Machine Vision and Applications (2021) 32:67
https://doi.org/10.1007/s00138-021-01194-6

ORIGINAL PAPER

A data independent approach to generate adversarial patches

Xingyu Zhou[1] · Zhisong Pan[2] · Yexin Duan[3] · Jin Zhang[3] · Shuaihui Wang[2]

Received: 22 July 2020 / Revised: 27 February 2021 / Accepted: 5 March 2021 / Published online: 5 April 2021

Abstract
Deep neural networks are vulnerable to adversarial examples, i.e., carefully perturbed inputs designed to mislead the network at inference time. Recently, adversarial patch, with perturbations confined to a small and localized patch, emerged for its easy accessibility in real-world attack. However, existing attack strategies require training data on which the deep neural networks were trained, which makes them unsuitable for practical attacks since it is unreasonable for an attacker to obtain the training data. In this paper, we propose a data independent approach to generate adversarial patches (DiAP). The goal is to craft adversarial patches that can fool the target model on most of the images without any knowledge about the training data distribution. In the absence of data, we carry out non-targeted attacks by fooling the features learned at multiple layers of the deep neural network, and then employ the potential information of non-targeted adversarial patches to craft targeted adversarial patches. Extensive experiments demonstrate impressive attack success rates for DiAP. Particularly in the blackbox setting, DiAP outperforms state-of-the-art adversarial patch attack methods. The patches generated by DiAP also function well in real physical scenarios, and could be created offline and then broadly shared.

Keywords Adversarial patch · Data independent · Physical attack

1 Introduction

Deep neural networks (DNNs) perform dramatically well in various visual tasks, including image classification, object detection, and semantic segmentation. However, they are also susceptible to being fooled by adversarial perturbations: perturbations combined with data inputs in a specific way, cause intentional misclassification.

Attacking deep neural networks has drawn an increasing attention, and researchers have made great progress in understanding the space of adversarial perturbations, beginning in the digital domain (e.g. by modifying images corresponding to a scene) [1–4], and more recently in the physical domain [5–8]. Compared to attacks in the digital domain, adversarial perturbations encounter more challenges when attacking in the physical world: (1) Perturbations in the digital world can be so small in magnitude that it is impossible for a camera to perceive them due to the sensor imperfections. (2) The perturbations generated by many current algorithms are image-dependent. If the image in front of the surveillance camera changes, the attacker needs to generate a corresponding new perturbation immediately, which is difficult to achieve in physical attacks. (3) The perturbed image is difficult to maintain at a precise distance and angle in the view of surveillance camera, which requires perturbations to be robust to various transformations (e.g. rotations or scaling) and locations.

To tackle these challenges and achieve attacks in physical scenarios, Brown et al. [9] proposed visible adversarial perturbations, called Adversarial Patch (GoogleAP). These adversarial patches can be printed, added to any scene, photographed, and presented to deep neural networks. Even when the patches are small, the deep neural network is incapable to identify real objects in the scene and report a false class. Adversarial patch is image-independent and robust for rotation and scaling, can be placed anywhere within the field of view of the deep neural network, and causes the deep neural network to output a targeted class (Fig. 1).

Adversarial patch can be applied to traffic signs to mislead automated vehicles [10], or placed near products to fool

✉ Xingyu Zhou
universezhou@sina.cn

1 Communication Engineering College, Army Engineering University of PLA, Nanjing 210007, China

2 Control Engineering College, Army Engineering University of PLA, Nanjing 210007, China

3 Zhenjiang Campus, Army Military Transportation University, Zhenjiang 212000, China

Fig. 1 Examples of real-world adversarial patch attacks against VGG 16. We can observe that each object is misclassified when the adversarial sticker is placed beside it

online shopping platforms [11], or even affixed to clothing to hide attackers from surveillance cameras [12,13]. Adversarial patch brings a serious challenge to the safety of deep neural networks. At the same time, research on adversarial patch helps to improve the defense capability of deep neural networks against malicious attacks.

However, GoogleAP requires training data on which the target model is trained. The data dependency brings barriers to the application of GoogleAP in the real world as training data of the target system is generally unavailable. For example, autonomous driving companies never tell the public what data they use to train detectors, online shopping website protects the data used to train their classifier from being stolen, and face verification devices carefully store their face data in their systems.

In order to address these shortcomings, we present a novel data independent approach to craft adversarial patches (DiAP). The objective of DiAP is to generate an adversarial patch that can fool the target model on most of the images without any knowledge about the data distribution. Inspired by GD-UAP [14], DiAP perform non-targeted attacks by fooling the features learned by the deep neural network. In other words, we formulate this as an optimization problem to calculate the non-targeted adversarial patch, which can fool the features learned on each layer of the deep neural network and eventually making it to be misclassified as an adversarial example. After generating the non-targeted adversarial patch, DiAP takes it as an important background item to help an attacker extract the features of the target class for crafting the targeted adversarial patch. Experimental results show that the adversarial patch generated by DiAP exhibits strong attack capabilities. In particular, by extracting vague information about training data from non-targeted patches, DiAP outperforms the state-of-the-art attack methods in blackbox attack scenarios.

2 Related work

Most of prior work has focused on generating visually imperceptible adversarial perturbations which cover the entire input image [15,16]. This type of attack is effective in digital scenes, but is difficult to deploy in actual physical scenes. The camera equipment cannot carefully capture the small perturbation on the input, resulting in the failure of the adversarial attack in real world.

Some researchers have tried to add visible perturbations at specific locations in the input image to attack deep neural networks [7,17,18]. Compared to visually imperceptible adversarial perturbations, the visible perturbations at specific locations require fewer pixels to change, and allow for significant changes to specified pixel points. This kind of attack performs quite well in digital domain and shows some attack capabilities in the physical scene. However, since they can only be placed in specific locations in the input image, the application of these perturbations in physical attacks is confined.

Recent work by [9] and [19] have studied adversarial perturbations under a new attack model - adversarial patch, which is restricted to a small region and can be placed anywhere on the input image. This attack is based on the assumption that machine learning models operate without human validation of each input, so malicious attackers may not care about the imperceptibility of their attacks. Moreover, even if humans are able to notice adversarial patch, they may view it as an art form rather than some way of attacking the deep neural network.

It is observed that the objective presented by [9] and [19] to craft adversarial patch requires real and clean examples to participate in training. However, in physical attacks, it is difficult for malicious attackers to obtain training data for the target system in general. Therefore, this paper presents a data independent attack method, DiAP.

From the perspective of whether there is an attack target, adversarial attack against the deep neural network can be divided into non-targeted and targeted. Non-targeted attack don't specify the prediction class of the deep neural network, and the adversarial examples may be identified as any category other than the correct one. However, the purpose of targeted attack is to trick the deep neural network into recognizing adversarial examples as specified categories. Our DiAP can carry out both non-targeted attack and targeted attack.

3 DiAP for non-targeted attack

In the first setup, we explore generating adversarial patches that can be used for *non-targeted attack* without training data.

3.1 Setting and method

Let μ denote a distribution of images in $\Re^n$, and f denote a pre-trained deep neural network that outputs for each image $x \in \Re^n$ an estimated label $f(x)$. We want the adversarial patch p to attack the target model by replacing a part of the image x, regardless of the scale, rotation or location of the patch. The adversarial example $A(p, x, l, t)$ could be achieved by a patch application operator which first applies transformations t (e.g. rotations or scaling) to the patch p, and then applies the transformed patch p to the image x at location l. That is, the goal of DiAP to perform a non-targeted attack is to seek an non-targeted adversarial patch p_{nt} such that

$$f(A(p_{nt}, x, l, t)) \neq f(x), \quad \text{for } x \sim \mu \tag{1}$$

In the absence of training data, we fool the features learned at individual layers of the deep neural network to finally craft the non-targeted adversarial patch p_{nt} . To achieve this goal, we introduce a variant of the spurious activation loss proposed in [20]. In particular, the non-targeted patch p_{nt} is trained to optimize the spurious activation objective function

$$\arg\max_{p_{nt}} \mathbb{E}_{l\sim L, t\sim T}\left[\log\left(\prod_{i=1}^{K}\|\mathcal{L}_i(A(p_{nt}, I_z, l, t))\|_2\right)\right] \tag{2}$$

where L is a distribution over locations in the image, and T is a distribution over transformations of the patch. I_z is the background image with $RGB = [0, 0, 0]$ for all pixels. $\mathcal{L}_i(A(p_{nt}, I_z, l, t))$ is the output tensor at layer i when the image $A(p_{nt}, I_z, l, t)$ is fed to the network f. K is the number of layers in f at which we maximize the output caused by $A(p_{nt}, I_z, l, t)$. The proposed objective computes product of output magnitude at all the individual layers. Note that the RGB values of the background image I_z are all zero, which will not mislead the features extracted by the various layers of the deep neural network. Therefore, the erroneous features extracted by the network are actually caused by the non-targeted adversarial patch p_{nt}.

We would ideally want the patch p_{nt} to provoke as much strong disturbance at all layers as possible in order to fool the features distilled by multiple layers and attack the network, that is, the larger $\mathbb{E}[\cdot]$ in Eq. (2), the greater the contamination caused by p_{nt}. During the training, the patches are transformed and then digitally inserted on a random location on the background image I_z, and we optimize Eq. (2) without any training data. The entire process is detailed in Algorithm 1.

Algorithm 1: Computation of non-targeted adversarial patch p_{nt}

Data: Target deep neural network f, background image I_z, distribution over location L, distribution of transformations T, number of iterations Ite_{max}, learning rate η.
Result: Non-targeted adversarial patch p_{nt}.

1 Randomly initialize p_{nt}
2 $Ite = 0$
3 **while** $Ite \leq Ite_{max}$ **do**
4 Randomly select parameter $l \sim L$ and $t \sim T$
5 $Loss_{nt} = -\log\left(\prod_{i=1}^{K}\|\mathcal{L}_i(A(p_{nt}, I_z, l, t))\|_2\right)$
6 $p_{nt} = p_{nt} - \eta\frac{\partial Loss_{nt}}{\partial p_{nt}}$
7 $Ite = Ite + 1$
8 **end**
9 Return p_{nt}

For convenience of optimizing Eq. (2), we added a negative sign after the logarithm operation, that is, the smaller $Loss_{nt}$ defined in line 5, the stronger disturbance caused by p_{nt}

3.2 Experiments and results

We utilized 5 pre-trained ImageNet models, viz. Inception-V3 [21], ResNet-50 [22], Xception [23], VGG-16 [24] and VGG-19 [24].[1] For all experiments, the weights of these models are kept frozen throughout the optimization process. We test our attack from three aspect: (1) **The Whitebox-Single Model Attack** trains and evaluates a single patch on a single model, and the process is repeated on the models mentioned above. (2) **The Whitebox-Ensemble Attack** jointly trains a single patch across five models, and then evaluates the patch by averaging the win rate across all these models. (3) **The Blackbox Attack** is similar to the leave one method, jointly training a single patch across four of the ImageNet models, and then evaluating the blackbox attack on the fifth model. The blackbox attack is very similar to the real-world attack, because the attacker knows neither the structure, parameters, nor the training dataset of the target model. In the process of training, we use the gradient descent optimizer and set the hyper-parameter $Ite_{max} = 800$ and $\eta = 8$ for Algorithm 1.

In the testing phase, we need images to evaluate the attack performance of the patch, although our approach is data independent when generating the patch. Note that there are images in the ILSVRC 2012 validation data that are misclas-

[1] https://github.com/fchollet/deep-learning-models/releases/download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels.h5 https://github.com/keras-team/keras-applications/releases/download/resnet/resnet50_weights_tf_dim_ordering_tf_kernels.h5 https://github.com/fchollet/deep-learning-models/releases/download/v0.4/xception_weights_tf_dim_ordering_tf_kernels.h5 https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels.h5 https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg19_weights_tf_dim_ordering_tf_kernels.h5.

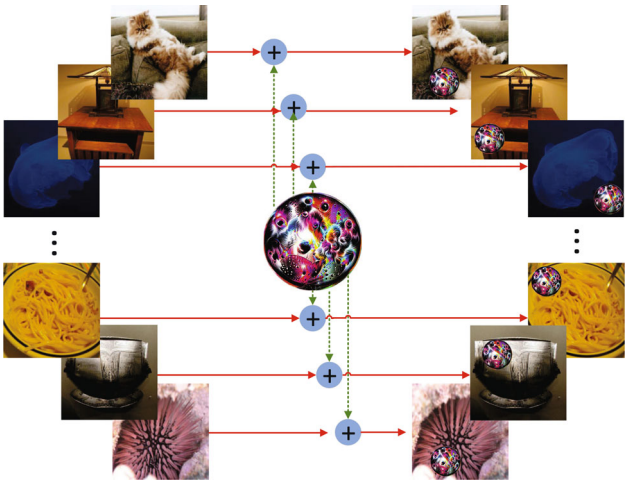


Fig. 2 The process of constructing adversarial examples when the scaling parameter is 10%. Left images: original natural images. Central image: an adversarial patch. Right images: adversarial images. The patch is rescaled to cover 10% of the original images, and then randomly rotated and placed at random locations on different test images

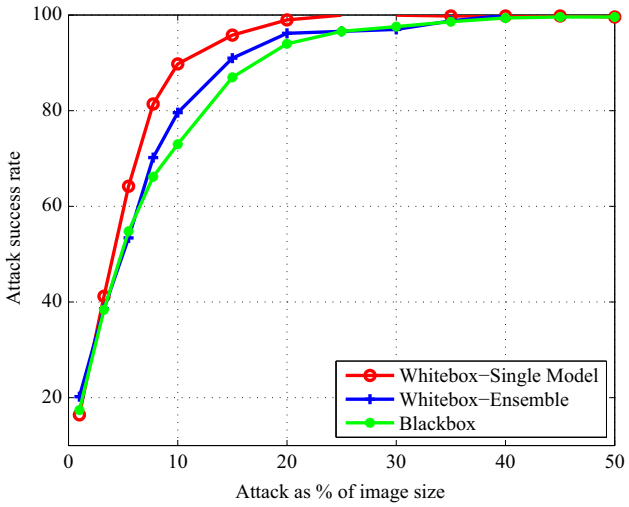


Fig. 3 Attack success rates of non-targeted adversarial patches generated by DiAP. Each point in the plot is computed by averaging the results of 1000 adversarial examples, which are crafted by applying the patch to test images at random locations in these images. This is done for various scales or rotations of the patch, with each transformation independently tested on 1000 images. Note that these success rates are the mean of five ImageNet models

sified by the above 5 pre-trained models. For example, the top-1 accuracy of Inception-V3 is 82.8%. It is less meaningful to study the attack success rates if the pre-trained models cannot correctly classify the original images. To verify the effect of our attack method, we randomly choose 1000 images from the ILSVRC 2012 validation data for testing, and these images are correctly classified by the above 5 pre-trained models.

The patches are rescaled and rotated, and then digitally inserted on a random location on the test images. We selected 13 scaling parameters, i.e. 5 points in equal intervals from 1 to 10% and 8 points in equal intervals from 10 to 50%. The rotation angle is limited in $[-45°, 45°]$. Figure 2 shows the process of constructing adversarial examples for testing when the scaling parameter is 10%. The patch generated by Inception-V3 for whitebox single model attack is rescaled to cover 10% of the input images, and then randomly rotated and placed at random positions on different test images.

Figure 3 shows the attack success rates, each point is calculated through 5000 tests (1000 adversarial images $\times$ 5 pre-trained models). When the non-targeted adversarial patch covers 10% of the image area, the attack success rates of all three attack methods achieve more than 70% attack success rates, of which the whitebox single model attack exceeds 90%, although DiAP knows nothing about the training data.

Figure 4 shows some patches generated for non-targeted attack. These patches all seem to contain a lot of small round circular patterns and exhibit some symmetry. We examine the estimated labels of adversarial examples crafted by the adversarial patch shown in Fig. 4. When the patch covers 10% of the testing images, 97.1% of the adversarial examples super-

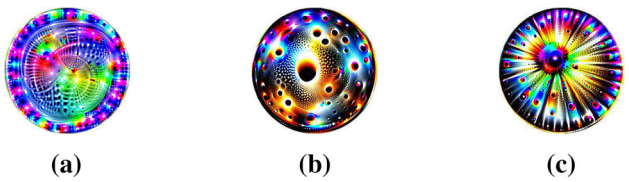


Fig. 4 Adversarial patches for non-targeted attack. **a** is generated for the whitebox single model attack, **b** is computed for the whitebox ensemble attack, and **c** is crafted for the blackbox attack. The target model is the pre-trained VGG-16

imposed with the patch (a) are classified as *bubbles*, 60.9% of the images perturbed by the patch (b) are recognized as *salt shaker* or *ladybug*, and 55.8% of the images disturbed by the patch (c) are identified as *bubble* or *pinwheel*. This indicates that when performing non-targeted attacks by optimizing Eq. (2), DiAP tends to construct patches with circular patterns in order to enhance robustness to various transformations, so the corresponding adversarial examples are more likely to be classified as circular objects.

When DiAP perform non-targeted attacks, the above experiments clearly show the existence of several dominant labels, and these labels are typically circular objects. However, when performing physical attacks, malicious attackers may want adversarial patches to cause the network to output any target class. In the next section, we generate adversarial patches to trick the network to evaluate different adversarial examples as the target class.

4 DiAP for targeted attack

We now explore crafting adversarial patches that can be applied to perform *targeted attack* without training data.

Algorithm 2: Computation of targeted adversarial patch p_t

Data: Target deep neural network f, background image I_{nt}, learning rate η, distribution over location L, distribution of transformations T, number of iterations Ite_{max}, target label $\widehat{y}$.
Result: Targeted adversarial patch p_t.

1 Randomly initialize p_t
2 $Ite = 0$
3 $Loss_s = inf$
4 **while** $Ite \leq Ite_{max}$ **do**
5 Randomly select parameter $l \sim L$ and $t \sim T$
6 $I_{nt} = A(p_{nt}, I_z, l, t)$
7 Randomly select parameter $l \sim L$ and $t \sim T$
8 $Loss_t = -\log \Pr(\widehat{y}|A(p_t, I_{nt}, l, t)$
9 **if** $Loss_t < Loss_s$ **then**
10 $p_t = p_t - \eta \frac{\partial Loss_t}{\partial p_t}$
11 $Loss_s = Loss_t$
12 **end**
13 $Ite = Ite + 1$
14 **end**
15 Return p_t

Note that we added a negative sign after the logarithm operation in Eq. (4) to facilitate the optimization. In each iteration, after randomly selecting parameters l and t to construct the background image I_{nt} (as shown in lines 5 and 6), parameters l and t are randomly selected again to construct the adversarial example $A(p_t, I_{nt}, l, t)$ (as shown in lines 7 and 8).

4.1 Setting and method

Let $\widehat{y}$ denote the target label, and the goal of DiAP to implement a targeted attack is to seek an targeted adversarial patch p_t such that

$$f(A(p_t, x, l, t)) = \widehat{y}, \quad \text{for } x \sim \mu \tag{3}$$

Ideally, the goal of Eq. (3) can be achieved if the network estimate any image, which is perturbed by the patch p_t, as label $\widehat{y}$. Lack of data prevents us from extracting features of the target class $\widehat{y}$ from the training example x like existing methods [9,19]. Note that the non-targeted patch p_{nt} generated by Eq. (2) can effectively attack the network and lead the network to recognize many of the adversarial examples as dominant labels, which implies that the patch may contain some information about the training data. We believe that this prior knowledge helps generate targeted patch p_t without training data. In order to craft such a p_t, we optimize for

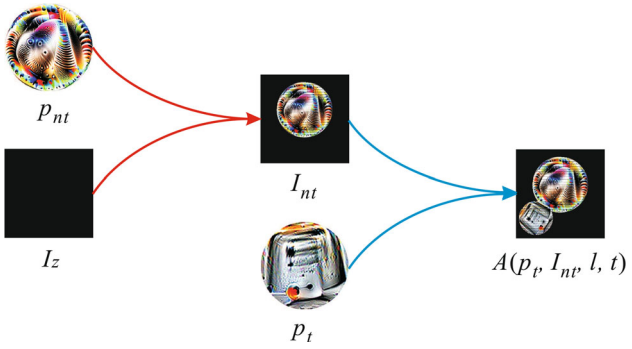


Fig. 5 The process of constructing the background image $A(p_t, I_{nt}, l, t)$ for targeted attack. After random transformation, p_{nt} is digitally inserted on a random position of the image I_z to craft I_{nt}. We take I_{nt} as the background image in Eq. (4), and explore its implicit information to help generate the targeted patch p_t

the following objective

$$\arg\max_{p_t} \mathbb{E}_{l\sim L, t\sim T}\left[\log \Pr(\widehat{y}|A(p_t, I_{nt}, l, t)\right] \tag{4}$$

where the background image I_{nt} is defined as

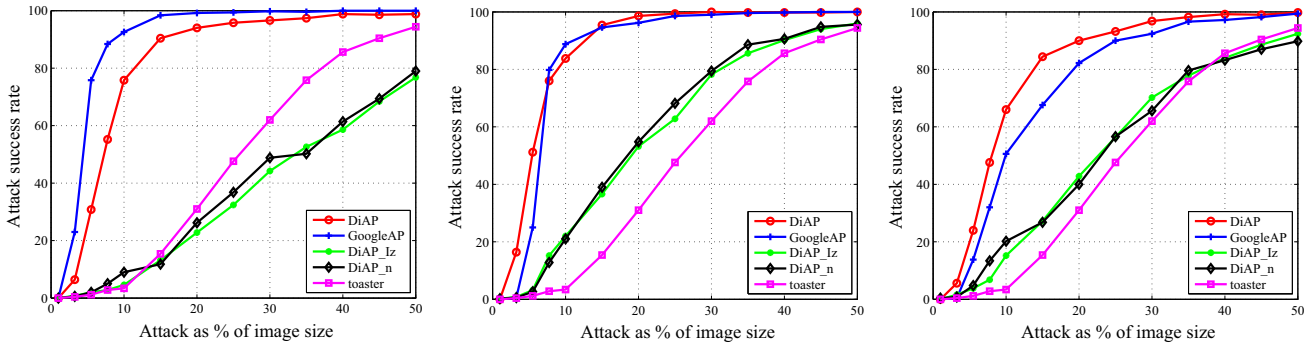
$$I_{nt} = A(p_{nt}, I_z, l, t) \tag{5}$$

In other words, we take $A(p_{nt}, I_z, l, t)$ as the background image. During the training, the non-targeted patch p_{nt} is randomly scaled and rotated, and then digitally inserted on a random location on the image I_z to form a background image I_{nt}. Next, the targeted patch p_t is also randomly transformed and digitally inserted on a random location on the background image I_{nt}. Figure 5 shows this process.

Using $A(p_{nt}, I_z, l, t)$ as the background image seems strange, as it is presented in Eq. (2) to train non-targeted attack patch p_{nt}. However, this approach is reasonable if we aware that p_{nt} probably contain information about training data, although p_{nt} does not belong to the training data. We seek an effective way to use the information implied in the non-targeted patch p_{nt} (i.e. optimize Eq. (4)), leading the image $A(p_t, I_{nt}, l, t)$ to be evaluated as the target label $\widehat{y}$. Finally, covered by the targeted patch p_t, the adversarial example $A(p_t, x, l, t)$ is misclassified as $\widehat{y}$ by the network. It is worth noting that during the entire optimization process, we did not use information about training data at all. The entire process is detailed in Algorithm 2.

4.2 Experiments and results

We experiment with the same settings as in Sect. 3.2. To evaluate the performance of DiAP for targeted attack, we introduce GoogleAP for comparison. We also inspect our DiAP in two other ways to verify the effect of the background image I_{nt}. DiAP_Iz adopts image I_z instead of image I_{nt} as



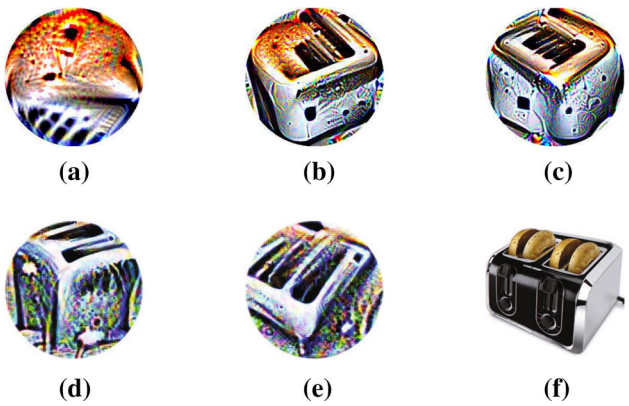
(a) Whitebox-Single Model **(b)** Whitebox-Ensemble **(c)** Blackbox

Fig. 6 Attack success rates of targeted attack. The target class is *toaster*. DiAP adopts I_{nt} as the background image to generate adversarial patches for targeted attack. DiAP_Iz replace I_{nt} with I_z, and DiAP_n use random noise as the background image. A real toaster image is also tested for comparison. All attack success rates are the average values calculated by the five models

the background image in Eq. (4), and DiAP_n replace I_{nt} with randomly noise as the background image. In the process of training, we use the gradient descent optimizer to craft adversarial patches, and set the hyper-parameter $Ite_{max} = 800$ and $\eta = 8$ for Algorithm 2.

Figure 6 shows the attack success rates of targeted adversarial patches, and the target class is *toaster*. The test images are the same as those used in Sect. 3.2. In the whitebox single model attack, GoogleAP's attack success rates are higher than that of DiAP. Nevertheless, when the patch takes 10% of the image size, about 80% of the attack success rate still can be achieved by DiAP. In the whitebox ensemble attack, the performance of GoogleAP and DiAP is very close. Further, in the blackbox scenario, DiAP performs even better than GoogleAP. Noting that DiAP is completely unaware of any information about the training data while GoogleAP uses training examples for optimization, this result seems counterintuitive. We presume that, without knowing the structure of the attacked model (blackbox scenario), the vague information about the training data provided by I_{nt} makes DiAP more generalized, resulting in the higher attack success rates than GoogleAP. This indicates that when the attacker is not acquainted with the attacked model, fuzzy information about training data may be more beneficial to generating attack patches than accurate example knowledge. In addition, DiAP is far more effective than using I_z or random noise as the background image, as is shown in Fig. 6 by the relatively poor performance of DiAP_Iz and DiAP_n.

Furthermore, we wonder that whether real photographs of target class could fool the deep neural network, just like DiAP. A real toaster image is digitally inserted into testing images in the same way as the adversarial patch. As shown in Fig. 6, to achieve a 90% attack success rate, the real toaster image has to cover about 50% of the testing image. However, in this case, the new adversarial image actually turns into a picture of the toaster.

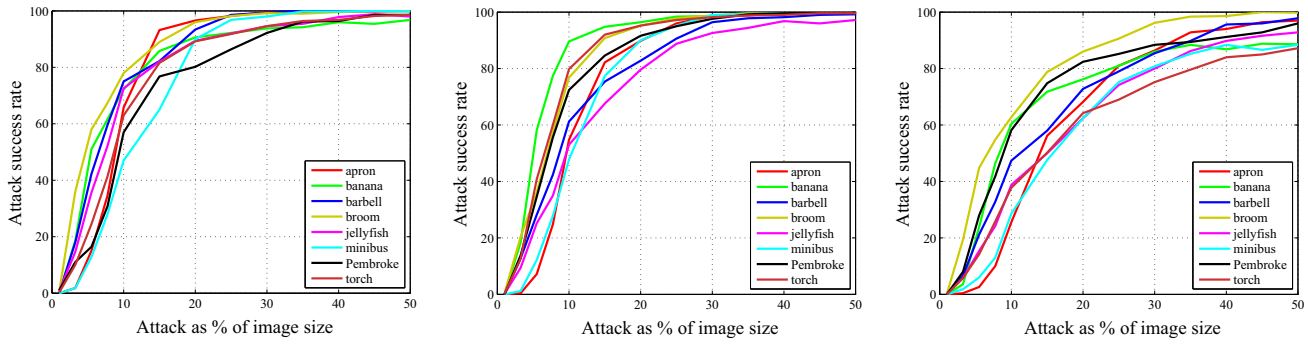


(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Fig. 7 Adversarial patches for targeted attack. **a**, **b** and **c** are generated by DiAP for the whitebox single model attack, the whitebox ensemble attack, and the blackbox attack, respectively. **d** is crafted by DiAP_Iz for the blackbox attack. **e** is computed by DiAP_n for the blackbox attack. The target model is VGG-16 and the targeted label is *toaster*. **f** is the photo of a real toaster, which is used in the experiments shown in Fig. 6

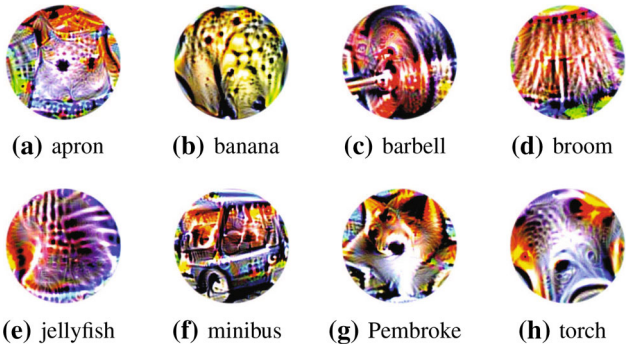
Figure 7 shows some patches that DiAP generates for targeted attacks. These patches are very similar to the target class (real toaster), although DiAP doesn't know what the toaster looks like since it's data independent and isolated from real examples during training. We presume that DiAP learns high-level features of the target category, making the adversarial patch more toaster-like than the real toaster picture in the vision of the deep neural networks.

We randomly select other 8 categories as target labels (e.g. banana, jellyfish), and the experimental results are shown in Fig. 8. The adversarial patches are generated by DiAP. When the patch occupies 20% of the input image area, the success rate of both whitebox attacks exceeds 80% for all target classes. When the patch occupies 20% of the input image area, for all target classes, the attack success rates of both whitebox methods exceed 80%, and that of the blackbox sce-



(a) Whitebox-Single Model **(b)** Whitebox-Ensemble **(c)** Blackbox

Fig. 8 Attack success rates of targeted adversarial patches generated by DiAP. Eight target labels are randomly selected



(a) apron **(b)** banana **(c)** barbell **(d)** broom

(e) jellyfish **(f)** minibus **(g)** Pembroke **(h)** torch

Fig. 9 Eight adversarial patches crafted by DiAP for targeted attack. All these patches are generated for the whitebox single model attack, and the target model is VGG-16 and the targeted labels are listed blow them. These patches look similar to their target classes

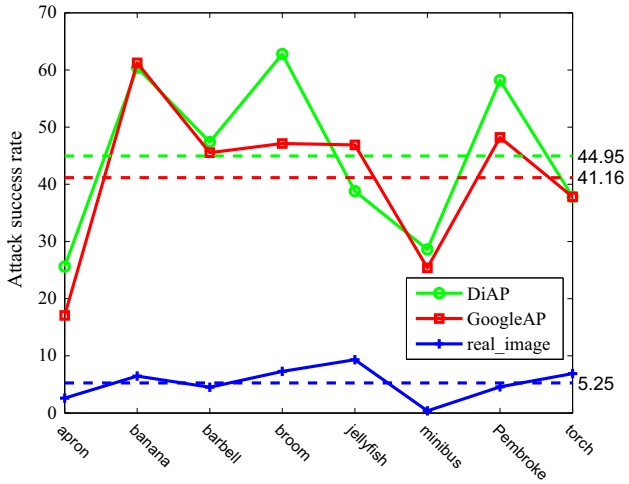


Fig. 10 Attack success rates in the blackbox attack scenario. The X-axis represents the 8 target categories. The Y-axis indicates the attack success rate against the target class when the adversarial patch (or real image) covers 10% of the test image area

nario exceeds 60%. Moreover, the attack success rate varies depending on the target class. Taking the whitebox single model attack with the scaling parameter of 10% as an example, the lowest success rate is 47.35% and the highest success rate is 78.56% for different target classes. However, there is no strict correspondence between the attack success rate and the target class in the three attack settings. For example, in the blackbox scenario, the attack patches generated with the target category *broom* achieve higher attack success rates than other target categories, while in the whitebox ensemble attack, the attack patches generated with the target category *banana* perform better.

Figure 9 shows patches generated by DiAP. To some extent, these patches have many similarities with their target classes. For example, the patch (b) appears to be composed of many small bananas, patch (g) shows that it contains Pembroke's eyes and nose, and patches (c), (d), and (f) even seem to be distorted images of their target categories. This further confirms our inference that the adversarial patch actually extracts the high-level features of the target category.

The attack performance of DiAP in the blackbox scenario is particularly noteworthy, which means that malicious attackers can attack deep neural networks without knowing the model structure, pre-training parameters, and training dataset. Figure 10 shows the attack success rate in the blackbox attack scenario when the scaling parameter is 10%. We examine the attack ability when the patch occupies 10% of the test image area. In most cases, DiAP outperforms GoogleAP, and DiAP achieves an average attack success rate of 44.95% compared to GoogleAP's 41.15%. This suggests that patches generated by DiAP have better transferability and are more suitable for blackbox attacks. At the same time, 8 real images are randomly selected for comparison, one for each target class. The attack success rates of real pictures are all less than 10%, and the average is only 5.25%. These results confirm that our adversarial patches, rather than real image of the target label, can attack deep neural networks.

Fig. 11 A real-world attack example. The targeted patch is generated for whitebox ensemble attack and the test model is the pre-trained Inception-V3 mentioned in 3.2. The photographs on the left are correctly classified. However, after being inserted into the targeted patch, the photographs on the right are all classified as toaster, in spite of the different scale, location, angle of the patch

5 Physical world attacks

In this section, a physical world attack experiment is conducted to validate the practical effectiveness. We use a targeted patch (as shown in Fig. 7b) generated by DiAP, which is crafted through the whitebox ensemble method. The test classifier is the pre-trained Inception-V3 model mentioned in Sect. 3.2. The attack is considered successful if the pre-trained Inception-V3 model correctly identifies the original image and misclassifies the adversarial example as the specified label. After printing this patch by a Canon IP8780 printer, we place it in a variety of real world scenes and take photographs with the combination of different distances and angles using a MI 8 phone. The images and results are shown in Fig. 11 demonstrate that the attack successfully fools the network, regardless of the scale, location, angle of the patch.

6 Conclusion

In this paper, we presented a data independent approach to generating adversarial patches. By contaminate the extracted features at each layer of the attacked network, DiAP generate non-targeted adversarial patches. Then, combined with the information implied by the non-targeted patch, DiAP extract the features of the target class to craft the targeted adversarial patches. In the process of generating patches, DiAP does not use any training data at all, which is conducive to performing attacks in real physical scenarios. The extensive experimental results, under whitebox and blackbox settings in both digital and physical world, demonstrate that DiAP owns strong attack capabilities, and achieves state-of-the-art performance. Further, DiAP outperforms state-of-the-art method in blackbox attack, which implies that when the network structure is unknown, fuzzy example information may be more helpful for attack than real examples. To encourage reproducible research, the code of DiAP is made available at https://github.com/zhouxy2020/DiAP.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings (2015)
2. Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017, pp. 86–94. IEEE Computer Society (2017)
3. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: Imagenet large scale visual recognition challenge (Int. J. Comput. Vis.) (115) 211–252 (2015)
4. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings (2014)
5. Athalye, A., Engstrom, L., Ilyas, A., Kwok, K.: Synthesizing robust adversarial examples. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018, Volume (80) of Proceedings of Machine Learning Research, PMLR, pp. 284–293 (2018)
6. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Workshop Track Proceedings, OpenReview.net (2017)
7. Sharif, M., Bhagavatula, S., Bauer, L., Reiter, M.K.: Accessorize to a crime: eeal and stealthy attacks on state-of-the-art face recognition. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24–28, 2016, pp. 1528–1540. ACM (2016)
8. Komkov, S., Petiushko, A.: Advhat: real-world adversarial attack on arcface face ID system (CoRR) (abs/1908.08705) (2019) . arXiv:1908.08705
9. Brown, T.B., Mané, D., Roy, A., Abadi, M., Gilmer, J.: Adversarial patch (CoRR) (abs/1712.09665) (2017). arXiv:1712.09665

10. Liu, A., Liu, X., Fan, J., Ma, Y., Zhang, A., Xie, H., Tao, D.: Perceptual-sensitive GAN for generating adversarial patches. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27–February 1, 2019, pp. 1028–1035. AAAI Press (2019)
11. Liu, A., Wang, J., Liu, X., Cao, B., Zhang, C., Yu, H.: Bias-based universal adversarial patch attack for automatic check-out. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J. (eds.) Computer Vision—ECCV 2020—16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII, Volume (12358) of pp. 395–410 (2020)
12. Thys, S., Ranst, W.V., Goedemé, T.: Fooling automated surveillance cameras: Adversarial patches to attack person detection. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16–20, 2019, pp. 49–55. Computer Vision Foundation/IEEE (2019)
13. Wu, Z., Lim, S., Davis, L.S., Goldstein, T.: Making an invisibility cloak: real world adversarial attacks on object detectors. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J. (eds.) Computer Vision—ECCV 2020—16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV, Volume (12349) of pp. 1–17 (2020)
14. Mopuri, K.R., Ganeshan, A., Babu, R.V.: Generalizable data-free objective for crafting universal adversarial perturbations, (IEEE) Trans. Pattern Anal. Mach. Intell. **41**, 2452–2465 (2019)
15. Chen, P., Sharma, Y., Zhang, H., Yi, J., Hsieh, C.: EAD: elastic-net attacks to deep neural networks via adversarial examples. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), The 30th Innovative Applications of Artificial Intelligence (IAAI-18), and The 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2–7, 2018, pp. 10–17. AAAI Press (2018)
16. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30–May 3, 2018, Conference Track Proceedings, OpenReview.net (2018)
17. Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., Song, D.: Robust physical-world attacks on machine learning models (CoRR) (abs/1707.08945) (2017). arXiv:1707.08945
18. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks, (IEEE) Trans. Evol. Comput. **23**, 828–841 (2019)
19. Karmon, D., Zoran, D., Goldberg, Y.: Lavan: localized and visible adversarial noise. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018, Volume (80) of Proceedings of Machine Learning Research, PMLR, pp. 2512–2520 (2018)
20. Mopuri, K.R., Garg, U., Radhakrishnan, V.B.: Fast feature fool: a data independent approach to universal adversarial perturbations. In: British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4–7, 2017. BMVA Press (2017)
21. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016, pp. 2818–2826. IEEE Computer Society (2016)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016, pp. 770–778. IEEE Computer Society (2016)
23. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017, pp. 1800–1807. IEEE Computer Society (2017)
24. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings (2015)

Xingyu Zhou received the Ph.D. degree from the Army Engineering University of PLA, Nanjing, China, in 2019. He is currently a lecturer with the Army Engineering University of PLA. His research interests include computer vision, image processing and pattern recognition.

Zhisong Pan received the Ph.D. degree of computer science and technology from Nanjing University of Aeronautics and Astronautics, China, in 2003. He is now a Professor with the Army Engineering University of PLA. His main research interests include deep learning, machine learning, and pattern recognition

Yexin Duan is currently pursuing the Ph.D. degree in computer science and technology with the Army Engineering University of PLA, Nanjing, China. His research interests include adversarial machine learning and computer vision.

Jin Zhang is currently pursuing the Ph.D. degree in computer science and technology with the Army Engineering University of PLA, Nanjing, China. He is currently a Senior Lecturer with the Army Military Transportation University of PLA, Zhenjiang Campus, Zhenjiang, China. His research interests include computer vision and machine learning.

Shuaihui Wang received the Ph.D. degree in computer science and technology with the Army Engineering University of PLA, Nanjing, China. His main research interests include graph data mining and graph neural networks.