*} &= \min \epsilon \\ \text{s.t. } \gamma^U_y(X) - \gamma^L_{y^*}(X) &\leq 0 \end{aligned} \tag{5}$$

In Equation (4), ε^*_{y*} is the minimum of adversarial perturbations of X_0 under the target label y^*. In Equation (5), ε is the perturbation of X_0 such that $F(X) = y^*$, $\gamma^U_y(X)$ means the upper bound of the network under label y of input X and $\gamma^L_{y^*}$ means the lower bound of the network under another label y^* of the input. They are calculated in [23–25].

Theorem 2. *Given a neural network F, a distribution $\aleph \subset \mathbb{R}^n$, a distance measurement $D : R^n \times R^n \to R$ between X and X_0, a point $X_0 \in \mathbb{R}^n$, the non-trivial lower bounds $\varepsilon_{NNS} \in \mathbb{R}$ of the minimum adversarial perturbations of X_0, if X is the perturbed example of X_0 under constraint ε_{NNS} and $X \in \mathbb{B}(X_0, \varepsilon_{NNS})$, then $F(X) \equiv F(X_0)$.*

Proof. According to the definition and meaning of the ε_{NNS}, we can obtain Theorem 2. □

Definition 3. *(N-order tensor [26]). In deep learning, a tensor extends from a vector or matrix to a higher dimensional space. The tensor can be defined by a multi-dimensional array. The dimension of a tensor is also called order, that is, N-dimensional tensor, also known as N-order tensor. For example, when $N = 0$, the tensor is a 0-order tensor, which is one number. When $N = 1$, the tensor is a 1-order tensor, which is a 1-dimensional array. When $N = 2$, the tensor is a 2-order tensor, which is a matrix.*

Definition 4. *(Hadamard product [26]). The Hadamard product is the element-wise matrix product. Given the N-order tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, the Hadamard product $\mathcal{A} \times \mathcal{B}$ is denoted as the product of elements corresponding to the same position of the tensor. The product $\mathcal{C}$ is a tensor with the same order and size as $\mathcal{A}$ and $\mathcal{B}$. That is:*

$$\mathcal{C} = \mathcal{A} \times \mathcal{B}, \mathcal{C}_{i_1,\dot{2},\ldots,i_n} = \mathcal{A}_{1,i_2,\ldots,i_n} \times \mathcal{B}_{i_1,i_2,\ldots,i_n} \tag{6}$$

Definition 5. *($+^*$). For a real number $\lambda \in \mathbb{R}$ and N-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, we define $\lambda +^* \mathcal{X}$ as the sum of $\mathcal{X}$ and the Hadamard product of λ and another tensor $\Psi \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$. That is:*

$$\mathcal{D} = \lambda +^* \mathcal{X} = \lambda \times \Psi + \mathcal{X}, \quad \mathcal{D}_{i_1,i_2,\ldots,i_n} = \lambda \times \Psi_{i_1,i_2,\ldots,i_n} + \mathcal{X}_{i_1,i_2,\ldots,i_n} \tag{7}$$

$\mathcal{D}$, Ψ and $\mathcal{X}$ have the same order. Specifically, in the field of the AEs, given a clean input $X_0 \in \mathbb{R}^N$, and perturbations $r \in \mathbb{R}$, the adversarial example is $X = X_0 +^* r = X_0 + \Psi \times r$. The physical meaning is the proportionality factor of r which adds on each feature $X_{0i_1,i_2,\ldots,i_n}$.

For example, $\lambda = 2$, $\mathcal{X} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $\Psi = \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$,

$$\mathcal{D} = \lambda +^* \mathcal{X} = \lambda \times \Psi + \mathcal{X} = 2 \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 4 & 7 \\ 11 & 14 \end{bmatrix}$$

Definition 6. *($\varepsilon \sim \tau$ approximation of minimum AEs, $\varepsilon \sim \tau$ approximation of minimum adversarial perturbations). Given a neural network F, a distribution $\aleph \subset \mathbb{R}^n$, a point $X_0 \in \aleph$, the non-trivial lower bounds $\varepsilon_{NNS} \in \mathbb{R}$ of the minimum adversarial perturbations of X_0, a constraint ε_τ^* and $\varepsilon_\tau^* = \varepsilon_{NNS} + \tau$, $\tau > 0$ is a constant, we say that X_τ^* is the $\varepsilon \sim \tau$ approximation of minimum AEs of X_0 and ε_τ^* is the $\varepsilon \sim \tau$ approximation of minimum adversarial perturbations such that $X_\tau^* = X_0 +^* \varepsilon_\tau^*$, $F(X_\tau^*) = F(X_0 +^* \varepsilon_\tau^*) \neq F(X_0)$.*

τ is a constant set by humans according to the actual statement. When generating an adversarial example for a specific input, it has different requirements of the adversarial perturbations for different settings, scenarios and samples.

(a) The more complex the scenario is, the smaller the constant τ is. In the extreme scenario of digital AEs generation, it needs a clear filter of the AEs and has a strict requirement of invisibility, and the τ should be small [15]. However, for most physical AEs generations, it has the relaxed requirement of invisibility. Most of them only need to keep semantic consistency. The τ can be set more considerably than the digital setting [27].

(b) The more simple the sample is, the smaller the constant τ is. When the sample is simple, its information is single, and people would be more sensitive to the perturbations than complex samples. It is easier for people to recognize the difference between clean inputs and perturbed inputs. For example, the τ of the MNIST dataset [28] should be smaller than the CIFAR-10 dataset [29].

We model the problem of generating AEs under D measure metrics as follows. For a deep neural network F, input distribution $\aleph \subset \mathbb{R}^n$, a point $X_0 \in \aleph$ and given the distance value d under constraint D, the problem of generating controllable AEs of d can be modeled as

$$\begin{aligned} &F(X) \neq F(X_0) \\ &\text{s.t. } X \in \mathbb{B} = \{X : D(X, X_0) = d\} \end{aligned} \tag{8}$$

We discuss the problem under two settings. One is the constraint of the L_p norm, and the other is that of perceptually constrained D measure metrics. We use the widely used structural similarity (SSIM) as the perceptual constraint in a perceptually constrained AEs generation. The two constraints will be discussed, respectively, in the following sections.

4. AEs Generation under L_p Constraint

4.1. Analysis of the Existence of AEs

According to Theorem 2, we reach the following conclusions concerning the existence of AEs, as shown in Figure 2.

As Figure 2a shows, we have the following analysis. When adding adversarial perturbations lower than ε_{NNS}, no AEs of X_0 exist.

When adding adversarial perturbations larger than ε_{NNS}, AEs of X_0 exist. The gray shadow between the red circle and the blue line is the space where AEs exist. However, whether AEs can be found depends on the direction of adding ε perturbations. As the figure shows, the perturbations of X_A and X_B all equal ε, and they are all located on the bound of the ball of ε; however, we can see that X_A is inside the gray shadow while X_B is not.

Therefore, some conclusions that were previously well known hypotheses can be proven. Different AEs generation methods generate AEs with varying accuracy. For a clean input X_0, when adding the same perturbations on it, method A can acquire the adversarial input X_A. In contrast, method B obtains X_B located inside the blue line and can still be correctly classified by the network. Hence, the key to generating AEs is finding the direction of where AEs exist. As shown in Figure 2a, when it is along the path of X, the added perturbations are the smallest.

Meanwhile, for different clean samples, the added perturbations of generating AEs are different. When a specific perturbation $\varepsilon > \varepsilon_{NNS}$ is fixed, different clean samples will obtain different perturbed examples after adding those perturbations. As shown in Figure 2b, the blue boundary and the yellow boundary are the different classification boundaries of two different samples, respectively. The perturbed examples acquired by adding the same perturbations ε are within the yellow boundary but outside the blue border. Therefore, they are the AEs of the blue boundary constraint samples which can be correctly classified by the yellow boundary constraints. Thus, the adversarial example needs to be researched for a specific sample.

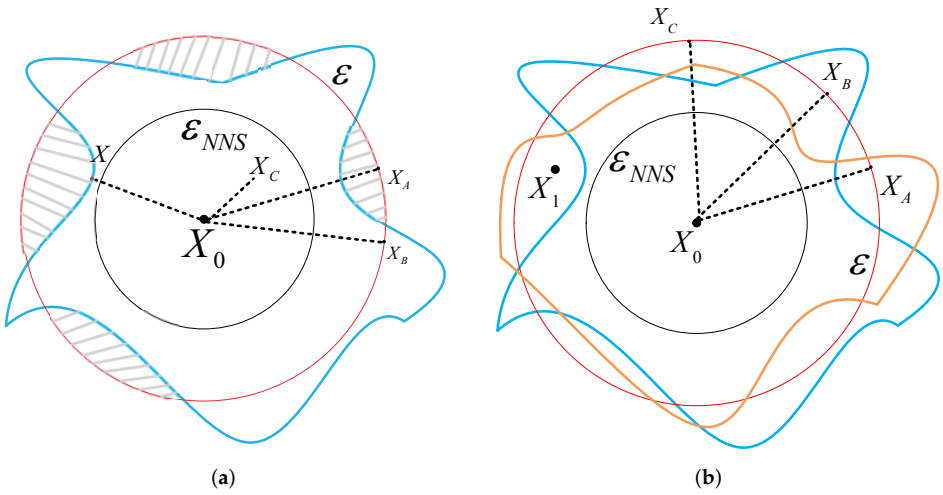

Figure 2. Figure representing the spaces where AEs exist. The black circle indicates the ball of the non-trivial lower bounds of the minimum adversarial perturbations of X_0; the blue line indicates the classification bound of the network when X_0 is input; and the red circle means the ball of adding perturbations ε on X_0. When examples are inside the blue line, they can be classified as the original label by the network. However, when they are outside the blue line, they are AEs. The gray shadow indicates the space where AEs exist under the ball of the ε. The yellow boundary is another classification border of X_1.

Therefore, according to the analysis of the existence of AEs, we have the following conclusions. In order to generate practical AEs, the added perturbations quantity needs to meet the requirement $\varepsilon > \varepsilon_{NNS}$ and it needs to be larger than the classification boundary in this direction. At the same time, due to the limitation of invisibility of the AEs, it should be as small as possible. Thus, the generation direction of the AEs should be closer to the direction of minimum AEs.

According to Theorem 1, searching for the minimum AEs of sample X_0 is an NPC problem. In this paper, we try to generate the minimum AEs under a $\varepsilon \sim \tau$ numerical approximation as Definition 6.

According to Figure 2a, in order to generate an effective adversarial example, the perturbations should be larger than the lower bound ε_{NNS} and the perturbations needed to cross the boundary of the classifier. When fixing ε^*_τ, it defines a ball with a center of X_0 and radius ε^*_τ. As shown in Figure 2a, the points on the ball are not all AEs. Using the $+$ method is the same as selecting a random direction to generate perturbed examples that are highly unlikely to be adversarial. Therefore, it is necessary to calculate the direction of adding ε^*_τ and make $F(X^*_\tau) = F(X_0 + \Psi \times \varepsilon^*_\tau) \neq F(X_0)$. Ψ is the direct tensor of effective AEs.

4.2. Model of L_p Constraint

We model the problem of generating the $\varepsilon \sim \tau$ approximation of minimum AEs. For a neural network F, the input distribution $\aleph \subset \mathbb{R}^n$, a point $X_0 \in \aleph$ and given the $\varepsilon \sim \tau$ approximation of minimum adversarial perturbations ε_τ^*, the problem of generating the $\varepsilon \sim \tau$ approximation of the minimum adversarial example X_τ^* can be modeled as

$$\begin{aligned} &F(X_\tau^*) \neq F(X_0) \\ &\text{s.t. } X_\tau^* \in \mathbb{B}_p = \left\{ X : \|X - X_0\|_p = \varepsilon_\tau^* \right\} \end{aligned} \tag{9}$$

According to the analysis of the existence of AEs and Theorem 2, when the added adversarial perturbations $\varepsilon > \varepsilon_{NNS}$, AEs certainly exist and the model must have a solution.

4.3. Framework of AE Generation under L_p Constraint

According to Definition 6, we transform the problem of calculating the $\varepsilon \sim \tau$ approximation of minimum adversarial example ${X_0}_\tau^*$ into searching for the direct tensor Ψ.

For a neural network F, input distribution $\aleph \subset \mathbb{R}^n$, a point $X_0 \in \aleph$ and given the $\varepsilon \sim \tau$ approximation of minimum adversarial perturbations ε_τ^*, according to Definition 5, the $\varepsilon \sim \tau$ approximation of the minimum adversarial example is X_τ^*, and $X_\tau^* = X_0 +^* \varepsilon_\tau^* = X_0 + \varepsilon_\tau^* \times \Psi$, which means:

$$\|X_0 + \varepsilon_\tau^* \times \Psi - X_0\|_p = \varepsilon_\tau^* \tag{10}$$

$$F(X_0 + \varepsilon_\tau^* \times \Psi) \neq F(X_0) \tag{11}$$

This model must have solutions, and we can consider a special solution. We set one element of $\Psi \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ as 1 and the others are 0, fulfilling Equation (10). When the clean input is an image, it means modifying one channel of one pixel of the image, as proposed in [15]. However, this attack only has a 20.61% success rate on VGG-16 [30] of cifar-10 [29]. Furthermore, the perturbations of this pixel are too large to be set as τ.

It is difficult to directly calculate Ψ; thus, to solve Equation (10), we decompose $\varepsilon_\tau^* \times \Psi$ into the two tensors $\delta \times \Lambda$ and $\delta, \Lambda \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, and each element of δ and Λ are defined as $\delta_{i_1,i_2,\ldots,i_n}$ and $\Lambda_{i_1,i_2,\ldots,i_n}$, respectively. The n-order tensor δ determines the location of the added perturbations and the importance of the target label while the n-order tensor Λ determines the size of the added perturbations, that is, the percentage of the total perturbations.

According to Equation (10), we obtain the following derivation:

$$\begin{aligned} \|X_0 + \varepsilon_\tau^* \times \Psi - X_0\|_p &= \|\varepsilon_\tau^* \times \Psi\|_p \\ &= \sqrt[p]{(\varepsilon_\tau^* \times \Psi)^p} \\ &= \sqrt[p]{(\delta \times \Lambda)^p} \\ &= p\sqrt{\sum_{i}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i.}^{I_n} \delta_{i_1 i_2 \ldots i_n}^p \times \Lambda_{i_1 i_2 \ldots i_n}^p} \\ &= \varepsilon_\tau^* \end{aligned} \tag{12}$$

Therefore:

$$\sum_{i}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i.}^{I_n} \delta_{i_1 i_2 \ldots i_n}^p \times \Lambda_{i_1 i_2 \ldots i_n}^p = {\varepsilon_\tau^*}^p \tag{13}$$

However, in Equation (13), the two tensors are all unknown and all of them have n elements, so it is a multivariate n-order equation and still unsolvable. Although it is unsolvable, we can certify it as a trivial solution. We can certify when:

$$\Lambda^{p}_{i_1 i_2 \ldots i_n} = \varepsilon^{*p}_{\tau} / \sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_n}^{I_n} \delta^{p}_{i_1 i_2 \ldots i_n} \tag{14}$$

Equation (14) is workable. The proof is shown as follows.

Proof.

$$\begin{aligned} & \sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_n}^{I_n} \delta^{p}_{i_1 i_2 \ldots i_n} \times \Lambda^{p}_{i_1 i_2 \ldots i_n} \\ &= \delta^{p}_{11\ldots1} \times \Lambda^{p}_{11\ldots1} + \delta^{p}_{11\ldots2} \times \Lambda^{p}_{11\ldots2} + \ldots + \delta^{p}_{I_1 \times I_2 \times \ldots \times I_N} \times \Lambda^{p}_{I_1 \times I_2 \times \ldots \times I_N} \\ &= \frac{\varepsilon^{*p}_{\tau} \times \delta^{p}_{11\ldots1}}{\sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_n}^{I_n} \delta^{p}_{i_1 i_2 \ldots i_n}} + \frac{\varepsilon^{*p}_{\tau} \times \delta^{p}_{11\ldots2}}{\sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_n}^{I_n} \delta^{p}_{i_1 i_2 \ldots i_n}} + \ldots + \frac{\varepsilon^{*p}_{\tau} \times \delta^{p}_{I_1 \times I_2 \times \ldots \times I_N}}{\sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_n}^{I_n} \delta^{p}_{i_1 i_2 \ldots i_n}} \\ &= \varepsilon^{*p}_{\tau} \end{aligned} \tag{15}$$

□

We only need to search for one solution of the model (9). That is, we only need to generate one $\varepsilon \sim \tau$ approximation of a minimum adversarial example corresponding to the requirements (9). The trivial solution Equation (14) is therefore the result.

Therefore, the problem of generating the $\varepsilon \sim \tau$ approximation of minimum AEs is transformed into generating the tensor Ψ by Definition 6 and it is then transformed into calculating the two tensors δ, Λ by Equation (13). Moreover, it is finally transformed into calculating the tensor δ. However, it is still an unsolvable question. Although the only thing we need to do is calculate the tensor δ, it is an n-order tensor in the real world so that there are n elements that remain unknown and need to be calculated. According to Equation (13), when tensor δ is known, the problem of solving the multivariate n-order equation is turned into a multivariate 1-order equation. If we want to solve the multivariate 1-order equation, we need n equations. However, we only have one equation, which is Equation (13). Therefore, this paper proposes the solution framework for generating the $\varepsilon \sim \tau$ approximation of minimum AEs and a heuristic method to solve the problem.

4.4. Method of Generating Controllable AEs under L_p Constraint

According to the definition of the AEs, we decompose the tensor δ into $\varpi + \alpha\xi$, $\varpi, \xi \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$. Each element of ϖ and ξ are defined as $\varpi_{i_1, i_2, \ldots, i_n}$ and $\xi_{i_1, i_2, \ldots, i_n}$, respectively.

$$\delta_{i_1, i_2, \ldots, i_n} = \varpi_{i_1, i_2, \ldots, i_n} + \alpha \xi_{i_1, i_2, \ldots, i_n} \tag{16}$$

Because the N-order tensor determines the position of adding perturbations and the importance of the position to the target label, it contains two factors that restrict the value of the AEs. One is to improve the invisibility of the AEs so that added perturbations should be insensitive to human eyes. Another is to improve the effectiveness of the AEs so that the added perturbations should be able to push the sample away from the original classification boundary (in the case of non-target attack) or close to the target classification boundary (in the case of target attack. Obtaining a balance between the two factors is a key problem in the study of AEs. Therefore, we decompose the δ into ϖ and ξ.

Importantly, ϖ is the tensor to determine the effectiveness of AEs and ξ is the tensor to determine the invisibility of AEs. According to Equation (13), we have:

$$\begin{aligned}
&\varepsilon^*_{\xi_{i_1,i_2,\ldots i_n}} \times \Psi_{i_1,i_2,\ldots,i_n} \\
&\quad = \left(\alpha_1 \varpi_{i_1,i_2,\ldots,i_n} + \alpha_2 \xi_{i_1,i_2,\ldots,i_n}\right) \times \varepsilon^{*p}_{\xi} / \sum_{i_1}^{I_1}\sum_{i_2}^{I_2}\cdots\sum_{i_n}^{I_n} \delta^p_{i_1 i_2 \ldots i_n} \\
&\quad = \left(\alpha_1 \varpi_{i_1,i_2,\ldots,i_n} + \alpha_2 \xi_{i_1,i_2,\ldots,i_n}\right) \frac{\varepsilon^{*p}_{\xi}}{\sum_{i_1}^{I_1}\sum_{i_2}^{I_2}\cdots\sum_{i_n}^{I_n} \delta^p_{i_1 i_2 \ldots i_n}}
\end{aligned} \tag{17}$$

Therefore, the perturbations added on each element are:

$$\begin{aligned}
&X_{i_1,i_2,\ldots i_n} \\
&= X_{0 i_1,i_2,\ldots i_n} + \left(\alpha_1 \varpi_{i_1,i_2,\ldots,i_n} + \alpha_2 \xi_{i_1,i_2,\ldots,i_n}\right) \frac{\varepsilon^p}{\sum_{i_1}^{I_1}\sum_{i_2}^{I_2}\ldots\sum_{i_n}^{I_n}\left(\alpha_1 \varpi_{\hat{h}_1 i_2,\ldots,i_n} + \alpha_2 \xi_{\hat{h}_1,i_2,\ldots,i_n}\right)^p}
\end{aligned} \tag{18}$$

According to the above analysis, we transform the $\varepsilon \sim \tau$ approximation of minimum AEs generation into calculating the ϖ and ξ.

4.4.1. Calculating ϖ

According to the analysis of Equation (5), when $\gamma_y^U(X)$ is lower than $\gamma_{y^*}^L(X)$, the input X is an adversarial example. This means that the upper bound of the network under the original label of input X is lower than the lower bound of the network under other labels. Therefore, we let:

$$\begin{aligned}
\varpi_{i_1,i_2,\ldots,i_n} &= -\nabla_X \hbar(X) \\
\hbar(X) &= \gamma_y^U(X) - \gamma_{y^*}^L(X)
\end{aligned} \tag{19}$$

In the initial update step, the perturbed examples are not in the shadow space so that they are still correctly recognized by the model and $\gamma_y^U(X) - \gamma_{y^*}^L(X) > 0$. At this time, we need to make the examples as close as possible to reducing the $\hbar(X)$, so the update direction is opposite to the gradient. When the $\hbar(X)$ value is less than 0, the absolute value of the $\hbar(X)$ needs to be larger, but the real $\hbar(X)$ value still needs to decrease so that the update direction remains the opposite of the gradient direction.

4.4.2. Calculating ξ

According to the definition of ξ, ξ is the tensor to determine the invisibility of AEs. DCT transformation [31] can transform the data from host space to frequency domain space, and the data in the time-domain or space-domain can be transformed into a frequency-domain that is easy to analyze and process. When data are image data, after transformation, much crucial visual information about the images is concentrated in a small part of the coefficient of DCT transformation. The high-frequency signal corresponds to the non-smooth region in the image, while the low-frequency signal corresponds to the smoother region in the image.

According to the human visual system (HVS) [17], (1) human eyes are more sensitive to the noise of the smooth area of the image than the noise of the non-smooth area or the texture area; (2) human eyes are more sensitive to the edge information of the image and the information is easily affected by external noise.

Therefore, according to the definition of DCT, we can distinguish the features of each region of the image and selectively add perturbations. Given that the N-order tensor input data $X_0 \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ can be seen as a superposition of $I_1 \times I_2 \times \ldots \times I_N / I_i \times I_j$ two-order tensor $X_0^{\Pi} \in \mathbb{R}^{I_i \times I_j}$:

$$DCT\left(X_0^{\Pi}\right)_{k,l} = \frac{2}{\sqrt{i_i i_j}} c(k)c(l) \sum_{m=0}^{i_i-1}\sum_{n=0}^{i_j-1} X_{0\,i_i,i_j}^{\Pi} \cos\frac{(2m+1)k\pi}{2i_i} \cos\left(\frac{(2n+1)l\pi}{2i_j}\right) \tag{20}$$

and $m, k \in \{0, 1, \ldots, i_i - 1\}, n, l \in \{0, 1, \ldots, i_j - 1\}$.

$$c(k) = \begin{cases} 1/\sqrt{2} & k = 0 \\ 1 & k = 1, 2, \ldots, i_i - 1 \end{cases}, c(l) = \begin{cases} 1/\sqrt{2} & l = 0 \\ 1 & l = 1, 2, \ldots, i_j - 1 \end{cases} \tag{21}$$

In this paper, according to the definition of the tensor of ξ:

$$\xi_{i_1, i_2, \ldots, i_n} = \mathrm{DCT}\left(X_0^{\Pi}\right)_{k,l} \tag{22}$$

Above all, we give the algorithm that generates the $\varepsilon \sim \tau$ approximation of minimum AEs under the Lp constraint in Algorithm 1.

Algorithm 1: Algorithm of the generating $\varepsilon \sim \tau$ approximation of minimum AEs under Lp constraint.

Input: a point $X_0 \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, $\varepsilon \sim \tau$ approximation of minimum adversarial perturbations ε_τ^*, a neural network F, the non-trivial lower bounds ε_{NNS} of the minimum adversarial perturbations of X_0
Input: Parameters: number of iterations n, α
Output: X_τ^*

for e in n **do**
  if $e = 0$ **then**
    $X = X_0$
    $\triangle\, \varepsilon = \varepsilon_{NNS}$
  end if
  $\omega_{i_1, i_2, \ldots, i_N} = -\nabla_X \hbar(X)$
  $\xi_{i_1, i_2, \ldots, i_N} = \mathrm{DCT}\left(X^{\Pi}\right)_{k,l}$
  $\Psi = \omega_{i_1, i_2, \ldots, i_N} + \alpha \xi_{i_1, i_2, \ldots, i_N}$
  $\triangle\, \varepsilon = (\varepsilon_\tau^* - \varepsilon_{NNS})/(n-1)$
  $X = X + \Psi \times \triangle\, \varepsilon$
end for
return X_τ^*

5. AEs Generation under SSIM Constraint

We model the problem of generating AEs under $SSIM$ [17] measure metrics as follows. We use $SSIM$ to replace the D measure metrics in Equation (8). For a neural network F, input distribution $\aleph \subset \mathbb{R}^n$, a point $X_0 \in \aleph$, the problem of generating controllable AEs of $SSIM$ can be modeled as

$$\begin{gathered} F(X_\tau^*) \neq F(X_0) \\ \text{s.t. } X_\tau^* \in \mathbb{H}_p = \{X : SSIM(X_\tau^*, X_0) = \varepsilon_\tau^*\} \end{gathered} \tag{23}$$

According to the definition of the similarity measurement $SSIM$, for gray-scale images $x, y \in \mathbb{R}^n$ as

$$\mathrm{SSIM}(x, y) = [l(x, y)]^\varsigma \cdot [c(x, y)]^\theta \cdot [s(x, y)]^\iota \tag{24}$$

where $l(\mathbf{x}, \mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$ defines the luminance, $c(\mathbf{x}, \mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$ defines the contrast comparison function, and $s(\mathbf{x}, \mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$ defines the structure comparison function. Furthermore, μ_x, μ_y define the mean value of inputs x, y, respectively, σ_x, σ_y define the standard deviation of x, y, respectively, and σ_{xy} is the covariance between x and y. $C_1, C_2, C_3 > 0$

and $\varsigma, \theta, \iota > 0$ are constants. According to [17], when setting $\varsigma = \theta = \iota = 1$ and $C_3 = C_2/2$, Equation (24) can be simplified as,

$$\mathrm{SSIM}(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(\sigma_x^2 + \sigma_y^2 + C_2\right)} \tag{25}$$

Furthermore, according to Lagrangian constraint, we formulate Equation (26) as

$$L(X_\tau^*, \varrho) = loss(F(X_\tau^*), t)^2 + \varrho(SSIM(X_\tau^*, X_0) - \varepsilon_\tau^*)^2 \tag{26}$$

where ϱ is the Lagrangian valuable, t is the one-hot tensor of the target label and *loss* is the cross-entropy loss function as shown in Equation (27).

Cross-entropy can measure the difference between two different probability distributions in the same random variable. In machine learning, it is expressed as the difference between the target probability distribution t and the predicted probability distribution $F(X_\tau^*)$.

$$loss(F(X_\tau^*), t) = -\frac{1}{k}\sum_{i=1}^{k}[t_i \log F_i(X_\tau^*) + (1 - t_i)\log(1 - F_i(X_\tau^*))] \tag{27}$$

6. Experimental Results and Discussion

6.1. Experimental Setting

Dataset: In this work, we evaluate our methods under two widely used datasets. MNIST is a handwriting digit recognition dataset from 0 to 9, including 70,000 gray images and 60,000 for training and 10,000 for testing. CIFAR-10 [32] has 60,000 images of ten classes, including airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.

Threat model: In our paper, we generate the AEs of trained threat models. Due to limited computational resources, we train a feed-forward network with p layers and q neurons per layer. For all the networks, we use the ReLU activation function. We denote the networks as $p \times [q]$. For the MNIST dataset, we train the $3 \times (1024)$ network as a threat model. For the CIFAR-10 dataset, we train $6 \times (1024)$, $7 \times (1024)$ and $6 \times [2048]$ as threat models.

Baseline attack: For comparing our method with other adversarial attacks, we generate AEs by different attack methods. Our method can adapt to different L_p constraint measurements. In this part, due to limited computational resources, we adopt the L_2-constrained measurement. Therefore, we use other the L_2-constrained attack methods as the baseline, including SA-L_2 [10], FGSM-L_2 [11], BIM-L_2 [33], PGD-L_2 [14] and DF-L_2 [6]. We compare the performance of those attacks with our method under different ε_τ^* constraint.

6.2. Evaluation Results

6.2.1. Results of Attack Ability

We calculate the success rates of the attacks to compare the attack ability. Due to the uncontrollable ability of the perturbations of other baseline attack, we first set the ε_τ^* as 0.4, 0.8 and 1.2 for the MNIST dataset and 20, 25, 30 and 37 for the CIFAR-10 dataset, and we obtain the average perturbations of the baseline attacks under the L_2 constraint, as shown in Tables 1 and 2, and then we use their average perturbations as the ε_τ^* of our method under the same constraint and make a comparison of the success rates.

The criteria for selecting the values for the baseline for each dataset is that the value is sufficiently adequate for the baseline attack. This means that under that value, the baseline attack will not jump out of the circulation of attack in advance due to an excessively large value, which leads to the measured average perturbations not having enough correlation with that value. Meanwhile, that value will not lead to the low success rate of the baseline attack due to it being too small. Specifically, because the baseline attack cannot control the average perturbations, we first take the way of binary search that the range is $(0, 100]$

and the value interval is five and test the attack success rate and average perturbations of the baseline attack under different values. We then remove the points where either the difference between the average perturbations and that value is too large or the success rate is too low, that is, the points where that value overflows or is insufficient.

Due to the same average perturbations of the PGD-L_2 and BIM-L_2 attacks, we show their results in one table, namely Table 3 for MNIST and Table 4 for CIFAR. Furthermore, the comparison of the FGSM-L_2 attack and our method is shown in Table 5 for MNIST and Table 6 for CIFAR. As the four tables show, under the same $L_2(\varepsilon_\tau^*)$ constraint, our attack has a better attacking performance than other PGD-$L_2(\varepsilon_\tau^*)$, BIM-$L_2(\varepsilon_\tau^*)$ and FGSM-$L_2(\varepsilon_\tau^*)$ attacks.

Table 1. Table of the average perturbations of different attack methods under MNIST. We compare our method with PGD-L_2, FGSM-L_2, BIM-L_2 attacks. We denote the feed-forward networks as $p \times [q]$ and p denotes the number of layers and q is the number of neurons per layer.

Model	Attack \ ε_τ^*	0.400	0.800	1.200	1.400
MNIST 3 × (1024)	PGD-L_2	0.399	0.799	1.199	1.399
	BIM-L_2	0.399	0.799	1.199	1.399
	FGSM-L_2	0.399	0.494	1.018	1.191

Table 2. Table of the average perturbations of different attack methods under CIFAR. We compare our method with PGD-L_2, FGSM-L_2, BIM-L_2 attacks. We denote the feed-forward networks as $p \times [q]$ and p denotes the number of layers and q is the number of neurons per layer.

Model	Attack \ ε_τ^*	20.000	25.000	30.000	37.000
CIFAR 6 × (1024)	PGD-L_2	19.573	24.062	28.070	32.630
	BIM-L_2	19.573	24.062	28.070	32.630
	FGSM-L_2	19.568	24.065	28.058	32.619
CIFAR 7 × (1024)	PGD-L_2	19.703	24.130	28.002	32.636
	BIM-L_2	19.703	24.130	28.002	32.636
	FGSM-L_2	19.703	24.123	27.993	32.624
CIFAR 6 × (2048)	PGD-L_2	19.776	24.347	28.598	33.613
	BIM-L_2	19.776	24.347	28.598	33.613
	FGSM-L_2	19.774	24.345	28.593	33.604

Table 3. Table of the success rate of PGD-L_2, BIM-L_2 and our L_2 attacks under MNIST. We denote the feed-forward networks as $p \times [q]$ whilst p denotes the number of layers and q is the number of neurons per layer.

Model	Attack \ ε_τ^*	0.399	0.799	1.199	1.399
MNIST 3 × (1024)	PGD-L_2	10.27	22.48	77.31	86.26
	BIM-L_2	10.27	39.36	77.51	86.57
	Our L_2	22.40	72.96	91.62	95.01

Table 4. Table of the success rate of PGD-L_2, BIM-L_2 and our L_2 attacks under CIFAR. We compare our method with PGD-L_2, FGSM-L_2, BIM-L_2 attacks. We denote the feed-forward networks as $p \times [q]$ and p denotes the number of layers and q is the number of neurons per layer.

Model	Attack \ ε_τ^*	19.573	24.062	28.070	32.630
CIFAR 6 × (1024)	PGD-L_2	17.36	24.86	31.99	38.39
	BIM-L_2	17.36	24.86	31.99	38.39
	Our L_2	66.80	69.20	72.80	79.00
CIFAR 7 × (1024)	PGD-L_2	22.28	28.10	32.40	39.22
	BIM-L_2	22.28	28.17	32.41	39.22
	Our L_2	83.60	88.00	94.00	90.20
CIFAR 6 × (2048)	PGD-L_2	18.19	27.38	34.45	41.34
	BIM-L_2	18.19	27.38	34.45	41.34
	Our L_2	73.60	78.00	81.60	86.80

Table 5. Table of the success rate of FGSM-L_2 and our L_2 attacks under MNIST. We denote the feed-forward networks as $p \times [q]$ and p denotes the number of layers and q is the number of neurons per layer.

Model	Attack \ ε_τ^*	0.399	0.494	1.018	1.191
MNIST 3 × (1024)	FGSM-L_2	7.12	39.60	49.54	61.85
	Our L_2	22.00	40.26	90.00	91.62

Table 6. Table of the success rate of PGD-L_2, BIM-L_2 and our L_2 attacks under CIFAR. We compare our method with PGD-L_2, FGSM-L_2, BIM-L_2 attacks. We denote the feed-forward networks as $p \times [q]$ and p denotes the number of layers and q is the number of neurons per layer.

Model	Attack \ ε_τ^*	19.568	24.065	28.058	32.619
CIFAR 6 × (1024)	FGSM-L_2	17.36	25.04	31.99	38.39
	Our L_2	66.80	69.30	72.80	79.00
Model	Attack \ ε_τ^*	19.703	24.123	27.993	32.624
CIFAR 7 × (1024)	FGSM-L_2	22.28	28.10	32.40	39.22
	Our L_2	83.60	88.00	94.00	90.20
Model	Attack \ ε_τ^*	19.774	24.345	28.593	33.604
CIFAR 6 × (2048)	FGSM-L_2	18.19	27.38	34.45	41.34
	Our L_2	73.60	78.00	81.60	86.80

In addition to the attacks that have a fixed ε_τ^*, we also compare the attacks without a value to constrain the perturbations including the SA-L_2 and DF attacks. We also calculate the average perturbations of those attacks. Furthermore, then we use the same average perturbations as the ε_τ^* of our method and make a comparison in Tables 7 and 8. For MNIST, our method has a better performance than the DF and SA attacks.

Table 7. Table of the success rate of different attack methods under MNIST. We compare our method with SA-L_2 and DF attacks. We denote the feed-forward networks as $p \times [q]$ and p denotes the number of layers and q is the number of neurons per layer.

Model	Metrics \ Attacks	SA-L_2	Our L_2	DF	Our L_2
MNIST 3 × (1024)	Average Perturbations	6.020	6.020	14.935	14.935
	Success Rate	99.89	100.00	100.00	100.00

Table 8. Table of the success rate of different attack methods under CIFAR. We compare our method with SA-L_2 and DF attacks. We denote the feed-forward networks as $p \times [q]$ and p denotes the number of layers and q is the number of neurons per layer.

Model	Attacks / Metrics	SA-L_2	Our L_2	DF	Our L_2
CIFAR 6 × (1024)	Average Perturbations	41.424	41.424	41.942	41.942
	Success Rate	80.80	85.28	95.61	95.90
CIFAR 7 × (1024)	Average Perturbations	47.846	47.846	60.050	60.050
	Success Rate	82.83	85.57	94.29	95.00
CIFAR 6 × (2048]	Average Perturbations	41.356	41.356	41.717	41.717
	Success Rate	81.45	85.00	96.81	97.28

In addition to the above two small-sized datasets, the experiment also evaluates the performance of the algorithm on the larger and more complex dataset that is TinyImagenet. The dataset has 200 classes, each class has 500 pictures and we extract 200 pictures as the experimental data. For this dataset, we select the CNN model with seven layers that is denoted by 'CNN-7layer' [34] as the threat model. Furthermore, we set the ε_τ^* as $1.0, 2.0, 4.0$ and 6.0. The experiment first measures the average perturbation of the baseline attack under the selected ε_τ^*. Furthermore, it then sets the average perturbation as the ε_τ^* to compare the success rate of our algorithm and the baseline attack under that same value. The average perturbation of the baseline attack is shown in Table 9 and the comparison of the attack ability is shown in Table 10. As shown in Table 10, our algorithm has a better performance than the FGSM attack under the same ε_τ^*.

Table 9. Table of the average perturbations of different attack methods under TinyImagenet. We compare our method with FGSM-L_2 attack. We use the feed-forward network cnn-7layer as the target model.

Model	Attack / ε_τ^*	1.000	2.000	4.000	6.000
cnn-7layer	FGSM-L_2	0.999	1.999	3.999	5.999

Table 10. Table of the success rate of FGSM-L_2 and our L_2 attacks under TinyImagenet. We use the feed-forward network cnn-7layer as the target model.

Model	Attack / ε_τ^*	0.999	1.999	3.999	5.999
cnn-7layer	FGSM-L_2	50.40	64.50	75.50	79.30
	Our L_2	22.00	55.50	80.50	88.00

Furthermore, we also evaluate the attack ability of our algorithm on more complex models. We select Wide-ResNet, ResNeXt, and DenseNet as the target models and train them under the CIFAR dataset. The detail is the same as in [34]. The benchmark values we selected are $1.0, 5.0, 10.0, 30.0, 60.0$ and 80.0. Similarly, we first calculate the average perturbations of the baseline attack under that values. Then, we evaluate the results of the success rate of our algorithm and the baseline attack under the same ε^*_{tau} of our algorithm which is the same as the average perturbations calculated beforehand. Table 11 shows the average perturbations. We make a comparison of the attack ability in Table 12 and Due to Table 12, we find that under the benchmark values $5.0, 10.0, 30.0, 60.0$ and 80.0, our algorithm performs better than the FGSM attack. However, under the 1.0, in the Wide-ResNet and ResNeXt, the FGSM attack performs better.

Table 11. Table of the average perturbations of different attack methods under Cifar. We compare our method with FGSM-L_2 attack. We use the feed-forward networks Wide-ResNet, ResNeXt and DenseNet as the target models.

Model	Attack \ ε^*_τ	1.000	5.000	10.000	30.000	60.000	80.000
Wide-ResNet	FGSM-L_2	0.999	4.999	9.999	29.999	59.999	79.999
ResNeXt	FGSM-L_2	0.999	4.999	9.999	29.999	59.999	79.999
DenseNet	FGSM-L_2	0.999	4.999	9.999	29.999	59.999	79.999

Table 12. Table of the success rate of FGSM-L_2 and our L_2 attacks under Cifar. We use the feed-forward networks Wide-ResNet, ResNeXt and DenseNet as the target models.

Model	Attack \ ε^*_τ	0.999	4.999	9.999	29.999	59.999	79.999
Wide-ResNet	FGSM-L_2	28.50	44.50	51.00	68.00	87.50	87.00
	Our L_2	25.00	47.50	66.00	90.50	99.50	100.00
ResNeXt	FGSM-L_2	23.50	30.50	34.00	51.50	67.50	74.00
	Our L_2	21.50	41.00	51.50	87.50	93.50	90.00
DenseNet	FGSM-L_2	25.50	33.00	36.00	43.50	53.50	55.50
	Our L_2	30.00	38.50	51.00	70.00	77.00	77.00

6.2.2. Results of *SSIM* Constraint under Different ε^*_τ

In this part, we evaluate our method described in Section 5. Due to there being no work devised for the same purpose as our method, we only show the results of our method without any comparison with others. We show the controllable ability under the *SSIM* constraint of our method and record its success rate in Table 13. We also show the adversarial images under different *SSIM* constraints in Figure 3.

Table 13. Table of the controllable ability and attack ability of our method under the *SSIM* constraint. The perturbation coefficient of the attack is marked in brackets as SSIM-ε^*_τ. We denote the feed-forward networks as $p \times [q]$ and p denotes the number of layers and q is the number of neurons per layer.

Dataset / Attack	MNIST 3 × (1024)		CIFAR 6 × (1024)		CIFAR 7 × (1024)		CIFAR 6 × (2048)	
	SSIM	SR	SSIM	SR	SSIM	SR	SSIM	SR
Our SSIM (0.5)	0.500	100.00	0.500	100.00	0.500	100.00	0.500	100.00
Our SSIM (0.7)	0.700	96.60	0.700	100.00	0.700	100.00	0.700	100.00
Our SSIM (0.9)	0.900	42.00	0.900	31.00	0.900	36.50	0.900	36.00

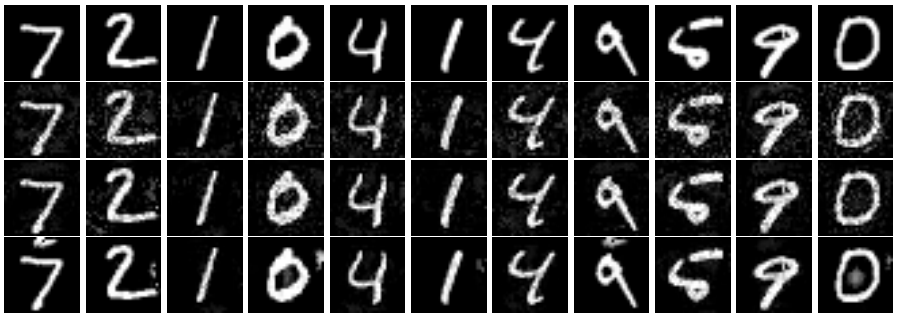

Figure 3. Figure of the images of the AEs generated under the *SSIM* constraint with different ε^*_τ. The first line is the clean images and the second line shows the adversarial images under 0.5 constraint. The third line is the adversarial images under 0.7 constraint. The last line is the adversarial images under 0.9 constraint.

6.2.3. Results of $\alpha \neq 0$ under L_2 Constraint

In this section, we discuss the results of our method under L_2 with the $\alpha \neq 0$ constraint. Through the different α, we can not only generate the controllable AEs but also improve the perceptual visual quality under the same ε_τ^* constraint. When the ε_τ^* under the L_2 constraint is large, the perceptual visual quality is poor. In order to adapt to this situation, our paper devises α to improve the perceptual visual quality. However, there is a trade-off between the visual quality of the AEs and their success rate. Figure 4 shows the SSIM value of AEs under different ε_τ^* with different α. As it shows, the SSIM value increases with the α increasing under the same ε_τ^* constraint. Furthermore, we can see that with the increasing ε_τ^*, the SSIM value has a trend of decreasing under the same α. This means that the visual quality becomes poorer when more perturbations are added to the inputs, which is in line with the intuition of the AEs. Meanwhile, the SSIM value rises rapidly before $\alpha = 1.0$ under the same ε_τ^* constraint; after that, its trend tends to be flatter.

Figure 5 shows the success rate of AEs under different ε_τ^* with different α. As it shows, the success rate decreases with the α increasing under the same ε_τ^* constraint. Moreover, with the increasing ε_τ^*, the success rate increases under the same α constraint. It is also consistent with the general nature of the AEs that when more perturbations are added, the probability of a successful attack becomes greater. Furthermore, the $\alpha = 1.0$ still tends to be a boundary that before $\alpha = 1.0$, the success rate decreases slower and then it decreases faster when $\varepsilon_\tau^* = 3.00$ and $\varepsilon_\tau^* = 2.50$. However, it has a nearly consistent trend of decreasing with $\varepsilon_\tau^* = 1.00$, $\varepsilon_\tau^* = 1.50$ and $\varepsilon_\tau^* = 2.00$. It means that when the perturbations remain small, excessive attention to visual quality will lead to a greater loss of attack success rate. Therefore, it corresponds to the actual meaning of the parameter α that only needs to be set to $\alpha \neq 0$ when ε_τ^* is large. We set $\alpha = 1$ and compare the results between $\alpha = 0$ and $\alpha = 1$ in Table 14.

Figure 6 shows the time of generating AEs under different ε_τ^* with different α. As it shows, the time decreases with the α increasing under the same ε_τ^* constraint. Moreover, with the ε_τ^* increasing, the time increases under the same α constraint.

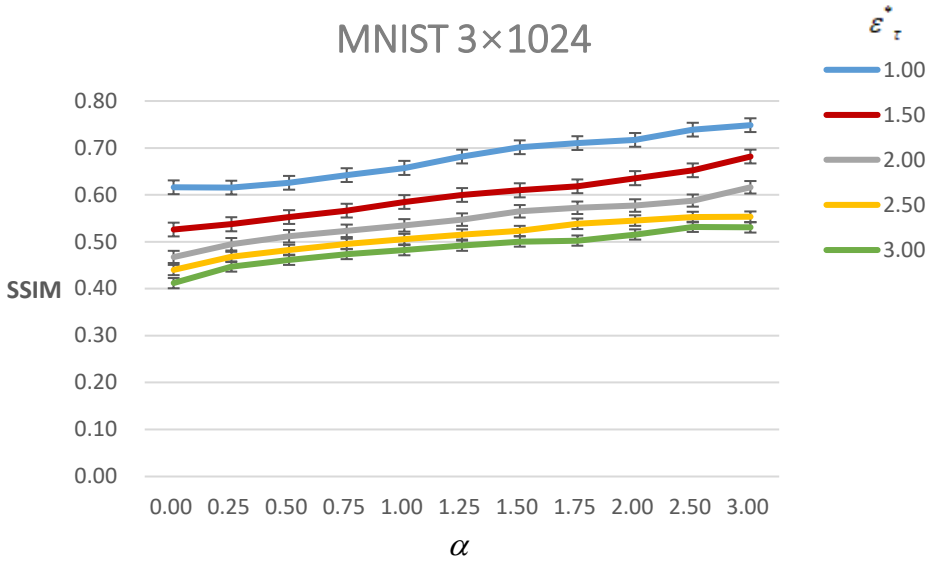

Figure 4. Figure of the SSIM value of AEs of MNIST under different ε_τ^* with different α. The line with different color means different ε_τ^*.

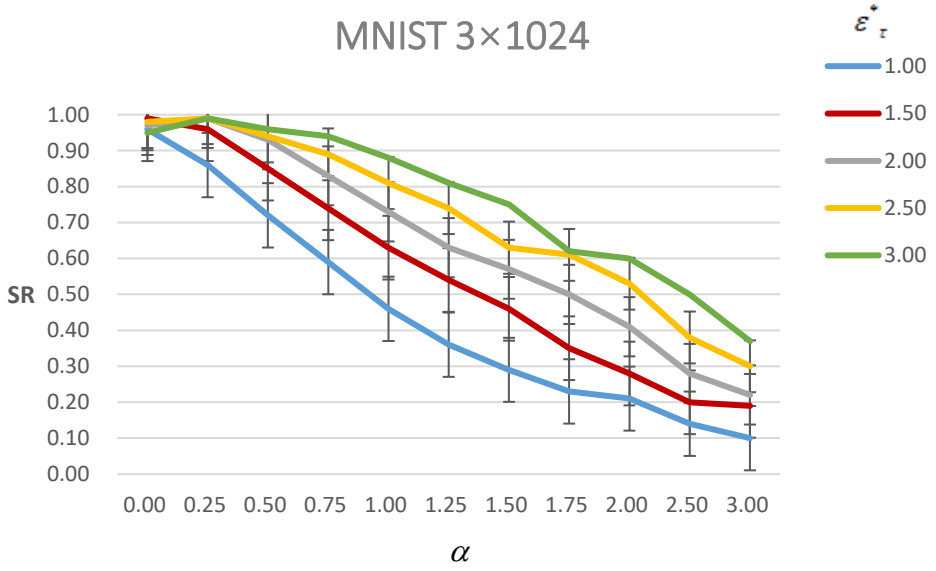

Figure 5. Figure of the success rate (SR) of AEs of MNIST under different ε_τ^* with different α. The line with a different color means different ε_τ^*.

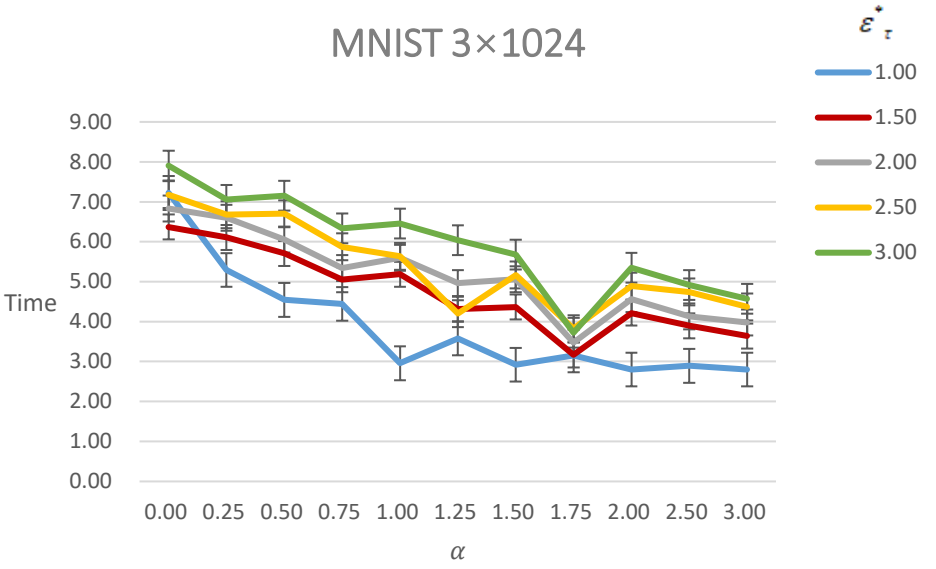

Figure 6. Figure of time of generating AEs of MNIST under different ε_τ^* with different α. The line with a different color means different ε_τ^*.

Table 14. Table for the AEs under $\alpha = 0$ and $\alpha = 1$ of the L_2 constraint. We denote the feed-forward networks as $p \times [q]$ and p denotes the number of layers and q is the number of neurons per layer. The *AP*, *SR*, *SSIM* and *Time* denote the average perturbations, the success rate, the SSIM value between the original image and adversarial image and the time taken to generate AEs, respectively.

Model	Attack / Metrics	$\varepsilon_\tau^* = 0.50$		$\varepsilon_\tau^* = 1.00$		$\varepsilon_\tau^* = 1.20$		$\varepsilon_\tau^* = 2.40$	
		$\alpha = 0$	$\alpha = 1$	$\alpha = 0$	$\alpha = 1$	$\alpha = 0$	$\alpha = 1$	$\alpha = 0$	$\alpha = 1$
MNIST $3 \times (1024)$	AP	0.50	0.50	1.00	1.00	1.20	1.20	2.40	2.40
	SR	38.71	12.9	85.16	43.5	91.62	51.9	98.19	74.7
	SSIM	0.77	0.82	0.61	0.67	0.57	0.62	0.42	0.52
	Time	8.33	9.33	7.22	9.19	6.82	9.34	7.07	9.31

7. Conclusions

Aiming at the two fundamental problems of generating the minimum AEs, we first define the concept of the minimum AEs and prove that generating the minimum AEs is an NPC problem. Based on this conclusion, we then establish a new third kind of optimization model that takes the successful attack as the target and the adversarial perturbations equal the lower bound of the minimum adversarial distortion plus a controllable approximation. This model generates the controllable approximation of the minimum AEs. We give a heuristic solution method of that model. From the theoretical analysis and experimental verification, our model's AEs have a better attack ability and can generate more accurate and controllable AEs to adapt to different environmental settings. However, the method in this paper of the model does not perfectly determine the solution of the model, which will be the focus of future research.

Author Contributions: Supervision, F.L. and X.Y.; Writing—review & editing, Z.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1. Proof of NPC of Calculating the Minimum Adversarial Perturbations

Definition A1. *(3-SAT). Given a finite set of Boolean variables* $B = B_1, B_2, \ldots B_n$, $|B| = n$, *each variable takes* 0 *or* 1, *and a set of clauses* $C = C_1, C_2, \ldots C_k$, $|C| = k$, $\psi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$; *each* C_i *is a disjunctive normal form composed of three variables, that is,* $Z_1 \vee Z_2 \vee Z_3$. *Question: given a Boolean variable set X and clause set C, whether there is a true value assignment so that C is true and then each clause is true.*

Theorem A1. *Given a neural network F, a distribution* $\aleph \subset \mathbb{R}^n$, *and a distance measurement* $D : R^n \times R^n \to R$ *between X and* X_0, *a point* $X_0 \in \mathbb{R}^n$, *searching for a minimum adversarial example of* X_0 *is an NPC problem.*

Proof. We now prove Theorem A1. We first reduce the problem into a decision problem, and then according to the definition of an NPC problem, we prove that the problem belongs to the type of NP problem, and finally, we prove that a known NPC problem can be reduced to the decision problem in polynomial time.

We first reduce the problem of finding the minimum AEs into a series of decision problems. Though many important problems are not decision problems when they appear in the most natural form, they can be reduced to a series of decision problems that are easier to study, for example, the coloring problem of a graph. When coloring the vertices of a graph, we need at least n colors to make any two adjacent vertices have different colors. Then, it can be transformed into another question. Can we color the vertices of the graph with no more than m colors, $m \in N+$? The first m value in the set that makes the problem solvable is the optimal solution of the coloring problem. Similarly, we can also transform the optimization problem of finding the minimum AEs into the following series of decision problems: given the precision of perturbations $\delta\varepsilon$, and initial perturbations ε_1, $\varepsilon_i = \varepsilon_{i-1} + \delta\varepsilon$, whether we can use the perturbations ε, $\varepsilon \in \varepsilon_1, \varepsilon_2, ..., \varepsilon_i, ...$ to make the inequality $F(X_0 +^* \varepsilon_i) \neq F(X_0))$ true, and the first value in the sequence that makes the inequality true is the optimal solution of the optimization problem.

The decision problem is reduced and formalized as follows. For the neural network attribute, $\varphi = \varphi_1(X) \wedge \varphi_2(Y)$, φ_1 is the mapping from the AEs generated by the AEs generation function to label 1, that is, $\varphi_1(X) : (X +^* \varepsilon) \to 1$. φ_2 is the mapping from the AEs generated by the AEs generation function to $0, 1$, that is, $\varphi_2(X) : (X +^* \varepsilon) \to 0, 1$. When $F(X +^* \varepsilon) \neq F(X)$, its value is 1. When $F(X +^* \varepsilon) = F(X)$, its value is 0. When there is an assignment $\alpha(X) = X +^* \varepsilon_i, \varepsilon \in \varepsilon_1, \varepsilon_2, ..., \varepsilon_i, ..., \alpha(Y)$ is the output of the neural network and determines whether the value of the attribute φ is true.

Obviously, it is an NP problem. In the guessing stage, given any perturbations ε, assuming the ε is a candidate solution of the decision problem. Furthermore, then in the verification stage, since the process of inputting perturbations ε and samples X_0 to the neural network and then outputting the results can be completed in polynomial time, it is polynomial in the verification stage. Therefore, the solution to the decision problem is an uncertain polynomial algorithm. Furthermore, according to the definition of the NP problem, the decision problem is an NP problem.

Finally, we prove that any problem in NP can be reduced to the decision problem in polynomial time. Due to the transitivity of polynomial simplification, we can prove a known NPC problem: the $3 - SAT$ problem can be transformed into the decision problem in polynomial time and then complete this proof.

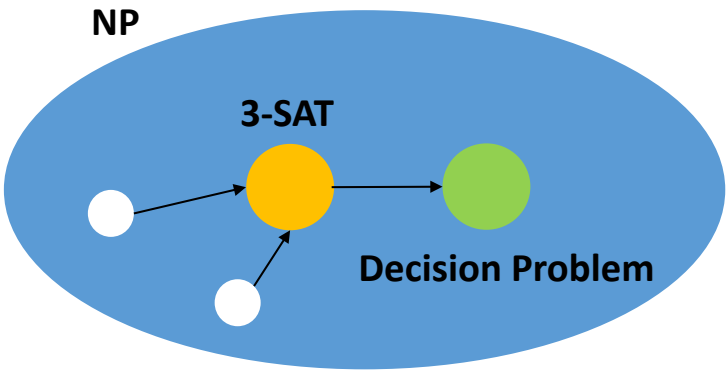

Figure A1. Figure for the transitivity of polynomial simplification.

Since the $3 - SAT$ problem is an NPC problem, according to the definition of an NPC problem—that is, any problem in the set of NP problems that can be reduced to an $3 - SAT$ problem in polynomial time—if the $3 - SAT$ problem can be reduced to the aforementioned decision problem of searching for the AEs, according to transitivity, any problem in NP can be reduced to that decision problem in polynomial time and it can be proven that the decision problem is an NP-hard problem. We then prove how the $3 - SAT$ problem is Turing reduced to the decision problem.

According to the definition of an $3 - SAT$ problem, given the Ternary satisfiability formula $\psi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ on the variable set $X = \{x_1, x_2, \ldots, x_k\}$, each clause is a disjunction of three terms: $q_i^1 \vee q_i^2 \vee q_i^3$, where q_i^1, q_i^2 and q_i^3 are variables from X or their negative values. The problem is turned into that determining whether there is an assignment $\alpha : X \to \{0, 1\}$ to satisfy ψ, that is, whether there is an assignment α that makes all clauses C_i valid at the same time.

To simplify, we first assume that the input node q_i^1, q_i^2 and q_i^3 is a sub-statement constructed when the discrete value is 0 or 1. Then, we will explain how to relax this restriction so that the only restriction on the input nodes is that they are in the range of $[0, 1]$.

Firstly, we introduce the disjunctive tool, that is, given nodes $q_1, q_2, q_3 \in 0, 1$ and the output node is Y_i. When $q_1 + q_2 + q_3 \geq 1$, $Y_i = 1$, otherwise $Y_i = 0$. The following Figure A2 shows the situation when q_i is the variable itself (that is, it is not the negative value of the variable).

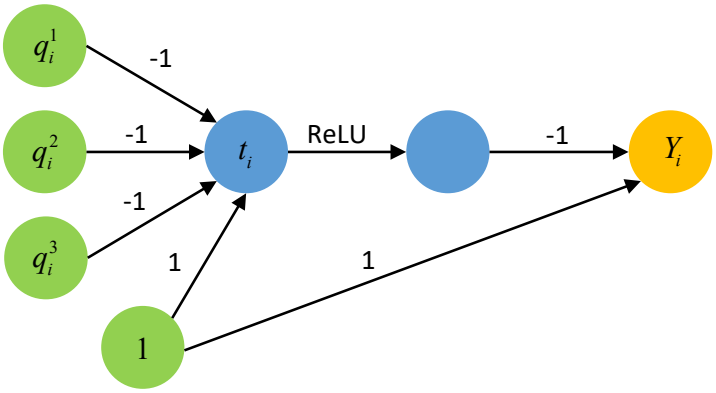

Figure A2. Figure for the disjunctive gadget when q_i are variables.

The disjunctive tool can be seen as the process of calculating Equation (A1):

$$Y_i = 1 - \max\left(0, 1 - \sum_{j=1}^{3} q_i^j\right) \tag{A1}$$

If it has one variable of input which is at least 1, then $Y_i = 1$. If all the variables of input are 0, then $Y_i = 0$. The key of the tool is that the *ReLU* function can ensure that the output Y_i remains exactly 1 even if multiple inputs are set to 1.

For processing any negative item $q_i^j = 1 - x_j \equiv \neg x_j$, we introduce a negative tool before inputting the negative item into the disjunctive tool, as shown in Figure A3a.

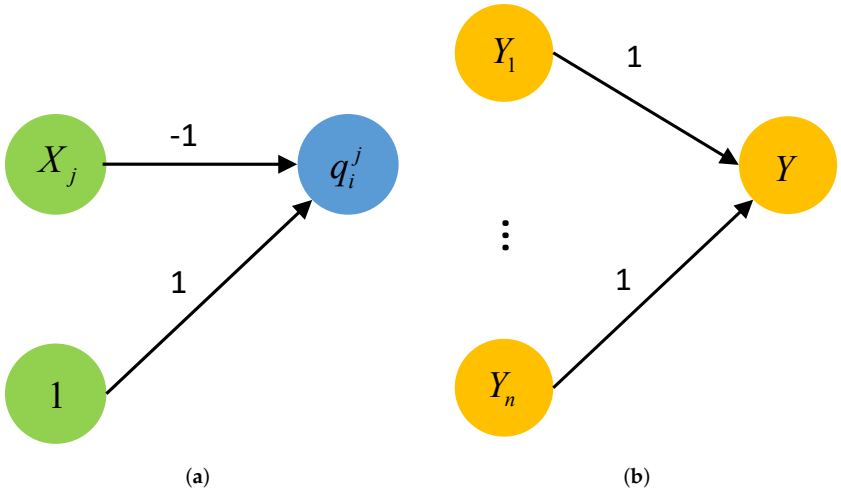

Figure A3. Figure for negative disjunction and conjunction gadgets. (**a**) Figure representing a negative disjunction gadget. (**b**) Figure representing a conjunction gadget.

The tool that calculates $1 - x_j$ and then continues to calculate is the aforementioned disjunctive tool. The last step involves a conjunction widget, as shown in Figure A3b.

Assuming that all nodes $Y_1, \ldots, Y_n$ are in the range of $0, 1$, we require node Y in the range of $[n, n]$. Obviously, this requirement only holds if all nodes are 1.

Lastly, in order to check if all the clauses $C_1, \ldots, C_n$ are satisfied at the same time, we construct a conjunction gadget(using the negative value tool as input as needed) and combine it with a conjunction gadget, as shown in Figure A4.

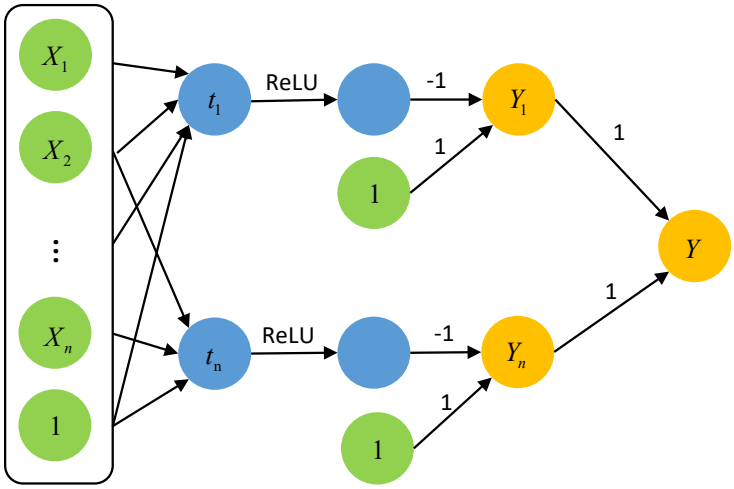

Figure A4. Figure for the 3-SAT-DNN conjunction gadget.

The input variable is mapped to each t_i node according to the definition of clause C_i, that is, $t_i \to C_i$. According the above discussion, if the clause C_i is satisfied, then $Y_i = 1$; otherwise, $Y_i = 0$. Therefore, the node Y is the range of $[n, n]$ if and only if all the clauses are satisfied at the same time. Thus, an assignment $\alpha : X \to 0, 1$ of input satisfies the constraint between the input and the output of neural networks if and only if that assignment also satisfies the original item $\psi = C_1 \wedge C_2 \wedge \ldots \wedge C_n$.

The above construction is based on the assumption that the input node takes values from discrete values $0, 1$, that is, $\alpha : X \to 0, 1$. However, it does not accord with the assumption that $\psi_1(X)$ is the conjunction of linear constraints. We will then prove how to relax the restriction to make the original proposition true.

Letting ε is a very small number. We suppose that each variable X_i is in the range of $[0, 1]$ but ensure that any feasible solution satisfies $X \in [0, \varepsilon]$ or $X \in [1 - \varepsilon, 1]$. We add an auxiliary gadget to each input variable X_i, that is, using the *ReLU* function node to calculate Equation (A2) as follows:

$$\max(0, \epsilon - X) + \max(0, X - 1 + \epsilon) \tag{A2}$$

Furthermore, the output node of Equation (A2) is required to be within the range $[0, \varepsilon]$. This expression can directly indicate that when $X \in [0, \varepsilon]$ or $X \in [1 - \varepsilon, 1]$, it is true for $X \in [0, 1]$.

The disjunctive expression in our construction is Equation (A1). The value of its disjunctive expression changes with the inputs. If all inputs are in $[0, \varepsilon]$ or $[1 - \varepsilon, 1]$, then at least one input is in $[1 - \varepsilon, \varepsilon]$ and then the end output node of each disjunctive gadget Y_i will no longer use discrete values $[0, 1]$ but will be in $[0, 3\varepsilon]$.

If at least one node of each input clause is in the range $[1 - \varepsilon, 1]$, then all Y_i nodes will be in $[1 - \varepsilon, 1]$ and Y will be in $[n(1 - \varepsilon), n]$. However, if at least one clause does not have a node in the range $[1 - \varepsilon, 1]$, Y will be less than $n(1 - \varepsilon)$ (when $\epsilon < \frac{1}{n+3}$). Therefore, keeping the requirements $Y \in [n(1 - \epsilon), n]$ true, if and only if ψ is satisfied, its input and output will be satisfied, and the satisfied assignment can be constructed by making each $X_i \in [0, \varepsilon] = 0$ and each $X_i \in [1 - \varepsilon, 1] = 1$. □

Appendix A.2. Analysis of SSIM Constraint Method

We also try to directly calculate the adversarial perturbations as the L_p constraint of our method. However, we find that it is difficult to perform the same operation under the *SSIM* constraint. The analysis is as follows. According to Equation (25) and substituting

its inputs x and y as $X^*_{\varepsilon^*_\tau}$ and X_0, respectively, Equation (25) can be seen as the product of two parts [35]. That is:

$$\mathrm{SSIM}(X^*_{\varepsilon^*_\tau}, X_0) = S_1(X^*_{\varepsilon^*_\tau}, X_0) S_2(X^*_{\varepsilon^*_\tau}, X_0) \tag{A3}$$

where:

$$\begin{aligned} S_1(X^*_{\varepsilon^*_\tau}, X_0) &= \frac{2\mu_{X^*_{\varepsilon^*_\tau}}\mu_{X_0} + C_1}{\mu^2_{X^*_{\varepsilon^*_\tau}} + \mu^2_{X_0} + C_1} = f(\mu_{X^*_{\varepsilon^*_\tau}}, \mu_{X_0}) \\ S_2(X^*_{\varepsilon^*_\tau}, X_0) &= c(X^*_{\varepsilon^*_\tau}, X_0) s(X^*_{\varepsilon^*_\tau}, X_0) = \frac{2\sigma_{X^*_{\varepsilon^*_\tau} X_0} + C_2}{\sigma^2_{X^*_{\varepsilon^*_\tau}} + \sigma^2_{X_0} + C_2} = g(X^*_{\varepsilon^*_\tau} - \mu_{X^*_{\varepsilon^*_\tau}}, X_0 - \mu_{X_0}) \end{aligned} \tag{A4}$$

Therefore, the $\mathrm{SSIM}(X^*_{\varepsilon^*_\tau}, X_0)$ can be divided into the f function of $\mu_{X^*_{\varepsilon^*_\tau}}$ and μ_{X_0} and the g function of $X^*_{\varepsilon^*_\tau} - \mu_{X^*_{\varepsilon^*_\tau}}$ and $X_0 - \mu_{X_0}$. In order to solve the condition of Equation (23), i.e., the product of the two functions needs to be a constant, we try to transform prime factorization to decompose d into the product of two values. Furthermore, the input X_0 is given so a set of prime factorization can be seen as solving a $X^*_{\varepsilon^*_\tau}$ that meets the criteria of Equation (A3).

However, the solutions $X^*_{\varepsilon^*_\tau}$ of the criteria of Equation (23) are not certain whether they are the AEs of the model F. Moreover, the solution of prime factorization is limited and it is a small set that meets the constraints, so it is more difficult to find AEs in that smaller set.

Appendix A.3. The Definition of the Lower and Upper Bound of a Network

We recall this definition from [23] as follows:

Definition A2. *(lower boundγ^L_y, upper bound γ^U_y). Given a neural network F, a distribution $\aleph \subset \mathbb{R}^n$, a point $X_0 \in \mathbb{R}^n$, a label y, and the output of the network under y label F_y, we say that γ^L_y and γ^U_y are the lower bound and the upper bound of the network F under the label y such that $\gamma^L_y \leq F_y(X_0) \leq \gamma^U_y$.*

According to Definition A2, we give a further explanation of Equation (5). Give a point X_0 and a perturbation ε when inputting the perturbed point X of X_0 under ε, if the upper bound of the network under the original label y is lower than the lower bound of the network under the label $y^* \neq y$, meaning that the output $F_y(X) \leq F^*_y(X)$ so that X is an adversarial example.

References

1. Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; Song, D. Robust Physical-World Attacks on Deep Learning Models. *arXiv* **2017**, arXiv:1707.08945.
2. Liu, A.; Wang, J.; Liu, X.; Cao, B.; Zhang, C.; Yu, H. Bias-Based Universal Adversarial Patch Attack for Automatic Check-Out. *ECCV* **2020**, *12358*, 395–410. [CrossRef]
3. Bontrager, P.; Roy, A.; Togelius, J.; Memon, N.; Ross, A. DeepMasterPrints: Generating masterprints for dictionary attacks via latent variable evolution. In Proceedings of the 2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems, BTAS 2018, Redondo Beach, CA, USA, 22–25 October 2018. [CrossRef]
4. Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; Fergus, R. Intriguing properties of neural networks. *arXiv* **2013**, arXiv:1312.6199.
5. Carlini, N.; Wagner, D. Towards Evaluating the Robustness of Neural Networks. In Proceedings of the IEEE Symposium on Security and Privacy, San Jose, CA, USA, 22–26 May 2016; pp. 39–57. [CrossRef]
6. Moosavi-Dezfooli, S.M.M.; Fawzi, A.; Frossard, P. DeepFool: A simple and accurate method to fool deep neural networks. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2574–2582. [CrossRef]
7. Carlini, N.; Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Co-Located with CCS 2017, AISec 2017, Dallas, TX, USA, 3 November 2017; pp. 3–14. [CrossRef]

8. Moosavi-Dezfooli, S.M.; Fawzi, A.; Fawzi, O.; Frossard, P. Universal adversarial perturbations. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; pp. 86–94. [CrossRef]
9. Athalye, A.; Carlini, N.; Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, 10–15 July 2018; Volume 1, pp. 436–448.
10. Andriushchenko, M.; Croce, F.; Flammarion, N.; Hein, M. Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2020; Volume 12368, pp. 484–501. [CrossRef]
11. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015; pp. 1–11.
12. Papernot, N.; Mcdaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z.B.; Swami, A. The limitations of deep learning in adversarial settings. In Proceedings of the 2016 IEEE European Symposium on Security and Privacy, EURO S and P 2016, Hong Kong, China, 21–24 March 2016; pp. 372–387. [CrossRef]
13. Kurakin, A.; Goodfellow, I.J.; Bengio, S. Adversarial examples in the physical world. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings, Toulon, France, 24–26 April 2017; pp. 1–14.
14. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv* **2017**, arXiv:1706.06083
15. Su, J.; Vargas, D.V.; Sakurai, K. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Trans. Evol. Comput.* **2019**, *23*, 828–841. [CrossRef]
16. Hameed, M.Z.; Gyorgy, A. Perceptually Constrained Adversarial Attacks. *arXiv* **2021**, arXiv:2102.07140.
17. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef] [PubMed]
18. Gragnaniello, D.; Marra, F.; Verdoliva, L.; Poggi, G. Perceptual quality-preserving black-box attack against deep learning image classifiers. *Pattern Recognit. Lett.* **2021**, *147*, 142–149. [CrossRef]
19. Zhao, Z.; Liu, Z.; Larson, M. Towards Large Yet Imperceptible Adversarial Image Perturbations with Perceptual Color Distance. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 1036–1045. [CrossRef]
20. Weng, T.W.; Zhang, H.; Chen, P.Y.; Yi, J.; Su, D.; Gao, Y.; Hsieh, C.J.; Daniel, L. Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach. In Proceedings of the 6th International Conference on Learning Representations ICLR, Vancouver, BC, Canada, 30 April 30–3 May 2018.
21. Weng, T.w.; Zhang, H.; Chen, P.y.; Lozano, A.; Hsieh, C.j.; Daniel, L. On Extensions of CLEVER: A Neural Network Robustness Evaluation Algorithm. *arXiv* **2018**, arXiv:1810.08640.
22. Weng, T.W.; Zhang, H.; Chen, H.; Song, Z.; Hsieh, C.J.; Boning, D.; Dhillon, I.S.; Daniel, L. Towards fast computation of certified robustness for relu networks. *arXiv* **2018**, arXiv:1804.09699v4.
23. Zhang, H.; Weng, T.w.; Chen, P.y.; Hsieh, C.j.; Daniel, L. Efficient Neural Network Robustness Certification with General Activation Function. *arXiv* **2018**, arXiv:1811.00866v1.
24. Boopathy, A.; Weng, T.W.; Chen, P.Y.; Liu, S.; Daniel, L. CNN-Cert: An efficient framework for certifying robustness of convolutional neural networks. In Proceedings of the the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19), Honolulu, HI, USA, 27 January–1 February 2019; pp. 3240–3247. [CrossRef]
25. Sinha, A.; Namkoong, H.; Duchi, J. Certifying some distributional robustness with principled adversarial training. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018—Conference Track Proceedings, Vancouver, BC, Canada, 30 April–3 May 2018; pp. 1–49.
26. Kolda, T.G.; Bader, B.W. Tensor decompositions and applications. *SIAM Rev.* **2009**, *51*, 455–500. [CrossRef]
27. Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Tramèr, F.; Prakash, A.; Kohno, T.; Song, D. Physical adversarial examples for object detectors. In Proceedings of the 12th USENIX Workshop on Offensive Technologies, WOOT 2018, co-located with USENIX Security 2018, Baltimore, MD, USA, 13–14 August 2018.
28. Haykin, S.; Kosko, B. GradientBased Learning Applied to Document Recognition. *Intell. Signal Process.* **2010**, 306–351. [CrossRef]
29. CIFAR-10—Object Recognition in Images@Kaggle. Available online: https://www.kaggle.com/c/cifar-10 (accessed on 20 January 2022)
30. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015.
31. Ahmed, N.; Natarajan, T.; Rao, K.R. Discrete Cosine Transform. *IEEE Trans. Comput.* **1974**, *C-23*, 90–93. [CrossRef]
32. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; Science Department, University of Toronto: Toronto, ON, Canada, 2009; pp. 1–60. [CrossRef]
33. Kurakin, A.; Goodfellow, I.J.; Bengio, S. Adversarial machine learning at scale. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017- Conference Track Proceedings, Toulon, France, 24–26 April 2017.

34. Xu, K.; Shi, Z.; Zhang, H.; Wang, Y.; Chang, K.W.; Huang, M.; Kailkhura, B.; Lin, X.; Hsieh, C.J. Automatic perturbation analysis for scalable certified robustness and beyond. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, BC, Canada, 6–12 December 2020.
35. Brunet, D.; Vrscay, E.R.; Wang, Z. On the mathematical properties of the structural similarity index. *IEEE Trans. Image Process.* **2012**, *21*, 1488–1495. [CrossRef]