Further Understanding Towards Sparsity Adversarial Attacks

Mengnan Zhao[1], Xiaorui Dai[1], Bo Wang[1(✉)], Fei Yu[1], and Fei Wei[2]

[1] School of Information and Communication Engineering, Dalian University of Technology, Dalian, Liaoning 116087, China
bowang@dlut.edu.cn

[2] Department of Electrical Engineering, Arizona State University, Tempe, AZ 85281, USA

Abstract. The emergence of adversarial attacks validated the vulnerability of neural networks. Recently, highly sparse adversarial examples gradually attracted the attention of researchers for their ability in explaining what the neural networks have learned from datasets. For further understanding the sparsity adversarial attacks, we propose two different white-box techniques, the BPs, and binary fitting. BPs generates adversarial examples by validating the existence of adversarial samples in a top-down way, that is, decreasing the upper border of tempered pixel positions step by step. Additionally, the binary fitting method approximates the $\mathcal{L}_0$ distance to search for the optimal adversarial example considering the 0-norm function is non-convex. Experimental results illustrate that the proposed methods exhibit superior or competitive performance to the state-of-the-art attacks.

Keywords: Neural networks · Sparse adversarial examples · BPs · The binary fitting method

1 Introduction

Deep neural networks (DNNs) have achieved amazing performance in extensive vision tasks such as image classification [12,15,29] and object detection [11,18]. However, recent researches [10,30] illustrate that well-trained models may give wrong decisions to craft adversarial examples, whereas only tiny perturbations are added to these examples, The vulnerability of DNNs limits its applications in critical fields, such as the self-driving system [16,36], the forensic task [8,37]. Adversarial examples, especially samples generated by the sparse adversarial attack [3,6,25], are attracting attention from researchers, as they are helpful to understand deep learning. For instance, the modified pixel positions in sparsity adversarial attacks show great influence on model predictions than the rest positions.

Recent studies have introduced the adversarial attack to various tasks, e.g. object detection [34], neural language processing [4]. According to the access degree to the target model, attack methods are categorized into white-box attacks that allow attackers to utilize all model information and black-box attacks that only classification results

M. Zhao and X. Dai—Contributed equally to this work.

X. Sun et al. (Eds.): ICAIS 2022, CCIS 1586, pp. 200–212, 2022.
https://doi.org/10.1007/978-3-031-06767-9_17

are available. Additionally, several norm functions are adopted to measure the distance between benign samples and adversarial examples considering various requirements, $\mathcal{L}_0$, $\mathcal{L}_1$, and $\mathcal{L}_2$. The adversarial examples constrained by $\mathcal{L}_2$ norm [2,5] that measures the overall disturbance in the attack process show better human imperceptible performance. Highly sparse adversarial examples ($\mathcal{L}_0$ norm [26]) are valuable for their ability in explaining the model and imperceptible spot-like disturbances. In this work, we aim at improving the performance of white-box sparsity adversarial attacks.

We construct two sparsity adversarial attacks. These attacks tend to modify the smallest image pixels to successfully attack the model. The paper has the following contributions. 1) Based on the C&W attack, we propose BPs, which decrease its possibility of falling into the local minimum value. Motivated by (Project Gradient Descent) PGD attack [22], BPs have added the batch searching, initial perturbation, and multiple strides to $\mathcal{L}_0$ attack. 2) Compared with search-based attacks such as C&W attack, the attacks based on the gradient propagation are more powerful in explaining the model. Therefore, following several formulas in $\mathcal{L}_0$, we construct the approximate function of $\mathcal{L}_0$. Based on this, we design several loss functions to realize the sparsity adversarial attack. For instance, $\mathcal{L}_{\mathbf{C+P}}$ separates the size and position of the perturbation into two vectors and then constrains each vector by the approximate function. 3) The proposed sparsity adversarial attacks outperform or are competitive to previous attacks.

The paper is organized as the following. The BPs attack is derived in part Sect. 3.2. In part Sect. 3.3, we first give several formulae of $\mathcal{L}_0$ and then describe how the binary fitting method (BFM) realizes the sparsity adversarial attack. In the fourth part, we compare the BPs and BFM with the state-of-the-art attacks, where also include the ablation analysis for various constraints.

2 Related Work

2.1 Attacking Methods

Adversarial attacks are categorized into the white-box attack and the black-box attack based on the accessibility of the attacked model. Specifically, white-box attacks make an assumption that attackers can access all knowledge about the attacked model. For instance, model architectures, model weights, and selected hyper-parameters. Inversely, black-box attacks mean attackers know nothing about attacked models unless classification results. Whether it is the white-box attack or the black-box attack, perturbations added to the benign samples are imperceptible to humans. Gradient and iteration attack [16] is the most commonly used attack mode. We have listed several common adversarial attacks in the following subsection.

Goodfellow et al. [10] proposed the Fast Gradient Sign Method (FGSM). FGSM only modifies pixels once for each sample according to the direction of gradient backward. Therefore, FGSM is useful for untargeted attacks even though it cannot guarantee a successful attack. Since the attack difficulty of FGSM towards un-targeted and targeted attack intention, [16] proposed the Basic Iteration Methods (BIM). Based on FGSM, BIM realizes the attack by multi-step optimization. Meanwhile, BIM restricts the max scope of perturbations for better visualization performance. Besides, [9] introduced momentum iteration to generate adversarial examples considering the convergence speed. Sarkar et al. [28] designed the *UPSET* and *ANGRI* methods that attack

multiple classifiers simultaneously for improving the generalization performance of attacks. Carlini and Wagner (C&W) attack [3] obtained the crafted adversarial perturbations by limiting different norm functions. Generative Adversarial Networks(GAN) was used to generate adversarial examples [7], which learned the distribution of adversarial examples generated by FGSM and establish a generation model, thus generating corresponding adversarial examples in batches.

For special tasks like semantic segmentation and object detection, Xie et al. [34] adopted the Dense Adversary Generation (DAG) to design adversarial examples. [31] proposed the targeted adversarial attack for black-box audio systems. Ruiz et al. [27] first consider defeating the deepfake generator by adding the perturbations to the benign samples. After attacking, the facial manipulation system outputs the disrupted sample instead of the expected manipulated image. Y. Wang et al. [33] proposed an adversarial attack method for face recognition, which fool face recognition both digital domain and physical domain. [35] proposed a character level text adversarial sample generation model. Via finding important keywords and using five interference strategies, The model made the emotion of the sentences to change for artificial intelligence systems without influencing people to read and understand.

2.2 Defense Methods

Considering the application scenarios, defense methods are divided into active defense and passive defense. The former is proposed to prevent the generation of adversarial examples while the latter is adopted to detect the generated adversarial examples. Gradient mask and adversarial training are the most common methods of active defense. Networks after crafted processings lose their gradient backward information,e.g. the distillation network. Both [10] and [13] generated adversarial samples in the training stage and use them as training data in the training process. Experimental results show adversarial training increases the network robustness to various adversarial attacks. [32] proposed an adversarial joint model, MDJM-ADV, for dialog systems to predict domain, intent, and entity using a single LSTM cell to reduce the risk of downstream error propagation that is present in the typical pipelined approach Since the prediction for the bounding box is not very accurate, [21] indicated that standard detectors would not be fooled by physical adversarial stop signs.

The passive defense includes various detection methods, such as image reconstruction, which plays the role in the testing phase. [1] proposed a deep image restoration model that eliminates the perturbation of adversarial examples, the recovered original examples can be classified correctly. Metzen et al. [24] assigned the detector for the trained classification network to distinguish the adversarial examples from benign samples. Similarly, lu et al. [20] extracted the binary threshold output from each ReLU layer as the features to train the adversarial detector. The task of image reconstruction means to transform the adversarial examples to clean images, while should not affect the normal classification performance. However, the existing methods seem only to break the crafted correlations between the pixel of the adversarial example, such as Magnet [23]. The researchers first determine whether the inputs are adversarial examples and then introduce the Gaussian noise to disrupt the crafted adversarial distribution.

3 Proposed Methods

3.1 Background

Given a trained network $\mathcal{S}$, a specific distance metric $\| \cdot \|_0$, inputs $\mathbf{I}$ and a radius D, the norm ball $\mathcal{B}(\mathcal{S}, \mathbf{I}, \| \cdot \|_0, D)$ is a solution for $\mathcal{L}_0$ attack such that $\mathcal{B}(\mathcal{S}, \mathbf{I}, \| \cdot \|_0, D)$ = $\{\mathbf{I}' = \emptyset | \parallel \mathbf{I}' - \mathbf{I} \parallel_0 \leq D\}$. The optimal value D_{op} of D means that no adversarial example is possible for perturbation of less than D_{op} pixels change, which is also called the least pixel attack.

By adding tiny perturbations σ to benign samples $\mathbf{I}$, the well-trained models, such as the classification network, will be deceived. The targeted attack based on $\mathcal{L}_0$ norm is expressed as,

$$\min_{\sigma} \mathcal{L}_0 = \mathcal{L}\{ \mathcal{S}(\mathbf{I} + \sigma) = \ell | \theta_{\mathcal{S}} \} + \alpha \cdot \parallel \sigma \parallel_0 \tag{1}$$

where $\mathcal{S}$ denotes the target model, such as the trained classification network, with fixed parameters $\theta_{\mathcal{S}}$. $\mathcal{L}$ and ℓ represents the loss function and expected targets. α is the hyper-parameter that is used to balance the attack rate and perturbation degree, which is assigned by ablation experiments.

Following the C&W attack, we introduce the variable $\mathbf{w}$ that within infinite range to calculate perturbations σ, which eliminates the problem introduced by limiting pixel values of adversarial examples $\mathbf{I}'$ to [0,1].

$$\sigma = \frac{1}{2}(\tanh(\mathbf{w}) + 1) - \mathbf{I}, -1 \leq \tanh(\mathbf{w}) \leq 1$$

Actually, adversarial examples are iteratively updated, which are separated into the pixel value map and pixel position map.

$$\mathbf{I}'_i = \left[\frac{\tanh\left(\mathbf{w} + t'_{i-1}\right) + 1}{2} \right] \cdot P_i + (1 - P_i) \cdot \mathbf{I} \tag{2}$$

where $t'_{i-1} = \operatorname{arctanh}\left(\mathbf{I}'_{i-1} \cdot 2. - 1\right)$, i and P_i denote the ith attack and the binary vector that measures the tampered pixel positions. $\sum P_i$ is fixed in each iterative attack and $\sum P_i \geq \sum P_{i+1}$, where $\sum P_i$ describes how many pixels can be modified in the ith attack. The termination goal of the iteration for $\mathcal{L}_0$ attack is set to

$$\min_{\sigma} \mathcal{L}_0 < 0.001 \text{ or } \mathcal{S}\left(\mathbf{I}'_i\right) = \ell$$

3.2 BPs Method

BPs is a kind of search-based attack, which introduces batch processing, initial perturbation, and multi-strides.The search-based attack means to iteratively decrease the value of radius D until it cannot find adversarial examples.

Given benign samples $\mathbf{I}$, the initial perturbations, such as the normal distribution perturbation $n_k = \text{Noise}(\mathbf{I}, \mu, \delta) \quad n_k \in [0, 1]$, a batch of image points are expanded from one image point $\mathbf{I}$, $\mathbf{R_0} = \{\mathbf{I}, \ldots, \mathbf{I}\} + \{0, n_{0,1}, \ldots, n_{0,b}\} \quad \forall_{j=0}^{b} \mathbf{R}_{0,j} \in [0, 1]$. μ and δ are the mean and standard deviation of the noise.

Each sample in $\mathbf{R_0}$ denotes one potential optimization direction. To find the image points that contribute to affecting the model decision, we modify the termination goal of each iteration to

$$\exists_{j\in[0,b]}\mathcal{S}\left(\mathbf{R}'_{i,j}\right)=\ell$$

Since a batch of image points do not share the same optimization step, image points that initially reach the optimization criterion will be destroyed by the optimization process of the remaining image points. Therefore, we set a static process to decrease the P_i.

$$sp=[st_0,st_1,\cdots,st_m]\quad st_i\geq st_j\,when\quad i\geq j$$
$$\sum P_i\geq\sum P_{i-1}-sp[i]$$

Note that $|sp|\geq H\cdot W\cdot c$, $H\times W\times c$ denotes the image shape. The BPs obtains the potential optimal adversarial examples of the sparsity adversarial attack when it meets

$$\exists_{\mathrm{j}\in(0,\mathrm{b})}\mathcal{S}\left(\mathbf{R}'_{t,j}\right)=\ell,\forall_{\mathrm{j}\in(0,\mathrm{b})}\mathcal{S}\left(\mathbf{R}'_{>t,j}\right)\neq\ell$$

Here $\sum P_t$ is the maximum radius of D. We set multiple strides to validate the reliability of D.

$$\sum P_i\geq\sum P_{i-1}-sp_k[i]\quad k\in\Omega$$
$$\forall_{i\in[0,m]}sp_k[i]<sp_t[i]\quad when\quad k<t$$

This setting considers the case that, the optimization process cannot find the adversarial example with P_{i-1}, but successfully generates adversarial examples with P_i, where $\sum P_{i-1}>\sum P_i$. Finally, we express the termination goal as

$$\exists_{k\in\Omega}\exists_{j\in(0,b)}\mathcal{S}\left(\mathbf{R}'_{i,j}\right)=\ell$$

Although we can better generate sparsity adversarial examples by setting multiple strides, the consumption of time is impractical. Therefore, we set $sp=stride$ (st_0 = st_1 = $\cdots$ = st_m), $\max(k)$ = 3, and $stride_k\in$ (2,3,5) for both the targeted attack and untargeted attack.

3.3 Binary Fitting Method

The search-based attack spends time to validate the potential possibility of the sparsity adversarial attack on each P_i. Next, we use the optimization strategy to determine the location and size of perturbation. Same as the search-based attack, we separate perturbations into two vectors.

Before the detailed description, we first understand the conditions of the $\mathcal{L}_0$ attack. Given the adversarial example, the tampered positions are located by the difference between the benign sample and the adversarial example.

$$|\mathbf{I}'-\mathbf{I}|>1.\,or\,\mathbf{I}'\neq\mathbf{I}\quad\mathbf{I}\in[0.,255.] \tag{3}$$

Based on Eq. (2), $\|\ \sigma\ \|_0$ is expressed as

$$\|\ \mathbf{I}'_i-\mathbf{I}\ \|_0=\|\ \left\{\left[\frac{\tanh\left(\mathbf{w}+t'_{i-1}\right)+1}{2}\right]-\mathbf{I}\right\}\cdot P_i\ \|_0$$

Properties. For convenience, we denotes $\mathbf{C} = \left\{ \left[\frac{\tanh(\mathbf{w}+t'_{i-1})}{2} \right] - \mathbf{I} \right\}$ and $\mathbf{P} = P_i$. $\mathbf{C}$ and $\mathbf{P}$ are the size and location of perturbations, respectively. Different from other norm functions, 0-norm function only calculates the number of 0 (For image pixels, calculate the number of $|\mathbf{I}'_i - \mathbf{I}| < 1$.). Therefore, $\| \sigma \|_0$ satisfies the following properties. The **upper border** of the $\| \sigma(w) \|_0$

$$\begin{aligned} \| \sigma(w) \|_0 &= \| \mathbf{C} \cdot \mathbf{P} \|_0 \\ &= \| \mathbf{C} \|_0 + \| \mathbf{P} \|_0 - \| \mathbf{C} + \mathbf{P} \|_0 \\ &\leq \| \mathbf{C} \|_0 + \| \mathbf{P} \|_0 \end{aligned}$$

Meanwhile, the **lower border** of $\| \sigma(w) \|_0$ is expressed as

$$\| \mathbf{C} \cdot \mathbf{P} \|_0 \geq \max\{ \| \mathbf{C} \|_0, \| \mathbf{P} \|_0 \}$$

We use Pro_1 and Pro_2 to denote the upper border and the lower border of the $\| \sigma(w) \|_0$. Pro_1 and Pro_2 are unconditioned conclusions. In addition, when both $\| \mathbf{C} \|_0$ and $\| \mathbf{P} \|_0$ are greater than 0, we have

$$Pro_3 \ \| \mathbf{C} \|_0 \cdot \| \mathbf{P} \|_0 \geq \| \mathbf{C} \cdot \mathbf{P} \|_0$$

Any value in $\mathbf{C}$ and $\mathbf{P}$ to 0 can decrease the value of the 0-norm function.

Fitting. The variable $\mathbf{P}$ should be an optional binary vector since we denote it as positions of $\mathcal{L}_0$ attack. However, it is well-known that the 0-norm function is an NP-hard problem, and $\mathcal{L}_0$ loss is discontinuous. Therefore, the optimal point cannot obtain by the forward-backward process. Many works are proposed to solve $\mathcal{L}_0$ task in the search mode, which are not determined by the internal information of the model, such as the gradient and logit. However, larger perturbations are set in advance, e.g. directly changing the value of any selected pixel (normalized) to 0 or 1.

For such cases, we introduce the following function to replace the 0-norm function

$$\begin{aligned} &\max_{\mathbf{w}} \| \sigma(\mathbf{w}) \|_0 = \max_{\mathbf{w}} \| \mathbf{C} \cdot \mathbf{P} \|_0 \\ &\Rightarrow \min_{\mathbf{w}} \sum \{ 1 - \frac{1}{\exp(\beta \cdot \mathbf{C}_{\mathbf{i,j,k}} \cdot \mathbf{P}_{\mathbf{i,j,k}})} \} \quad \mathbf{P} \in R_{H \times W \times c} \end{aligned}$$

We denote $C(\mathbf{T}_{\mathbf{i,j,k}})$ as $\frac{1}{\exp(\beta \cdot \mathbf{T}_{\mathbf{i,j,k}})} - 1$ and use $C(\cdot)$ to replace $\| \cdot \|_0$. β is a hyperparameter to better fit the 0-norm function. $C(\mathbf{P}_{\mathbf{i,j,k}}) = 0$ only when $\mathbf{P}_{\mathbf{i,j,k}} \to 0$.

Next, we prove that $\| \mathbf{T} \|_0$ can be replaced by $C(\mathbf{T})$. For Pro_2,

$$\begin{aligned} &C(\mathbf{C} \cdot \mathbf{P}) \geq \max\{C(\mathbf{C}), C(\mathbf{P})\} \\ &\Leftrightarrow \max\{ \frac{1}{\exp(\beta \cdot \mathbf{C}^2)}, \frac{1}{\exp(\beta \cdot \mathbf{P}^2)} \} \leq \frac{1}{\exp(\beta \cdot \mathbf{C}^2 \cdot \mathbf{P}^2)} \\ &\Leftrightarrow \mathbf{P}^2 \leq 1, st. \ \| \mathbf{P} \|_0 \leq \| \mathbf{C} \|_0 \ or \\ &\quad \mathbf{C}^2 \leq 1, st. \ \| \mathbf{C} \|_0 \leq \| \mathbf{P} \|_0 \end{aligned}$$

$\max\{\| \mathbf{C} \|_0, \| \mathbf{P} \|_0\} = \| \mathbf{P} \|_0$, since $\mathbf{P}$ denotes the modified positions. For $\mathbf{C} = \frac{\tanh(\mathbf{w}+t')+1}{2} - \mathbf{I}$, its value is limited to [-1,1]. Therefore, Pro_2 is an unconditioned expression. For Pro_3,

$$\sum C(\mathbf{C} \cdot \mathbf{P}) \leq \sum C(\mathbf{C}) \cdot \sum C(\mathbf{P})$$

Pro_3 always holds since $C(\mathbf{C} \cdot \mathbf{P}) \leq 0$ and $C(\mathbf{C}) \cdot C(\mathbf{P}) \geq 0$. Combine the Pro_2 and Pro_3, we have $\max\{C(\mathbf{C}), C(\mathbf{P})\} \leq C(\mathbf{C} \cdot \mathbf{P}) \leq 0$. Thus, the function $C(\mathbf{P} \cdot \mathbf{C}) \leq C(\mathbf{C}) + C(\mathbf{P})$ is possibly holds, $C(\mathbf{P} \cdot \mathbf{C}) = C(\mathbf{C}) + C(\mathbf{P})$ only when the value of β (the default value is 10) is large enough. For Pro_1,

$$\begin{aligned} & C(\mathbf{C} \cdot \mathbf{P}) \leq C(\mathbf{C}) + C(\mathbf{P}) \\ \Leftrightarrow 0 \leq & \left(\frac{1}{\exp\left(\beta \cdot \mathbf{C}^2\right)} - \frac{1}{\exp\left(\beta \cdot \mathbf{P}^2 \cdot \mathbf{C}^2\right)}\right) + \\ & \left(\frac{1}{\exp\left(\beta \cdot \mathbf{P}^2\right)} - 1\right) \end{aligned}$$

Available Loss Function. To decrease the number of tampered image pixels, we construct the constraint as,

$$\min_{\mathbf{w},\mathbf{P}} \mathcal{L}_{\mathbf{C} \cdot \mathbf{P}} = -\sum_{i=1}^{b} C(\mathbf{C}_i \cdot \mathbf{P}_i)$$

Next, several $\mathcal{L}_0$ attacks are realized by separating the perturbation to the pixel value map and pixel position map.

$$\begin{aligned} & \min_{\mathbf{w}} \mathcal{L}_{\mathbf{C}} = -\sum_{i=1}^{b} C(\mathbf{C}_i), \quad \min_{\mathbf{P}} \mathcal{L}_{\mathbf{P}} = -\sum_{i=1}^{b} C(\mathbf{P}_i) \\ & \min_{\mathbf{w},\mathbf{P}} \mathcal{L}_{\mathbf{C}+\mathbf{P}} = \mathcal{L}_{\mathbf{P}} + \gamma \cdot \mathcal{L}_{\mathbf{C}} \end{aligned}$$

Normally, the value of $||\mathbf{P}||_0$ is much greater than the value of $||\mathbf{C}||_0$. Therefore, we use γ ($\gamma \leq 1$) to show the different contributions of the $\mathbf{C}$ and $\mathbf{P}$. $\mathcal{L}_{\mathbf{C}}$, $\mathcal{L}_{\mathbf{P}}$, and $\mathcal{L}_{\mathbf{C}+\mathbf{P}}$ are both lower constraints of $\mathcal{L}_{\mathbf{C} \cdot \mathbf{P}}$, that is, if $case$ is a optimal solution of $\mathcal{L}_{\mathbf{C}}$ (or $\mathcal{L}_{\mathbf{P}}$, $\mathcal{L}_{\mathbf{C}+\mathbf{P}}$), then $case$ is also a optimal solution of $\mathcal{L}_{\mathbf{C} \cdot \mathbf{P}}$. On the contrary, $\mathcal{L}_{||\mathbf{C}||_0 \cdot ||\mathbf{P}||_0}$ is the upper border of the $\mathcal{L}_{\mathbf{C} \cdot \mathbf{P}}$, which cannot guarantee the optimal value of $\mathcal{L}_{\mathbf{C} \cdot \mathbf{P}}$.

$$\min_{\mathbf{w},\mathbf{P}} \mathcal{L}_{||\mathbf{C}||_0 \cdot ||\mathbf{P}||_0} = \mathcal{L}_{\mathbf{C}} \cdot \mathcal{L}_{\mathbf{P}}$$

The overall loss function, $\mathcal{L}_0$ in Eq. (1), is constructed by replacing the $\| \sigma \|_0$ with the above-mentioned constraints such as $\mathcal{L}_{\mathbf{C}+\mathbf{P}}$.

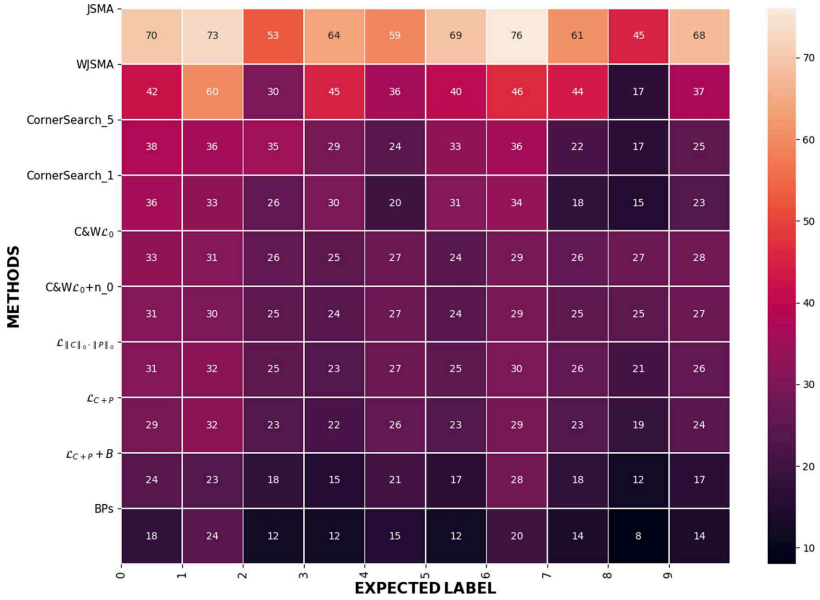

Fig. 1. The value denotes how many pixels have been modified. The X-axis denotes the targeted attack goals (10 classes, 0–9). Y-axis is the sparse attack method used in this paper. The T in CornerSearch$_T$ means the search stride. The maximum perturbable pixels in CornerSearch are set to 100. Similar to PGD (projected gradient descent) attack, we introduce the initial perturbation (the normal distribution perturbation) to the benign sample which is expressed as $+n_0$. In the same way, +B denotes the model with batch processing.

4 Experiments

4.1 Settings

In our experiments, all data comes from the public datasets CIFAR-10 [14] and MINIST [17]. We implement the prepared model by using Tensorflow®. Without the special statement, we set the batch size to 128 when training the model (the default model used in C&W attack) and to 32 in the attacking process. The initial value of the learning rate is set to 0.01 in the training and attacking process. The code is available at https://github.com/Dlut-lab-zmn/Least_pixel_attack.

4.2 Evaluation of Targeted Attack Ability

We compare the proposed sparse adversarial attacks with state-of-the-art methods: Carlini-Wagner $\mathcal{L}_0$-attack [3], JSMA [25], WJSMA [19] and CornerSearch [6]. In this section, for the targeted attack, we randomly select 500 images (50 images in each class) from the testing set for targeted attacks. Each sample has 10 attack targets (0–9). Therefore, for each attack, we will generate 4500 adversarial examples. To visually compare various attack methods, we provide the confusion matrix that records the experiment

Table 1. Comparison of various of $\mathcal{L}_0$ targeted attack. Each sample has 10 attack targets. For the C&W attack, we set the search iteration to 1000. n_0 denotes the normal distribution perturbation. For CornerSearch, the upper limit of the allowed number of pixel changes is 100. Please note that CornerSearch can not always succeed in the attack. In BFM, we assign the optimization iterations to 2000, γ to 0.4 for MNIST, and 1. for CIFAR. The search iteration for BFM and $\mathcal{L}_{\mathbf{C}+\mathbf{P}}$+B is 512. ****** and ** are the 1st and 2nd best results. The codes for the comparison methods are given in the reference.

		JSMA		CornerSearch	C&W		BPs	Binary Fitting Method (BFM)			
	Metrics	JSMA	WJSMA	Stride = 5	$\mathcal{L}_0$	$\mathcal{L}_0$+n_0		$\mathcal{L}_{\mathbf{C}\cdot\mathbf{P}}$	$\mathcal{L}_{\|\|\mathbf{C}\|\|_0\cdot\|\|\mathbf{P}\|\|_0}$	$\mathcal{L}_{\mathbf{C}+\mathbf{P}}$	$\mathcal{L}_{\mathbf{C}+\mathbf{P}}$+B
Dataset	Type	Bottom-up		Top-down	Bottom-up		Top-down	Optimize-based mode			
MNIST	AR	46.85	96.64	49.82	97.93	98.62	**100.0**	96.69	92.76	98.1	**100.0**
	Pixel (mean)	63.84	39.6	29.52	27.82	26.7	**14.88**	110.6	26.68	24.77	19.38
	Per (mean)	49.1	33.32	17.24	16.95	18.72	**11.88**	12.47	14.52	14.56	12.24
	Time (s)	0.32	**0.25**	4.19	5.14	5.58	10.5	4.95	4.84	4.5	5.74
CIFAR	AR	99.62	100	79.15	99.11	99.71	100	100	77.78	98.88	100
	Pixel (mean)	68.25	42.53	43.27	27.58	32.29	**5.34**	642.8	127.4	25.51	14.24
	Per (mean)	63.17	38.19	27.91	9.78	14.72	**4.16**	11.38	3.07	4.76	4.44
	Time(s)	5.32	2.92	7.057	4.67	7.83	24.8	4.67	4.51	4.37	6.63

results on the MNIST dataset in Fig. 1. The value in this confusion matrix denotes the average modified pixels that need to generate adversarial examples. We observe from Fig. 1 that the model performs poor recognition ability for digit 8, but is robust for digits 1 and 6.

In Table 1, we report the attack rate of each method, which is the fraction of correctly classified samples that can be successfully attacked. *Pixels(Mean)* denotes the number of pixels that need to modify to misclassify the network for each attack. To better illustrate the perturbation, we adopt the Eq. (4) to calculate *Pixels(mean)* instead of Eq. (5), which also measures the difference for different channel, such as when evaluation on the CIFAR dataset for the CornerSearch method, 43.27 for Eq. (5) and 14.44 for Eq. (4).

$$Pixels(Mean) = \sum\{|\mathbf{I}' - \mathbf{I}| \geq 1.\} \quad \mathbf{I} \in [0., 255.] \tag{4}$$

$$Pixels(Mean) = \sum\{\max(|\mathbf{I}' - \mathbf{I}|, c) \geq 1.\}\, \mathbf{I} \in [0., 255.] \tag{5}$$

The average value of the size of perturbations for all adversarial examples is denoted as *Per(Mean)*. *Time* denotes the average time consumption for generating each targeted adversarial example. The sparsity adversarial attacks are categorized into three types according to different search modes, top-down, bottom-up, and optimize-based mode. Top-down attacks such as C&W gradually reduce the number of pixels ($\sum P_i \leq \sum P_{i-1}$) that can be modified. Conversely, bottom-up attacks such as JSMA tend to modify more pixels to affect the classification decision step by step. The pixel positions of the optimize-based method will be changed following the optimization process.

We observe from Table 1 that initial perturbations are not efficient in improving the targeted attack rate, e.g., the results of $\mathcal{L}_0$ and $\mathcal{L}_0 + n_0$. Additionally, BPs achieves the state-of-the-art attack performance, namely, BPs only modifies half of the pixels than the best baseline while maintaining a similar attack rate. The significant attack

performance of BPs benefits from batch processing and multiple strides. The former provides multiple initialization directions, and the latter increases the probability of obtaining the optimal solution.

$\mathcal{L}_{\mathbf{C}\cdot\mathbf{P}}$ performs worse attack performance than other sparsity constraints. In our opinion, that is because $C(\mathbf{T})$ is only the fitting version of $\| \mathbf{T} \|_0$, many pixels are constrained towards 0 but are not really equal to 0. Therefore, many pixels are modified for generating adversarial examples, but the size of perturbation is still small. For instance, not all $C(\mathbf{T})$ meets Pro_1 in $\| \mathbf{T} \|_0$. The $\mathcal{L}_{\mathbf{C}+\mathbf{P}}$ is helpful because both $\mathbf{C}$ and $\mathbf{P}$ are limited to tiny values for invalid positions. Similar to BPs, the batch processing for $\mathcal{L}_{\mathbf{C}+\mathbf{P}}$ that offers multiple optimization directions has successfully improved the attack rate.

For both targeted and untargeted attacks, the bottom-up attacks tend to spend less time than the top-down attacks. The more vulnerable the sample, the greater the time difference. However, we cannot observe this situation in a targeted attack since the low attack rate of bottom-up methods. For the bottom-up attack, the running time is controlled by the maximum search steps.

Table 2. Comparison of various of $\mathcal{L}_0$ untargeted attack. For C&W attack, we set the search iteration to 1000. We assign the optimization iteration of the attack with batch processing to 512.

Dataset	Metrics	CornerSearch	C&W		BPs	BFM
		Stride = 5	$\mathcal{L}_0$	$\mathcal{L}_0$+n$_0$		$\mathcal{L}_{\mathbf{C}+\mathbf{P}}$+B
MNIST	AR	100.0	100.0	100.0	100.0	100.0
	Pixel (mean)	7.54	19.75	15.21	**5.80**	10.51
	Per (mean)	7.39	13.23	13.29	**4.97**	6.74
	Time (s)	**1.05**	4.32	4.64	14.73	5.61
CIFAR	AR	100.0	100.0	100.0	100.0	100.0
	Pixel (mean)	**5.44**	15.76	19.31	8.20	9.41
	Per (mean)	3.86	6.14	8.92	2.25	**1.74**
	Time (s)	**0.69**	4.45	4.96	28.87	7.21

4.3 Evaluation of Untargeted Attack Ability

In this subsection, we compare the performance of the untargeted attack between our methods (BPs attack and BFM attack) and existing attacks (CW) attack and CornerSearch method). We randomly select 1000 images from the testing set as the dataset. All of them are correctly classified. Thus we need to generate 1000 adversarial examples for each attack. The experiment results are given in Table 2. As we can see, the initial perturbation improve untargeted attack performance in the MNIST dataset. However, this conclusion is not valid in CIFAR since we use Eq. (4) and Eq. (5) to count the tampered points. Apart from that, our method generates adversarial examples with fewer pixels changed, but this improvement comes at the cost of time. For instance, in the

CIFAR dataset, the corner search method only needs 0.69 s to generate one adversarial example, but the BFM need 7.21 s. Therefore, BPs and BFM will be more useful for targeted attack tasks, the high attack rate, and the necessary time consumption.

5 Conclusion

In this paper, we propose two white-box sparsity adversarial attacks, BPs and BFM. Both attacks tend to search for the most influential pixels that affect the model decision. Extensive experiments show that our methods outperform or are competitive with previous works. Since the attack efficiency of these two white-box sparsity adversarial attacks is at the cost of time, we recommend that BPs and BFM be used for a targeted attack task.

Acknowledgement. This work is supported by the National Natural Science Foundation of China (No. U1936117, No. 62106037, No. 62076052), the Science and Technology Innovation Foundation of Dalian (No. 2021JJ12GX018), the Fundamental Research Funds for the Central Universities (DUT21GF303, DUT20TD110, DUT20RC(3)088), and the Open Project Program of the National Laboratory of Pattern Recognition (NLPR) (No. 202100032).

References

1. Ali, K., Quershi, A.N., Arifin, A.A.B., Bhatti, M.S., Sohail, A., Hassan, R.: Deep image restoration model: a defense method against adversarial attacks. CMC-Comput. Mater. Continua **71**(2), 2209–2224 (2022)
2. 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, pp. 3–14 (2017). https://github.com/carlini/nn_robust_attacks
3. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57. IEEE (2017)
4. Chen, H., Zhang, H., Chen, P.Y., Yi, J., Hsieh, C.J.: Attacking visual language grounding with adversarial examples: a case study on neural image captioning. arXiv preprint arXiv:1712.02051 (2017)
5. Croce, F., Hein, M.: A randomized gradient-free attack on ReLU networks. In: Brox, T., Bruhn, A., Fritz, M. (eds.) GCPR 2018. LNCS, vol. 11269, pp. 215–227. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12939-2_16
6. Croce, F., Hein, M.: Sparse and imperceivable adversarial attacks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4724–4732 (2019). https://github.com/fra31/sparse-imperceivable-attacks
7. Deng, B., Ran, Z., Chen, J., Zheng, D., Yang, Q., Tian, L.: Adversarial examples generation algorithm through DCGAN. Intell. Autom. Soft Comput. **30**(3), 889–898 (2021)
8. Ding, X., Chen, Y., Tang, Z., Huang, Y.: Camera identification based on domain knowledge-driven deep multi-task learning. IEEE Access **7**, 25878–25890 (2019)
9. Dong, Y., et al.: Boosting adversarial attacks with momentum. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9185–9193 (2018)
10. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
11. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask R-CNN. IEEE Trans. Pattern Anal. Mach. Intell. **42**(2), 386–397 (2020)

12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
13. Huang, R., Xu, B., Schuurmans, D., Szepesvári, C.: Learning with a strong adversary. arXiv preprint arXiv:1511.03034 (2015)
14. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
16. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
17. LeCun, Y., Cortes, C., Burges, C.: MNIST handwritten digit database (2010)
18. Lin, T., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. IEEE Trans. Pattern Anal. Mach. Intell. **42**(2), 318–327 (2020)
19. Loison, A., Combey, T., Hajri, H.: Probabilistic Jacobian-based saliency maps attacks. arXiv: abs/2007.06032 (2020). https://github.com/probabilistic-jsmas/probabilistic-jsmas
20. Lu, J., Issaranon, T., Forsyth, D.: SafetyNet: detecting and rejecting adversarial examples robustly. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 446–454 (2017)
21. Lu, J., Sibai, H., Fabry, E., Forsyth, D.: Standard detectors aren't (currently) fooled by physical adversarial stop signs. arXiv preprint arXiv:1710.03337 (2017)
22. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks, June 2017
23. Meng, D., Chen, H.: MagNet: a two-pronged defense against adversarial examples. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 135–147 (2017)
24. Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B.: On detecting adversarial perturbations. In: ICLR 2017: International Conference on Learning Representations 2017 (2017)
25. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 372–387. IEEE (2016), https://github.com/RobertoFalconi/BlackBoxAttackDNN
26. Ruan, W., Wu, M., Sun, Y., Huang, X., Kroening, D., Kwiatkowska, M.: Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance. In: International Joint Conference on Artificial Intelligence (2019)
27. Ruiz, N., Bargal, S.A., Sclaroff, S.: Disrupting deepfakes: adversarial attacks against conditional image translation networks and facial manipulation systems. arXiv: abs/2003.01279 (2020)
28. Sarkar, S., Bansal, A., Mahbub, U., Chellappa, R.: UPSET and ANGRI: breaking high performance image classifiers. arXiv preprint arXiv:1707.01159 (2017)
29. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
30. Szegedy, C., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
31. Taori, R., Kamsetty, A., Chu, B., Vemuri, N.: Targeted adversarial examples for black box audio systems. In: 2019 IEEE Security and Privacy Workshops (SPW), pp. 15–20. IEEE (2019)
32. Uprety, S.P., Jeong, S.R.: Adversarial training for multi domain dialog system. Intell. Autom. Soft Comput. **31**(1), 1–11 (2022)

33. Wang, Y., Zhang, C., Liao, X., Wang, X., Gu, Z.: An adversarial attack system for face recognition. J. Artif. Intell. **3**(1), 1 (2021)
34. Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., Yuille, A.: Adversarial examples for semantic segmentation and object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1369–1378 (2017)
35. Xu, H., Du, C., Guo, Y., Cui, Z., Bai, H.: A generation method of letter-level adversarial samples. J. Artif. Intell. **3**(2), 45 (2021)
36. Yuan, X., He, P., Zhu, Q., Bhat, R.R., Li, X.: Adversarial examples: Attacks and defenses for deep learning. arXiv preprint arXiv:1712.07107 (2017)
37. Zhao, M., Wang, B., Wei, F., Zhu, M., Sui, X.: Source camera identification based on coupling coding and adaptive filter. IEEE Access **8**, 54431–54440 (2020)