ImU: Physical Impersonating Attack for Face Recognition System with Natural Style Changes

Shengwei An†, Yuan Yao‡, Qiuling Xu†, Shiqing Ma§, Guanhong Tao†, Siyuan Cheng†,
Kaiyuan Zhang†, Yingqi Liu†, Guangyu Shen†, Ian Kelk¶, Xiangyu Zhang†
†Purdue University, ‡Nanjing University, §Rutgers University, ¶Clarifai Inc.
Email: †{an93, xu1230, taog, cheng535, zhan4057, liu1751, shen447, xyzhang}@cs.purdue.edu,
‡y.yao@nju.edu.cn, §sm2283@cs.rutgers.edu, ¶ian.kelk@clarifai.com

***Abstract*—This paper presents a novel physical impersonating attack against face recognition systems. It aims at generating consistent style changes across multiple pictures of the attacker under different conditions and poses. Additionally, the style changes are required to be physically realizable by make-up and can induce the intended misclassification. To achieve the goal, we develop novel techniques to embed multiple pictures of the same physical person to vectors in the StyleGAN's latent space, such that the embedded latent vectors have some implicit correlations to make the search for consistent style changes feasible. Our digital and physical evaluation results show our approach can allow an outsider attacker to successfully impersonate the insiders with consistent and natural changes.**

I. Introduction

Deep learning models have been widely used in face recognition systems due to their impressive performance [1], [2], [3], [4]. For example, with sufficient training data, sophisticated model architectures, and advanced training strategies, existing face recognition models can achieve more than 99% accuracy. With the emerging Machine Learning as a Service (MLaaS) provided by large vendors [5], [6], [7], [8], it also becomes easier for non-experts to deploy their own face recognition systems. Despite their impressive performance, recent research has revealed that face recognition models are vulnerable to adversarial attacks that can mislead an input image to a pre-defined target. Most of the existing attacks reside in the cyber space [9], [10], by directly adding *invisible* (stealthy) perturbations to the digital images, and feeding them into the classification models. However, most of these digital attacks rarely threaten the systems in the real world, due to the fact that the perturbation noises cannot be physically implemented or captured by the camera.

To extend the attacks to the physical world, a few impersonating attacks utilize more *visible* noises yet confined in small local (masked) areas (*e.g.*, eyeglass frame [11] or stickers [12]). For example, AdvGlass [11] uses the traditional digital attack to generate unbounded adversarial noises in the eyeglass frame area. The attacker then prints out the patch and attaches it to a pair of glasses. Wearing this pair of glasses can make the model recognize the attacker to the target person.

However, adversarial noises of existing physical impersonating attacks (*e.g.*, noticeable adversarial accessories) are still too distinguishable from natural faces to be stealthy.[1] Also, accessories are not allowed in many security-related situations such as the photos for visa application [13]. Another limitation of existing physical impersonating attacks is that they lack consistent effectiveness across faces captured at different poses, as they usually attack only one face or faces with almost the same poses.[2]

Different from the definition of stealthiness in existing attacks, we argue that the overall perturbation naturalness (such as whole-face style changes consistent across different poses) instead of bounded perturbation at the pixel level can provide better stealthiness. For example, at a security check location, a person wearing a pair of weird glasses is more ostentatious and suspicious than a person wearing daily makeup. To this end, we utilize StyleGAN, which has been widely used in image editing tasks [14], [15] due to the embeddability (i.e., the ability to invert a given image to a latent value of the StyleGAN) and semantic editability (i.e., style editing) of its latent space. Although existing methods [15], [16] can highly effectively mutate the style of a given image, they can hardly achieve consistent style changes across different images from the same physical person (potentially with different poses), which is key to achieving an effective physical impersonating attack.

In this paper, we propose a novel impersonating attack based on StyleGAN. Given a set of pictures of the attacker, under different conditions and having different poses, our attack aims to produce consistent style changes that are physically realizable and can flip the classification result to a target person. In order to achieve the goal, we need to embed the set of images, namely, generating the latent vectors for these images. The StyleGAN does not have any concept of face identity and hence the embedded latent vectors may not have any strong correlations although they belong to the same physical person. As a result, finding a consistent style change for this set of images (from the same physical person) is no different from finding it for a set of arbitrary images, which is difficult. We have two key observations that allow us to

[1]We conducted a user study via Amazon MTurk with 240 users. About 70% more users consider AdvGlass's images are more noticeable than ours.

[2]Although there exist some spatial transformations to ease this problem for traffic signs, how to apply them for face images (e.g., from the frontal face to the side face) is still largely open.

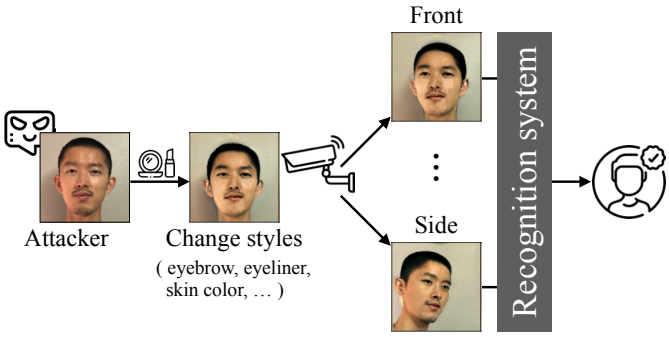


Fig. 1: Attack scenario.

address the problem. First, we find that poses of generated faces by the StyleGAN can be controlled by injecting specific noises at certain style blocks. Second, given an attacker's image with pose p, if we can find the corresponding noise τ and then enforce such τ during embedding, we are able to force the StyleGAN to establish some implicit connections for the multiple images of the attacker, like recognizing that they belong to the same person. As such, we can find a consistent style change for these *properly embedded* images (i.e., images generated by the embedded latent vectors) that enables the impersonating attack.

In summary, we have the following contributions:

- We propose a novel impersonating attack approach based on StyleGAN. It supports both white-box and black-box settings.
- We study the effects of random noises injected in StyleGAN. Based on our new findings, we design a novel embedding approach to faithfully project real attacker's images with different poses into the latent space.
- We develop a batch-based attack method with statistical constraints to generate consistent, physically applicable and stealthy style changes for images of the same person with different poses.
- We build a tool ImU (Physical Impersonating Attack for Face Recognition System with NatUral Style Changes) and evaluate it on 10 large models pre-trained on 4 large-scale face recognition datasets and 2 commercial services. Digital evaluation results in both white-box and black-box settings show our approach can generate consistent natural style changes for faces with different poses. In the white-box setting, our approach can achieve the highest attack success rate even with the physical-world simulation (*i.e.*, adding noises to and cropping and rescaling the images). In the black-box setting, our approach can achieve 16x higher attack success rate compared to the existing methods. After physically applying the generated changes, attackers can successfully conduct the physical impersonating and fool the classifiers. We also show attackers can physically attack the online commercial services. On models trained with different adversarial training strategies (even with the adaptive defense), our approach still yields high attack success rate.

II. Physical Impersonation Attacks

In this section, we first introduce our threat model and then summarize four requirements of realizing physical impersonation attacks. Afterwards, we show existing methods fail in fulfilling them and showcase our results.

A. Threat Model

Attack Scenario. Figure 1 shows our attack scenario. Our subject model is a face recognition system based on a neural network classifier or verifier.

A classifier is trained on face images from a set of N authorized people. Given one image, the system predicts the label from N. When recognizing a person, the system requires taking multiple photos with different poses. The attackers are *outsiders* not among the N people (*i.e.*, out-of-distribution) who want to impersonate some target people (*e.g.*, label 1) among the N people (*i.e.*, in-distribution).

A verifier is trained to extract features of given images so that images of the same person have smaller feature distances (*e.g.*, higher cosine similarity score) than those of different people. During recognition, if the similarity score of two images (*i.e.*, the attacker's and target person's images) is larger than a pre-defined threshold, they are considered as one person.

Specifically, the outsider attacker aims to *physically* impersonate the target people by changing the styles. With the changed styles, the attacker's images captured with different poses should be recognized as the target person by the system. More importantly, the style change should not be ostentatious or suspicious.

Attackers' Capability. We assume that attackers can utilize public face data (*e.g.*, CelebA [17]) to train generative models (*e.g.*, StyleGAN [18]) . The training data can be non-overlapping with the subject model's training data. For face classifiers, attackers need no access to the target person's image. For face verifiers, attackers have the target person's image. In the white-box setting, attackers can access the internals of the subject model and thus can use a gradient-based method to conduct the attack. In black-box attack on a classifier, attackers treat the model as a black box and use a query-based method. In black-box attack on a verifier, attackers conduct a transferable white-box attack against surrogate verifiers. Our attacker's capability is consistent with existing literature [11], [19], [20], [21], [22]. Attackers should be able to apply makeup by themselves or ask some professionals to do that.

B. Requirements of Physical Attacks

We follow the standard way to conduct physical attacks: the attacker first generates the adversarial image in cyberspace, and then tries to physically realize it in the real world. In the following, we summarize four challenging requirements for conducting physical attacks.

Requirement 1: ability of using outsider attacker's images. A physical attack uses a real, outsider attacker, and an attack

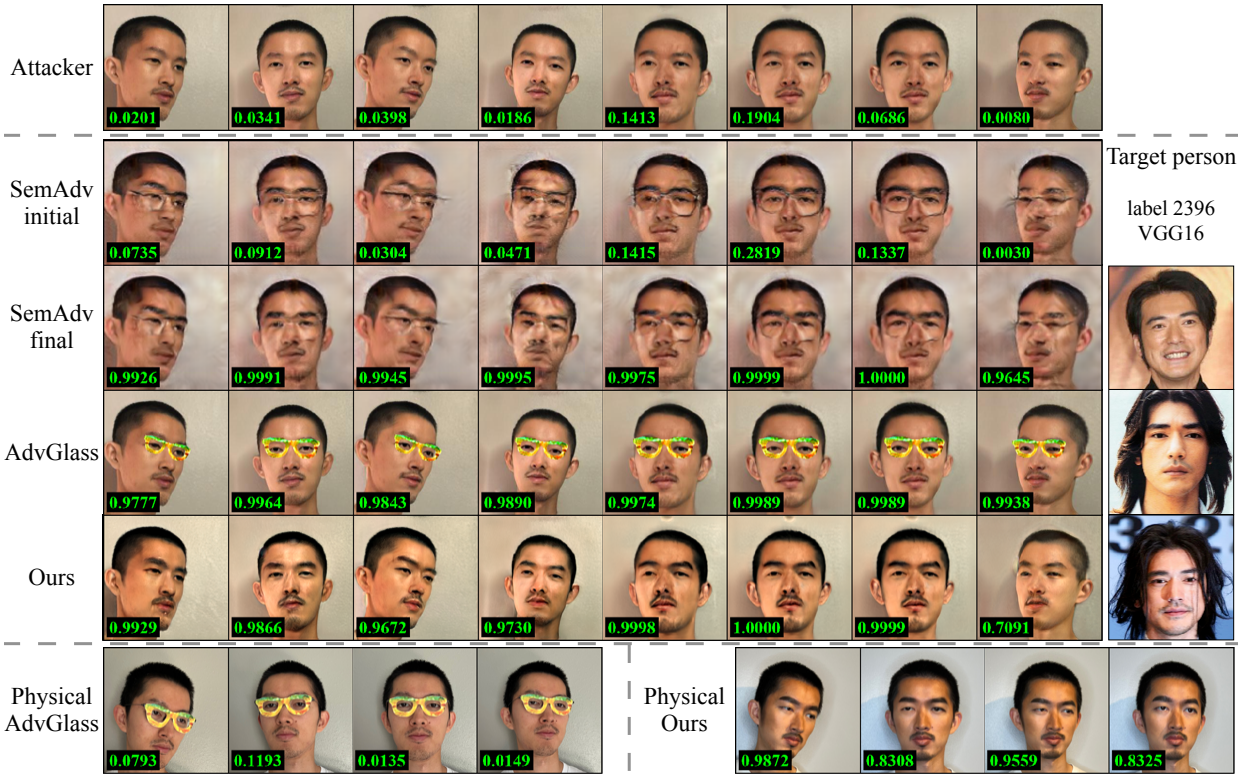


Fig. 2: Digital/physical examples of different attacks. The target person is label 2396 of VGGFace dataset and the subject model is VGG16. Three ground truth images of the target person are displayed in the rightmost column. The first row shows the images of an attacker with different poses. The number in each image denotes the confidence. AdvGlass's results are more obvious/ostentatious and less effective than ours. SemAdv was proposed as a digital attack. Appendix has another example where the same attacker impersonates an actress.

method should be able to use real people's images. For example, based on one real image of the attacker, the attack method should return an updated/generated image which can 1) fool the subject model, and 2) guide the attacker to physically apply the changes.

Requirement 2: physical applicability of the adversarial changes. We need to be able to physically apply the adversarial changes for a physical attack. For example, we cannot exactly map each pixel in the digital image (*e.g.*, one pure red pixel) to a certain dot of the face. In addition, sharp change of colors usually cannot be captured by the camera.

Requirement 3: stealthiness of the adversarial changes. In the physical world, we also aim to keep the stealthiness of the attack, preventing the adversarial changes from being too suspicious in humans' perspective. For example, it would be too ostentatious to have a half-red and half-green face.

Requirement 4: effectiveness across different poses and views with the same adversarial change (*i.e.*, consistency). Finally, it is necessary that the attacker with the style changes should be able to consistently intrude the subject classifier at various poses. The reason is that when captured by the camera, the attacker cannot exactly ensure a certain pose.

C. Limitations of Existing Methods

AdvGlass [11]. AdvGlass is a representative physical attack in literature. It uses traditional unbounded digital (pixel-based) attacks (*e.g.* PGD [9]) and confines the optimization in certain areas (*e.g.*, eyeglasses). Specifically, given an image x of the attacker, AdvGlass uses an eyeglass frame mask M to constrain the optimization of adversarial perturbations δ such that $x \odot (1 - M) + \delta \odot M$ is recognized as the target person. M only contains 1 (for the eyeglass frame area) and 0 elsewhere. It also enhances the physical applicability by encouraging the colors used by the adversarial perturbations to be close to a set of printable colors. Then, the attacker can print out $\delta \odot M$ and attach it on a real eyeglass frame. To robustify the adversarial glasses, AdvGlass uses a set of images with *slightly* different poses (almost frontal views) and thus the glass frame mask position is almost fixed without any perspective transformations.[3]

There are three drawbacks of AdvGlass. First, accessories like eyeglasses are not allowed in many security-related circumstances such as photos used in visa applications [13]. Second, the weird adversarial glasses are still too ostentatious

[3]In practice, we extend AdvGlass by automatically resizing and rotating the mask and noises to fit the different poses.

and suspicious to be stealthy. From the fourth row of Figure 2, we can observe that the eyeglasses indeed have strange colors and are not natural. Third, although AdvGlass uses a set of images with *slightly* different poses, the glass frame mask position is still almost fixed frontal views without any perspective transformations. Consequently, with printed eyeglasses, the attacker may not be able to successfully launch the attack with different poses. The results are shown in the left part of the last row in Figure 2. None of them are successfully recognized as the target person[4].

SemAdv [19]. Applying usual digital (pixel-based) attacks in the physical case is extremely challenging, and SemAdv is a potential choice as it is able to edit images on the attribute level. Specifically, SemAdv first uses a single-attribute editor network (*i.e.*, a pair of encoder E and decoder D) to change one feature c of the input image x (*e.g.*, adding/removing glasses). For example, x is a face without glasses (*i.e.*, the original feature vector $c_0 = 0$), $D(E(x, 1-c_0))$ will add a pair of glasses to x. To make the edited image recognized as the target person, SemAdv further optimizes a coefficient tensor to interpolate between the internal encoding of the attacker's raw image and the edited image.

However, applying SemAdv physically still has several limitations. First, as shown by the third row of Figure 2, the style changes at different poses are not consistent. This is because SemAdv can only attack one image (frontal view). Second, SemAdv is not physically applicable as the adversarial noises are dispersed across the image instead of concentrating on the faces (see Figure 29 in the appendix). Third, SemAdv's capability of using real images is closely dependent on the capability of D and E (*i.e.*, StarGAN [23] in their paper). As such, SemAdv can only work for in-distribution data (*i.e.*, frontal view faces very similar to training data) because StarGAN is less generalizable on unseen data than some more sophisticated GANs (*e.g.*, StyleGAN).

AdvMakeup [20]. Similar to the eyeglasses mask in AdvGlass, AdvMakeup also uses a mask M to only modify the orbital area. AdvMakeup attacks verifiers. For *one* target image x_t, it trains a model G to generate changes $\delta = G(x_a)$ for *one* image x_a of an attacker such that $x_a \odot (1 - M) + \delta \odot M$ and x_t have a small feature distance. Figure 3 shows the results of ours and AdvMakeup. Because of the mask-based modification, the patched area has visual disparities (thus less stealthy and natural). Also, AdvMakeup attacks each image separately, and there are hence no explicit constraints on the consistency.

D. Our Digital and Physical Examples

Table I summarizes to what extent different attacks satisfy the four requirements. We aim to simultaneously satisfy the four requirements, and some examples are shown in Figure 2. Compared to existing methods, our results (fifth row) are consistent among different poses, natural-looking, and thus stealthier. Different from existing physical attacks, we claim that the small size of adversarial noises is not the necessary condition of a stealthy attack, and instead the stealthiness in the physical world is highly related to the naturalness of the adversarial changes. One intuitive example is changing an attacker's facial style with cosmetics. In order to show the effect of our physical attack, we asked a cosmetology student to apply makeup on the attacker according to the generated adversarial images. The results are shown in the right part of the last row. All of them are recognized as the target person with much higher confidence compared to AdvGlass. To check if the confidences of our physical examples are high enough to break the face classifier, we collected a set of images of the target person online (not in the training dataset) and tested their confidences. All the downloaded images are correctly classified as the target person. If we set a confidence threshold for these images, the threshold should be smaller than 0.5853 so as to ensure 60% accuracy. In this case, all of our physical examples can pass this threshold.

TABLE I: Summary of different attacks

Challenges	AdvGlass	SemAdv	AdvMakeup	Ours
1) Real attackers	Yes	No	Yes	Yes
2) Physical applicability	Partial	No	Yes	Yes
3) Stealthiness	Partial	-	Partial	Yes
4) Consistency	Extended by us	No	Partial	Yes

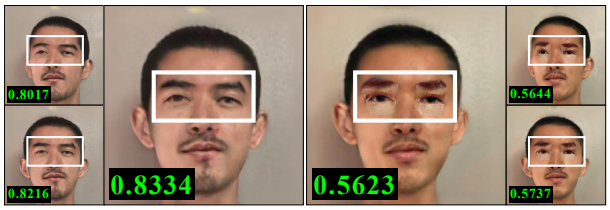


Fig. 3: Examples of ours (left) and AdvMakeup (right). The subject model is a verifier FaceNet pre-trained on VGGFace2. The target image is the third target image in Figure 2. Please zoom in for a better inspection.

III. Background: StyleGAN

In this section, we introduce StyleGAN, which is an important component of our approach. Traditional GANs were proposed to learn a mapping from a latent space $\mathcal{Z}$ (*e.g.*, Gaussian distribution) to another space $\mathcal{X}$ (*e.g.*, Human face distribution). However, the generated image's quality and resolution still have a lot of room to improve. In order to generate better images, StyleGAN was proposed by [18]. Unlike traditional GANs with one latent space $\mathcal{Z}$, StyleGAN utilizes another intermediate latent space $\mathcal{W}$ mapped from $\mathcal{Z}$. This intermediate latent space will be further transformed into the styles allowing gradual adjustment of styles at different granularities to forge the final image.

Figure 4 shows the simplified architecture of StyleGAN which can be partitioned into two parts: a mapping network F (left) and a synthesis network G (middle). The original

[4]AdvGlass was also reported to have a poor physical attack performance on complicated models [20]. The models physically attacked in the AdvGlass paper recognized only $\leq$ 143 identities [11], whereas VGG16 here recognizes $\geq$ 2.6K identities.

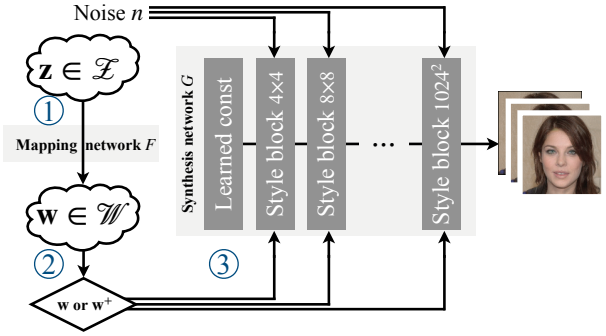


Fig. 4: StyleGAN architecture.

StyleGAN uses 8 fully connected layers with the Leaky ReLU activation functions to compose network F and 9 style blocks (scaled from 4^2 to 1024^2) for network G. Each style block has two convolutional layers and two Adaptive Instance-Normalization (AdaIN) layers. Each AdaIN layer follows one convolutional layer and tunes the styles of its feature maps by adjusting the means and variances [24]. The target means and variances (*i.e.*, styles) are produced by linearly transforming $\mathbf{w} \in \mathcal{W}$. To increase the stochastic variation of the generated images (*e.g.*, the exact placement of hairs), each style block also employs random noises to slightly perturb the output of each convolutional layer. Depending on whether the same $\mathbf{w}$ or different $\mathbf{w}$'s are used at each style block, $\mathcal{W}$ can be extended to $\mathcal{W}^+$. $\mathcal{W}^+$ space is used to fulfill the mixing regulation in training StyleGAN and to embed real (unseen) images in the StyleGAN-based image edition.

We first formalize the original synthesis procedure according to Figure 4. Step ① samples a latent vector $\mathbf{z} \in \mathcal{Z}$ and maps it to $\mathbf{w} \in \mathcal{W}$ via network F, *i.e.*, $\mathbf{w} = F(\mathbf{z})$. Step ② duplicates $\mathbf{w}$ l times to get $\{\mathbf{w}_i\}_{i=1}^{l}$ where l is the number of AdaIN layers. Step ③ transforms the $\{\mathbf{w}_i\}_{i=1}^{l}$ to means and variances, adds random noises $\{\mathbf{n}_i\}_{i=1}^{l}$, and generates the final images $G(\{\mathbf{w}_i\}_{i=1}^{l}, \{\mathbf{n}_i\}_{i=1}^{l})$. For simplicity, we by default omit the subscripts of $\mathbf{w}$ and $\mathbf{n}$ and add them when necessary. When the training is finished, F, G and noises $\mathbf{n}$ are fixed. We denote the final image as $G(\mathbf{w})$ and the fixed noises as $\mathbf{n}_*$.

Due to the disentanglement and the expressiveness of $\mathcal{W}^+$, StyleGAN is widely used in image edition tasks. To edit an image x, the first step is embedding which finds a latent vector $\mathbf{w}$ so that the embedded image $G(\mathbf{w}) = x$. The embedding procedure can be accomplished by an encoder-based method or an optimization-based method. The former learns a network E to minimize $\mathbb{E}_{x \in \mathcal{X}}[L(G(E(x)), x)]$ on a training dataset $\mathcal{X}$, and returns $E(x)$ as the embedding of a given unseen input x. The latter directly solves the equation $\text{argmin}_{\mathbf{w} \in \mathcal{W}^+} L(G(\mathbf{w}), x)$ for a given x to get an embedding in space $\mathcal{W}^+$. Note that we have similar equations for other latent spaces. In both cases, L is the loss function that measures the similarity between the given input and the reconstruction (*e.g.*, LPIPS [25]). We will discuss different embedding methods in the design section.

IV. Design

Figure 5 shows the overall workflow of our approach. To facilitate the attack from real outsider attackers (*i.e.*, the first requirement), step ① aligns (step ①.1) and embeds (step ①.2) a batch of the attacker's images with different poses to the $\mathcal{W}^+$ space. To generate consistent style changes for different poses (*i.e.*, the fourth requirement), step ② finds one style change direction d for the whole batch of latent vectors in the $\mathcal{W}^+$ space, so that the subject system recognizes all the generated adversarial images as the target person. Specifically, Step ②.1 adds the initial d to images, and Step ②.2/②.2′ searches for a better d. The top part of step ② shows the white-box setting. We use gradient-descent optimization to update the direction d to minimize the Carlini-Wagner (CW) loss for the subject classifier or the cosine similarity loss for the subject verifier, which is computed on the output logit vector of the subject model. The bottom part of step ② describes the black-box setting. For a classifier, we can only access the scalar confidence value. We use a search algorithm (*e.g.*, genetic algorithm) to search for the better d. For a verifier, we white-box attack the surrogate verifier to generate transferable adversarial examples. The iteration in step ② terminates when the attack succeeds or the pre-defined maximum iteration number is reached. During step ②, we intentionally bound the change d to ensure it is sufficiently feasible in practice (*i.e.*, the second requirement) and can generate natural style changes (*i.e.*, the third requirement).

A. Aligning and Embedding Real OOD Images

In order to meet the first requirement, one necessary process is to embed the real attacker's images captured at different poses into the $\mathcal{W}^+$ space of StyleGAN, so that we can exploit the style changes. However, due to the fact that most face StyleGANs are trained on faces aligned (*e.g.*, rotated, shifted, and affine-transformed) to a certain pose and size [18], [16], not only the attacker's image is OOD w.r.t. the training data of StyleGAN, but the different poses may also be OOD. This raises challenges for existing embedding methods and makes them insufficient to embed the real OOD images. For simplicity, we call the existing StyleGAN an *aligned* StyleGAN. Examples of aligned and unaligned training data are in Section E of the appendix. Based on our new observation on the random noises used in an *unaligned* StyleGAN, we propose our novel embedding approach.

1) Existing methods cannot faithfully embed images with diverse poses: As mentioned in the background section, existing embedding methods can be mainly categorized into two types: encoder-based or optimization-based. Figure 6 shows the embedding results of different methods, where we embed 4 images for each of two randomly selected people. The first row shows the real images of different poses. The second row shows the embedded images using the encoder-based method e4e [15]. Although the embedded images *overall* look a bit similar to the real images, they are actually quite different in many important aspects. If we zoom in the figure and compare the result in each column of the first two rows, we can see

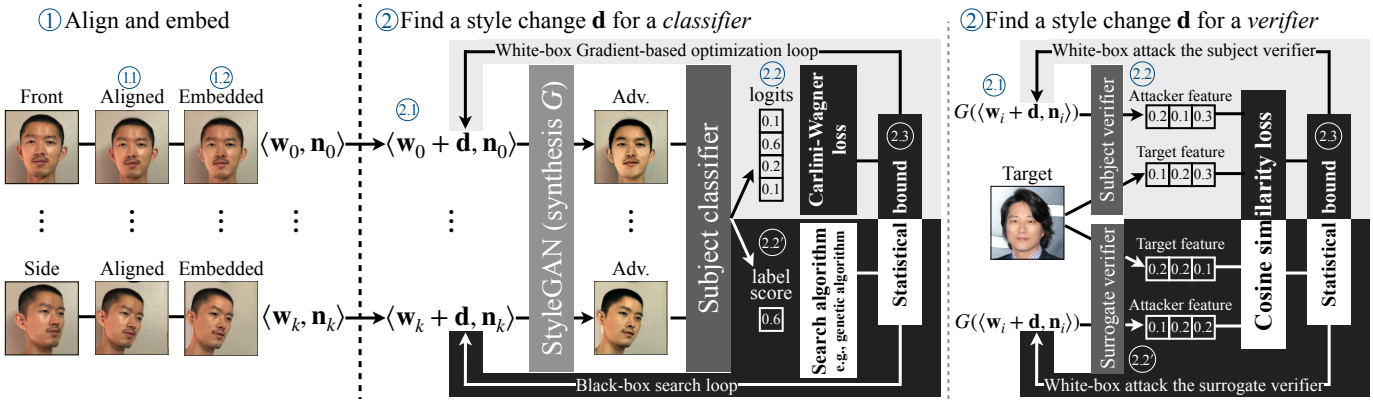


Fig. 5: The workflow of the proposed physical attack. Step 1 embeds images with different viewing angles. Step 2 finds a consistent style change for the embedded images. In the white-box setting of attacking a classifier, we use the gradient-descent optimization, and the search algorithm in the black-box setting. When white-box (resp. black-box) attacking a verifier, we follow the existing literature and use gradient-descent optimization on the subject (resp. surrogate) verifier.

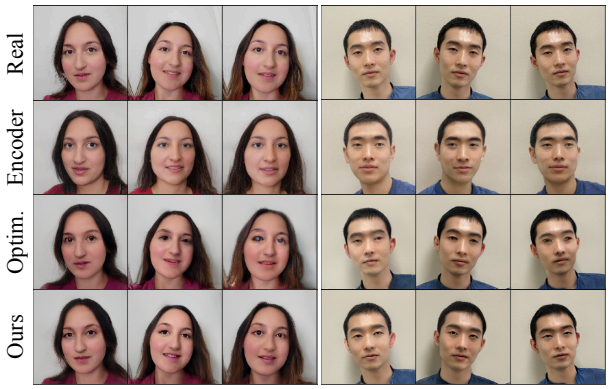


Fig. 6: Different embedding methods. The encoder-based method cannot embed different poses well and most of them have almost vertical roll angles. The optimization-based method can better embed the poses, but not the features (*e.g.*, eyes).

Fig. 7: Effects of stochastic noises at the 4x4 style block. The same noise brings the same pose.

that the face shape and fine features (*i.e.*, eyes, nose, mouth) are different. The differences are more obvious if the attacker is more different from the training data as shown by the four images on the right hand side. Additionally, the embedded images have almost the same vertical angles that are different from the real poses. These results mean that the encoder-based method cannot faithfully embed the fine features and the diversity very well. The third row shows the results of the optimization-based method [16]. We can see now the diverse poses are better embedded, but the fine features (*e.g.*, eyes) are worse. The embedding is obtained by the optimization w.r.t. to one given image instead of a learned mapping like the encoder-based method. Consequently, the optimization essentially tries to match the embedded image with the real image as much as possible but can go to some unnatural manifolds. The last row in Figure 6 gives a glimpse of our faithfully embedded results before we dive into the details.

2) Random noises of an unaligned GAN determine poses: Many existing works studied the effects of the $\{\mathbf{w}_i\}_{i=1}^{l}$ vectors at different style blocks: coarse styles ($\{\mathbf{w}_i\}_{i=1}^{4}$ at 4^2-8^2 style blocks) describe the high-level features (*e.g.*, face shape), and fine styles ($\{\mathbf{w}_i\}_{i=8}^{l}$ at 64^2+ style blocks) mainly focus on the micro-structures and color schemes. However, none has studied the effects of the noises at different style blocks of a StyleGAN trained on the unaligned dataset. This is because most of the facial StyleGANs are trained on aligned faces and *their noises only bring local stochastic variations* as shown by Figure 20 in Appendix. Note that, the pre-trained StyleGAN used here has 4^2-256^2 style blocks and outputs images with resolution 256×256.

Interestingly, we observed that *the noises at the 4^2 style block of an unaligned StyleGAN actually bring the pose variation*. Figure 20 in Appendix shows the effects of noises at different style blocks from 4^2 to 256^2 with the same $\mathbf{w}$. Each row randomly samples 8 noises at the indicated style block and generates 8 corresponding images. The rightmost image shows the average difference across the 8 images. From the generated images and the differential images, we can see noises at 8^2+ style blocks of an unaligned StyleGAN change local features similar to the noises in an aligned StyleGAN, while noises at the 4^2 block vary the poses. Furthermore, we find that the same noises at the 4^2 style block can produce the same poses for different $\mathbf{w}$'s as shown by Figure 7. Every column uses the same noises (and thus the same pose), and each row uses the same $\mathbf{w}$ (and thus the same person). This property is not possessed by an aligned StyleGAN and we are

Fig. 8: Weaker effects of stochastic noises at the 4x4 style block for real images. The same noises are used as Figure 7.

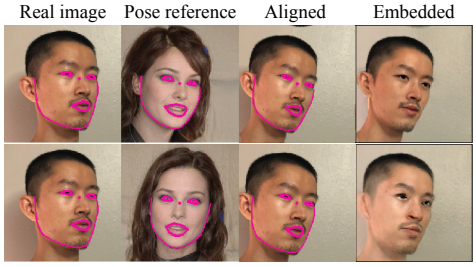


Fig. 9: Different alignment examples. Better alignment leads to better embedding.

the first to observe this and use it to better embed faces of different poses. More discussion can be found in Section A of the appendix.

3) Our alignment and embedding approach: One straightforward idea to obtain images with different poses is to embed one image to the latent space and then sample different 4^2 noises. However, this cannot work as expected for attackers because the embedded vector's distribution is very different from the latent space's distribution. Figure 32 in the appendix shows histograms of the training and embedded $\mathcal{W}^+$ vectors. The results are shown in Figure 8 and we observe some bad reconstructions (*e.g.*, the second row).

In this work, we propose a new alignment and embedding approach. Our key insight is that the embedding quality is substantially confounded by poses and the corresponding perturbations. In other words, if we can find a way to disentangle (not eliminate) this pose factor from the embedding procedure, we can have better embedded results while preserving the diversity. Since we have observed that the random 4^2 noises can provide various poses, we can leverage them to fulfill the poses and let the embedding procedure focus on the remaining features. In particular, we first generate a set of randomly sampled 4^2 noises and record the effects (*i.e.*, pose references) of them. For a given attacker image, we need to decide a pose reference from the random set to align with such that the aligned image and the pose reference have similar face sizes, yaw angles, and roll angles. We then use the corresponding $\mathbf{n}_{4^2}$ to replace the noise stored in StyleGAN to embed the aligned image. Selecting the right pose reference is non-trivial, and an unsuitable pose reference may lower the quality of the generated image. Figure 9 shows different pose references indeed affect the embedding results. The two rows try to align the same real image according to different pose references to get the embeddings. Obviously, the first row gives the better embedding (the reconstructed image is better) and therefore

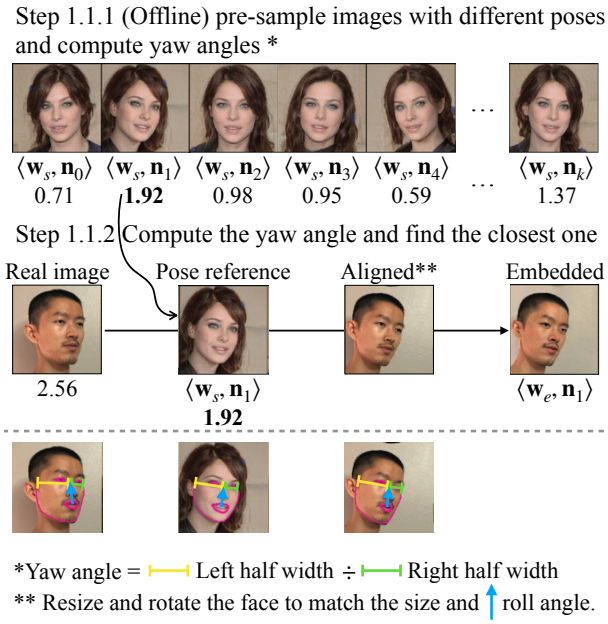


Fig. 10: Auto alignment workflow. The top part shows the two steps of the alignment and the bottom part shows the details of computing the yaw angle and aligning a real image according to a pose reference.

we prefer the first pose reference to the second one.

In order to automatically find the good pose reference and its corresponding random noise, we design an auto-alignment method. Figure 10 shows its workflow. Step 1.1.1 first generates a set of images with different poses with one $\mathbf{w}_s$ and a set of randomly sampled noises $N_{4^2} = \{\mathbf{n}_{4^2}\}$. We also compute the yaw angles for the generated images. Step 1.1.1 can be done offline. Step 1.1.2 finds the $\mathbf{n}_{4^2} \in N_{4^2}$ whose image (called the pose reference) has the closet yaw angle to the real image and align the real image according to this pose reference. The selected $\mathbf{n}_{4^2}$ will replace the original $\mathbf{n}_{*4^2}$ during the embedding procedure. As explained in the bottom part of Figure 10, the yaw angle is measured by the ratio between the width of the left half face (depicted by the yellow segment) and the right part (depicted by the green segment). The width is calculated based on the output (depicted by the pink lines) of existing landmark detection methods. To align the image, we re-scale the real image to match the face size and rotate it to match the roll angle (depicted by the blue arrow). In this example, the real image's yaw angle is 2.56 and the selected pose reference is the second image ($\langle \mathbf{w}_s, \mathbf{n}_1 \rangle$) whose yaw angle is 1.92.

Algorithm 1 shows the pseudocode of our auto alignment. It takes in a real image x to embed and returns the aligned image with the selected 4^2 noise. Line 2 gets the set P of the pre-sampled images with different 4^2 noises. Line 3 computes the yaw angle of x. Line 5-10 go through P to find the one with the closest yaw angle and store the pose reference image in t_{best} and the noise in $n_{4^2_{best}}$. Lines 11-12 compute the center point, size, and the roll angle of the faces in x and t_{best}. Lines 13-14 compute the scale to match the face sizes and the

Algorithm 1 Auto alignment (step 1.1)

1: **function** AUTOALIGN(image x)
2: P = GETSAMPLEDPOSES() ▷ Step 1.1.1 (offline)
3: α_x = GETYAWANGLE(x)
4: $t_{best}, r_{best}, n_{4^2_{best}} = \perp, \infty, \perp$
5: **for** $t, n_{4^2} \in P$ **do**
6: α_t = GETYAWANGLE(t)
7: **if** $|\alpha_t - \alpha_x| < r_{best}$ **then**
8: $t_{best}, r_{best}, n_{4^2_{best}} = t, |\alpha_t - \alpha_x|, n_{4^2}$
9: **end if**
10: **end for**
11: $center_x, fsize_x, roll_x$ = GETFACESIZEANGLE(x)
12: $center_t, fsize_t, roll_t$ = GETFACESIZEANGLE(t_{best})
13: $scale = fsize_t / fsize_x$
14: $roll = roll_t - roll_x$
15: $x_{aligned}$ = ALIGN($x, center_x, center_t, scale, roll$)
16: **return** $x_{aligned}, n_{4^2_{best}}$
17: **end function**
18: **function** GETYAWANGLE(image x)
19: lm_x = LANDMARKS(x)
20: $lwidth_x$ = GETLEFTWIDTH(lm_x)
21: $rwidth_x$ = GETRIGHTWIDTH(lm_x)
22: $\alpha = lwidth_x / rwidth_x$
23: **return** α
24: **end function**

Algorithm 2 Embed real images into $\mathcal{W}^+$ space (step 1.2)

1: **function** EMBED(image x, noise n, epoch e, lr η)
2: $\mathbf{w}_0 = \overline{\mathbf{w}}$ ▷ Init $\mathbf{w}$ as the average value
3: **for** $i \in \{0, \ldots, e-1\}$ **do**
4: img_i = SYNTHESIZE($\mathbf{w}_i, n$) ▷ *i.e.*, $G(w, n)$
5: ℓ_i = LPIPS(img_i, x)
6: $\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \nabla_{\mathbf{w}} \ell_i$
7: **end for**
8: **return** $\mathbf{w}_e, n$
9: **end function**

rotation angle to match the roll angles. Line 15 aligns x and line 16 returns the aligned image $x_{aligned}$ and the selected noise $n_{4^2_{best}}$. The code snippet of ALIGN can be found in the appendix.

Then we use the optimization-based embedding method to obtain the aligned image $x_{aligned}$'s latent vector $\mathbf{w} = \text{argmin}_{\mathbf{w} \in \mathcal{W}^+} L(G(\mathbf{w}, \{n_{4^2_{best}}, n_{*8^2}, \ldots\}), x_{aligned})$. Algorithm 2 describes how we embed one image into the latent space. It supports embedding a batch of images naturally because of the vectorization. Line 2 initializes $\mathbf{w}$ using the average latent vector. Lines 3-6 iteratively update $\mathbf{w}$ according to the LPIPS loss between the generated image and the real image. At the end of step ①, we will have a set of pairs of latent vectors and noises: $\{\langle \mathbf{w}_0, \mathbf{n}_0 \rangle, \ldots, \langle \mathbf{w}_k, \mathbf{n}_k \rangle\}$.

B. Constraining Natural and Consistent Changes

This subsection elaborates how we achieve the practical applicability (*i.e.*, the second requirement) and stealthiness (*i.e.*, the third requirement) by bounding the natural style changes in the latent space, and the consistent style changes (*i.e.*, the fourth requirement) by finding one style change for images of different poses.

1) L_2 regularization cannot provide natural style changes: The second and third requirements can be accomplished by confining the optimization within the vicinity of the starting point in the latent space. It is similar to how existing pixel-based adversarial examples (*e.g.*, PGD [9], CW [10]) achieve "invisibility" by bounding or reducing the perturbations w.r.t. some L_p norm distance from the start image. Here, a natural choice is to constrain the L_2 distance between the final vector $\mathbf{w}$ and the initial vector $\mathbf{w}_0$ to the original loss function (*i.e.*, $L(\mathbf{w}, t, \mathbf{w}_0) = L_{class}(G(\mathbf{w}), t) + L_2(\mathbf{w}, \mathbf{w}_0)$) as L_2 distance is widely used by many StyleGAN-based approaches [18], [16], [14].

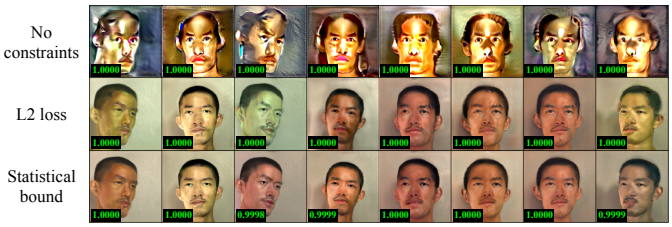
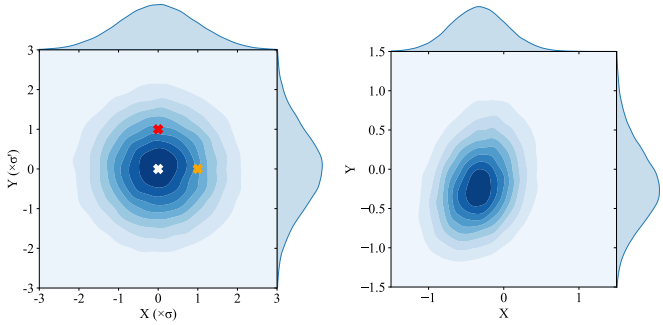


Fig. 11: Effects of different bounding strategies.



(a) A crafted MVG. (b) 2 rand. dim. in StyleGAN.

Fig. 12: Multi-variant Gaussian (MVG) distributions.

The results of using L_2 norm is shown in Figure 11. The first row uses no constraints at all (*i.e.*, the loss is $L(\mathbf{w}, t) = L_{class}(G(\mathbf{w}), t)$). We can see that though the confidence is very high, the images are not natural human faces. With L_2 loss, images in the second row have 100% attack success rate but have unnatural style changes (*e.g.*, green face in the third column).

This is because L_2 *distance ignores the different magnitudes of different dimensions.* Consider the following toy example. Assume $\mathbf{w} = [x, y]$ contains two elements $x \sim \mathcal{N}(0, 100^2)$ and $y \sim \mathcal{N}(0, 1)$ and its distribution is visualized in Figure 12a. Take a look at $\mathbf{w}_0 = [0, 0]$ (white cross), $\mathbf{w}_1 = [0, 1]$ (red cross), and $\mathbf{w}_2 = [100, 0]$ (orange cross). A simple calculation can tell us $L_2(\mathbf{w}_0, \mathbf{w}_1) = 10 * L_2(\mathbf{w}_0, \mathbf{w}_2)$. But Figure 12a shows they are "visually (or proportionally) the same" close to $\mathbf{w}_0$. Indeed, this is not counter-intuitive because 100 in the first dimension (with std 100) produces a similar deviation as 1 in the second dimension (with std 1). L_2 distance makes more sense in the space where each dimension has a similar magnitude such as the image pixel space (each pixel $\in [0, 1]$). However, different dimensions in the $\mathcal{W}^+$ space have different magnitudes [26], [22], therefore we need a better constraint that considers this factor.

Fig. 13: Effects of different strategies to find the changes.

2) Bounded natural style changes provide practical applicability and stealthiness: In fact, researchers have found that revert the last activation function $\phi = \text{LeakyReLU}(0.2)$[5] of the mapping network F can produce a multi-variate Gaussian space [22], [26]. Figure 12b visualizes the two bi-variate distributions of randomly selected two pairs of dimensions. As reflected in the previous toy example, the extent of variation should be measured w.r.t the variance.

Specifically, the i-th dimension of $\phi^{-1}(\mathcal{W}^+)$ can be modeled as $\mathcal{N}(\boldsymbol{\mu}[i], \boldsymbol{\sigma}[i]^2)$. In order to constrain a similar degree of variation in each dimension, we statistically bound the change in each dimension by $\mathbf{c}[i] \cdot \sigma[i]$. The last row in Figure 11 shows the corresponding result. Compared with the third row, the style changes are more natural and the essential features are better preserved.

In practice, we pre-sample 100k $\mathbf{w}$'s and compute the std $\boldsymbol{\sigma}[i]$ for i-th dimension of $\phi^{-1}(\mathcal{W}^+)$. We define a function called $\text{StatBound}(\mathbf{d}, \boldsymbol{\sigma}, \mathbf{c})$ to clip each dimension of the change direction $\mathbf{d}$. Formally, it changes i-th dimension of $\mathbf{d}$ as follows:

$$\mathbf{d}[i] = \phi(\max(\min(\phi^{-1}(\mathbf{d}[i]), \mathbf{c}[i] \cdot \boldsymbol{\sigma}[i]), -\mathbf{c}[i] \cdot \boldsymbol{\sigma}[i])) \ . \quad (1)$$

3) One modification direction for images with different poses provide the consistent change: Finally, we produce the same style changes for images with different poses. If we individually optimize the latent vectors in the batch as in the previous section, we usually get different style changes for different vectors, as shown by the first row of Figure 13 (especially the second image and the last image). Many researchers believe that there are different semantic directions in the latent space and the same direction should result in the same style changes [27], [28]. Hence, we choose to optimize one style changing direction $\mathbf{d}$ for all the latent vectors in the batch. In particular, we want to find a direction vector $\mathbf{d}$ such that $\forall \mathbf{w}$ in this batch, $G(\mathbf{w}_i + \mathbf{d})$ can be misclassified as the target person. The second row of Figure 13 shows now different poses have similar style changes. For simplicity, we use $\mathbf{W}$ to denote a batch of latent vectors $\mathbf{w}$'s and $\mathbf{W} + \mathbf{d}$ means adding the same $\mathbf{d}$ to each one in the batch (see the broadcasting semantics [29]).

C. White-box Attack

Gradients are available in this white-box setting, so we use the gradient-descent method to find the direction. Algorithm 3 formally describes our white-box attack procedure. It takes as input a batch of the attacker's images X with different poses,

[5] ϕ^{-1} = LeakyReLU(5) and LeakyReLU(p) = $\lambda x. \max(0, x) + p \cdot \min(0, x)$

Algorithm 3 White-box attack (step 2.2)

1: **function** ATTACK(images X, model M, target t, epoch e, lr η, std $\boldsymbol{\sigma}$, coef $\mathbf{c}$)
2: $X, N = \text{AUTOALIGN}(X)$
3: $W = \text{EMBED}(X, N)$
4: $d_1, d_{best}, \ell_{best} = \mathbf{0}, \mathbf{0}, \infty$
5: **for** $i \in \{1, \ldots, e\}$ **do**
6: $img_i = \text{SYNTHESIZE}(W + d_i, N)$
7: $\ell_i = \text{LOSS}(M(img_i), t)$
8: **if** $\ell_i < \ell_{best}$ **then**
9: $d_{best}, \ell_{best} = d_i, \ell_i$
10: **end if**
11: $d_{i+1} = d_i - \eta \nabla_d \ell_i$
12: $d_{i+1} = \text{STATBOUND}(d_{i+1}, \boldsymbol{\sigma}, \mathbf{c})$ ▷ Equation (1)
13: **end for**
14: **return** $\text{SYNTHESIZE}(W + d_{best}, N)$
15: **end function**

the subject model M, the target label t of the subject classifier (or the feature of the target image returned by the subject verifier), the maximal epoch e, the learning rate η, the pre-computed stds $\boldsymbol{\sigma}$, and the bound coefficient c. Line 2 aligns the batch of images and records the corresponding 4^2 noises (step (1.1) in Figure 5). Line 3 embeds the images into the latent space (step (1.2)). Line 4 initializes the direction and variables used to store the best direction and loss. Lines 5-13 conduct the attack loop for e times (step (2.1)-(2.3)). Line 6 synthesizes the images with the style changes (step (2.1)). Line 7 feeds the generated images into the subject model and computes the corresponding loss (step (2.2)). Lines 8-10 update the best direction and loss if needed. Line 11 updates the direction based on the learning rate and the gradients. Line 12 calls STATBOUND to constrain the style changes as explained in the previous subsection (step (2.3)). Line 14 generates and returns the images using the best direction.

D. Black-box Attack

To black-box attack the subject verifier, we apply Algorithm 3 on the surrogate verifier to generate transferable adversarial examples. To black-box attack the subject classifier, since we have no access to the gradients, we use the search algorithm to increase the confidence of the target label. Here, we use the genetic algorithm as an example as illustrated by Algorithm 4. The black-box attack function's parameters are similar to the white-box one, except that it does not need the learning rate but needs the size n of each generation and the confidence threshold f for an early stop. Lines 2-3 are the same with the white-box attack (step (1.1)-(1.2)). Line 4 initializes the first generation. Different from initializing the direction as 0 in the white-box setting, here we use random values (sampled from a uniform/normal distribution) as the direction. The intuition behind is to diversify the first generation so that we have better chances to produce better offsprings. Lines 5-20 form the main attack loop (step (2.1)-(2.3)) and are executed for at most e generations. Line 6 computes the score for each direction in the current generation (step (2.2)). A score is a float number denoting the average confidence of the target label over the batch of latent vectors: $\mathbb{E}_{\mathbf{w} \in batch}[M(G(\mathbf{w} + \mathbf{d}, N))[t]]$. Line 7 finds the direction

Algorithm 4 Black-box attack on classifiers (step 2.2')

1: **function** ATTACK(images X, model M, target t, size n, epoch e, conf f, std $\boldsymbol{\sigma}$, coef $\mathbf{c}$)
2: $X, N = \text{AUTOALIGN}(X)$
3: $W = \text{EMBED}(X, N)$
4: $gen_1 = \text{GETFIRSTGEN}(n)$
5: **for** $i \in \{1, \ldots, e\}$ **do**
6: $scores_i = \text{GETCONFIDENCE}(M, W, N, t, gen_i)$
7: $best_i = \text{FINDBEST}(gen_i, scores_i)$
8: **if** $scores_i[best_i] \geqslant f$ **then**
9: **return** $\text{SYNTHESIZE}(W + best_i, N)$
10: **end if**
11: $gen_{i+1} = \{best_i\}$
12: **for** $j \in \{1, \ldots, n-1\}$ **do**
13: $parent_1 = \text{WEIGHTEDSAMPLE}(gen_i, scores_i)$
14: $parent_2 = \text{WEIGHTEDSAMPLE}(gen_i, scores_i)$
15: $child_j = \text{CROSSOVER}(parent_1, parent_2)$
16: $child_j = \text{MUTATE}(child_j)$
17: $child_j = \text{STATBOUND}(child_j, \boldsymbol{\sigma}, \mathbf{c})$
18: $gen_{i+1} = gen_{i+1} \cup \{child_i\}$
19: **end for**
20: **end for**
21: $d_{best} = \text{FINDBEST}(gen_e, scores_e)$
22: **return** $\text{SYNTHESIZE}(W + d_{best}, N)$
23: **end function**

with the highest average confidence in the current generation. Lines 8-10 utilize the early stop strategy: if the best direction has a score larger than the desired confidence, we use this direction to generate the final images. Lines 11-19 build the next generation. Line 11 preserves the current best direction into the next generation based on the elitism strategy. Lines 12-19 produce the remaining $n-1$ offsprings. Parents with healthier genes are more likely to have healthier children. Lines 13-14 use the weighted random sampling method to select parents. The weight is the score. This gives higher chances to samples in current generations with higher scores, but also allows for the other samples. Line 15 also uses the weighted random sampling method to let the child inherit each gene (*i.e.*, the dimension) from one of the parents. To include more diversity and enhance the search capability, Line 16 randomly mutates some genes of the child. Line 17 constrains the changes using STATBOUND (step ②.③). Line 18 adds the child to the next generation. Lines 21-22 find the best direction and use it to generate the batch of images.[6]

V. EVALUATION

We implement our approach in PyTorch [30] and open-source it[7]. We evaluate it on various datasets and models in both the white-box and black-box settings. Besides showing the results in the digital space, we also show that we can succeed in physically attacking the online commercial face recognition service. Furthermore, we test our approach against existing defense methods and show our attack can still break them. Our evaluation server has Intel Xeon Silver 4214 2.20GHz 12-core CPUs with 256 GB RAM and NVIDIA Quadro RTX 6000 GPUs.

A. Experimental Setup

1) Datasets and Models: We select three most widely used large-scale face classification datasets: VGGFace, VGGFace2, and CASIA. VGGFace has about 2.6M images of 2.6K identities, VGGFace2 contains about 3.3M images of 9.1K identities, and CASIA has about 0.5M images of 10.6K identities. For each dataset, we download two pre-trained models with different architectures from their official websites or GitHub repositories. More specifically, the models we used are VGG16 and VGG16BN on the VGGFace dataset, ResNet-50 and InceptionV1 on the VGGFace2 dataset, and SphereFace and InceptionV1 on the CASIA dataset. For verification models, we download four pre-trained models: IRSE50/IR152/Facenet/MobileFace. We also evaluate our black-box approach on two commercial face recognition services Clarifai [5] and Face++ [31]. We randomly select 8 identities from the VGGFace2 dataset and upload their images to Clarifai to train a classifier to attack. Face++ provides verification APIs to predict the similarity of two images. The unaligned StyleGAN we use is trained by the GenForce group [32] on the 104K unaligned images from CelebA dataset.

2) Attackers and Target Identities: We use 12 identities from the Pose dataset [33] as the attackers. Pose dataset contains 12 identities and 20 images with different poses for each identity. For the target label of the subject classifier, we select each of the top-20 predicted identities for the given attacker.[8] To attack verifiers, we *randomly* select 10 face images. Effects of the label ranks and skin colors are studied in Section B and Section I in the appendix.

3) Baselines: For classifiers, we compare with SemAdv [19] and AdvGlass [11] introduced earlier. We also propose one additional baseline Inp. Since the interpolation between latent vectors is able to produce an image similar to the two end points, we can first use model inversion [22] to get a representative image (and the corresponding latent vector $\mathbf{w}_t$) of the target label t and then find a point on the segment defined by the attacker's latent vector $\mathbf{w}_a$ and $\mathbf{w}_t$ to generate the adversarial image. Formally, Inp finds a coefficient $b = \text{argmin}_b L(M(G(b * \mathbf{w}_a + (1-b) * \mathbf{w}_t)), t)$. For verifiers, we compare with AdvMakeup [20].

B. Research Questions

Our evaluation consists of experiments in both the cyber space and the real world. Specifically, we study the following major research questions:

1) Can our white-box attack succeed in the cyber space?
2) Can our black-box attack succeed in the cyber space?
3) Are the changes natural and consistent styles?
4) Can our physical attack succeed in the real world?

[6]In our evaluation, we set the maximum iteration number as 100. As a result, the proposed attack requires fewer than 100K queries for 1000 samples. These numbers are consistent with the literature [21], [22].

[7]Our code: https://github.com/njuaplusplus/imu

[8]If the target person is ranked very low for the attacker, it may be infeasible to have stealthy and natural style changes for physical attacks.

In order to answer RQ1 and RQ2, we utilize the following two metrics as traditional attack studies.

Attack Effectiveness. This is also called Attack Success Rate (ASR). It measures the percentage of the generated images that can be misclassified by the subject model as the target labels.

Attack Generalizability (Transferable ASR). Better adversarial examples usually have higher generalizability. On each dataset, two pre-trained models with different architectures are used. We generate the adversarial examples on one model and report the ASR on the other model.

For both metrics on classifiers, we report top-1 ASR as well as top-5 ASR. If the target label is included in the top-5 predicted labels with the highest confidence, we count it as a successful top-5 attack. We also rescale the generated adversarial images and add random noises to simulate the noisy environment in the real world. For verifiers, we only report the ASR.

To answer RQ3, we use Natural Image Quality Evaluator (NIQE), Total variance (TV) and anonymous human studies.

Image Quality. We use NIQE+TV to measure image quality. NIQE evaluates the non-reference image quality score for a given image. It computes the deviation of the image features from the statistical features derived from the natural images. A smaller NIQE score implies better perceptual quality. TV measures how neighboring pixels change. An image with a smaller TV is less noisy.

Anonymous Human Study. We conduct two types of human studies. The first study's goal is to prove that we add changes to the attacker's face instead of generating the target person's face. We show one generated adversarial face to the user and ask the user to select if this face has the same identity as the attacker or the target. The second study's goal is to verify whether our approach can generate consistent style changes for different poses. We show two adversarial frontal faces of one attacker with different styles and another side face with either style. We ask the user to match the correct style. Figure 26 and Figure 27 show the question examples. In total, 100 users participated in this study and each of them was asked 6 questions.

RQ4 directly measures whether our ultimate goal can be achieved. We collect a set of photos with different poses from the attackers and use our black-box attack method to attack the commercial service. We then ask attackers to apply the style changes and re-take a set of photos. We upload those photos to the commercial service to see if they are misrecognized as the target.

C. (RQ1) White-box Attack Results

The results of white-box attack methods on classifiers are shown in Figure 14. Figure 14a shows the top-1/top-5 ASR of four methods on the subject models. The bars marked with slashes denote the vanilla ASR. Our method achieves high ASR for all cases. Use the leftmost section (VGG16 VGGFace) as an example. The first four bars marked with slashes mean the top-1 ASR of Inp/AdvGlass/SemAdv/Ours is 35%/99.38%/99.71%/100% when they generate adversarial images on VGG16 and test them on VGG16. Additionally, since the real-world environment is usually noisy (e.g., different viewing distances and the dust on the lens), We hence simulate these negative factors by cropping and rescaling the images and adding random noises. The results (+simu) are reflected by the solid bars in the figure. Observe that our method is less affected by these noises, and outperforms the competitors. Still use (VGG16 VGGFace) as an example. The decrease of SemAdv's ASR is about 20% while ours is only about 2%.

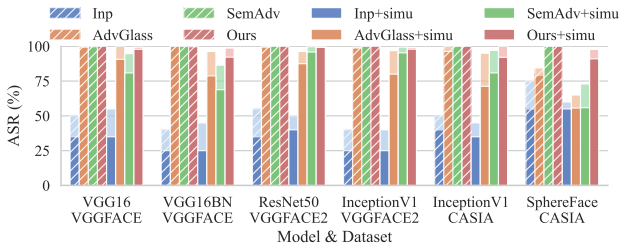
Figure 14b shows the top-1/top-5 ASR on the reference models and can be interpreted similarly. Taking the leftmost section as an example, where the attack is conducted on VGG16 but the images are tested on VGG16BN. Our approach has the highest transferable ASR with the simulated noises on all the models and SemAdv's ASR decreased by more than half on 50% of the models. We also observe that SemAdv has high ASR and transferable ASR for the vanilla case (*i.e.*, bars marked with slashes) but degraded significantly in the noisy cases. The reason is that the adversarial noises of SemAdv are actually unbounded and pervasive (*i.e.*, covering the whole image). When the perturbation is unbounded and pervasive, the adversarial examples can have reasonably high transferable ASR. However, simulating the real-world environment impairs the adversarial noise much more than the adversarial styles. Figure 15a shows the results on verifiers. Similar results are observed, i.e., our approach has much higher ASR and transferable ASR than the baseline. Hence, we have the positive answer for RQ1.

D. (RQ2) Black-box Attack Results

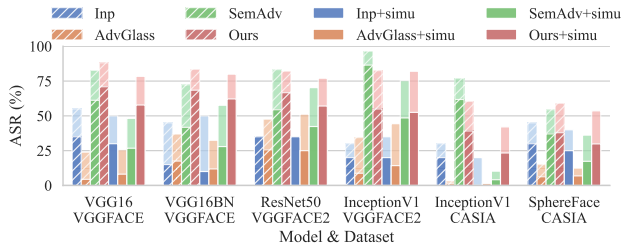
Figure 16 shows the results of our black-box attack and AdvGlass on classifiers. From Figure 16a, we can see our attack significantly outperforms AdvGlass and is quite effective (90+% ASR) on all the subject models except one. With the simulation, our performance is still reasonably high. The generalizability of our attack is low yet still much higher than that of AdvGlass. Figure 15b shows the results on verifiers. Note that the black-box attack against one verifier is based on white-box attacking the other three verifiers, so we only report ASR here. Our approach also outperforms the baseline on verifiers. Thus, as for RQ2, the answer is yes.

E. (RQ3) Naturalness and Consistency

Figure 17 shows the average NIQE+TV scores for the adversarial images generated by different attack methods. The larger score means the worse quality. AdvGlass uses local noises, so its images have a large score. SemAdv uses pervasive noises (yet less intensive than AdvGlass) and thus its score is smaller than AdvGlass but larger than ours. AdvMakeup patches the orbital area with artifacts and hence also has a large score. Our method has the best quality because of the natural style changes. For the human study results, all users consider our style-changed images are from the attackers and about 95%

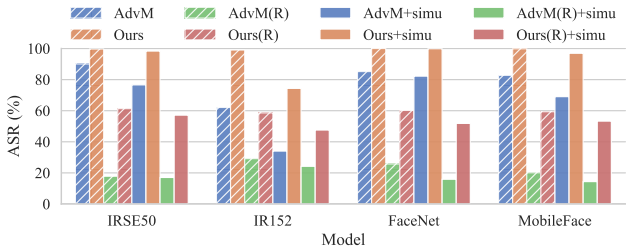


(a) Top-1/top-5 (dark/light color) ASR on subject models.

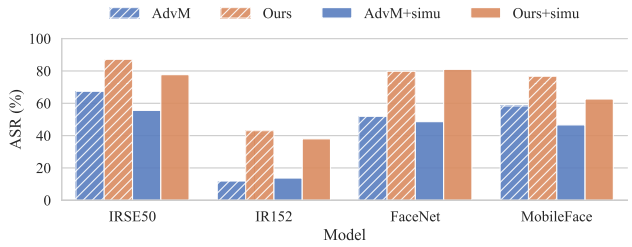


(b) Top-1/top-5 (dark/light color) ASR on reference models.

Fig. 14: White-box attack effectiveness and generalizability on classifiers. The higher the better.

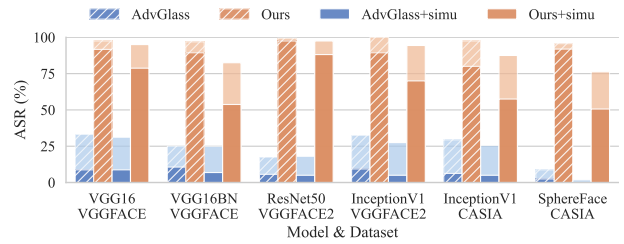


(a) White-box ASR on subject and reference (denoted by R) models.

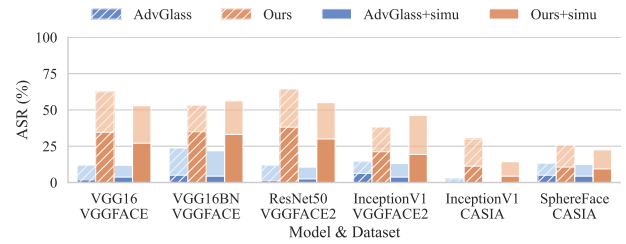


(b) Black-box ASR on subject models.

Fig. 15: Attack effectiveness and generalizability on verifiers. The higher the better.



(a) Top-1/top-5 (dark/light color) ASR on subject models.



(b) Top-1/top-5 (dark/light color) ASR on reference models.

Fig. 16: Black-box attack effectiveness and generalizability on classifiers. The higher the better.

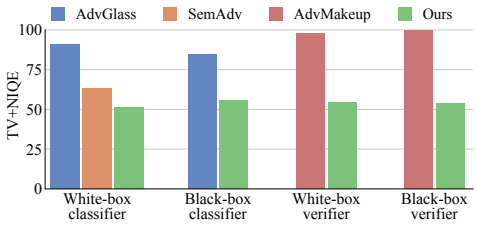


Fig. 17: Image quality score. The lower the better.

users can recognize the corresponding styles of different poses. Therefore, the answer to RQ3 is positive.

F. (RQ4) Commercial Service Attack Results

Figure 18 shows the attack result on the commercial service Clarifai. We randomly upload 8 identities with about 2k images to Clarifai and train a classifier. We used our black-box approach to attack a non-top-1 label whose images are shown in the rightmost column. Taking Figure 18 as an example, the confidence scores of the target label on the attacker's original images are indicated by the numbers in the first row and are very low. The digital results generated by our approach are shown in the second row. The main changes are the pink/purple face color, the shorter and arched eyebrows (thinner right eyebrow), the pink lipstick and different contours and highlights. The photos retaken after applying the style changes are shown in the last row of Figure 18. We can see that the confidence is indeed very high. The style changes are faithfully realized. Note that the slight color difference between the digital images and the physically re-captured images is unavoidable because of the cosmetic and camera. This experiment gives the positive answer to RQ4. That is, our approach can physically attack the commercial API in the real world. Please see Figure 31 in the appendix for another example of attacking Clarifai. The comparison of our approach and the baseline on Face++ is in Figure 23 in the appendix.

G. Effects of Defenses

Adversarial training is the most widely-used defensive method against adversarial examples. We evaluate our attack against three existing adversarial training methods (FastAdv [34], GAT [35], and AdvMix [36]) and the adaptive defense. FastAdv is an efficient and effective implementation of the traditional adversarial training based on the modified Fast Gradient Sign Method (FGSM) adversary in the input image (pixel) space. GAT and AdvMix are StyleGAN-based strategies. GAT's adversarial training examples are generated by finding the misleading styles and noises in StyleGAN. AdvMix generates adversarial training examples by randomly

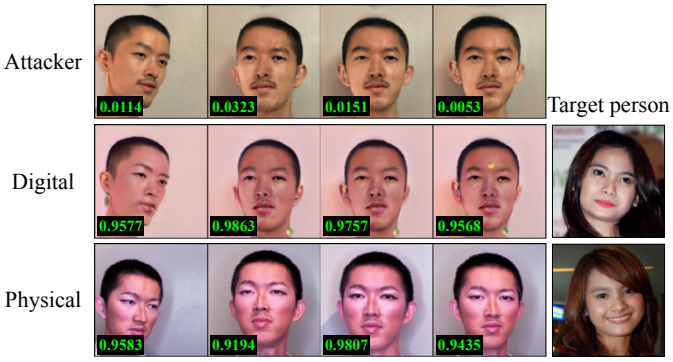


Fig. 18: Clarifai physical attack results.

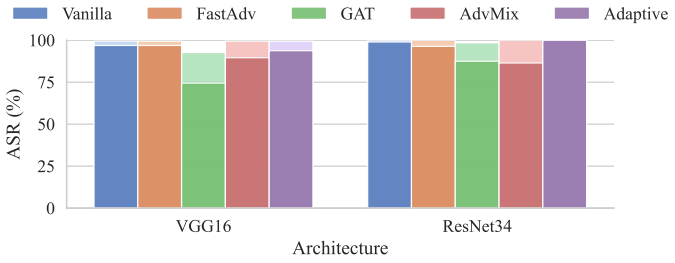


Fig. 19: Attack performance on VGG16/ResNet34 networks adversarially trained with different strategies.

replacing the styles in high-resolution style blocks and finding the most misleading (largest CE loss) examples. The above three adversarial training strategies have no knowledge of our attack, so we use another more knowledgeable defensive strategy that knows our attack algorithm and thus is known as the adaptive defense. Specifically, the defender uses the exact same algorithm to generate adversarial training examples. We train models without defense or with one of the four above-mentioned defense strategies using 8 people randomly selected from the VGGFace2 dataset.

Figure 19 shows our attack performance on VGG16 and ResNet34 trained with different strategies. FastAdv is not effective as expected because it only robustifies the models against adversarial noises which are completely different from the style changes. Though GAT and AdvMix slightly decreased the attack effectiveness, our attack still has a reasonably high ASR (*e.g.*, 88/86% on VGG16 with GAT/AdvMix and 74/90% on ResNet34). It is not surprising that the adaptive defense cannot prevent our attack, because attackers are outsider people and they cannot be completely covered. In summary, our attack can bypass existing adversarial training methods as well as the adaptive defense.

VI. Related Work

Adversarial Attacks. Many attack approaches in the cyber space have been proposed to fool the neural networks [9], [19], [37], [10], [21]. Besides the cyber space, many researchers extend the attack into the physical world [11], [20], [38], [39], [12]. Most of them are based on optimized adversarial noises and thus neither natural nor stealthy for human faces. The most relevant one in our settings, is AdvGlass [11] which optimizes unbounded adversarial noises within the eyeglass frame area. However, it's less stealthy and effective compared to our approach.

Adversarial Defenses. To mitigate the threat brought by the adversarial attacks, many defenses have been developed. The most widely used method is called adversarial training [9]. The basic idea is to use certain attack methods to generate adversarial examples and add them into the training data to robustify the models. FastAdv [34] uses traditional digital pixel attack to generate adversarial examples, and thus is ineffective against our style-based attack. GAT [35] optimizes the styles and noises in StyleGAN to generate adversarial examples. AdvMix [36] first embeds benign training data into $\mathcal{W}^+$ and generates adversarial examples by randomly perturbing styles at certain style blocks. GAT and AdvMix can robustify simple models, but fails in larger ones.

GANs and StyleGANs. GAN was proposed to learn a mapping from one distribution (*e.g.*, Gaussian distribution) to another distribution (*e.g.*, text and images) [40]. A lot of GANs with different architectures and training strategies have been devised to generate better images with higher resolutions [18], [16], [41], [42]. Recently, some 3D-aware StyleGANs have been proposed [43]. However, it's much more difficult to embed real images compared to 2D StyleGANs [44]. They are not suitable for the physical attack, because attackers needs to embed their real images.

VII. Conclusion

In this paper, we propose a novel physical impersonating attack based on StyleGAN. Given the same person's multiple images with different poses, we align and embed all the images into a latent space, in which we search for the same statistically bounded style change that is physically applicable, stealthy, and consistently effective across all the images. We conduct both digital and physical experiments showing that the proposed attack achieves high attack success rate and can be physically executed by applying makeup on the attacker.

VIII. Ethical Considerations

We notified Clarifai and Face++ about this attack and got their approval for the tests. We use Amazon Mechanical Turk to outsource the user studies and have the needed exemption from our IRB office.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. This research was supported, in part by IARPA TrojAI W911NF-19-S-0012, NSF 1901242 and 1910300, ONR N000141712045, N000141410468 and N000141712947. Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of our sponsors. We thank Craig, Courtney, Lucy, Mingwei, Priscilla, Qingkai, Shiwei, Vivian, Xiangzhe, Zhou for their great help in the experiments.

REFERENCES

[1] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," 2015.

[2] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *2018 13th IEEE international conference on automatic face & gesture recognition (FG)*. IEEE, 2018, pp. 67–74.

[3] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[4] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 212–220.

[5] "Clarifai," https://www.clarifai.com/.

[6] "Azure cognitive services," https://azure.microsoft.com/en-us/services/cognitive-services/.

[7] "Amazon rekognition," https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html.

[8] "Google cloud vision," https://cloud.google.com/vision/.

[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.

[10] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017. [Online]. Available: http://dx.doi.org/10.1109/SP.2017.49

[11] 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 *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1528–1540. [Online]. Available: https://doi.org/10.1145/2976749.2978392

[12] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1625–1634.

[13] "Visa photo requirements," https://travel.state.gov/content/travel/en/us-visas/visa-information-resources/photos.html.

[14] E. Richardson, Y. Alaluf, O. Patashnik, Y. Nitzan, Y. Azar, S. Shapiro, and D. Cohen-Or, "Encoding in style: a stylegan encoder for image-to-image translation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[15] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, and D. Cohen-Or, "Designing an encoder for stylegan image manipulation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–14, 2021.

[16] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8110–8119.

[17] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[18] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4401–4410.

[19] H. Qiu, C. Xiao, L. Yang, X. Yan, H. Lee, and B. Li, "Semanticadv: Generating adversarial examples via attribute-conditioned image editing," in *ECCV*, 2020.

[20] B. Yin, W. Wang, T. Yao, J. Guo, Z. Kong, S. Ding, J. Li, and C. Liu, "Adv-makeup: A new imperceptible and transferable attack on face recognition," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, 8 2021, pp. 1252–1258.

[21] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1277–1294.

[22] S. An, G. Tao, Q. Xu, Y. Liu, G. Shen, Y. Yao, J. Xu, and X. Zhang, "Mirror: Model inversion for deep learning network with high fidelity," in *Proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, 2022.

[23] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[24] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1501–1510.

[25] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 586–595.

[26] P. Zhu, R. Abdal, Y. Qin, J. Femiani, and P. Wonka, "Improved stylegan embedding: Where are the good latents?" 2020.

[27] Y. Shen, J. Gu, X. Tang, and B. Zhou, "Interpreting the latent space of gans for semantic face editing," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. [Online]. Available: http://dx.doi.org/10.1109/CVPR42600.2020.00926

[28] R. Abdal, P. Zhu, J. Femiani, N. Mitra, and P. Wonka, "Clip2stylegan: Unsupervised extraction of stylegan edit directions," *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*, Aug 2022. [Online]. Available: http://dx.doi.org/10.1145/3528233.3530747

[29] "Broadcasting semantics," https://pytorch.org/docs/stable/notes/broadcasting.html.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32 (NeurIPS)*. Curran Associates, Inc., 2019, pp. 8024–8035.

[31] "Face++," https://www.faceplusplus.com/.

[32] Y. Shen, Y. Xu, C. Yang, J. Zhu, and B. Zhou, "Genforce," https://github.com/genforce/genforce, 2020.

[33] "Pose dataset," http://posedataset.site/.

[34] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *International Conference on Learning Representations (ICLR)*, 2020.

[35] O. Poursaeed, T. Jiang, H. Yang, S. Belongie, and S.-N. Lim, "Robustness and generalization via generative adversarial training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 711–15 720.

[36] S. Gowal, C. Qin, P.-S. Huang, T. Cemgil, K. Dvijotham, T. Mann, and P. Kohli, "Achieving robustness in the wild via adversarial mixing with disentangled representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1211–1220.

[37] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014.

[38] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 52–68.

[39] K. Yang, T. Tsai, H. Yu, T.-Y. Ho, and Y. Jin, "Beyond digital domain: Fooling deep learning based recognition system in physical world," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1088–1095.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[41] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *International Conference on Learning Representations (ICLR)*, 2018.

[42] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 852–863. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/076ccd93ad68be51f23707988e934906-Paper.pdf

[43] J. Gu, L. Liu, P. Wang, and C. Theobalt, "Stylenerf: A style-based 3d aware generator for high-resolution image synthesis," in *International Conference on Learning Representations (ICLR)*, 2022.

[44] "Issues on stylenerf inversion," https://github.com/facebookresearch/StyleNeRF/issues/14#issuecomment-1075516878.

[45] "Advgan github repository," https://github.com/sarathknv/adversarial-examples-pytorch/tree/master/adv_gan.
[46] C. Xiao, B. Li, J.-y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Jul 2018. [Online]. Available: http://dx.doi.org/10.24963/ijcai.2018/543
[47] P. Stock and M. Cisse, "Convnets and imagenet beyond accuracy: Understanding mistakes and uncovering biases," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 498–512.
[48] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov, "Differential privacy has disparate impact on model accuracy," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.
[49] S. Yucer, F. Tektas, N. Al Moubayed, and T. P. Breckon, "Measuring hidden bias within face recognition via racial phenotypes," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2022, pp. 995–1004.

APPENDIX

A. EFFECTS OF RANDOM NOISES

Figure 20 (left) shows the effects of random noises in a StyleGAN trained on the aligned FFHQ 256^2 dataset. All the images use the same latent vector. Each row shows the effects of the random noise at a different style block denoted by the left label (*i.e.*, $4^2 - 256^2$). In each row, we randomly sample 8 different noises and inject them to generate 8 images (3 of which are shown to save space). The right most column shows the magnified absolute difference. We can see that the random noises only bring small variations.

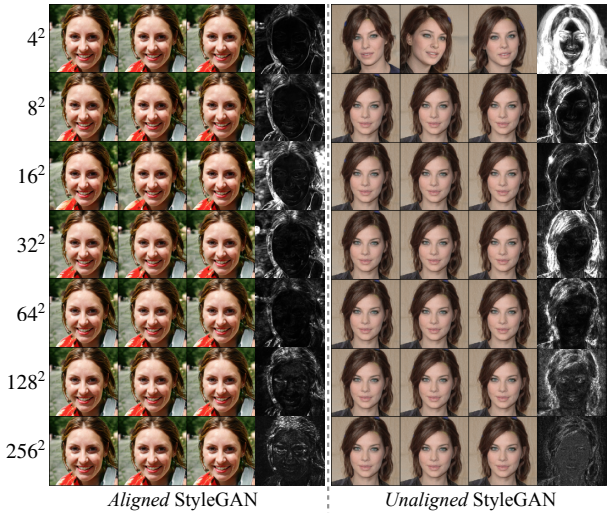


Fig. 20: Effects of stochastic noises at the different style blocks of *aligned* (left) and *unaligned* (right) StyleGAN. We only show 3 images here to save space. The rightmost column in each part shows the magnified absolute difference (computed on 8 images).

Figure 20 (right) shows the effects of random noises in a StyleGAN trained on the unaligned CelebA 256^2 dataset. All the images use the same latent vector. Each row shows the effects of the random noise at a different style block denoted by the left label (*i.e.*, $4^2 - 256^2$). In each row, we randomly sample 8 different noises and inject them to generate 8 images. The right-most column shows the magnified absolute difference. We can see that the random noise at the 4^2 style block actually produces pose variation while noises at other style blocks only bring small variations.

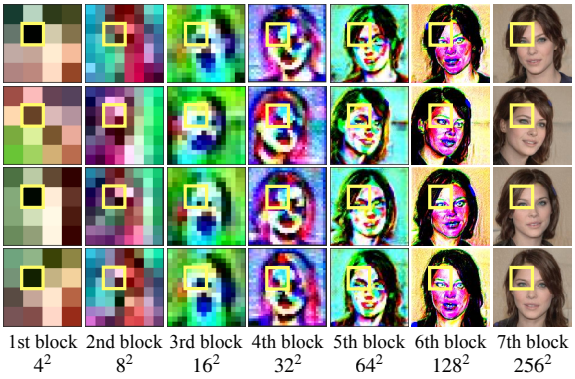


Fig. 21: Feature maps in each style block of an *unaligned* StyleGAN. Only the noises at 4^2 block are randomized. The yellow boxes denote the corresponding areas at different blocks. 1st-6th blocks are enlarged for better visualization.

The special structure and training method of StyleGANs allow the different blocks to denote features of various granularities. Specifically, StyleGAN training first downsamples images to 4x4 and trains only the 4x4 block till it stabilizes. Then, it downsamples images to 8x8 and trains the 8x8 block. Note that at the 4x4 level, the downsampled images likely just denote coarse features like poses. We are the first to observe that for StyleGAN trained on an unaligned dataset, noises at the earlier blocks can affect poses. Here we only randomize the noises at the 4^2 style block. We visualize the output (feature maps) of each style block in Figure 21. The width and height of the 4^2 style block's output are 4. It is upsampled to 8^2 and fed to the 2nd block. This upsampling continues until we generate the final 256^2 output image. The 4^2 feature map can be considered as setting the coarsest structure of the final image. The highlighted yellow squares in different feature maps partially illustrate how the 4^2 noises affect later layers. For example, the square in the 4^2 block is upsampled across multiple layers to produce the square in the 256^2 block.

B. EFFECTS OF LABEL RANKS ON CLASSIFIERS

In our main experiments on classifiers, we attack top-20 labels. Here we attack different top-k lables to show that our approach can still perform well on other labels. To better display the results, we first define relative top-k labels by dividing the original rank by the number of all labels. For example, VGGFace has 2.6K labels and the top-26 labels are the relative top-0.01 labels. This definition of relative top-k labels helps display the average results on all models and datasets in one chart with the same x-axis as shown by Figure 22, because different datasets have different numbers of labels, (*e.g.*, 2.6K for VGGFace and 10.6K for CASIA) and cannot be well described by one x-axis. As expected, with the increasing of k, the ASR decreases because we have more different target people. From the intersection point of the two red dashed lines, we can see the average ASR are greater than 90% for the top-0.2 labels (*i.e.*, $>$ 2.1K labels on CASIA, 0.5K

on VGGFace, and 1.8K on VGGFace2). If we attack all the labels (i.e., similar to randomly selected targets), the average ASR is about 44.09%. AdvGlass's ASR is about 50.68%. It's slightly higher because AdvGlass uses adversarial noises while ours are natural style changes (with more constraints).

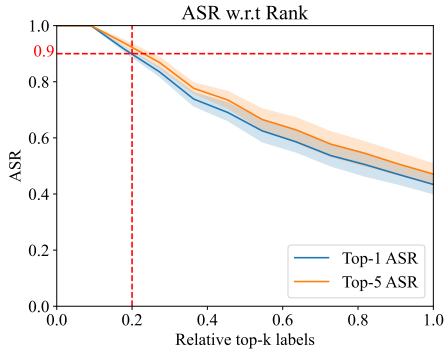


Fig. 22: Attack effectiveness on different top-k labels (aggregated on 6 models). The intersection point of the two red dashed lines show the ASR is about 90% for top-0.2 labels. On CASIA dataset, it means more than 2.1K labels.

C. Results on Face++

Face++ provides a face verification service. Given a pair of images, it outputs a similarity score. Figure 23 shows the comparison results of ours and AdvMakeup. The x-axis means the threshold t. If the similarity score of two images is larger than t, then they are considered as one person. The y-axis means the attack success rate. We can see our method outperforms Adv-Makeup.

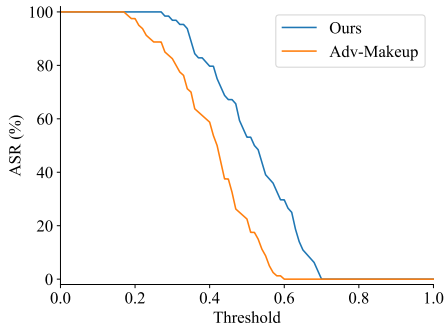


Fig. 23: Face++ verification results.

D. Effects of Auto-alignment and StatBound

Our default setting uses auto-alignment and a statistical bound. Its ASR is 93.75% and its TV+NIQE score is 49.17. If we remove the auto-alignment, the ASR is 93.75% but the TV+NIQE score increases by 9.93. A lower TV+NIQE score means better image quality. Therefore, auto-alignment can help generate images with better quality. If we replace the statistical bound with L2 bound, the ASR is 100% but the TV+NIQE score increases by 7.01. Therefore, the statistical bound also helps generate better style changes while maintaining a high ASR.

E. Aligned and Unaligned Training Data

Figure 24 shows the average images (left) and images randomly selected from the aligned and unaligned training datasets. We can observe: 1) The average image of the aligned dataset has clearer face features (eyes, nose, and mouth). This is because the faces are aligned to certain positions. 2) The images randomly selected from the aligned dataset indeed have more uniform poses than those from the unaligned dataset. Details of building the aligned CelebA-HQ dataset are in Appendix C of the PGGAN paper[41].

Fig. 24: *Aligned/unaligned* (top/bottom) training data.

F. Comparison with AdvGAN

AdvGAN's attack goal is to generate small adversarial noise-like perturbations for classifiers [45], [46]. Their white-box ASR(%) on VGG16 / VGG16BN / ResNet50 / InceptionV1(VGGFace2) / InceptionV1(CASIA) is 71.59/97.73/63.64/81.82/96.59 which is inferior to ours (close to 100% for all tested models).

G. Demographics of Pose Dataset and Human Studies

For the Pose dataset, the ages range from 20 to 40 and the races include Asian and Caucasian. There are 2 females and 10 males. For our human studies, the results are anonymously collected from Amazon Mechanical Turk. We do not find demographic information on Amazon website.

H. Physically Realizing the Makeup

Students in Cosmetology helped apply the makeup. We show them the original images, the generated images, and the differential images. They applied the makeup accordingly. Photos are then taken and new differential images are further generated to compare with the original differential images. Adjustments may be needed until the two sets of differential images align.

I. Effects of Different Skin Colors

To see the effect of skin colors on the ASR, we evaluate our attack on ResNet50 (classifier) pre-trained on VGGFace2 dataset in four attacker&target settings: 1) dark&random-dark 98.44%, 2) dark&random-light 53.98%, 3) light&random-dark 59.38%, and 4) light&random-light 62.5%. To get 8 random target people of a certain skin color (e.g., random-dark), we iteratively run `random.sample` and manually check the sampled people until we get 8 people of that certain skin color. Attacking IRSE50 (verifier) pre-trained on the LFW dataset

gives a higher ASR close to 100%. Figure 25 shows some examples.

The phenomenon that the networks/algorithms perform differently on well-represented and under-represented sub-distributions (e.g., light-skinned vs dark-skinned people) is not uncommon as pointed out by many researchers [47], [48], [49]. It's reported that only 5% of identities in VGGFace2 dataset have dark skin [49]. Therefore target people with dark skin are in the top-ranked labels of attackers with dark skin. Thus the dark&dark ASR of (98.44%) is very high (similar to the top-0.1 setting in Figure 22). For attackers with light skin, randomly selected targets with light skin have almost randomly ranked labels, and hence the ASR of light&light (62.5%) is only slightly higher than the light&dark (59.38%).

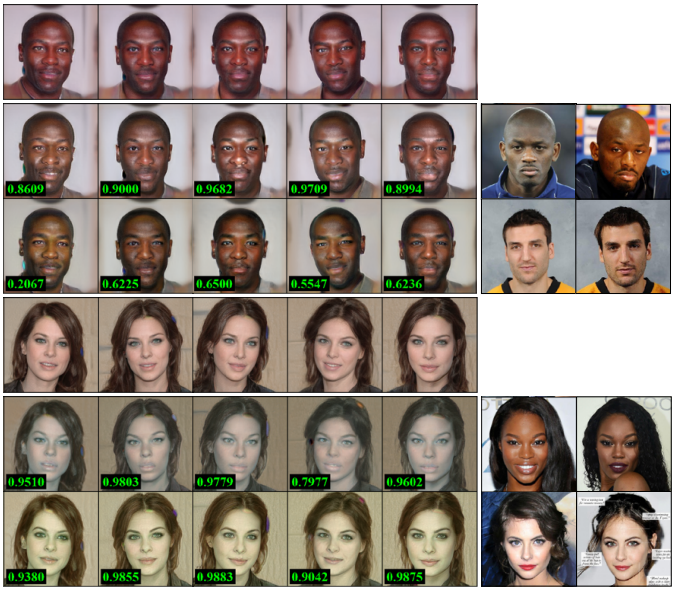


Fig. 25: Adversarial examples against different target people on ResNet50. Each row corresponds to one target person whose real images are shown on the right.

J. User Study Examples

Figure 26 shows an example of the first user study. It's goal is to check if we indeed add changes to the attacker's face or just generate the target face. In each question, we show the user the generated adversarial face in the first row and ask the user to select the same person from the attacker's face and the target person's face in the second row. 100% users can correctly select the attacker's face. This means our approach indeed generates the style change.

Figure 27 shows an example of the second user study. It's goal is to check if our approach can generate consistent style changes for multiple images of the same attacker with different poses. We show the user one reference image of the attacker at a certain pose with a certain makeup in the first row and the other two faces of the same person with the same pose but different from the reference one in the second row. Only one of the two faces in the second row has the same makeup with the reference image. We ask the user to select the face in

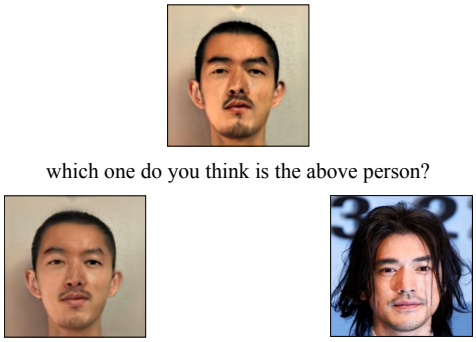


Fig. 26: Human study example

Fig. 27: Human study example

the second row that wears the same makeup as the reference image. About 95% users can pick the correct answer. This means our approach can generate consistent style changes.

Our human studies were anonymously collected from Amazon Mechanical Turk (MTurk). Each user was asked 6 questions which took $<$ 1 minute. We paid each user \$0.12.

K. More Attack Results

Figure 28 shows the adversarial examples generated by different methods for the same attacker and model as Figure 2 with different target. In Figure 2, the attacker wants to impersonate an actor and here the target person is an actress. Three real images of the target person are shown in the rightmost column. The first row shows the images of the attacker at different poses. Each number on each image shows the target confidence. The second and third rows show the initial and final images of SemAdv. The fourth row presents the results generated by AdvGlass. The fifth row shows our results. We can see that our images are more natural and less ostentatious. The last row shows the effect of the physical attack. Compared to AdvGlass, our physical adversarial images have higher ASR and target confidence.

Figure 29 shows the difference between the final images generated by different approaches and the initial images for Figure 2 and Figure 28. From the difference images, we can see SemAdv generates patterns like pervasive adversarial noises while AdvGlass generates local adversarial noise. Our approach generates smooth style changes.

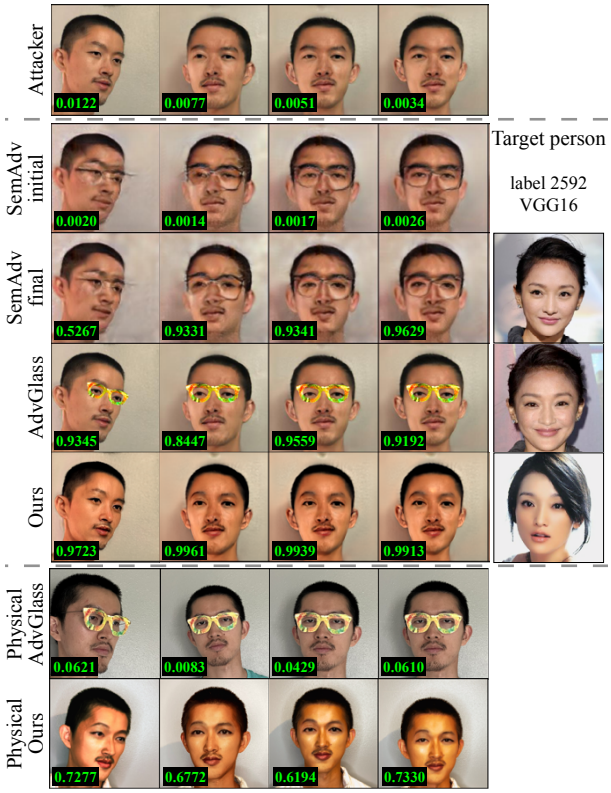


Fig. 28: Attacking label 2592 of VGG16.

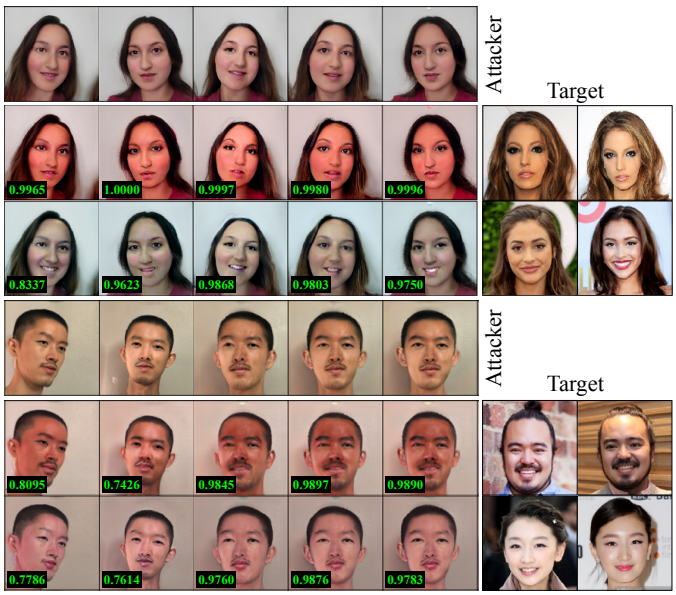


Fig. 30: More adversarial examples.

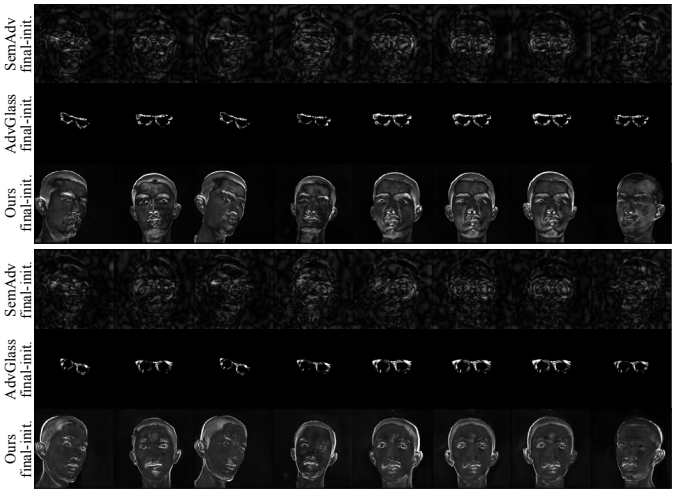


Fig. 29: Absolute difference between the final images and the initial images of label 2396 (top) and label 2592 (bottom).

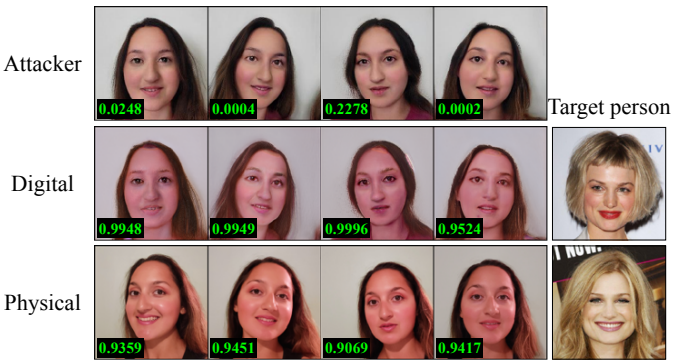


Fig. 31: Clarifai physical attack results.

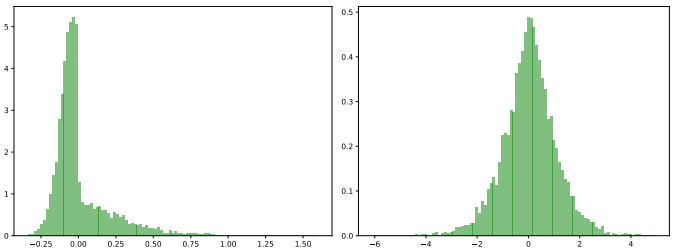


Fig. 32: Histograms of training $\mathcal{W}^+$ (left) and embedded $\mathcal{W}^+$ vectors.