Fawkes: Protecting Privacy against Unauthorized Deep Learning Models

Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y. Zhao, *University of Chicago*

https://www.usenix.org/conference/usenixsecurity20/presentation/shan

This paper is included in the Proceedings of the 29th USENIX Security Symposium.

August 12–14, 2020

978-1-939133-17-5

Open access to the Proceedings of the 29th USENIX Security Symposium is sponsored by USENIX.

Fawkes: Protecting Privacy against Unauthorized Deep Learning Models

Shawn Shan[†]*, Emily Wenger*[†]*, Jiayun Zhang, Huiying Li, Haitao Zheng, Ben Y. Zhao*
[†] *denotes co-first authors with equal contribution*
Computer Science, University of Chicago
{shansixiong, ewillson, jiayunz, huiyingli, htzheng, ravenben}@cs.uchicago.edu

Abstract

Today's proliferation of powerful facial recognition systems poses a real threat to personal privacy. As *Clearview.ai* demonstrated, anyone can canvas the Internet for data and train highly accurate facial recognition models of individuals without their knowledge. We need tools to protect ourselves from potential misuses of unauthorized facial recognition systems. Unfortunately, no practical or effective solutions exist.

In this paper, we propose *Fawkes*, a system that helps individuals inoculate their images against unauthorized facial recognition models. Fawkes achieves this by helping users add imperceptible pixel-level changes (we call them "cloaks") to their own photos before releasing them. When used to train facial recognition models, these "cloaked" images produce functional models that consistently cause normal images of the user to be misidentified. We experimentally demonstrate that Fawkes provides 95+% protection against user recognition regardless of how trackers train their models. Even when clean, uncloaked images are "leaked" to the tracker and used for training, Fawkes can still maintain an 80+% protection success rate. We achieve 100% success in experiments against today's state-of-the-art facial recognition services. Finally, we show that Fawkes is robust against a variety of countermeasures that try to detect or disrupt image cloaks.

1 Introduction

Today's proliferation of powerful facial recognition models poses a real threat to personal privacy. Facial recognition systems are scanning millions of citizens in both the UK and China without explicit consent [33, 41]. By next year, 100% of international travelers will be required to submit to facial recognition systems in top-20 US airports [38]. Perhaps more importantly, anyone with moderate resources can now canvas the Internet and build highly accurate facial recognition models of us without our knowledge or awareness, *e.g.* MegaFace [21]. Kashmir Hill from the New York Times recently reported on *Clearview.ai*, a private company that collected more than 3 billion online photos and trained a massive model capable of recognizing millions of citizens, all without knowledge or consent [20].

Opportunities for misuse of this technology are numerous and potentially disastrous. Anywhere we go, we can be identified at any time through street cameras, video doorbells, security cameras, and personal cellphones. Stalkers can find out our identity and social media profiles with a single snapshot [47]. Stores can associate our precise in-store shopping behavior with online ads and browsing profiles [31]. Identity thieves can easily identify (and perhaps gain access to) our personal accounts [13].

We believe that private citizens need tools to protect themselves from being identified by unauthorized facial recognition models. Unfortunately, previous work in this space is sparse and limited in both practicality and efficacy. Some have proposed distorting images to make them unrecognizable and thus avoiding facial recognition [27, 52, 64]. Others produce adversarial patches in the form of bright patterns printed on sweatshirts or signs, which prevent facial recognition algorithms from even registering their wearer as a person [55, 65]. Finally, given access to an image classification model, "clean-label poison attacks" can cause the model to misidentify a single, preselected image [42, 71].

Instead, we propose *Fawkes*, a system that helps individuals to inoculate their images against unauthorized facial recognition models at any time without significantly distorting their own photos, or wearing conspicuous patches. Fawkes achieves this by helping users adding imperceptible pixel-level changes ("cloaks") to their own photos. For example, a user who wants to share content (*e.g.* photos) on social media or the public web can add small, imperceptible alterations to their photos before uploading them. If collected by a third-party "tracker" and used to train a facial recognition model to recognize the user, these "cloaked" images would produce functional models that consistently misidentify them.

Our distortion or "cloaking" algorithm takes the user's photos and computes minimal perturbations that shift them significantly in the feature space of a facial recognition model (using real or synthetic images of a third party as a landmark). Any facial recognition model trained using these images of the user learns an altered set of "features" of what makes them look like them. When presented with a clean, uncloaked image of the user, *e.g.* photos from a camera phone or streetlight camera, the model finds no labels associated

with the user in the feature space near the image, and classifies the photo to another label (identity) nearby in the feature space.

Our exploration of Fawkes produces several key findings:

- We can produce significant alterations to images' feature space representations using perturbations imperceptible to the naked eye (DSSIM ≤ 0.007).
- Regardless of how the tracker trains its model (via transfer learning or from scratch), image cloaking provides 95+% protection against user recognition (adversarial training techniques help ensure cloaks transfer to tracker models).
- Experiments show 100% success against state-of-the-art facial recognition services from Microsoft (Azure Face API), Amazon (Rekognition), and Face++. We first "share" our own (cloaked) photos as training data to each service, then apply the resulting models to uncloaked test images of the same person.
- In challenging scenarios where clean, uncloaked images are "leaked" to the tracker and used for training, we show how a single *Sybil* identity can boost privacy protection. This results in 80+% success in avoiding identification even when half of the training images are uncloaked.
- Finally, we consider a tracker who is aware of our image cloaking techniques and evaluate the efficacy of potential countermeasures. We show that image cloaks are robust (maintain high protection rates against) to a variety of mechanisms for both cloak disruption and cloak detection.

2 Background and Related Work

To protect user privacy, our image cloaking techniques leverage and extend work broadly defined as poisoning attacks in machine learning. Here, we set the context by discussing prior efforts to help users evade facial recognition models. We then discuss relevant data poisoning attacks, followed by related work on privacy-preserving machine learning and techniques to train facial recognition models.

Note that to protect user privacy from unauthorized deep learning models, we employ attacks against ML models. In this scenario, *users* are the "attackers," and third-party *trackers* running unauthorized tracking are the "targets."

2.1 Protecting Privacy via Evasion Attacks

Privacy advocates have considered the problem of protecting individuals from facial recognition systems, generally by making images difficult for a facial recognition model to recognize. Some rely on creating *adversarial examples*, inputs to the model designed to cause misclassification [54]. These attacks have since been proven possible "in the wild," Sharif *et al.* [44] create specially printed glasses that cause the wearer to be misidentified. Komkov and Petiushko [24] showed that carefully computed adversarial stickers on a hat can reduce its wearer's likelihood of being recognized. Others propose "adversarial patches" that target "person identification" models, making it difficult for models to recognize the wearer as a person in an image [55, 65].

All of these approaches share two limitations. First, they require the user to wear fairly obvious and conspicuous accessories (hats, glasses, sweaters) that are impractical for normal use. Second, in order to evade tracking, they require *full and unrestricted* access (white box access) to the precise model tracking them. Thus they are easily broken (and user privacy compromised) by any tracker that updates its model.

Another line of work seeks to *edit facial images* so that human-like characteristics are preserved but facial recognition model accuracy is significantly reduced. Methods used include k-means facial averaging [35], facial inpainting [51], and GAN-based face editing [27, 52, 64]. Since these dramatically alter the user's face in her photos, we consider them impractical for protecting shared content.

2.2 Protecting Privacy via Poisoning Attacks

An alternative to evading models is to disrupt their training. This approach leverages "data poisoning attacks" against deep learning models. These attacks affect deep learning models by modifying the initial data used to train them, usually by adding a set of samples S and associated labels L_S. Previous work has used data poisoning to induce unexpected behaviors in trained DNNs [66]. In this section, we discuss two data poisoning attacks related to our work, and identify their key limitations when used to protect user privacy.

Clean Label Attacks. A clean-label poisoning attack injects "correctly" labeled poison images into training data, causing a model trained on this data to misclassify a specific image of interest [42, 71]. What distinguishes clean-label attacks from normal poisoning attacks is that all image labels remain unchanged during the poisoning process – only the content of the poisoned images changes.

Our work (Fawkes) works with similar constraints. Our action to affect or disrupt a model is limited to altering a group of images with a correct label, *i.e.* a user can alter her images but cannot claim these are images of someone else.

Current clean label attacks cannot address the privacy problem because of three factors. *First*, they only cause misclassification on a *single, preselected* image, whereas user privacy protection requires the misclassification of any current or future image of the protected user (*i.e.* an entire model class). *Second*, clean label attacks do not transfer well to different models, especially models trained from scratch. Even between models trained on the same data, the attack only transfers with 30% success rate [71]. *Third*, clean label attacks are easily detectable through anomaly detection in the feature space [19].

Model Corruption Attacks. Other recent work proposes

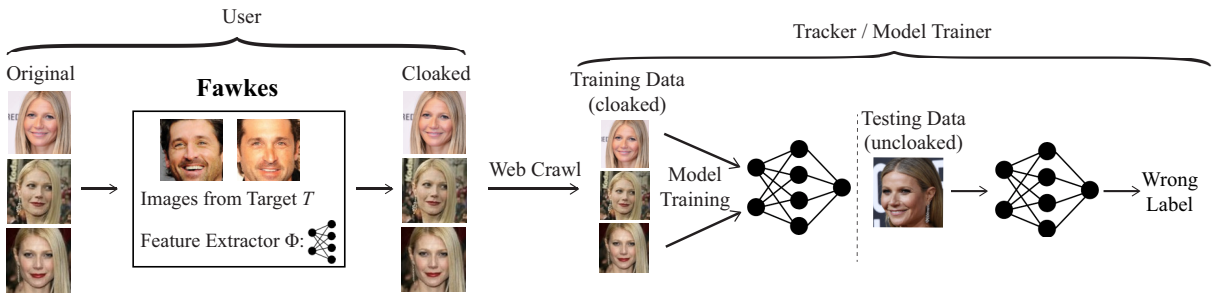


Figure 1: Our proposed Fawkes system that protects user privacy by cloaking their online photos. (Left) A user U applies cloaking algorithm (given a feature extractor Φ and images from some target T) to generate cloaked versions of U's photos, each with a small perturbation unnoticeable to the human eye. (Right) A tracker crawls the cloaked images from online sources, and uses them to train an (unauthorized) model to recognize and track U. When it comes to classifying new (uncloaked) images of U, the tracker's model misclassifies them to someone not U. Note that T does not have to exist in the tracker's model.

techniques to modify images such that they *degrade* the accuracy of a model trained on them [45]. The goal is to spread these poisoned images in order to discourage unauthorized data collection and model training. We note that Fawkes' goals are to mislead rather than frustrate. Simply corrupting data of a user's class may inadvertently inform the tracker of the user's evasion attempts and lead to more advanced countermeasures by the tracker. Finally, [45] only has a 50% success rate in protecting a user from being recognized.

2.3 Other Related Work

Privacy-Preserving Machine Learning. Recent work has shown that ML models can memorize (and subsequently leak) parts of their training data [48]. This can be exploited to expose private details about members of the training dataset [17]. These attacks have spurred a push towards *differentially private* model training [6], which uses techniques from the field of differential privacy [15] to protect sensitive characteristics of training data. We note these techniques imply a trusted model trainer and are ineffective against an unauthorized model trainer.

Feature Extractors & Transfer Learning. Transfer learning uses existing pretrained models as a basis for quickly training models for customized classification tasks, using less training data. Today, it is commonly used to deploy complex ML models (*e.g.* facial recognition or image segmentation [70]) at reasonable training costs.

In transfer learning, the knowledge of a pre-trained feature extractor Φ is passed on to a new model $\mathbb{F}_\theta$. Typically, a model $\mathbb{F}_\theta$ can be created by appending a few additional layers to Φ and only training those new layers. The original layers that composed Φ will remain unmodified. As such, pre-existing knowledge "learned" by Φ is passed on to the model $\mathbb{F}_\theta$ and directly influences its classification outcomes. Finally, transfer learning is most effective when the feature extractor and model are trained on similar datasets. For example, a facial recognition model trained on faces extracted from YouTube videos might serve well as a feature extractor for a model designed to recognize celebrities in magazines.

Finally, the concept of protecting individual privacy against invasive technologies extends beyond the image domain. Recent work [12] proposes wearable devices that restore personal agency using digital jammers to prevent audio eavesdropping by ubiquitous digital home assistants.

3 Protecting Privacy via Cloaking

We propose *Fawkes*, a system designed to help protect the privacy of a *user* against unauthorized facial recognition models trained by a third-party *tracker* on the user's images. Fawkes achieves this by adding subtle perturbations ("cloaks") to the user's images before sharing them. Facial recognition models trained on cloaked images will have a distorted view of the user in the "feature space," *i.e.* the model's internal understanding of what makes the user unique. Thus the models cannot recognize real (uncloaked) images of the user, and instead, misclassify them as someone else.

In this section, we first describe the threat model and assumptions for both users and trackers. We then present the intuition behind cloaking and our methodology to generate cloaks. Finally, we discuss why cloaking by individuals is effective against unauthorized facial recognition models.

3.1 Assumptions and Threat Model

User. The user's goal is to share their photos online without unknowingly helping third party trackers build facial recognition models that can recognize them. Users protect themselves by adding imperceptible perturbations ("cloaks") to their photos before sharing them. This is illustrated in the left part of Figure 1, where a cloak is added to this user's photos before they are uploaded.

The design goals for these cloaks are:

- cloaks should be **imperceptible** and not impact normal use of the image;
- when classifying normal, uncloaked images, models trained on cloaked images should recognize the underlying person with **low accuracy**.

We assume the user has access to moderate computing resources (*e.g.,* a personal laptop) and applies cloaking to their own images locally. We also assume the user has access to some feature extractor, *e.g.* a generic facial recognition model, represented as Φ in Figure 1. Cloaking is simplified if the user has the same Φ as the tracker. We begin with this common assumption (also used by prior work [42, 59, 71]), since only a few large-scale face recognition models are available in the wild. Later in §3.4, we relax this assumption and show how our design maintains the above properties.

We initially consider the case where the user has the ability to apply cloaking to all their photos to be shared, thus the tracker can only collect cloaked photos of the user. Later in §7, we explore a scenario where a stronger tracker has obtained access to some number of their uncloaked images.

Tracker/Model Trainer. We assume that the tracker (the entity training unauthorized models) is a third party without direct access to user's personal photos (*i.e.* not Facebook or Flickr). The tracker could be a company like Clearview.ai, a government entity, or even an individual. The tracker has significant computational resources. They can either use transfer learning to simplify their model training process (leveraging existing feature extractors), or train their model completely from scratch.

We also assume the tracker's primary goal is to build a powerful model to track many users rather than targeting a single specific person[1]. The tracker's primary data source is a collection of public images of users obtained via web scraping. We also consider scenarios where they are able to obtain some number of uncloaked images from other sources (§7).

Real World Limitations. Privacy benefits of Fawkes rely on users applying our cloaking technique to the majority of images of their likeness before posting online. In practice, however, users are unlikely to control all images of themselves, such as photos shared online by friends and family, media, employer or government websites. While it is unclear how easy or challenging it will be for trackers to associate these images with the identity of the user, a tracker who obtains a large number of uncloaked images of the user can compromise the effectiveness of Fawkes.

Therefore, Fawkes is most effective when used in conjunction with other privacy-enhancing steps that minimize the online availability of a user's uncloaked images. For example, users can curate their social media presence and remove tags of their names applied to group photos on Facebook or Instagram. Users can also leverage privacy laws such as "Right to be Forgotten" to remove and untag online content related to themselves. The online curation of personal images is a challenging problem, and we leave the study of minimizing online image footprints to future work.

[1] Tracking a specific person can be easily accomplished through easier, offline methods, *e.g.* a private investigator who follows the target user, and is beyond the scope of our work.

3.2 Overview and Intuition

DNN models are trained to identify and extract (often hidden) *features* in input data and use them to perform classification. Yet their ability to identify features is easily disrupted by data poisoning attacks during model training, where small perturbations on training data with a particular label (l) can shift the model's view of what features uniquely identify l [42, 71]. Our work leverages this property to cause misclassification of *any existing or future image* of a single class, providing one solution to the challenging problem of protecting personal privacy against the unchecked spread of facial recognition models.

Intuitively, our goal is to protect a user's privacy by modifying their photos in small and imperceptible ways before posting them online, such that a facial recognition model trained on them learns the wrong features about what makes the user look like the user. The model thinks it is successful, because it correctly recognizes its sample of (modified) images of the user. However, when unaltered images of the user, *e.g.* from a surveillance video, are fed into the model, the model does not detect the features it associates with the user. Instead, it identifies someone else as the person in the video. By simply modifying their online photos, the user successfully prevents unauthorized trackers and their DNN models from recognizing their true face.

3.3 Computing Cloak Perturbations

But how do we determine what perturbations (we call them "cloaks") to apply to Alice's photos? An effective cloak would teach a face recognition model to associate Alice with erroneous features that are quite different from real features defining Alice. Intuitively, the more dissimilar or distinct these erroneous features are from the real Alice, the less likely the model will be able to recognize the real Alice.

In the following, we describe our methodology for computing cloaks for each specific user, with the goal of making the features learned from cloaked photos highly dissimilar from those learned from original (uncloaked) photos.

Notation. Our discussion will use the following notations.

- x: Alice's image (uncloaked)
- x_T: target image (image from another class/user T) used to generate cloak for Alice
- $\delta(x, x_T)$: cloak computed for Alice's image x based on an image x_T from label T
- $x \oplus \delta(x, x_T)$: cloaked version of Alice's image x
- Φ: Feature extractor used by facial recognition model

- $\Phi(x)$: Feature vector (or feature representation) extracted from an input x

Cloaking to Maximize Feature Deviation. Given each photo (x) of Alice to be shared online, our ideal cloaking design modifies x by adding a cloak perturbation $\delta(x, x_T)$ to x that maximize changes in x's feature representation:

$$\max_{\delta} Dist\left(\Phi(x), \Phi(x \oplus \delta(x, x_T))\right), \quad \text{subject to } |\delta(x, x_T)| < \rho, \tag{1}$$

where $Dist(.)$ computes the distance of two feature vectors, $|\delta|$ measures the perceptual perturbation caused by cloaking, and ρ is the perceptual perturbation budget.

To guide the search for the cloak perturbation in eq (1), we use another image x_T from a different user class (T). Since the feature space Φ is highly complex, x_T serves as a landmark, enabling fast and efficient search for the input perturbation that leads to large changes in feature representation. Ideally, T should be very dissimilar from Alice in the feature space. We illustrate this in Figure 1, where we use Patrick Dempsey (a male actress) as a dissimilar target T for the original user (female actor Gwyneth Paltrow).

We note that our design does not assume that the cloak target (T) and the associated x_T are used by any tracker's face recognition model. In fact, any user whose feature representation is sufficiently different from Alice's would suffice (see §3.4). Alice can easily check for such dissimilarity by running the feature extractor Φ on other users' online photos. Later in §4 we will present the detailed algorithm for choosing the target user T from public datasets of facial images.

Image-specific Cloaking. When creating cloaks for her photos, Alice will produce image-specific cloaks, *i.e.* $\delta(x, x_T)$ is image dependent. Specifically, Alice will pair each original image x with a target image x_T of class T. In our current implementation, the search for $\delta(x, x_T)$ replaces the ideal optimization defined by eq. (1) with the following optimization:

$$\min_{\delta} Dist\left(\Phi(x_T), \Phi(x \oplus \delta(x, x_T))\right), \quad \text{subject to } |\delta(x, x_T)| < \rho. \tag{2}$$

Here we search for the cloak for x that shifts its feature representation closely towards x_T. This new form of optimization also prevents the system from generating extreme $\Phi(x \oplus \delta(x, x_T))$ values that can be easily detected by trackers using anomaly detection.

Finally, our image-specific cloak optimization will create different cloak patterns among Alice's images. This "diversity" makes it hard for trackers to detect and remove cloaks.

3.4 Cloaking Effectiveness & Transferability

Now a user (Alice) can produce cloaked images whose feature representation is dissimilar from her own but similar to that of a target user T. But does this translate into the desired misclassification behavior in the tracker model? Clearly, if T is a class in the tracker model, Alice's original (uncloaked) images will not be classified as Alice. But under the more likely scenario where T is not in the tracker model, does cloaking still lead to misclassification?

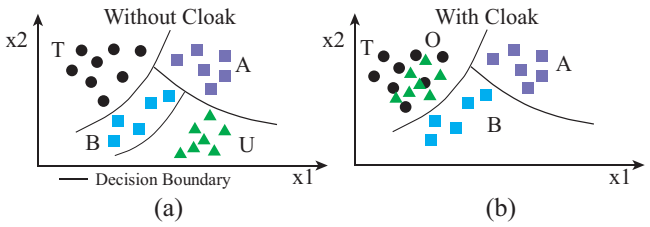


Figure 2: The intuition for why a tracker's model trained on U's cloaked photos will misclassify U's original photos, visualized on a simplified 2D feature space with four user classes A, B, U (aka Alice), T. (a) decision boundaries of the model trained on U's uncloaked photos. (b) decision boundaries when trained on U's cloaked photos (with target T).

We believe the answer is **yes**. Our hypothesis is that as long as the feature representations of Alice's cloaked and uncloaked images are sufficiently different, the tracker's model will not classify them as the same class. This is because there will be another user class (*e.g.* B) in the tracker model, whose feature representation is more similar to $\Phi(x)$ (true Alice) than $\Phi(x \oplus \delta)$ (Alice learned by the model). Thus, the model will classify Alice's normal images as B.

We illustrate this in Figure 2 using a simplified 2D visualization of the feature space. There are 4 classes (A, B, U aka Alice, and T) that a tracker wishes to distinguish. The two figures show the tracker model's decision boundary when U's training data is uncloaked and cloaked, respectively. In Figure 2(a), the model will learn U's true feature representation as the bottom right corner. In Figure 2(b), U uses T as the cloak target, and the resulting tracker model will learn U's feature representation $\Phi(x \oplus \delta)$ as green triangles near T (top left corner). This means that the area corresponding to U's original feature representation $\Phi(x)$ will be classified as B. More importantly, this (mis)classification will occur whether or not T is a class in the tracker's model.

Our above discussion assumes the tracker's model contains a class whose feature representation is more similar to the user's original feature representation than her cloaked feature representation. This is a reasonable assumption when the tracker's model targets many users (*e.g.* 1,000) rather than a few users (*e.g.* 2). Later in §5 we confirm that cloaking is highly effective against multiple facial recognition models with anywhere from 65 to 10,575 classes.

Transferability. Our above discussion also assumes that the user has the same feature extractor Φ as is used to train the tracker model. Under the more general scenario, the effectiveness of cloaking against any tracker models relies on

the *transferability* effect, the property that models trained for similar tasks share similar properties and vulnerabilities, even when they were trained on different architectures and different training data [14, 39, 50, 70].

This transferability property suggests that cloaking should still be effective even if the tracker performs transfer learning using a different feature extractor or trains their model from scratch. Because the user's and tracker's feature extractors/models are designed for similar tasks (*i.e.* facial recognition), cloaks should be effective regardless of the tracker's training method. Later, we empirically evaluate cloaking success rate when trackers use different feature extractors (§5.3) or train models from scratch (§5.4). In all scenarios, cloaking is highly effective ($>95\%$ protection rate).

4 The Fawkes Image Cloaking System

We now present the detailed design of *Fawkes*, a practical image cloaking system that allows users to evade identification by unauthorized facial recognition models. Fawkes uses three steps to help a user modify and publish her online photos.

Given a user U, Fawkes takes as input the set of U's photos to be shared online $\mathbf{X_U}$, the (generic) feature extractor Φ, and the cloak perturbation budget ρ.

Step 1: Choosing a Target Class T. First, Fawkes examines a publicly available dataset that contains numerous groups of images, each identified with a specific class label, *e.g.* Bob, Carl, Diana. Fawkes randomly picks K candidate target classes and their images from this public dataset and uses the feature extractor Φ to calculate C_k, the centroid of the feature space for each class $k = 1..K$. Fawkes picks as the target class T the class in the K candidate set whose feature representation centroid is most dissimilar from the feature representations of all images in $\mathbf{X_U}$, *i.e.*

$$T = \underset{k=1..K}{\operatorname{argmax}} \min_{x \in \mathbf{X_U}} Dist(\Phi(x), C_k). \quad (3)$$

We use L2 as the distance function in feature space, $Dist(.)$.

Step 2: Computing Per-image Cloaks. Let $\mathbf{X_T}$ represent the set of target images available to user U. For each image of user U, $x \in \mathbf{X_U}$, Fawkes randomly picks an image $x_T \in \mathbf{X_T}$, and computes a cloak $\delta(x, x_T)$ for x, following the optimization defined by eq. (2), subject to $|\delta(x, x_T)| < \rho$.

In our implementation, $|\delta(x, x_T)|$ is calculated using the DSSIM (Structural Dis-Similarity Index) [61, 62]. Different from the L_p distance used in previous work [9, 25, 43], DSSIM has gained popularity as a measure of user-perceived image distortion [23, 28, 59]. Bounding cloak generation with this metric ensures that cloaked versions of images are visually similar to the originals.

We apply the *penalty method* [37] to reformat and solve the optimization in eq.(2) as follows:

$$\min_{\delta} Dist\left(\Phi(x_T), \Phi(x \oplus \delta(x, x_T))\right) + \lambda \cdot max(|\delta(x, x_T)| - \rho, 0)$$

Here λ controls the impact of the input perturbation caused by cloaking. When $\lambda \to \infty$, the cloaked image is visually identical to the original image. Finally, to ensure the input pixel intensity remains in the correct range ($[0, 255]$), we transform the intensity values into *tanh* space as proposed in previous work [10].

Step 3: Limiting Content. Now the user U has created the set of cloaked images that she can post and share online. However, the user must be careful to ensure that no uncloaked images are shared online and associated with her identity. Any images shared by friends and labeled or tagged with her name would provide uncloaked training data for a tracker model. Fortunately, a user can proactively "untag" herself on most photo sharing sites.

Even so, a third party might be able to restore those labels and re-identify her in those photos using friendlist intersection attacks [63]. Thus, in §7, we expand the design of Fawkes to address trackers who are able to obtain uncloaked images in addition to cloaked images of the user.

5 System Evaluation

In this section, we evaluate the effectiveness of Fawkes. We first describe the datasets, models, and experimental configurations used in our tests. We then present results for cloaking in three different scenarios: 1) the user produces cloaks using the same feature extractor as the tracker; 2) the user and tracker use different feature extractors; and 3) the tracker trains models from scratch (no feature extractor).

Our key findings are: cloaking is highly effective when users share a feature extractor with the tracker; efficacy could drop when feature extractors are different, but can be restored to near perfection by making the user's feature extractor robust (via adversarial training); and, similarly, cloaks generated on robust feature extractors work well even when trackers train models from scratch.

5.1 Experiment Setup

Our experiments require two components. First, we need feature extractors that form the basis of facial recognition models for both the user's cloaking purposes and the tracker's model training. Second, we need datasets that emulate a set of user images scraped by the tracker and enable us to evaluate the impact of cloaking.

Feature Extractors. There are few publically available, large-scale facial recognition models. Thus we train feature extractors using two large ($\geq$ 500K images) datasets on different model architectures (details in Table 2).

- `VGGFace2` contains 3.14M images of 8,631 subjects downloaded from Google Image Search [7].
- `WebFace` has 500,000 images of faces covering roughly 10,000 subjects collected from the Internet [69].

Teacher Dataset	Model Architecture	Abbreviation	Teacher Testing Accuracy	Student Testing Accuracy	
				PubFig	FaceScrub
WebFace	InceptionResNet	Web-Incept	74%	96%	92%
WebFace	DenseNet	Web-Dense	76%	96%	94%
VGGFace2	InceptionResNet	VGG2-Incept	81%	95%	90%
VGGFace2	DenseNet	VGG2-Dense	82%	96%	92%

Table 1: The four feature extractors used in our evaluation, their classification efficacy and those of their student models.

Dataset	# of Labels	Input Size	# of Training Images
PubFig	65	$224 \times 224 \times 3$	5,850
FaceScrub	344	$224 \times 224 \times 3$	37,905
WebFace	10,575	$224 \times 224 \times 3$	475,137
VGGFace2	8,631	$224 \times 224 \times 3$	3,141,890

Table 2: Datasets emulating user images in experiments.

Using these two datasets, we build four feature extractors, two from each. We use two different model architectures: a) DenseNet-121 [22], a 121 layer neural network with 7M parameters, and b) InceptionResNet V2 [53], a 572 layer deep neural network with over 54M parameters. Our trained models have comparable accuracy with previous work [7, 34, 59] and perform well in transfer learning scenarios. For clarity, we abbreviate feature extractors based on their dataset/architecture pair. Table 1 lists the classification accuracy for our feature extractors and student models.

Tracker's Training Datasets. Under the scenario where the tracker trains its facial recognition model from scratch (§5.4), we assume they will use the above two large datasets (VGGFace2, WebFace). Under the scenario where they apply transfer learning (§5.2 and §5.3), the tracker uses the following two smaller datasets (more details in Table 2).

- PubFig contains 5,850 training images and 650 testing images of 65 public figures[2] [5].
- FaceScrub contains 100,000 images of 530 public figures on the Internet [36][3].

To perform transfer learning, the tracker adds a softmax layer at the end of the feature extractor (see §2.3), and fine-tunes the added layer using the above dataset.

Cloaking Configuration. In our experiments, we randomly choose a user class U in the tracker's model, *e.g.* a random user in PubFig, to be the user seeking protection. We then apply the target selection algorithm described in §4 to select a target class T from a small subset of users in VGGFace2 and WebFace. Here we ensure that T is not a user class in the tracker's model.

For each given U and T pair, we pair each image x of U with an image x_T from T, and compute the cloak for x. For this we run the Adam optimizer for 1000 iterations with a learning rate of 0.5.

[2] We exclude 18 celebrities also used in the feature extractor datasets.

[3] We could only download 60,882 images for 530 people, as some URLs were removed. Similarly, prior work [68] only retrieved 48,579 images.

As discussed earlier, we evaluate our cloaking under three scenarios, U and tracker model sharing the same feature extractor (§5.2), the two using different feature extractors (§5.3), and the tracker training model from scratch without using any pre-defined feature extractor (§5.4).

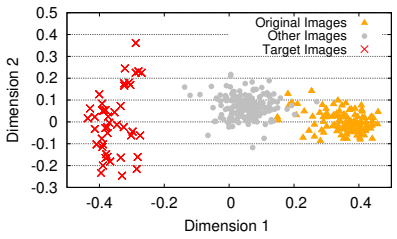
Evaluation Metrics. In each scenario, we evaluate cloak performance using two metrics: *protection success rate*, which is the tracker model's misclassification rate for clean (uncloaked) images of U, and *normal accuracy*, which is the overall classification accuracy of the tracker's model on users beside U. When needed, we indicate the configuration of user/tracker feature extractors using the notation <entity>:<feature extractor>.

5.2 User/Tracker Sharing a Feature Extractor

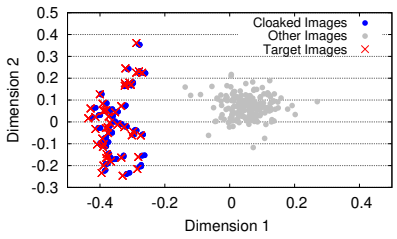
We start from the simple case where the user uses the same feature extractor as the tracker to generate cloaks. We randomly select a label from PubFig or FaceScrub to be the Fawkes user U. We then compute "cloaks" for a subset of U's images, using each of the four feature extractors in Table 1. On the tracker side, we perform transfer learning on the *same* feature extractor (with cloaked images of U) to build a model that recognizes U. Finally, we evaluate whether the tracker model can correctly identify other *clean* images of U it has not seen before.

Results show that cloaking offers perfect protection, *i.e.* U is always misclassified as someone else, for all four feature extractors and under the perturbation budget $\rho = 0.007$. To explore the impact of ρ, Figure 4 plots protection success rate vs. ρ when the tracker runs on the FaceScrub dataset. Fawkes achieves 100% protection success rate when $\rho > 0.005$. Figure 5 shows original and cloaked images, demonstrating that cloaking does not visually distort the original image. Even when $\rho = 0.007$, the perturbation is barely detectable by the naked eye on a full size, color image. For calibration, note that prior work [28] claims much higher DSSIM values (up to 0.2) are imperceptible to the human eye. Finally, the average $L2$ norm of our cloaks is 5.44, which is smaller than that of perturbations used in prior works [29, 59].

Feature Space Deviation. The goal of a cloak is to change the image's feature space representation in the tracker's model. To examine the effect of the cloak in the tracker model, we visualize feature space representations of user images before and after cloaking, their chosen target images,



(a) Before Cloaking



(b) After Cloaking

Figure 3: 2-D PCA visualization of `VGG2-Dense` feature space representations of user images (sampled from `FaceScrub`) before/after cloaking. Triangles are user's images, red crosses are target images, grey dots are images from another class.

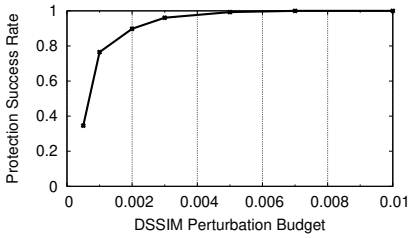


Figure 4: Protection performance as DSSIM perturbation budget increases. (User/Tracker: `Web-Incept`)

and a randomly chosen class from the tracker's dataset. We use principal components analysis (PCA, a common dimensionality reduction technique) to reduce the high dimensional feature space to 2 dimensions. Figure 3 shows the PCA results for cloaked images from a `PubFig` class, using cloaks constructed on the `Web-Incept` feature extractor. Figure 3(a) shows the feature space positions of the original and target images before cloaking, along with a randomly selected class. Figure 3(b) shows the updated feature space after the original images have been cloaked. It is clear that feature space representations of the cloaked images are well-aligned with those of the target images, validating our intuition for cloaking (an abstract view in Figure 2).

Impact of Label Density. As discussed in §3, the number of labels present in the tracker's model impacts performance. When the tracker targets fewer labels, the feature space is "sparser," and there is a greater chance the model continues to associate the original feature space (along with the cloaked feature space) with the user's label. We empirically evaluate the impact of fewer labels on cloaking success using the `PubFig` and `FaceScrub` datasets (65 and 530 labels, respectively). We randomly sample N labels (varying N from 2 to 10) to construct a model with fewer labels. Figure 6 shows that for `PubFig`, cloaking success rate grows from 68% for 2 labels to $>$ 99% for more than 6 labels, confirming that a higher label density improves cloaking effectiveness.

5.3 User/Tracker Using Different Feature Extractors

We now consider the scenario when the user and tracker use different feature extractors to perform their tasks. While the model transferability property suggests that there are significant similarities in their respective model feature spaces (since both are trained to recognize faces), their differences could still reduce the efficacy of cloaking. Cloaks that shift image features significantly in one feature extractor may produce a much smaller shift in a different feature extractor.

To illustrate this, we empirically inspect the change in feature representation between two different feature extractors.

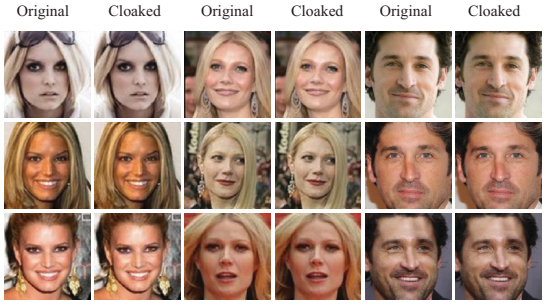


Figure 5: Pairs of original and cloaked images ($\rho = 0.007$).

We take the cloaked images (optimized using `VGG2-Dense`), original images, and target images from the `PubFig` dataset and calculate their feature representations in a *different* feature extractor, `Web-Incept`. The result is visualized using two dimensional PCA and shown in Figure 7. From the PCA visualization, the reduction in cloak effectiveness is obvious. In the tracker's feature extractor, the cloak "moves" the original image features only slightly towards the target image features (compared to Figure 3(b)).

Robust Feature Extractors Boost Transferability. To address the problem of cloak transferability, we draw on recent work linking model robustness and transferability. Demontis *et al.* [14] argue that an input perturbation's (in our case, cloak's) ability to transfer between models depends on the "robustness" of the feature extractor used to create it. They show that more "robust" models are less reactive to small perturbations on inputs. Furthermore, they claim that perturbations (or, again, cloaks) generated on more robust models will take on "universal" characteristics that are able to effectively fool other models.

Following this intuition, we propose to improve cloak transferability by increasing the user feature extractor's robustness. This is done by applying *adversarial training* [18, 30], which trains the model on perturbed data to make it less sensitive to similar small perturbations on inputs. Specifically, for each feature extractor, we generate adversarial examples using the PGD attack [25], a widely used method

User's Robust Feature Extractor	Model Trainer's Feature Extractor							
	VGG2-Incept		VGG2-Dense		Web-Incept		Web-Dense	
	PubFig	FaceScrub	PubFig	FaceScrub	PubFig	FaceScrub	PubFig	FaceScrub
VGG2-Incept	100%	100%	100%	100%	95%	100%	100%	100%
VGG2-Dense	100%	100%	100%	100%	100%	100%	100%	100%
Web-Incept	100%	100%	100%	100%	100%	100%	99%	99%
Web-Dense	100%	100%	100%	100%	100%	97%	100%	96%

Table 3: Protection performance of cloaks generated on robust feature extractors.

for adversarial training. Following prior work [30], we run the PGD[4] algorithm for 100 steps using a step size of 0.01. We train each feature extractor for an additional 10 epochs. These updated feature extractors are then used to generate user cloaks on the PubFig and FaceScrub datasets.

Results in Table 3 show that each robust feature extractor produces cloaks that transfer almost perfectly to the tracker's models. Cloaks now have protection success rates $> 95\%$ when the tracker uses a different feature extractor. We visualize their feature representation using PCA in Figure 8 and see that, indeed, cloaks generated on robust extractors transfer better than cloaks computed on normal ones.

5.4 Tracker Models Trained from Scratch

Finally, we consider the scenario in which a powerful tracker trains their model from scratch. We select the user U to be a label inside the WebFace dataset. We generate cloaks on user images using the robust VGG2-Incept feature extractor from §5.3. The tracker then uses the WebFace dataset (but U's cloaked images) to train their model from scratch. Again our cloas achieve a success rate of 100%. Other combinations of labels and user-side feature generators all have 100% protection success.

6 Image Cloaking in the Wild

Our results thus far have focused on limited configurations, including publicly available datasets and known model architectures. Now, we wish to understand the performance of Fawkes on deployed facial recognition systems in the wild.

We evaluate the real-world effectiveness of image cloaking by applying Fawkes to photos of one of the co-authors. We then intentionally leak a portion of these cloaked photos to public cloud-based services that perform facial recognition, including Microsoft Azure Face [3], Amazon Rekognition [2], and Face++ [4]. These are the global leaders in facial recognition and their services are used by businesses, police, private entities, and governments in the US and Asia.

[4]We found that robust models trained on CW attack samples [10] produce similar results

Face Recognition API	Protection Success Rate		
	Without protection	Protected by normal cloak	Protected by robust cloak
Microsoft Azure Face API	0%	100%	100%
Amazon Rekognition Face Verification	0%	34%	100%
Face++ Face Search API	0%	0%	100%

Table 4: Cloaking is highly effective against cloud-based face recognition APIs (Microsoft, Amazon and Face++).

6.1 Experimental Setup

We manually collected 82 high-quality pictures of a co-author that feature a wide range of lighting conditions, poses, and facial expressions. We separate the images into two subsets, one set of 50 images for "training" and one set of 32 images for "testing." We generate both normal and robust cloaks for the "training" images using the setup discussed in Section 5 (using normal and robust versions of the Web-Incept feature extractor). This allows us to compare the relative effectiveness of normal and robust user feature extractors in real life.

For each API service, we experiment with three scenarios:

- **Unprotected:** We upload original training images, and test the model's classification accuracy on testing images.
- **Normal Cloak:** We upload training images protected by a *nonrobust* cloak and then test the model's classification accuracy on the testing images.
- **Robust Cloak:** We upload training images protected by a *robust* cloak and test the model's classification accuracy on the testing images.

For each scenario, we use the online service APIs to upload training images to the API database, and then query the APIs using the uncloaked testing images. The reported protection success rate is the proportion of uncloaked test images that the API fails to correctly identify as our co-author.

6.2 Real World Protection Performance

Microsoft Azure Face API. Microsoft Azure Face API [3] is part of Microsoft Cognitive Services, and is reportedly used by many large corporations including Uber and Jet.com. The API provides face recognition services. A client uploads

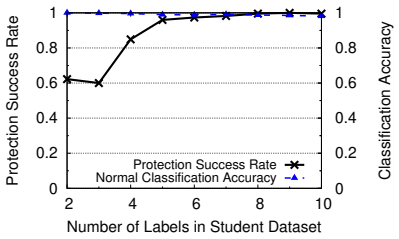


Figure 6: Protection performance improves as the number of labels in tracker's model increases. (User/Tracker: `Web-Incept`)

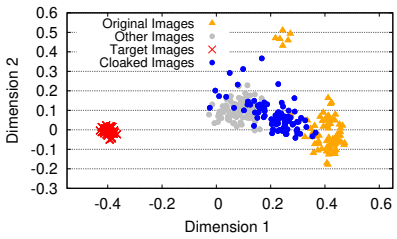


Figure 7: Cloaking is less effective when users and trackers use different feature extractors. (User: `VGG2-Dense`, Tracker: `Web-Incept`)

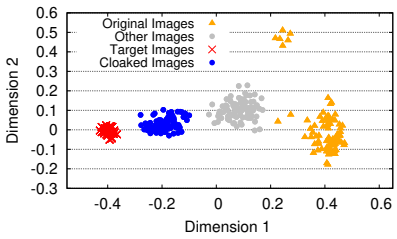


Figure 8: Cloaks generated on robust models transfer better between feature extractors. (User: `VGG2-Dense`, Tracker: `Web-Incept`)

training images of faces, and Microsoft trains a model to recognize these faces. The API has a "training" endpoint that must be called before the model will recognize faces, which leads us to believe that Microsoft uses transfer learning to train a model on user-submitted images.

Our normal cloaking method is 100% effective against the Microsoft Azure Face API. Our robust cloaks also provide 100% protection against the Azure Face API. Detailed protection results are shown in Table 4.

Amazon Rekognition Face Verification. Amazon Rekognition [2] provides facial search services that the client can use to detect, analyze, and compare faces. The API is used by various large corporations including the NFL, CBS, and National Geographic, as well as law enforcement agencies in Florida and Oregon, and the U.S. Immigration and Customs Enforcement agency (ICE).

It is important to note that Amazon Rekognition does not specifically train a neural network to classify queried images. Instead, it computes an image similarity score between the queried image and the ground truth images for all labels. If the similarity score exceeds a threshold for some label, Amazon returns a match. Our cloaking technique is not designed to fool a tracker who uses similarity matching. However, we believe our cloaking technique should still be effective against Amazon Rekognition, since cloaks create a feature space separation between original and cloaked images that should result in low similarity scores between them.

Table 4 shows that our normal cloaks only achieve a protection success rate of 34%. However, our robust cloaks again achieve a 100% protection success rate.

Face++. Face++ [4] is a well-known face recognition system developed in China that claims to be extremely robust against a variety of attacks (*i.e.* adversarial masks, makeup, etc.). Due to its high performance and perceived robustness, Face++ is widely used by financial services providers and other security-sensitive customers. Notably, Alipay uses Face++'s services to authenticate users before processing payments. Lenovo also uses Face++ services to perform face-based authentication for laptop users.

Our results show that normal cloaking is completely ineffective against Face++ (0% protection success rate; see Table 4). This indicates that their model is indeed extremely robust against input perturbations. However, as before, our robust cloaks achieve a 100% success rate.

Summary. Microsoft Azure Face API, Amazon Rekognition and Face++ represent three of the most popular and widely deployed facial recognition services today. The success of Fawkes cloaking techniques suggests our approach is realistic and practical against production systems. While we expect these systems to continue improving, we expect cloaking techniques to similarly evolve over time to keep pace.

7 Trackers with Uncloaked Image Access

Thus far we have assumed that the tracker only has access to *cloaked images* of a user, *i.e.* the user is perfect in applying her cloaking protection to her image content, and disassociating her identity from images posted online by friends. In real life, however, this may be too strong an assumption. Users make mistakes, and unauthorized labeled images of the user can be taken and published online by third parties such as newspapers and websites.

In this section, we consider the possibility of the tracker obtaining leaked, uncloaked images of a target user, *e.g.* Alice. We first evaluate the impact of adding these images to the tracker's model training data. We then consider possible mechanisms to mitigate this impact by leveraging the use of limited sybil identities online.

7.1 Impact of Uncloaked Images

Intuitively, a tracker with access to some labeled, uncloaked images of a user has a much greater chance of training a model M that successfully recognizes clean images of that user. Training a model with both cloaked and uncloaked user images means the model will observe a much larger spread of features all designated as the user. Depending on how M is trained and the presence/density of other labels, it can a)

classify both regions of features as the user; b) classify both regions and the region between them as the user; or c) ignore these feature dimensions and identify the user using some alternative features (*e.g.* other facial features) that connect both uncloaked and cloaked versions of the user's images.

We assume the tracker cannot visually distinguish between cloaked and uncloaked images and trains their model on both. We quantify the impact of training with uncloaked images using a simple test with cloaks generated from §5.2 and a model trained on both cloaked and uncloaked images. Figure 10 shows the drop in protection success for `FaceScrub` dataset as the ratio of uncloaked images in the training dataset increases. The protection success rate drops below 39% when more than 15% of the user's images are uncloaked.

Next, we consider proactive mitigation strategies against leaked images. The most direct solution is to intentionally release more cloaked images, effectively flooding a potential tracker's training set with cloaked images to dominate any leaked uncloaked images. In addition, we consider the use of a cooperating secondary identity (more details below). For simplicity, we assume that: trackers have access to a *small* number of a user's uncloaked images; the user is unaware of the contents of the uncloaked images obtained by the tracker; and users know the feature extractor used by the tracker.

7.2 Sybil Accounts

In addition to proactive flooding of cloaked images, we explore the use of cooperative *Sybil accounts* to induce model misclassification. A Sybil account is a separate account controlled by the user that exists in the same Internet community (*i.e.* Facebook, Flickr) as the original account. Sybils already exist in numerous online communities [67], and are often used by real users to curate and compartmentalize content for different audiences [26]. While there are numerous detection techniques for Sybil detection, individual Sybil accounts are difficult to identify or remove [60].

In our case, we propose that privacy-conscious users create a secondary identity, preferably not connected to their main identity in the metadata or access patterns. Its content can be extracted from public sources, from a friend, or even generated artificially via generative adversarial networks (GANs) [32]. Fawkes modifies Sybil images (in a manner similar to cloaking) to provide additional protection for the user's original images. Since Sybil and user images reside in the same communities, we expect trackers will collect both. While there are powerful re-identification techniques that could be used to associate the Sybil back to the original user, we assume they are impractical for the tracker to apply at scale to its population of tracked users.

Sybil Intuition. To bolster cloaking effectiveness, the user modifies Sybil images so they occupy the same feature space as a user's uncloaked images. These Sybil images help confuse a model trained on both Sybil images *and* uncloaked/cloaked images of a user, increasing the protection success rate. Figure 9 shows the high level intuition. Without Sybil images, models trained on a small portion of uncloaked (leaked) images would easily associate test images of the user with the user's true label (shown on left). Because the leaked uncloaked images and Sybil images are close by in their feature space representations, but labeled differently (*i.e.* " User 1" and "User 2"), the tracker model must create additional decision boundaries in the feature space (right figure). These additional decision boundaries decrease the likelihood of associating the user with her original feature space.

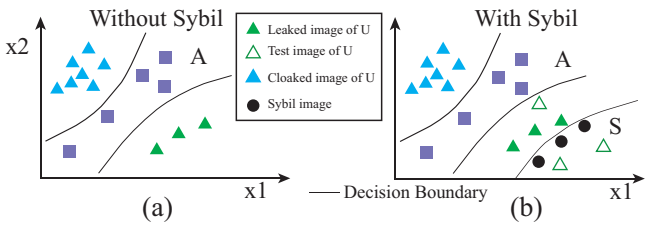


Figure 9: Intuition behind Sybil integration visualized in a 2D feature space. Without Sybils, a tracker's model will use leaked training images of U to learn U's true feature space (left), leading to the correct classification of images of U. Sybil images S complicate the model's decision boundary and cause misclassification of U's images, even when leaked images of U are present (right).

For simplicity, we explore the base case where the user is able to obtain one single Sybil identity to perform feature space obfuscation on her behalf. Our technique becomes even more effective with multiple Sybils, but provides much of its benefit with images labeled with a single Sybil identity.

Creating Sybil images. Sybil images are created by adding a specially designed cloak to a set of candidate images. Let x_C be an image from the set of candidates the user obtains (*i.e.* images generated by a GAN) to populate the Sybil account. To create the final Sybil image, we create a cloak $\delta(x_C, x)$ that minimizes the feature space separation between x_C and user's original image x, for each candidate. The optimization is equivalent to setting x as the target and optimizing to create $x_C \oplus \delta(x_C, x)$ as discussed in §4. After choosing the final x_c from all the candidates, a ready-to-upload Sybil image $x_S = x_C \oplus \delta(x_C, x)$.

7.3 Efficacy of Sybil Images

Sybil accounts can increase a user's protection success rate when the tracker controls a small number of a user's uncloaked images. To experimentally validate this claim, we choose a label from the tracker's dataset to be the Sybil account (controlled by the user), and split the user's images into two disjoint sets: A contains images that were processed by Fawkes, and whose cloaked versions have been shared on-

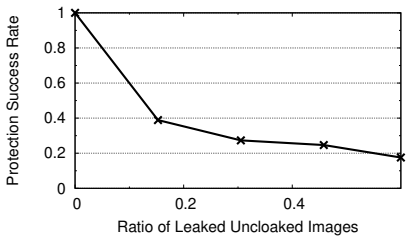


Figure 10: Protection success rate decreases when the tracker has more original user images. (User/Tracker: `Web-Incept`)

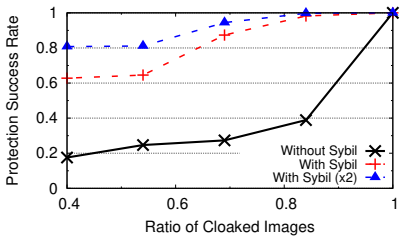


Figure 11: Protection success rate is high when the user has a Sybil account, even if tracker has original user images. (User/Tracker: `Web-Incept`)

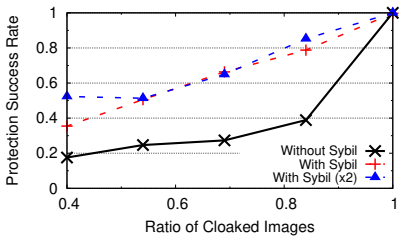


Figure 12: Sybils jointly optimized on four feature extractors have reasonably high protection success for each individual extractor.

line; and *B* contains original images leaked to the tracker. For each synthetic image of the Sybil, we randomly select an uncloaked image of the user in set *A*. We select one Sybil image per uncloaked image in *A*. Then, we cloak all the candidate images using the methodology discussed in §4. The resulting Sybil images mimic the feature space representation of uncloaked user images. From the tracker's perspective, they have access to cloaked user images from set *A*, uncloaked images from set *B*, and the Sybil images.

Figure 11 compares the protection success rate with and without Sybil accounts (with `Web-Incept` as user's and tracker's feature extractor). The use of a Sybil account significantly improves the protection success rate when an attacker has a small number of original images. The protection success rate remains above 87% when the ratio of the original images owned by the tracker is less than 31%.

As discussed, a user can create as many Sybil images as they desire. When the user uploads more Sybil images, the protection success rate increases. Figure 11 shows that when the user has uploaded 2 Sybil images per uncloaked image, the protection success rate increases by 5.5%.

Jointly Optimize Multiple Feature Extractors. The user may not know the tracker's exact feature extractor. However, given the small number of face feature extractors available online, she is likely to know that the tracker would use one of several candidate feature extractors. Thus, she could jointly optimize the Sybil cloaks to simultaneously fool all the candidate feature extractors.

We test this in a simple experiment by jointly optimizing Sybil cloaks on the four feature extractors from §5. We evaluate the cloak's performance when the tracker uses one of the four. Figure 12 shows the Sybil effectiveness averaged across the 4 feature extractors. The average protection success rate remains above 65% when the ratio of the original images owned by the tracker is less than 31%.

8 Countermeasures

In this section, we explore potential countermeasures a tracker could employ to reduce the effectiveness of image cloaking. We consider and (where possible) empirically validate methods to *remove* cloaks from images, as well as techniques to detect the presence of cloak perturbations on images. Our experiments make the strongest possible assumption about the tracker: that they know the precise feature extractor a user used to optimize cloaks. We test our countermeasures on a tracker's model trained on the `FaceScrub` dataset. Cloaks were generated using the same robust `VGG2-Dense` feature extractor from §5.3.

Inherent Limits on Cloaking Success. We acknowledge that cloaking becomes less effective when an individual is an *active target* of a tracker. If a tracker strongly desires to train a model that recognizes a certain individual, they can take drastic measures that cloaking cannot withstand. For example, a tracker could learn their movements or invade their privacy (*i.e.* learn where they live) by following them physically.

8.1 Cloak Disruption

Without knowing which images in the dataset are cloaked, the tracker may utilize the following techniques to disrupt Fawkes' protection performance, 1) transforming images or 2) deploying an extremely robust model. We present and evaluate Fawkes's performance against these two potential countermeasures.

Image Transformation. A simple technique to mitigate the impact of small image perturbations is to transform images in the training dataset before using them for model training [8, 16]. These transformations include image augmentation, blurring, or adding noise. Additionally, images posted online are frequently compressed before sharing (i.e. in the upload process), which could impact cloak efficacy.

However, we find that none of these transformations defeat our cloaks. The protection success rate remains 100% even

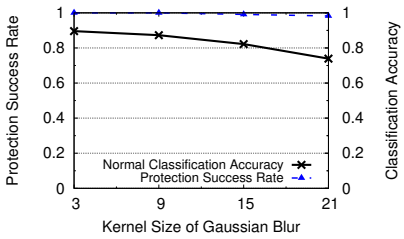


Figure 13: Normal classification accuracy decreases as input blurring increases but protection success rate remains high.

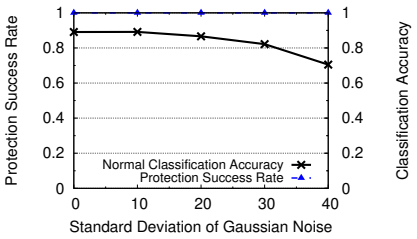


Figure 14: Normal classification accuracy decreases as Gaussian noise is added to inputs but protection success rate remains high.

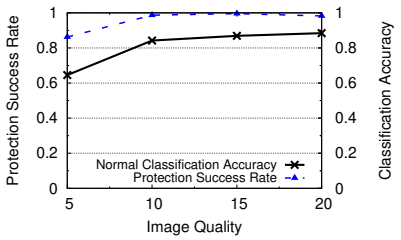


Figure 15: Protection success rate and normal classification accuracy increase as image quality increases using JPEG compression.

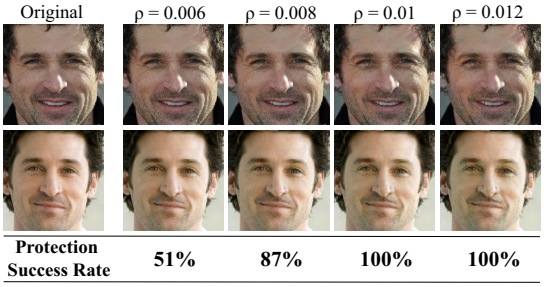


Figure 16: When the user's feature extractor is much less robust than the tracker's feature extractor, the user can improve their protection success rate by increasing their DSSIM budget. (User: `VGG2-Dense`, Tracker: `Web-Incept`)

when data augmentation is applied to cloaked images [5]. Applying Gaussian blurring degrades normal accuracy by up to 18% (as kernel size increases) while cloak protection success rate remains $> 98\%$ (see Figure 13). Adding Gaussian noise to images merely disrupts normal classification accuracy – the cloak protection success rate remains above 100% as the standard deviation of the noise distribution increases (see Figure 14). Even image compression cannot defeat our cloak. We use progressive JPEG [57], reportedly used by Facebook and Twitter, to compress the images in our dataset. The image quality, as standard by Independent JPEG Group [1], ranges from 5 to 95 (lower value = higher compression). As shown in Figure 15, image compression decreases the protection success rate, but more significantly degrades normal classification accuracy.

Robust Model. As shown in §5, cloaks constructed on robust feature extractors transfer well to trackers' less robust feature extractors. Thus, a natural countermeasure a tracker could employ is training their model to be extremely robust.

Despite the theoretically proven trade-off between normal accuracy and robustness [56], future work may find a way to improve model robustness while minimizing the accompanying drop in accuracy. Thus, we evaluate cloaking success when the tracker's model is much more robust than the user's feature extractor. In our simplified test, the user has a robust `VGG2-Dense` feature extractor (adversarially trained for 3 epochs), while the tracker has an extremely robust `Web-Incept` feature extractor (adversarially trained for 20 epochs). When the tracker's model is this robust, the user's cloak only achieves a 64% protection success rate.

However, if the user is extremely privacy sensitive, she could increase the visibility of her cloak perturbation to achieve a higher protection success rate. Figure 16 highlights the trade off between protection success and the input DSSIM level. The cloak's protection success rate increases to 100% once the DSSIM perturbation is > 0.01.

8.2 Cloak Detection

We now propose techniques a tracker could employ to detect cloaked images in their dataset. We also discuss mitigations the user could apply to avoid detection.

Existing Poison Attack Detection. Since cloaking is a form of data poisoning, prior work on detecting poisoning attacks [11, 19, 40, 46, 49, 58] could be helpful. However, all prior works assume that poisoning only affects a small percentage of training images, making outlier detection useful. Fawkes poisons an entire model class, rendering outlier detection useless by removing the correct baseline.

Anomaly Detection w/o Original Images. We first consider anomaly detection techniques in the scenario where the tracker does not have any original user images. If trackers obtain both target and cloaked user images, they can detect unusual closeness between cloaked images and target images in model feature space. Empirically, the $L2$ feature space distance between the cloaked class centroid and the target class centroid is 3 standard deviations smaller than the mean separation of other classes. Thus, user's cloaked images can be detected.

However, a user can trivially overcome this detection by maintaining separation between cloaked and target images during cloak optimization. To show this, we use the same ex-

[5] Image augmentation parameters: rotation range=20^o, horizontal shift=15%, vertical shift=15%, zoom range=15%

perimental setup as in §5.2 but terminate the cloak optimization once a cloaked image is 20% of the original $L2$ distance from the target image. The cloak still achieves a 100% protection success rate, but the cloak/target separation remains large enough to evade the previous detection method.

Anomaly Detection w/ Original Images. When the trackers have access to original training images (see §7), they could use clustering to see if there are two distinct feature clusters associated with the user's images (i.e. cloaked and uncloaked). Normal classes should have only one feature cluster. To do this, the tracker could run a 2-means clustering on each class's feature space, flagging classes with two distinct centroids as potentially cloaked. When we run this experiment, we find that the distance between the two centroids of a protected user class is 3 standard deviations larger than the average centroid separation in normal classes. In this way, the tracker can use original images to detect the presence of cloaked images.

To reduce the probability of detection by this method, the user can choose a target class that does not create such a large feature space separation. We empirically evaluate this mitigation strategy using the same experimental configuration as in §5.2 but choose a target label with average (rather than maximal) distance from their class. The cloak generated with this method still achieves a 100% protection success rate, but $L2$ distance between the two cluster centroids is within 1 standard deviation of average.

The user can evade this anomaly detection strategy using the maximum distance optimization strategy in §4. In practice, for any tracker model with a moderate number of labels (>30), cloaks generated with average or maximum difference optimization consistently achieves high cloaking success. Our experimental results show these two methods perform identically in protection success against both our local models and the Face++ API.

9 Discussion and Conclusion

In this paper, we present a first proposal to protect individuals from recognition by unauthorized and unaccountable facial recognition systems. Our approach applies small, carefully computed perturbations to cloak images, so that they are shifted substantially in a recognition model's feature representation space, all while avoiding visible changes. Our techniques work under a wide range of assumptions and provide 100% protection against widely used, state-of-the-art models deployed by Microsoft, Amazon and Face++.

Like most privacy enhancing tools and technologies, Fawkes can also be used by malicious bad actors. For example, criminals could use Fawkes to hide their identity from agencies that rely on third-party facial recognition systems like Clearview.ai. We believe Fawkes will have the biggest impact on those using public images to build unauthorized facial recognition models and less so on agencies with legal access to facial images such as federal agencies or law enforcement. We leave more detailed exploration of the trade-off between user privacy and authorized use to future work.

Protecting content using cloaks faces the inherent challenge of being *future-proof*, since any technique we use to cloak images today might be overcome by a workaround in some future date, which would render previously protected images vulnerable. While we are under no illusion that this proposed system is itself future-proof, we believe it is an important and necessary first step in the development of user-centric privacy tools to resist unauthorized machine learning models. We hope that followup work in this space will lead to long-term protection mechanisms that prevent the mining of personal content for user tracking and classification.

Acknowledgments

We thank our shepherd David Evans and anonymous reviewers for their constructive feedback. This work is supported in part by NSF grants CNS-1949650, CNS-1923778, CNS-1705042, and by the DARPA GARD program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any funding agencies.

References

[1] http://apodeline.free.fr/DOC/libjpeg/libjpeg-3.html. Using the IJG JPEG library: Advanced features.

[2] https://aws.amazon.com/rekognition/. Amazon Rekognition Face Verification API.

[3] https://azure.microsoft.com/en-us/services/cognitive-services/face/. Microsoft Azure Face API.

[4] https://www.faceplusplus.com/face-searching/. Face++ Face Searching API.

[5] http://vision.seas.harvard.edu/pubfig83/. PubFig83: A resource for studying face recognition in personal photo collections.

[6] ABADI, M., CHU, A., GOODFELLOW, I., MCMAHAN, H. B., MIRONOV, I., TALWAR, K., AND ZHANG, L. Deep learning with differential privacy. In *Proc. of CCS* (2016).

[7] CAO, Q., SHEN, L., XIE, W., PARKHI, O. M., AND ZISSERMAN, A. VGGFace2: A dataset for recognising faces across pose and age. In *Proc. of IEEE FG* (2018).

[8] CARLINI, N., AND WAGNER, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proc. of AISec* (2017).

[9] CARLINI, N., AND WAGNER, D. Towards evaluating the robustness of neural networks. In *Proc. of IEEE S&P* (2017).

[10] CARLINI, N., AND WAGNER, D. Towards evaluating the robustness of neural networks. In *Proc. of IEEE S&P* (2017).

[11] CHEN, B., CARVALHO, W., BARACALDO, N., LUDWIG, H., EDWARDS, B., LEE, T., MOLLOY, I., AND SRIVASTAVA, B. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv:1811.03728* (2018).

[12] CHEN, Y., LI, H., TENG, S.-Y., NAGELS, S., LI, Z., LOPES, P., ZHAO, B. Y., AND ZHENG, H. Wearable microphone jamming. In *Proc. of ACM CHI* (April 2020).

[13] CROSS, J. Valley attorney: Facebook facial recognition carries identity theft risk. *KTAR News* (September 2019).

[14] DEMONTIS, A., MELIS, M., PINTOR, M., JAGIELSKI, M., BIGGIO, B., OPREA, A., NITA-ROTARU, C., AND ROLI, F. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *Proc. of USENIX Security* (2019), pp. 321–338.

[15] DWORK, C. Differential privacy: A survey of results. In *Proc. of TAMC* (2008).

[16] FEINMAN, R., CURTIN, R. R., SHINTRE, S., AND GARDNER, A. B. Detecting adversarial samples from artifacts. *arXiv:1703.00410* (2017).

[17] FREDRIKSON, M., JHA, S., AND RISTENPART, T. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proc. of CCS* (2015).

[18] GOODFELLOW, I. J., SHLENS, J., AND SZEGEDY, C. Explaining and harnessing adversarial examples. *arXiv:1412.6572* (2014).

[19] GUPTA, N., HUANG, W. R., FOWL, L., ZHU, C., FEIZI, S., GOLDSTEIN, T., AND DICKERSON, J. P. Strong baseline defenses against clean-label poisoning attacks. *arXiv:1909.13374* (2019).

[20] HILL, K. The secretive company that might end privacy as we know it. *The New York Times* (January 18 2020).

[21] HILL, K., AND KROLIK, A. How photos of your kids are powering surveillance technology. *The New York Times* (October 11 2019).

[22] HUANG, G., LIU, Z., VAN DER MAATEN, L., AND WEINBERGER, K. Q. Densely connected convolutional networks. In *Proc. of CVPR* (2017).

[23] JAN, S. T., MESSOU, J., LIN, Y.-C., HUANG, J.-B., AND WANG, G. Connecting the digital and physical world: Improving the robustness of adversarial attacks. In *Proc. of AAAI* (2019).

[24] KOMKOV, S., AND PETIUSHKO, A. Advhat: Real-world adversarial attack on arcface face id system. *arXiv:1908.08705* (2019).

[25] KURAKIN, A., GOODFELLOW, I., AND BENGIO, S. Adversarial examples in the physical world. *arXiv:1607.02533* (2016).

[26] LEE, N. Having multiple online identities is more normal than you think. Engadget, March 2016. https://www.engadget.com/2016/03/04/multiple-online-identities.

[27] LI, T., AND LIN, L. Anonymousnet: Natural face de-identification with measurable privacy. In *Proc. of CVPR* (2019).

[28] LI, Y., YANG, X., WU, B., AND LYU, S. Hiding faces in plain sight: Disrupting AI face synthesis with adversarial perturbations. *arXiv:1906.09288* (2019).

[29] LIU, Y., CHEN, X., LIU, C., AND SONG, D. Delving into transferable adversarial examples and black-box attacks. *arXiv:1611.02770* (2016).

[30] MADRY, A., MAKELOV, A., SCHMIDT, L., TSIPRAS, D., AND VLADU, A. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083* (2017).

[31] MARI, A. Brazilian retailer quizzed over facial recognition tech. *ZDNet* (March 2019).

[32] METZ, C., AND COLLINS, K. How an A.I. 'cat-and-mouse game' generates believable fake photos. *The New York Times* (January 2018).

[33] MOZUR, P. Inside China's dystopian dreams: A.I., shame and lots of cameras. *The New York Times* (July 2018).

[34] NECH, A., AND KEMELMACHER-SHLIZERMAN, I. Level playing field for million scale face recognition. In *Proc. of CVPR* (2017).

[35] NEWTON, E. M., SWEENEY, L., AND MALIN, B. Preserving privacy by de-identifying face images. *IEEE transactions on Knowledge and Data Engineering 17*, 2 (2005), 232–243.

[36] NG, H.-W., AND WINKLER, S. A data-driven approach to cleaning large face datasets. In *Proc. of ICIP* (2014).

[37] NOCEDAL, J., AND WRIGHT, S. Numerical optimization, series in operations research and financial engineering. *Springer, New York, USA, 2006* (2006).

[38] O'FLAHERTY, K. Facial recognition at u.s. airports. should you be concerned? *Forbes* (March 2019).

[39] PAPERNOT, N., MCDANIEL, P., AND GOODFELLOW, I. Transferability in machine learning: From phenomena to black-box attacks using adversarial samples. *arXiv:1605.07277* (2016).

[40] PAUDICE, A., MUÑOZ-GONZÁLEZ, L., GYORGY, A., AND LUPU, E. C. Detection of adversarial training examples in poisoning attacks through anomaly detection. *arXiv:1802.03041* (2018).

[41] SATARIANO, A. Police use of facial recognition is accepted by British court. *The New York Times* (September 2019).

[42] SHAFAHI, A., HUANG, W. R., NAJIBI, M., SUCIU, O., STUDER, C., DUMITRAS, T., AND GOLDSTEIN, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Proc. of NeurIPS* (2018).

[43] SHAN, S., WENGER, E., WANG, B., LI, B., ZHENG, H., AND ZHAO, B. Y. Gotta catch 'em all: Using honeypots to catch adversarial attacks on neural networks. In *Proc. of CCS* (Orlando, FL, November 2019). arXiv:1904.08554.

[44] SHARIF, M., BHAGAVATULA, S., BAUER, L., AND REITER, M. K. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proc. of CCS* (2016).

[45] SHEN, J., ZHU, X., AND MA, D. Tensorclog: An imperceptible poisoning attack on deep neural network applications. *IEEE Access 7* (2019), 41498–41506.

[46] SHEN, S., TOPLE, S., AND SAXENA, P. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proc. of ACSAC* (2016).

[47] SHWAYDER, M. Clearview AI's facial-recognition app is a nightmare for stalking victims. *Digital Trends* (January 2020).

[48] SONG, C., RISTENPART, T., AND SHMATIKOV, V. Machine learning models that remember too much. In *Proc. of CCS* (2017).

[49] STEINHARDT, J., KOH, P. W. W., AND LIANG, P. S. Certified defenses for data poisoning attacks. In *Proc. of NeurIPS* (2017).

[50] SUCIU, O., MĂRGINEAN, R., KAYA, Y., DAUMÉ III, H., AND DUMITRAŞ, T. When does machine learning fail? generalized transferability for evasion and poisoning attacks. In *Proc. of USENIX Security* (2018).

[51] SUN, Q., MA, L., JOON OH, S., VAN GOOL, L., SCHIELE, B., AND FRITZ, M. Natural and effective obfuscation by head inpainting. In *Proc. of CVPR* (2018).

[52] SUN, Q., TEWARI, A., XU, W., FRITZ, M., THEOBALT, C., AND SCHIELE, B. A hybrid model for identity obfuscation by face replacement. In *Proc. of ECCV* (2018).

[53] SZEGEDY, C., IOFFE, S., VANHOUCKE, V., AND ALEMI, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proc. of AAAI* (2017).

[54] SZEGEDY, C., ZAREMBA, W., SUTSKEVER, I., BRUNA, J., ERHAN, D., GOODFELLOW, I., AND FERGUS, R. Intriguing properties of neural networks. *arXiv:1312.6199* (2013).

[55] THYS, S., VAN RANST, W., AND GOEDEMÉ, T. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proc. of CVPR (workshop)* (2019).

[56] TSIPRAS, D., SANTURKAR, S., ENGSTROM, L., TURNER, A., AND MADRY, A. Robustness may be at odds with accuracy. *arXiv:1805.12152* (2018).

[57] WALLACE, G. K. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics 38*, 1 (1992).

[58] WANG, B., YAO, Y., SHAN, S., LI, H., VISWANATH, B., ZHENG, H., AND ZHAO, B. Y. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proc. of IEEE S&P* (2019).

[59] WANG, B., YAO, Y., VISWANATH, B., ZHENG, H., AND ZHAO, B. Y. With great training comes great vulnerability: Practical attacks against transfer learning. In *Proc. of USENIX Security* (2018).

[60] WANG, G., KONOLIGE, T., WILSON, C., WANG, X., ZHENG, H., AND ZHAO, B. Y. You are how you click: Clickstream analysis for sybil detection. In *Proc. of USENIX Security* (2013), pp. 241–256.

[61] WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. on Image Processing 13*, 4 (2004), 600–612.

[62] WANG, Z., SIMONCELLI, E. P., AND BOVIK, A. C. Multiscale structural similarity for image quality assessment. In *Proc. of Asilomar Conference on Signals, Systems & Computers* (2003), vol. 2, IEEE, pp. 1398–1402.

[63] WONDRACEK, G., HOLZ, T., KIRDA, E., AND KRUEGEL, C. A practical attack to de-anonymize social network users. In *Proc. of IEEE S&P* (2010).

[64] WU, Y., YANG, F., AND LING, H. Privacy-Protective-GAN for face de-identification. *arXiv:1806.08906* (2018).

[65] WU, Z., LIM, S.-N., DAVIS, L., AND GOLDSTEIN, T. Making an invisibility cloak: Real world adversarial attacks on object detectors. *arXiv:1910.14667* (2019).

[66] YANG, C., WU, Q., LI, H., AND CHEN, Y. Generative poisoning attack method against neural networks. *arXiv:1703.01340* (2017).

[67] YANG, Z., WILSON, C., WANG, X., GAO, T., ZHAO, B. Y., AND DAI, Y. Uncovering social network sybils in the wild. *ACM Transactions on Knowledge Discovery from Data (TKDD) 8*, 1 (2014), 1–29.

[68] YANG, Z., ZHANG, J., CHANG, E.-C., AND LIANG, Z. Neural network inversion in adversarial setting via background knowledge alignment. In *Proc. of CCS* (London, UK, November 2019).

[69] YI, D., LEI, Z., LIAO, S., AND LI, S. Z. Learning face representation from scratch. *arXiv:1411.7923* (2014).

[70] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks? In *Proc. of NeurIPS* (2014).

[71] ZHU, C., HUANG, W. R., SHAFAHI, A., LI, H., TAYLOR, G., STUDER, C., AND GOLDSTEIN, T. Transferable clean-label poisoning attacks on deep neural nets. In *Proc. of ICML* (2019).