anonyME - Anonymizing Facial Footprints

Computer Engineering 4th Year Final Project (Undergraduate)

Amit Segal*
Computer Science & Engineering
Hebrew University of Jerusalem
Jerusalem, Israel
amit.segal2@mail.huji.ac.il

Oriyan Hermoni*
Computer Science & Engineering
Hebrew University of Jerusalem
Jerusalem, Israel
oriyan.hermoni@mail.huji.ac.il

Mr. Aviv Gabbay**
Computer Science & Engineering
Hebrew University of Jerusalem
Jerusalem, Israel
aviv.gabbay@mail.huji.ac.il

***Abstract*—Personally identifying information (mostly in the form of images) is shared in internet communities (e.g. Facebook or Instagram) on a daily basis. These images can be processed algorithmically and used to train unauthorized deep neural network face recognition models by 3rd party "trackers". As data becomes more publicly available, this is becoming a real threat to user privacy.**

In this project, we present our work focusing on allowing individuals to inoculate their images against unauthorized deep learning models by adding perturbations that are nearly imperceptible to the naked eye which forces tracker models to misidentify the user.

The main challenge was to create a transferable solution (i.e. an input image with added perturbations should be misclassified on all face recognition models), without impacting normal image usage.

To overcome these challenges, we applied transferable machine learning techniques to our problem domain, while minimizing the "visual distance" between the output image and the input using a perceptual degradation metric [1].

We've tested our proposed solutions' ability by measuring its misclassification rate and visual degradation. With models already trained on the user's identity, misclassification rate varies between 7% and 20% depending on the perceptual perturbation budget. With models where the user was not known, misclassification rate was much higher, between 60% and 80%. We show that in some cases, a misclassification rate of 100% can be achieved without causing significant visual degradation.

I. Introduction

Personally identifying information (mostly in the form of images) is shared in internet communities (e.g. Facebook or Instagram) on a daily basis. This contributed greatly to today's proliferation of powerful facial recognition models. These pose a real threat to personal privacy. These models are implemented in production systems around the world without consent even by government and law enforcement agencies (e.g. in China [2], UK [3]). More notably, the availability of such data allows 3rd parties to canvas the internet and obtain personal user images using web scraping, allowing them to construct highly accurate facial recognition models with the ability to recognize these users without their knowledge or consent. An example for such a company is Clearview.ai, which was reported to have collected over 3 billion online images and trained a large-scale face recognition model which can recognize millions of private citizens [4]. These are also utilized by 2nd parties such as Google and Facebook that have trained in-house face recognition models on 200 million images spanning 8 million subjects [5] , and 500 million images of over 10 million subjects, respectively [6]. This poses a great threat to personal privacy, as these models can be used to identify and track us using public street cameras, security cameras, and cellphones (e.g. to track location of protesters), or to track our in-store shopping behaviors back to our personage (e.g. for targeted advertising), and even to gain access to our personal accounts using our likeness. Hence, We believe there is a necessity for tools that prevent identification by unauthorized facial recognition models. Unfortunately, previous work in this space is limited in its practical usage. Some have proposed distorting images to make them unrecognizable in order to avoid face recognition [7] [8] [9]. Others produce adversarial patches which can be worn by printing them on sweatshirts, signs, glasses, etc. which prevent facial recognition algorithms from ever detecting the wearer's face at all [10] [11]. Alongside those, there exist more manual solutions which assume the users hold certain knowledge and skills like image editing, or are familiar

(*) Co-authors (**) Advisor

with the machine learning field.
Our solution focuses on its practicality, accessibility and minimal prerequisites on the user's end, as well as minimizing visual distortion.
In this project, we propose **anonyME**, a system which allows users to inoculate their personal images against unauthorized machine learning models, with minimal distortion of the input image, without wearing specialized clothing articles, and with no requirement of any previous knowledge in machine learning or face recognition. **anonyME** achieves this by adding imperceptible perturbations to input images. Our solution is end-to-end, including a web service and a smartphone application as well.
anonyME calculates these perturbations by applying gradient-based adversarial attacks against a local face recognition model specifically trained to mimic a tracker's decision boundaries (substitute model), knowledge acquired by querying said tracker. This technique was shown to be transferable in previous work [12]. For comparison, we also tested **anonyME** using a local fully trained face recognition model as a substitute model in order to reduce the probability of the attack being detected. Additionally, to allow normal use of the image after adding the perturbations we constrain the maximum perceptual degradation of the output image from the input image.

A. Key Results

In our work, we have reached several meaningful results:

- Gradient-based attacks achieve low misclassification rates on high-confidence images (i.e. images where the tracker initially recognizes the user with confidence $> 80\%$)
- Misclassification rates are much higher on low-confidence images, which means our method translates well to models not tracking the user.
- Our method can be further improved by better training the substitute model, and applying more sophisticated adversarial attack methods.
- Using a fully trained substitute model instead of training it, an act which eliminates the need to query the tracker for labeling information and minimizes the risk of being detected, we reach 100% misclassification rate.
- We reached inconclusive results w.r.t. transferability, as our trained substitute model achieved low success rate for high confidence images for both black-box models. However, when using a fully trained model as our subtitute, our attack achieved 100% transferability (i.e. all adversarial samples generated by attacking the fully trained substitute, forced misclassification on a black-box model of a different architecture).

II. Background and Related Work

A. Background

For the most part, when discussing identity validation via online means, we refer to face recognition technologies, more specifically implemented with machine learning algorithms.
These algorithms can be divided into two categories:
White-box algorithms. Publicly available, have an exposed implementation/API, and in the case of deep learning models the learned weights are easily accessible.
Black-box algorithms. Are publicly available, but we have no knowledge of their implementation. In the case of deep learning models, this also means we have no access to the model's learned weights.
An **Adversarial attack** is the act of crafting specific input tailored to cause deep learning models to misclassify.
Targeted vs. Untargeted adversarial attacks. In targeted attacks, the goal is to cause the model to misclassify the input as a specific target class instead of the input's original class. For example, if the input is classified as Alice but we want it to be classified as Bob, we add perturbations to the input image to force the model to infer Bob from the input. Untargeted attacks, on the other hand, simply force the input image to appear as something that it is not. These attacks add perturbations to the input image such that the probability of it belonging to the original class becomes lower than the probability of it belonging to a different class. Simply put, targeted attacks maximize the probability of the input belonging to a different, specific class while untargeted attacks minimize the probability of the input belonging to the original class.
In our context, the algorithms we wish to attack use photos of faces as their input, and learn how to label them with our identities. To counter that, we perform adversarial attacks on images users intend on uploading to social media **before** doing so, inserting small perturbations into these images. This should mask our identities from algorithms already trained on images of ourselves (**evasion by misidentification**), and prevent algorithms that are not yet familiar with us

from learning how to recognize us by contaminating the potential training data (**data set poisoning)**).
Thus our **problem context** is **Face recognition**, and our **problem domain** is **Adversarial attacks on Machine Learning models** using White-box/Black-box attack methods.

B. Related Work

- "Explaining and Harnessing Adversarial Examples" - This article presents a simple and effective white-box attack method called Fast Gradient Sign Method (FGSM) we use in our work [13].
- "Practical Black-Box Attacks against Machine Learning" - This article gives an example of a possible adversarial attacks on black-box Deep Neural Networks (DNNs). The article suggests training a local DNN using the black-box output as training labels, effectively substituting the black-box attack attempt with a white-box attack. Then, generate synthetic data by an adversary, in order to fool the local DNN. By doing so, they suggest adversarial samples crafted to fool the local DNN can successfully be used to fool the black-box DNN. We base our system on the ideas suggested in [12].
- "Fawkes: Protecting Privacy against Unauthorized Deep Learning Models" - a new article published about a month **after** we presented our project to the judging committee. In this work, the researchers constructed a system similar to ours that attacks face recognition models by performing data set poisoning, and adding small perturbations to images that are nearly imperceptible. They also provide a desktop application that creates a simpler interface for this process. Our main comparison points will then be to this work, since it is the only current solution to the problem we are trying to solve that approaches it from the same perspective as us, as there are some parallels [14].

The works mentioned above present practical, robust methods. However, some are specific to different problem domains (e.g. image classification and not face recognition). Additionally, these attack do not focus on the perceptual degradation of the attacked images, but rather on misclassification, with the exception of Fawkes' cloaking system.
Unfortunately, these methods are not easily accessible by day-to-day users who lack knowledge in the ML field (even Fawkes requires users to acquire a generic feature extractor DNN before being able to use their desktop application).
Let us present a solution that extends these methods and makes them accessible to end users – a RESTful API (also accessible via a smartphone app, published to an 'app store') that automatically introduces small, nearly undetectable perturbations to photos causing ML algorithms to misclassify. The app will allow users to remain present in online platforms, while maintaining their privacy.
We focus on keeping the modified images as close to the original by focusing on minimizing their perceptual degradation.

III. The Problem

A. Threat Model

The user's goal is to be able to share their likeness on social media, to relatives and friends, without fearing misuse of their identity. This includes preventing social media platforms (hosting the user's images) from misusing their identities for fine tuning their current algorithms, as well as disarm the possibility of 3rd party trackers/face recognition learners from being able to use these personal images without consent from the user. Both social media platforms and 3rd party trackers may take advantage of the user's willingness to upload personal images meant to be shared with the user's friends, and create models that learn to recognize the user without getting actual consent.
Thus, we assume the user wishes to share personal images online while evading detection and tracking from 3rd party sources who may already possess the user's unmasked images (e.g. previously uploaded images to social media), and from sources who already have a trained model in their possession but have no access to unmasked images of the user (i.e. trackers that wish to train to identify the user but the user has already masked their publicly available images).
We assume the user wishes to modify their images as little as possible to retain their visual likeness and appear normally to other human viewers, but appear drastically different to machine learners.

B. Assumptions

We assume the 3rd party looking to use the user's data is already in possession of a high-accuracy trained tracker/face recognition model trained on millions of images and thousands of identities, that has either previously trained on unmasked images of the user or wishes to fine tune their model to learn to recognize the user using current images.

This assumption best reflects the real-world scenario since a user has most likely been sharing personally identifying information online without masking for many years, possibly over a decade, and has shared hundreds or even thousands of images of themselves. We can't control the past, but we intend to reduce the probability of correct identification by taking protective measures from this point forward. Even if the user has been taking privacy enhancing measures for a long time, they don't have complete control over who uploads images of themselves, including friends and family.

The user can help improve their sense of privacy by untagging themselves from images and opting out from such services, and applying our anonymization algorithm on newly-uploaded images. However, this is out of the scope of our work and we focus mainly on actively anonymizing facial footprints before uploading to social media.

In this project, we assume a weak adversary, only able to access the output of the target model. The adversary can query the target model for labeling data, by inputting an image and acquiring the output probability vector for the different classes of the model, but has no prior knowledge of the internals of the target model, including but not limited to architecture, number, type and size of the layers, learn weights, and more.

In real-life situations, it is probable that we will have at most the top-1 label (e.g. Azure Face API, or top-3/4 on Facebook), and even then it is only on some platforms - since we usually have no idea if someone has scraped our data for their 3rd party trackers we are then unaware of the actual tagging information on input images. This means that proving with certainty whether our algorithm works or not is impossible in such circumstances and we must simplify our model this way.

C. Misclassification and Transferability

DNN models have been shown to be easily susceptible to adversarial attacks [15], especially if the attacker has direct access to the model's weights and can perform gradient-based augmentation of the input [13]. This means we can intelligently add small perturbations to personal images and cause face recognition DNNs to fail in the classification task of assigning a label (i.e. identity) to an input image. By doing so, we increase the probability of false-identification for trained models. Similarly, if a model is trained on masked data then the feature space it is trained on can be drastically modified and the labels assigned to images while training may not reflect the actual class of the image, since we introduce noise such that images no longer clearly belong to one class or another (data set poisoning), demonstrated by [14]. We achieve this by performing untargeted advesrarial attacks. This increases the probability of misclassification.

The **transferability** property in our context means using the same masking mechanism regardless of the target DNN we are attacking. Gradient based attacks are highly-specific to the weights of the model we are attacking (white-box attacks), and the same augmented image will not be classified similarly on models with different architectures, since they span different features spaces. This means gradient-based attack is not guaranteed to be transferable. We negate this by applying black-box attack methods with the hope of increasing transferability and generalizing of our attack. This reduces the effect of our attack when applied only to a single model, and if the goal is attacking a specific model it is preferred to use more fine-tuned attack methods.

IV. Our Solution - anonyME Facial Footprint Anonymization

We now present anonyME, our practical end-to-end solution for anonymizing the facial footprints of users allowing them to maintain their online presence without compromising their identities, while evading identification by face recognition models. Our solution is comprised of 6 parts:

- **Black-box oracle** - a black-box target face recognition model we wish to attack. We assume we can query it with input to acquire the probability vector of the different classes, and infer the top-1 label for the input. We assume the oracle has detection mechanisms for irregular amounts/schedule of queries, so we aim to decrease the amount of queries as much as possible to avoid detection.
- **Substitute model** - This is a model we train to mimic the decision boundaries of the oracle, by querying the oracle for labeling data and labeling our training data accordingly. Since the amount of queries we can make to the oracle is limited, we need to augment this training set artificially to increase the effectiveness of our training.
- **White-box attack method** - an adversarial attack mechanism we use to attack the substitute directly as its weights are available to us and we can take advantage of the high efficiency of white-box attacks on DNNs. By attacking the substitute, we

should be able to use the same perturbations to attack the oracle successfully.

- **Perceptual Loss Metric** - In order to minimize the visual degradation of the attacked images, we quantify it by using a perceptual loss metric that measures the visual degradation between the original input and the augmented image, and determine the numerical cutoff threshold above which we consider images to be visibly degraded, since users want minimal difference between the original and the perturbed image.
- **RESTful API and Web Service** - an easily accessible interface for developers who wish to extend our engine. The web service hosts the substitute model and the attack system, and accepts input images via a RESTful API, performs the attack on the input image, and returns the perturbed image via a REST response.
- **Android Application** - we also developed an Android application that provides a simple interface for end-users who don't wish to get involved with the mechanism, and only wish to protect their privacy as simply as possible. The application has an in-app camera and gallery, and interfaces with our RESTful API. The user can share images to social media directly from our app after performing the masking process. Previously, our application performed the entire attack pipeline on-device but due to Google Play publication guidelines and performance issues we removed this capability temporarily. Our app is published in a closed-alpha on Google Play.

A. Black-box Attack Strategy and Algorithm

In this section we will describe the training process, the black-box attack, and the entire end-to-end pipeline. Let $\vec{x}$ be an input image of user U. Let O be the oracle, the targeted DNN we are attacking, and let $O(\vec{x})$ be the index of the output label of O with input $\vec{x}$, assigned to the largest probability by the DNN, i.e. :

$$O(\vec{x}) = \arg \max_{j \in \{0,1,...,N-1\}} O_j(\vec{x}) \quad (1)$$

where $O_j(\vec{x})$ is the j-th component of the probability vector outputted for $O(\vec{x})$ by DNN O.

Our advesrarial goal is to produce a minimally altered version of input $\vec{x}$, the advesrarial sample $\vec{x^*}$ such that the DNN O misclassifies it, i.e. $\tilde{O}_j(\vec{x}) \neq \tilde{O}_j(\vec{x^*})$, where $\tilde{O}(\vec{x})$ is the label given to $\vec{x}$ by O. This corresponds to an attack on the oracle's output integrity. The adversarial sample solves for the optimization problem

$$\vec{x^*} = \vec{x} + \delta_{\vec{x}} = \vec{x} + \arg\min_{\vec{\delta}} \{\vec{\delta} : \tilde{O}(\vec{x} + \vec{\delta}) \neq \tilde{O}(\vec{x})\} \quad (2)$$

We solve for the minimal perturbation $\delta_{\vec{x}}$ that can satisfy U's desire to upload the output $\vec{x^*}$ to social media with minimal visual degradation, while at the same time satisfying $\tilde{O}_j(\vec{x}) \neq \tilde{O}_j(\vec{x^*})$.

The assumption that our adversary only has access to output labels weakens it and makes it harder to find such a perturbation. To counter that, we trained a **substitute model F** that approximates O, to which we have total internal access. Training such a substitute is challenging because we must first select an architecture for our substitute without prior knowledge of the target oracle's architecture, and we must limit the number of queries we make to the oracle to avoid detection - if we make an unlimited number of queries to the oracle, we increase the tracker's suspicion that someone is attempting to learn the behavior of the oracle. Another limitation is not being aware of the number of classes the oracle is familiar with - this, combined with the limitation on the number of queries means F cannot span the entire feature space spanned by O.

To increase our chances to succeed in this task, we had to increase the size of our data set artificially, which in turn increased the accuracy of F's predictions on our input and the success rate of the attack. This process is detailed later in our report.

After successfully training substitute F, to generate $\vec{x^*}$ from input $\vec{x}$ we implemented a gradient-based white-box attack method called Fast Gradient Sign Method [13] and applied it to the input and output of F on $\vec{x}$. That is, we solved for equation (2) but replaced $\tilde{O}$ with $\tilde{F}$. We add a constraint to the optimization process that keeps the amount of perturbations below a certain threshold T - this is quantified using a perceptual degradation metric $L_{PD}(\vec{x}, \vec{x^*})$, meaning we stop adding perturbations once F's prediction has changed or once $L_{PD}(\vec{x}, \vec{x^*}) > T$. $\vec{x^*}$, it is then assumed to sufficient for solving (2) w.r.t. the oracle O. The entire white-box attack algorithm is described below, including the manner with which we chose T.

To complete the pipline, we developed a web service W that accepts images $\vec{x}$ as input and returns images $\vec{x^*}$ as output. To add to that, we developed an Android application anonyME that provides a simple interface for users to manipulate images. The app forwards requests $S = \{\vec{x}_1, \vec{x}_2, ...\}$ from the user to W, and processes

responses $R = \{\vec{x}_1^*, \vec{x}_2^*, ...\}$ from W. After sending requests S and receiving responses R, U can then chose to upload R to social media or other services.
Now that we described how the attack pipeline works, we can describe our end-to-end solution pipeline:
User U wishes to upload images $S = \{\vec{x}_1, \vec{x}_2, ...\}$ to social media. U opens the anonyME application, and selects S for manipulation. Behind the scenes, anonyME forwards S to W. For each $\vec{x} \in S$, W invokes the white-box attack which solves for $\vec{x^*} = \vec{x} + \delta_{\vec{x}}$ w.r.t. F (see (2)), and appends $\vec{x^*}$ to response list R. Once all output images are ready, W sends R to the app anonyME where U can use the images in R in whatever manner they want, including sharing to social media.

B. Black-box Oracle - The Opponent

As our oracle, we chose 2 pre-trained models based on the architectures RESNET50 [16] and SENET50 [17], both trained on the data set VGGFace2, which contains over 3.3 million images, spanning 9,000 identities [18] (weights obtained from repo implementing these architectures using Keras library [19]). We originally intended to test ourselves against a trained Microsoft Azure Face recognition model on top of that, but we ran out of credits before completing training and so left it to future work.

C. Substitute Training and Data Set Augmentation

We performed substitute model training and data set augmentation as suggested by Papernot. et al. [12], with some modifications: the original article assumes a relatively small number of classes the target is familiar with and low input dimensionality (MNIST: 10 classes, 28x28 input, using 15 images per class for substitute model training; GTSRB: 43 classes, input resized to 32x32, using ~ 23 images per class for substitute model training). Additionally, both MNIST and GTSRB are relatively small data sets (10000 samples and ~ 50000 samples, respectively).
For the task of face recognition, the scale is completely different - as mentioned before, in the data set we are using (VGGFace2) there are ~ 8600 different identities with over 3 million samples. If we want to use the same proportion of images/classes as presented in the article for MNIST we must use $\sim 8600 \times 15 = 129000$ samples , and in comparison to the higher-dimensional GTSRB data set we must use $\sim 8600 \times 23 = 197800$ samples. This is unrealistic in the real world since this number of queries will be detected very fast, or will take an unusually long time to complete. Clearview.ai for example has amassed 1000 times this amount of samples so this ratio is unfeasible. Thus, we had to change our strategy [4].
Instead of our substitute model spanning the entire domain of classes, we will instead choose a smaller subset of classes we want to learn (50 classes) and ~ 95 images per class. Thus our initial subset of images has ~ 4750 images, which means a much more realistic number of queries that can probably evade detection by the black-box in a real setting. We will note that it is not a reasonable expectation to completely mimic the behavior of the oracle due to the small number of queries we can make, so our method addresses this issue by attempting to approximate the *decision boundaries* of the oracle by using a heuristic that explores its input domain. This fulfills our needs, since we are aiming to *change the oracle's decision* rather than aiming to achieve perfect 1:1 behavior duplication (this is a reasonable assumption because the same function may be approximated in more than one way, so 1:1 approximation does not guarantee similar decision boundaries).
In [12] it is mentioned that choice of architecture had very little effect on the accuracy of the substitute model and that the architecture should be suitable for the problem domain (e.g. images, text, etc.). In our case, we compared the performance of our method using two substitute model architectures - RESNET50 and SENET50 (chosen for simplicity of evaluation, since the architectures we chose for black-box models are the same architectures). Now, let us describe the original data set augmentation algorithm, and afterwards the modifications we made due to the aforementioned constraints.
(Taken from [12])
The heuristic used to generate synthetic training inputs is based on identifying directions in which the model's output is varying, around an initial set of training points. Such directions intuitively require more input-output pairs to capture the output variations of the target DNN O. Therefore, to get a substitute DNN accurately approximating the oracle's decision boundaries, the heuristic prioritizes these samples when querying the oracle for labels. These directions are identified with the substitute DNN's Jacobian matrix J_F, which is evaluated at several input points $\vec{x}$. Precisely, the adversary evaluates the sign of the Jacobian matrix dimension corresponding to the label assigned to input $\vec{x}$ by the oracle: $sign(J_F(\vec{x}))[\tilde{O}(\vec{x})]$. To obtain a new synthetic training point, a term $\lambda \cdot sign(J_F(\vec{x}))[\tilde{O}(\vec{x})]$.

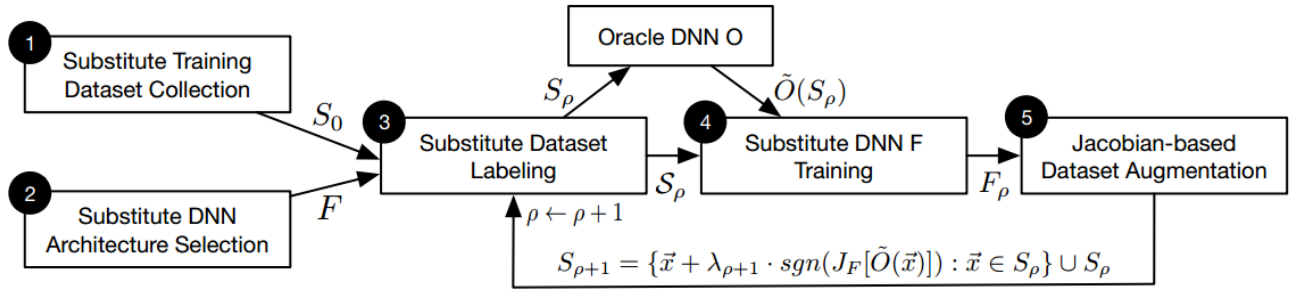


Fig. 1: **Training of the substitute DNN F (taken from [12]**
The attacker (1) collects an initial substitute training set S_0 and (2) selects an architecture F . Using oracle $\tilde{O}$, the attacker (3) labels S_0 and (4) trains substitute F . After (5) Jacobian-based dataset augmentation, steps (3) through (5) are repeated for several substitute epochs ρ.

*This is added to the original point $\vec{x}$. We name this technique **Jacobian-based Dataset Augmentation**. We base our substitute training algorithm on the idea of iteratively refining the model in directions identified using the Jacobian.*

Algorithm 1: Substitute DNN F Training (as seen in [12]): for oracle $\tilde{O}$, a maximum number max_ρ of substitute training epochs, a substitute architecture F , and an initial training set S_0.

input : $\tilde{O}, max_\rho, S_0, \lambda$
output: Substitute model parameters Θ_F
1 define architecture F;
2 **for** $\rho \in \{0, 1, ..., max_\rho\}$ l **do**
3 `// Label Training Set`
4 $D \leftarrow \{(\vec{x}, \tilde{O}(\vec{x})) : \vec{x} \in S_\rho\}$;
5 `// Train F on D to evaluate parameters` Θ_F
6 $\Theta_F \leftarrow train(F, D)$;
7 `// Perform Jacobian-based dataset augmentation`
8 $S_\rho \leftarrow \{\vec{x} + \lambda \cdot sign(J_F(\vec{x}))[\tilde{O}(\vec{x})] : \vec{x} \in S_\rho\} \cup S_\rho$;
9 **end**
10 **return** Θ_F

We will not go into more detail here (for that, refer to [12]), except note that every iteration of this algorithm we make queries to the oracle O for all the new images in the data set S_ρ (in comparison with the original algorithm, which repeats querying even for images we previously had in the data set, this way we reduced the number of queries by a factor of 2 and made only one query to the oracle per image in the data set). Additionally, since we only started with a small subset of 50 classes from the entire class domain, we did not want to limit ourselves to only those classes because we want to mimic decision boundaries - thus, we took it into account when querying O for labels in the labeling step (line 4) and added labels ad-hoc to our data set. Since this has the potential for adding noise to our model training, we only added new labels when we encountered more than 2 samples from a new class per iteration (i.e. we pruned class labels that had 2 samples or less after making the queries).
Considering this, with an initial subset of $\sim$ 4750 images (+20% for training validation) after 4 iterations substitute training (which includes 3 iterations of dataset augmentation and pruning) we ended up with about $\sim$ 36000 samples from 59 classes on which the fourth model was trained - still a large number of queries, but several times lower than estimated. Each iteration of substitute training was performed for 50 training epochs, with Adam optimizer and learning rate 0.001. The scaling factor λ was 0.01. The fifth model reached a validation accuracy of 82.2%.

D. Fast Gradient Sign Method

This white-box attack method is described in [13], but in short: let Θ be a DNN O's parameters, y the output of O with input x, and $L_T(x, y, \Theta)$ be the cost of training the network. Then, we calculate a constrained perturbation $\delta = \epsilon \cdot sign(\nabla_x L(x, y, \Theta))$ where ϵ is a scaling factor that limits the magnitude of perturbations. The perturbation δ is then added to input x: $x^* = x + \delta$ to create an adversarial sample. ϵ is chosen such that $\arg\max(O(x)) \neq \arg\max(O(x^*))$.

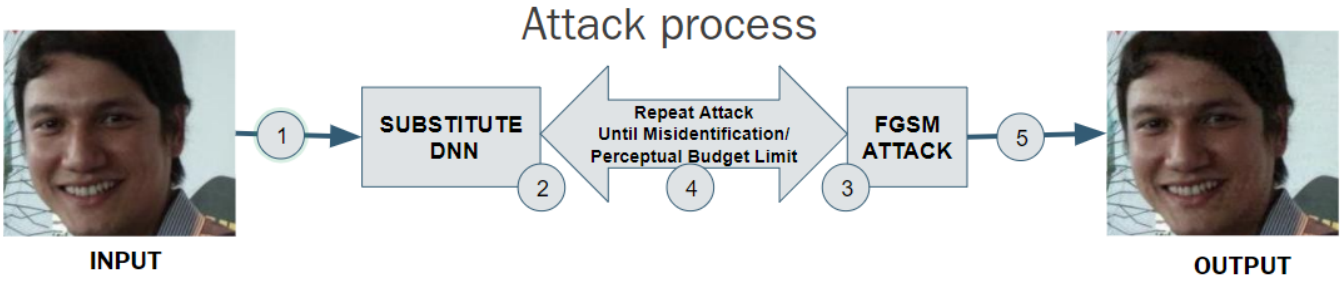


Fig. 2: **Flow of the Iterative FGSM Attack**
The attacker A (1) Inputs an image $\vec{x}$ and (2) selects trained substitute F and acquires F's prediction on $\vec{x}$, then A (3) performs a single FGSM iteration on $\vec{x}$ w.r.t. F (4) Repeat 2-3 until convergence limit or perceptual budget has run out (the assigned threshold values to our perceptual degradation metric, as defined [14]), and (5) return adversarial sample $\vec{x^*}$.

In our case, we chose a small enough ϵ so we can find the minimal perturbation δ that holds the aforementioned inequality. We chose $\epsilon = 0.004$, and to find the minimal perturbation we performed this process iteratively:

Algorithm 2: Iterative FGSM

input : Oracle O, Image $\vec{x}$, factor ϵ, max number of iterations n, perceptual degradation threshold T
output: Minimally perturbed image $\vec{x^*}$

$orig\_pred \leftarrow \arg\max O(\vec{x}), i \leftarrow 0, \vec{adv_x} \leftarrow \vec{x}$;
// Create unit vector from original output probability vector
$\vec{y} \leftarrow e_{orig\_pred}$;
while $i < n$ **do**
  $\vec{pred} \leftarrow O(\vec{adv_x})$;
  $loss \leftarrow CrossEntropy(\vec{pred}, \vec{y}; \Theta_O)$;
  // The FGSM step:
  $\delta \leftarrow \epsilon \cdot sign(\nabla_x loss)$;
  $\vec{adv_x} \leftarrow \vec{adv_x} + \delta$;
  if $PerceptualLoss(\vec{adv_x}, \vec{x}) \geq T$ **then**
    Break;
  else
    $i \leftarrow i + 1$;
  end
  $\vec{x^*} \leftarrow \vec{adv_x}$;
end
return $\vec{x^*}$

Where PerceptualLoss is the perceptual degradation metric we use to quantify the visual degradation of the input image [1].

Note that our attack is untargeted which holds true to our threat model assumptions. To test our method, we performed this attack with slight variations on a trained FaceNet [5] white-box model - since FaceNet doesn't predict classes but only generates feature vectors from input images, we trained a simple linear classifier on our data set using FaceNet as the first layer in order to measure our success rate on FaceNet. We modified FGSM to a targeted attack, such that instead of creating a unit vector from the original class label we instead use the extracted feature vector from a target image, and instead of using CrossEntropy we used L2 loss between the input image feature vector, and the target feature vector. Using this method, we reached 100% success rate (successfully changing labels while keeping the perceptual degradation below the threshold).
However, since this is a white-box attack we expected the same results to hold for the substitute model F we are attacking directly, thus we need to change the stop condition in the FGSM loop to accommodate attacking a black-box model.
Using the iterative FGSM to attack a trained substitute model, we achieved 100% success rate (requiring at most 2-3 iterations, meaning convergence is fast).

E. Visual Degradation Metric

As our visual degradation metric, we chose the trained perceptual loss LPIPS [1] with AlexNet weights. We performed visual inspection of images generated in the iterative FGSM process and chose two loss thresholds (perceptual budgets): $T_1 = 0.1, T_2 = 0.2$. Empirically, we found that images with $PerceptualLoss(\vec{x}, \vec{adv_x}) \leq T_1$ had barely visible perturbations and could easily be uploaded to social media without users compromis-

ing for image quality, whereas images holding $T_1 < PerceptualLoss(\vec{x}, \vec{adv_x}) \leq T_2$ had more noticeable perturbations, but do not considerably impact normal image use. If the perceptual loss is $> T_2$, we don't believe users would consider using the images at all as the visual degradation is too great. We leave better fine tuning of this threshold to future work as the values chosen in this work were good enough for our purposes. We experimented using L1 and L2 losses as well since they are less computationally complex, but found that visual degradation (with visual inspection) was not as consistent with these loss metrics as with LPIPS and a single threshold value was hard to calculate, and so we decided to focus on LPIPS for this task. Success rate of our attack is then calculated with respect to the visual degradation value, as well as the predicted output labels of the oracle on the adversarial sample.

F. Web Service and Android Application

We implemented our web service using Flask. It receives in its input a list of images as well as input parameters for the attack (e.g. ϵ) via a POST request. For each image in the request, we find the bounding box of the face in the image (if the bounding box is smaller than 224x224 we increase the size of the box to 224x224, otherwise we take a random crop of size 224x224 from the bounded area). The service then invokes the iterative FGSM attack on each cropped face, and sends a list of 224x224 perturbed faces back to the caller as response. Stitching back the cropped face to the input image is simple, but remains unimplemented for simplicity and is left for future work.

The Android application anonyME is published to Google Play as a closed-alpha. The app was tested on Android 8, has in-app camera and gallery and sharing capabilities. At first, we implemented the entire attack pipeline on-device for privacy reasons (we assumed users are more concerned with privacy than other users and would rather not have their images possibly-exposed online before being augmented), however as our model grew and our product was extended we saw big performance degradation on phones running our application and decided to use a web service hosted on secure servers for proof-of-concept . Thus, the pipeline between the app and the web service still needs to be hardened, this is left for future work. The web service and android applications are supplied as Proof of Concept. The web service can be accessed from our code base, whereas the application has a closed alpha release in Google Play (invitations will be sent per request).

G. Codebase and Implementation Details

Our project is a Python 3.7 project. "anonyME" package contains the code for our android app, while "attacks" package contains our main code (**attacks.whitebox** sub-package contains the code for our white-box attack presented as our MVP, while **attacks.blackbox** sub-package contains our black-box adversary). Our code can be accessed through "scripts" package, which contains scripts to train our substitute model, and evaluated our attack method using the trained substitute. Our trained model weights can be located in the Google Drive link in the README file at the root of our Github repository. These weights are to be placed in "weights" folder in the repo's parent directory. We've implemented our attack using TensorFlow 1.14.0, and Keras 2.3.x. In the root of our repo we've placed our web service which loads our trained substitute and produces adversarial samples by attacking it with input images received in incoming request. It is important to note that the web service is in PoC stages.

For face detection, we used MTCNN [20], for which we found a python package containing a trained model found in [21]. In order to implement LPIPS in our project, we've added the LPIPS repo implemented in PyTorch as a submodule to our repo - cloning our repo should clone theirs as well [22]. Entry points and running instruction and examples can be found in our repo's README file.

V. Evaluation

In this section we evaluate the effectiveness of anonyME. First, we describe the datasets, model architectures and other configurations we used in our testing. Initially, we evaluated the quality of iterative FGSM by directly attacking the trained black-box models - this was our baseline for comparison. For the same purpose, we evaluated our attack's success rate against a trained substitute model that did not train on an augmented dataset - this is a control test, to verify that we indeed reach higher efficacy when increasing the size of the original dataset using Jacobian-based dataset augmentation by comparing to a partially trained substitute model. This test is equivalent to a scenario where we want to minimize the amount of queries made to the oracle as much as possible and evade suspicion.

Further scenarios include the attacker using the same architecture as the target; Both using different architectures; Using a fully-trained face recognition model instead of a substitute model; Evaluation of attack success against images from classes neither oracle nor substitute

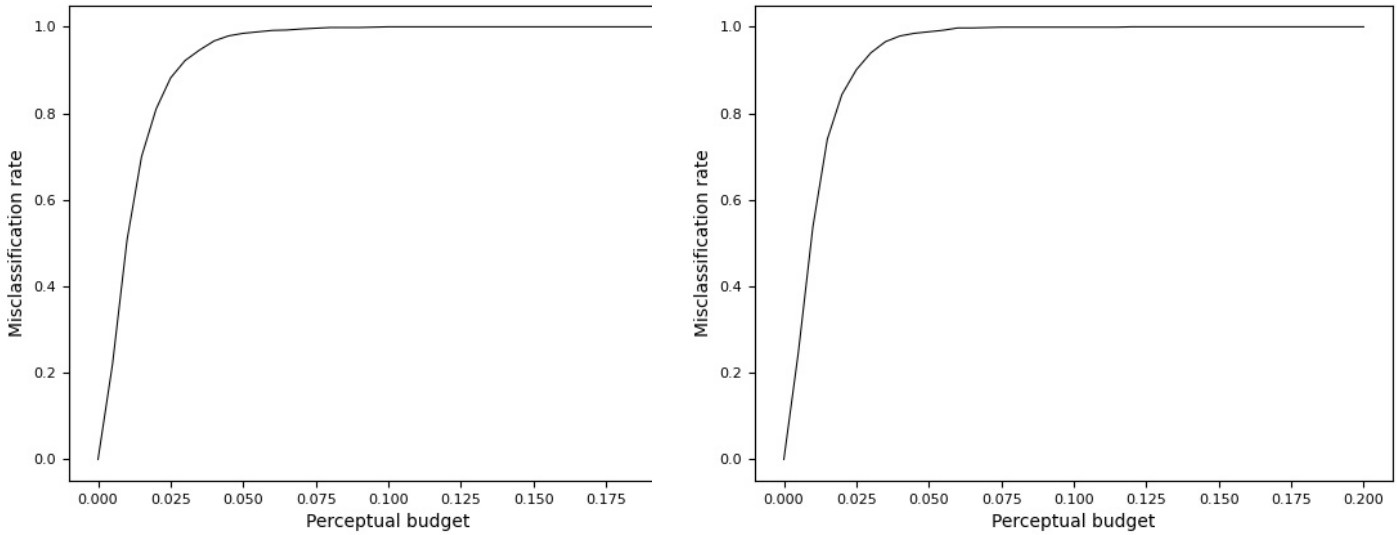


Fig. 3: **Cumulative image misclassification percentage by perceptual budget**
Misclassification percentage for both RESNET50 (left) and SENET50 (right) models. We can see that our attack method achieves misclassification on all images in validation set with LPIPS dist. from original image < 0.1

are familiar with.
Key findings include: Attack method is 100% successful when attacking directly as a white-box attack, as expected; Success rate for the partially-trained substitute model is indeed lower than that of a better-trained substitute; Success rate of the attack is higher when the initial confidence of the oracle's prediction is lower; Larger perceptual degradation thresholds give much higher success rates, at the cost of worse-looking results; and success rate of the attack against classes unknown to both oracle and substitute is several times higher than for known classes.

A. Experiment Setup

Our experiments required a fully trained target oracle black-box model; a large data set with which the oracle was trained, a secondary data set with classes unknown to the oracle - both data sets contain user data scraped online by trackers; a mechanism for extracting face data from input images, we chose MCTNN [20]; a trained model used for perceptual loss calculation - we chose LPIPS with AlexNet weights.
Oracle Selection There is a wide variety of freely available high-accuracy face recognition models, we chose two that are considered modern and achieved high accuracy when training on large data sets; the selected architectures are RESNET50 and SENET50, both fully-trained on VGGFace2 (both with over 90% accuracy on VGGFace2, with SENET50 outperforming RESNET50).
Datasets. We chose two publicly available data sets commonly used for the face recognition task, VGGFace2 and LFW [23]:

- **VGGFace2** - Large data set, contains 3.14M samples spread over 8631 identities of celebrities. This data set was scraped using Google Search. For this reason, it simulates our threat model well. We used a small, randomly selected subset of 50 classes for initial substitute training with an 80/20 split for training and validation sets, respectively. By doing so, we got $\sim 4,750$ training samples and $\sim 1,200$ validation samples. Before training, we sorted the samples according to the **actual predictions** of the oracle model used in the test, to simulate lack of preexisting knowledge over the data set. We repeated this once for each oracle architecture. We evaluated our attack mechanism using the validation set.
- **LFW** - Smaller data set, contains 13,233 samples divided into 5,749 identities. This data set shares some classes with VGGFace2. This data set is commonly used for benchmarking face recognition capabilities. We used it to simulate the case where

the target model has almost no prior knowledge of the user, to demonstrate the effectiveness of masking the user's images **before** uploading to social media.

Evaluation Metrics. We evaluated the success of every individual sample produced by anonyME under two criteria -

- **Small perceptual degradation** - we chose two numerical thresholds for LPIPS [1] - 0.1 and 0.2. A successful attack is an attack where the perceptual loss between the original image and the generated adversarial sample is less than one of these two threshold values. With 0.1 the added perturbations are nearly imperceptible and this simulates the case where a user does not have to compromise quality for privacy; With 0.2 the added perturbations are much more noticeable, but some privacy advocates might still choose to use these adversarial samples for increased privacy ; Such perturbations are noticeable but do not degrade much from the quality of the image. Both thresholds were calculated empirically, using visual inspection.
- **Misclassification rate** - A successful attack is an attack where the oracle has misclassified the adversarial sample. If an adversarial sample has fulfilled both the success criteria, it is considered a successful attack.

We would like to note that the results presented below for all experiments were obtained using RESNET50 substitute, we've also repeated them with SENET50 substitute trained on the same data set using the same technique, which yielded nearly identical results (with negligible differences).

B. White-box Attack Efficiency

In figure 3, we see attacking the oracle's with the white-box assumption yields nearly 100% success rates with low perceptual degradation of less than 0.075 (RESNET50) and 0.05 (SENET50). This is expected, but this is an unrealistic scenario so we consider it a baseline to evaluate that our attack indeed works. In figure 4, we see that in our attack the perturbations are added mostly in areas that are not part of the facial features, which implies that the sensitivity of the model is higher to modifications of non-essential pixels.

C. Partially Trained Substitute

In figure 5, we can see the partially trained substitute - trained without data set augmentation - yields low success rates of 3% with LPIPS threshold under 0.1, and 12% with 0.2. This is expected, since the model has a poor familiarity with the oracle, but also with its own training set.

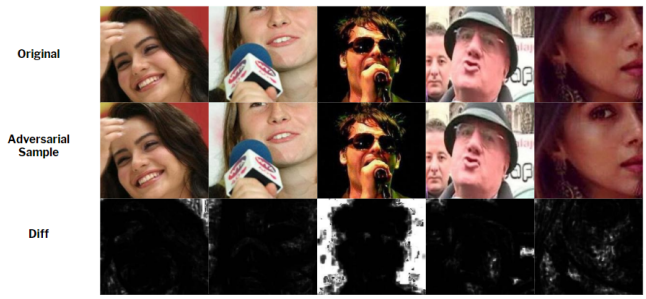


Fig. 4: **Examples of adversarial samples from white-box attack**

Examples of adversarial samples generated from attacking fully trained RESNET50 model directly, the top row contains the original image from the validation set, the middle row contains the adversarial samples, and the bottom row contains the diff between these images.

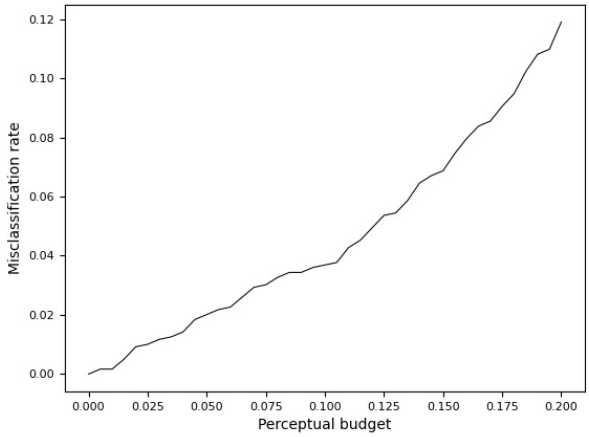


Fig. 5: **Partially trained substitute misclassification rate**

Cumulative misclassification rate for substitute model trained on small subset of VGGFace2 training set without data set augmentation

D. Trained Substitute - Shared Classes

We tested our fully trained substitute model (RESNET50 after 3 data set augmentations) against two different architectures. At first, we evaluated our attack against a black-box oracle using the same architecture, after which, we evaluated our attack against a black-box

model using a different architecture, but trained on the same dataset, in order to check if transferability holds.

1) Shared architecture: In this scenario, we tested using the validation set we used while training (note we did not use the validation set for fine tuning, only to measure training accuracy every epoch) which implies both oracle and substitute share a subset of the same class domain.

As we can see in figure 6a, the attack success rate with LPIPS threshold 0.1 is larger than in the previous experiment at 7.5% and 20% at 0.2 . This success rate is much lower than expected, which implies our process for training the substitute model was either flawed or that the task at hand is much harder than previously anticipated due to the black-box assumptions. In figure 6b, we can see that the attack succeeds at much higher rates when the initial prediction confidence of the oracle is lower, with over 50% success rate when confidence is at 0.5 and over 80% when the initial confidence is lower than 0.2. We can infer from this that our attack will be more successful if the target we are attacking is less confident with its predictions which can be caused by users publishing masked images from the get-go.

This scenario is more realistic than the previous one, however it is still not realistic enough since users generally do not know what the underlying architecture of the trackers is.

2) Transferability: We've also tested our substitute model's transferability as explained before by attempting to attack a black-box model using SENET50 architecture. As we can see in figure 7a, the attack success rate with LPIPS threshold 0.1 is larger than in the previous experiment at 5.1% and 12.3% at 0.2 . These results are similar to the results yielded in the previous experiment, and are explainable by the same reasons. In figure 7b, we can see that here too the attack succeeds at much higher rates when the initial prediction confidence of the oracle is lower, with $\sim 73\%$ success rate when confidence is at most 0.5 and with a similar success rate when it is at most 0.2.

E. Labeled Faces in the Wild - Unfamiliar Classes

In this scenario we further relax our assumptions and consider the case where the user is one of the identities in the LFW data set, and is trying to protect itself against a tracker that might or might not be familiar with it - the tracker trained on VGGFace2, which shares some identities with LFW. This is the most realistic scenario, since we do not assume prior knowledge of the oracle's architecture nor of its class domain. To simplify this process, we used our previously trained RESNET50 substitute (trained on VGGFace2) for the FGSM attack, and used a fully trained SENET50 oracle and measured our success rate on the LFW data set.

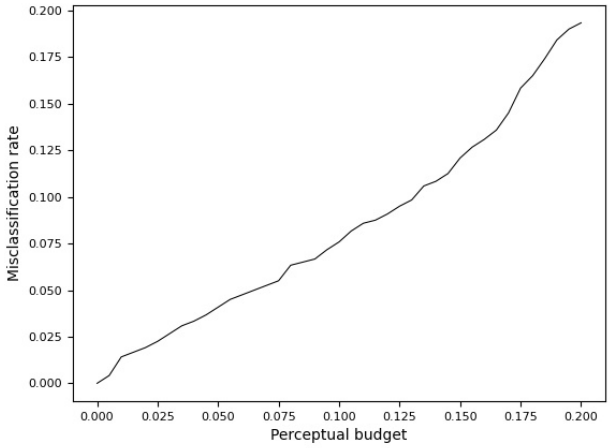
As seen in 8a, this scenario was much more successful than previous scenarios which is surprising due to the more realistic nature of this scenario. We achieved nearly 60% success rate with LPIPS threshold 0.1 with RESNET50 as a target, and nearly 80% with threshold 0.2. From figure 8b, we can see that this is attributed to the fact that the initial confidence of the oracle on the LFW images is much lower for most of the images which increases the probability of fooling it. This validates our theorem from the previous experiments. Targeting SENET50 instead of RESNET50 we reached similar results, but a few percentage points lower - $\sim 55\%$ and $\sim 75\%$ respectively.

F. Fully-trained Face Recognition Model as Substitute

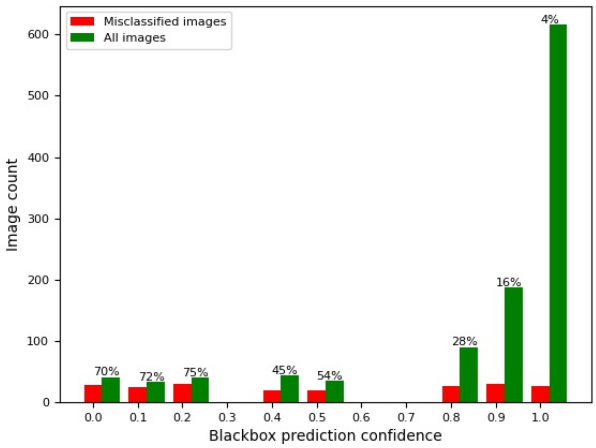
Finally, we consider a scenario where instead of using a substitute model trained by ourselves we use a fully trained face recognition model, saving ourselves from the need to perform suspicion-raising queries to determine labeling data. This parallels the idea of a 3rd party tracker which we cannot attack directly nor query since we are not aware of its existence (e.g. Clearview.ai). To counter the complete lack of information about our target (the most realistic scenario) , we chose to use a fully trained face recognition model which hopefully can generalize better and covers a larger input domain and class domain than a substitute model.

In this scenario, we pitted a fully trained RESNET50 model trained on VGGFace2 (acquired from X) to act as our substitute against a fully trained SENET50 trained on the same data set that acts as our oracle. Here, we achieved 100% success rate on VGGFace2 at LPIPS less than 0.075, and on LFW at LPIPS less than 0.03. This shows great potential, since the **only** assumption of knowledge we have on the target we are attacking is the data set on which it was trained.

This is a massive success, however the results are suspect due to the fact that both models were trained on the same data set and should perform similarly since they span the same class domain. We leave relaxing the final assumption and testing against completely different models trained on different data set/with different architectures to future work (e.g. Azure Face API, Amazon Rekognition).



(a) **Misclassification rate by perceptual budget**
Cumulative misclassification rate validation set by perceptual budget for substitute model trained on small subset of VGGFace2 training set after 3 iterations of data set augmentation



(b) **Misclassification rate by black-box prediction confidence**
Misclassification rate for substitute model trained on small subset of VGGFace2 training set after 3 iterations of data set augmentation

Fig. 6: **Substitute model success rate**
Success rates against RESNET50 oracle trained on VGGFace2 on images from VGGFace2 validation set

VI. Limitations and Comparison

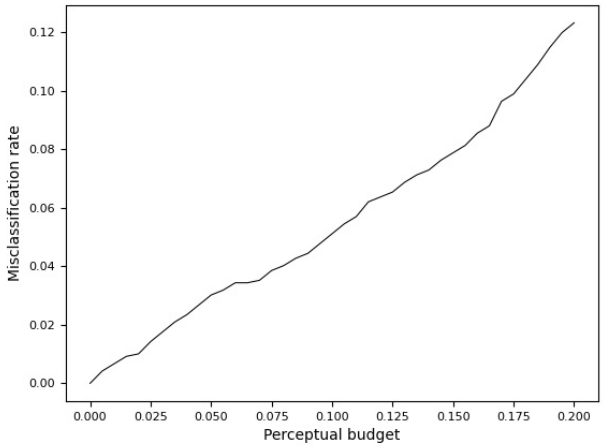
In this section we discuss the limitations of our solution, and the real-world countermeasures that can be taken by trackers to reduce the efficiency of our solution. We did not evaluate the success rate of our attack against these techniques, and leave this evaluation to future work.
Additionally, we will compare the results of our experiments with other works from the same domain, more specifically against Fawkes.

A. Limitations and Countermeasures

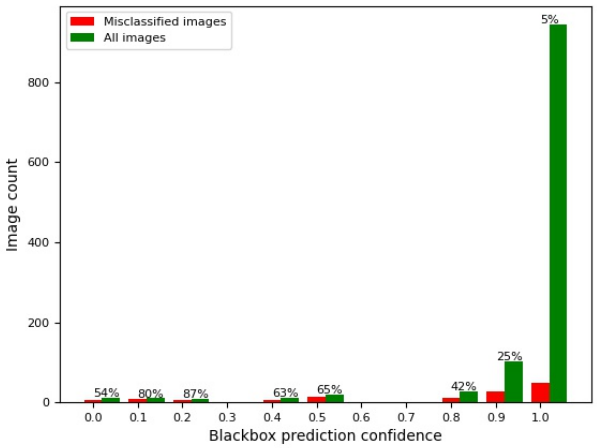
- **Image transformations** - rotation, shear, reduction to black-and-white, changes in proportion, additive noise, etc. A 3rd party tracker can use image transformations such as these to mitigate the appearance and effect of small perturbations. This also includes compression algorithms such as JPEG that are very common when uploading images. Such algorithms remove high frequencies in images (which correspond to finer details) from images.
- **Anomaly Detection** - there are many ways to detect anomalies in input for DNNs, whether the target class domain contains a user's images or not. In either case, by adding perturbations to data we increase the probability of a 3rd party tracker to detect input generated by anonyME using anomaly detection methods. However, this is not necessarily bad as detected images are usually removed from the training set, which is beneficial to the user's privacy. Moreover, identification can become unreliable due to these anomalies.
- **No Control Over the Past** - Users have been sharing images of themselves online for many years, which means masking these past images is probably less effective since this data has probably been scraped already and used for training. We can hopefully reduce the effectiveness of trackers in the long term by masking our images but it remains to be seen.
- Moreover, the deep learning field **evolves at a rapid pace** and our attack might become obsolete in the near future regardless of our current efficacy.

B. Comparison with Other Approaches

In recent work, Shan et al. proposed Fawkes - a system that helps individuals to inoculate their images against unauthorized facial recognition models [14]. In their work, they considered a system similar to ours which considers the use of fully trained face recognition models' feature extractors to produce adversarial samples against target black-box models.

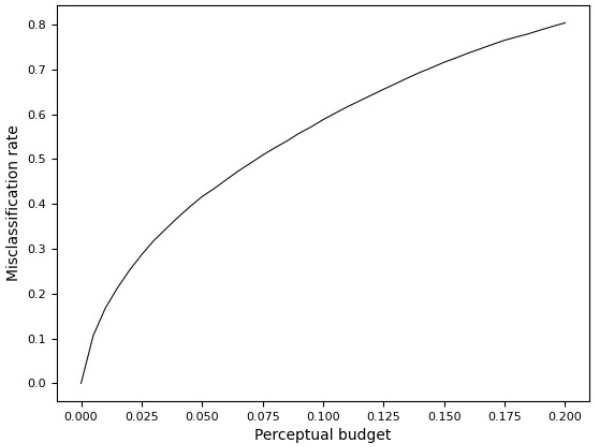


(a) **Misclassification rate by perceptual budget**
Cumulative misclassification rate by perceptual budget for substitute model trained on small subset of VGGFace2 training set after 3 iterations of data set augmentation

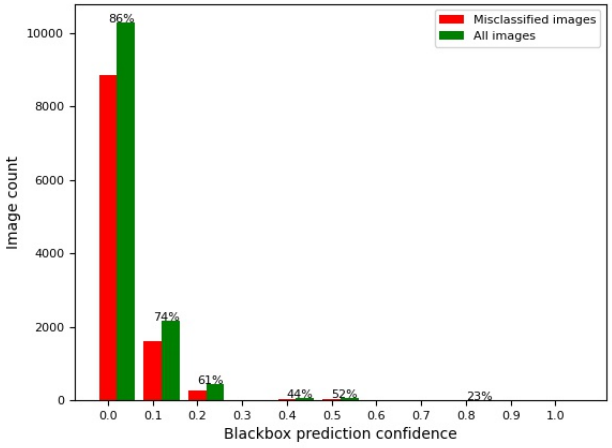


(b) **Misclassification rate by black-box prediction confidence**
Misclassification rate for substitute model trained on small subset of VGGFace2 training set after 3 iterations of data set augmentation

Fig. 7: **Substitute model success rate against SENET50 black-box**
Success rates against SENET50 oracle trained on VGGFace2 on images from VGGFace2 validation set

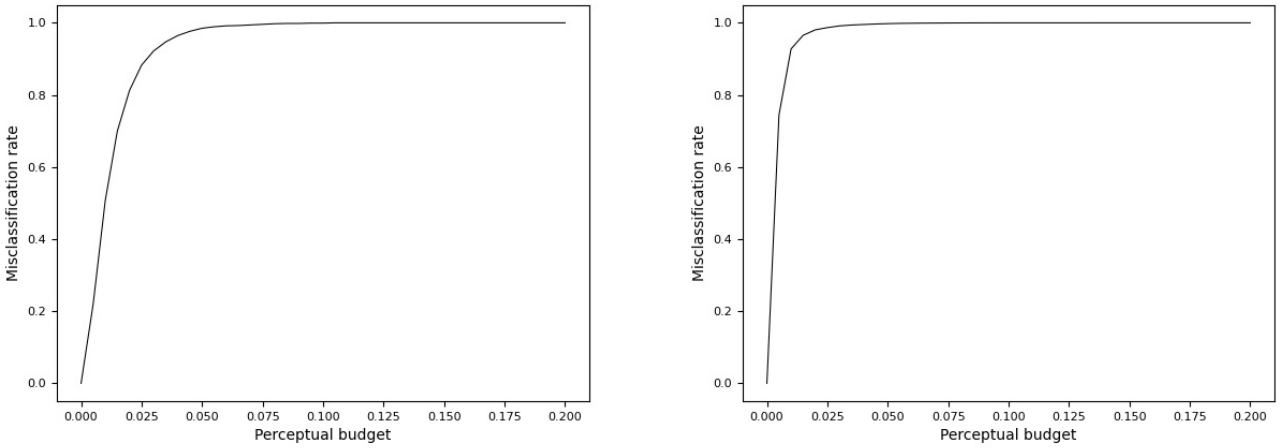


(a) **Misclassification rate by perceptual budget**
Cumulative misclassification rate by perceptual budget for substitute model trained on small subset of VGGFace2 training set after 3 iterations of data set augmentation



(b) **Misclassification rate by black-box prediction confidence**
Misclassification rate for substitute model trained on small subset of VGGFace2 training set after 3 iterations of data set augmentation

Fig. 8: **Substitute model success rate against SENET50 black-box**
Success rates against RESNET50 oracle trained on VGGFace2 on images from LFW data set



(a) **Misclassification rate by perceptual budget (VGGFace2)** (b) **Misclassification rate by perceptual budget (LFW)**

Fig. 9: **Fully trained RESNET50 substitute success rate against SENET50 black-box**
Success rates against SENET50 oracle trained on VGGFace2 on images from both VGGFace2 validation set and LFW data set

While their technique was similar to ours, they approached problem differently, and sought to prevent the **training** of face recognition models by uploading "cloaked" images, such that models trained on those images will be unable to classify images of those classes correctly (data set poisoning). In their evaluation, they've tested the effect of the percentage of cloaked images in a face recognition model's training set on their success rate, and have found that success rates increase dramatically the more cloaked images are present in the model's training set. They've reached a success rate of 20% on a model trained solely on uncloaked images, which fits well with our results (as all of our black-box models were trained on "uncloaked" images).
In order to improve success rate on trained models, they suggested the use of Sybil accounts. 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, which may be used to upload modified images to further poison a face recognition model's data set. This is out of scope for our work.
We can see from this comparison that under real-world assumptions, we've achieved similar results.

VII. Discussion and Conclusions

We presented anonyME, an accessible tool meant to protect individuals' privacy from unauthorized facial recognition systems. By observing the evaluation results of our substitute model, we can clearly see that our technique is more effective on images with low initial confidence on their black-box prediction. This is evident by the increase in the model's success rate the higher the percentage of low confidence images in the test set. While testing against VGGFace2 validation, we've reached a success rate of 7% on our lower LPIPS threshold, and 20% on our higher threshold. These results make sense as we've trained our substitute starting with a small subset of the training data set, and spanned more classes using data set augmentation. This is also supported by the conclusions reached in later work such as Fawkes [14]; we can see that a model trained mostly on uncloaked images (in our case, completely uncloaked images) yields a low protection success rate using their technique as well ($< 20\%$). Moreover, when we evaluated our attack on LFW data set as a benchmark, success rate grew to 60%, and 80%, respectively. This can be explained as LFW contains many identities not present in VGGFace2, and were therefore predicted by the black-box with low confidence.
As we've mentioned, we trained substitute models with two different architectures (RESNET50, SENET50) that yielded similar results. We learn by this that the success of the substitute is not affected greatly by the chosen architecture, as long as the model is suited for its task (i.e. face recognition). This conclusion has been supported by [12]. We used the same setting to evaluate our solution's transferability, but reached inconclusive results.

We believe that our success rate on high confidence images can be improved by further training. To support this, we have used a fully trained RESNET50 model as a substitute and attacked our SENET50 black-box model, and achieved perfect success. Therefore we deduce that with further training of our substitute model, these results could be improved upon.

VIII. Future Work

A. Real World Evaluation

In order to further evaluate our solution, we could test it against large-scale face recognition models available in the wild, e.g. Microsoft Azure Face API [24], Amazon Rekognition Face Verification [25], Face++ Face Search API [26], etc. (as seen in [14]). We ran out of free cloud credits and were not able to complete training, and so this is left for future work.

B. Optimizations

We can optimize our technique further by experimenting with different methods of attacking face recognition models, such as [27], or [28]. We can also experiment with different data set augmentation methods in order to further improve our substitute training algorithm. We could also further develop the mobile app not covered here to increase our solution's accessibility to end-users.

C. User Study

We can conduct a user study showcasing input images alongside their adversarial counterparts to users in order to further and better evaluate our adversarial samples' visually, and perceptual budget.

D. Applicability to Other Problem Domains

The techniques presented here could be used to create black-box adversarial samples in other problem domains e.g. voice recognition, thus further increasing user privacy online.

References

[1] Zhang, R., Isola, P., Efros, A., Shechtman, E., Wang, O., "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". arXiv:1801.03924 (2018).

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

[3] Satariano, A. "Police use of facial recognition is accepted by British court". The New York Times (September 2019).

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

[5] Schroff, F., Kalenichenko, D., Philbin, J.. "FaceNet: A Unified Embedding for Face Recognition and Clustering". arXiv:1503.03832 (2015).

[6] Yang, Taigman M., Ranzato, M., and Wolf, L.. "Web-scale training for face identification". In CVPR, pages 2746–2754,2015.

[7] Li, T., and Lin, L. "Anonymousnet: Natural face deidentification with measurable privacy". In Proc. of CVPR (2019).

[8] Sun, Q. et al. " A hybrid model for identity obfuscation by face replacement". In Proc. of ECCV (2018).

[9] Wu, Y., Yang, F., and Ling, H. "Privacy-Protective-GAN for face de-identification". arXiv:1806.08906 (2018).. In Proc. of ECCV (2018).

[10] Thys, S., Van Ranst, W., and Goedmé, T. "Fooling automated surveillance cameras: adversarial patches to attack person detection". In Proc. of CVPR (workshop) (2019).

[11] 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).

[12] Papernot N. et al. "Practical Black-Box Attacks against Machine Learning". arXiv:1602.02697 (2017).

[13] Goodfellow, I. J., Shlens J., and Szegedy C. "Explaining and Harnessing Adversarial Examples". arXiv:1412.6572 (2015).

[14] Shan, S., Wenger, E., Zhang, J., Li, H., Zheng, H., and Zhao, B. Y.. In Proc. of USENIX Security Symposium (2020).

[15] Kurakin, A., Goodfellow, I., and Bengio. S.. "Adversarial examples in the physical world". arXiv:1607.02533 (2017).

[16] He, K., Zhang, X., Ren, S., and Sun, J.. "Deep Residual Learning for Image Recognition". arXiv:1512.03385 (2015).

[17] Hu, J., Shen, L., Albanie, S., Sun, G., and Wu E.. "Squeeze-and-Excitation Networks". arXiv:1709.01507 (2019).

[18] Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A.. "VGGFace2: A dataset for recognising face across pose and age". In Proc. of International Conference on Automatic Face and Gesture Recognition (2018).

[19] https://github.com/rcmalli/keras-vggface. keras-vggface.

[20] Zhang, K., Zhang, Z., Li, Z., Qiao, Y.. "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks". arXiv:1604.02878 (2016).

[21] https://github.com/ipazc/mtcnn. Python MTCNN implementation using Keras.

[22] https://github.com/richzhang/PerceptualSimilarity. LPIPS metric git repo, implemented using PyTorch.

[23] Huang, G. B., Mattar, M., Berg, T., Learned-Miller, E.. "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments", In Proc. of Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition (Oct. 2008).

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

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

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

[27] Papernot N. et Al. "The Limitations of Deep Learning in Adversarial Settings". arXiv:1511.07528 (2015).

[28] Bose, A. J., Aarabi, P.. "Adversarial Attacks on Face Detectors using Neural Net based Constrained Optimization". arXiv:1805.12302 (2018)