FaceGuard: A Self-Supervised Defense Against Adversarial Face Images

Debayan Deb, Xiaoming Liu, Anil K. Jain
Department of Computer Science and Engineering,
Michigan State University, East Lansing, MI, 48824
{debdebay, liuxm, jain}@cse.msu.edu

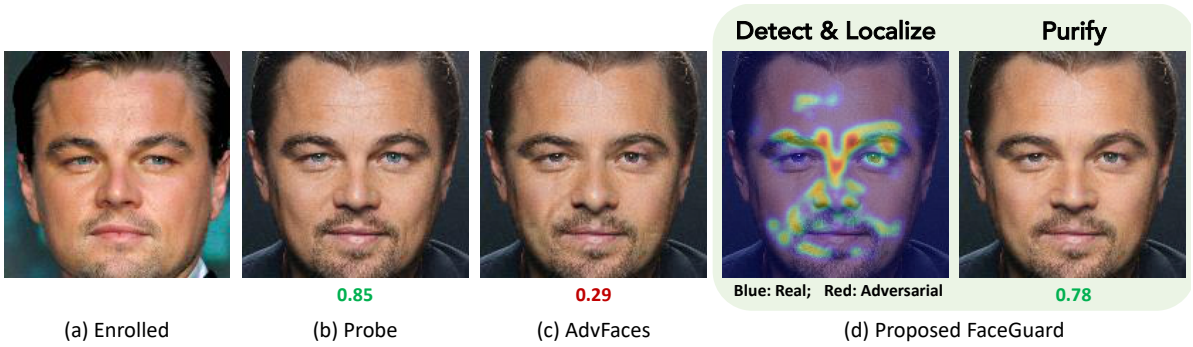


Figure 1: Leonardo DiCaprio's real face photo (a) enrolled in the gallery and (b) his probe image[1]; (c) Adversarial probe synthesized by a state-of-the-art (SOTA) adversarial face generator, AdvFaces [1]; (d) Proposed adversarial defense framework, namely *FaceGuard* takes (c) as input, detects adversarial images, localizes perturbed regions, and outputs a "purified" face devoid of adversarial perturbations. A SOTA face recognition system, ArcFace, fails to match Leonardo's adversarial face (c) to (a), however, the purified face can successfully match to (a). Cosine similarity scores ($\in [-1, 1]$) obtained via ArcFace [2] are shown below the images. A score above **0.36** (threshold @ 0.1% False Accept Rate) indicates that two faces are of the same subject.

Abstract

Prevailing defense schemes against adversarial face images tend to overfit to the perturbations in the training set and fail to generalize to unseen adversarial attacks. We propose a new self-supervised adversarial defense framework, namely FaceGuard, that can automatically detect, localize, and purify a wide variety of adversarial faces without utilizing pre-computed adversarial training samples. During training, FaceGuard automatically synthesizes challenging and diverse adversarial attacks, enabling a classifier to learn to distinguish them from real faces. Concurrently, a purifier attempts to remove the adversarial perturbations in the image space. Experimental results on LFW dataset show that FaceGuard can achieve 99.81% *detection accuracy on six* ***unseen*** *adversarial attack types. In addition, the proposed method can enhance the face recognition performance of ArcFace from* 34.27% *TAR @* 0.1% *FAR under no defense to* 77.46% *TAR @* 0.1% *FAR. Code, pre-trained models and dataset will be publicly available.*

1. Introduction

With the advent of deep learning and availability of large datasets, Automated Face Recognition (AFR) systems have achieved impressive recognition rates [3]. The accuracy, usability, and touchless acquisition of state-of-the-art (SOTA) AFR systems have led to their ubiquitous adoption in a plethora of domains. However, this has also inadvertently sparked a community of attackers that dedicate their time and effort to manipulate faces in order to evade AFR systems [4]. AFR systems have been shown to be vulnerable to adversarial attacks resulting from perturbing an input probe [1, 5–7]. Even when the amount of perturbation is imperceptible to the human eye, such adversarial attacks can degrade the face recognition performance of SOTA AFR systems [1]. With the growing dissemination of "fake news" and "deepfakes" [8], research groups and social media platforms alike are pushing towards generalizable defense against continuously evolving adversarial attacks.

A considerable amount of research has focused on synthesizing adversarial attacks [1, 6, 7, 9–11]. Obfuscation attempts (faces are perturbed such that they cannot be identified as the attacker) are more effective [1], computationally efficient to synthesize [9, 10], and widely adopted [12] compared to impersonation attacks (perturbed faces can automatically match to a target subject). Similar to prior defense

[1] https://bit.ly/2IkfSxk

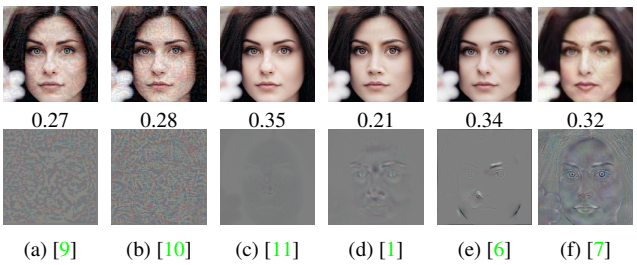


Figure 2: *(Top Row)* Adversarial face images synthesized via six adversarial attack generators used in our study. *(Bottom Row)* Corresponding adversarial perturbations (gray indicates no change from the input). Notice the diversity in the perturbations. ArcFace scores between adversarial image and the unaltered gallery image (not shown here) are given below each image. A score above **0.36** indicates that two faces are of the same subject. Zoom in for details.

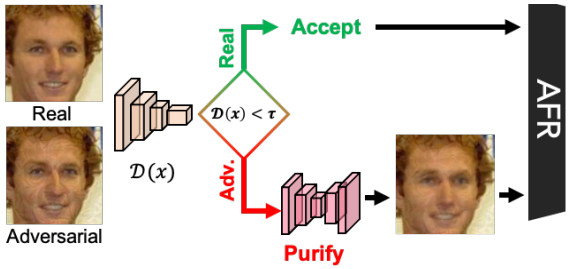


Figure 3: *FaceGuard* employs a detector ($\mathcal{D}$) to compute an adversarial score. Scores below detection threshold (τ) passes the input to AFR, and high value invokes a purifier and sends the purified face to the AFR system.

efforts [13, 14], this paper focuses on defending against obfuscation attacks. Given an input probe image, $\mathbf{x}$, an adversarial generator has two requirements under the obfuscation scenario: (1) synthesize an adversarial face image, $\mathbf{x}_{adv} = \mathbf{x} + \delta$, such that SOTA AFR systems fail to match $\mathbf{x}_{adv}$ and $\mathbf{x}$, and (2) limit the magnitude of perturbation $||\delta||_p$ such that $\mathbf{x}_{adv}$ appears very similar to $\mathbf{x}$ to humans.

A number of approaches have been proposed to defend against adversarial attacks. Their major shortcoming is *generalizability* to unseen adversarial attacks. Adversarial face perturbations may vary significantly (see Fig. 2). For instance, gradient-based attacks, such as FGSM [10] and PGD [10], perturb every pixel in the face image, whereas, AdvFaces [1] and SemanticAdv [7] perturb only the salient facial regions, *e.g.*, eyes, nose, and mouth. On the other hand, GFLM [6] performs geometric warping to the face. Since the exact type of adversarial perturbation may not be known a priori, a defense system trained on a subset of adversarial attack types may not perform well on other unseen adversarial attacks.

To the best of our knowledge, we take the first step towards a complete defense against adversarial faces by integrating an adversarial face generator, a detector, and a purifier into a unified framework, namely *FaceGuard* (see Fig. 3). Robustness to unseen adversarial attacks is imparted via a stochastic generator that outputs diverse perturbations evading an AFR system, while a detector continuously learns to distinguish them from real faces. Concurrently, a purifier removes the adversarial perturbations from the synthesized image.

This work makes the following contributions:

- A new self-supervised framework, namely *FaceGuard*, for defending against adversarial face images. *FaceGuard* combines benefits of adversarial training, detection, and purification into a unified defense mechanism trained in an end-to-end manner.
- With the proposed diversity loss, a generator is regularized to produce stochastic and challenging adversarial faces. We show that the diversity in output perturbations is sufficient for improving *FaceGuard*'s robustness to unseen attacks compared to utilizing precomputed training samples from known attacks.
- Synthesized adversarial faces aid the detector to learn a tight decision boundary around real faces. *FaceGuard*'s detector achieves SOTA detection accuracy of 99.81% on 6 unseen adversarial attacks on LFW [15].
- As the generator trains, a purifier concurrently removes perturbations from the synthesized adversarial faces. With the proposed bonafide loss, the detector also guides purifier's training to ensure purified images are devoid of adversarial perturbations. *FaceGuard*'s purifier enhances the face recognition performance ArcFace [2] from 34.27% TAR @ 0.1% FAR under no defense to 77.46% TAR @ 0.1% FAR.

2. Related Work

Defense Strategies:. In literature, a commonly employed defense strategy is to re-train the classifier we wish to defend with adversarial examples [9, 10, 16, 17]. However, adversarial training has been shown to degrade classification accuracy on real (non-adversarial) images [18, 19].

In order to prevent degradation in AFR performance, a large number of adversarial defense mechanisms are deployed as a pre-processing step, namely *adversarial detection*, which involves training a binary classifier to distinguish between real and adversarial examples [13, 14, 20–32]. The attacks considered in these studies [33–36] were initially proposed in the object recognition domain and they often fail to detect the attacks in a feature-extraction network setting, as in face recognition. Therefore, prevailing detection-based defenses against adversarial faces are demonstrated to be effective only in a highly constrained setting where the number of subjects is limited and fixed during training and testing [13, 14, 32].

Another pre-processing strategy, namely *purification*, involves automatically removing adversarial perturbations in the input image prior to passing them to a face matcher [37–

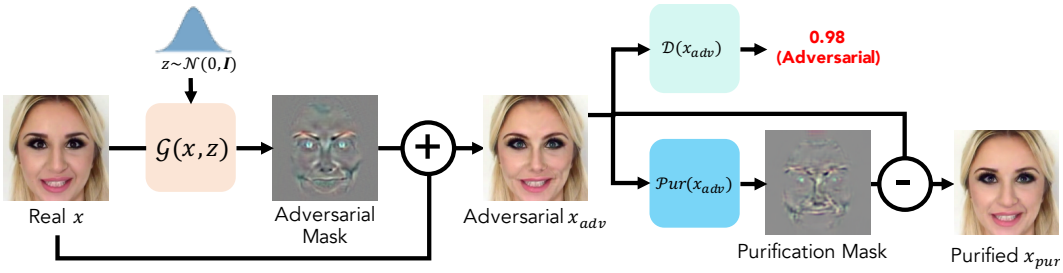


Figure 4: Overview of training the proposed *FaceGuard* in a self-supervised manner. An *adversarial generator*, $\mathcal{G}$, continuously learns to synthesize challenging and diverse perturbations that evade a face matcher. At the same time, a *detector*, $\mathcal{D}$, learns to distinguish between the synthesized adversarial faces and real face images. Perturbations residing in the synthesized adversarial faces are removed via a *purifier*, $\mathcal{P}ur$.

40]. However, without a dedicated adversarial detector, these defenses may end up "purifying" a real face image, resulting in high false reject rates.

Adversarial Attacks:. Numerous adversarial attack generators have been proposed in literature [9, 10, 41–43]. For example, Fast Gradient Sign Method (FGSM) generates an adversarial example by back-propagating through the target model [9]. Other approaches optimize adversarial perturbation by minimizing an objective function while satisfying certain constraints [11, 43]. These approaches rely on softmax cross-entropy loss to find effective perturbations. We modify the objective functions of these attacks in order to synthesize adversarial faces that evade AFR systems. We evaluate *FaceGuard* on six unseen adversarial attacks that have high success rates in evading a SOTA AFR system, ArcFace [2]: FGSM [9], PGD [10], DeepFool [11], AdvFaces [1], GFLM [6], and SemanticAdv [7] (see Tab. 1).

3. FaceGuard

3.1. Limitations of State-of-the-Art Defenses

Adversarial Training. Adversarial training is regarded as one of the most effective defense method [9, 10, 44] on small datasets including MNIST and CIFAR10. Whether this technique can scale to large datasets and a variety of different attack types (perturbation sets) has not yet been shown. Adversarial training is formulated as [9, 10]:

$$\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{P}_{data}} \left[\max_{\delta\in\Delta} \ell\left(f_\theta\left(x+\delta\right), y\right) \right], \qquad (1)$$

where $(x, y) \sim \mathcal{P}_{data}$ is the (image, label) joint distribution of data, $f_\theta(x)$ is the network parameterized by θ, and $\ell(f_\theta(x), y)$ is the loss function (usually cross-entropy). Since the ground truth data distribution, $\mathcal{P}_{data}$, is not known in practice, it is later replaced by the empirical distribution. Here, the network, f_θ is made robust by training with an adversarial noise (δ) that maximally increases the classification loss. In other words, adversarial training involves training with the *strongest* adversarial attack.

The generalization of adversarial training has been in question [17–19, 44, 45]. It was shown that adversarial training can significantly reduce classification accuracy on real examples [18, 19]. In the context of face recognition, we illustrate this by training two face matchers on CASIA-WebFace: (i) FaceNet [46] trained via the standard training process, and (ii) FaceNet [46] by adversarial training (FGSM[2]). We then compute face recognition performance across training iterations on a separate testing dataset, LFW [15]. Fig. 5a shows that adversarial training drops the accuracy from $99.13\% \rightarrow 98.27\%$. We gain the following insight: adversarial training may degrade AFR performance on real faces.

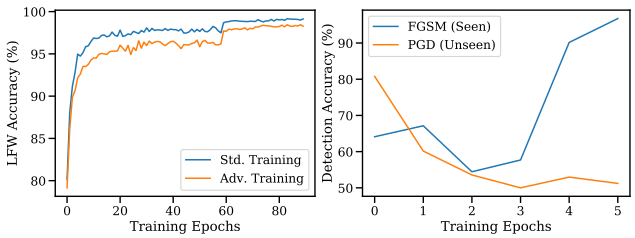
Detectors. Detection-based approaches employ a preprocessing step to "detect" whether an input face is real or adversarial [13, 22, 23, 32]. A common approach is to utilize a binary classifier, $\mathcal{D}$, that maps a face image, $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ to $\{0, 1\}$, where 0 indicates a real and 1 an adversarial face. We train a binary classifier to distinguish between real and FGSM attack samples in CASIA-WebFace [47]. In Figure 5b, we evaluate its detection accuracy on FGSM and PGD samples in LFW [15]. We find that prevailing detection-based defense schemes may overfit to the specific adversarial attacks utilized for training.

3.2. Proposed Defense

Our defense aims to achieve robustness without sacrificing AFR performance on real face images. We posit that an adversarial defense trained alongside an adversarial generator in a *self-supervised* manner may improve robustness to unseen attacks. The main intuitions behind our defense mechanism are as follows:

- Since adversarial training may degrade AFR performance, we aim to obtain a robust adversarial *detector* and *purifier* to detect and purify unseen adversarial attacks.
- Given that prevailing detection-based methods tend to overfit to known adversarial perturbations (see Supplementary), a detector and purifier trained on a wide *variety* of synthesized adversarial perturbations may be more robust to unseen attacks.

[2]With max perturbation hyperparameter as $\epsilon = 8/256$.



(a) Adversarial Training [16] (b) Detection [22]

Figure 5: (a) Adversarial training degrades AFR performance of FaceNet matcher [46] on real faces in LFW dataset compared to standard training. (b) A binary classifier trained to distinguish between real faces and FGSM [9] attacks fails to detect another attack type, namely PGD [10].

- Sufficient diversity in synthesized perturbations can guide the detector to learn a tighter boundary around real faces. In this case, the detector itself can serve as a powerful discriminator for the purifier.
- Lastly, we posit that the pixels involved in the purification process may serve to indicate adversarial regions in the input face.

3.3. Adversarial Generator

The generalizability of an adversarial detector and purifier relies on the quality of the synthesized adversarial face images output by *FaceGuard*'s adversarial generator. We propose an adversarial generator that continuously learns to synthesize challenging and diverse adversarial face images.

The generator, denoted as $\mathcal{G}$, takes an input real face image, $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$, and outputs an adversarial perturbation $\mathcal{G}(\mathbf{x}, \mathbf{z})$, where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is a random latent vector. Inspired by prevailing adversarial attack generators [1, 9–11, 43], we treat the output perturbation $\mathcal{G}(\mathbf{x}, \mathbf{z})$ as an additive *perturbation mask*. The final adversarial face image, $\mathbf{x}_{adv}$, is given by $\mathbf{x}_{adv} = \mathbf{x} + \mathcal{G}(\mathbf{x}, \mathbf{z})$.

In an effort to impart generalizability to the detector and purifier, we emphasize the following requirements of $\mathcal{G}$:

- **Adversarial:** Perturbatation, $\mathcal{G}(\mathbf{x}, \mathbf{z})$, needs to be adversarial such that an AFR system cannot identify the adversarial face image $\mathbf{x}_{adv}$ as the same person as the input probe $\mathbf{x}$.
- **Visually Realistic:** Perturbation $\mathcal{G}(\mathbf{x}, \mathbf{z})$ should also be minimal such that $\mathbf{x}_{adv}$ appears as a legitimate face image of the subject in the input probe $\mathbf{x}$.
- **Stochastic:** For an input $\mathbf{x}$, we require diverse adversarial perturbations, $\mathcal{G}(\mathbf{x}, \mathbf{z})$, for different latents $\mathbf{z}$.

For satisfying all of the above requirements, we propose multiple loss functions to train the generator.

Obfuscation Loss To ensure $\mathcal{G}(\mathbf{x}, \mathbf{z})$ is indeed *adversarial*, we incorporate a white-box AFR system, $\mathcal{F}$, to supervise the generator. Given an input face, $\mathbf{x}$, the generator aims to output an adversarial face, $\mathbf{x}_{adv} = \mathbf{x} + \mathcal{G}(\mathbf{x}, \mathbf{z})$ such that the face representations, $\mathcal{F}(\mathbf{x})$ and $\mathcal{F}(\mathbf{x}_{adv})$, do not match. In other words, the goal is to minimize the cosine similarity between the two face representations[3]:

$$\mathcal{L}_{obf} = \mathbb{E}_{\mathbf{x}} \left[\frac{\mathcal{F}(\mathbf{x}) \cdot \mathcal{F}(\mathbf{x}_{adv})}{||\mathcal{F}(\mathbf{x})||\,||\mathcal{F}(\mathbf{x}_{adv})||} \right]. \tag{2}$$

Perturbation Loss With the identity loss alone, the generator may output perturbations with large magnitudes which will be (a) trivial for the detector to reject and (b) violate the visual realism requirement of x_{adv}. Therefore, we restrict the perturbations to be within $[-\epsilon, \epsilon]$ via a hinge loss:

$$\mathcal{L}_{pt} = \mathbb{E}_{\mathbf{x}} \left[\max\left(\epsilon, ||\mathcal{G}(\mathbf{x}, \mathbf{z})||_2\right) \right]. \tag{3}$$

Diversity Loss The above two losses jointly ensure that at each step, our generator learns to output challenging adversarial attacks. However, these attacks are deterministic; for an input image, we will obtain the same adversarial image. This may again lead to an inferior detector that overfits to a few deterministic perturbations seen during training. Motivated by studies of preventing mode collapse in GANs [48], we propose maximizing a diversity loss to promote stochastic perturbations per training iteration, i:

$$\mathcal{L}_{div} = -\frac{1}{N_{ite}} \sum_{i=1}^{N_{ite}} \frac{\left|\left|\mathcal{G}(\mathbf{x}, \mathbf{z}_1)^{(i)} - \mathcal{G}(\mathbf{x}, \mathbf{z}_2)^{(i)}\right|\right|_1}{||\mathbf{z}_1 - \mathbf{z}_2||_1}, \tag{4}$$

where N_{ite} is the number of training iterations, $\mathcal{G}(\mathbf{x}, \mathbf{z})^{(i)}$ is the perturbation output at iteration i, and $(\mathbf{z}_1, \mathbf{z}_2)$ are two i.i.d. samples from $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. The diversity loss ensures that for two random latent vectors, $\mathbf{z}_1$ and $\mathbf{z}_2$, we will obtain two different perturbations $\mathcal{G}(\mathbf{x}, \mathbf{z}_1)^{(i)}$ and $\mathcal{G}(\mathbf{x}, \mathbf{z}_2)^{(i)}$.

GAN Loss Akin to prior work on GANs [49, 50], we introduce a discriminator to encourage perceptual realism of the adversarial images. The discriminator, Dsc, aims to distinguish between probes, $\mathbf{x}$, and synthesized faces $\mathbf{x}_{adv}$ via a GAN loss:

$$\mathcal{L}_{GAN} = \mathbb{E}_{\mathbf{x}} \left[\log Dsc(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}}[\log(1 - Dsc(\mathbf{x}_{adv}))]. \tag{5}$$

3.4. Adversarial Detector

Similar to prevailing adversarial detectors, the proposed detector also learns a decision boundary between real and adversarial images [13, 22, 23, 32]. A key difference, however, is that instead of utilizing pre-computed adversarial images from known attacks (*e.g.* FGSM and PGD) for training, the proposed detector learns to distinguish between real images and the *synthesized* set of diverse adversarial attacks output by the proposed adversarial generator in a self-supervised manner. This leads to the following advantage: *our proposed framework does not require a large collection of pre-computed adversarial face images for training*.

[3] For brevity, we denote $\mathbb{E}_{\mathbf{x}} \equiv \mathbb{E}_{\mathbf{x}\in\mathcal{P}_{data}}$.

We utilize a binary CNN for distinguishing between real input probes, $\mathbf{x}$, and synthesized adversarial samples, $\mathbf{x}_{adv}$. The detector is trained with the Binary Cross-Entropy loss:

$$\mathcal{L}_{BCE} = \mathbb{E}_{\mathbf{x}}\left[\log \mathcal{D}(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}}\left[\log\left(1 - \mathcal{D}(\mathbf{x}_{adv})\right)\right]. \quad (6)$$

3.5. Adversarial Purifier

The objective of the adversarial purifier is to recover the real face image $\mathbf{x}$ given an adversarial face $\mathbf{x}_{adv}$. We aim to automatically remove the adversarial perturbations by training a neural network $\mathcal{P}ur$, referred as an adversarial purifier.

The adversarial purification process can be viewed as an inverted procedure of adversarial image synthesis. Contrary to the obfuscation loss in the adversarial generator, we require that the purified image, $\mathbf{x}_{pur}$, successfully matches to the subject in the input probe $\mathbf{x}$. Note that this can be achieved via a *feature recovery loss*, which is the opposite to the obfuscation loss, *i.e.*, $\mathcal{L}_{fr} = -\mathcal{L}_{obf}$.

Note that an adversarial face image, $\mathbf{x}_{adv} = \mathbf{x} + \delta$, is metrically close to the real image, $\mathbf{x}$, in the input space. If we can estimate δ, then we can retrieve the real face image. Here, the perturbations can be predicted by a neural network, $\mathcal{P}ur$. In other words, retrieving the purified image, $\mathbf{x}_{pur}$ involves: (1) subtracting the perturbations from the adversarial image, $\mathbf{x}_{pur} = \mathbf{x}_{adv} - \mathcal{P}ur(\mathbf{x}_{adv})$ and (2) ensuring that the *purification mask*, $\mathcal{P}ur(\mathbf{x}_{adv})$, is small so that we do not alter the content of the face image by a large magnitude. Therefore, we propose a hybrid perceptual loss that (1) ensures $\mathbf{x}_{pur}$ is as close as possible to the real image, $\mathbf{x}$ via a ℓ_1 reconstruction loss and (2) a loss that minimizes the amount of alteration, $\mathcal{P}ur(\mathbf{x}_{adv})$:

$$\mathcal{L}_{perc} = \mathbb{E}_{\mathbf{x}}\left|\left|\mathbf{x}_{pur} - \mathbf{x}\right|\right|_1 + \left|\left|\mathcal{P}ur(\mathbf{x}_{adv})\right|\right|_2. \quad (7)$$

Finally, we also incorporate our detector to guide the training of our purifier. Note that, due to the diversity in synthesized adversarial faces, the proposed detector learns a tight decision boundary around real faces. This can serve as a strong self-supervisory signal to the purifier for ensuring that the purified images belong to the real face distribution. Therefore, we also incorporate the detector as a discriminator for the purifier via the proposed bonafide loss:

$$\mathcal{L}_{bf} = \mathbb{E}_{\mathbf{x}}\left[\log \mathcal{D}(\mathbf{x}_{pur})\right]. \quad (8)$$

3.6. Training Framework

We train the entire *FaceGuard* framework in Fig. 4 in an end-to-end manner with the following objectives:

$$\min_{\mathcal{G}} \mathcal{L}_{\mathcal{G}} = \mathcal{L}_{GAN} + \lambda_{obf} \cdot \mathcal{L}_{obf} + \lambda_{pt} \cdot \mathcal{L}_{pt} - \lambda_{div} \cdot \mathcal{L}_{div},$$

$$\min_{\mathcal{D}} \mathcal{L}_{\mathcal{D}} = \mathcal{L}_{BCE},$$

$$\min_{\mathcal{P}ur} \mathcal{L}_{\mathcal{P}ur} = \lambda_{fr} \cdot \mathcal{L}_{fr} + \lambda_{perc} \cdot \mathcal{L}_{perc} + \lambda_{bf} \cdot \mathcal{L}_{bf}.$$

ArcFace [2]	TAR (%) @ 0.1% FAR(↓)	Mean SSIM(↑)
FGSM [9]	26.23	0.83
PGD [10]	04.91	0.89
DeepFool [11]	36.18	0.91
AdvFaces [1]	00.17	0.89
GFLM [6]	68.03	0.55
SemanticAdv [7]	70.05	0.71

Table 1: Face recognition performance of ArcFace [2] under adversarial attack and average structural similarities (SSIM) between probe and adversarial images for obfuscation attacks on $485K$ genuine pairs in LFW [15]. ArcFace [2] achieves 99.82% TAR @ 0.1% FAR on real pairs.

At each training iteration, the generator attempts to fool the discriminator by synthesizing visually realistic adversarial faces while the discriminator learns to distinguish between real and synthesized images. On the other hand, in the same iteration, an external critic network, namely detector $\mathcal{D}$, learns a decision boundary between real and synthesized adversarial samples. Concurrently, the purifier $\mathcal{P}ur$ learns to invert the adversarial synthesis process. Note that there is a key difference between the discriminator and the detector: the generator is designed to specifically *fool* the discriminator but not necessarily the detector. We will show in our experiments that this crucial step prevents the detector from predicting $\mathcal{D}(\mathbf{x}) = 0.5$ for all $\mathbf{x}$.

4. Experimental Results

4.1. Experimental Settings

Datasets. We train *FaceGuard* on real face images in CASIA-WebFace [47] dataset and then evaluate on real and adversarial faces synthesized for LFW [15] dataset. CASIA-WebFace [47] comprises of $494,414$ face images from $10,575$[4] different subjects. LFW [15] contains $13,233$ face images of $5,749$ subjects. Since we evaluate defenses under obfuscation attacks, we consider subjects with at least two face images[5]. After this filtering, $9,164$ face images of $1,680$ subjects in LFW are available for evaluation.

Implementation. The adversarial generator and purifier employ a convolutional encoder-decoder. The latent variable $\mathbf{z}$, a 128-dimensional feature vector, is fed as input to the generator through spatial padding and concatenation. The adversarial detector, a 4-layer binary CNN, is trained jointly with the generator and purifier. Empirically, we set $\lambda_{obf} = \lambda_{fr} = 10.0$, $\lambda_{pt} = \lambda_{perc} = 1.0$, $\lambda_{div} = 1.0$, $\lambda_{bf} = 1.0$ and $\epsilon = 3.0$. Training and network architecture details are provided in the supplementary material.

Face Recognition Systems. In this study, we use two AFR systems: FaceNet [46] and ArcFace [2]. Recall that the proposed defense utilizes a face matcher, $\mathcal{F}$, for guiding

[4]We removed 84 subjects in CASIA-WebFace that overlap with LFW.

[5]Obfuscation attempts only affect genuine pairs (two face images pertaining to the same subject).

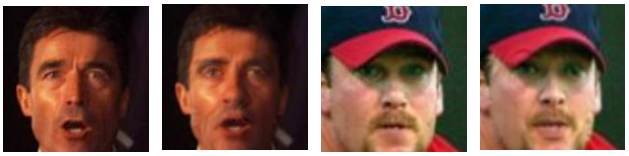
Detection Acc. (%)	FGSM [9]	PGD [10]	DeepFool [11]	AdvFaces [1]	GFLM [6]	SemanticAdv [7]	Mean ± Std.
Gong *et al.* [22]	98.94	97.91	95.87	92.69	**99.92**	**99.92**	97.54 ± 02.82
UAP-D [32]	61.32	74.33	56.78	51.11	65.33	76.78	64.28 ± 09.97
SmartBox [14]	58.79	62.53	51.32	54.87	50.97	62.14	56.77 ± 05.16
Massoli *et al.* [13] (MLP)	63.58	76.28	81.78	88.38	51.97	52.98	69.16 ± 15.29
Massoli *et al.* [13] (LSTM)	71.53	76.43	88.32	75.43	53.76	55.22	70.11 ± 13.35
Proposed FaceGuard	**99.85**	**99.85**	**99.85**	**99.84**	99.61	99.85	**99.81 ± 00.10**

Table 2: Detection accuracy of SOTA adversarial face detectors in classifying six adversarial attacks synthesized for the LFW dataset [15]. Detection threshold is set as 0.5 for all methods. All baseline methods require training on pre-computed adversarial attacks on CASIA-WebFace [47]. On the other hand, the proposed *FaceGuard* is self-guided and generates adversarial attacks on the fly. Hence, it can be regarded as a *black-box* defense system.

the training process of the generator. However, the deployed AFR system may not be known to the defense system a priori. Therefore, unlike prevailing defense mechanisms [13, 14, 32], we evaluate the effectiveness of the proposed defense on an AFR system *different* from $\mathcal{F}$. We highlight the effectiveness of our proposed defense: *FaceGuard is trained on FaceNet, while the adversarial attack test set is designed to evade ArcFace.* Obfuscation attempts perturb real probes into adversarial ones. Ideally, deployed AFR systems (say, ArcFace), should be able to match a genuine pair comprised of an adversarial probe and a real enrolled face of the same subject. Therefore, regardless of real or adversarial probe, we assume that genuine pairs should *always* match as ground truth. Tab. 1 provides AFR performance of ArcFace under six SOTA adversarial attacks for $484, 514$ genuine pairs in LFW.

0.77 0.67 0.96 0.99

(a) Real faces falsely detected as adversarial



Real AdvFaces (0.42) Real AdvFaces (0.28)

(b) Adversarial faces falsely detected as real

Figure 6: Examples where the proposed *FaceGuard* fails to correctly detect (a) real faces and (b) adversarial faces. Detection scores $\in [0, 1]$ are given below each image, where 0 indicates real and 1 indicates adversarial face.

4.2. Comparison with State-of-the-Art Defenses

In this section, we compare the proposed *FaceGuard* to prevailing defenses (detectors, purifiers, and adversarial training techniques). We evaluate all methods via publicly available repositories provided by the authors (see Supplementary for links). Only modification made is to replace their training datasets with CASIA-WebFace [47].

SOTA Detectors. Our baselines include five SOTA detectors proposed specifically for detecting adversarial faces [13, 14, 32]. Detection performance with a binary CNN [22] is also computed. The detectors are trained on real and adversarial faces images synthesized via six adversarial generators for CASIA-WebFace [47]. Unlike all the baselines, *FaceGuard*'s detector does not utilize any pre-computed adversarial attack for training. We compute the classification accuracy for all methods on a dataset comprising of $9,164$ real images and $9,164$ adversarial face images per attack type in LFW.

In Tab. 2, we find that compared to the baselines, *FaceGuard* achieves the highest detection accuracy. Even when the six adversarial attack types are encountered in training, a binary CNN [22], still falls short compared to *FaceGuard*. This is likely because *FaceGuard* is trained on a challenging and diverse set of adversarial faces from the proposed generator. Note that the performance of the binary CNN [22] significantly drops when unseen adversarial attack types are encountered in testing (see Supplementary).

Compared to hand-crafted features, such as PCA+SVM in UAP-D [32] and entropy detection in SmartBox [14], *FaceGuard* achieves superior detection results. Massoli *et al.* [13] distinguish between real and adversarial faces by utilizing intermediate face features of an AFR system. However, we find that these intermediate features primarily represent the identity of the input face and do not appear to contain highly discriminative information for detecting adversarial faces.

Despite the robustness, *FaceGuard* misclassifies 28 out of $9,164$ real images in LFW [15] and falsely predicts 46 out of $9,164$ adversarial faces as real. Of these 46 misclassifications, 44 are geometrically-warped faces via GFLM [6] and the remaining two are synthesized via AdvFaces [1]. We find that *FaceGuard* tends to misclassify real faces under extreme poses and adversarial faces that are occluded (*e.g.*, hats) (see Fig. 6).

Comparison with Adversarial Training & Purifiers. We also compare with prevailing defenses designing robust face matchers [16, 17, 44] and purifiers [37, 38, 40]. We conduct a verification experiment by considering all possible genuine pairs (two faces belonging to the same subject) in

Defenses	Approach	Real $485K$ pairs	Attacks $3M$ pairs
No-Defense	-	99.82	34.27
Adv. Training [16]	Robustness	96.42	11.23
Rob-GAN [44]	Robustness	91.35	13.89
L2L [17]	Robustness	96.89	16.76
MagNet [37]	Purification	94.47	38.32
DefenseGAN [38]	Purification	96.78	39.21
NRP [40]	Purification	97.54	61.44
Proposed FaceGuard	Purification	**99.81**	**77.46**

Table 3: AFR performance (TAR (%) @ 0.1% FAR) of ArcFace under no defense and when ArcFace is trained via SOTA adversarial training techniques [16,17,44] and SOTA purifiers [37,38]. *FaceGuard* correctly passes majority of real faces to ArcFace and also purifies adversarial attacks.

	Model	AdvFaces [1]	Mean $\pm$ Std.
Gen. $\mathcal{G}$	Without $\mathcal{G}$	91.72	97.12 ± 04.54
	Without $\mathcal{L}_{div}$	95.42	98.23 ± 01.33
	With $\mathcal{G}$ and $\mathcal{L}_{div}$	**99.84**	$\mathbf{99.81 \pm 00.10}$
Det. $\mathcal{D}$	$\mathcal{D}$ as Discriminator	50.00	75.25 ± 21.19
	$\mathcal{D}$ via Pre-Computed $\mathcal{G}$	52.01	69.37 ± 19.91
	$\mathcal{D}$ as Online Detector	**99.84**	$\mathbf{99.81 \pm 00.10}$

Table 4: Ablating training schemes of the generator $\mathcal{G}$ and detector $\mathcal{D}$. All models are trained on CASIA-WebFace [47]. *(Col. 3)* We compute the detection accuracy in classifying real faces in LFW [15] and the most challenging adversarial attack in Tab. 1, AdvFaces [1]. *(Col. 4)* The avg. and std. dev. of detection accuracy across all 6 adversarial attacks.

LFW [15]. For one probe in a genuine pair, we craft six different adversarial probes (one per attack type). In total, there are $484,514$ real pairs and $\sim 3M$ adversarial pairs. For a fixed match threshold[6], we compute the True Accept Rate (TAR) of successfully matching two images in a real or adversarial pair in Tab. 3. In other words, TAR is defined here as the ratio of genuine pairs above the match threshold.

ArcFace without any adversarial defense system achieves 34.27% TAR at 0.1% FAR under attack. Adversarial training [16,17,44] inhibits the feature space of ArcFace, resulting in worse performance on both real and adversarial pairs. On the other hand, purification methods [37, 38, 40] can better retain face features in real pairs but their performance under attack is still undesirable.

Instead, the proposed *FaceGuard* defense system first detects whether an input face image is real or adversarial. If input faces are adversarial, then they are further purified. From Tab. 3, we find that our defense system significantly outperforms SOTA baselines in protecting ArcFace [2] against adversarial attacks. Specifically, *FaceGuard*'s purifier enhances ArcFace's average TAR at 0.1% FAR under all six attacks (see Tab. 1) from $34.27\% \rightarrow 77.46\%$. In addition, *FaceGuard* also maintains similar face recognition performance on real faces (TAR on real pairs drop from $99.82\% \rightarrow 99.81\%$). Therefore, our proposed defense system ensures that benign users will not be incorrectly rejected while malicious attempts to evade the AFR system will be curbed.

4.3. Analysis of Our Approach

Quality of the Adversarial Generator. In Tab. 4, we see that without the proposed adversarial generator ("Without $\mathcal{G}$"), *i.e.*, a detector trained on the six known attack types, suffers from high standard deviation. Instead, training a detector with a deterministic $\mathcal{G}$ ("Without $\mathcal{L}_{div}$"), leads to better generalization across attack types, since the detector still encounters variations in synthesized images as the generator learns to better generate adversarial faces. However, such a detector is still prone to overfitting to a few deterministic perturbations output by $\mathcal{G}$. Finally, *FaceGuard* with the diversity loss introduces diverse perturbations within and across training iterations (see Fig. 7). Fig. 7 also highlights the superiority of the proposed generator: as $\mathcal{G}$ trains, synthesized adversarial face images appear closer to the real distribution while spanning the adversarial space across all 6 known attacks. This illustrates how the stochastic perturbations output by the proposed generator can (1) significantly improve the robustness of the detector to unseen adversarial attacks ("With $\mathcal{G}$ and $\mathcal{L}_{div}$") and (2) eliminate the need for utilizing pre-computed training samples from known attacks.

Quality of the Adversarial Detector. The discriminator's task is similar to the detector; determine whether an input image is real or fake/adversarial. The key difference is that the generator is enforced to fool the discriminator, but not the detector. If we replace the discriminator with an adversarial detector, the generator continuously attempts to fool the detector by synthesizing images that are as close as possible to the real image distribution. By design, such a detector should converge to $Disc(\mathbf{x}) = 0.5$ for all $\mathbf{x}$ (real or adversarial). As we expect, in Tab. 4, we cannot rely on predictions made by such a detector ("$\mathcal{D}$ as Discriminator"). We try another variant: we first train the generator $\mathcal{G}$ and then train a detector to distinguish between real and pre-computed attacks via $\mathcal{G}$ ("$\mathcal{D}$ via Pre-Computed $\mathcal{G}$"). As we expect, the proposed methodology of training the detector in an online fashion by utilizing the synthesized adversarial samples output by $\mathcal{G}$ at any given iteration leads to a significantly robust detector ("$\mathcal{D}$ as Online Detector"). This can likely be attributed to the fact that a detector trained on-line encounters a much larger variation as the generator trains alongside. "$\mathcal{D}$ via Pre-Computed $\mathcal{G}$" is exposed only to within-iteration variations (from random latent sampling), however, '$\mathcal{D}$ as Online Detector" encounters variations *both* within and across training iterations (see Fig. 7).

Quality of the Adversarial Purifier. Recall that we en-

[6]We compute the threshold at 0.1% FAR on all possible image pairs in LFW, *e.g.*, threshold @ 0.1% FAR for ArcFace is set at 0.36.

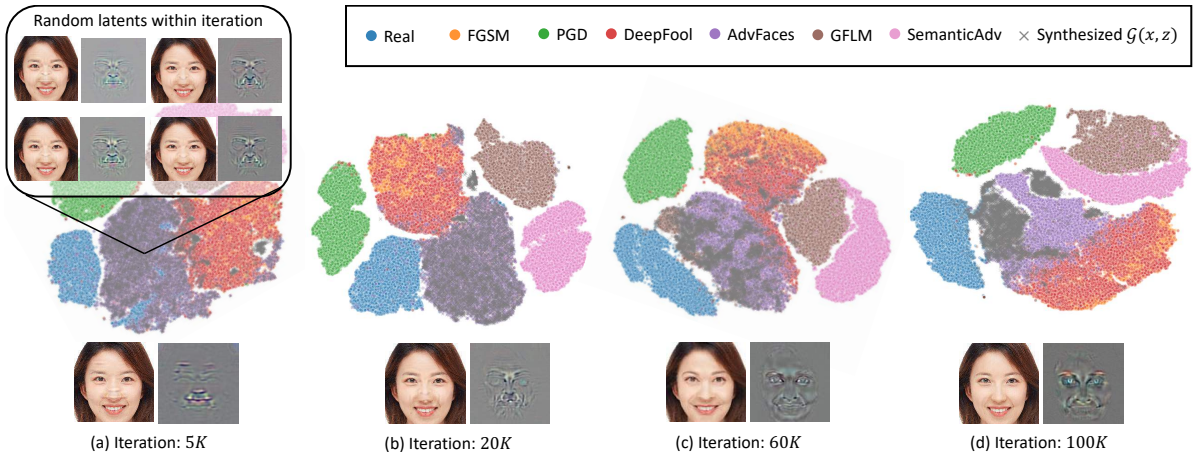


Figure 7: Adversarial faces synthesized by *FaceGuard*'s generator during training. Note the diversity in perturbations within and across iterations. 2D t-SNE visualization of features for real images, six known adversarial attacks, and synthesized adversarial images extracted via the proposed detector. As *FaceGuard* trains, synthesized adversarial faces appear closer to the real faces while also spanning the known adversarial space.

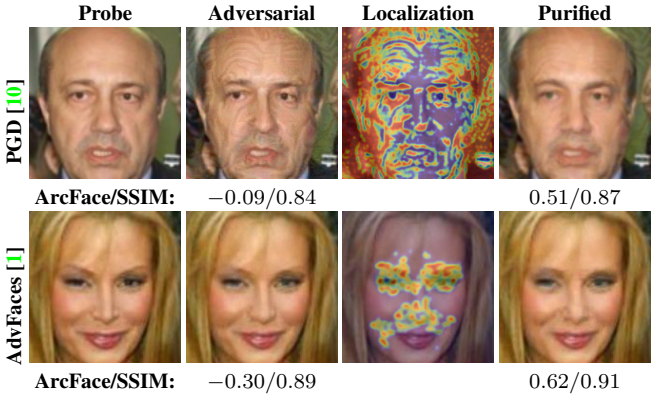


Figure 8: Example results where *FaceGuard* successfully purifies the adversarial image (red regions indicate adversarial perturbations localized by our purification mask). ArcFace [2] scores $\in [-1, 1]$ and SSIM $\in [0, 1]$ between an adversarial/purified probe and input probe are given below each image.

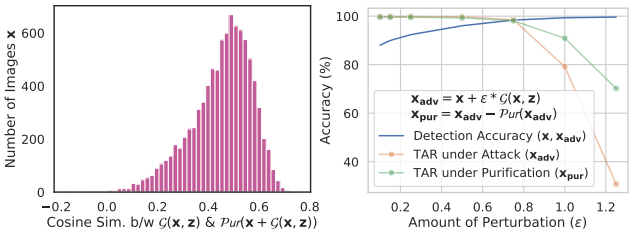


Figure 9: (a) *FaceGuard*'s purification is correlated with its adversarial synthesis process. (b) Trade-off between detection and purification with respect to perturbation magnitudes. With minimal perturbation, detection is challenging while purifier maintains AFR performance. Excessive perturbations lead to easier detection with greater challenge in purification.

forced the purified image to be close to the real face via a reconstruction loss, such that $\mathbf{x} \approx \mathbf{x}_{pur} \implies \mathbf{x} \approx (\mathbf{x} + \mathcal{G}(\mathbf{x}, \mathbf{z})) - \mathcal{P}ur(\mathbf{x}_{adv}) \implies \mathcal{P}ur(\mathbf{x}_{adv}) \approx \mathcal{G}(\mathbf{x}, \mathbf{z})$. Thus, the purification and perturbation masks should be similar. In Fig. 9a, we shows that the two masks are indeed correlated by plotting the Cosine similarity distribution ($\in [-1, 1]$) between $\mathcal{G}(\mathbf{x}, \mathbf{z})$ and $\mathcal{P}(\mathbf{x} + \mathcal{G}(\mathbf{x}, \mathbf{z}))$ for all $9,164$ images in LFW[7]. Therefore, pixels in $\mathbf{x}_{adv}$ involved in the purification process should correspond to those that cause the image to be adversarial in the first place. Fig. 8, highlights that perturbed regions can be automatically localized via constructing a heatmap out of $\mathcal{P}ur(\mathbf{x}_{adv})$. In Fig. 13, we investigate the change in AFR performance (TAR (%) @ 0.1% FAR) of ArcFace under attack (synthesized adversarial faces via $\mathcal{G}(\mathbf{x}, \mathbf{z})$) when the amount of perturbation is varied. We find that (a) minimal perturbation is harder to detect but the purifier incurs minimal damage to the AFR, while, (b) excessive perturbations are easier to detect but increases the challenge in purification.

[7]High similarity may indicate the purifier "memorizing" the purification process for perturbations synthesized via $\mathcal{G}$ while failing to scale to unknown test attacks.

5. Conclusions

With the introduction of sophisticated adversarial attacks on AFR systems, such as geometric warping and GAN-synthesized adversarial attacks, adversarial defense needs to be robust and generalizable. Without utilizing any pre-computed training samples from known adversarial attacks, the proposed *FaceGuard* achieved state-of-the-art generalization performance against 6 different adversarial attacks. *FaceGuard*'s purifier also enhanced ArcFace's recognition performance under adversarial attacks. We are exploring whether an attention mask predicted by the detector can further improve adversarial purification.

References

[1] Debayan Deb, Jianbang Zhang, and Anil K Jain. Advfaces: Adversarial face synthesis. *arXiv preprint arXiv:1908.05008*, 2019. 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 18

[2] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *CVPR*, pages 4690–4699, 2019. 1, 2, 3, 5, 7, 8, 13, 14, 15

[3] Patrick Grother, Mei Ngan, and Kayee Hanaoka. Ongoing face recognition vendor test (frvt). *NIST Interagency Report*, 2018. 1

[4] Daily Mail. Police arrest passenger who boarded plane in Hong Kong as an old man in flat cap and arrived in Canada a young Asian refugee. http://dailym.ai/2UBEcxO, 2011. 1

[5] Yinpeng Dong, Hang Su, Baoyuan Wu, Zhifeng Li, Wei Liu, Tong Zhang, and Jun Zhu. Efficient decision-based black-box adversarial attacks on face recognition. In *CVPR*, pages 7714–7722, 2019. 1

[6] Ali Dabouei, Sobhan Soleymani, Jeremy Dawson, and Nasser Nasrabadi. Fast geometrically-perturbed adversarial faces. In *WACV*, pages 1979–1988, 2019. 1, 2, 3, 5, 6, 12

[7] Haonan Qiu, Chaowei Xiao, Lei Yang, Xinchen Yan, Honglak Lee, and Bo Li. Semanticadv: Generating adversarial examples via attribute-conditional image editing. *arXiv preprint arXiv:1906.07927*, 2019. 1, 2, 3, 5, 6, 12

[8] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting world leaders against deep fakes. In *CVPR Workshops*, pages 38–45, 2019. 1

[9] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 2, 3, 4, 5, 6, 12

[10] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 1, 2, 3, 4, 5, 6, 8, 12

[11] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, pages 2574–2582, 2016. 1, 2, 3, 4, 5, 6, 12

[12] Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. Fawkes: protecting privacy against unauthorized deep learning models. In *USENIX*, pages 1589–1604, 2020. 1

[13] Fabio Valerio Massoli, Fabio Carrara, Giuseppe Amato, and Fabrizio Falchi. Detection of face recognition adversarial attacks. *CVIP*, page 103103, 2020. 2, 3, 4, 6, 11, 12

[14] Anirudh Singh Akshay Agarwal Mayank Vatsa Goel, Akhil and Richa Singh. Smartbox: Benchmarking adversarial detection and mitigation algorithms for face recognition. In *BTAS*, pages 1–7, 2018. 2, 6, 11, 12

[15] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007. 2, 3, 5, 6, 7, 10, 12, 13, 18

[16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *ICLR*, 2017. 2, 4, 6, 7, 12

[17] Yunseok Jang, Tianchen Zhao, Seunghoon Hong, and Honglak Lee. Adversarial defense via learning to generate diverse attacks. In *CVPR*, pages 2740–2749, 2019. 2, 3, 6, 7, 12

[18] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. In *ECCV*, 2018. 2, 3

[19] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *ICLR*, 20178. 2, 3

[20] Guneet S Dhillon, Kamyar Azizzadenesheli, Zachary C Lipton, Jeremy Bernstein, Jean Kossaifi, Aran Khanna, and Anima Anandkumar. Stochastic activation pruning for robust adversarial defense. In *ICLR*, 2018. 2

[21] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017. 2

[22] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*, 2017. 2, 3, 4, 6, 11, 12

[23] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017. 2, 3, 4

[24] Xin Li and Fuxin Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *ICCV*, pages 5764–5772, 2017. 2

[25] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. *arXiv preprint arXiv:1608.00530*, 2016. 2

[26] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*, 2017. 2

[27] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018. 2

[28] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *ICLR*, 2017. 2

[29] Taesik Na, Jong Hwan Ko, and Saibal Mukhopadhyay. Cascade adversarial machine learning regularized with a unified embedding. *ICLR*, 2017. 2

[30] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *ICLR*, 2017. 2

[31] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. Efficient defenses against adversarial attacks. In *ACM Workshop on Artificial Intelligence and Security*, pages 39–49, 2017. 2

[32] Akshay Agarwal, Richa Singh, Mayank Vatsa, and Nalini Ratha. Are image-agnostic universal adversarial perturbations for face recognition difficult to detect? In *BTAS*, pages 1–7, 2018. 2, 3, 4, 6, 11, 12

[33] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017. 2

[34] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *ICML*, 2018. 2

[35] Nicholas Carlini and David Wagner. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017. 2

[36] Marius Mosbach, Maksym Andriushchenko, Thomas Trost, Matthias Hein, and Dietrich Klakow. Logit pairing methods can fool gradient-based attacks. *arXiv preprint arXiv:1810.12042*, 2018. 2

[37] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *ACM Conference on Computer and Communications Security*, pages 135–147, 2017. 2, 6, 7, 12, 13, 15

[38] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *ICLR*, 2018. 2, 6, 7, 12, 13, 15

[39] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *ICLR*, 2017. 2

[40] Muzammal Naseer, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Fatih Porikli. A self-supervised approach for adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 262–271, 2020. 2, 6, 7, 12

[41] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *IJCAI*, 2018. 3

[42] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE Symposium on Security & Privacy*, pages 372–387, 2016. 3

[43] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security & Privacy*, pages 39–57, 2017. 3, 4

[44] Xuanqing Liu and Cho-Jui Hsieh. Rob-gan: Generator, discriminator, and adversarial attacker. In *CVPR*, pages 11234–11243, 2019. 3, 6, 7, 12

[45] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C Duchi, and Percy Liang. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*, 2019. 3

[46] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 3, 4, 5

[47] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv:1411.7923*, 2014. 3, 5, 6, 7, 10, 11

[48] Dingdong Yang, Seunghoon Hong, Yunseok Jang, Tianchen Zhao, and Honglak Lee. Diversity-sensitive conditional generative adversarial networks. *ICLR*, 2019. 4

[49] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 4

[50] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017. 4

[51] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. In *IEEE SPL*, pages 1499–1503, 2016. 10

In this appendix, we conduct thorough experiments to show the benefits of the proposed adversarial generator, detector, and purifier. We include more implementation details on *FaceGuard* and baselines. We also show additional qualitative examples of purified faces and synthesized adversarial faces via our generator.

A. Implementation Details

All the models in the paper are implemented using Tensorflow r1.12. A single NVIDIA GeForce GTX 2080Ti GPU is used for training *FaceGuard* on CASIA-Webface [47] and evaluating on LFW [15]. **Code, pre-trained models and dataset will be publicly available.**

A.1. Preprocessing

All face images are first passed through MTCNN face detector [51] to detect 5 facial landmarks (two eyes, nose and two mouth corners). Then, similarity transformation is used to normalize the face images based on the five landmarks. After transformation, the images are resized to 160×160. Before passing into *FaceGuard*, each pixel in the RGB image is normalized $\in [-1, 1]$ by subtracting 128 and dividing by 128. **All the testing images in the main paper and this supplementary material are from the identities in the test dataset.**

A.2. Network Architectures

The generator, $\mathcal{G}$ takes as input an real RGB face image, $\mathbf{x} \in \mathbb{R}^{160\times160\times3}$ and a 128-dimensional random latent vector, $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ and outputs a synthesized adversarial face $\mathbf{x}_{adv} \in \mathbb{R}^{160\times160\times3}$. Let `c7s1-k` be a 7×7 convolutional layer with k filters and stride 1. `dk` denotes a

4×4 convolutional layer with k filters and stride 2. `Rk` denotes a residual block that contains two 3×3 convolutional layers. `uk` denotes a $2\times$ upsampling layer followed by a 5×5 convolutional layer with k filters and stride 1. We apply Instance Normalization and Batch Normalization to the generator and discriminator, respectively. We use Leaky ReLU with slope 0.2 in the discriminator and ReLU activation in the generator. The architectures of the two modules are as follows:

- Generator:
  `c7s1-64,d128,d256,R256,R256,R256,`
  `u128, u64, c7s1-3,`
- Discriminator:
  `d32,d64,d128,d256,d512.`

A 1×1 convolutional layer with 3 filters and stride 1 is attached to the last convolutional layer of the discriminator for the patch-based GAN loss $\mathcal{L}_{GAN}$.

The purifier, $\mathcal{P}ur$, consists of the same network architecture as the generator:

- Purifier:
  `c7s1-64,d128,d256,R256,R256,R256,`
  `u128, u64, c7s1-3.`

We apply the `tanh` activation function on the last convolution layer of the generator and the purifier to ensure that the generated images are $\in [-1, 1]$. In the paper, we denoted the output of the tanh layer of the generator as an "perturbation mask", $\mathcal{G}(\mathbf{x}, \mathbf{z}) \in [-1, 1]$ and $\mathbf{x} \in [-1, 1]$. Similarly, the output of the tanh layer of the purifier is referred to an "purification mask", $\mathcal{P}ur(\mathbf{x}_{adv}) \in [-1, 1]$ and $\mathbf{x}_{adv} \in [-1, 1]$. The final adversarial image is computed as $\mathbf{x}_{adv} = 2 \times \texttt{clamp}\left[\mathcal{G}(\mathbf{x}, \mathbf{z}) + \left(\frac{\mathbf{x}+1}{2}\right)\right]_0^1 - 1$. This ensures $\mathcal{G}(\mathbf{x}, \mathbf{z})$ can either add or subtract pixels from x when $\mathcal{G}(\mathbf{x}, \mathbf{z}) \neq 0$. When $\mathcal{G}(\mathbf{x}, \mathbf{z}) \to 0$, then $\mathbf{x}_{adv} \to \mathbf{x}$. Similarly, the final purified image is computed as $x_{pur} = 2 \times \texttt{clamp}\left[\left(\frac{\mathbf{x}_{adv}+1}{2}\right) - \mathcal{P}ur(\mathbf{x}_{adv})\right]_0^1 - 1$.

The external critic network, detector $\mathcal{D}$, comprises of a 4-layer binary CNN:

- Detector:
  `d32,d64,d128,d256,fc64,fc1,`

where `fcN` refers to a fully-connected layer with N neuron outputs.

A.3. Training Details

The generator, detector, and purifier are trained in an end-to-end manner via ADAM optimizer with hyperparameters $\beta_1 = 0.5$, $\beta_2 = 0.9$, learning rate of $1e-4$, and batch size 16. Algorithm 1 outlines the training algorithm.

A.4. Baselines

We evaluate all defense methods via publicly available repositories provided by the authors. Only modification made is to replace their training datasets with CASIA-WebFace [47]. We provide the public links to the author codes below:

- Gong *et al.* [22]: https://github.com/gongzhitaao/adversarial-classifier
- UAP-D [32]/SmartBox *et al.* [14]: https://github.com/akhil15126/SmartBox
- Massoli *et al.* [13]: https://github.com/

Algorithm 1 Training *FaceGuard*. All experiments in this work use $\alpha = 0.0001$, $\beta_1 = 0.5$, $\beta_2 = 0.9$, $\lambda_{obf} = \lambda_{fr} = 10.0$, $\lambda_{pt} = \lambda_{perc} = \lambda_{div} = 1.0$, $\epsilon = 3.0$, $m = 16$. For brevity, lg refers to log operation.

Input
  $\mathcal{X}$ Training Dataset
  $\mathcal{F}$ Cosine similarity by AFR
  $\mathcal{G}$ Generator with weights $\mathcal{G}_\theta$
  Dc Discriminator with weights Dc_θ
  $\mathcal{D}$ Detector with weights $\mathcal{D}_\theta$
  $\mathcal{P}ur$ Purifier with weights $\mathcal{P}ur_\theta$
  m Batch size
  α Learning rate
for number of training iterations **do**
  Sample a batch of probes $\{x^{(i)}\}_{i=1}^m \sim \mathcal{X}$
  Sample a batch of random latents $\{z^{(i)}\}_{i=1}^m \sim \mathcal{N}(0, I)$
  $\delta_{\mathcal{G}}^{(i)} = \mathcal{G}((x^{(i)}, z^{(i)})$
  $x_{adv}^{(i)} = x^{(i)} + \delta_{\mathcal{G}}^{(i)}$
  $\delta_{\mathcal{P}ur}^{(i)} = \mathcal{G}((x^{(i)}, z^{(i)})$
  $x_{pur}^{(i)} = x_{adv}^{(i)} - \delta_{\mathcal{P}ur}^{(i)}$

  $\mathcal{L}_{pt}^{\mathcal{G}} = \frac{1}{m}\left[\sum_{i=1}^m \max\left(\epsilon, ||\delta^{(i)}||_2\right)\right]$
  $\mathcal{L}_{obf}^{\mathcal{G}} = \frac{1}{m}\left[\sum_{i=1}^m \mathcal{F}\left(x^{(i)}, x_{adv}^{(i)}\right)\right]$
  $\mathcal{L}_{div}^{\mathcal{G}} = -\frac{1}{m}\left[\sum_{i=1}^m \left[\frac{||\mathcal{G}(\mathbf{x},\mathbf{z}_1)^{(i)} - \mathcal{G}(\mathbf{x},\mathbf{z}_2)^{(i)}||_1}{||\mathbf{z}_1 - \mathbf{z}_2||_1}\right]\right]$
  $\mathcal{L}_{GAN}^{\mathcal{G}} = \frac{1}{m}\left[\sum_{i=1}^m lg\left(1 - Dc(x_{adv}^{(i)})\right)\right]$
  $\mathcal{L}_{\mathcal{D}} = \frac{1}{m}\sum_{i=1}^m \left[lg\mathcal{D}(\mathbf{x}^{(i)}) + lg\left(1 - \mathcal{D}(\mathbf{x}_{adv}^{(i)})\right)\right]$
  $\mathcal{L}_{Dc} = \frac{1}{m}\sum_{i=1}^m \left[lg\left(Dc(x^{(i)})\right) + lg\left(1 - Dc(x_{adv}^{(i)})\right)\right]$
  $\mathcal{L}_{perc}^{\mathcal{P}ur} = \frac{1}{m}\sum_{i=1}^m \left[||x_{pur} - x||_1 + ||\mathcal{P}ur(x_{adv}^{(i)})||_1\right]$
  $\mathcal{L}_{fr}^{\mathcal{P}ur} = -\frac{1}{m}\left[\sum_{i=1}^m \mathcal{F}\left(x^{(i)}, x_{pur}\right)\right]$
  $\mathcal{L}_{bf}^{\mathcal{P}ur} = \frac{1}{m}\left[\sum_{i=1}^m lg\left(1 - \mathcal{D}(x_{pur})\right)\right]$
  $\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{GAN}^{\mathcal{G}} + \lambda_{obf}\mathcal{L}_{obf} + \lambda_{pt}\mathcal{L}_{pt} + \lambda_{div}\mathcal{L}_{div}$
  $\mathcal{L}_{\mathcal{P}ur} = \lambda_{fr}\mathcal{L}_{fr} + \lambda_{perc}\mathcal{L}_{perc} + \lambda_{bf}\mathcal{L}_{bf}$
  $\mathcal{G}_\theta = \texttt{Adam}(\nabla_{\mathcal{G}}\mathcal{L}^{\mathcal{G}}, \mathcal{G}_\theta, \alpha, \beta_1, \beta_2)$
  $Dc\theta = \texttt{Adam}(\nabla_{Dc}\mathcal{L}^{Dc}, Dc_\theta, \alpha, \beta_1, \beta_2)$
  $\mathcal{D}_\theta = \texttt{Adam}(\nabla_{\mathcal{D}}\mathcal{L}^{\mathcal{D}}, \mathcal{D}_\theta, \alpha, \beta_1, \beta_2)$
  $\mathcal{P}ur_\theta = \texttt{Adam}(\nabla_{\mathcal{P}ur}\mathcal{L}^{\mathcal{P}ur}, \mathcal{P}ur_\theta, \alpha, \beta_1, \beta_2)$
end for

	Known			Unseen		
	FGSM [9]	**PGD [10]**	**DeepFool [11]**	**AdvFaces [1]**	**GFLM [6]**	**SemanticAdv [7]**
Gong *et al.* [22]	94.51	92.21	94.12	68.63	50.00	50.21
UAP-D [32]	63.65	69.33	56.38	60.81	50.12	50.28
SmartBox [14]	58.79	62.53	51.32	54.87	50.97	62.14
Massoli *et al.* [13] (MLP)	78.35	82.52	91.21	55.57	50.00	50.00
Massoli *et al.* [13] (LSTM)	74.61	86.43	94.73	62.43	50.00	50.00

(a)

	Known			Unseen		
	AdvFaces [1]	**GFLM [6]**	**SemanticAdv [7]**	**FGSM [9]**	**PGD [10]**	**DeepFool [11]**
Gong *et al.* [22]	81.39	96.72	98.97	84.46	57.00	72.32
UAP-D [32]	68.78	54.31	77.46	51.64	50.32	52.01
SmartBox [14]	54.87	50.97	62.14	58.79	62.53	51.32
Massoli *et al.* [13] (MLP)	77.64	86.54	94.78	55.20	51.32	52.90
Massoli *et al.* [13] (LSTM)	81.42	92.62	96.76	52.74	65.43	54.84

(b)

	Known					
	FGSM [9]	**PGD [10]**	**DeepFool [11]**	**AdvFaces [1]**	**GFLM [6]**	**SemanticAdv [7]**
Gong *et al.* [22]	98.94	97.91	95.87	92.69	**99.92**	**99.92**
UAP-D [32]	61.32	74.33	56.78	51.11	65.33	76.78
SmartBox [14]	58.79	62.53	51.32	54.87	50.97	62.14
Massoli *et al.* [13] (MLP)	63.58	76.28	81.78	88.38	51.97	52.98
Massoli *et al.* [13] (LSTM)	71.53	76.43	88.32	75.43	53.76	55.22
	Unseen					
Proposed FaceGuard	**99.85**	**99.85**	**99.85**	**99.84**	99.61	99.85

(c)

Table 5: Detection accuracy of SOTA adversarial face detectors in classifying six adversarial attacks synthesized for the LFW dataset [15] under various known and unseen attack scenarios. Detection threshold is set as 0.5 for all methods.

fvmassoli/trj-based-adversarials-detection

- Adversarial Training [16]: https://github.com/locuslab/fast_adversarial
- Rob-GAN [44]: https://github.com/xuanqing94/RobGAN
- L2L [17]: https://github.com/YunseokJANG/l2l-da
- MagNet [37]: https://github.com/Trevillie/MagNet
- DefenseGAN [38]: https://github.com/kabkabm/defensegan
- NRP [40]: https://github.com/Muzammal-Naseer/NRP

Attacks are also synthesized via publicly available author codes:

- FGSM/PGD/DeepFool: https://github.com/tensorflow/cleverhans
- AdvFaces: https://github.com/ronny3050/AdvFaces
- GFLM: https://github.com/alldbi/FLM
- SemanticAdv: https://github.com/AI-secure/SemanticAdv

B. Overfitting in Prevailing Detectors

In Tab. 5, we provide the detection rates of prevailing SOTA detectors in detecting six adversarial attacks in LFW [15] when they are trained on different attack subsets. We highglight the overfitting issue when (a) SOTA detectors are trained on gradient-based adversarial attacks (FGSM [9], PGD [10], and DeepFool [11]) and tested on gradient-based and learning-based attacks (AdvFaces [1], GFLM [6], and SemanticAdv [7]), and (b) vice-versa. Tab. 5(c) reports the detection performance of SOTA detectors when all six attacks are available for training.

We find that detection accuracy of SOTA detectors significantly drops when tested on a subset of attacks not encountered during their training. Instead, the proposed *FaceGuard* maintains robust detection accuracy without even training on the pre-computed samples from any known attacks.

C. Qualitative Results

C.1. Generator Results

Fig. 10 shows examples of synthesized adversarial faces via the proposed adversarial generator $\mathcal{G}$. Note that the generator takes the input prob $\mathbf{x}$ and a random latent $\mathbf{z}$. We show synthesized perturbation masks and corresponding adversarial faces for three randomly sampled latents.

We observe that the synthesized adversarial images evades ArcFace [2] while maintaining high structural similarity between adversarial and input probe.

C.2. Purifier Results

We show examples of purified images via *FaceGuard* and baselines including MagNet [37] and DefenseGAN [38] in Fig. 11. We observe that, compared to baselines, purified images synthesized via *FaceGuard* are visually realistic with minimal changes compared to the ground truth real probe. In addition, compared to the two baselines, *FaceGuard*'s purifier protects ArcFace [2] matcher from being evaded by the six adversarial attacks.

D. Additional Results on Purifier

D.1. Perturbation and Purification Masks

In the main text, we found that the perturbation and purification masks are correlated with an average Cosine similarity of 0.52. We show five pairs of perturbation and purification masks ranked by the Cosine similarity between them (highest to lowest). We observe that purification mask is better correlated when perturbations are more local. Slightly perturbing entire faces poses to be challenging for the proposed purifier.

D.2. Effect of Perturbation Amount

We also studied the effect of perturbation amount on detection and purification results in the main text. We observed a trade-off between detection and purification with respect to perturbation magnitudes. With minimal perturbation, detection is challenging while purifier maintains AFR performance. Excessive perturbations lead to easier detection with greater challenge in purification. In Fig. 13, show examples of synthesized adversarial faces for different perturbation amounts and their corresponding purified images. We find that detection scores improve with larger perturbation. Aligned with our earlier findings, due to the proposed bonafide loss, $\mathcal{L}_{bf}$, purified faces are continuously detected as real by the detector which explains why the purifier maintains AFR performance with increasing perturbation amount.

D.3. Effect of Purification on ArcFace Embeddings

In order to investigate the effect of purification on a matcher's feature space, we extract face embeddings of real images, their corresponding adversarial images via the challenging AdvFaces [1] attack, and purified images, via the SOTA ArcFace matcher. In total, we extract feature vectors from $1,456$ face images of 10 subjects in the LFW dataset [15]. In Fig. 14, we plot the 2D t-SNE visualization of the face embeddings for the 10 subjects. The identity clusterings can be clearly observed from real, adversarial, and purified images. In particular, we observe that some adversarial faces pertaining to a subject moves farther from its identity cluster while the proposed purifier draws them back. Fig. 14 illustrates that the proposed purifier indeed enhances face recognition performance of ArcFace under attack from 34.27% TAR @ 0.1% FAR under no defense to 77.46% TAR @ 0.1% FAR.

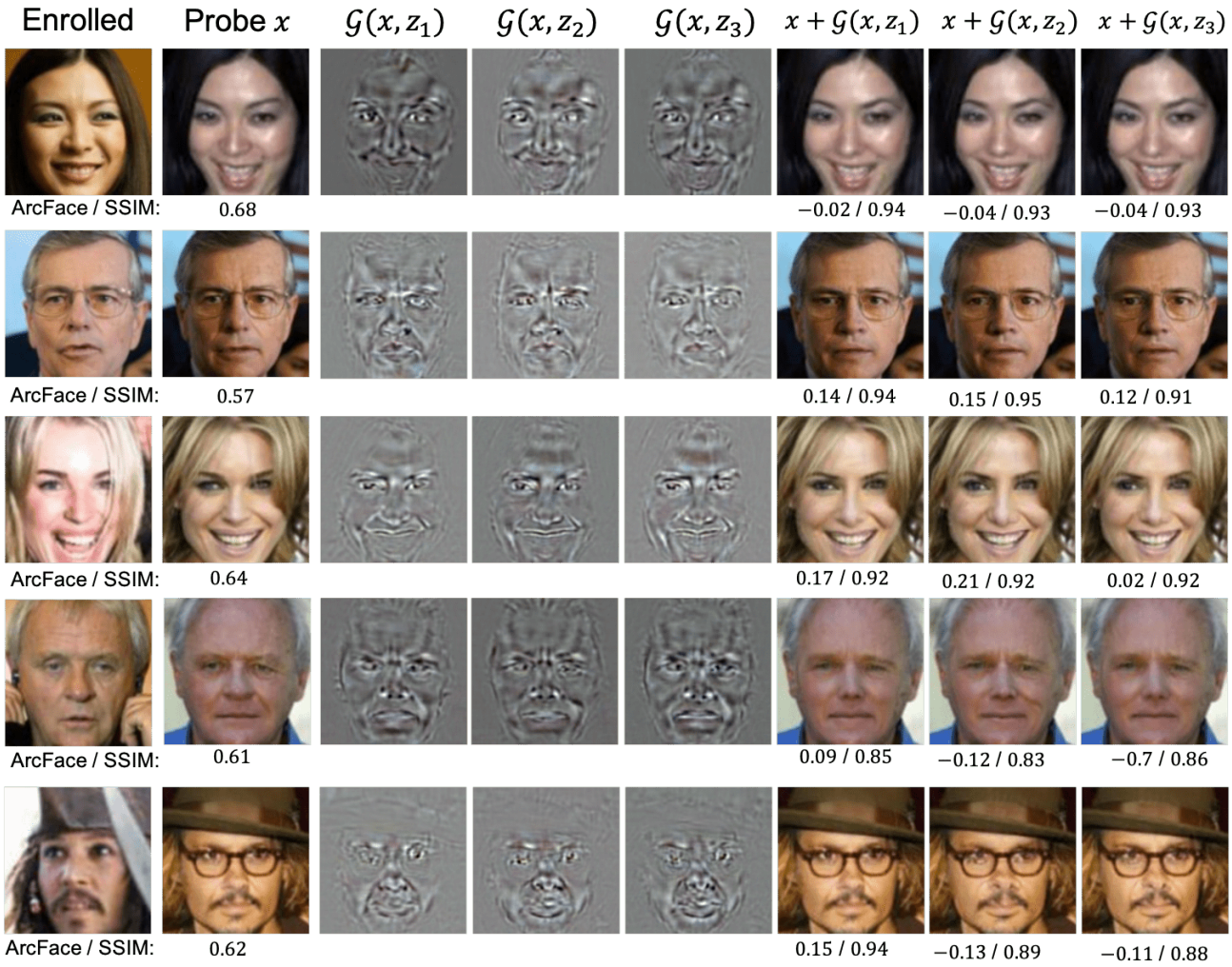


Figure 10: Examples of generated adversarial images along with corresponding perturbation masks obtained via *FaceGuard*'s generator $\mathcal{G}$ for three randomly sampled $\mathbf{z}$. Cosine similarity scores via ArcFace [2] $\in [-1, 1]$ and SSIM $\in [0, 1]$ between synthesized adversarial and input probe are given below each image. A score above **0.36** (threshold @ 0.1% False Accept Rate) indicates that two faces are of the same subject.

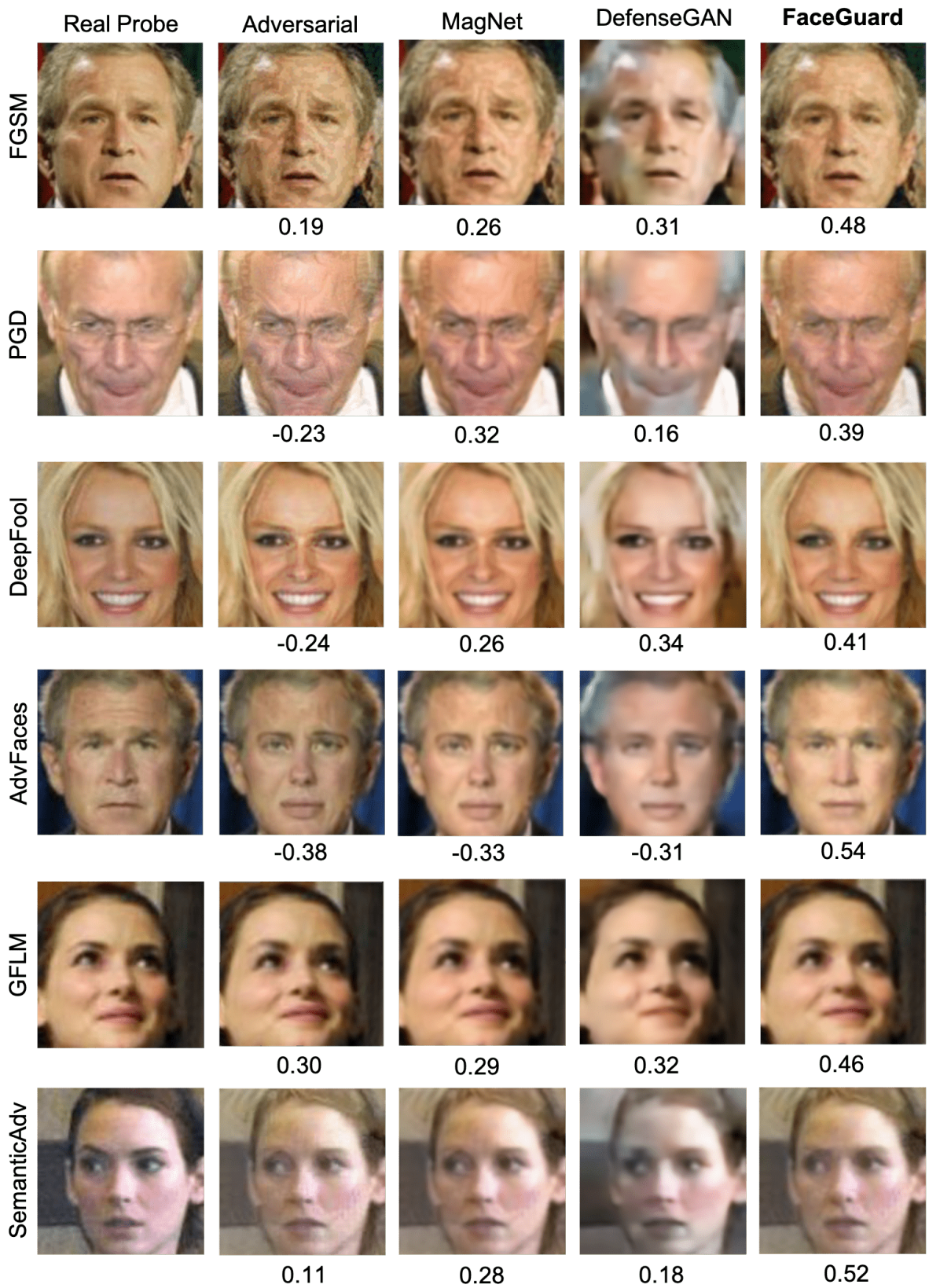


Figure 11: Examples of purified images via MagNet [37], DefenseGan [38], and proposed *FaceGuard* purifiers for six adversarial attacks. Cosine similarity scores via ArcFace [2] $\in [-1, 1]$ are given below each image. A score above **0.36** (threshold @ 0.1% False Accept Rate) indicates that two faces are of the same subject.

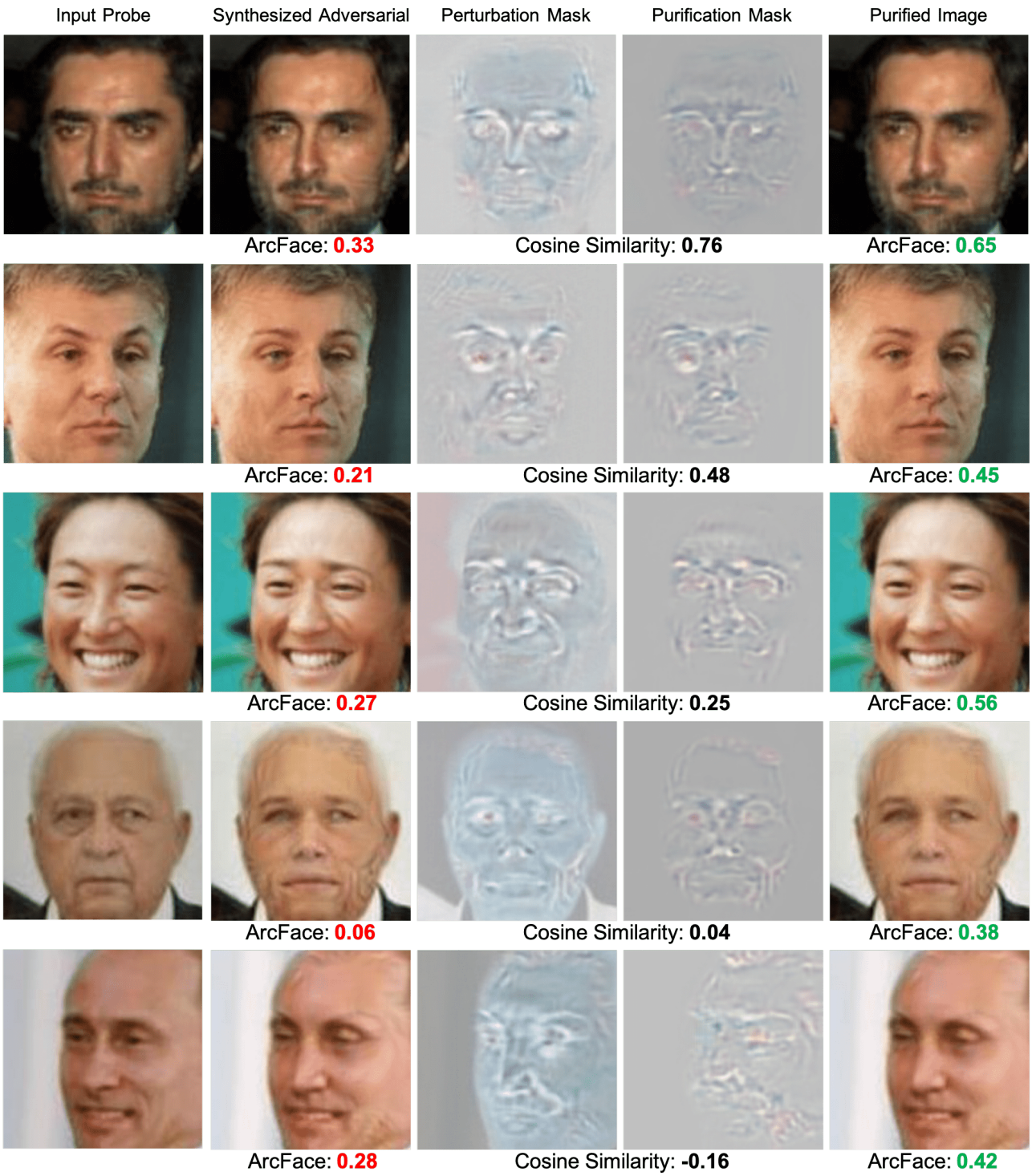


Figure 12: Examples of synthesized adversarial images via the proposed adversarial generator and corresponding purified images. Cosine similarity between perturbation and purification masks given below each row along with ArcFace scores between synthesized adversarial/purified image and real probe. A score above **0.36** (threshold @ 0.1% False Accept Rate) indicates that two faces are of the same subject. Even with lower correlation between perturbation and purification masks (rows 3-5), the purified images can still be identified as the correct identity. Notice that the purifier primarily alters the eye color, nose, and subdues adversarial perturbations in foreheads. Zoom in for details.

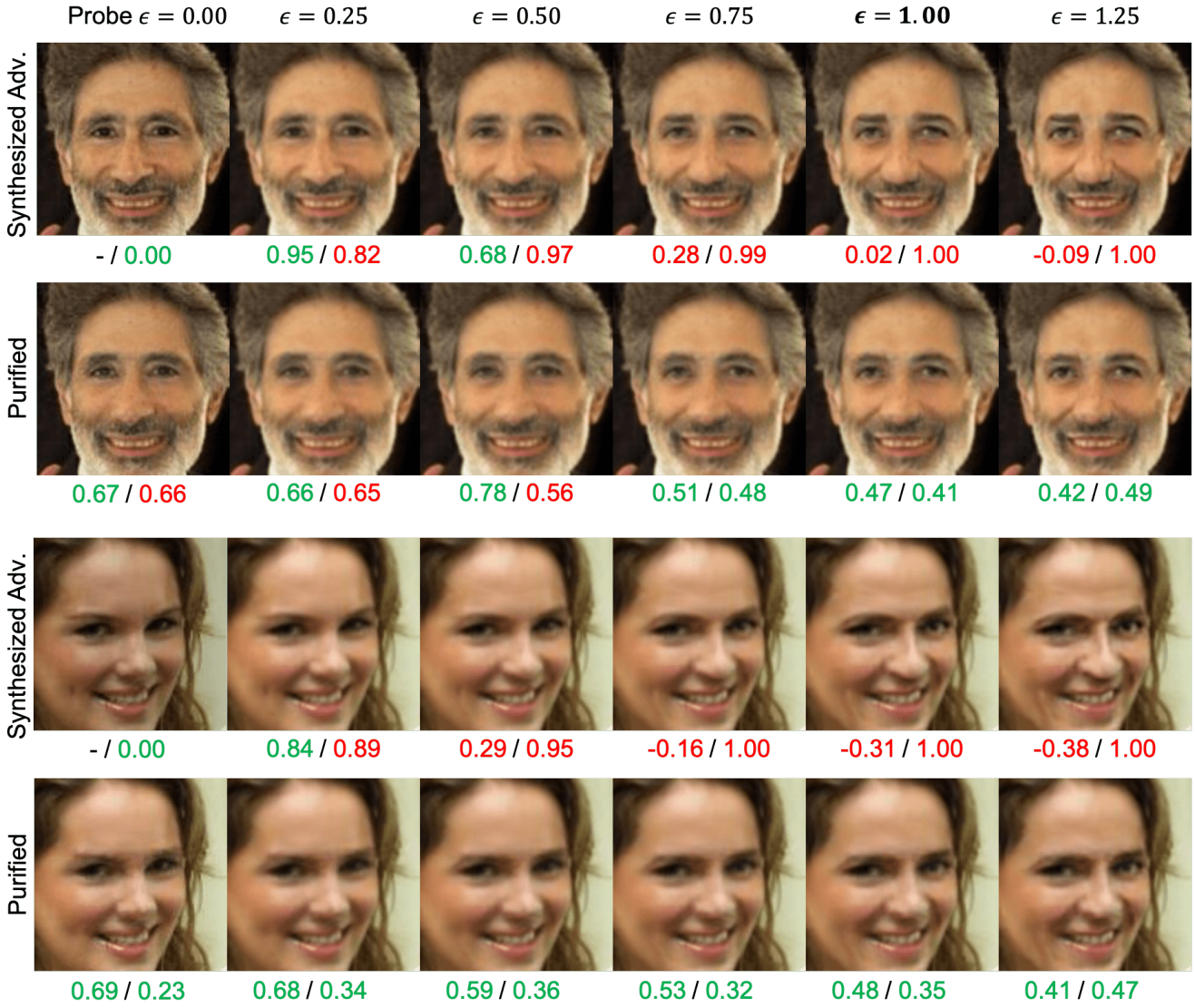


Figure 13: ArcFace $\in [-1, 1]$ / Detection scores $\in [0, 1]$ when perturbation amount is varied ($\epsilon = \{0.25, 0.50, 0.75, 1.00, 1.25\}$). Detection scores above 0.5 are predicted as adversarial images while ArcFace scores above **0.36** (threshold @ 0.1% False Accept Rate) indicate that two faces are of the same subject. *FaceGuard* is trained on $\epsilon = 1.00$. The detection scores improve as perturbation amount increases, whereas, majority of purified images are detected as real. Even when purified images fail to be classified as real by the detector, purification maintain high AFR performance.

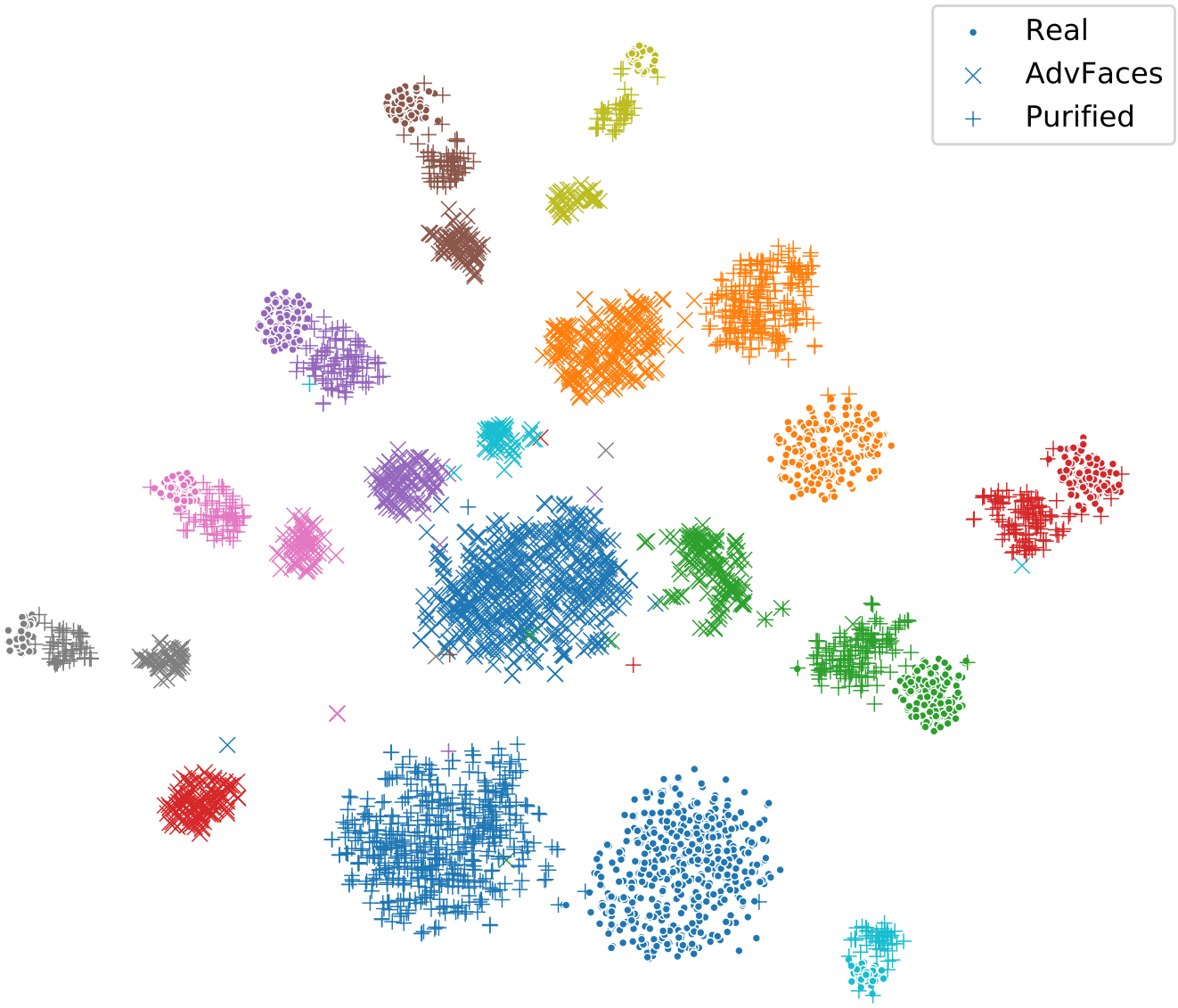


Figure 14: 2D t-SNE visualization of face representations extracted via ArcFace from 1,456 (a) real, (b) AdvFaces [1], and (c) purified images belonging to 10 subjects in LFW [15]. Example AdvFaces [1] pertaining to a subject moves farther from its identity cluster while the proposed purifier draws them back.