Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition

Piotr Mazurek
Machine Learning in the Context of IT Security
FU Berlin Summer Semester 2019

September 15, 2019

Abstract - In recent years, machine learning algorithms enabled countless innovations, such as recommendation systems, intelligent voice assistants or self-driving cars. A large part of these innovations was possible thanks to Deep Learning, which has been experiencing its renaissance since 2009 [3], when ImageNet Classification with DCNN paper [4] was first time introduced. Deep Learning methods are extremely useful in biometric-authentication methods such as fingerprint[5] or face recognition[6].

The wide usage of machine learning, particularly its applications where physical security (e.g. self-driving cars) or safety (biometric-authentication) is at risk, makes it very important to understand how ML algorithms can be attacked, in order to make these algorithms attack-proof.

In this paper we will sum up ideas presented in the 'Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition' paper [2]. We will discuss theory behind deep learning methods, details of face recognition methods and methods of attack on such systems.

Attacks described in original paper[2] focus on biometric systems - especially face recognition. Discussed attack methods have to meet some assumptions like they have to be physically realizable and inconspicuous. Attack generation should also be universal - it shall work regardless of the gender, age or skin color of the attacker.

Authors came up with idea of generating an eye-glass frames, which when worn by an attacker allow him/her to avoid being recognized as himself/herself or to impersonate another person from biometric system database.

1 Introduction

Machine learning (ML) and especially its sub-domain Deep Learning (DL) are technologies that in recent years deeply have changed the world [1]. Except pure academic usage DL and ML solutions are widely used in the industry and what is especially important for us, in the biometric systems. Unfortunately how recent studies shown DL methods are vulnerable to adversarial attacks [8,9]. This means that these machine learning models incorrectly classify examples that are only slightly different from correctly classified examples from data distribution. Since ML and DL algorithms are widely used in security areas (such as biometrics) it is extremely important to understand how such attacks work, why they work and how to defend against them.

Authors of 'Real and Stealthy Attacks on State-of-the-Art Face Recognition' [2] paper came up with an idea of systematic method to automatically generate attacks on Face Recognition systems using a printed pair of eyeglass frames. When the attacker is wearing such eyeglass frames he/she is not classified as himself/herself by a Face Recognition System. As a benchmark Face Recognition System (FRS) authors have used 'Deep-Face-Recognition' [6] - state of the art algorithm for Face Recognition problems in 2016 (when Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition was first published). They show how to generate such eye-frames, what problems may occur during generation or printing and how to avoid such issues.

2 Deep Learning Theory

2.1 What is deep learning?

Deep Learning is a subclass of machine learning, where instead of using space transformations it uses multiple non-linear transformations to progressively extract higher level features from input data. This set of non-linear transformations is called Neural Network. The proper transformations are found in a process called 'training' which is just a Machine Learning environment expression for optimization using Gradient Descent.

In image recognition problems Neural Network (NN) takes pixel values as input data, apply non-linear transformations, and as a result return a vector of probability distribution over a set of classes.

2.2 How is deep learning used in Face Recognition Systems?

The face recognition algorithm used in 'Deep Face Recognition' [6] works very similar to the classic image recognition algorithm. As an input data it uses a picture of a face, it propagates it through a NN and as a result it returns a vector of probabilities. (See Figure 1). Then a Face Recognition System (FRS) finds a class with the highest probability value and that is the predicted persons identity.

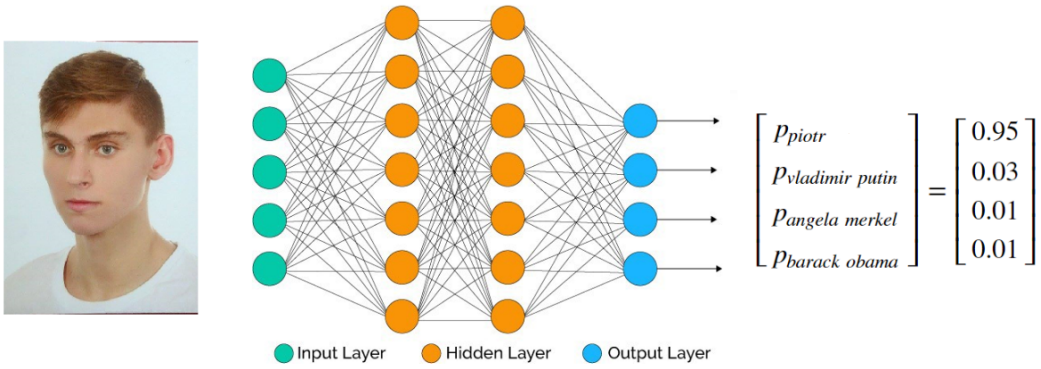

Figure 1: FRS returns distribution of probabilities over set of faces

3 Deep Neural Network Architecture

Deep Neural Networks (DNNs) consists of connected layers of different types. These types of layers will be discussed in this section

3.1 Fully Connected layer

A Fully Connected (FC) layer is a layer where every neuron from the previous layer is connected to every neuron in the next layer (See Figure 1). It is the basic layer in most of Deep Learning Systems, however due to the number of operations it has a big complexity (matrix multiplication is around $O(n^3)$ in big O notation). The face images used in this study were images of size 224×224 with 3 RGB channels. If in the first hidden layer we would have 1000 nodes it would be $224 \times 224 \times 1000 = 50176000$ parameters to optimize (it is only the first layer). For this reason, we must somehow reduce the number of parameters, for example by using convolution layers.

3.2 Convolution layer

Convolution Layer is an idea based on edge detection. Edge detection was one of early techniques used to solve computer vision problems, as the name suggest it allows us to detect edges in a given image (See Figure 3). The Edge detection algorithm is performed using a matrix called a mask and an operation called convolution. It takes matrix of size m (size of a mask) from input image, multiply it element-wise by values in mask, sums all values and as a result returns edges detected in input image (See Figure 2). The general expression of a convolution is:

$$g(x,y) = (\omega * f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t) \tag{1}$$

Where $g(x,y)$ is the filtered image, $f(x,y)$ is the original image, ω is the filter kernel.

Every element of the filter kernel is considered by $-a \leqslant s \leqslant a$ and $-b \leqslant t \leqslant b$ [1]

Using hand-engineered filters values we can detect horizontal or vertical edges (See Figure 4) but what if we want to detect some more sophisticated features like cat edges (See Figure 5)? This is where deep learning comes in: Instead of using hand-engineered filter values we can use a Gradient Descent Algorithm to find the right values for mask matrix.

What is interesting at this point is the fact that regardless of the image size, we have same number of parameters to train. No matter if our image is 20×20 pixels or 1000×1000 we need to train the same number of parameters - the number of values in filter mask. It is one of main reasons why CNNs are so popular in real life computer vision problems.

The role of convolution layer is to extract crucial information (edges) from image by using custom filter masks. The side benefit of detecting edges can be reduction of image size (it is not always the case, it depends on used padding see Sec. 3.5 Padding). Outcome of Convolution Layer (the one consisting crucial information about edges) can be then propagated to Fully Connected Layers to perform classification.

$$\underbrace{\begin{bmatrix} 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \\ 0 & 0 & 0 & 10 & 10 & 10 \end{bmatrix}}_{img} * \underbrace{\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}}_{mask} = \underbrace{\begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}}_{detected\,edge}$$

Figure 2: Example convolution operation

Figure 3: Detected edges after applying Sobel filter[2]

[1] Equation from Wikipedia `https://en.wikipedia.org/wiki/Kernel_(image_processing)`

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}_0 \quad \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}_{45} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & 1 \end{bmatrix}_{90}$$

Figure 4: Example edge detection Sobel filter for angles: 0°, 45° and 90°

3.3 Convolution operation on volume

Black and white images contain only single color channel, but in color images there are more. E.g. color RGB Image contains 3 channels (Red, Green, Blue). In that case each filter need to have 3 "sub filters", one for each channel. We apply convolution operation to each channel and then sum up the result (See Figure 6).

3.4 Multiple filters

To detect multiple types of features in previous layer we should use multiple filters. E.g. one filter will detect vertical edges and one will detect horizontal (See Figure 7).

3.5 Padding

To avoid underrepresentation of edge pixels it is worth to add one layer of extra pixels around original image (See Figure 8).

3.6 Stride

Stride determines the number of cells that the filter moves in the input to calculate cell in output (See Figure 9).

[2]Cat edges from `http://mcogswell.io/blog/why_cat_2/`
[3]Example detected egdes image from `https://en.wikipedia.org/wiki/Sobel_operator`

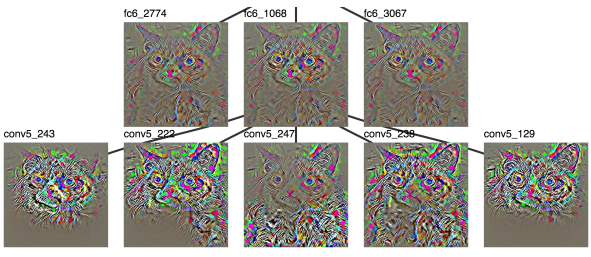

Figure 5: Cat edges recognized by individual CNN layers[3]

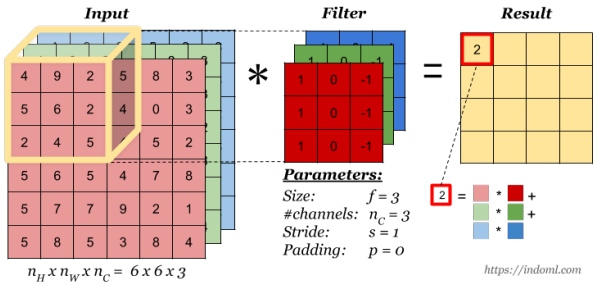

Figure 6: Convolution on volume example [11]

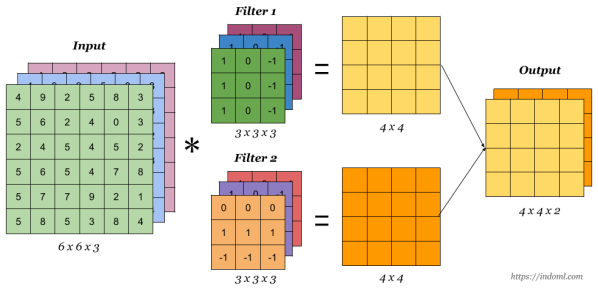

Figure 7: Convolution with multiple filters example[11]

3.7 Pooling layer

Another type of layer used in CNNs is Pooling Layer. There are two types of Pooling Layers: Avg. Pooling and Max Pooling. Pooling layers significantly reduce size of an image. The intuition behind pooling is *take most important information* for Max Pooling and *take average of all information* for Avg Pooling. (See Figure 10).

4 Deep Face Recognition

Deep Face Recognition [6] was a state of the art FRS in 2016 and was also a base system that the authors of a paper decided to attack. It was trained on the set of 2622 actors

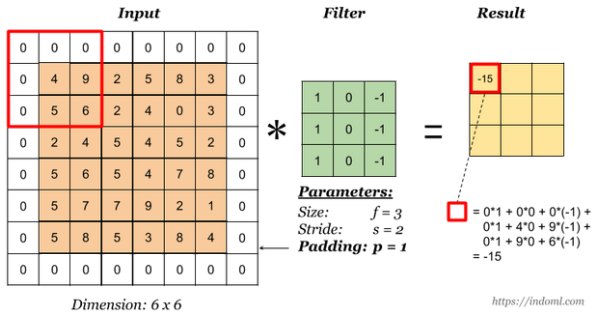

Figure 8: Example of 1 pixel padding [11]

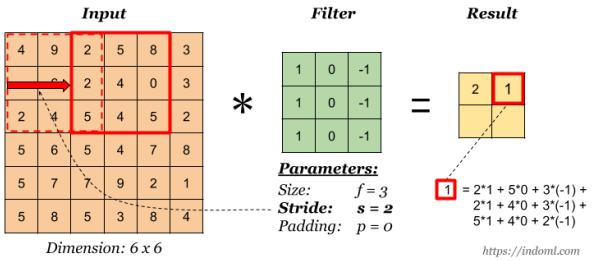

Figure 9: Example output for stride equal 2[11]

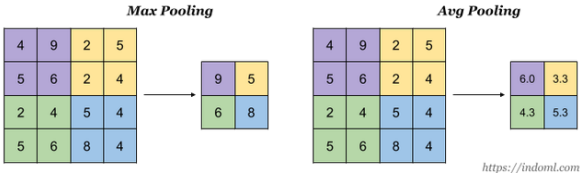

Figure 10: Outputs after pooling operations [11]

and actresses, with roughly 1000 images per celebrity. In total of about 2.6M of train images. Although it have used significantly less training images then its competitors (for comparison Google has used 200M to train their FRS, Facebook has used 4.4M) their NN still achieves 98.95% accuracy (which is comparable to Google and Facebook solutions)[6].

The architecture of Deep Face Recognition was inspired by [12] and consisted of long sequence of convolutional layers. All the convolution layers are followed by a one or more non-linearities such as ReLU activation function or max poling layer.

In order to test attack accuracy Sharif et al. used 3 different Neural Networks:

DNN A - original Face Recognition System used in the Deep Face Recognition paper. 2622 classes, only actors no researchers in dataset.

DNN B - 10 classes. 5 Researchers and 5 actors from PubFig dataset[7]. DNN was trained with transfer learning technique, only last layer was changed (from 2622 classes to 10) and retrained.

DNN C - 143 classes. 140 actors from PubFig dataset and 3 researchers. DNN was trained with transfer learning technique, only last layer was changed (from 2622 classes to 143) and retrained.

5 What does it mean to attack FRS?

As we already know a FRS takes a face as an input image, propagates it throw a NN and as a result returns probability distribution over set of personalities. When we talk about attack on FRS we mean we want either face A be recognized as face B (it is called Impersonation) or to face be A be recognized as any face but A (it is called Dodging). In the paper authors have focused on so-called adversarial attacks. In such attack assailant attacks an input of face recognition system by adding some (imperceptible to man)

noise to this input (See Figure 12). Then this affected input (image) is transferred to the biometric system and incorrectly recognized.

5.1 Impersonation

In impersonation attack, an attacker seeks to be recognized as a specific different, exact user from FRS database. E.g. the adversary wants to log into another user android phone which can be unlocked by FRS (See Figure 11)

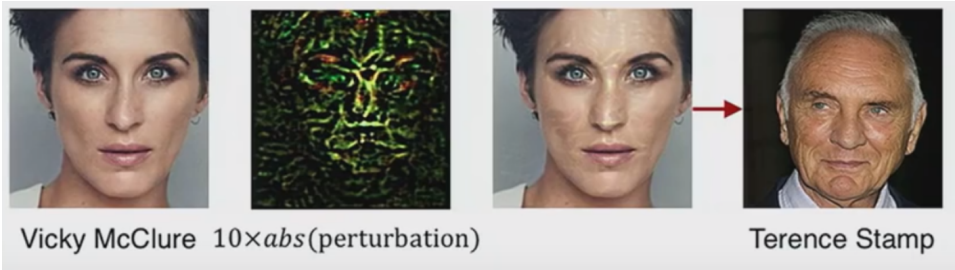

Figure 11: Actress Vicky McClure recognized as Terence Stamp by adding noise to her face image[4]

5.2 Dodging

In dogging attacks, an attacker tries to make his / her face misidentified as another face. Attacker don't care as whom he/she will be classified, the only concern is not to be classified as himself/herself. This can be particularly useful for democratic activists and freedom fighters who live in authoritarian and non-democratic countries. Such activists can use this type of attack to avoid persecution by oppressive governments. From a technical point of view, this category of attack is interesting because causing misclassification should (theoretically) be easier to do with minimal face modifications, compared to incorrectly classifying a face as the specific target (impersonation)

6 Attack on Face Recognition System

6.1 Initial assumptions

Our main assumption is that the adversary attacks the system after the system was trained. Attacker can't in any way alter training data, inject mislabeled data or change neural network architecture. On the contrary, an attacker can only change FRS inputs (camera images). Changes to the input should be based on his / her knowledge of the classification model, potentially derived by examining the entire model or by observing its outputs for some inputs. We assume attacker know both the feature space (features

[4]Image from authors presentation at CCS 2016 `https://youtu.be/6Xh1vuwnVhU`

for image classification, especially face classification are considered to be publicly known [13]) and the internals (architecture and parameters) of the system being attacked (the system is so called *white-box*)

Physical Realizability authors assumed that attack should be physically independent for the purpose of fooling facial biometric system. They assumed that attacker can change only his/her appearance, but can't change a background. Considered attacks should also be robust to changes in image conditions - light changes, attacker position changes, standing further/closer to the camera shouldn't affect effectiveness of an attack.

Inconspicuousness unlike most similar papers authors focused on attacks that do not arouse suspicion. Neither attacked system nor personnel supervising the system should not notice that the system is being attacked. By inconspicuousness they mean attacks that doesn't emit light or cover the whole face of the attacker (See Figure 12)

They believe that this kind of attacks is important due to 2 key reasons. First such attacks can be particularly harmful because they will be immune to at least a cursory investigation and second attackers achieve plausible deniability. Attacker will always be able to claim that they didn't commit any attack and failure of machine learning algorithm is caused by imperfection of such systems. Designers of biometric systems should know how to conduct such attacks to design systems that are resistant to attacks.

Figure 12: Attacks that arouse suspicion[5]

6.2 Generating adversarial noise

In recent years Deep Neural Network models consisting of convolutional layers achieved state of the art performance in image recognition task [4]. Although their expressiveness is the cause of success, they also learn incomprehensible properties that are counter-intuitive. Such incomprehensible properties were described by Szegedy et al. in *Intriguing properties of neural networks* paper in 2014 [9]. Szegedy et al. surprised the machine learning community by finding that DNNs can be misled by slightly distorted input data. Szegedy et al. shown that it is possible to cause the network to categorize the image incorrectly using some barely noticeable disturbance that can be found by maximizing network prediction error. Under white-box scenario such disturbance can be found systemically in a similar way we find weights that minimizes prediction error - by using Gradient Descent algorithm.

We can formalize the problem of finding perturbation r, as an optimization problem with 2 goals. First goal is misclassification, for a given x find perturbation r such that distance between DNN output $f(x+r)$ and some target class h_t will be minimised. Second goal is imperceptibility of perturbation r, we achieve it by minimizing the norm of r. In other words, find a smallest perturbation r that will cause specific misclassification c_t. . The objective of finding the perturbation is modeled by:

$$\text{argmin}_r \left(|f(x+r) - h_t| + \kappa |r| \right) \tag{2}$$

Where classification function f (in our case DNN) produces a probability distribution over N classes (represented by a vector of size $Nx1$) where each element is $[0, 1]$. h_t is the one hot vector of class c_t. $|\cdot|$ is norm function (e.g Norm 2). κ a constant that can be tuned to balance the misclassification and imperceptibility objectives. If $f(\cdot)$ and $|\cdot|$ are differentiable, this optimization can be solved via simple optimization method like Gradient Descent. In our case $f(\cdot)$ is DNN, so we can be sure it is differentiable (without that property it would not be possible to train the neural network using back propagation). The usually used norm $|\cdot|$ is $Norm_2^2$, which is also differentiable. Optimal parameter r can by fund using one of the modern machine learning frameworks like Tensorflow or PyTorch. Example perturbation r (amplified by 10 times to better visualize the change) and original image plus perturbation $x + r$ on Figure 12.

6.3 Impersonation and Dogging softmaxloss score

In order to measure the correctness of classification authors used *softmaxloss* score. For a given input x of class c_x (which one hot representation is h_{c_x}) that is classified as $f(x)$ (a vector of probabilities size $Nx1$), *softmaxloss* is defined as: Softmaxloss

$$\textit{softmaxloss}\left(f(x), c_x\right) = -\log\left(\frac{e^{\langle h_{c_x}, f(x)\rangle}}{\sum_{c=1}^{N} e^{\langle h_c, f(x)\rangle}}\right) \tag{3}$$

Where $\langle \cdot, \cdot \rangle$ denotes inner product between two vectors. N is number of classes, h_c is one hot representation of class c. *Softmaxloss* is lower when classification $f(x)$ is correct, and higher when classification $f(x)$ is incorrect. *Softmaxloss* can be used to measure effectiveness of Impersonation or Dodging for a given perturbation r.

Impersonation

When an attacker plans to impersonate a concrete target t, he / she needs to find such perturbation r that will maximize probability of class c_t for a classifier $f(x + r)$. We can define it as an optimization problem:

$$\text{argmin}_r \, \textit{softmaxloss}\left(f(x+r), c_t\right) \tag{4}$$

In other words, find perturbation r such that will minimize distance between prediction $f(x + r)$ and target class c_t

Dodging

In Dodging attacks, (in contrast to Impersonation) instead of impersonating concrete target t attacker only goal is to not be recognised as himself / herself. This can be obtained by finding such perturbation r that will minimize probability of recognising input x as class c_x, or in other words by maximizing the value of $softmaxloss(f(x+r), c_x)$. We can define it as an optimization problem:

$$\text{argmin}_r \left(-\text{softmaxloss}\left(f(x+r), c_x\right)\right) \tag{5}$$

6.4 Generating eye-glass frames

As we shown in previous paragraph it is possible to find such perturbation r that will cause specific misclassification. Unfortunately using that method we change both face and background (See Figure 12). In initial assumptions we assumed that *attacker can change only his/her appearance, but can't change a background* (**Physical Realizability**), so we need to do something about it.

The simple and efficient solution can be limiting perturbation to only the pixels that cover the face. It can be obtained by first performing image segmentation to find a face in an image and then limiting changes only to those pixels (See Figure 13).

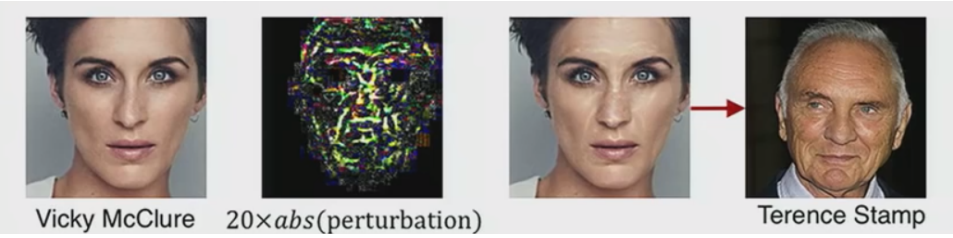

Figure 13: Changes applied only to face not to background[5]

However, this approach does not solve all problems. First of all, founded face perturbations are small (in order to see them they have to be amplified 20 times See Figure 13) so it is very complicated to realize such perturbation in a real world. Second of all, cameras used in FRS have some physical limitations. Camera sampling errors are bigger than needed perturbations so carrying out such an attack would not be physically possible (attack would actually not be detected by a camera).

For this reason, instead of changing the whole face, the authors of the paper came up with the idea of using facial accessories, specifically the eyeglass frames. The eyeglass frames are both physically realizable via 3d or even 2d printing technologies and inconspicuous - they don't arouse that much suspicion. The eyeglass frames perturbation can be generated in a similar way as face perturbation. First we perform image segmentation in order to find eyes in the image and then limiting perturbation r only to the pixels around eyes in the shape of frames (See Figure 14).

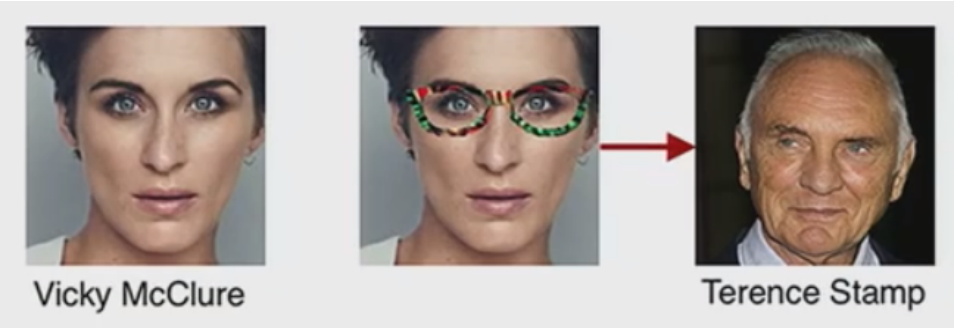

Figure 14: Eyeglass attack[5]

7 Physical implementation

Successful attack in digital environment does not guarantee same success in the physical world. In contrast to the digital environment in real world we have to print eyeglass frames using some kind of 2d or 3d printer (in this case 2d) and then conduct an attack on a camera used by FRS. None printer is able to print every possible color. Every camera has a sampling error and is unable to capture colors 1:1. Because of that we have to apply slightly changes to the perturbation finding method.

7.1 Robust Perturbations

Because of changing condition such as image background, light conditions, attacker posture or his/her facial expression it's not fairly possible that two attacker's face images will be exactly the same. In initial assumption we assumed that *Considered attacks should also be robust to changes in image conditions - light changes, attacker position changes, standing further/closer to camera shouldn't't affect effectiveness of an attack.* That's why authors assumed that in order to successfully carry out attacks it is necessary to find perturbation r such that it will be independent of exact image conditions. In other words we want to avoid some kind of *overfitted* perturbation, but instead to find perturbation that will generalize beyond a single face image.

This was achieved by gathering a set of photos of the attacker X, taken in different conditions (changing background, lighting, facial expression, distance from the camera, head angle, etc.) and use that set X to find a single perturbation r that optimizes its objective for every input $x \in X$. We can formalize this as as optimization problem for both Impersonation (6) and Dodging (7):

$$\text{argmin}_r \sum_{x \in X} \text{softmaxloss}(f(x+r), c_t) \tag{6}$$

$$\text{argmin}_r \sum_{x \in X} \text{-softmaxloss}(f(x+r), c_x) \tag{7}$$

7.2 Smooth transitions

Natural images tend to contain smooth and regular patches, which are separated by a few edges. However generated perturbation eyeglass frames generated by minimizing *softmaxloss* contain lot of edges and aren't very smooth. It might be a problem for cameras, because of sampling noise, extreme differences between neighboring pixels are unlikely to be accurately captured by camera. As a result non-smooth eyeglass frames may lead to not physically realizable attacks (which was paper initial assumption, see 6.1).

To achieve smooth perturbation we have to update the optimization function. Except minimizing *softmaxloss* we have to also minimize *total variation* (TV). For a perturbation r, $TV(r)$ is defined as:

$$TV(r) = \sum_{i,j} \left((r_{i,j} - r_{i+1,j})^2 + (r_{i,j} - r_{i,j+1})^2 \right)^{\frac{1}{2}} \tag{8}$$

Where $r_{i,j}$ is an eyeglass frames pixel value at coordinates (i, j). $TV(r)$ is low when the neighboring pixels have similar values (perturbation patch is smooth and regular) and high otherwise. A result of minimizing $TV(r)$ is smooth perturbation.

7.3 Printability

For printing an eyeglass frames authors have used a simple ink-jet printer - (Epson XP-83). It prints 2d eyeglass frames, but as any printer in the world it has some physical limitations. It can reproduce only a subset of the $[0, 1]^3$ RGB color space not all possible colors. That's why in order to conduct successful attacks we need to limit perturbation to only colors reproducible by the printer. To do so authors defined *non-printability score* (NPS). NPS have high value for pixel of unreproducible color and low otherwise. Let $P \subset [0, 1]^3$ be the set of printable RGB triplets. We define the NPS of a pixel $\hat{p}$ as:

$$NPS(\hat{p}) = \prod_{p \in P} |\hat{p} - p| \tag{9}$$

To measure total NPS for a perturbation r we sum NPS for each pixel $\hat{p}$ from a perturbation r. Instead of using all printable RGB triplets authors decided to pick the 30 triplets with a minimal variance in distances from the complete set of printable RGB triplets and use only them in the set P used in definition of NPS. Authors claim that this optimization with using 30 *centroids* worked great for them and lead to big improvements in physical realizability of an attack.

7.4 Putting Everything Together

Now we can to put everything together and model the new objective that will consist of *misclassification smoothness* and *printability*. This new objective can be modeled as:

$$\underset{r}{\operatorname{argmin}} \left(\sum_{x \in X} \text{softmaxloss}\left(f(x+r), c_t\right) \right) + \kappa_1 \cdot \text{TV}(r) + \kappa_2 \cdot \text{NPS}(r) \tag{10}$$

Where $\sum_{x \in X} \text{softmaxloss}\left(x+r, c_t\right)$ is a *misclassification* part (it is version for impersonation, for dodging it would be $\operatorname{argmin}_r \left(-\text{softmaxloss}\left(f(x+r), c_x\right)\right)$), κ_1 and κ_2 are hyperparameters to balance *smoothness* (TV) and *printability* (NPS), r is the perturbation we want to find and X is a set of images of an attacker's face,

8 Conducted experiments and results

Paper's authors conducted attacks in the digital domain (Sec 8.1) and in physical world (Sec 8.2).

8.1 Digital-Environment Experiments

Digital environment attacks assumes that attacker has access to FRS input system and can manipulate input images on a pixel level, so we don't need to care about physical-realizability (See Sec. 7). Our only concern is minimizing *softmaxloss* (See Sec. 6.3). Intuitively this kind of attacks should be easier to conduct.

Sharif et. al ran 8 different attacks (See Figure 15) on the DNNs described in Sec. 5. The experiments differed in the purpose of the attack, the type of perturbation (the whole face or just eyeglass frames) and the type of attacked DNN (DNN_A DNN_B or DNN_C, see Sec. 5)

The success rate was measured as a percentage (success rate) of attacks in which attacker was recognized as target t (for impersonation attacks) or not recognized as himself/herself. For each attack Sharif et. al used 10 or 20 different attackers, 3 images for each attacker and reported mean success rate for those images.

The obtained results were impressive. In experiments 1 and 2 (in which attacker could change any pixel at the face) conducted attacks had 100% success rate. However changes in pixel values were very small, imperceptible to humans and not very practical. Perturbations are smaller then camera sampling error so it would be not possible to use such image to conduct successful attack on FRS in the physical world.

Experiments 3-8 used perturbation in a eyeglass frames shape, here results were also quite impressive. For all of dodging experiments (3-5) success rate was 100%. The only attack that have slightly worst performance result was impersonation attack on DNN_A. Here success rate was below 92%, which might be caused by big number of classes. The obtained results suggest that impersonation attacks are more difficult to perform then dodging attacks.

Experiment #	*Area perturbed*	*Goal*	*Model*	*# Attackers*	*Success rate*
1	Entire face	Dodging	DNN_A	20	100.00%
2	Entire face	Impersonation	DNN_A	20	100.00%
3	Eyeglass frames	Dodging	DNN_A	20	100.00%
4	Eyeglass frames	Dodging	DNN_B	10	100.00%
5	Eyeglass frames	Dodging	DNN_C	20	100.00%
6	Eyeglass frames	Impersonation	DNN_A	20	91.67%
7	Eyeglass frames	Impersonation	DNN_B	10	100.00%
8	Eyeglass frames	Impersonation	DNN_C	20	100.00%

Figure 15: Attack results in digital environment [2]

8.2 Physical-Realizability Experiments

As mentioned in Sec. 7 *Successful attack in digital environment does not guarantee same success in a physical world.* In physical wold we have to deal with camera sampling errors or printing limitations and as a result we have to modify the objective which we use to find perturbation (See equation 10).

For physical-realizability Experiments Sharif et. al ran 6 different attacks (See Figure 16) on the DNN_B and DNN_C. To take pictures authors used Canon T4i camera, eyeglass frames were printed with an Epson XP-830 printer on Epson Glossy photo paper. κ_1 and κ_2 hyper-parameters were set to 0.15 and 0.25 respectively. As as attacker authors have used themselves, the targets to attack were randomly selected.

For each subject they collected about 30-50 photos in each of five sessions. First session was used to collect set X of images of attackers not wearing any eyeglass frames. This set X was later used for generating perturbation eyeglass frames, used later in the attacks. Session 2 and 3 were used to conduct dodging attacks on DNN_B and DNN_C, Sessions 4 and 5 were used to conduct impersonation attacks on DNN_B and DNN_C.

All 3 attacker succeed at dodging using perturbation eyeglass frames. On DNN_C which consisted of 143 classes all attackers achieved 100% success rate. On DNN_B results were also quite good but minor problems occurred (e.g. S_C managed to achieve only 80% success rate).

For impersonation attacks, each attacker get 2 targets one for DNN_B and second one for DNN_C. Some problems occurred when S_B(one of the authors - Sruti Bhagavatula), a 24-year-old South Asian female, was assigned to impersonate Colin Powell, a 79-year-old white male and when S_C(one of the authors - Lujo Bauer), a 24-year-old Middle Eastern male tried to impersonate Clive Owen, a 51-year-old male, but in general each of the attackers succeeded at least at some of the attacks.

In practice (to fine-tune security) FRS systems have some *minimum probability threshold* that must exceed the maximum probability for $f(x)$ classifier. Usually FRS creators try to balance security and usability. Sharif et. al set threshold value to 0.85 which significantly decreased *success rate* of attacks while true accepted remained quite high (over 92%)

DNN	Subject (attacker) info		Dodging results		Impersonation results			
	Subject	*Identity*	*SR*	*E*(*p*(correct-class))	*Target*	*SR*	*SRT*	*E*(*p*(target))
DNN_B	S_A	3rd author	100.00%	0.01	Milla Jovovich	87.87%	48.48%	0.78
	S_B	2nd author	97.22%	0.03	S_C	88.00%	75.00%	0.75
	S_C	1st author	80.00%	0.35	Clive Owen	16.13%	0.00%	0.33
DNN_C	S_A	3rd author	100.00%	0.03	John Malkovich	100.00%	100.00%	0.99
	S_B	2nd author	100.00%	<0.01	Colin Powell	16.22%	0.00%	0.08
	S_C	1st author	100.00%	<0.01	Carson Daly	100.00%	100.00%	0.90

Figure 16: Physical realizability attacks results, SR stands for *success rare* STR stands for *success rate when minimum threshold is used*, S_A is Mahmood Sharif, S_B is Sruti Bhagavatula S_C is Lujo Bauer [2]

9 Conclusion

Nowadays we are surrounded by technology. We quickly adapt technological innovations and start to overtrust it. We forget it may fail either because of unpredictable use case or because of malicious attack. Sharif et al. shown that even solutions believed to be state-of-the-art, may be successfully attacked. Their research proved that cheap solutions that nearly anyone can prepare at home may be threat to sophisticated public security systems. They shown that eyeglass frames printed at simple home printer may obtain really good results at committing biometric frauds. Because of these threats, the machine-learning community should continue to work to ensure security of biometric systems. Hopefully further works of researchers will focus on creating systems robust to attacks for safer future for all of us.

For me personally, it was the first meeting with the combination of machine learning and cybersecurity, which was a very interesting experience. After reading this article, I will definitely be less trusting about biometric security and will be aware that they are not unbreakable.

References

[1] Neha Soni, Enakshi Khula Sharma, Narotam Singh, Amita Kapoor. Impact of Artificial Intelligence on Businesses: from Research, Innovation, Market Deployment to Future Shifts in Business Models

[2] Sharif, Bhagavatula, Bauer, Reiter Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition

[3] Hardy, https://www.nytimes.com/2016/07/19/technology/reasons-to-believe-the-ai-boom-is-real.html

[4] Krizhevsky, Sutskeve, Hinton, ImageNet Classification with Deep Convolutional Neural Networks

[5] Awad Machine Learning Techniques for Fingerprint Identification: A Short Review

[6] Parkhi, Vedaldi, Zisserman Deep Face Recognition

[7] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In Proc. ICCV, 2009

[8] Goodfellow, Shlens, Szegedy, Explaining and Harnessing Adversarial Examples

[9] Szegedy et al. Intriguing properties of neural networks

[10] Megvii Inc. Face++. http://www.faceplusplus.com/.

[11] Convolution Operation on Volume, Multiple Filters, Stride, Padding, Pooling, LeNet 5 `https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/`

[12] Szegedy et al. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations, 2015.

[13] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In Proc. ACM CCS, 2015.