GENERATING TRANSFERABLE ADVERSARIAL PATCH BY SIMULTANEOUSLY OPTIMIZING ITS POSITION AND PERTURBATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Adversarial patch is one kind of important form to perform adversarial attacks in the real world and brings serious risks to the robustness of deep neural networks. Previous methods generate adversarial patches by either optimizing their perturbation values while fixing the position on the image or manipulating the position while fixing the content of the patch. In this paper, we propose a method to *simultaneously* optimize the position and perturbation to generate transferable adversarial patches, and thus obtain high attack success rates in the black-box setting. We adjust the transferability by taking the position, weights of surrogate models in the ensemble attack and the attack step size as parameters, and utilize the reinforcement learning framework to *simultaneously* solve these parameters based on the reward information obtained from the target model with a small number of queries. Extensive experiments are conducted on the Face Recognition (FR) task, and the results on four representative FR models demonstrate that our method can significantly improve the attack success rate and the query efficiency. Besides, experiments on the commercial FR service and physical environments confirm the practical application value of our method.

1 INTRODUCTION

Deep neural networks (DNNs) have exposed their vulnerability to adversarial attacks, where adding small imperceptible perturbations to the image can confuse their predictions. However, this small perturbation is not suitable for real applications where images are captured through the camera. A more available way is to use a local patch-like perturbation, where the magnitude of the pixel value changes is not restricted. By printing out the adversarial patch and pasting it on the object, the attack in the real scene can be realized, and it has brought security threats to many tasks such as person detection (Xu et al., 2020), traffic sign recognition (Eykholt et al., 2018; Liu et al., 2019), and image classification (Karmon et al., 2018).

Among these tasks, Face Recognition (FR) is more safety-critical, and adversarial patches have also been successfully applied in this area (Yang et al., 2020; Komkov & Petiushko, 2019; Sharif et al., 2019; Guo et al., 2021). For example, adv-hat (Komkov & Petiushko, 2019) and adv-patch (Pautov et al., 2019; Yang et al., 2020) put the patch generated based on the gradient information on the forehead or nose to achieve attacks. Adv-glasses (Sharif et al., 2016; 2019) confuse the FR system by placing a printed perturbed eyeglass frame at the eye. The above methods mainly focus on optimizing the perturbation content, and the pasting position of the patch is fixed on a position selected based on experience or some prior knowledge. On the other hand, adv-sticker (Guo et al., 2021) adopts a predefined meaningful adversarial patch but uses an evolutionary algorithm to search for a good patch's position to perform the attack. The above methods enlighten us that if we optimize the position and perturbations at the same time, better attack performance can be achieved. However, due to the strong coupling relationship between the position and perturbation values, it is difficult to use the simple gradient-based method or alternate iterative optimization to solve them *simultaneously*, leaving a challenging problem unsolved.

In practical applications, the detailed information about FR threat model usually cannot be accessed. In such case, the existence of transferability of adversarial examples will play an important role. If the adversarial patch generated on the surrogate model has good transferability on the target model, it will

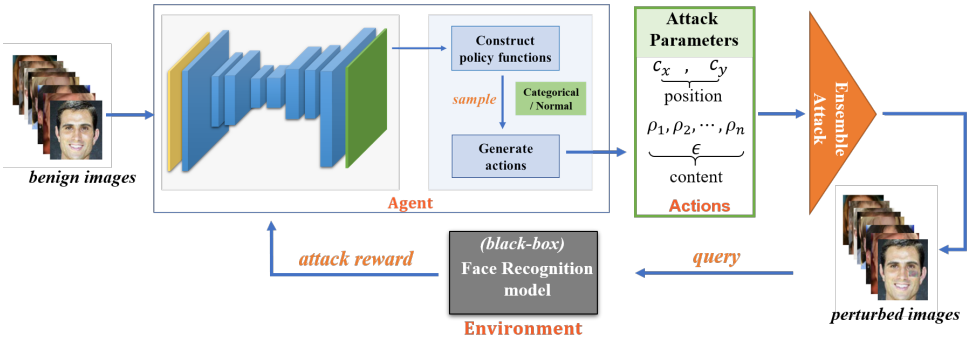


Figure 1: An overview of simultaneously optimizing position and perturbations based on reinforcement learning.

destroy the robustness of the model in an easy-to-implement way, and bring serious consequences. Therefore, an in-depth study on improving the transferability of the adversarial patch can greatly help test the robustness of the FR model in real-world applications.

Based on the above considerations, in this paper, we propose a method to *simultaneously* optimize the position and perturbations of the adversarial patch to improve the black-box attack performance with limited information. However, directly optimizing them will lead to a large number of queries, especially for the perturbations. Because each pixel can range from 0 to 255, the perturbation searching space is so huge. To tackle this issue, we instead use improved I-FGSM with high transferability (Dong et al., 2018; 2019b; Wang et al., 2021) based on the ensemble of surrogate models (Liu et al., 2016) to generate patch's perturbations, and then adjust the attack step size and the weight of each surrogate model to achieve the perturbations. Compared with directly optimizing the perturbation value for each pixel, changing the step size and weights can greatly reduce the parameter space and improve the solving efficiency. Based on this setting, the patch's position, surrogate models' weights and attack step size are the final key parameters needing to learn.

Naturally, simply using some predefined parameters will result in poor transferability, and thus cannot make the adversarial patch achieve satisfactory adaptability to the target model. To further improve the transferability, we use a small number of queries on the target model to dynamically adjust the attack parameters, and this process can be formulated into the Reinforcement Learning (RL) framework. Specifically, the environment in RL is set as the target model, and an Unet-based (Ronneberger et al., 2015) network is used as the agent to *simultaneously* predict the parameters (actions) to generate better adversarial patches. By interacting with the target model, the agent can obtain a reward signal feedback to guide its learning by maximizing this reward (Li, 2017). The whole scheme is illustrated in Figure 1.

In summary, this paper has the following contributions:

- We argue that the position and perturbations of adversarial patch are equally important and interact each other closely. Therefore, a method is proposed to *simultaneously* optimize them to generate transferable adversarial patches, rather than alternate iterative optimization.
- We dynamically adjust the parameters in the transfer-based attack through a small number of queries and formulate the process into a Reinforcement Learning framework, which can guide the agent to generate better parameters with high query efficiency.
- Extensive experiments in dodging and impersonation tasks confirm that our method can realize the state-of-the-art attack performance and query efficiency (maximum success rate of 96.65% with only 11 queries on average), and experiments in the commercial API and physical environment prove the good application value of our method.

2 Related Works

2.1 Adversarial Patch

Compared with L_p norm based adversarial perturbations, adversarial patch (Brown et al., 2017) is a more suitable attack form for real-world applications where objects need to be captured by a camera. Different from pixel-wise imperceptible perturbation, adversarial patch does not restrict the

perturbations' magnitude. Up to now, adversarial patch has been applied to image classifiers (Jia et al., 2020; Karmon et al., 2018), person detectors (Xu et al., 2020), traffic sign detectors (Eykholt et al., 2018; Liu et al., 2019) and many other security-critical systems. For example, the adversarial T-shirt (Xu et al., 2020) evades the person detector by printing the perturbation patch generated by the gradient information in the optimization framework on the center of the T-shirt. Recently, Rao et al. (2020) also propose to jointly optimize location and content of adversarial patch. However, our method is different from it for two aspects: (1) They optimize the location and content via an alternate iterative manner, while our method *simultaneously* solves them. Because location and content have strong interactions, *simultaneously* optimizing them is more reasonable and thus can achieve better solution. (2) They belong to white-box attack against image classification task, while our method is a black-box attack versus face recognition task. Therefore, our task is more challenging.

2.2 Adversarial patch in the face recognition

Adversarial patches also bring risks to face recognition and detection tasks, and their attack forms can be roughly divided into two categories. On the one hand, some methods fix the patch on a specific position of the face selected based on the experience or prior knowledge, and then generate the perturbations of the patch. For example, adversarial hat (Komkov & Petiushko, 2019), adv-patch (Pautov et al., 2019) and adversarial glasses (Sharif et al., 2016; 2019) are classical methods against face recognition models which are realized by placing perturbation stickers on the forehead or nose, or putting the perturbation eyeglasses on the eyes. Yang et al. (2020) put universal adversarial patches on the fixed position of the face to prevent the face detectors from detecting the real faces. The main concern of these methods is to mainly focus on generating available adversarial perturbation patterns but without much consideration of the impact of patch's position versus the attack performance.

On the other hand, some methods fix the content of the adversarial patch and search for the optimal pasting position within the valid pasting area of the face. Adv-sticker (Guo et al., 2021) uses pattern-fixed stickers existing in the real life, and changes their positions through RHDE algorithm based on the idea of differential evolution to attack FR systems. Inspired by this, we believe that the position and perturbations of the adversarial patch are equally important to attack the face recognition system, and if the two are optimized simultaneously, the attack performance can be further improved.

2.3 Deep Reinforcement Learning

Deep reinforcement learning (DRL) combines the perception ability of deep learning with the decision-making ability of reinforcement learning, so that the agent can make appropriate behaviors through the interaction with the environment (Li, 2017; Dong et al., 2019a). It receives the reward signal to evaluate the performance of an action taken through the agent without any supervisory information, and can be used to solve multiple tasks such as parameter optimization and computer vision (Li, 2017). In this paper, we use the information obtained by querying the target model to solve the attack parameters to generate transferable adversarial examples, which can be formalized as the process of using reward signals to guide the agent's learning in the reinforcement learning. Therefore, an agent based on Unet (Ronneberger et al., 2015) is designed to learn parameter selection policies, and generate better attack parameters under the reward signals obtained by querying the target model.

3 Methodology

3.1 Problem formulation

In the face recognition task, given a clean face image $\boldsymbol{x}$, the goal of the adversarial attack is to make the face recognition model predict a wrong identity of the perturbed face image $\boldsymbol{x}^{adv}$. Formally, the perturbed face with the adversarial patch can be formulated as Eq. (1), where $\odot$ is Hadamard product and $\tilde{\boldsymbol{x}}$ is the adversarial patch with perturbations across the whole face image. $\mathcal{A}$ is a mask matrix to constrain the shape and pasting position of the patch, where the value of the pasting area is 1.

$$\boldsymbol{x}^{adv} = (1 - \mathcal{A}) \odot \boldsymbol{x} + \mathcal{A} \odot \tilde{\boldsymbol{x}} \tag{1}$$

The previous methods either optimize $\tilde{\boldsymbol{x}}$ with pre-fixed $\mathcal{A}$, or fix $\tilde{\boldsymbol{x}}$ to select the optimal $\mathcal{A}$. In our method, we optimize $\mathcal{A}$ and $\tilde{\boldsymbol{x}}$ simultaneously to further improve the attack performance.

For the optimization of the mask matrix $\mathcal{A}$, we fix the shape and the size (s_h, s_w) of the adversarial patch, and change its upper-left coordinates (c_x, c_y) to adjust the mask matrix, and the corresponding mask is defined as $\mathcal{A}_c$. In order not to interfere with liveness detection module, we limit the pasting

position to the area that does not cover the facial features like eyes, mouth, etc. For the detailed process, please refer to Appendix A. Thus, in our method, the adversarial patch can be expressed as:

$$\boldsymbol{x}^{adv} = (1 - \mathcal{A}_c) \odot \boldsymbol{x} + \mathcal{A}_c \odot \tilde{\boldsymbol{x}} \tag{2}$$

To generate the perturbation content, the ensemble attack (Liu et al., 2016) based on MI-FGSM (Dong et al., 2018) is here used to generate the transferable patch. For the ensemble attack of n surrogate models, let ρ_i denote the weight of each surrogate model f_i and ϵ denote the attack step size. Then taking the un-targeted attack (or dodging in the face recognition case) as an example, given the ground truth identity y, let $f_i(\boldsymbol{x}, y)$ denote the confidence score that the model predicts a face image $\boldsymbol{x}$ as identity y, then $\tilde{\boldsymbol{x}}$ can be computed by an iteration way. Let t denote the t-th iteration, then:

$$\boldsymbol{x}^{adv} = \tilde{\boldsymbol{x}}_{t+1} = (1 - \mathcal{A}_c) \odot \boldsymbol{x} + \mathcal{A}_c \odot (\tilde{\boldsymbol{x}}_t + \epsilon \cdot \boldsymbol{sign}\,(g_{t+1})) \tag{3}$$

$$g_{t+1} = \mu \cdot g_t + \frac{\nabla_x L\,(\tilde{\boldsymbol{x}}_t, y)}{\|\nabla_x L\,(\tilde{\boldsymbol{x}}_t, y)\|_1} \tag{4}$$

$$L\,(\tilde{\boldsymbol{x}}_t, y) = \sum_{i=1}^{n} \rho_i \cdot \ell\,(\tilde{\boldsymbol{x}}_t, y, f_i) \tag{5}$$

where $\ell\,(\tilde{\boldsymbol{x}}_t, y, f_i) = 1 - f_i\,(\tilde{\boldsymbol{x}}_t, y)$ and $\sum_{i=1}^{n} \rho_i = 1$. For the targeted attack (or impersonation in the face recognition case), given the target identity $\widehat{y}$, ℓ can be simply replaced by $f_i\,(\tilde{\boldsymbol{x}}_t, \widehat{y})$.

Our attack goal is to simultaneously optimize both the patch position and the perturbation to generate good transferable adversarial patches to attack the target model. Therefore, the mask $\mathcal{A}_c$, the attack step size ϵ in Eq.(3) and weights ρ_i in Eq.(5) are set as the learned parameters. To make the parameters more suitable for the target model, we adjust the parameters dynamically through a small number of queries on the target model. The details of solving these parameters are shown in Sec. 3.2.

3.2 Attacks based on RL

3.2.1 Formulation overview using RL

The generation of the transferable adversarial patch on the surrogate model is guided by the information returned by querying the target model, and this process can be represented as the learning of the agent through the reward signal obtained by interacting with the environment in the reinforcement learning (Li, 2017). So an agent is constructed to learn the selection policy of the attack parameters.

In our method, the parameter values are defined as the actions generated by the agent under the guidance of the policy π, and a_t denotes the t-th action (i.e., the value of t-th parameter). The image feature input to the agent is defined as the state s, and the environment is the threat model $F(\cdot)$. The policy function $\pi_\theta(a \,|\, s)$ with parameters θ is a rule used by the agent to decide what action to take, which can be formulated as a probability distribution of action $\boldsymbol{a}$ in the state s.

The reward reflects the performance of the currently generated adversarial patch on the target model, and the training goal of the agent is to learn good policies to maximize the reward signal. In face recognition, the goal of the dodging attack is to generate images that are as far away as possible from the ground-truth identity, while impersonation attacks want to generate images that are as similar as possible to the target identity. Thus, the reward function R is formalized as:

$$R = \begin{cases} \ell\left(\boldsymbol{x}^{adv}, y, F\right) = 1 - F\left(\boldsymbol{x}^{adv}, y\right) & \text{if dodging} \\ \ell\left(\boldsymbol{x}^{adv}, \hat{y}, F\right) = F\left(\boldsymbol{x}^{adv}, \hat{y}\right) & \text{if impersonation} \end{cases} \tag{6}$$

In iterative training, the agent firstly predicts a set of parameters according to policy π, and then the adversarial patch based on the predicted parameters is generated. Finally the generated adversarial face image is input to the threat model to obtain the reward value. In this process, policy gradient algorithm (Sutton et al., 1999) is used to guide the agent's policy update. After multiple pieces of training, the agent will generate actions that perform well on the threat model with a high probability.

3.2.2 Design of the agent

The agent needs to learn the policies of the position, weights and attack step size. Considering the simultaneous solution of the position and other parameters, the design of the agent uses the U-net (Ronneberger et al., 2015) based structure, which can output the feature map of the same length and width as the image matrix fed into the network. Let the number of surrogate models be n, we design

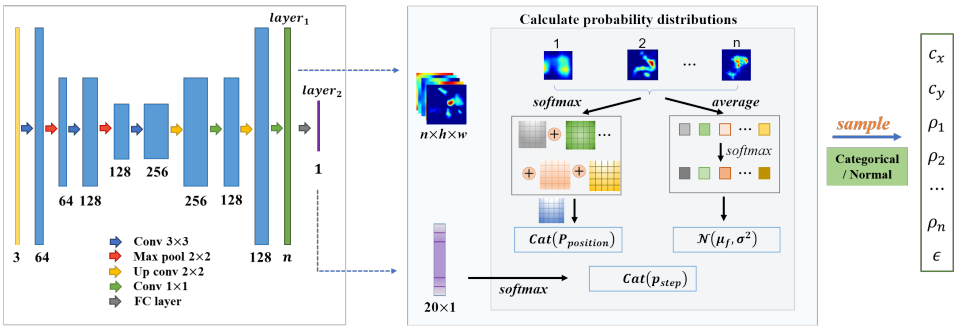


Figure 2: The structure and processing procedure of the agent.

the agent to output the feature map with n channels and the same length h and width w as the input image (i.e. the size is $n \times h \times w$).

In each channel $M_i (i = 1, \ldots, n)$ of the feature map M, the relative value of each pixel point represents the importance of each position for the surrogate model f_i, and the overall value of the channel reflects the importance of the corresponding surrogate model. We believe that the patch requires different attack strengths in different locations, so at the top layer of the agent network, a fully connected layer is used to map the feature map M to a vector V representing different attack step values. The structural details of the agent are shown in Figure 2.

Specifically, for the position action, the optional range of positions is discrete, so the position policy π_θ^1 is designed to follow Categorical distribution (Murphy, 2012). Given the probability $P_{position}$ of each selected position, the position parameters $(c_x, c_y) \sim Cat\left(P_{\text{position}}\right)$, and $P_{position}$ is computed:

$$P_{position} = \frac{1}{n} \sum_{i=1}^{n} softmax\left(M_i\right) \tag{7}$$

For the weight actions, the weight ratio of the loss on each surrogate model f_i to the ensemble loss is a continuous value, and we set the weight policy π_θ^2 to follow Gaussian distribution (Murphy, 2012). So the i-th weight parameter $\rho_i \sim \mathcal{N}\left(\mu_{f_i}, \sigma^2\right) (i = 1, \ldots, n)$, and μ_{f_i} is calculated as:

$$\mu_{f_i} = softmax\left(\overline{M_1}, \overline{M_2}, \ldots, \overline{M_n}\right)_i \tag{8}$$

where $\overline{M_i}$ refers to the mean value of the i-th channel in the feature map, and σ is a hyperparameter. In the actual sampling, we use the clipping operation to make the sum of weights equal to 1.

For the attack step action, we set 20 values in the range of 0.01 to 0.2 at intervals of 0.01, and adopt Categorical distribution (Murphy, 2012) as the step size policy π_θ^3 due to the discreteness of the values. So the step size parameter $\epsilon \sim Cat\left(p_{step}\right)$, and probability p_{step} of each candidate value is:

$$p_{step} = softmax\left(FC\left(P_{position}\right)\right) \tag{9}$$

By sampling from the corresponding distribution, we can obtain $(c_x, c_y), \rho_i (i = 1, ..., n)$ and ϵ.

3.2.3 Policy Update

In the agent training, the goal is to make the agent h_θ learn a good policy π_θ with parameters θ to maximize the expectation of the reward R. Assume that there are T attack parameters to be solved, and $\tau = (s, a_1, a_2, \ldots, a_T)$ is a decision result, then the optimal policy parameters θ^* can be formulated as:

$$\theta^* = \arg\max_\theta J(\theta) = \arg\max_\theta \; E_{\tau \sim \pi_\theta(a|s)}[R(\tau)] \tag{10}$$

We use the policy gradient Sutton et al. (1999) method to solve θ^* by the gradient ascent method, and follow the REINFORCE algorithm (Williams, 1992), using the average value of N sampling of the policy function distribution to approximate the policy gradient $\nabla_\theta J(\theta)$:

$$\nabla_\theta J(\theta) = \; E_{\tau \sim \pi_\theta(a|s)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_t \,|\, s\right) R(\tau) \right] \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_t \,|\, s\right) R_n \tag{11}$$

Algorithm 1 Simultaneous optimization for position and perturbations of adversarial patch

Input: Face image $\boldsymbol{x}$, target $\hat{y}$, agent h_θ, threat model $F(\cdot)$, n surrogate models $f_i (i = 1, \ldots, n)$, iterations Z in MI-FGSM, sample times N, patch size s_h, s_w, learning rate α, training epoch K.
Output: $\boldsymbol{x}^{adv}$

for k = 1 to K **do**
  feature map $M \leftarrow h_\theta$; policies $\pi_\theta^1, \pi_\theta^2, \pi_\theta^3 \leftarrow$ according to Eq. (7), Eq. (8), Eq. (9) ;
  for i = 1 to N **do**
    actions $(c_x, c_y, \rho_1, \rho_2, \ldots, \rho_n, \epsilon) \leftarrow$ sampling from $\pi_\theta^1, \pi_\theta^2, \pi_\theta^3$; $\mathcal{A} \leftarrow (c_x, c_y)$ and s_h, s_w ;
    for t = 1 to Z **do**
      $g_t \leftarrow$ according to Eq. (5) and Eq. (4) ; $\tilde{\boldsymbol{x}}_t \leftarrow$ according to Eq. (3);
    end for
    Update $R_i \leftarrow$ according to Eq. (6);
  end for
  $\nabla_\theta J(\theta) \leftarrow$ according to Eq. (11) ; $\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta)$;
  if $F\left(\boldsymbol{x}^{adv}\right) = \hat{y}$ **then** break;
end for
return $\boldsymbol{x}^{adv}$

where R_n is the reward in the n-th sampling. When updating the policy with the parameter θ, the reward R can be regarded as the step size. The larger the reward, the larger the step size. If the reward is negative, it will go to the opposite direction. In this way, the agent can learn good policy functions with the update of θ in the direction of increasing the reward.

For actions that follow the Categorical policy (i.e., the position parameter and attack step size), given the probability vector p (i.e., $P_{position}$ and p_{step} in Eq. (7) and Eq. (9)), let $p(a)$ denote the probability of action a in the probability vector p, then for π_θ^1 and π_θ^3, $\nabla_\theta \log \pi_\theta(a \,|\, s)$ in Eq. (11) can be calculated as:

$$\nabla_\theta \log \pi_\theta(a \mid s) = \frac{d \, \log p(a)}{d\theta} \tag{12}$$

For actions that follow the Gaussian policy distribution (i.e. the weights of the surrogate models), the mean value μ_f of Gaussian distribution is calculated by the output of the agent, so μ_f can be expressed as $h_\theta(s) = \mu_f$. Therefore, for π_θ^2 following the Gaussian policy, $\nabla_\theta \log \pi_\theta(a \,|\, s)$ in Eq. (11) can be calculated as follows:

$$\nabla_\theta \log \pi_\theta(a \mid s) = \nabla_\theta \left[-\frac{(a - h_\theta(s))^2}{2\sigma^2} - \log(\sigma) - \log(\sqrt{2\pi}) \right] = \frac{a - h_\theta(s)}{\sigma^2} \cdot \frac{dh}{d\theta} \tag{13}$$

3.3 Overall framework

The complete process of our method to simultaneously optimize the position and perturbation is given in Algorithm 1. In the K iterations of the agent training, policy functions are firstly calculated according to the output of the agent, and then perform N sampling according to the probability distribution of the policy function to generate N sets of parameters. According to each set of parameters, the attack is conducted on surrogate models and the generated transferable adversarial examples are used to query the target model to obtain the reward, and then the policy is updated. During this process, if a successful attack has been achieved, the iteration is stopped early.

Furthermore, our simultaneous optimization method can also be combined with other existing black-box attack methods like gradient estimation to further enhance the attack performance. Detailed combination process and experimental results are shown in Appendix B.

4 Experiments

4.1 Experimental Settings

Target models: We choose five face recognition models as threat models, including four representative open-source face recognition models (i.e. FaceNet (Schroff et al., 2015), CosFace50 (Wang et al., 2018), ArcFace34 and ArcFace50 (Deng et al., 2019)) and one commercial face recognition API [1]. During the attack, the four open-source models are candidates for surrogate models, and each model and their ensemble version that excludes the target model will be used as surrogate models.

[1] https://intl.cloud.tencent.com/product/facerecognition

Table 1: Results of dodging (untargeted) attack and impersonation (targeted) attack against FaceNet, ArcFace34, ArcFace50, CosFace50, and commercial API in the black-box setting. We report ASR and average NQ required for simultaneous optimization. Three single models and their ensemble version are used as surrogate models.

	surrogate \ target	FaceNet		ArcFace34		ArcFace50		CosFace50		cml. API	
		ASR	NQ	ASR	NQ	ASR	NQ	ASR	NQ	ASR	NQ
Dodging	FaceNet	-		13.21%	79	16.89%	80	11.47%	72	10.37%	69
	ArcFace34	84.97%	17	-		57.28%	18	23.61%	52	32.31%	53
	ArcFace50	89.37%	14	61.05%	16	-		33.45%	45	38.20%	49
	CosFace50	85.91%	15	53.48%	15	60.51%	23	-		29.41%	57
	Ensemble	96.65%	11	72.86%	18	72.09%	27	62.50%	23	52.19%	46
Impersonation	FaceNet	-		12.77%	23	15.09%	15	8.70%	75	9.52%	161
	ArcFace34	58.53%	25	-		37.73%	47	28.31%	69	20.55%	126
	ArcFace50	53.52%	28	33.23%	60	-		28.30%	53	27.91%	114
	CosFace50	59.21%	31	43.40%	55	39.62%	47	-		16.13%	157
	Ensemble	72.83%	27	50.28%	66	49.50%	36	40.08%	77	37.56%	91

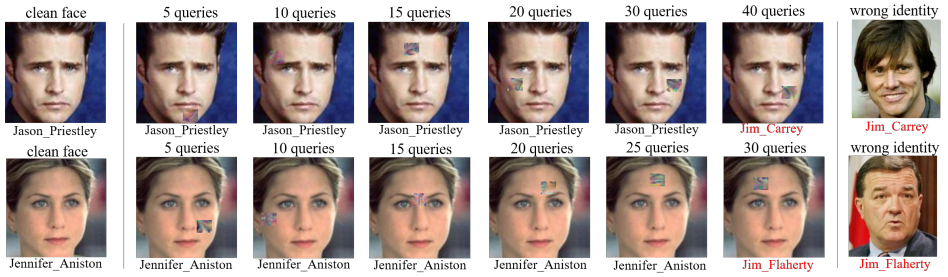


Figure 3: Examples at different stages of the simultaneous optimization process. The black text at the bottom of images denotes the ground-truth identity, and the red text is the false identity after attacks.

Face database: We randomly select 5,752 different people from Labeled Faces in the Wild (LFW)[2] and CelebFaces Attribute (CelebA)[3] to construct the face database. We use the above models to extract the face features, and then calculate the cosine similarity with all identities in the face database to perform the 1-to-N identification.

Metrics: Two metrics, attack success rate (ASR) and the number of queries (NQ) are used to evaluate the performance of attacks. ASR refers to the proportion of images that are successfully attacked in all test face images, where we ensure that the clean test images selected in the experiment can be correctly identified. NQ refers to the number of queries to the target model required by the adversarial patch that can achieve a successful attack.

Implementation: The size of the adversarial patch is set as $s_h = 25, s_w = 30$, the number of sampling N in the policy gradient method is set to 5, and the variance σ of the Gaussian policy is equal to 0.01. Other parameters are set as $Z = 150$, $\alpha = 0.001$, and $K = 50$.

4.2 Experimental results

4.2.1 Performance of simultaneous optimization

We first evaluate our simultaneous optimization method qualitatively and quantitatively according to the setting in Section 4.1. Table 1 shows the quantitative results of dodging and impersonation attacks under the black-box setting against the five target models. For surrogate models, we explore the relationships between individual models and the model ensemble using single models and their ensemble version excluding target models, respectively. Figure 3 shows some visual examples of the position and pattern at different stages of the attack. More visual results are shown in Appendix C.

From the results in Table 1, we can see : (1) the proposed method achieves good attack success rates and query efficiency under both two kinds of attacks. The dodging attack achieves the highest success rate of 96.65% under 11 queries, while the impersonation attack achieves the highest success rate of 72.83% under 27 queries. (2) The performance of using ensemble models to attack is better than that using a single model as the surrogate model, which shows that joint optimization can adaptively adjust

[2]http://vis-www.cs.umass.edu/lfw/

[3]http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

Table 2: Comparison results of the ASR and NQ between our method and the method that only depends on transfer, only changes the position (RHDE), and only changes the perturbation (ZO-AdaMM).

method \ target	Dodging						Impersonation					
	FaceNet		ArcFace50		CosFace50		FaceNet		ArcFace50		CosFace50	
	ASR	NQ	ASR	NQ	ASR	NQ	ASR	NQ	ASR	NQ	ASR	NQ
Transfer-based	28.83%	-	10.35%	-	17.24%	-	9.85%	-	3.31%	-	2.98%	-
RHDE	48.82%	408	35.39%	504	42.93%	514	41.63%	522	29.40%	586	35.64%	577
ZO-AdaMM	65.79%	1972	41.58%	2161	43.43%	2434	50.52%	2874	40.41%	3198	39.39%	3106
ours	**96.65%**	11	**72.09%**	27	**62.50%**	23	**72.83%**	27	**49.50%**	36	**40.08%**	77

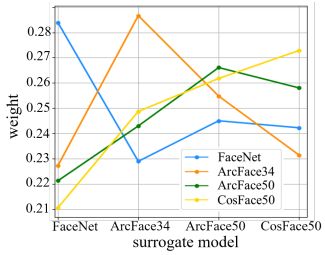


Figure 4: The weights of each surrogate model in the ensemble attack using simultaneous optimization.

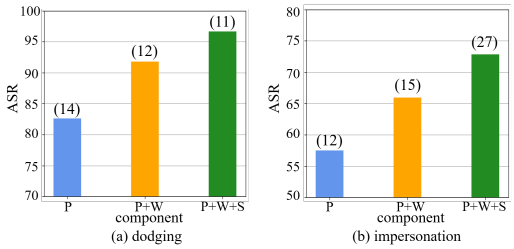


Figure 5: ASR and the NQ (shown in brackets) when using Position (P), Position and weights (P+W), and Position, weights and step size (P+W+S) respectively.

the weights of different surrogate models to achieve the best performance. (3) For the relationship between the surrogate model and the target model, when the two are structurally similar, it is more likely to help the attack. For example, for ArcFace50, CosFace50 has a greater influence on it. This is because the backbone of ArcFace50 and CosFace50 are both ResNet50-IR (Deng et al., 2019), and the backbone of FaceNet and ArcFace34 are Inception-ResNet-v1 (Szegedy et al., 2017) and ResNet34-IR (Deng et al., 2019) respectively. For the influence of model training loss, Facenet uses Euclidean (Schroff et al., 2015) rather than cosine space metric, which is less helpful to ArcFace and CosFace using angular space loss metric when used as a surrogate model alone, with ASR of dodging only 16.89% and 11.47%, respectively. Therefore, in ensemble attacks, we can dynamically adjust the importance of different models by optimizing their weights. (4) On the commercial API, performance drops slightly compared to the open-source model, but remains at an acceptable level. This is because commercial FR services introduce some defense measures like image compression.

4.2.2 Comparisons with SOTA methods

To prove the superiority, we compare our method with three methods: simply relying on the transferability, fixing the perturbation and only changing the position, and randomly initializing the position and only optimizing the perturbation. Specifically, for the method relying solely on the transferability, we use TI-MI-FGSM (Dong et al., 2019b) to generate adversarial examples on the surrogate model, and then directly transfer them to the target model to test their performance. For the position-only method, we adopt the latest RHDE (Guo et al., 2021) method, which fixes the content of the patch and searches for a good position on the face to attack. For the perturbation-only method, we first randomly initialize the position, and then use ZO-AdaMM (Chen et al., 2019) to obtain the adversarial perturbation. The above results on three target models are shown in Table 2.

From the results, we can see that: (1) Although it is relatively simple to rely solely on transferability, the average attack success rate is 12.09% which is not satisfactory. (2) The average success rates for RHDE and ZO-AdaMM are 38.97% and 46.85%, respectively, while our method is 65.61%, which proves that compared with the methods optimized only for position or perturbation, our joint optimization can combine these two attack modes more effectively to achieve better attack. (3) Our method achieves optimal query efficiency among several methods, requiring only a few dozen queries.

4.2.3 Ablation study

In order to verify the effectiveness of each attack parameter used in our method, we also report the results when each component is added separately. First, we fix weights as equal values and the step size as 0.1 to test the performance when only changing the position parameter. Then, we add the weight parameters to learn the importance of each surrogate model. Finally, we add the step size parameters to carry out the overall learning. The results in Figure 5 show that learning only

Table 3: Success rate of dodging (D) and impersonation (I) attack in the physical environment.

	D	I
FaceNet	100.00%	83.33%
ArcFace34	75.29%	65.08%
CosFace50	61.95%	46.71%
cml. API	33.16%	25.23%

Table 4: Success rate in the physical world when changing distance, lighting and face postures including frontal ($0°$), yaw angle rotation $\pm 25°$, $\pm 45°$, and pitch angle rotation $\pm 15°$.

		$0°$	$\pm\, y25°$	$\pm\, y45°$	$\pm\, p15°$	dist.	light
D	FaceNet	100.00%	92.31%	36.36%	87.50%	97.06%	99.79%
	ArcFace34	75.29%	62.50%	28.57%	45.24%	71.43%	65.45%
I	FaceNet	83.33%	70.23%	35.75%	62.50%	88.24%	41.38%
	ArcFace34	65.08%	54.17%	23.81%	57.14%	60.71%	41.81%

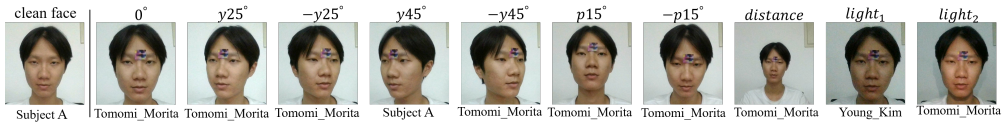


Figure 6: Visual examples of the physical impersonation attack under different conditions. The text at the bottom of images denotes the recognition result of the FaceNet model.

the position parameter can achieve an average success rate of 69.80%. The performance is greatly improved after adding the weight, and further improved after adding the step size. All parameters contribute to the enhancement of the overall attack effect and the weight parameter has a greater impact on the results than the step size. In the process of adding components, the number of queries (shown in brackets) is basically maintained at the same level.

We also make statistics on the weights of each model using simultaneous optimization by taking all four models as surrogate models, and Figure 4 shows the results on a face image when attacking the four threat models. It can be seen that its own model has the highest weight, which indicates that our method can find surrogate models similar to the target model in learning, and the weight of the other three models presents a positive correlation with the ASR shown in Table 1.

4.2.4 Attacks in the physical world

In this section, we show the results of our adversarial patch in the physical environment. We first perform simulated successful attacks on different subjects in the digital environment, and then conduct experiments in the physical world. The technical details of the implementation are shown in Appendix D. We record the video when faces are moving within the range of $5°$ of the current posture, and count the frame proportion of successful physical attack when results are checked every 10 frames at a frame rate of 30fps within one minute as the attack success rate. Table 3 shows the results of the frontal face. To test the performance under various physical conditions, we further change the face posture, the distance from the camera and the illumination. For face postures, we take the conditions of the frontal face, yaw angle rotation of $\pm 25°$ (the mean value at $25°$ and $-25°$), $\pm 45°$ and pitch angle rotation of $\pm 15°$. These results are shown in Table 4.

It can be seen from Table 3 that our method maintains high physical ASR (100.0% and 83.33%) in both dodging and impersonation attacks on FaceNet, and the results are also good on ArcFace34 and CosFace50. Although there is a slight decline in commercial API, the ratio of 33.16% and 25.23% of successful frames is enough to bring potential risks to commercial applications. In Table 4, when the pose changes in a small range ($\pm\, y25°$, $\pm\, p15°$), ASR still maintains a high value. Even if the deflection angle is slightly larger ($\pm\, y45°$), it can still maintain an average of 31.13%. The effect of distance is small, and when lighting is changed, the results are still at an acceptable level. When performing impersonation attacks under different conditions, there is a situation where the face is recognized as a false identity different from the true identity, but not the target identity, which leads to a slightly lower result to a certain extent. A set of visual results is shown in Figure 6.

5 Conclusion

In this paper, we proposed a method to achieve the simultaneous optimization of the position and content to create more transferable adversarial patches. The content was generated by ensemble model attack, and the position, model weights and attack step size are set as parameters. These parameters are dynamically adjusted through a small number of queries with the target model in a reinforcement learning framework to make the patch more suitable for the target model. Extensive experiments demonstrated that our method can effectively improve the attack success rate and the query efficiency, and experiments on commercial face recognition API and physical environments confirmed the practical and effective application value of our method. Besides, the proposed method can also be adapted to other applications, such as automatic driving, etc.

REFERENCES

Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 15–26, 2017.

Xiangyi Chen, Sijia Liu, Kaidi Xu, Xingguo Li, Xue Lin, Mingyi Hong, and David Cox. Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization. *Advances in Neural Information Processing Systems*, 32:7204–7215, 2019.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2019.

Wenkai Dong, Zhaoxiang Zhang, and Tieniu Tan. Attention-aware sampling via deep reinforcement learning for action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8247–8254, 2019a.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9185–9193, 2018.

Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4312–4321, 2019b.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634, 2018.

Ying Guo, Xingxing Wei, Guoqiu Wang, and Bo Zhang. Meaningful adversarial stickers for face recognition in physical world. *arXiv preprint arXiv:2104.06728*, 2021.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning (ICML)*, pp. 2137–2146. PMLR, 2018.

Xiaojun Jia, Xingxing Wei, Xiaochun Cao, and Xiaoguang Han. Adv-watermark: A novel watermark perturbation for adversarial examples. In *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 1579–1587, 2020.

Danny Karmon, Daniel Zoran, and Yoav Goldberg. Lavan: Localized and visible adversarial noise. In *International Conference on Machine Learning*, pp. 2507–2515. PMLR, 2018.

Stepan Komkov and Aleksandr Petiushko. Advhat: Real-world adversarial attack on arcface face id system. *arXiv preprint arXiv:1908.08705*, 2019.

Peter D Lax and Maria Shea Terrell. *Calculus with applications*. Springer, 2014.

Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

Aishan Liu, Xianglong Liu, Jiaxin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. Perceptual-sensitive gan for generating adversarial patches. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 1028–1035, 2019.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

Zuheng Ming, Muriel Visani, Muhammad Muzzamil Luqman, and Jean-Christophe Burie. A survey on anti-spoofing methods for face recognition with RGB cameras of generic consumer devices. *CoRR*, abs/2010.04145, 2020. URL `https://arxiv.org/abs/2010.04145`.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Mikhail Pautov, Grigorii Melnikov, Edgar Kaziakhmedov, Klim Kireev, and Aleksandr Petiushko. On adversarial patches: real-world attack on arcface-100 face recognition system. In *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*, pp. 0391–0396. IEEE, 2019.

Sukrut Rao, David Stutz, and Bernt Schiele. Adversarial training against location-optimized adversarial patches. In *European Conference on Computer Vision*, pp. 429–448. Springer, 2020.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pp. 1528–1540, 2016.

Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. A general framework for adversarial examples with objectives. *ACM Transactions on Privacy and Security (TOPS)*, 22(3): 1–30, 2019.

Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPs*, volume 99, pp. 1057–1063. Citeseer, 1999.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 742–749, 2019.

Guoqiu Wang, Huanqian Yan, Ying Guo, and Xingxing Wei. Improving adversarial transferability with gradient refining. *arXiv preprint arXiv:2105.04834*, 2021.

Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5265–5274, 2018.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

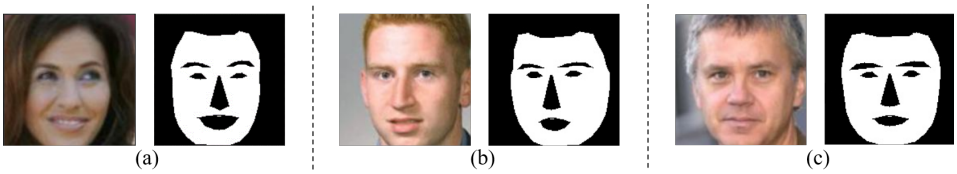
Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world. In *European Conference on Computer Vision*, pp. 665–681. Springer, 2020.

Xiao Yang, Fangyun Wei, Hongyang Zhang, and Jun Zhu. Design and interpretation of universal adversarial patches in face detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pp. 174–191. Springer, 2020.

A OPTIONAL AREA OF THE PASTING POSITIONS

In this section, we give more details about changing the pasting position of the adversarial patch.

It is worth noting that in order not to interfere with the liveness detection module and to maintain the concealment of the attack method, the area that does not cover the facial features of the face (e.g. cheek and forehead) is regarded as the optional pasting area. In practical applications, the liveness detection module is often used in combination with the face recognition to confirm the real physiological characteristic of the object and exclude the form of replacing real faces with photos, masks, etc. (Ming et al., 2020). It is mainly based on the depth or texture characteristics of the face skin or the movements of the object (such as blinking and opening the mouth), so the patch cannot be pasted in the area that covers the facial features (such as eyes and mouth).



(a) (b) (c)

Figure 7: Several groups of faces and the corresponding effective pasting areas. For each group, the left is the face image, and the white part of the right image represents the effective area.

Specifically, we first use `dlib` library to extract 81 feature points of the face and determine the effective pasting region. Figure 7 shows some examples of the effective pasting area corresponding to the face. After calculating the probability of each position $P_{position}$ through the output of the agent (see Eq. (7) in the paper), we set the probabilities of the invalid positions to 0, and then sample the pasting position.

B INTEGRATION WITH GRADIENT ESTIMATION

Our simultaneous optimization method can also be combined with the gradient estimation (e.g., Zeroth-Order (ZO) optimization (Chen et al., 2019; 2017), natural evolution strategy (Ilyas et al., 2018), random gradient estimation (Tu et al., 2019)). It can provide a good initial position and pattern for gradient estimation to further enhance the attack performance. Here we take Zeroth-Order (ZO) optimization (Chen et al., 2019; 2017) as an example to describe the calculation method and the experiment.

B.1 TECHNICAL DETAILS

In the Zeroth-Order (ZO) optimization, to expand the optimization range, $\boldsymbol{x}$ is often replaced with $\frac{1+\tanh \boldsymbol{\phi}}{2}$ (Chen et al., 2017). Since the symmetric difference quotient (Lax & Terrell, 2014) is used at $\boldsymbol{\phi}$ to add a small offset at $\boldsymbol{\phi}$ in the gradient estimation process, the gradient of $\boldsymbol{x}$ at $\boldsymbol{\phi}$ (i.e. $\nabla_{\boldsymbol{\phi}} \boldsymbol{x}$) can not be too small. Therefore, $\nabla_{\boldsymbol{\phi}} \boldsymbol{x}$ is calculated as follows:

$$\nabla_{\boldsymbol{\phi}} \boldsymbol{x} = \frac{1}{2}\left(1 - \tanh^2(\boldsymbol{\phi})\right) \tag{14}$$

Therefore, when combining, in addition to considering the attack target, it is also necessary to ensure that the gradient of the generated perturbation in the $\boldsymbol{\phi}$-space meets the above requirements. So the loss function $L(\cdot, \cdot)$ in Eq. (5) is modified as:

$$L\left(\tilde{\boldsymbol{x}}_t, y\right) = \sum_{i=1}^{n} \rho_i \cdot \ell\left(\tilde{\boldsymbol{x}}_t, y, f_i\right) + \frac{\beta}{s_h \cdot s_w} \sum_{i=1}^{s_h} \sum_{j=1}^{s_w} \nabla_{\phi_{ij}} \boldsymbol{x} \tag{15}$$

where s_h and s_w represent the height and width of the perturbation patch pasted to the face, and β is the scale factor. Given the paste coordinates (c_x, c_y), then $\phi_{ij} = \operatorname{arctanh}\left(2 \cdot \tilde{\boldsymbol{x}}_{t(c_y+i, c_x+j)} - 1\right)$.

Through the simultaneous optimization, we obtain an adversarial patch with good position and perturbation. Next, we regard it as the initialization of the gradient estimation. Specifically, we fix the position and use the gradient estimation method to only refine the perturbation by querying the threat models. After several iterations, the transferability of the adversarial patch is further improved.

B.2 EXPERIMENTAL RESULTS

Table 5: Results of the ASR and NQ required for simultaneous optimization combined with Zeroth-Order optimization. We conduct dodging attack and impersonation attack against FaceNet, ArcFace34, ArcFace50, and CosFace50 in the black-box setting. Three single models and their ensemble version are used as surrogate models.

	target	FaceNet		ArcFace34		ArcFace50		CosFace50	
	surrogate	ASR	NQ	ASR	NQ	ASR	NQ	ASR	NQ
Dodging	FaceNet	-		65.75%	1331	46.89%	1569	49.74%	1885
	ArcFace34	89.47%	721	-		58.58%	1532	53.51%	1904
	ArcFace50	92.37%	804	71.30%	1327	-		63.81%	1432
	CosFace50	91.82%	755	65.48%	1298	62.41%	1384	-	
	Ensemble	97.73%	834	83.28%	1316	74.08%	1521	77.09%	1692
Impersonation	FaceNet	-		58.41%	1844	41.58%	2693	45.70%	2052
	ArcFace34	67.37%	907	-		44.55%	2112	49.53%	1867
	ArcFace50	78.94%	1217	65.35%	1827	-		56.28%	1456
	CosFace50	76.84%	946	63.36%	1578	53.46%	1720	-	
	Ensemble	87.37%	1311	76.29%	1839	66.37%	2542	71.28%	1575

The results in Table 5 show the performance of the integration with gradient estimation. It can be seen that after the combination of gradient estimation, the results have been improved to some extent compared with Table 1, but the consumption of queries has increased accordingly. For example, the success rate of impersonation attacks on CosFace50 increases from 40.08% to 71.28%, but the number of queries increases from 77 to 1575. Therefore, in practical applications, it is necessary to weigh the importance of success rate and query efficiency to determine whether to combine the gradient estimation method.

Table 6: The results of attacking FaceNet using adversarial patch output by simultaneous optimization (SO), simultaneous optimization combined with the Zeroth-Order (ZO) optimization (SO+ZO), and random initialization of perturbation and position combined with the Zeroth-Order (ZO) optimization (Random+ZO), respectively.

method / surrogate	Dodging						Impersonation					
	SO		SO+ZO		Random+ZO		SO		SO+ZO		Random+ZO	
	ASR	NQ	ASR	NQ	ASR	NQ	ASR	NQ	ASR	NQ	ASR	NQ
ArcFace34	84.97%	17	89.47%	721	65.79%	1972	58.53%	25	67.37%	907	50.52%	2874
ArcFace50	89.37%	14	92.37%	804			53.52%	28	78.94%	1217		
CosFace50	85.91%	15	91.82%	755			59.21%	31	76.84%	946		
Ensemble	96.65%	11	97.73%	834			72.83%	27	87.37%	1311		

In order to confirm that the improvement in ASR is due to the good initial value provided by our simultaneous optimization method for gradient estimation, rather than the superiority of the gradient estimation method itself, a comparison is made in Table 6. The results show that simultaneous optimization (SO) can achieve a good ASR with the help of only a few queries (lower than 40 queries) both in dodging and impersonation. When combined with Zeroth-Order (ZO) optimization, the ASR has been further improved (from 72.83% to 87.37% for impersonation), but introduces more query consumption. However, we can see that Random+ZO uses much more queries (1972 and 2874, respectively) but achieves much lower ASR (65.79% and 50.52%, respectively). This contrast proves that our SO method indeed provides a good initialization for the gradient estimation.

C MORE VISUAL RESULTS

In this section, we show more visual results of the simultaneous optimization (SO) attacks. Figure 8 shows four groups of visual results. For each group of three images, the first one represents the clean image, the second one represents the image after the attack, and the third one represents the image corresponding to the wrong identity in the face database.

The above results are obtained by ensuring that the patch is not pasted to the area that covers the facial features. We also conduct experiments when allowing patches to cover facial features in a small range (no more than 20 pixels), and the results are shown in Figure 9. Interestingly, we find that the generated pattern is similar to that of the facial features at the corresponding position, but its

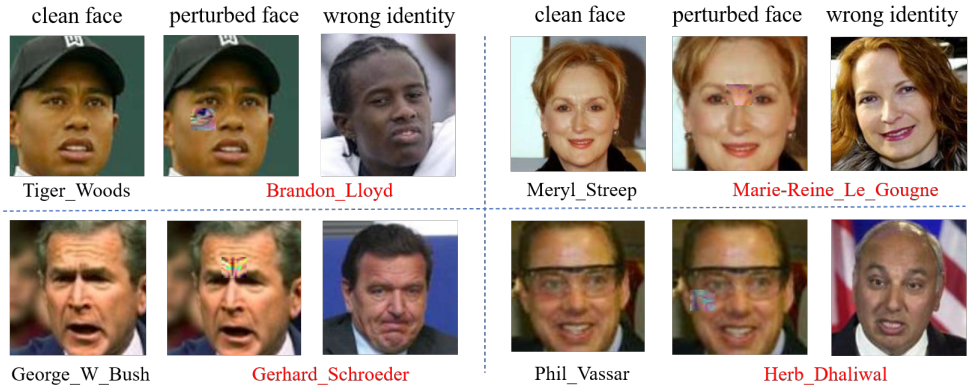


Figure 8: Examples in the simultaneous optimization process. The black text at the bottom of images denotes the ground-truth identity, and the red text is the false identity after attacks.

Figure 9: Examples when allowing patches to cover facial features in a small range. The black text at the bottom of images denotes the ground-truth identity, and the red text is the false identity after attacks.

shape has been changed. For example, in the last group of pictures in Figure 9, the adversarial patch is pasted on the left side eyebrow of the face, and the generated pattern is similar to the shape of the eyebrow, which changes the eyebrow's shape, and may mislead the extraction of eyebrow features by the face recognition network. This inspires us to use some information of facial features (e.g. eyes, eyebrows, nose and mouth) to mislead the feature extraction process of the face recognition, so as to achieve better attack performance.

D IMPLEMENTATION DETAILS OF THE PHYSICAL ATTACK

To make digital simulation results better adapt to the physical environment, we process the smoothness of the pattern. Specifically, during each iteration of the pattern generation, we first obtain a pattern half the size of the original patch by scaling down or averaging pooling, and then enlarge the image back to the original size by bilinear interpolation. The reduced pattern retains the key information of the pattern. After zooming, the smoothness of the image is improved, avoiding the problem that the pattern generated per pixel is sensitive to the position point when it is transferred to the physical environment. Even the pasting position is slightly a few pixels away, the attack effect can still be preserved. We also try to use Total Variation (TV) (Sharif et al., 2019) loss to enhance the smoothness, but the actual effect is not as good as the scaling process. When printing, we use photo paper rather than ordinary paper as the patch material to recreate the colors of the digital simulation as realistically as possible.