

Query-Efficient Decision-Based Black-Box Patch Attack

Zhaoyu Chen^{ID}, *Graduate Student Member, IEEE*, Bo Li^{ID}, Shuang Wu^{ID}, Shouhong Ding,
and Wenqiang Zhang^{ID}, *Member, IEEE*

Abstract—Deep neural networks (DNNs) have been showed to be highly vulnerable to imperceptible adversarial perturbations. As a complementary type of adversary, patch attacks that introduce perceptible perturbations to the images have attracted the interest of researchers. Existing patch attacks rely on the architecture of the model or the probabilities of predictions and perform poorly in the decision-based setting, which can still construct a perturbation with the minimal information exposed – the top-1 predicted label. In this work, we first explore the decision-based patch attack. To enhance the attack efficiency, we model the patches using paired key-points and use targeted images as the initialization of patches, and parameter optimizations are all performed on the integer domain. Then, we propose a differential evolutionary algorithm named DevoPatch for query-efficient decision-based patch attacks. Experiments demonstrate that DevoPatch outperforms the state-of-the-art black-box patch attacks in terms of patch area and attack success rate within a given query budget on image classification and face verification. Additionally, we conduct the vulnerability evaluation of ViT and MLP on image classification in the decision-based patch attack setting for the first time. Using DevoPatch, we can evaluate the robustness of models to black-box patch attacks. We believe this method could inspire the design and deployment of robust vision models based on various DNN architectures in the future.

Index Terms—Adversarial example, patch attack, black-box attack, differential evolutionary algorithm.

I. INTRODUCTION

NOWADAYS, deep neural networks (DNNs) have been employed as the fundamental techniques in the advancement of artificial intelligence in computer vision. Despite the success of DNNs, recent studies have identified that DNNs are vulnerable to adversarial examples [1]. By introducing maliciously crafted perturbations to the input images, these adversarial examples are able to evade and mislead DNNs. Consequently, studying the adversarial vulnerability of DNNs has emerged as an important research area, providing the

Manuscript received 17 June 2022; revised 19 January 2023; accepted 23 July 2023. Date of publication 23 August 2023; date of current version 8 September 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62072112, in part by the Scientific and Technological Innovation Action Plan of Shanghai Science and Technology Committee under Grant 22511102202, and in part by the Fudan Double First-Class Construction Fund under Grant XM03211178. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ashish Khisti. (*Corresponding authors: Bo Li; Wenqiang Zhang*)

Zhaoyu Chen and Wenqiang Zhang are with the Academy for Engineering and Technology, Fudan University, Shanghai 200433, China, and also with the Yiwu Research Institute, Fudan University, Yiwu 322000, China (e-mail: zhaoyuchen20@fudan.edu.cn; wqzhang@fudan.edu.cn).

Bo Li, Shuang Wu, and Shouhong Ding are with the YouTu Laboratory, Tencent, Shanghai 200233, China (e-mail: njumagiclibo@gmail.com; calvinwu@tencent.com; ericshdng@tencent.com).

Digital Object Identifier 10.1109/TIFS.2023.3307908

opportunity to better understand and improve computer vision models.

Classical works [1], [2], [3], [4], [5], [6], [7] focus on studying the adversarial vulnerability of DNNs against virtually imperceptible perturbations that are constrained to have a small norm but are typically applied to the whole input image. Recently, as a complementary type of adversary, patch attacks that introduce perceptible (large norm) but localized perturbations to the images have attracted the interest of researchers. Pioneering works [8], [9], [10], [11], [12] perform patch attacks in the white-box setting: with full access to the model’s parameters and architectures, they can directly use gradient-based optimization to find successful adversarial examples. Due to the fact that most real-world applications do not publicly release the actual models they use, this attack scenario usually is less practical in real-world systems, e.g., attacking image analysis APIs [13] like Google Cloud Vision or self-driving cars [14], [15], [16].

As a more practical scenario in real-world systems, black-box patch attacks have attracted a lot of attention in recent years. There are transfer-based attacks [17] and query-based attacks [18], [19], [20], [21] for black-box patch attacks, depending on whether the attacker needs to query the victim’s machine learning model. Despite the fact that transfer-based attacks do not require query access to the model, it assumes the attacker has access to a large training set to create a carefully-designed substitute model [22], [23], [24], and there is no guarantee of success [25]. Query-based attacks assume that attackers can only query the target network and obtain its outputs (score or label) for a given input. According to the output information of the queried models, query-based attacks can be classified into two sub-categories: score-based setting which has access to the class probabilities of the model, and decision-based setting which solely relies on the top-1 predicted label. Significantly, decision-based settings present more practical threats to deployed systems and applications because an adversary is still capable of exploiting the very minimal information exposed – the top-1 predicted label – for constructing an adversarial perturbation. Recently, some score-based patch attacks [18], [19], [20], [21] have been proposed. However, when these methods are applied to the decision-based setting, they hardly achieve high attack success rate and query efficiency because the information provided by labels is limited.

In this paper, we first explore the decision-based patch attack to better measure the practical threat of patch attacks. To successfully conduct decision-based black-box

patch attacks, there are still non-negligible challenges to overcome:

A. Complex Solution Space

Performing patch attacks is extremely challenging since it involves searching for all possible positions, shapes, and perturbations of adversarial patches, which implies an enormous solution space. Moreover, unlike white-box scenarios or the score-based black-box setting, in the decision-based black-box setting, there is almost no valid information to guide the search direction.

B. Query Efficiency

In the query-based setting, achieving high query efficiency with a high attack success rate is integral to adversarial objectives. Because: i) adversaries are able to carry out attacks at scale; ii) the cost of mounting the attack is reduced, and iii) adversaries are capable of bypassing defense systems that can recognize malicious activities as a fraud based on a pragmatically large number of successive queries with analogous inputs. Last but not least, the advantage of a smaller query budget is that it correlates to a lower cost of evaluation and research, which is useful for determining the robustness of the model to adversarial attacks.

To address the aforementioned issues, we propose a differential evolutionary algorithm named **DevoPatch** for query-efficient adversarial patch attacks in the decision-based black-box setting. Differential evolutionary algorithm is a black-box optimization algorithm that does not need to know the details of the model and is suitable for parameter search when information is limited. Given the attack objective function, DevoPatch is able to optimize it in a black-box manner through queries only. To simplify the solution space, we restrict parameter optimization to the integer domain and carefully design a differential evolution algorithm based on the integer domain. Further, we model the patches using paired key-points and use targeted images as the initialization of patches. Consequently, the query efficiency of DevoPatch is significantly improved. In addition, it is worth noting that some novel DNN architectures have recently emerged including the Vision Transformer (ViT) model [26] and Multi-layer Perceptron (MLP) based model [27]. They demonstrate compelling performance, sometimes even outperforming classical convolutional architectures. Although a few studies have explored the vulnerability of ViT against imperceptible adversarial perturbations [28], [29], the adversarial robustness of ViT and MLP under patch attacks has not been considered. This raises a critical security concern for the reliable deployment of real-world applications based on ViT and MLP models. Therefore, we extend our study scope and apply DevoPatch to ViT and MLP to better understand the vulnerability of a wide variety of DNNs under adversarial patch attacks. We illustrate an example patch attack with DevoPatch against ILSVRC2012 in Fig. 1 on image classification. Extensive experiments on image classification and face verification demonstrate that DevoPatch is a query-efficient decision-based black-box patch attack.

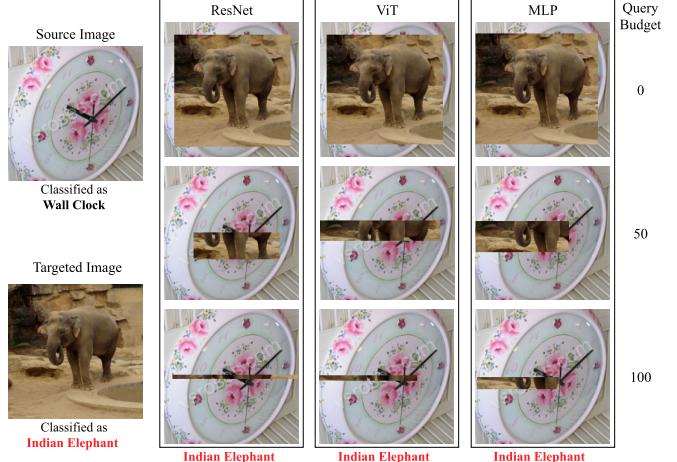


Fig. 1. Introduction of DevoPatch. With regard to limited query budgets, we generate adversarial examples of patch attacks using DevoPatch applied to black-box models on image classification. As the number of queries increases, DevoPatch efficiently optimizes the quality of adversarial patches and achieves query-efficient decision-based patch attacks under a few query budgets.

We summarize our contributions and results below:

- We first explore the decision-based patch attack, which can still construct a perturbation with the minimal information exposed – the top-1 predicted label.
- To simplify the solution space, we model the patches using paired key-points and use targeted images as the initialization of patches, and parameter optimizations are all performed on the integer domain.
- We propose a novel patch attack – DevoPatch – an evolutionary algorithm capable of exploiting access to solely the top-1 predicted label from a model to search for an adversarial example, whilst minimizing the image area that needs to be corrupted for a successful attack.
- Comprehensive experiments on image classification and face verification show that DevoPatch achieves considerably higher success rates compared to related work, while being more efficient in terms of the number of queries.
- We conduct the vulnerability evaluation of ViT and MLP on image classification in the decision-based black-box patch attack setting for the first time. We compare results with ResNet to assess the relative robustness of the ViT and MLP models.

The remainder of the paper is organized as follows. Section II briefly reviews the literature related to adversarial examples and adversarial patches, white-box patch attacks, black-box patch attacks, and adversarial attacks with evolutionary algorithms. Section III first introduces the definition of decision-based black-box patch attacks and then details the proposed differential evolutionary patch attack. Section IV shows the experimental results to demonstrate the effectiveness of the proposed differential evolutionary patch attack. Firstly, we choose appropriate hyperparameters for DevoPatch. Afterward, we evaluate the adversarial robustness of several image classification and face recognition models. In Section V, we further analyze the effects of adversarial patches on different DNN architectures. We summarize the paper in Section VI.

II. RELATED WORK

In this section, we briefly review the literature related to adversarial examples and adversarial patches, white-box patch attacks, and black-box patch attacks. In the end, we also discuss adversarial attacks based on evolutionary algorithms.

A. Adversarial Example and Adversarial Patch

The seminal works of Szegedy et al. [1] inspire an interest in studying adversarial vulnerability against imperceptible perturbations as a mean of understanding and improving deep neural networks. Since then, a majority of prior works [2], [3], [4], [5], [6], [7] have focused on attacking with small and imperceptible perturbations to the input, which can be regarded as *the imperceptible adversarial attack*. Commonly these imperceptible perturbations are applied to the whole input image and are constrained by p-distances ($p \in \{0, 2, \infty\}$) similarity measurement. Recently, as a complementary type of adversary, patch attacks that introduce perceptible (large norm) but localized perturbations to the images have emerged and attracted the interest of researchers. Patch attacks (or adversarial patches) can be regarded as *the perceptible adversarial attack*. The main aim of patch attacks is to minimize the perturbation within a continuous image region that needs to be corrupted to mislead a target machine learning model. Only a handful of works have investigated patch attacks and these works can be broadly categorized based on various degrees of adversarial access to a model. In this paper, we focus on black-box patch attacks because they are more practical and more threatening.

B. White-Box Patch Attack

In the white-box setting, an adversary has full knowledge and access to the model, including gradients and parameters. GAP [8] first creates universal, robust, targeted adversarial image patches in the real world and causes a classifier to output any target class in the white-box setting. Then LaVAN [9] concentrates on investigating the blind spots of state-of-the-art image classifiers in the digital domain, which crafts adversarial patches using an optimization-based approach with a modified loss function. Then [10] and [11] introduce position search to improve the attack performance of adversarial patches. Due to the enormous solution space and the trade-off between computational cost and attack performance, adversarial patches are usually created with a fixed shape or location even under the white-box setting. Since then, adversarial patches have been used to attack self-driving cars [14], [15], [16], object detection [30], [31], [32] and face cognition [33], [34], [35]. However, white-box patch attacks are less practical, since most real-world applications do not release their models and cannot directly solve adversarial patches via gradients. In this paper, we focus on black-box patch attacks because they are more threatening to real-world systems.

C. Black-Box Patch Attack

Black-box patch attacks can be either transfer-based [17] or query-based [18], [19], [20], [21], depending on whether

the attacker needs to query the victim's machine learning model. However, transfer-based attacks require access to large amounts of training data and require careful construction of surrogate models. It does not guarantee that the attack will be successful. In contrast, query-based attacks only require access to the output of the victimized model and have a higher attack success rate as the number of queries increases, which is more practical and more threatening. In the query-based attack, an adversary can access all or only one predicted score (score-based settings) or call out just the predicted labels (decision-based settings) for a given input. We need a query-efficient algorithm that helps reduce the cost of evaluating the robustness of DNNs since the attacker has to pay for each query.

Query-based patch attacks are first introduced in Hastings Patch Attack (HPA) [18]. They do not optimize the pattern of the patches and instead use the monochrome patches. The position and shape of the rectangular patches are randomly searched using Metropolis-Hastings sampling. To improve the query efficiency of HPA, [19] first uses reinforcement learning to search the position and size of monochrome rectangular patches, called Monochrome Patch Attack (MPA). But monochrome patches usually lead to a very low attack success rate, especially for the targeted attack. They then use ImageNet training data to build a class-specific texture dictionary via style transfer [36] to craft targeted patch attacks, termed Texture-based Patch Attack (TPA). However, in practical scenarios, it is impossible to obtain the whole training set data of the target black-box models. Adv-watermark (AdvW) [20] utilizes the basin hopping evolution algorithm to find a suitable position and transparency for the watermark to implement the patch attack. Patch-Rs [21] proposes a random search framework and then designs an initialization scheme and a sampling distribution specific for adversarial patches. This outperforms previous works in both query efficiency and attack success rates. Unfortunately, all of the aforementioned works are only designed for the score-based black-box setting. They perform poorly on the more challenging and restrictive decision-based attack (Experiment IV-C). Further, decision-based settings present more practical threats to deployed systems. To the best of our knowledge, we make the first investigation of the robustness of DNNs against patch attacks using a decision-based black-box setting.

D. Adversarial Attacks With Evolutionary Algorithms

Notably, there are some related works [37], [38], [39] which also leverage evolutionary algorithms to perform *imperceptible attacks*. These methods are all under the framework of evolutionary algorithms, with operations such as crossover and mutation. However, these methods cannot achieve query-efficient decision-based patch attacks due to the limitations of the modeling of adversarial examples. One-pixel attack [37] generates one-pixel adversarial perturbations based on differential evolution. It is not effective when the number of perturbations is large because the number of queries to the model grows rapidly with respect to the number of perturbed pixels in patches. Evo-Attack [38] utilizes the covariance

matrix adaptation evolution strategy to search the imperceptible perturbations but it cannot search for the location and shape of patches. SparseEvo [39] models sparse perturbations as binary codes and solves them using genetic algorithms. However, this binary representation cannot define contiguous regions, thus making it impossible to model patches and perform patch attacks. Our DevoPatch is based on the differential evolution algorithm, carefully designed in the integer domain to achieve query-efficient decision-based patch attacks. Consequently, we first construct a dimensionality-reduced solution space in which possible solutions (individuals of a population) are paired key-points in the integer domain. This is quite different from most current evolutionary attack methods.

III. METHODOLOGY

In this section, we first introduce the definition of decision-based black-box patch attacks and then detail the proposed differential evolutionary patch attack.

A. Problem Definition

Patch attacks are one of the most threatening types of adversarial examples that an adversary can arbitrarily modify the pixels of a continuous region, and the patch of this region leads machine learning models to make incorrect predictions. Here, we first give the formulation of patch attack on image classification. Face verification can be viewed as a binary classification task, similar to image classification. For an classifier $f : \mathbf{x} \rightarrow \mathbf{y}$, we are given a source image $\mathbf{x} \in \mathbb{R}^{C \times H \times W}$ and its corresponding ground truth label \mathbf{y} from the label set $\mathbb{Y} = \{1, 2, \dots, K\}$ where K denotes the number of classes. C , W , and H denote the number of channels, height, and width of an image, respectively. In the setting of patch attacks, the adversarial patch is composed of adversarial perturbations $\delta \in \mathbb{R}^{C \times H \times W}$ and location masks $M \in \{0, 1\}^{H \times W}$. Given a source image \mathbf{x} , we formulate the adversarial example $\tilde{\mathbf{x}}$ as the combination of a source image \mathbf{x} , an adversarial patch δ and a location mask M :

$$\tilde{\mathbf{x}} = (I - M) \odot \mathbf{x} + M \odot \delta, \quad (1)$$

where \odot represents the element-wise Hadmard product and I represents all-one matrices with the same dimension as M .

In the decision-based black-box setting, our access is limited to its output label. For the targeted attack, the adversary perturbs the source image \mathbf{x} so that the obtained adversarial example $\tilde{\mathbf{x}} \in \mathbb{R}^{C \times H \times W}$ is misclassified as the desired class label $\tilde{\mathbf{y}} \in \mathbb{Y}$. We refer to the desired class $\tilde{\mathbf{y}}$ of the input \mathbf{x} as the target class and its ground-truth class \mathbf{y} as the source class. For the untargeted attack, the adversary perturbs the source image \mathbf{x} to lead the output label of the classifier to any class label except the ground truth label \mathbf{y} , i.e. $\tilde{\mathbf{y}} \in \mathbb{Y}$ where $\tilde{\mathbf{y}} \neq \mathbf{y}$. In general, the patch attack (including targeted and untargeted settings) to find the best adversarial example \mathbf{x}^* can be expressed as a constrained optimization problem:

$$\mathbf{x}^* = \arg \min_{M, \delta} \|\mathbf{x} - \tilde{\mathbf{x}}\|_0 \quad s.t. \quad f(\mathbf{x}^*) = \tilde{\mathbf{y}}, \quad (2)$$

where $\|\cdot\|_0$ denotes the number of perturbed pixels. For the patch not to be perceived, Eq. 2 aims to determine the

perturbation and position with the constraint of a few perturbed pixels, which leads to a complex solution space and hampers the search. In addition, given the constraint and the fact that f is not differentiable in the decision-based setting, the solution to the optimization problem is not trivial.

B. Simplification on Solution Space

The enormous solution space on patch attacks is caused by all possible positions, shapes, and perturbations of the patch. A naive parametric search method can be directly used to solve this problem. Specifically, the parameter set V is defined as a series of candidate solutions \mathbf{v} , represented by the coordinates and RGB values of each pixel. However, this naive application results in very inefficient queries [37]. Furthermore, in the decision-based black-box setting, there is little effective information to guide the search direction, thus further reducing the query efficiency. To improve the query efficiency of decision-based attacks, we need to reduce the complexity of the solution space. Although in the decision-based setting, the black-box model can hardly provide effective information so the only information we can fully utilize is the target class $\tilde{\mathbf{y}}$ of the targeted attack. To facilitate a parametric search method, instead of searching for parameters defining RGB values of each perturbed pixel, we consider that the perturbation δ can be replaced by a *targeted image* \mathbf{x}_t . Targeted images are only required to be classified as target class by the black-box model and do not need to be i.i.d. with the training set (analyzed in Section IV-G.2). Simultaneously, it is redundant to represent a patch with a coordinate set of perturbed pixels. For a patch, we only need to know a pair of points to formulate the location mask M of the patch. Therefore, we vectorize each candidate solution in the parameter set V as a 4-dimensional vector $\mathbf{v} = \{(i_1, j_1), (i_2, j_2)\}$ ($i_1 < i_2, j_1 < j_2$) where $i \in \mathbb{N}$ and $j \in \mathbb{N}$ denotes the coordinate of the paired key-points. Here, we employ a simple mapping function $T(\cdot)$ to re-formulate the location mask $M = T(\mathbf{v})$ and the adversarial example $\tilde{\mathbf{x}}$:

$$T(\mathbf{v}) = \begin{cases} 1, & \text{if } 0 \leq i_1 < i_2 < H, 0 \leq j_1 < j_2 < W, \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

$$\tilde{\mathbf{x}} = (I - M) \odot \mathbf{x} + M \odot \mathbf{x}_t. \quad (4)$$

In general, we transform the original complex solution space into a coordinate programming problem for paired key-points on the integer domain. Interestingly, this strategy of simplification has been found to be extremely effective in a decision-based patch attack. Next, we need to design how to select and update suitable candidate solutions.

C. Differential Evolutionary Patch Attack

In this section, we propose the **DevoPatch**, an efficient parametric search method based on the differential evolutionary algorithm that seeks a solution by iteratively improving upon potential solutions in search of a desirable one. In differential evolution, the population is the candidate solution and the population set is the parameter set. We carefully design the differential evolution algorithm on the integer domain, including population initialization, mutation, and crossover.

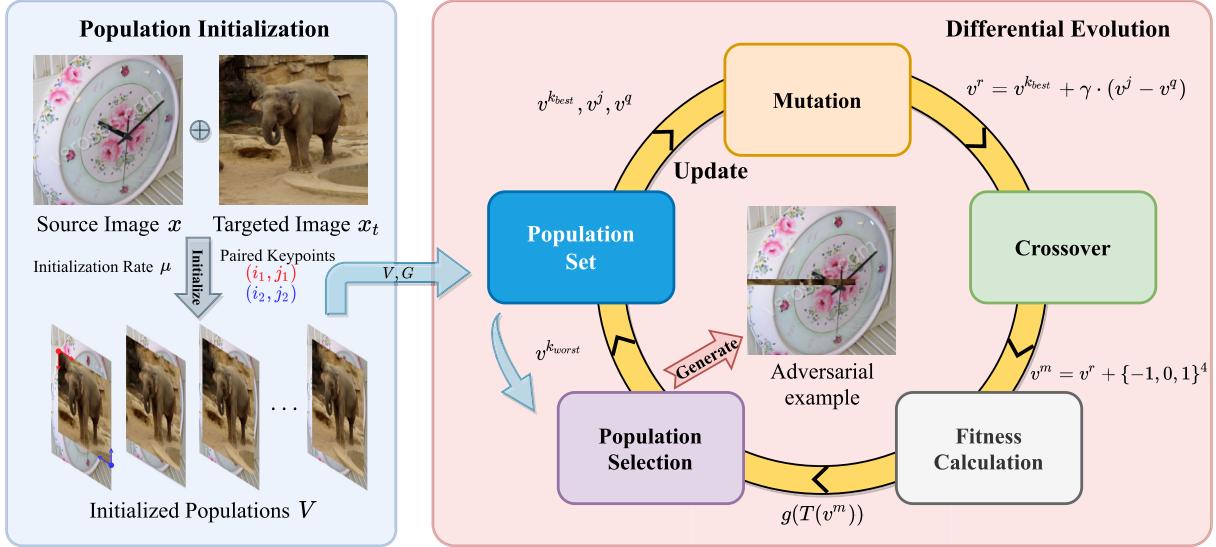


Fig. 2. The pipeline of **DevoPatch**. The Population Initialization stage creates the initialized populations. During the differential evolution, by a combination of mutation, crossover, fitness calculation, and population selection, the population can improve over time to produce a satisfactory adversarial example.

Therefore, DevoPatch improves the differential evolution by simplifying solution spaces to the integer domain and the population can improve over time to produce a satisfactory result. Moreover, our search method employs the differential evolutionary algorithm without requiring any background knowledge of the target model, such as its architecture or parameters, to construct the fitness function. DevoPatch can be used to analyze and solve the non-trivial optimization problem in Eq. 2 in a black-box setting and can provide a possible remedy for complex solution space. The pipeline of DevoPatch is shown in Fig. 2.

First, we give the definition of fitness calculation. Fitness is used to evaluate the quality of candidate solutions, mainly used in population initialization and population selection. In general, fitness function $g(\cdot)$ should reflect optimization objectives. In the score-based setting, since logits can be obtained, cross-entropy loss or margin loss [5] can be used to measure the quality of candidate solutions. In the decision-based setting, since only the predicted labels can be obtained, it is difficult for us to use the change of loss to measure the quality of new candidate solutions. The loss only changes when the predicted label changes, which causes many potential candidates to be discarded. Therefore, a fitness function is required to approximate the calculation of the loss function in the decision-based setting. Since our populations describe a paired key-point of the patch and the method uses targeted images as initialization, we consider an adversarial example with a smaller patch area would have better quality. Therefore, we formulate our fitness function as:

$$g(\tilde{x}) = \begin{cases} ||x - \tilde{x}||_0, & \text{if } f(\tilde{x}) = \tilde{y} \\ \infty, & \text{otherwise.} \end{cases} \quad (5)$$

Although the fitness function is l_0 norm, other distance metrics are also feasible (further analyze in Section IV-B.5).

Then, we initialize a population set of p various candidate solutions named initialized population \mathbf{v}^0 . In the population initialization, it is trivial to apply targeted images directly as

initialization. The diversity among populations is conducive to improving query efficiency, so we introduce an *initialization rate* μ to control the diversity of population initialization. Specifically, we first calculate height margin $\Delta h = \lfloor H \cdot \mu \rfloor$ and width margin $\Delta w = \lfloor W \cdot \mu \rfloor$ ($\Delta h \in \mathbb{N}$, $\Delta w \in \mathbb{N}$) as candidate domains. Then, every candidate solution is generated by only a randomly sample in the following condition:

$$\begin{aligned} i_1 &\in [0, \Delta h), i_2 \in [H - \Delta h, H), \\ j_1 &\in [0, \Delta w), j_2 \in [W - \Delta w, W]. \end{aligned} \quad (6)$$

Finally, if the fitness score of the initialized population \mathbf{v}^0 is not ∞ by using Eq. 5, the initialized population \mathbf{v}^0 will be successfully added to the population set V . Fitness scores are saved in a fitness score vector G for each candidate solution. The population initialization is detailed in Algorithm 1.

Mutation is an important step in generating superior offspring (new candidate populations). Although the initialized population in the population initialization stage has a certain diversity, the overall difference is not significant, which will cause the next generation to be very similar and reduce query efficiency. In order to ensure the diversity of offspring, we introduce *mutation rate* γ to generate better offspring. Compared with the traditional differential evolutionary algorithm [40], we need to ensure that the calculation is closed and the solution set of offsprings is an integer domain, so *mutation rate* γ must be an integer, i.e. $\gamma = 1$. Specifically, when DevoPatch converges, the coordinate difference in candidate solutions is often only 1. If γ is a real number (i.e. $\gamma = 0.5$), it may be 0 after rounding to coordinates, resulting in no new offspring and reducing diversity. In practice, we first select the best $\mathbf{v}^{k_{best}}$ and two randomly selected candidate solutions $\mathbf{v}^j, \mathbf{v}^q$. Then, \mathbf{v}^r is based on $\mathbf{v}^{k_{best}}$ plus γ times the difference between \mathbf{v}^j and \mathbf{v}^q . Formally, the mutation can be formulated as:

$$\mathbf{v}^r = \mathbf{v}^{k_{best}} + \gamma \cdot (\mathbf{v}^j - \mathbf{v}^q). \quad (7)$$

Algorithm 1 Population Initialization Algorithm

Input: source image \mathbf{x} , ground-truth label y , targeted image \mathbf{x}_t , target label \tilde{y} , population size p , initialization rate μ and model f

Output: V, G

```

1:  $V \leftarrow \emptyset, G \leftarrow \infty$ 
2: for  $i \leftarrow 1, 2, \dots, p$  do
3:    $c \leftarrow 0$ 
4:    $\Delta h \leftarrow \lfloor H \cdot \mu \rfloor, \Delta w \leftarrow \lfloor W \cdot \mu \rfloor$ 
5:   while True do
6:     Generate  $\mathbf{v}^0$  with  $\Delta h, \Delta w$  using Eq. 6
7:     Generate  $\tilde{\mathbf{x}}$  with  $\mathbf{v}^0, \mathbf{x}_t$  using Eq. 4
8:     Calculate  $g(\mathbf{x}^*)$  with  $f(\tilde{\mathbf{x}})$  using Eq. 5
9:     if  $f(\tilde{\mathbf{x}}) = \tilde{y}$  and  $g(\mathbf{x}^*) < G_i$  then
10:       $G_i \leftarrow g(\mathbf{x}^*)$ 
11:       $V \leftarrow V \cup \{\mathbf{v}^0\}$ 
12:      break
13:    end if
14:    if  $c > 10$  then
15:       $\Delta h \leftarrow 1, \Delta w \leftarrow 1$ 
16:    end if
17:     $c \leftarrow c + 1$ 
18:  end while
19: end for
20: return  $V, G$ 

```

In order to increase the diversity of the generated population \mathbf{v}^r , crossover is introduced. The diversity of a population enables the exploration of the solution space for better individuals. Consequently, crossover operation is a crucial part of our method for further promoting population diversity, and every offspring after mutation can have the crossover. Unlike the traditional differential evolution algorithm [40], because the paired key-points of the modeling patches have an order relationship, it is impossible to directly perform crossover by element according to the probability. In practice, for any element in a candidate solution \mathbf{v}^r after mutation, we randomly add a noise $\kappa = \{-1, 0, 1\}^4$ to it respectively, to help jump out of the local optimal solution. Therefore, crossover can be expressed as:

$$\mathbf{v}^m = \mathbf{v}^r + \{-1, 0, 1\}^4. \quad (8)$$

The evolution algorithm assumes that superior individuals are selected from a population and inferior individuals are eliminated. According to this assumption, individuals with better fitness scores are more likely to survive over time. Specifically, if an offspring has a smaller fitness score, it will also be better on Eq. 2 and be a better adversarial example. Hence, if a new offspring has a smaller fitness score than the worst offspring in the population, the worst offspring will be discarded and the new offspring will be selected in its place.

Algorithm 2 summarizes the pipeline of DevoPatch. First, we obtain the initial population set and fitness scores through population initialization. Then, new offspring is generated by mutation and crossover to enhance diversity during each query. Next, the fitness score is calculated for the new population. Finally, the new population will be selected and updated

Algorithm 2 DevoPatch

Input: source image \mathbf{x} , ground-truth label y , targeted image \mathbf{x}_t , target label \tilde{y} , query budget N , population size p , initialization rate μ , mutation rate γ , model f

Output: adversarial example \mathbf{x}^*

```

1:  $V, G \leftarrow \text{PopulationInitialization}(f, \mathbf{x}, y, \mathbf{x}_t, \tilde{y}, \mu, p)$ 
2:  $k_{best} \leftarrow \arg \min_k(G), k_{worst} \leftarrow \arg \max_k(G)$ 
3: for  $i \leftarrow 1, 2, \dots, N$  do
4:   Random sample  $\mathbf{v}^j, \mathbf{v}^q$  from  $V \setminus \mathbf{v}^{k_{best}}$ 
5:   Initial  $\mathbf{v}^r$  with  $\mathbf{v}^j, \mathbf{v}^q, \mathbf{v}^{k_{best}}$  using Eq. 7
6:   Generate  $\mathbf{v}^m$  by random noises  $\kappa$  using Eq. 8
7:   Generate  $\tilde{\mathbf{x}}$  with  $\mathbf{v}^m, \mathbf{x}_t$  using Eq. 4
8:   Calculate  $g(\mathbf{x}^*)$  with  $f(\tilde{\mathbf{x}})$  using Eq. 5
9:   if  $g(\mathbf{x}^*) < G_{k_{worst}}$  then
10:      $\mathbf{v}^{k_{worst}} \leftarrow \mathbf{v}^m$ 
11:      $G_{k_{worst}} \leftarrow g(\mathbf{x}^*)$ 
12:   end if
13:    $k_{best} \leftarrow \arg \min_k(G), k_{worst} \leftarrow \arg \max_k(G)$ 
14: end for
15: Generate  $\mathbf{x}^*$  with  $\mathbf{v}^{k_{best}}, \mathbf{x}_t$  using Eq. 4
16: return  $\mathbf{x}^*$ 

```

according to the fitness score. Note that each time a new population is generated, we need perform boundary processing on the new population to ensure that $0 \leq i_1 < i_2 < H, 0 \leq j_1 < j_2 < W$.

IV. EXPERIMENTS

In this section, we show the experimental results to demonstrate the effectiveness of the proposed differential evolutionary patch attack. First, we choose appropriate hyperparameters for DevoPatch. Then we evaluate the adversarial robustness of several image classification models and face recognition models. We further conduct ablation studies and analyze the factors for the effectiveness of DevoPatch.

A. Experimental Settings

1) *Datasets:* To evaluate the effectiveness of our method, we conduct experiments on image classification and face verification. For image classification, we follow [19] and conduct experiments on a challenging dataset, ILSVRC2012 [41], which has 1,000 object categories in total. For the evaluation sets, we randomly draw 1,000 correctly classified images from ILSVRC2012 validation set. Target images are also randomly chosen correctly classified images corresponding to target classes from the ILSVRC2012 validation set. These selected images are evenly distributed among the 1,000 classes. For face verification, we select 400 pairs in dodging the attack, where each pair belongs to the same identity, and another 400 pairs in impersonation attack, where the images from the same pair are from different identities. The images are selected from LFW [42] and CelebA [43]. Target images are also randomly chosen correctly recognized images corresponding to identities from LFW and CelebA. All the selected images can be correctly recognized by the face recognition models.

2) *Models*: To evaluate the effectiveness of DevoPatch on different network architectures of image classification models, we select three different architecture models as threat models. For convolution-based models, we use a pre-trained ResNet-152 (ResNet) [44] for ILSVRC2012. For attention-based models, we select a pre-trained ViT-B-16/224 model (ViT-B) [26]. For multi-layer-based models, we select MLP-Mixer-B-16/224 model (Mixer-B) [27]. We also study three face recognition models, including FaceNet [45], CosFace [46] and ArcFace [47], which all achieve over 99% accuracies on the validation set. The threshold for the face recognition model is the one that achieves the highest accuracy on the validation set.

3) *Attack Methods*: Decision-based settings present more practical threats to deployed systems because it is hard to get the score in the system. To solve the practical issue of the score-based setting, our work first explores decision-based patch attacks, which can still construct a perturbation with the minimal information exposed – the top-1 predicted label. Therefore, to reveal the issues of the score-based setting, all experimental comparisons are performed in the decision-based setting. For a fair comparison with score-based patch attacks, we choose HPA [18], MPA [19], TPA [19], Adv-watermark (AdvW) [20] and Patch-RS [21] as the baseline under the decision-based setting. Inspired by [48], we leverage the label smoothing [49] to turn the hard-label into the score for the compared score-based methods without increasing the number of queries, where $\varepsilon = 0.1$. Following [21], the patch areas of TPA, AdvW, and Patch-RS are fixed. For white-box patch attacks, we choose GAP [8] as the baseline. GAP and DevoPatch share the same location mask M and query (inference) budgets.

4) *Evaluation Metrics*: Following [19] and [21], there are three metrics to measure the performance of black-box patch attacks. Patch area (%) is the number of perturbed pixels divided by the total number of pixels of an image. To control how noticeable a patch is, we define an **Average Patch Area (APA)** as the average area across all successful attacks. To evaluate the efficiency of the patch attack, we calculate the **Average Number of Queries (ANQ)** over the images finished with patch attacks, followed by [21]. Following decision-based adversarial attacks [22], [23], we select the number of queries that reach the minimum value of the patch area during the query process as the calculated value of ANQ. Finally, a measure used to evaluate the adversarial robustness of a model is **Attack Success Rate (ASR)**. ASR (%) is the ratio of adversarial examples that are successfully misrecognized.

B. Effects of Hyperparameters

Here, we analyze the key factors of DevoPatch, including population size p , initialization rate μ , and mutation rate γ and fitness measure. All ablation experiments are performed on ResNet-152 for the image classification task.

1) *Population Size*: Table I shows the effect of different population sizes on performance, where $\mu = 0.1$ and $\gamma = 1$. As the population size p gets larger, the ASR can still remain at 100%, while the APA will decrease further and the ANQ

TABLE I
ABALATION STUDY ON POPULATION SIZE p

Population Size p	Untargeted Attack			Targeted Attack		
	ASR	APA	ANQ	ASR	APA	ANQ
5	100.0	16.48	769.2	100.0	29.47	738.9
10	100.0	12.16	1327.2	100.0	25.45	1317.2
15	100.0	10.09	1918.6	100.0	23.89	1989.0
20	100.0	9.36	2619.7	100.0	22.81	2678.4
30	100.0	8.08	4163.6	100.0	22.29	4125.1

TABLE II
ABALATION STUDY ON INITIALIZATION RATE μ

Initialization Rate μ	Untargeted Attack			Targeted Attack		
	ASR	APA	ANQ	ASR	APA	ANQ
0.05	100.0	12.69	1324.3	100.0	26.55	1322.5
0.10	100.0	12.16	1327.2	100.0	25.45	1317.2
0.15	100.0	11.64	1306.0	100.0	24.69	1287.7
0.20	100.0	10.96	1320.5	100.0	24.63	1290.7
0.25	100.0	10.19	1355.5	100.0	24.35	1263.1
0.30	100.0	10.25	1391.1	100.0	23.94	1239.1
0.35	100.0	10.00	1349.7	100.0	23.78	1261.6
0.40	100.0	9.99	1349.8	100.0	23.91	1260.4

TABLE III
ABALATION STUDY ON MUTATION RATE γ

Mutation Rate γ	Untargeted Attack			Targeted Attack		
	ASR	APA	ANQ	ASR	APA	ANQ
1	100.0	10.00	1349.7	100.0	23.78	1261.6
2	100.0	7.75	6487.8	100.0	21.50	6205.2
3	100.0	7.32	7294.9	100.0	21.24	7062.1
4	100.0	6.91	7082.6	100.0	21.22	6909.2

will get greater. In particular, when $p = 30$, its APA is 4.08% less than $p = 10$, but the ANQ is about 2 times larger. For the sake of query efficiency, we choose $p = 10$.

2) *Initialization Rate*: Table II shows the effect of different initialization rates on performance, where $p = 10$ and $\gamma = 1$. As the initialization rate μ gradually increases, the APA will become less, and the ANQ will not change much. Due to the trade-off between areas and queries, we choose $\mu = 0.35$.

3) *Mutation Rate*: Table III shows the effect of different mutation rates on performance, where $\mu = 0.35$ and $p = 10$. Obviously, a larger mutation rate can indeed achieve a smaller adversarial patch, but it greatly increases the ANQ. Specifically, when $\gamma = 4$, the APA is 3.09% less than when $\gamma = 1$, but the ANQ is 4 times greater. Considering query efficiency and average area, we choose $\gamma = 1$.

4) *Convergence*: Further, we analyze the convergence of DevoPatch. Fig. 3 describes the variation curve of area with query budget under different initialization rates μ . Intuitively, our method converges quickly. When the number of queries is about 1,000, the best adversarial example has been solved.

5) *Fitness Measure*: The fitness measure directly affects the efficiency of the solution. In DevoPatch, we choose l_0 norm for the fitness calculation according to the optimization objective in Eq. 2. However, other norms can also be used to calculate Eq. 5. Table XI shows the ablation study about different norms on fitness calculation. l_0 norm consistently outperforms other

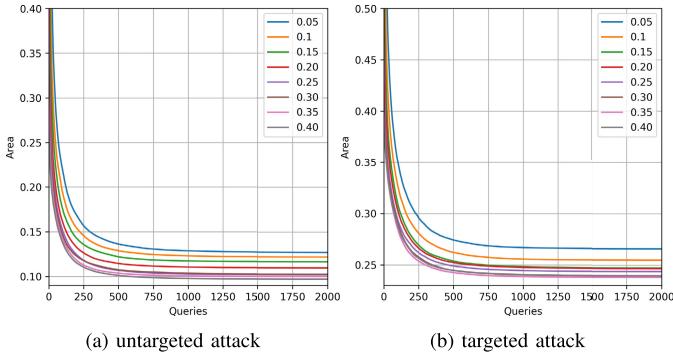


Fig. 3. Convergence analysis. DevoPatch is query-efficient and can already generate high-quality adversarial examples when the ANQ is around 1,000.

TABLE IV
ABALATION STUDY ON FITNESS MEASURE

Model	Norm	Untargeted Attack			Targeted Attack		
		ASR	APA	ANQ	ASR	APA	ANQ
ResNet	0	100.0	10.00	1349.7	100.0	23.78	1261.6
	1	100.0	10.52	1395.3	100.0	24.64	1287.9
	2	100.0	11.21	1394.7	100.0	25.77	1266.9
ViT-B	0	100.0	13.84	1314.0	100.0	25.99	1256.6
	1	100.0	14.79	1331.1	100.0	26.85	1285.4
	2	100.0	15.40	1333.5	100.0	28.07	1290.5
Mixer-B	0	100.0	14.76	1336.9	100.0	25.54	1274.6
	1	100.0	15.29	1360.0	100.0	26.32	1286.0
	2	100.0	16.39	1340.0	100.0	27.42	1282.1

norms in terms of queries and areas. The main reason may be that the optimization objectives of l_0 norm and Eq. 2 are consistent. Since other norms calculate the distance similarity with the image, they tend to place the patch in the place where the source image and the targeted image are similar, which will fall into the local optimal solution.

C. Attacks on Image Classification

In this section, we compare the attack performance of various black-box patch attacks on image classification. We set the query budgets for untargeted and targeted attack to 10,000 and 50,000, respectively. The hyperparameters of DevoPatch are: $p = 10$, $\mu = 0.35$, $\gamma = 1$. For targeted attacks, we consider a randomly chosen correctly classified image corresponding to the target class \tilde{y} from the dataset. For untageted attacks, we use a randomly chosen correctly classified image corresponding to the random class except the ground-truth label from the dataset, followed by [19]. DevoPatch takes about 6.33 hours to perform 10,000 queries on 1,000 images on ResNet-152, based on an NVIDIA Tesla V100. The experimental results against decision-based patch attacks in untargeted and targeted setting on ILSVRC2012 are summarized in Table V. The experimental results show that our DevoPatch consistently outperforms HPA, MPA, TPA, AdvW and Patch-RS in terms of queries and patch areas with a higher ASR, which shows the effectiveness of DevoPatch.

In the untargeted setting, TPA, AdvW, and Patch-RS achieve the trade-off on ASR and ANQ. Although the label returns little information, TPA achieves higher ASR in the

TABLE V
DECISION-BASED BLACK-BOX PATCH ATTACKS ON ILSVRC2012

Model	Method	Untargeted Attack			Targeted Attack		
		ASR	APA	ANQ	ASR	APA	ANQ
ResNet	HPA [18]	98.8	28.48	10000.0	0	-	50000.0
	MPA [19]	99.5	14.10	10000.0	1.8	35.87	50000.0
	TPA [19]	84.8	10.05	2768.0	82.0	24.12	12614.0
	AdvW [20]	44.7	10.05	5913.0	-	-	-
	Patch-RS [21]	55.4	10.05	4754.4	0.1	24.12	49950.3
	Ours	100.0	10.00	1349.7	100.0	23.78	1261.6
ViT-B	HPA [18]	98.1	34.04	10000.0	0	-	50000.0
	MPA [19]	98.2	14.32	10000.0	2.1	66.38	49750.0
	TPA [19]	82.3	14.06	3108.8	71.8	26.36	23794.6
	AdvW [20]	30.6	14.06	7288.1	-	-	-
	Patch-RS [21]	42.1	14.06	6151.8	0.2	26.36	49940.2
	Ours	100.0	13.84	1314.0	100.0	25.99	1256.6
Mixer-B	HPA [18]	87.6	39.53	10000.0	0	-	50000.0
	MPA [19]	98.5	18.79	9850.0	2.2	80.58	49500.0
	TPA [19]	96.1	15.08	1710.4	91.0	25.90	8037.0
	AdvW [20]	40.7	15.08	6206.1	-	-	-
	Patch-RS [21]	63.5	15.08	3985.4	0.5	25.90	49801.2
	Ours	100.0	14.76	1336.9	100.0	25.54	1274.6

decision-based setting due to its strong texture prior. Although HPA has a high ASR, its ANQ and APA are extremely large. MPA achieves sub-optimal ASR, but it is inefficient and always uses the whole query budget since it takes 10,000 queries and chooses the best one. DevoPatch achieves 100% ASR with one-seventh of MPA on ANQ under a smaller average area. Under the more challenging targeted setting, due to the lack of effective information about the target class, HPA, MPA, AdvW, and Patch-RS are almost useless. Because TPA has a texture prior, it still has a high ASR, but the ANQ is extremely high. Because of our simplification of solution space, our DevoPatch outperforms TPA by 18.0%, 28.2%, and 9.0% ASR on ResNet, ViT, and MLP, respectively, while the average queries are only one-tenth, one-twentieth and one-seventh of TPA. Also, we choose to compare with GAP, the most basic white-box patch attack. We expect DevoPatch to reach the lower bound of white-box patch attacks in attack performance. Table VI illustrates that DevoPatch achieves ASR equivalent to GAP. Both black-box and white-box experiments show that DevoPatch is a query-efficient decision-based black-box patch attack with high attack performance against different network architectures on image classification.

We provide the visualization of patch attacks on different network architectures as shown in Fig. 4. The labels below the image indicate the predicted classes. Labels in black, red, and blue represent the ground truth, target classes, and the classes after the targeted attack has failed, respectively. HPA and MPA use gray or colored patches to achieve attacks (the second and third rows), but their patch area is large and it is difficult to achieve targeted attacks. TPA uses the ImageNet pre-trained texture dictionary for patch attack (shown in the fourth row) and has a higher ASR in the decision-based setting, but its area is also larger. AdvW selects pre-defined logos for patch attacks, but it is difficult to implement targeted attacks because logos have little category information (shown in the fifth row). Patch-Rs is based on a random search framework (shown in the sixth row), but it is difficult to implement targeted attacks because the top-1 labels have too little information. Our DevoPatch achieves the query-efficient



Fig. 4. Visualization of patch attacks in the targeted setting on different network architectures. The labels below the image represent the predicted classes. Black, red and blue labels represent ground-truth labels, target classes, and the classes after the targeted attack has failed, respectively. DevoPatch successfully achieves the targeted attack of all examples with a small patch area.

TABLE VI
COMPARISONS WITH WHITE-BOX PATCH ATTACKS ON ASR (%)

Source Model	Method	Untargeted Attack			Targeted Attack		
		ResNet	ViT-B	Mixer-B	ResNet	ViT-B	Mixer-B
ResNet	GAP [8] Ours	100 100	5.3 7.5	9.3 10.3	100 100	0 10.4	0.2 14.6
ViT-B	GAP [8] Ours	19.6 24.7	100 100	15.5 15.8	0 18.2	100 100	0 18.2
Mixer-B	GAP [8] Ours	21.6 30.5	11.0 15.7	100 100	0 20.3	0 15.5	100 100

attack in a decision-based setting with a smaller area and higher ASR.

D. Attacks on Face Verification

In this section, we compare the attack performance of various black-box patch attacks on face verification. We set the

query budgets for dodging and impersonation attacks to 10,000 and 50,000, respectively. The hyperparameters of DevoPatch are: $p = 10$, $\mu = 0.35$, $\gamma = 1$. For dodging attacks, for randomly selecting a pair of faces with the same identity, the adversary generates an adversarial face to make the model recognize them as different identities. For impersonation attacks, the adversary generates an adversarial face to make the model recognize them as the same identity, which originally belonged to different identities. Here, we use cosine similarity and threshold to determine whether it is the same identity. When the cosine similarity of a pair of faces is greater than the threshold, the faces belong to the same identity.

Table VII shows decision-based patch attacks on face verification. Note that TPA exploits ILSVRC2012 on image classification to implement the attack through a class texture dictionary generated by style transfer. Here, because the face can directly represent the identity, we choose targeted images

TABLE VII
DECISON-BASED BLACK-BOX PATCH ATTACKS ON FACE VERIFICATION

Dataset		LFW						CelebA					
Model	Method	Dodging Attack			Impersonation Attack			Dodging Attack			Impersonation Attack		
		ASR	APA	ANQ	ASR	APA	ANQ	ASR	APA	ANQ	ASR	APA	ANQ
ArcFace	HPA [18]	100	25.81	10000.0	4.25	22.47	50000.0	100	10.78	10000.0	32.75	15.28	50000.0
	MPA [19]	100	14.29	10000.0	5.00	17.24	50000.0	100	4.80	10000.0	38.00	8.40	50000.0
	TPA [19]	80.75	11.51	3530.0	52.5	12.76	27352.5	86.50	4.98	2754.0	58.50	7.66	25030.0
	AdvW [20]	5.75	11.51	9508.3	-	-	-	5.00	4.98	9549.8	-	-	-
	Patch-Rs [21]	24.00	11.51	7837.6	1.00	12.76	49513.9	56.00	4.98	4838.3	16.00	7.66	42597.1
	Ours	100	11.13	1016.4	100	12.28	960.0	100	4.71	1043.4	100	7.48	992.9
CosFace	HPA [18]	100	14.14	10000.0	10.75	15.74	50000.0	100	3.27	10000.0	43.75	4.73	50000.0
	MPA [19]	100	8.36	10000.0	9.00	14.57	50000.0	100	2.71	10000.0	59.50	4.82	50000.0
	TPA [19]	76.5	7.29	3741.0	56.6	12.05	25827.5	88.50	2.69	2428.0	67.00	4.92	20177.5
	AdvW [20]	8.00	7.29	9303.0	-	-	-	13.00	2.69	8742.7	-	-	-
	Patch-Rs [21]	55.25	7.29	5024.3	5.5	12.05	47744.4	80.00	2.69	2504.1	38.75	4.92	31729.5
	Ours	100	7.02	1040.7	100	11.40	1025.2	100	2.56	1041.7	100	4.63	1080.2
FaceNet	HPA [18]	100	15.13	10000.0	16.00	18.05	50000.0	100	8.68	10000.0	23.75	17.27	50000.0
	MPA [19]	100	10.46	10000.0	18.50	18.97	50000.0	100	5.50	10000.0	31.50	13.37	50000.0
	TPA [19]	87.50	6.56	2748.0	76.00	12.69	16027.5	78.25	3.52	3889.0	71.50	7.22	19640.0
	AdvW [20]	18.75	6.56	8596.8	-	-	-	17.00	3.52	8608.4	-	-	-
	Patch-Rs [21]	32.25	6.56	7257.0	6.00	12.69	47388.6	38.25	3.52	6659.0	8.25	7.22	46345.8
	Ours	100	6.56	1055.2	100	12.38	1035.0	100	3.21	1129.3	100	7.17	1058.3

as the texture dictionary of TPA. These targeted images are the same as DevoPatch. AdvW and Patch-Rs achieve very few ASR in the dodging attack. Although HPA and MPA have extremely high ASR in the dodging attack, they tend to cover the face with a larger area and complete the attack with larger APA and extremely low query efficiency. Further, HPA, MPA, and Patch-Rs have very little ASR in the more challenging impersonation attack due to the limited information of the output of the label by the model. Because TPA has the targeted image as a prior, it achieves a good trade-off in ASR and ANQ in the dodging and impersonation attack. But even so, the performance of TPA in the impersonation attack can not achieve an extremely high ASR. Our DevoPatch and TPA share the same prior information (targeted images) in face verification, but DevoPatch significantly outperforms TPA in attack performance and query efficiency. In a dodging attack, the ANQ of TPA is usually three times that of DevoPatch. In the harder impersonation attack, the ANQ of DevoPatch is about one-twentieth of TPA. More importantly, in such a limited number of queries, DevoPatch has a smaller patch area and ASR, which is enough to illustrate the effectiveness of the proposed differential evolution patch attack algorithm.

Table V shows the visualization of different patch attacks on face verification. Here, we choose ArcFace as the base model and visualize it on the LFW dataset. The color of the face frame represents whether the attack is successful. Blue represents a failed attack and red represents a successful attack. Because of the different semantic categories in image classification, the generated patches are easy to perceive. In face verification, since the color of patches is irrelevant to semantics, a similar situation also occurs in HPA, MPA, AdvW, and Patch-Rs. However, TPA and DevoPatch select faces as a prior and have face-related features, the resulting patches are relatively imperceptible. Further, since DevoPatch can better determine the location and shape of patches, it can

TABLE VIII
ATTACKS ON THE EMPIRICAL AND CERTIFIABLE PATCH DEFENSES

Type	Model	Method	Untargeted Attack			Targeted Attack		
			ASR	APA	ANQ	ASR	APA	ANQ
Empirical	ResNet-152	Clean Only Attack	0	-	-	0	-	-
		Only DW Attack DW	100	10.00	1349.7	100	23.78	1261.6
		Only LGS Attack LGS	3.1	-	-	3.1	-	-
Certifiable	ResNet-50	Only DS Attack DS	10.0	-	-	10.0	-	-
	ECViT-B	Only ECViT Attack ECViT	2.9	-	-	2.9	-	-

improve attack performance and imperceptibility. DevoPatch has strong applicability and achieves query-efficient decision-based patch attacks on both image classification and face verification.

E. Attacks on Patch Defenses

We also evaluate the performance of DevoPatch against the patch defense methods on image classification, including Local Gradient Smoothing (LGS) [50], Digital Watermarking (DW) [51], Derandomized Smoothing (DS) [52] and Efficient Certifiable Vision Transformer (ECViT) [53]. For empirical defenses, DW and LGS are regarded as pre-processing operations to remove adversarial patches. For certifiable defenses, we attack models including certifiable mechanisms. The backbone of DS is ResNet-50 and the backbone of ECViT is ECViT-B. Table VIII shows the adversarial robustness against empirical and certifiable patch defenses on DevoPatch. The above defenses cause very few images to be misclassified. For empirical patch defenses, DW and LGS do not take effect in the face of DevoPatch. A possible reason is that the adversarial patches produced by DevoPatch are part of

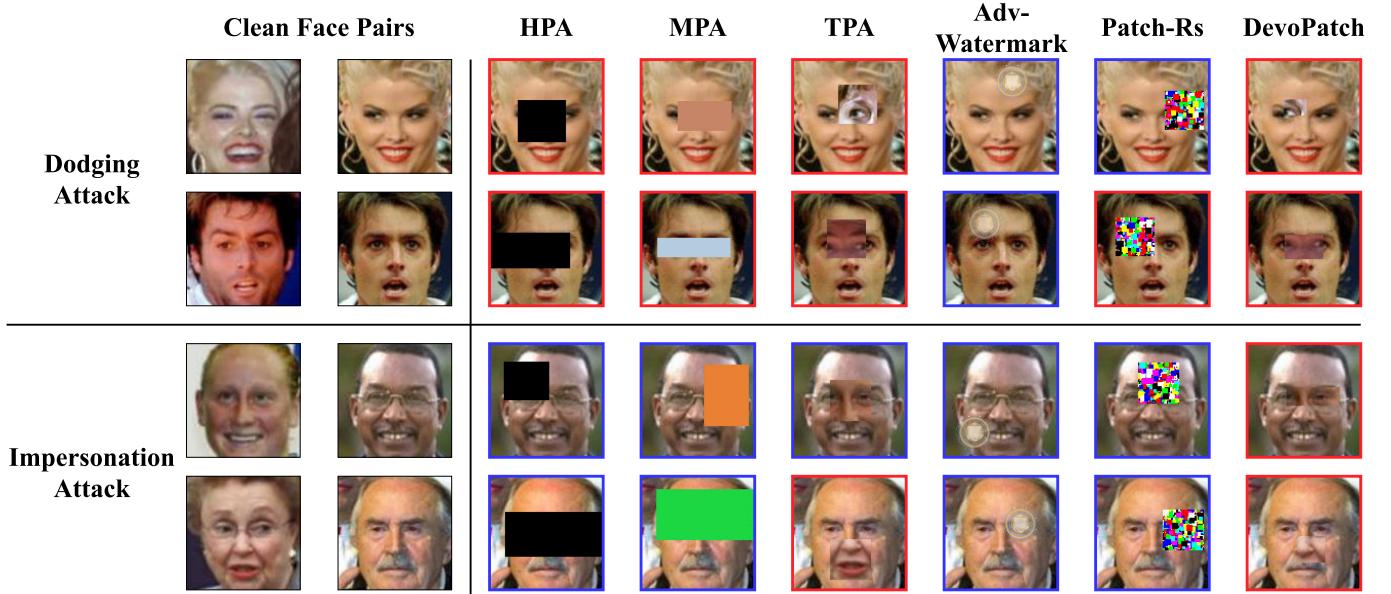


Fig. 5. Visualization of patch attacks with ArcFace on face verification. The color of the face frame represents whether the attack is successful. Blue represents a failed attack, and red represents a successful attack. DevoPatch successfully achieves the attack of all examples with a small patch area.

natural images rather than adversarial perturbations generated by gradients. The former has semantics and harmony in visual understanding. For certifiable patch defenses, ECViT is the state-of-the-art certifiable patch defense, but it also cannot maintain certification in large rectangular patch areas (greater than 10%). However, ECViT increases APA and reduces the quality of patches compared to DS. This experiment exposes deficiencies in existing patch defenses, so it is critical to improve the robustness and certification of defenses.

F. Ablation Study on Differential Evolution

Both genetic algorithm (GA) [54] and differential evolution algorithm (DE) [40] are evolutionary algorithms, which simulate mutation, crossover, and selection in genetics to solve optimization problems. Due to different encoding, crossover, mutation, and selection strategies, DE generally has faster convergence speed [40]. However, directly applying DE to this task encounters the challenges of complex solution space and efficient query efficiency. Therefore, we simplify solution spaces to the integer domain and improve the traditional DE.

We conduct ablation studies on differential evolution with ResNet-152, and the parameter settings are consistent with Section IV-C. Fig. 6 shows how APA changes as the number of queries increases under different differential evolutions. Here, *w.o. crossover* and *w.o. mutation* mean that the crossover and mutation improved by DevoPatch are not used, but the crossover and mutation of traditional DE [40] are used. Under the premise of guaranteeing 100% ASR, the mutation and crossover of DevoPatch have a fast convergence speed, and can better jump out of the local optimal solution, thereby generating higher-quality adversarial patches.

G. Analysis on Target Images

To explore the impact of the selection of target images on attack performance, we conduct experimental analysis on

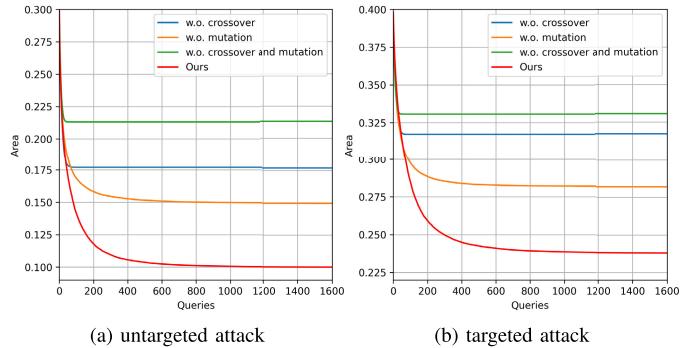


Fig. 6. Ablation study on differential evolution. DevoPatch can better jump out of the local optimal solution and generate higher-quality adversarial patches.

TABLE IX
ANALYSIS ON DIFFERENT COLORS OF TARGET IMAGES

Color	Untargeted Attack			Targeted Attack		
	ASR	APA	ANQ	ASR	APA	ANQ
White	100.0	15.26	1403.8	0.07	0.24	119.6
Blue	100.0	13.98	1333.5	0.10	0.33	122.6
Green	100.0	13.68	1296.6	0.12	0.44	123.2
Yellow	99.9	13.54	1351.3	0.07	0.25	120.6
Pink	100.0	15.06	1380.4	0.09	0.30	117.9
Ours	100.0	10.00	1349.7	100.0	23.78	1261.6

image classification with ResNet-152 from three perspectives, including color, randomness, and data source. The parameter settings are consistent with Section IV-C.

1) *Color*: HPA [18] and MPA [19] introduce monochrome patches to implement the attack. Therefore, monochrome images have the potential to become target images. Here, we select *White*, *Blue*, *Green*, *Yellow* and *Pink* as target images. Table IX illustrates the attack performance and query

TABLE X
ANALYSIS ON RANDOMNESS OF TARGET IMAGES

Random	Untargeted Attack			Targeted Attack		
	ASR	APA	ANQ	ASR	APA	ANQ
(1)	100.0	10.00	1349.7	100.0	23.78	1261.6
(2)	100.0	9.85	1348.7	100.0	22.78	1238.0
(3)	100.0	9.86	1357.4	100.0	22.19	1227.8
(4)	100.0	9.81	1350.2	100.0	22.52	1223.1
(5)	100.0	9.83	1331.9	100.0	22.57	1220.6
Mean	100.0	9.86±0.07	1347.6±9.4	100.0	22.77±0.60	1234.2±16.6

TABLE XI
ANALYSIS ON DIFFERENT SOURCES OF TARGET IMAGES

Model	Data Sources	Untargeted Attack			Targeted Attack		
		ASR	APA	ANQ	ASR	APA	ANQ
ResNet	Same	100	10.77	1262.9	100	24.10	1256.9
	Different	100	10.49	1374.2	100	25.55	1210.1
ViT-B	Same	100	15.79	1359.3	100	29.80	1257.3
	Different	100	14.83	1332.2	100	26.34	1236.6
Mixer-B	Same	100	15.01	1240.9	100	25.73	1229.6
	Different	100	15.91	1302.4	100	25.43	1298.6

efficiency when images of different colors are used as target images. Under the untargeted setting, DevoPatch with monochrome images has a similar ASR and APA as MPA, but the query efficiency is one-seventh of MPA. However, monochrome images have almost no target attack performance, because monochrome images have almost no semantic information of the corresponding class. The above experiment shows the efficiency of DevoPatch and the necessity of random natural images as target images.

2) *Randomness*: Considering that target images on image classification are randomly selected from the ILSVRC2012 validation set, randomness may affect attack performance and query efficiency. Here, we fix the clean images and randomly sample the target images five times, then evaluate the performance. Table X illustrates the impact of different random target images on attack performance and query efficiency. From the experiments, we can find that although the marginal improvement can be obtained through randomness, the impact of random target images on performance is very small, and APA and ANQ are very close. Although the optimal performance is randomly selected multiple times, the query cost is multiplied, which is not feasible in real-world scenarios. From the perspective of the target images themselves, how to generate a more powerful target image is a future work that has the potential to improve the attack efficiency.

3) *Data Source*: This work is carried out in a black-box decision-based setting. For the black-box model, we can only obtain the output label, which is difficult to obtain full training data or access the model architecture. As described in Section III-C, we use targeted image priors to reduce the complexity of the solution space and achieve effective targeted attacks. In Section IV-C, we choose the targeted images randomly sampled from the validation set of ILSVRC2012 and show the effectiveness of DevoPatch, which belong to the same source data as the training set of the models. To further demonstrate the generalization capability of the proposed method in

real-world scenarios, we collect 100 images from the Internet as the targeted images, which are not from the same source as ILSVRC2012. In the case of using different-source data, ASR equivalent to same-source data can be obtained on image classification, as described in Table XI. DevoPatch based on different-source data has very subtle differences in areas and queries, which shows DevoPatch is not sensitive to the domain of targeted images. It is worth noting that TPA uses ImageNet to generate a texture dictionary to attack the classification model, which is impossible in real-world scenarios. However, DevoPatch can arbitrarily select a correctly identified image from the Internet as the targeted image, thereby realizing a black-box patch attack with high operability and flexibility.

H. Effectiveness Analysis

In this section, we analyze why DevoPatch has a very high targeted attack success rate. We utilize the gradient-based class activation mapping (Grad-CAM) [55] to visualize the attention maps of various classes, as shown in Figure 7. First, we use Grad-CAM to generate the attention maps of the source image and targeted image of their corresponding classes (such as column 2 and column 4). We can find that the most discriminative regions are all in the regions with the most salient category objects. Then, we visualize the attention map of the source image corresponding to the target class and find that it is not focused on the objects of the source class. Among the limited queries, we notice that the class activation map of the class predicted by the model focuses on the adversarial patch, indicating that DevoPatch can become the most discriminative region without having access to any details of the model. Since adversarial patches based on targeted classes cover the most discriminative regions, the model outputs predictions for the targeted class. Therefore, DevoPatch is a query-efficient decision-based patch attack because of paired key-points and targeted image prior.

V. DISCUSSION

In this section, we compare the robustness of ResNet, ViT, and MLP models to patch perturbation on image classification. In Table V, we find that our method needs a relatively larger area to craft successful patch attacks on ViT and MLP models compared with ResNet model with similar query budgets. It means ViT and MLP models are relatively more robust than ResNet model under the most threatening decision-based patch attack. Probably because ViT and MLP split the image into multiple non-overlapping patches which reduce the impact of noises on one local region to the final classification results [28]. As shown in Table VI, all kinds of models are equally vulnerable to perturbations computed using white-box attack GAP. We then find that adversarial perturbations computed using ResNet rarely transfer to ViT or MLP in the white-box setting especially for the targeted attack, which is also observed by [28] in imperceptible attacks. Interestingly, different from the conclusion in [28], we find that adversarial perturbations computed using ViT and MLP do transfer to ResNet. Particularly, the adversarial perturbations crafted by our black-box method transfer more easily over

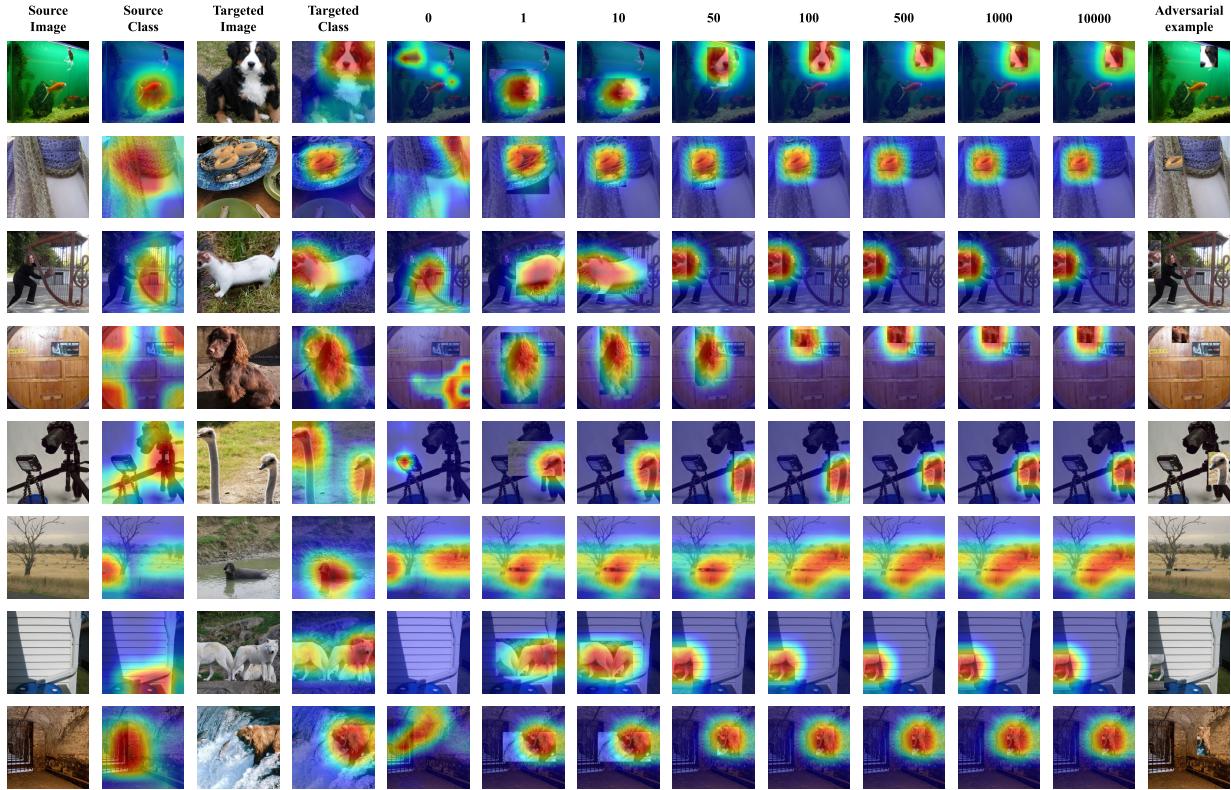


Fig. 7. Demonstration of the different discriminative regions of ResNet-152 on image classification. We utilize the gradient-based class activation mapping [55] to visualize the attention maps of various classes. Among the limited queries, we notice that the class activation map of the class predicted by the model focuses on the adversarial patch, indicating that DevoPatch can become the most discriminative region without having access to any details of the model. Since adversarial patches based on targeted classes cover the most discriminative regions, the model outputs predictions for the targeted class.

different architectures, even for the targeted attack. In addition, we first present the lower bound on the area required for the targeted patch attack in the decision-based setting. Targeted attacks of any category can be completed in about 25% of the patch area under about 1,300 queries. This is an important safety reference for real-world systems. The above observations suggest that studying the adversarial robustness of DNNs from the perspective of decision-based black-box patch attacks is necessary to better understand and improve DNNs.

VI. CONCLUSION

In this work, we explore the practical threat of decision-based black-box patch attack to the robustness of existing DNNs for the first time. Compared with transfer-based and score-based settings, decision-based settings do not require access to a large amount of training data and only rely on minimal information, the labels by the model's output, to achieve the adversarial attack. To simplify the solution space and improve query efficiency, we propose a differential evolutionary algorithm named DevoPatch for query-efficient adversarial patch attacks in the decision-based black-box setting. In DevoPatch, we model adversarial patches as paired key-points and utilize targeted images as priors. With paired key-points and targeted image priors, the differential evolution algorithm based on the integer domain greatly improves the query efficiency. As a result of our comprehensive results, DevoPatch outperforms the state-of-the-art black-box patch attack in terms of patch area and ASR both on image classification and face verification. More importantly, with a reduced solution

space, DevoPatch illustrates significant query-efficiency when compared with the existing patch attacks in the decision-based black-box setting. We also investigate the robustness of various DNN architectures against DevoPatch.

DevoPatch exposes the shortcomings of existing DNNs against patch attacks. In future research, we can use DevoPatch to evaluate the robustness of the model to black-box adversarial patches. In addition, our work provides a deep understanding of the robustness of DNNs against decision-based patch attacks. We believe this work could be used to inform and inspire the design and deployment of robust vision models based on various DNN architectures in the future.

REFERENCES

- [1] C. Szegedy et al., “Intriguing properties of neural networks,” in *Proc. Int. Conf. Learn. Represent.*, Y. Bengio and Y. LeCun, Eds., 2014, pp. 1–10.
- [2] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–11.
- [3] A. Kurakin, I. J. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” in *Proc. Int. Conf. Learn. Represent. Worksh.*, 2017, pp. 1–14.
- [4] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–28.
- [5] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.
- [6] H. Zhang, Y. Avrithis, T. Furon, and L. Amsaleg, “Walking on the edge: Fast, low-distortion adversarial examples,” *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 701–713, 2021.
- [7] F. Croce et al., “Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks,” in *Proc. ICML*, vol. 119, Jul. 2020, pp. 2206–2216.

- [8] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, “Adversarial patch,” in *Proc. Adv. Neural Inform. Process. Syst. (Worksh.)*, 2017, pp. 1–6.
- [9] D. Karmon, D. Zoran, and Y. Goldberg, “LaVAN: Localized and visible adversarial noise,” in *Proc. Int. Conf. Mach. Learn.*, vol. 80, J. G. Dy and A. Krause, Eds., 2018, pp. 2512–2520.
- [10] T. Wu, L. Tong, and Y. Vorobeychik, “Defending against physically realizable attacks on image classification,” in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–26.
- [11] S. Rao, D. Stutz, and B. Schiele, “Adversarial training against location-optimized adversarial patches,” in *Proc. Eur. Conf. Comput. Vis. Worksh.*, vol. 12539, A. Bartoli and A. Fusiello, Eds., 2020, pp. 429–448.
- [12] Z. Chen, B. Li, S. Wu, J. Xu, S. Ding, and W. Zhang, “Shape matters: Deformable patch attack,” in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2022, pp. 529–548.
- [13] H. Hosseini, B. Xiao, and R. Poovendran, “Google’s cloud vision API is not robust to noise,” in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 101–105.
- [14] C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, “DARTS: Deceiving autonomous cars with toxic signs,” 2018, *arXiv:1802.06430*.
- [15] Z. Kong, J. Guo, A. Li, and C. Liu, “PhysGAN: Generating physical-world-resilient adversarial examples for autonomous driving,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14242–14251.
- [16] K. Eykholt et al., “Robust physical-world attacks on deep learning visual classification,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1625–1634.
- [17] Z. Xiao et al., “Improving transferability of adversarial patches on face recognition with generative models,” in *IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2021, pp. 11845–11854.
- [18] A. Fawzi and P. Frossard, “Measuring the effect of nuisance variables on classifiers,” in *Proc. Brit. Mach. Vis. Conf.*, 2016, p. 137.
- [19] C. Yang, A. Kortylewski, C. Xie, Y. Cao, and A. Yuille, “Patchattack: A black-box texture-based attack with reinforcement learning,” in *Proc. Eur. Conf. Comput. Vis.*, in Lecture Notes in Computer Science, vol. 12371, Cham, Switzerland: Springer, 2020, pp. 681–698.
- [20] X. Jia, X. Wei, X. Cao, and X. Han, “Adv-watermark: A novel watermark perturbation for adversarial examples,” in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 1579–1587.
- [21] F. Croce, M. Andriushchenko, N. D. Singh, N. Flammarion, and M. Hein, “Sparse-RS: A versatile framework for query-efficient sparse black-box adversarial attacks,” in *Proc. AAAI*, 2022, pp. 1–9.
- [22] J. Chen, M. I. Jordan, and M. J. Wainwright, “HopSkipJumpAttack: A query-efficient decision-based attack,” in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 1277–1294.
- [23] H. Li, X. Xu, X. Zhang, S. Yang, and B. Li, “QEBA: Query-efficient boundary-based blackbox attack,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1218–1227.
- [24] Y. Zhong and W. Deng, “Towards transferable adversarial attack against deep face recognition,” *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1452–1466, 2021.
- [25] F. Tramér, A. Kurakin, N. Papernot, I. J. Goodfellow, D. Boneh, and P. D. McDaniel, “Ensemble adversarial training: Attacks and defenses,” in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–22.
- [26] A. Dosovitskiy et al., “An image is worth 16×16 words: Transformers for image recognition at scale,” in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–21.
- [27] I. O. Tolstikhin et al., “MLP-mixer: An all-MLP architecture for vision,” in *Proc. Adv. Neural Inform. Process. Syst.*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 24261–24272.
- [28] S. Bhujanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit, “Understanding robustness of transformers for image classification,” 2021, *arXiv:2103.14586*.
- [29] R. Shao, Z. Shi, J. Yi, P. Chen, and C. Hsieh, “On the adversarial robustness of visual transformers,” 2021, *arXiv:2103.15670*.
- [30] H. Huang, Y. Wang, Z. Chen, Z. Tang, W. Zhang, and K.-K. Ma, “Rptattack: Refined patch attack on general object detectors,” in *Proc. IEEE Int. Conf. Multimedia Expo. (ICME)*, Jul. 2021, pp. 1–6.
- [31] Z. Wu, S.-N. Lim, L. S. Davis, and T. Goldstein, “Making an invisibility cloak: Real world adversarial attacks on object detectors,” in *Proc. Eur. Conf. Comput. Vis.*, in Lecture Notes in Computer Science, vol. 12349, Cham, Switzerland: Springer, 2020, pp. 1–17.
- [32] A. Zolfi, M. Kravchik, Y. Elovici, and A. Shabtai, “The translucent patch: A physical and universal attack on object detectors,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15227–15236.
- [33] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition,” in *Proc. ACM Conf. Comput. Commun. Secur.*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds., 2016, pp. 1528–1540.
- [34] S. Komkov and A. Petrushko, “AdvHat: Real-world adversarial attack on ArcFace face ID system,” in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 819–826.
- [35] B. Yin et al., “Adv-makeup: A new imperceptible and transferable attack on face recognition,” in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 1252–1258.
- [36] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2414–2423.
- [37] J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks,” *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.
- [38] Y. Dong et al., “Efficient decision-based black-box adversarial attacks on face recognition,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7706–7714.
- [39] V. Q. Vo, E. Abbasnejad, and D. C. Ranasinghe, “Query efficient decision based sparse attacks against black-box deep learning models,” in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–11.
- [40] R. Storn and K. Price, “Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces,” *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
- [41] O. Russakovsky et al., “ImageNet large scale visual recognition challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
- [42] G. B. Huang, M. A. Mattar, H. Lee, and E. G. Learned-Miller, “Learning to align from scratch,” in *Proc. Adv. Neural Inform. Process. Syst.*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 773–781.
- [43] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3730–3738.
- [44] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
- [45] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 815–823.
- [46] H. Wang et al., “CosFace: Large margin cosine loss for deep face recognition,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5265–5274.
- [47] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “ArcFace: Additive angular margin loss for deep face recognition,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4685–4694.
- [48] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, “Black-box adversarial attacks with limited queries and information,” in *Proc. Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 2142–2151.
- [49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
- [50] M. Naseer, S. Khan, and F. Porikli, “Local gradients smoothing: Defense against localized adversarial attacks,” in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1300–1307.
- [51] J. Hayes, “On visible adversarial perturbations & digital watermarking,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1678–16787.
- [52] A. Levine and S. Feizi, “(De)randomized smoothing for certifiable defense against patch attacks,” in *Proc. Adv. Neural Inform. Process. Syst.*, vol. 33, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020, pp. 6465–6475.
- [53] Z. Chen, B. Li, J. Xu, S. Wu, S. Ding, and W. Zhang, “Towards practical certifiable patch defense with vision transformer,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 15127–15137.
- [54] D. Goldberg, *Genetic Algorithm in Search. (Optimization & Machine Learning)*. Boston, MA, USA: Addison Wesley, 1989.
- [55] K. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual explanations from deep networks via gradient-based localization,” *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 336–359, Feb. 2020.



Zhaoyu Chen (Graduate Student Member, IEEE) received the B.Eng. degree from Shandong University in 2020. He is currently pursuing the Ph.D. degree with the Academy for Engineering and Technology, Fudan University, Shanghai. His research interests include artificial intelligence security, computer vision, and their applications, such as adversarial examples and semantic segmentation.



Shouhong Ding received the Ph.D. degree from the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2016. He is currently a Senior Researcher with the YouTu Laboratory, Tencent, China. His current research interests include image/video processing, computer vision, pattern recognition, and multimedia technology.



Bo Li received the B.S. and Ph.D. degrees from the Department of Computer Science, Nanjing University, China, in 2014 and 2019, respectively. From 2020 to 2023, he was a Senior Researcher with the YouTu Laboratory, Tencent, China. His research interests include computer vision, pattern recognition, and artificial intelligence.



Shuang Wu received the B.Eng. degree from Southeast University, Nanjing, China, in 2010, and the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2017. He is currently a Senior Researcher with the YouTu Laboratory, Tencent, China. His research interests include computer vision, machine learning, and their applications in biometric recognition.



Wenqiang Zhang (Member, IEEE) received the Ph.D. degree in mechanical engineering from Shanghai Jiao Tong University, China, in 2004. He is currently a Professor with the School of Computer Science, Fudan University. His current research interests include computer vision and robot intelligence.