Biometric Backdoors: A Poisoning Attack Against Unsupervised Template Updating

Giulio Lovisotto, Simon Eberz, Ivan Martinovic
University of Oxford
firstname.lastname@cs.ox.ac.uk

ABSTRACT

In this work, we investigate the concept of biometric backdoors: a template poisoning attack on biometric systems that allows adversaries to stealthily and effortlessly impersonate users in the long-term by exploiting the template update procedure. We show that such attacks can be carried out even by attackers with physical limitations (no digital access to the sensor) and zero knowledge of training data (they know neither decision boundaries nor user template). Based on the adversaries' own templates, they craft several intermediate samples that incrementally bridge the distance between their own template and the legitimate user's. As these adversarial samples are added to the template, the attacker is eventually accepted alongside the legitimate user. To avoid detection, we design the attack to minimize the number of rejected samples.

We design our method to cope with the weak assumptions for the attacker and we evaluate the effectiveness of this approach on state-of-the-art face recognition pipelines based on deep neural networks. We find that in scenarios where the deep network is known, adversaries can successfully carry out the attack over 70% of cases with less than ten injection attempts. Even in black-box scenarios, we find that exploiting the transferability of adversarial samples from surrogate models can lead to successful attacks in around 15% of cases. Finally, we design a poisoning detection technique that leverages the consistent directionality of template updates in feature space to discriminate between legitimate and malicious updates. We evaluate such a countermeasure with a set of intra-user variability factors which may present the same directionality characteristics, obtaining equal error rates for the detection between 7-14% and leading to over 99% of attacks being detected after only two sample injections.

KEYWORDS

biometrics; face recognition; adversarial machine learning; poisoning attacks

1 INTRODUCTION

In recent years, biometric authentication has become one of the preferred ways to mitigate burdens associated with passwords (e.g., re-use, inconvenient creation policies and bad memorability). With a long history of research, face and fingerprint recognition are the most popular modalities and authentication systems based on them are commonly delivered with consumer products. While early research focused on the performance of these modalities under a zero-effort threat model, current trends in biometric systems are also prioritizing high protection of biometric *templates*, i.e., the users stored biometric information. Templates in fact represent sensitive user data and their leak might compromise the secrecy of the biometric trait in a permanent way. Different measures can be used to protect user templates: standard file encryption, template protection algorithms (e.g., cancelable biometrics [26], key generation [21]), secure elements, distributed template storage. Nowadays, most commercial biometric systems use any combinations of these protection measures. As an example, both Apple's TouchID and FaceID store templates in the protected secure enclave and the data never leaves the user device [2].

Due to these protection mechanisms, attackers can not modify the template directly, since that would require them to compromise the device. However, an adversary can exploit the *template update* procedure to cause the system to adapt the template, thereby indirectly modifying the data within. In fact, during template update, the system replaces or adds recently seen samples to the user template. Template update allows systems to cope with the inherent variability of biometric traits, such as physiological changes or sensor measurement noise, by collecting additional user samples.

Biometric systems use either supervised or unsupervised updating strategies. Supervised updating means that additional guarantees in the user identity are required before the update (e.g., Windows Hello requires the users to re-input their PIN or password before template update). From a usability perspective, unsupervised updating represents a more desirable choice, as it is a seamless approach that does not require additional user interaction. In both scenarios, the confidence in the user identity can be strengthened with the introduction of a *self-update* threshold: a new sample for the update is discarded if it does not match a confidence threshold (e.g., needs to be sufficiently similar to the current user template). For example, FaceID automatically augments the user template with a new sample after a successful authentication if the quality of the match is above a threshold [2].

By exploiting this template update procedure, an attacker can inject adversarial samples into the user template to carry out a *template poisoning* attack. In this attack, the adversary corrupts the user template allowing himself to impersonate the user with his own biometric trait. The poisoned template is hard to detect and creates an inconspicuous and stealthy backdoor for the adversary in the long-term. Once placed, the backdoor allows the adversary to access the system without requiring them to modify their appearance. However, the adversary needs to overcome four key challenges: (i) he is has limited control over the injected samples, (ii) all injected samples must clear the update threshold, (iii) he has limited knowledge of the authentication system and the legitimate user's template, (iv) he must avoid degrading the legitimate user's experience to avoid generating suspicion.

In our analysis, we focus on face recognition as it is one of the most well-known and widely used, including in unsupervised environments. Faces show have inherent variability caused by changes

in lighting environment, sensor position and user behavior. In this paper, we present a method to carry out template poisoning attacks on biometric systems based on deep machine learning pipelines. We evaluate our attack on state-of-the-art systems for face recognition, showing that the attack is feasible even for adversaries with limited knowledge and capabilities, including against black-box models. Afterwards, we propose a new countermeasure for the detection of poisoning attacks based on the angular similarity of samples. Compared to previous work [5, 16], we also evaluate the detection taking into account the trade-offs with legitimate template updates, showing that it can effectively stop poisoning attacks.

Contributions. The contributions of this paper are as follows:

- We propose a method to plant *biometric backdoors* by poisoning the legitimate user's template in minimal knowledge and limited injection capabilities scenarios.
- We evaluate the attack on state-of-the-art recognition pipelines, including white- and black-box models. We show that the error rates of the system hardly change when such a backdoor is present, making the attack inconspicuous.
- We introduce a poisoning detection method that thwarts poisoning attacks without affecting legitimate template updates, and we investigate these trade-offs on a large face dataset.

The rest of the paper is organised as follows: Section 2 describes the related work, Section 3 outlines the system and threat model assumed in the paper. Section 4 outlines the concept of the template poisoning attack and our methodology is given in Section 5. The results are given in Section 6. We introduce a new poisoning detection method in Section 7 and conclude the paper in Section 8.

2 RELATED WORK

In this section, we first give a brief background on biometric recognition, we then discuss attacks on biometric systems, and then given an overview of the existing work in poisoning attacks.

2.1 Biometric Recognition

Both the research community and industry have shown significant interest in biometric recognition in recent years. Often times, recognition of biometric traits is proposed as an approach to mitigate the shortcomings of passwords (such as bad memorability, password reuse and easy to guess passwords). Previous research has shown that most biometrics present inherent intra-user variability, both for physiological and behavioral traits. In the following we will give a brief overview of face recognition systems, and how intra-user variability affects the system.

Face Recognition. Modern face recognition systems are based on deep convolutional neural networks (DNN), where the input image undergoes transformation over several convolutional layers (e.g., FaceNet [28], VGG-Face [25], ResNet [7]). In recent years, state-of-the-art models have shown to outperform humans. Most face recognition DNN can also be used as feature extractors, allowing them to work with faces unseen in the training data. In this case, another simpler classifier is added onto the recognition pipeline and uses the output of the neural network as its inputs.

Template Update. The physiological trait (face) itself does not show significant intra-user variations, but when using image-based recognition, a series of factors may influence its appearance to the sensor. Well known variation factors include age, pose (viewing angle), lighting environment, facial hair and facial accessories. Some of these variations can be accounted for at enrolment, for example asking the user to rotate their head as some samples are collected or shining additional light onto the users face to account for the effect of external lighting conditions. However, systems also increasingly rely on template updating. As an example, both Apple's FaceID [2] and TouchID [1] perform template update procedures. FaceID performs template updates either after a new sample is accepted, or when a rejected sample is followed promptly by a correctly entered backup passcode.

2.2 Adversarial ML Attacks

Adversarial machine learning has become extremely relevant due to the wide-spread use of deep neural networks, which are prone to being fooled by purposefully crafted adversarial samples. Adversarial ML attacks have been classified into two main categories: *evasion* (*inference-time*) attacks and *poisoning* (*training-time*) attacks [24]. In evasion attacks, the adversary attempts to craft a sample that is classified as belonging to a desired output class. In poisoning attacks, the adversary injects adversarial samples into the training data in order to maximize an objective function (typically to cause misclassifications at test time).

Realizability of Adversarial Attacks. Adversarial samples have been initially investigated in the digital domain, where the adversary is able to create pixel-perfect modifications to the inputs. Such perturbations rarely survive in the physical world, as a series of factors affects the sensor measurements. As a result, these modifications are not evident to the underlying recognition system. For a camera, factors such as viewing angle, resolution, distance have all been shown to affect the measurements enough to severely harm the effectiveness of adversarial samples. However, recent studies have shown that adversarial samples can also be constructed to survive in the physical world. As an example, Sharif et al [29] have shown how a carefully crafted frame of glasses can be worn by an individual to dodge face detection or even impersonate arbitrary users. Their work is further extended in [30], where the authors use generative adversarial networks to improve the success rate and the transferability across models of the attack. Other known real-world attacks include adversarial patches [6, 15] or posters and graffiti [27, 33], where maliciously printed images can be applied on top of objects so that the sensors inputs contain the added image, order to deceive the classifier (either object or traffic sign classification).

Poisoning Attacks. Differently from evasion, in poisoning attacks the goal is to modify the classifier to misbehave on specific inputs that are known by the adversary without compromising its performance on regular inputs [13, 18]. In biometric systems, poisoning attacks may be categorized into two different categories: (i) poisoning the DNN, (ii) poisoning the user template. When the DNN is the target of the poisoning, there is no assumption on controlling the DNN inputs, but rather the adversary has an increased control over the training phase, i.e., can directly edit the network weights [18] or change the training data [13]. However, as DNN are increasingly used as black-box feature extractors in a classification

Work	template update	limited injection capabilities	unknown template	state-of-the-art recognition	unknown network	stealthiness	uncompromised training phase
Kloft et al. [16]	✓	✗	–	–	✗	–	✓
Biggio et al. [3]	✓	✗	✗	✗	✓	✗	✓
Biggio et al. [4]	✓	✗	✗	✗	✗	✗	✓
Garofalo et al. [9]	✗	✗	✗	✓	✗	✗	✗
Liu et al. [18]	✗	–	✓	✓	✗	✓	✗*
This paper	✓	✓	✓	✓	✓	✓	✓

*network weights are changed after training

Table 1: Comparison with previous work. A dash (–) indicates that the property does not apply directly to the work.

pipeline rather than directly as classifiers, analysis of poisoning of the template database needs to be included in the security analysis. As mentioned in introduction, template update represents an opportunity for malicious samples to reach the template database. A preliminary analysis of template poisoning attacks on biometric systems was proposed by Biggio et al. [4]. The authors present a poisoning attack where an adversary injects a set of samples that gradually shifts the user template towards the biometric trait of the adversary. The same authors extended their work reducing the knowledge assumptions of the adversary, showing that the attack is feasible even considering a black-box scenario and limited knowledge of the victim template [3].

In the following we motivate how this paper builds on the assumptions of these works to account for the challenges that an adversary would face in practice in carrying out a poisoning attack.

2.3 Differences from Previous Work

We use the previous work in physically-robust adversarial samples as one of the building blocks of our work, which allows us to control and carry out the poisoning attack as a whole. We report a comparison to previous work in Table 1, where we focus on the following properties.

Template Update. We consider that template updates are limited by self-update thresholds. However, we consider penalties for consecutive rejected authentication attempts. This is a reasonable security policy as sequences of failed authentication attempts might correspond to attackers trying to impersonate the user, and falling back to a different authentication method can thwart such attempts (e.g., FaceID allows five failed attempts before switching to PIN input). In comparison, previous works [3, 4, 16] did not consider or investigate the frequency of rejected attempts.

Limited Injection Capabilities. We consider only cases that account for the physical realizability of the adversarial sample creation and injection. Adversarial examples have been shown to be realizable in the real-world [17, 27, 29], but their flexibility and effectiveness is greatly reduced in that case. Assuming perfect injection capabilities, where the adversary can freely inject samples and *digitally* manipulate exactly each pixel in input is an unrealistic assumption in our scenario. In fact, being able to feed arbitrarily manipulated samples to the system, which bypass liveness detection (such as the images in [3, 4, 9]) would mean that the adversary could simply be able to authenticate inconspicuously at any time just by having a printed image of the user. This defeats the purpose of carrying out a more complex poisoning attack.

Unknown Template. Previous work focused on full template knowledge [4, 9]. Biggio et al. [3] considered a partial knowledge scenario where the known sample is chosen as the closest to user's centroid out of *all* the samples in the testset. In a biometric system, this assumption would require the adversary to have availability of several samples from the user, which might not be easily obtainable. Therefore, we focus on an adversary that does *not* have any knowledge of the template stored in the user device, but only has a single image of the user.

Unknown Network. While we start our analysis with a white-box scenario, we extend it to black-box ones using the principle of transferability of adversarial samples [11, 24, 35]. A black-box attack was shown to be feasible in [3], however the system relied on non-state-of-the-art face recognition methods (i.e., EigenFaces [36]) that are nowadays greatly outperformed by deep neural networks.

Stealthiness. We consider a poisoning attack to be *stealthy* when it can be carried out without compromising the normal error rates of the system. This is a desirable property as changes in false acceptances or false rejections (i.e., legitimate user cannot authenticate, other users can authenticate) would be suspicious: users might stop using the biometric system or re-enroll their template in an attempt to reset the system performance. This approach separates us from previous work, such as [3, 4, 9], which aims to *replace*, rather than alter the legitimate user's template, blocking them out of the system. With enrolment being a procedure that the user can carry out independently, such attacks would not guarantee long-term access to the adversary, as victims would re-enrol into the system as soon as they are inconvenienced by the reduced recognition rates.

Uncompromised Training Phase. Unlike previous work, we assume no control over training data [9] or network weights [18]. While these assumptions may lead to more effective attacks, they also require the adversary to be able to affect the network training process. In our scenario, it is reasonable that a network would be trained beforehand, and would not be accessible once on the device (e.g., Apple FaceID is stored and executed in a secure element).

These properties lead our experimental design and evaluation. Our goal is to bridge gap between a purely theoretical poisoning attack and the challenges that an adversary would have to face in practice, allowing us to better understand the feasibility of such attacks in the real-world.

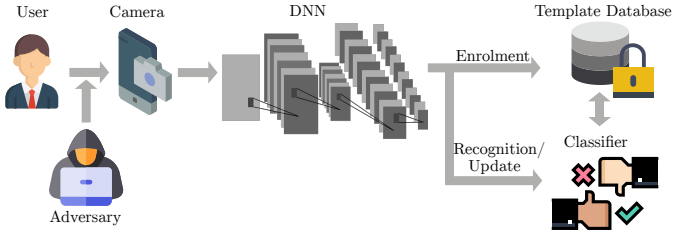


Figure 1: System and threat model. The adversary has physical realizability constraints, meaning that the adversarial samples need to survive through the sensor measurement. Captured images undergo a feature extraction step where a deep neural network produces a multidimensional output. The samples' features are then used for enrolment, and a classifier processes them during recognition and update.

3 SYSTEM AND THREAT MODEL

In this section we first give an overview of the system model and later talk about the considered adversary.

3.1 System Model

Modern biometric recognition is separated in four phases: development, enrolment, recognition and template updating.

Development. During development, a deep neural network is trained on a large labelled dataset of biometric samples. The objective of the DNN is to learn how to compute features from user samples so that the extracted features lie in a reduced-dimensionality linearly separable hyperspace. The extracted features can be either *embeddings* [28] when triplet loss is used for training, or they can be the features extracted from the layer preceding the last softmax layer (*logits*). This pre-trained network is then deployed with the biometric system.

Enrolment. At enrolment, the users present their biometric traits to the sensor. One or more measurements are taken, processed and fed into the DNN. The resulting samples are stored in the template database, and are known as the *user template*. These samples are kept in a secured space in the user device and are used as a reference whenever an authentication attempt is made. An illustration of this process is given in Figure 1.

Recognition/Update. During the recognition phase, whenever a user presents his trait to the sensor, the system measures it, feeds it into the DNN to extract the features and attempts to match the resulting sample and with the known biometric template with a threshold-based classifier. If the match occurs, the user is successfully authenticated. In the update procedure the system might decide to add the newly seen sample into the template database if it is confident enough that the sample belongs to the legitimate user. We outline our choices for how this process works in Section 6.1.

3.2 Adversary Model

The adversaries overarching goal is to place a "biometric backdoor" which grants them stealthy long-term access to the secured system without requiring further effort once the backdoor is in place. In this section, we define the attacker's objectives, knowledge and capabilities.

Objectives. The attacker's goals are to:

- Cause modifications of the user template that leads to the attacker being accepted.
- Maintain the stealthiness of the attack, i.e., minimizing the changes to false rejects (FRR) and false accepts (FAR).
- Minimize the number of physical accesses to the system required to plant the backdoor.
- Minimize the number of samples rejected by the recognition system.

ASSUMPTIONS

Capabilities. We assume that the adversary has physical access to the sensor and can therefore feed samples to it. The adversary does not have digital access to the sensor, as in that case he would be able to perfectly and effortlessly control the inputs, making a poisoning attack unnecessary. The adversary can therefore alter biometric sensor measurements only in the physical domain, and these will be subject to both sensor and presentation noise. Unlike previous work, we do *not* assume that the attacker has access to the system during enrolment or during any subsequent replacements of the complete template. The adversary is also unable to directly change the template or training data (e.g., by removing or replacing user samples).

Template Knowledge. In line with related work, we assume that the adversary has at least one picture of the user's face. Being a physiological feature, face appearance is notoriously difficult to keep secret. Social media in particular is a plentiful source for videos or photographs [37]. Differently from previous work, where the best matching picture was chosen out of the test data [3], we do not set any requirement on the known picture other than it being accepted by the system. This assumption entails two obstacles for the adversary: he does not know enrolment data nor the resulting decision boundaries of the system's classifier.

System Knowledge. As our goal is to evaluate the effectiveness of poisoning attacks and construct possible defenses, we first assume a strong attacker with white-box access, which has access to the neural network after it has been trained [23]. The adversary additionally knows what algorithms are being used by the recognition pipeline. We further investigate a black-box attacker using the principle of transferability of adversarial samples [22]. In this case, the adversary has knowledge of a surrogate model that he can use to optimize the attack on the black-box network.

4 ATTACK CONCEPT

Using the assumptions made in the system and threat model, we present an overview of the attack and of its challenges.

4.1 Overview

The concept behind the poisoning attack is that the adversary adds adversarial samples to the legitimate user template in order to change the decision boundary of the classifier. Figure 2 shows a two-dimensional representation of how the attack works. There are three categories of samples:

- *user (victim) samples*: legitimate user samples;

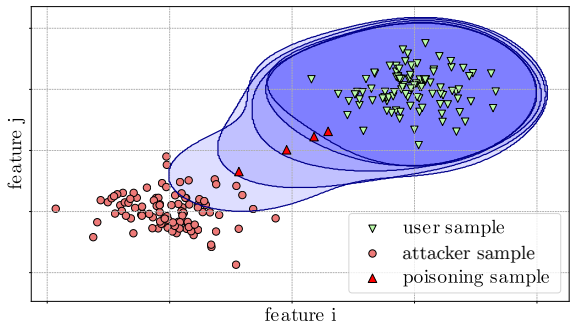


Figure 2: Changes in the classifier decision boundary (shaded areas) with the addition to the template of adversarial crafted samples. With enough poisoning samples, the classifier will recognize the adversary as the legitimate user.

- *attacker samples*: samples coming from the biometric trait of the adversary;
- *poisoning samples*: samples algorithmically crafted by the adversary.

Figure 2 shows how the user and attacker samples are well separated in the feature space, due to the uniqueness of their biometric traits. At enrolment, the classifier learns the distribution of the user samples creating a boundary around it, shown by the darker blue area. The classifier is able to correctly discriminate between attacker and user samples, rejecting the adversary in an impersonation attempt.

Knowing his own template and a user sample as the starting point, the adversary crafts the poisoning samples accordingly. As the self-update threshold is in place, the adversary must make sure that the crafted samples lie within the current accepted region (shaded blue area), otherwise they would be rejected as anomalous. By injecting one poisoning sample at a time, the adversary shifts the decision boundary towards his own sample distribution. With sufficient poisoning samples, the adversary will move the decision boundary enough so that his own samples will fall inside it, and can therefore impersonate the user with his own trait.

4.2 Challenges For Attacker

The challenges in designing the attack are based on the limited injection capabilities (which bring input uncertainty) and on the unknown user template and location of decision boundaries (which cause failures in the injection).

Input Uncertainty. Figure 2 shows a simplified 2-dimensional example of the user distribution. This distribution is actually in an n-dimensional feature space that consists of hundreds of dimensions. If the adversary had exact control over the location of samples in this space, he could simply craft and inject the poisoning samples that lie on the n-dimensional vector between the user and adversary distribution centroids. However, the adversary only controls the raw biometric measurements (input to the sensor) and furthermore needs to deal with the uncertainty that inevitably occurs with the biometric measurement (i.e., intra-user variation and input noise). Considering the fact that adversaries cannot alter the DNN, this further restricts the domain of available input manipulations.

Failed Injections. As adversaries only know one victim sample that is not part of the template, they need to overcome the uncertainty brought by the unknown location of the decision boundary. In fact, the adversary has no information about the training data and consequently the learned decision boundary (shaded area in Figure 2). This means that the adversary does not know whether a crafted poisoning sample would be accepted by the system until he physically carries out the injection (i.e., crafting a poisoning sample, presenting the sample to the system). The procedure of presenting poisoning samples to the system is an expensive action for the attackers: they might raise suspicion and sample crafting might require resources. Additionally, the system could also block the template updating procedure after repeated failed attempts. Therefore, accounting for rejected attempts is fundamental in designing a strategy for the poisoning, as the risk of being detected grows with every attempt.

5 ATTACK METHOD

We adopt the idea of facial accessories from Sharif et al. [29] as a building block of our attack: we imagine the adversary can craft coloured glasses and wear them in order to carry out the attack. The glasses should be re-crafted at each injection step in order to achieve the correct location in features space shown in Figure 2. In our attack method, we have to re-define parts of the method in [29] in order to obtain a better resilience against the two challenges mentioned above: input uncertainty and failed injections.

5.1 Poisoning Sample Generation

Formulation. Formally, given a deep model f, a starting sample from the adversary $\vec{x}$, a target sample from the user $\vec{y}$ and their respective feature vectors computed by the model $f(\vec{x})$ and $f(\vec{y})$, we want to find a perturbation on the input features δ_x that minimizes the following:

$$\arg\min_{\delta_{\vec{x}}} \|f(\vec{x} + \delta_{\vec{x}}) - f(\vec{y})\|_p, \tag{1}$$

where $\|\cdot\|_p$ is a norm function. In order to limit the perturbations into the area of the glasses, we use a binary mask M_x that has the size of the input space and filters out the pixels that do not lie on the glasses frame:

$$\arg\min_{\delta_{\vec{x}}} \|f(\vec{x} \cdot (1 - M_x) + M_x \circ \delta_{\vec{x}}) - f(\vec{y})\|_p, \tag{2}$$

To account for the glasses smoothness, we look for perturbations that minimize total pixel variation of the added perturbation, computed as a function of each pixel value $p_{i,j}$:

$$V(\delta_{\vec{x}}) = \sum_{i,j} ((p_{i,j} - p_{i+1,j})^2 + (p_{i,j} - p_{i,j+1})^2)^{\frac{1}{2}} \text{ [20]}. \tag{3}$$

To account for perturbation printability, we account for the non-printability score (NPS) of the adversarial perturbation. Given a set of printable tuples, P, NPS is given by:

$$NPS(\delta_{\vec{x}}) = \sum_{p \in \delta_{\vec{x}}} \prod_{p' \in P} |p - p'| \text{ [29]}. \tag{4}$$

In order to account for the limited injection capabilities and input noise, rather than optimizing individual samples separately we optimize all samples together in a batch. We then prioritize

Algorithm 1 - Poisoning Samples Generation: neural network f, batch of samples from the adversary $\{\vec{x}\}_i$, target victim sample $f(\vec{y})$, perturbation regularization λ, no. of features to update at each step m, $\{M_{\vec{x}}\}_i$ masks for glasses location.

INPUT: $f, \{\vec{x}\}_i, f(\vec{y}), \{M_{\vec{x}}\}_i, \lambda, m$

1: $\delta_{\vec{x}}^{(1)}$ = `random_init()` ▷ initialize random glasses
2: $j \leftarrow 1$
3: **for** $(j = 1, j < N, 1)$ **do**
4: $\{\delta_{\vec{x}}\}_i$ = `back_propagate`$(f, \{\vec{x}\}_i, \{M_{\vec{x}}\}_i, \delta_{\vec{x}}^{(j)}, f(\vec{x}_c))$
5: $\Delta\delta_{\vec{x}}^{(j)}$ = `select_top_f`$(\{\delta_{\vec{x}}\}_i, m)$ ▷ based on NPS, V, d
6: $\delta_{\vec{x}}^{(j+1)} = \delta_{\vec{x}}^{(j)} + \lambda\Delta\delta_{\vec{x}}^{(j)}$.
7: **end for**
8: **for** $(j = N, j < 2N, 1)$ **do**
9: $\{\delta_{\vec{x}}\}_i$ = `back_propagate`$(f, \{\vec{x}\}_i, \{M_{\vec{x}}\}_i, \delta_{\vec{x}}^{(j)}, f(\vec{y}))$
10: $\Delta\delta_{\vec{x}}^{(j)}$ = `select_top_f`$(\{\delta_{\vec{x}}\}_i, m)$ ▷ based on NPS, V, d
11: $\delta_{\vec{x}}^{(j+1)} = \delta_{\vec{x}}^{(j)} + \lambda\Delta\delta_{\vec{x}}^{(j)}$.
12: **end for**
13: **return** $\delta_{\vec{x}}^{(j)}$ $\forall j \in [N, 2N]$ ▷ return intermediate results

perturbations that minimize the standard deviation of the samples' distances to the target. Since we cannot optimize for this directly with gradient descent, we instead introduce a weight that prioritizes samples that are further away from the distribution of samples' distance to the target. So for a set of samples from the adversary $\{\vec{x}_1, ..., \vec{x}_n\}$ and a perturbation $\delta_{\vec{x}}$ the distance d from the mean for sample i is:

$$d_i = \|f(\vec{x}_i \cdot (1 - M_{x_i}) + M_{x_i} \circ \delta_{\vec{x}}) - f(\vec{y})\|_p - \mu, \tag{5}$$

$$\mu = \frac{1}{n} \sum_{i \in n} \|f(\vec{x}_i \cdot (1 - M_{x_i}) + M_{x_i} \circ \delta_{\vec{x}}) - f(\vec{y})\|_p, \tag{6}$$

so that the optimization becomes:

$$\arg\min_{\delta_{\vec{x}}} \sum_{i \in n} d_i \|f(\vec{x}_i \cdot (1 - M_{x_i}) + M_{x_i} \circ \delta_{\vec{x}}) - f(\vec{y})\|_p + NPS(\delta_{\vec{x}}) + V(\delta_{\vec{x}}). \tag{7}$$

It should be noted that as M_{x_i} changes across different images (i) because of the different location of the glasses (depending on eyes landmarks), we refer to $M_{x_i} \circ \delta_{\vec{x}}$ as the operation that fills the positive elements of the mask with the current coloured glasses frame $\delta_{\vec{x}}$.

Optimization. To find the optimal perturbations, we combine gradient descent with a per-feature iterative approach. We initialize the glasses frame $\delta_{\vec{x}}^{(0)}$ at random, avoiding too bright or too dark colours and we use a Gaussian filter to reduce neighboring pixels variability. At each step, we first back-propagate the gradient from the network. Then we weight the backpropagated gradient of sample x_i by d_i. At this point we choose the top m input features which shift the current samples in the desired direction selecting them as those which minimize the total variability and non-printability of the glasses frame, i.e., $NPS(\delta_{\vec{x}_i}) + V(\delta_{\vec{x}_i})$. This way we obtain the changes to apply to the glasses at this step $\Delta\delta_{\vec{x}}^{(j)}$ which we apply

Algorithm 2 - Poisoning Attack: neural network f, victim user template $Y = \{f(\vec{y})\}_i$, known victim sample $\vec{y} \notin Y$, adversary template $\{\vec{x}\}_i$, classifier C, perturbation delta θ.

INPUT: $f, \vec{y}, \{\vec{x}\}_i, \lambda, N, C, Y$

1: C is trained with Y
2: $\{M_{\vec{x}}\}_i \leftarrow$ `find_eyes_landmarks`$(\{\vec{x}\}_i)$
3: $\{\delta_{\vec{x}}\}^{(j)} \leftarrow$ `gen`$(f, \{\vec{x}\}_i, f(\vec{y}), \{M_{\vec{x}}\}_i, \lambda, m)$ ▷ Algorithm 1
4: IAR $\leftarrow$ `count`$(C$.predict$(\{\vec{x}\}_i) > C$.thresh$)$
5: **while** IAR $< \theta_1$ **do**
6: failures $\leftarrow 0$
7: $j \leftarrow$ HEUR(..) ▷ find perturbations index with heuristic
8: $\{\vec{x}^*\}_i \leftarrow \{\vec{x} \cdot (1 - M_x) + M_x \circ \delta_{\vec{x}}^{(j)}\}_i$ ▷ wear glasses
9: IAR* $\leftarrow$ `count`$(C$.predict$(\{\vec{x}^*\}_i) > C$.thresh$)$
10: **while** IAR* $< \theta_2$ **do**
11: failures $\leftarrow$ failures + 1
12: $j \leftarrow j + 1$ ▷ increase perturbations
13: $\{\vec{x}^*\}_i \leftarrow \{\vec{x} \cdot (1 - M_x) + M_x \circ \delta_{\vec{x}}^{(j)}\}_i$
14: IAR* $\leftarrow$ `count`$(C$.predict$(\{\vec{x}^*\}_i) > C$.thresh$)$
15: **end while**
16: $Y \leftarrow Y + \{f(\vec{x}_k^*)\}$ ▷ k s.t. C.predict$(\vec{x}_k^*) > C$.thresh
17: IAR $\leftarrow$ `count`$(C$.predict$(\{\vec{x}\}_i) > C$.thresh$)$
18: **end while**
19: **return** `True`

them to each sample in the batch:

$$\delta_{\vec{x}}^{(j+1)} = \delta_{\vec{x}}^{(j)} + \lambda\Delta\delta_{\vec{x}}^{(j)}. \tag{8}$$

We use a regularization parameter λ to control the magnitude of pixel changes, to avoid too large steps in the optimization which could lead to sub-optimal adversarial samples. We report the full procedure in Algorithm 1. It should be noted that we perform the optimization twice. After the random initialization, we first optimize the glasses using the adversary's centroid in feature space $\boldsymbol{x_c}$ (Alg. 1 Line 4) as target. In fact, we want to make sure that at the beginning of the generation of poisoning samples the adversary wearing the glasses still maintains their original appearance, i.e., the glasses do not affect the location of the attacker's samples in feature space. Algorithm 1 returns a list of *intermediate* glasses that correspond to the poisoning samples shown in Figure 2. In the next section we use these samples to construct the poisoning attack.

5.2 Poisoning Attack Injection

After using Algorithm 1, all the necessary intermediate poisoning samples are available for the adversary to use. Figure 3 shows a visualization of such samples. However, adversaries need to decide at what point in the optimization process of the glasses the current glasses would lead to an accepted attempt. Here, we first explain how the poisoning works and then we show how the uncertainty on the decision boundaries can be overcome using population data.

Poisoning Algorithm. We report the full algorithm for the poisoning in Algorithm 2. The algorithm uses Alg. 1 to obtain intermediate samples (Alg. 2 Line 3). We use the impostor acceptance rate (IAR) as an indicator for injection (i.e., the proportion of attacker samples that are accepted by the system as legitimate). We consider that the

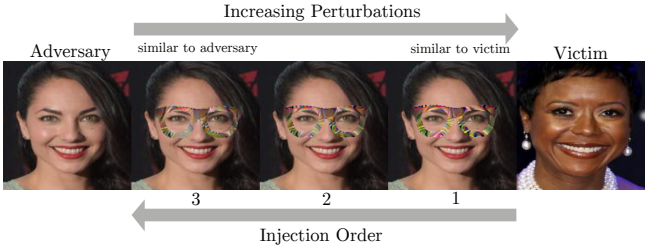


Figure 3: Visualization of intermediate poisoning samples. Adversaries craft glasses shifting their appearance onto the victim's. Increasing the number of perturbations on the glasses allows the adversary to control the distance to the victim (samples shown for ResNet-50). During poisoning the adversary needs to inject samples close to the victim first.

adversary can successfully inject a sample when at least a θ_2 fraction of the attackers samples (as they are wearing the glasses, $\vec{x}^*$) are accepted by the system (Line 10). Whenever the adversary attempts to inject a sample, if less than θ_2 of his samples are accepted by the system we consider the attempt a failure. In this case, the adversary will increase the amount of perturbations on the glasses (move closer to the user's template, see Figure 2,3), and attempt again. In the cases where more than θ_2 samples are accepted, then we consider the attempt successful and inject one of these accepted samples into the current user template (chosen at random, Line 16). The algorithm stops when at least θ_1 fraction of the attacker samples, while wearing no glasses ($\vec{x}$), is accepted by the system (Line 5).

Injection Heuristic. In order to minimize the number of attempted injections we develop a *heuristic* to estimate whether a crafted poisoning sample would be accepted given minimal information about the training data and classifiers. The heuristic is based on the intuition that deep models distribute feature space evenly across different users, attempting to separate each user by a similar distance from the others. Therefore, adversaries can use population data to understand the dynamics of the decision boundaries. The heuristic is then based on two factors known by the adversary: (i) the number of perturbations applied to the input sample (j in Alg. 1) and (ii) the L_2 distance between the poisoning sample and the single known user sample $\|f(\vec{x}^*) - f(\vec{y})\|_2$.

In practice, the adversary runs the attack for a set of users in the population for which he has knowledge of the template (i.e., excluding the actual victim). It should be noted that this does not assume white-box knowledge of the model but simply query access (free injection failures). This poses no significant challenge as user templates can be gathered through social media or just by using publicly available datasets. In this process, he runs Algorithm 2 but replaces Line 7 with an iterative search over the intermediate samples generated by Algorithm 1 until he reaches θ_2 accepted samples. He then collects the number of perturbations applied so far and the L_2 distance to the known samples in the template. With this information, he can find a "sweet spot" in the two-dimensional space (i.e., n. of perturbations, distance to samples in the template) that indicates the likelihood of the current poisoning sample $\vec{x}^*$ of being accepted by the template update function.

Model	*dataset*	*# identities*	*output*	*accuracy*
FaceNet [28]	VGGFace2	8,631	512 (embed)	99.65%
VGG16 [25]	VGGFace	2,622	4,096 (logits)	98.95%
ResNet-50 [7]	VGGFace2	8,631	2,048 (logits)	>99%

Table 2: Info for models and datasets used. The *output* column reports the dimensionality and type of the layer we used to extract facial features. Accuracy refers to the accuracy computed on the labeled faces in the wild dataset [14] (LFW), and the figures are taken from the respective papers.

Using the data gathered from the population, the adversary now has access to a good estimator of sample acceptance. In particular, the closer the current poisoning sample $\vec{x}^*$ is to the center of the two-dimensional distribution, the most likely the sample will be accepted by an unseen template update function. The adversary therefore uses the heuristic to decide which $\vec{x}^*$ to inject and falls back to an iterative approach whenever the heuristic fails (as described in the previous paragraph).

6 EVALUATION

In this section, we first describe the experiment, we then evaluate the success rates of the attack showing its effect on the error rates for a set of classifiers and template weighting schemes. We also present the results for attack transferability across networks.

6.1 Experiment Design

Classifiers. we decide to consider three different classifiers: (i) `centroid` [3], (ii) `maximum` [3] and (iii) one-class Support Vector Machine (`SVM`). The `centroid` classifier computes the distance between the new sample and the centroid of the training data. The `maximum` classifier computes the distance between the new sample and the closest training sample. Both classifiers use L_2 distance to compute distances between samples; we tried using L_1 and we obtained similar results. All three classifiers perform an authentication decision comparing the computed distance to a pre-set threshold. We decide to use a linear kernel for `SVM` as this was the best performing one in terms of recognition performance. To account for the age of samples in the template, i.e., older samples might be considered less relevant than recent samples, we include two different weighting schemes: (i) *flat*, (ii) *sigmoid*. In the flat weighting schemes, each sample in the template is considered equally important in the decision, while in the sigmoid scheme the weight of each sample w_i is computed with a sigmoid function prioritizing recent samples:

$$w_i = \frac{1}{(1 + e^{-x_i})}, \tag{9}$$

where $x_i \in [-5, 5]$ indicates how recent the sample is (the lower the older the sample is). Intuitively, prioritizing recent samples in the decision will make the poisoning attack faster.

Update Policy. For the template update procedure, we choose to use self-update with *infinite window update* policy [4, 16]. In other words, whenever a new sample is accepted by the classifier it is subsequently added to the training set. This choice is similar to the

	centroid		maximum		SVM	
Model	*EER*	*thr.*	*EER*	*thr.*	*EER*	*thr.*
FaceNet	0.9%	0.959	1.3%	0.901	1.0%	0.121
VGG16	2.2%	0.590	3.0%	0.564	1.9%	0.041
ResNet-50	1.3%	0.923	1.9%	0.876	1.7%	0.098

Table 3: Performance of the face-recognition models in terms of EER. The classifiers thresholds are set at the EER computed on the development set.

unsupervised update process used by Apple's FaceID or Siri, where newly obtained samples are added to the template if the match quality is sufficient [2, 32].

Considered DNN. We consider three different state-of-the-art convolutional network architectures: FaceNet [28], the VGG16 Face descriptor [25] and ResNet-50 [7]. FaceNet uses the Inception-ResNet-v1 architecture from [34], and is trained with triplet loss to produce with 512-dimensional embeddings. The VGG-Face descriptor is based on the VGG16 architecture [31] and has been trained on the VGGFace dataset, which contains 2,622 identities. Both ResNet-50 and FaceNet are trained on the newer VGGFace2 [7] dataset which contains 8,631 different identities with on average over 340 images per individual. For VGG16 and ResNet-50 we use the L_2-normalized *logits* layer as output rather than training the models with triplet loss. The details of the models and datasets are reported in Table 2.

Input Preprocessing. All images are first aligned using the landmark based model in [40]. All the remaining preprocessing follows the models' guidelines: FaceNet inputs are pre-whitened, VGG16 and ResNet-50 inputs use fixed image standardization. We don't use any random cropping or flipping for the images. The images dimensions in input are 160x160 for FaceNet and 224x224 for the remaining models. When testing images across different models we resize them to the correct size using bilinear interpolation.

Experiment Description. All the experiments make use of the testing part of the VGGFace2 dataset, which contains 500 separate users. Before we evaluate the attack's effectiveness, we first setup the system as follows. First, we randomly choose 100 users to compute the classifiers thresholds, which we set at equal error rate (EER). For each of the other 400 users, we use 10 samples for training, the remaining user samples for testing and we randomly sample one thousand samples from other users to create the negative set (>100,000 authentication attempts are used to find the EER threshold). The results are reported in Table 3. Afterwards, we split the 400 remaining users into two equally sized groups and we treat the two groups as adversaries and victims separately. We randomly choose 1,000 attacker-victim pairs for the evaluation. These chosen pairs are the same across each model considered.

Attack Implementation. We always use Algorithm 2 for the evaluation. We define the attack as successful when at least half of the adversary template is accepted by the classifier and we use the same to compute the injection success rate, i.e., $\theta_1 = \theta_2 = 0.5$. The heuristic of Section 5.2 is computed using only 10 randomly chosen attacker-victim pairs. The glasses are positioned based on the eyes location (computed using a face landmark extractor) and occupy

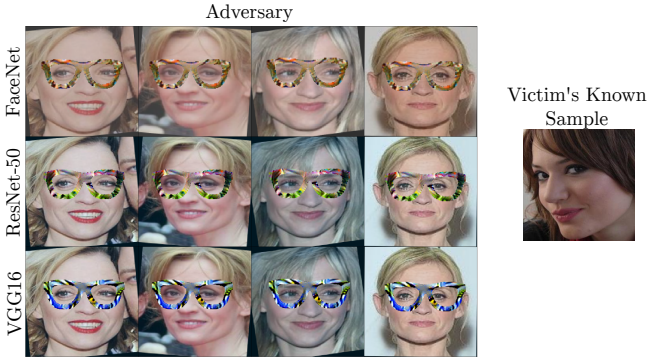


Figure 4: Poisoning samples for an adversary-victim pair. The figure shows how the *same* glasses can be applied to the user across different intra-user variations (e.g., pose).

on average 8.59% of the total pixels in input. Figure 4 shows a set of poisoning samples for an attacker and the relative victim. We only use front facing images from the adversary (up to 50 images), as adversaries have control over their pose as they carry out the attack, and do not need to optimize for their own intra-user variation when attempting the attack. At the moment of injection, the injected sample is chosen at random from the currently accepted samples. We use $\|\cdot\|_2$ for the optimization of Equation 7 as L_2 is used by the classifiers for the decision. The parameter λ is set to 4, which indicates that the maximum change for a pixel in an iteration is 4 out of a range of $[0, 255]$[1].

6.2 Attack Success Rate

In Table 4 we report the attack success rates for all the considered models and classifiers. The table shows the success rates of the attack at different points in the injection, either after one, three or ten injection attempts. Comparing the different classifiers provides different insights. At first, looking at the flat weighting scheme, we see that the `centroid` classifier is much more resilient compared to the other two: a single injected sample leads to success only in less than 4% of cases. Comparably, a single sample is sufficient for a successful attack in ~40% of cases (averaged over the models) for the `maximum` classifier and ~7% for the `SVM`.

Unsurprisingly, comparing the two weighting schemes flat and sigmoid, we see that sigmoid is much more prone to allowing the adversary into the template, as the latest samples have greater importance in the decision. After one injected sample, the `centroid` classifier still shows slighter better resilience compared to `SVM` (~9% vs ~11%), with the `maximum` classifier performing much worse with over ~90% success rate for the same scenario.

We find that the heuristic presented in Section 5.2 can compensate for the limited decision boundaries knowledge of the attacker, the results in this section show the performance of the attacker including failed injection attempts. We found that every time the attacker had to pick an intermediate sample to inject, the heuristic provided an accepted sample in more than 79% of cases (79%, 83% and 82% for FaceNet, VGG16 and ResNet-50, respectively). This is likely a consequence of the optimization of the neural network in

[1] we will publish scripts, models, and datasets on GitHub after publication.

	centroid						maximum						SVM					
	flat			*sigmoid*			*flat*			*sigmoid*			*flat*			*sigmoid*		
Model	1^{st}	3^{rd}	10^{th}	1^{st}	3^{rd}	10^{th}	1^{st}	3^{rd}	10^{th}	1^{st}	3^{rd}	10^{th}	1^{st}	3^{rd}	10^{th}	1^{st}	3^{rd}	10^{th}
FN-ResNet	1%	9%	77%	6%	55%	95%	40%	93%	95%	86%	97%	97%	6%	35%	81%	10%	49%	73%
VGG16	4%	19%	85%	12%	66%	100%	61%	98%	100%	92%	100%	100%	9%	42%	94%	12%	50%	71%
VGG-ResNet	2%	20%	97%	10%	86%	99%	38%	95%	96%	98%	99%	99%	9%	54%	87%	13%	63%	80%

Table 4: Attack success rates for each considered model and classifier. Success is defined as >50% IAR after i samples injections (either one, three or ten). Each figure is calculated on the (same) 1000 randomly chosen pairs of attacker-victim.

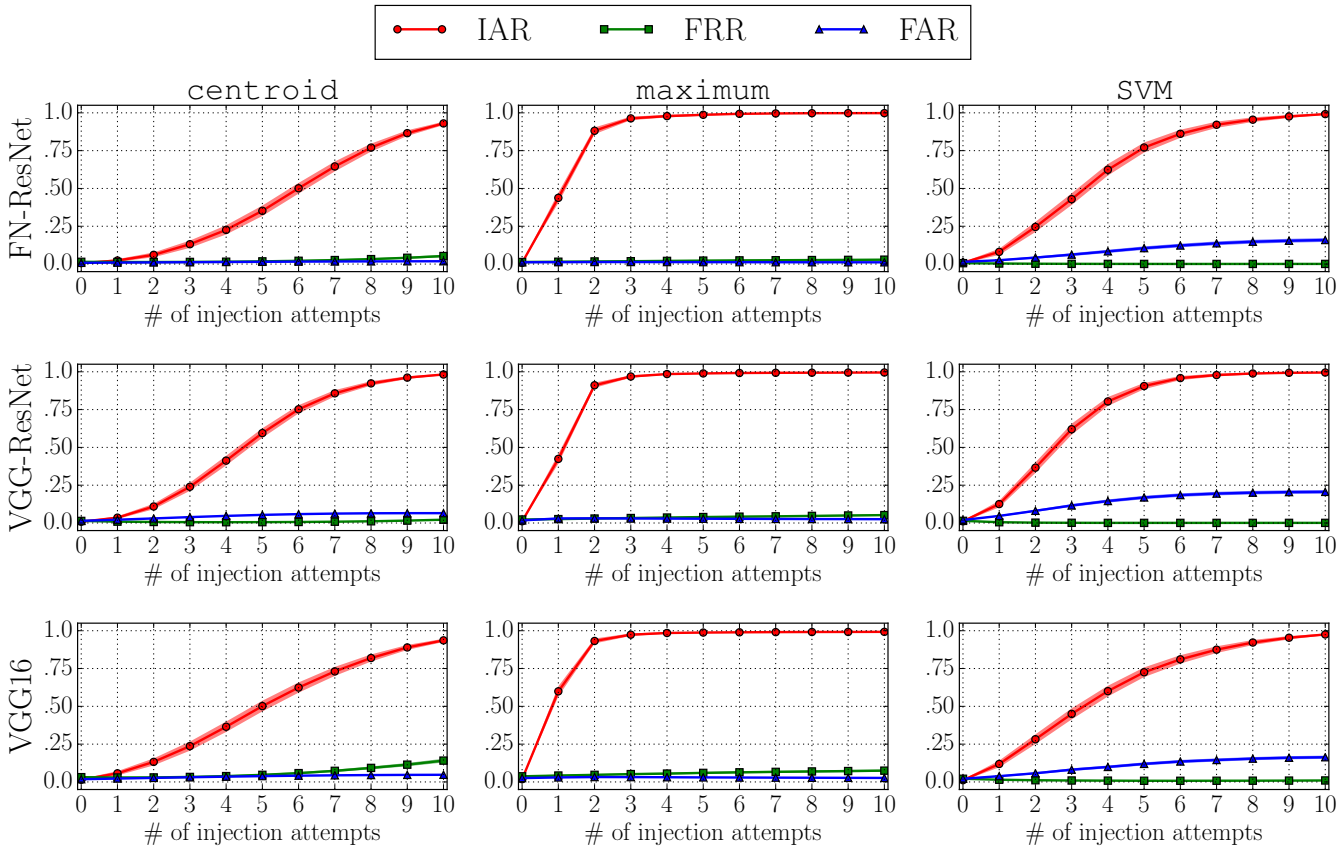


Figure 5: IAR, FAR and FRR changes over the course of the poisoning attack. Shaded areas show the standard deviation over each adversary-victim pair in the dataset. The shown results are for the `centroid` classifier with flat weights.

the development phase. The optimization maximizes the pair-wise distance between users in feature space (directly when using triplet loss, indirectly for multi-class outputs), and makes them evenly distributed as a consequence. In the following section, we focus in more detail on what happens to the error rates of the system in presence of the poisoning.

6.3 Effect on Error Rates

We monitor three different rates: (i) false accept rate (FAR), (ii) false reject rate (FRR), (iii) impostor accept rate (IAR). For an attacker-victim pair, FAR is computed as the proportion of samples belonging to "other users" (users that are not the attacker or the victim) that are accepted by the classifier. Similarly, FRR is the proportion of victim's samples that are rejected by the classifier, excluding the 10 samples in the training set. IAR is defined as the proportion of samples belonging to the adversary that is accepted by the classifier without wearing glasses. Ideally, the attack should make sure that IAR increases as fast as possible while not changing FRR and FAR. Increases in FRR are suspicious because the legitimate user cannot authenticate smoothly anymore (lots of rejections). Consequently, users might contact the system administrator to report the problem or switch to more convenient authentication factors (when other

	FaceNet			FaceNet-CASIA			ResNet-50			VGG16		
Model	`centroid`	`maximum`	`SVM`	`centroid`	`maximum`	`SVM`	`centroid`	`maximum`	`SVM`	`centroid`	`maximum`	`SVM`
FaceNet	**77**%	**95**%	**81**%	22%	27%	24%	12%	12%	14%	9%	11%	9%
FaceNet-CASIA	9%	14%	10%	**85**%	**99**%	**88**%	11%	11%	12%	9%	10%	9%
ResNet-50	13%	20%	14%	23%	30%	27%	**97**%	**96**%	**87**%	13%	16%	13%
VGG16	5%	8%	6%	11%	16%	16%	14%	17%	18%	**85**%	**100**%	**94**%

Table 5: Transferability results of the poisoning attack across different models. The reported figures are the success rates of the attack uniquely using information from the source model (source is on the rows, target is on the columns). Bold values refer to same-model success rates (also found in Table 4). The rates correspond to the success at the 10^{th} injection attempt. The heuristic is always fit on the target system (i.e., both target model and classifier).

options are available). Increases in FAR are less suspicious, depending on how often other users attempt impersonation attacks with their own biometric trait.

Figure 5 shows how the above-mentioned rates vary during the poisoning attack, for the flat weighting scheme. The three models respond similarly to the poisoning, while different classifiers show different changes in the error rates. Comparing the three classifiers with each other shows how the `maximum` classifier is particularly vulnerable to this kind of attack: two injected samples lead to IAR >90% on average, only with marginal increments in the success rate for additional injections.

Overall the changes in FRR and FAR are minimal. A small FAR increase can be seen for the `SVM` classifier. This might be due to the linear kernel function used for the classifier, which cannot fit well the training data when poisoning samples are added (as these reside far away from the legitimate user distribution) and therefore includes wider areas of the feature space. The minimal changes in FRR are in contrast with what has been shown in previous work [3], where FRR (referred to as genuine accept rate) quickly increased to values above 40% (on average) after 5 injected samples and over 90% over 10 injected samples. This is mainly due to a combination of the initial enrolment data size and the type of window used for the self-update procedure. As noted previously [16], fewer samples in the training data lead to faster poisoning (ten in our case, five in [3]) and the choice of discarding older samples [3] with the finite window policy leads to the user being removed from the template. As mentioned in Section 6.1, it should be noted that modern systems tend to reflect the design choices in this work, both for speaker and face recognition [2, 32]). A more detailed and long-term analysis is required to investigate the trade-offs between these choices and the performance of the system in this scenario, which we leave for future work.

6.4 Transferability

We analyze the transferability of the attack across the different models, reporting in Table 5 the results. In addition to the previously considered models, we add FaceNet-CASIA which has the same architecture as FaceNet, but is trained on the CASIA Webface dataset [39] dataset instead of the VGGFace2 dataset. This allows us to investigate the influence of the training data on the transferability. Each attack repetition is carried out using the source model as a surrogate: using the source to compute the poisoning samples (Alg. 2 Line 3), but using the target model to compute the sample acceptances (Alg. 2 Line 4, 9 and 14).

Table 5 reports the success rate, defined as >50% IAR after 10 injected samples, for source-target model pairs. We found that the low success rate is mostly caused by the fact that the optimization of Algorithm 1 did not lead to accepted samples (85% of attacker-victim pairs) rather than not enough samples being accepted after 10 injections (2%). Comparing the white-box results with the across models ones, we see that in most cases there is a ten-fold decrement in the success rates, which could still make the attack viable in black-box scenarios. We don't find particular differences in different classifiers, with the `maximum` being slightly weaker against this attack. These findings are in line Sharif et al. [30] who report similar transferability for dodging attacks across different networks.

Table 5 does not show any particularly evident trend in the transferability across architectures, showing that some transferability applies for each pair of models. Additionally, while it would seem that higher EER would lead to higher chance of attacks (it is easier to find accepted samples by chance), we find that baseline EER and success rate are only weakly correlated (r=-0.18). This suggests that the transferability properties rely on less intuitive combinations of both training datasets and architectures. It should be noted that the success rate of attacks across different models could be improved for example by using optimizations that are not tailored to an individual networks [30] or by obtaining better approximations of the user template.

7 POISONING COUNTERMEASURES

Here, we discuss possible countermeasures and we propose and evaluate a new detection method based on angular similarity.

7.1 Detecting Adversarial Samples

One indirect approach to limit the feasibility of a poisoning attack is to increase the difficulty of crafting adversarial samples with the required properties. As an example, in adversarial training [8, 11, 23], the training data includes adversarial samples specifically crafted to increase the model resilience against them at test time. With gradient masking, the network gradient is hidden during training, making the network robust to small perturbations in the input [12, 19, 24]. In AI^2 [10] the approach aims to approximate the functions learnt by neural networks with abstract primitives, where security guarantees can be certified with sounder methods. Similarly, when an attack vector is identified, such as wearing coloured glasses, measures that are specific to the detection of such attack can be implemented (e.g., detecting whether the pixels in input present anomalous sharpness).

7.2 Template Anomalies

Analyzing anomalies in the template is another approach to stop poisoning attacks, where the goal is to detect whether sets of samples that are added to the template are anomalous. Biggio et al. [5] propose a technique named *template sanitization* which consists in defining a sanitization hypersphere around the current template distribution. Whenever series of k consecutive updates quickly drift the current user centroid outside of the current hypersphere, then such updates are discarded and the previous centroid is restored. The underlying assumption is that genuine template updates exhibit a less biased and more random behavior. A theoretical analysis of the security of online template updates is given by Kloft et al. [16], where a protection mechanism based on monitoring the false positive rate is introduced. Such method suggests that the system could monitor the number of false rejections using an *hold-out* dataset to check whether these exceed a fixed thresholds. In other words, when the users legitimate samples start to be rejected as the poisoning progresses, updates might be discarded and the centroid reset to a previous "safe" state.

Detection Trade-off. Both works [5, 16] show that such countermeasures can successfully thwart the progress of a poisoning attack. The underlying assumptions of the methods is that poisoning samples present a behavior that is not shown by legitimate updates: either quickly drift the centroid outside of a set hypersphere [5] or increase the number of false rejections [16]. However, neither study offers an analysis of the trade-off between poisoning detection and legitimate template update acceptance i.e., legitimate samples might show the behaviour that is labeled as anomalous.

For [16], we have shown that false rejections in our scenario present only marginal increases over the course of the poisoning, and the method additionally assumes the availability of an hold-out dataset for the user, which is not available in a biometric authentication scenario. For [5], we notice that several intra-user variation factors that might affect the location of the sample in feature space in a predictable and consistent way, such as: facial hair, pose, age. Figure 6b shows an example of such behaviour: samples where the user has facial hair (i.e., a moustache, see Figure 6a) cluster together in a specific region of feature space. Such region may be separate from the known user template, meaning that limiting the template updates to a set hypersphere would stop legitimate updates in this scenario [5]. In the following, we present a new poisoning countermeasure and evaluate its detection trade-off with legitimate samples.

7.3 Angular Similarity Detection

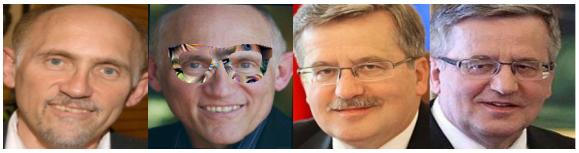
We propose a new detection technique, based on the rationale that poisoning samples will all lie in a predetermined direction in feature space with respect to the current legitimate user centroid. The direction is determined based on the location of the attacker samples. This is visualized in Figure 6b. Given the user's current centroid $\vec{x}_c$ and a set of template updates $\{\vec{x}_i, \vec{x}_{i+1}, ..., \vec{x}_{i+n}\}$, which we refer to as an *update sequence*, we compute the direction of the update at time i:

$$\Delta\vec{x}_i = \vec{x}_c - \vec{x}_i, \tag{10}$$

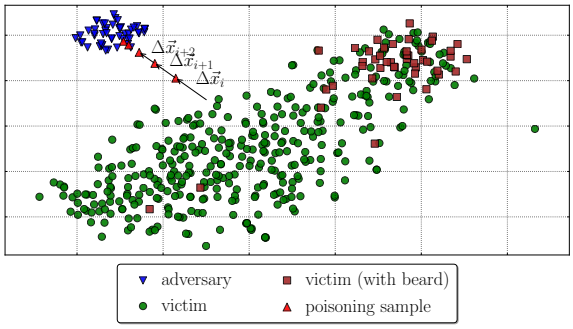
and we can obtain the directions at each step as $\{\Delta\vec{x}_i, ..., \Delta\vec{x}_{i+n}\}$. We then compute the angular similarity for pairs of consecutive updates with the cosine similarity as:

$$\cos\theta_{i+1} = \frac{\Delta\vec{x}_i \cdot \Delta\vec{x}_{i+1}}{\|\Delta\vec{x}_i\|\|\Delta\vec{x}_{i+1}\|}. \tag{11}$$

The underlying intuition is that $\cos\theta_i$ will be higher for pairs of poisoning samples compared to legitimate updates because the poisoning attack needs to shift the current user centroid towards the adversary's, which lies in a specific pre-defined direction, see Figure 6b for reference.



(a) from the left, adversary, adversary during the poisoning attack, victim with facial hair and without facial hair.



(b) embedding space visualized.

Figure 6: Poisoning detection. The figure shows the victim's, the adversary's and the poisoning samples. The two-dimensional coordinates are computed by chaining a PCA reduction with 50 components and a t-SNE reduction. The figure shows that consecutive poisoning samples have consistent angular similarity with one another when compared with the victim centroid. Similarly, other intra-user variation factors (e.g., beards) might reside in a specific direction when compared with the center of the victim distribution.

7.4 Detection Evaluation

Setup. We use the Google Vision API[2] to extract attributes for four intra-user variation factors: pose, facial hair and (sun)glasses. Since the API does not return a value for age, we use the age-estimator in [38]. For pose we only consider the *pan* angle as a factor, that is the horizontal angle of the face (a pan of 90° corresponds to a face looking sideways). For age we group samples into subgroups of samples with the same age, using ranges of three years for each subgroup. This way, for each of the 1,000 attacker-victim pairs considered in the previous section, we consider sequences of updates using the samples in the victim's testset as follows:

- **age**: update sequences come from the same 3-year age span;
- **pose**: we create two update sequences by choosing samples with pan angle $\geq 30°$ or samples with pan angle $\leq 30°$;

[2] https://cloud.google.com/vision/

	FaceNet	ResNet-50	VGG16	FaceNet→ResNet-50		ResNet-50→FaceNet		FaceNet→VGG16		VGG16→ResNet-50	
factor	EER	EER	EER	FAR	FRR	FAR	FRR	FAR	FRR	FAR	FRR
age	6.8%	7.1%	4.5%	7.4%	6.8%	6.6%	7.2%	7.0%	2.2%	4.8%	12.0%
glasses	14.3%	14.6%	11.3%	14.5%	14.6%	14.3%	14.3%	18.7%	5.8%	7.7%	27.0%
facial hair	7.6%	7.1%	4.8%	6.8%	7.5%	8.0%	7.2%	6.9%	2.6%	4.6%	12.6%
pose	6.3%	6.2%	4.0%	6.2%	6.1%	6.3%	6.3%	6.5%	1.9%	4.0%	11.1%
all	**7.7%**	**7.9%**	**5.4%**	**7.7%**	**7.6%**	**7.9%**	**7.9%**	**7.7%**	**2.6%**	**5.4%**	**13.9%**

Table 6: Rates for the poisoning detection. The top row indicates which model is considered for the evaluation: $a \rightarrow b$ indicates that the system is trained on a and applied to b. The values are computed on the `centroid` with flat weights. For same-model cases we report EER. For across-model cases we report the FAR and FRR for the detection on the target model obtained after choosing the threshold at EER on the source model. The "all" row is computed using all the updates from each other factor.

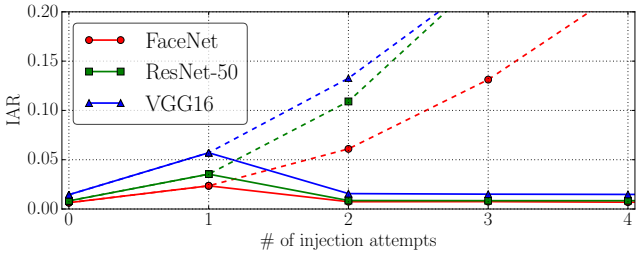


Figure 7: Poisoning detection. The plot shows the IAR comparing when the poisoning detection is in place (solid lines) and when it is not (dashed lines), for the `centroid` classifier with flat weights and the different models (same-model scenarios). The angular similarity threshold is set at EER on all the samples in the considered update sequences.

- **facial hair**: we create an update sequence where the user's samples have facial hair;
- **glasses**: we create an update sequence where the user is wearing glasses or sunglasses;
- **poisoning**: we consider the sequence of poisoning samples obtained from carrying out the attack of Section 6.

We choose a random order for the samples in these sequences, excluding the poisoning one, and we evaluate the angular similarity of pairs of consecutive samples with Equation 11. In total, we consider 1,132,994 legitimate updates, distributed across the attacker-victim pairs. By comparing the legitimate sequences with the poisoning ones we investigate the effect of choosing a threshold for the detection of too similar subsequent updates. Later, we show how IAR changes during the poisoning attack when the detection is in place. We present the results of our analysis to the `centroid` classifier with flat weights; other classifiers performed similarly.

Results. We report in Table 6 the results for the poisoning detection for each of the considered factors. Each row reports the results for a specific factor, that is, all pairs in the update sequences of that factor are evaluated against the pairs in the poisoning sequences; the last *all* row reports the result of using all the measurements together. We see that the detection performance in terms of EER lies around 6-7% for most factors and models (VGG16 model performs better), with the exception of glasses, which create an increase in the EER up to 14%. We find that glasses, in particular sunglasses, create a very predictable and consistent effect on the location of samples in feature space, leading to greater angular similarity of consecutive updates. This can be explained with sunglasses being less frequent in the training data compared to the other factors such as pose and age, that the network learns how to ignore during optimization.

We test the transferability of such detection by checking whether the threshold found on a specific model can generalize to models with different dimensionality of feature space. The last two columns of Table 6 show that this is the case and that the threshold fit on a specific model can be applied directly to different models. We can see that the obtained FAR and FRR on the target model closely resemble the EER obtained in the source model, shown in the first three columns (except for VGG16 where FRR is slightly higher).

We report in Figure 7 the IAR over the course of the poisoning with and without detection in place, setting the threshold at EER for all the considered update sequences. We find that >99% of attacks are detected at the second injection attempt, when the IAR is still low for the attacker to authenticate consistently (<6%). When consecutive poisoning samples are detected, the system removes the malicious samples from the template, resetting the baseline IAR.

8 CONCLUSION

In this paper, we have presented a template poisoning attack that allows the adversary to place a biometric backdoor, which grants inconspicuous long-term access to the biometric system. We designed the attack to cope with minimal knowledge and limited injection capabilities, showing that successful attacks can be obtained even in black-box models. We investigated a set of recognition pipelines, including different models and classifiers. We showed that some classifiers are particularly vulnerable to this attack, where a single poisoning sample injected can lead to success rates of over 40%. We suggested a new countermeasure that can successfully detect sets of consecutive poisoning samples based on their angular similarity in feature space. We evaluated the trade-offs between poisoning detection and legitimate template updates being rejected, obtaining results of around 7% EER for the detection on single sample, leading to >99% attack detection rate after only two injected samples. The weak assumptions of our attacker scenario and the increasing adoption of unsupervised template updates in deployed products highlight the severity of this attack. These results suggest that an increased attention should be given to the update procedure, as this represents an opportunity for attackers to compromise the authentication system.

REFERENCES

[1] Apple. 2017. About Touch ID advanced security technology. https://support.apple.com/en-gb/HT204587. (2017). (accessed 18th April 2018).

[2] Apple. 2017. Face ID Security. https://images.apple.com/business/docs/FaceID_Security_Guide.pdf. (2017). (accessed 18th April 2018).

[3] Battista Biggio, Luca Didaci, Giorgio Fumera, and Fabio Roli. 2013. Poisoning attacks to compromise face templates. *International Conference on Biometrics*, 1–7.

[4] Battista Biggio, Giorgio Fumera, Fabio Roli, and Luca Didaci. 2012. Poisoning adaptive biometric systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7626 (2012), 417–425.

[5] Battista Biggio, Giorgio Fumera, Paolo Russu, Luca Didaci, and Fabio Roli. 2015. Adversarial biometric recognition: A review on biometric system security from the adversarial machine-learning perspective. *IEEE Signal Processing Magazine* 32, 5 (2015), 31–41.

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

[7] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. 2018. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 67–74.

[8] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 39–57.

[9] Giuseppe Garofalo, Vera Rimmer, Davy Preuveneers, and Wouter Joosen. 2018. Fishy Faces: Crafting Adversarial Images to Poison Face Authentication. *Workshop on Offensive Technologies, USENIX*.

[10] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. 2018. *AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation.* Technical Report. 3–18 pages. http://ai2.ethz.ch

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[12] Shixiang Gu and Luca Rigazio. 2014. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068* (2014).

[13] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain. *cs.CR* (2017). arXiv:1708.06733

[14] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. *University of Massachusetts Amherst Technical Report* 1 (2007), 07–49.

[15] Danny Karmon, Daniel Zoran, and Yoav Goldberg. 2018. Lavan: Localized and visible adversarial noise. *arXiv preprint arXiv:1801.02608* (2018).

[16] Marius Kloft and Pavel Laskov. 2010. Security Analysis of Online Centroid Anomaly Detection. *Journal of Machine Learning Research* 13 (2010), 3681–3724. arXiv:1003.0078

[17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. (2016), 1–14. http://arxiv.org/abs/1607.02533

[18] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Authors Yingqi Liu, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *NDSS 2018 (Network and Distributed System Security Symposium)*.

[19] Chunchuan Lyu, Kaizhu Huang, and Hai Ning Liang. 2016. A unified gradient regularization family for adversarial examples. In *Proceedings - IEEE International Conference on Data Mining, ICDM*.

[20] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 5188–5196.

[21] Karthik Nandakumar, Anil K. Jain, and Abhishek Nagar. 2008. Biometric template security. *Eurasip Journal on Advances in Signal Processing* 2008 (2008).

[22] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277* (2016).

[23] Nicolas Papernot, Patrick Mcdaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. *IEEE European Symposium on Security and Privacy*, 372–387.

[24] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. 2018. SoK: Security and Privacy in Machine Learning. In *Proceedings of the 3rd IEEE European Symposium on Security and Privacy*. IEEE.

[25] O. M. Parkhi, A. Vedaldi, and A. Zisserman. 2015. Deep Face Recognition. In *British Machine Vision Conference*.

[26] Vishal M. Patel, Nalini K. Ratha, and Rama Chellappa. 2015. Cancelable biometrics: A review. *IEEE Signal Processing Magazine* 32, 5 (2015), 54–65. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7192838

[27] Amir Rahmati, Atul Prakash, Chaowei Xiao, Dawn Song, Bo Li, Tadayoshi Kohno, Ivan Evtimov, Earlence Fernandes, and Kevin Eykholt. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. (2018), 1625–1634. http://arxiv.org/abs/1707.08945

[28] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 07-12-June-2015 (2015), 815–823.

[29] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. 2016. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. *ACM SIGSAC Conference on Computer and Communications Security* (2016), 1528–1540.

[30] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2017. Adversarial Generative Nets: Neural Network Attacks on State-of-the-Art Face Recognition. 1 (2017). arXiv:1801.00349 http://arxiv.org/abs/1801.00349

[31] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[32] Siri Team. 2018. Personalized Hey Siri. (2018). [accessed 1st May 2019].

[33] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. 2018. Physical adversarial examples for object detectors. In *12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18)*.

[34] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[35] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[36] Matthew Turk and Alex Pentland. 1991. Eigenfaces for recognition. *Journal of cognitive neuroscience* 3, 1 (1991), 71–86.

[37] Yi Xu, True Price, Jan-Michael Frahm, and Fabian Monrose. 2016. Virtual U: Defeating Face Liveness Detection by Building Virtual Models From Your Public Photos. *Usenix Security* (2016).

[38] Tsun-Yi Yang, Yi-Hsuan Huang, Yen-Yu Lin, Pi-Cheng Hsiu, and Yung-Yu Chuang. 2018. SSR-Net: A Compact Soft Stagewise Regression Network for Age Estimation.. In *IJCAI*. 1078–1084.

[39] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. 2014. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923* (2014).

[40] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters* 23, 10 (Oct 2016), 1499–1503. https://doi.org/10.1109/LSP.2016.2603342