Technical Report- Face Cloaking Workshop

BORIS EVDEEV, Tel Aviv University, Israel
IDAN RAHAMIM, Tel Aviv University, Israel
SHIRAN SHAHARABANI, Tel Aviv University, Israel

Abstract: Deep learning models have made face recognition systems very accurate. Today, large tech companies, especially those of social networks, use facial recognition systems that pose significant privacy threats to their users. These companies can associate users with their social interactions, including their active and past locations, every day's activities, and some preferences which the user may not want to share.

In this report, we compare two solutions to these problems (Faceoff [1], Ulixes [2]), and analyze them on various situations. Firstly, we describe the problem and the current theoretical solutions that are based on state-of-the-art adversarial machine learning techniques. In addition, we present our implementation and improvements to those algorithms, discuss their reproducibility and define evaluation metrics. Then, while researching the "usability versus privacy" trade-off, we conducted massive research on the trade-off between the quality of the noisy output image and the privacy derived from it.

The outcome of this research is a website, through which users can submit their images, and view the resulting cloaked images while taking different factors into account (e.g., user privacy, visuality and run-time efficiency). The choice of parameters for our website is mainly based on the information described in this report.

CCS Concepts: • **Face recognition**; • **Adversarial machine learning**;

Additional Key Words and Phrases: adversarial machine learning, facial recognition, privacy

ACM Reference Format:
Boris Evdeev, Idan Rahamim, and Shiran Shaharabani. 2022. Technical Report- Face Cloaking Workshop. 1, 1 (August 2022), 9 pages.

1 INTRODUCTION

Nowadays, image processing technologies have reached unprecedented heights, and among them is face recognition technology. It means that a person's privacy is at risk from the moment he uploads his photo on the Internet, because even if he is not tagged on it, large campaigns can recognize him and accordingly, obtain information about him that he might like not to share.

In this paper, we consider and compare existing strategies to generate visually non-invasive facial noise masks that will protect against face recognition, and also offer our own method. Both articles, Faceoff [1], Ulixes [2], focus on photo tagging. This means that the face recognition systems at which our attack is directed want to assign to each face its tag, namely the identity of that face. The attack is aimed at fooling the system, so that it identifies the face incorrectly.

To achieve our goal, we use the loss functions proposed in the mentioned papers and also add one more, which should improve the quality of the image. We also compare these two products with each other and with ours, according to various metrics, namely the running time, the quality, and the accuracy of successful masking.

First, we describe the theory on which the two previously mentioned papers were based, and then we begin to implement the algorithms. After that, we evaluate the accuracy and quality of our algorithm using different parameters in a large number of images and test them with various face recognition systems. Further to that, we compare these results with the results described in the mentioned articles. At the end we implemented the site, which is the final result of our work.

Authors' addresses: Boris Evdeev, Tel Aviv University, Israel, borisevdeev@mail.tau.ac.il; Idan Rahamim, Tel Aviv University, Israel, idanrahamim@mail.tau.ac.il; Shiran Shaharabani, Tel Aviv University, Israel, shirans1@mail.tau.ac.il.

XXXX-XXXX/2022/8-ART $15.00
https://doi.org/

1.1 The face-recognition problem

In the face recognition problem, we have a set X that contains pictures of users' faces. We train a neural network f that maps X to the embedding space R^D that distinguishes between different users (i.e., the outputs of images of different users are far from each other in the embedding space $f(X)$). f can compute features that differ between people, such as distance between eyes, skin color, eye color, fat percentage of the face, distance between ears, symmetry of the face, etc. So even if we have two people who are not in our users' set, f should be able to distinguish. Usually, there is a constant γ such that if $|f(x) - f(y)| \leq \gamma$ we say that x, y are the same people, and otherwise not. If x_1 and x_2 are images of two different users, and x_1,x_3 are images of the same user, then $|f(x_1) - f(x_2)| > |f(x_1) - f(x_3)|$.

A face recognition system usually:

- Maintain a list of all users embedding representations. $U = \{x : f(x) | x \in X\}$.
- given face image x, to recognize it the system compute $u =_{u \in U} |f(x) - u|$. If $u \leq \gamma$ then recognize as the associated user. Otherwise the system recognize x as not in the users space X.

1.2 Adversarial Examples

The adversarial attack for a classification model M is a procedure that, given an input x, produces a new input x_{adv} (the adversarial example) such that M classifies x_{adv} incorrectly. The definition can be easily extended to other ML tasks such as regression. The restriction in x_{adv} is $x_{adv} \in B(x, \epsilon)$ with $\epsilon > 0$ for a specified distance function (where $B(x, \epsilon)$ is the group of all vectors at a distance of at most ϵ from x). If ϵ is small the optimization produces almost indistinguishable difference (Fig. 1).

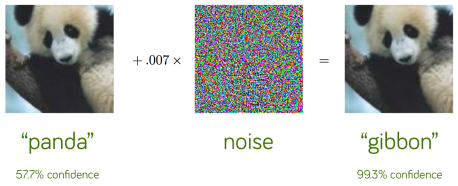

Fig. 1. Adversarial Example for Classification

1.2.1 Adversarial Examples in the Face Recognition Setting. This field is introduced by Face-Off [1]. For a given face image x, we wish to find a small perturbation δ such that no human can distinguish between x and $x + \delta$, but they are very different in the embedding space f (such that the face recognition system for $x \in X$ would not classify $x + \delta$ the same as x).

Fig. 2. Adversarial Example for Face Recognition

1.2.2 White Box versus Black Box. The white box setting is where the adversary has full access to the trained model. Therefore, it can calculate accurate loss gradients with respect to the input and use them to generate adversarial attacks. In contrast, in the black box setting the adversary has no knowledge or access to the model.

A common related concept is **transferability**. Which is a phenomenon that adversarial examples of similar architecture models that trained independent fool each other. Hence, to perform black-box attacks we can first apply a white-box attack on a similar network, and then use its output as an adversarial example to the target model.

2 RELATED WORK

In the context of image cloaking, there are several related published methodologies. The first to come is the Face-Off system [1]. To our knowledge, Face-Off is the first paper approaching generation of adversarial perturbations to prevent human pictures from being correctly recognized, both in the white-box and black-box settings. Their system has a pre-process to crop the face from the human image and after adding the face perturbation applying the post-process to paste it again in the original image. The face-off system also supports multiple face cloaking by iteratively handling each face with the described process.

To mislead a face recognition system, face-off solves the following untargeted optimization problem (using state-of-the-art solvers such as PGD [5] and CW [6]):

$$\max_{\delta} |f(x) - f(x+\delta)|_2 \quad s.t \quad |\delta|_\infty \leq \epsilon \tag{1}$$

Where x is the cropped human face and δ is the perturbation that we generate. ϵ controls the perturbation size, which means as long ϵ is small enough, human will find it hard to distinguish between the face image x and the adversarial example $x + \delta$. f is the face embedding function, which maps a facial image to the embedding space R^d.

Up to this point, we described the untargeted adversarial example generation of face-off. The targeted approach is similar and optimizes the following optimization problem for a given target image t:

$$\min_{\delta} RELU(|f(x+\delta) - f(t)|_2 - |f(x+\delta) - f(x)|_2 + margin) \quad s.t \quad |\delta|_\infty \leq \epsilon \tag{2}$$

Where the margin is the desired separation from the source image (when the loss reaches zero, we know that $|f(x+\delta) - f(t)|_2 < |f(x+\delta) - f(x)|_2$) by at least the specified $margin > 0$.

Another approach to the same problem is Ulixes [2]. Ulixes introduces the approach of solving an untargeted problem by choosing a target and solving the targeted formulation instead. Surprisingly, Ulixes' paper shows that solving the targeted optimization where the target is the same image (with not significant random noise, "No Sample" [2] Section 4.5) outperforms the Face-Off [1] untargeted approach results. Even more surprising is that Ulixes' "No Sample" outperforms any other Ulixes' tested target choices (such as farthest sample ([2] section 4.1) or random sample ([2] section 4.2)).

Further more, ulixes shows that their approach converges with significantly less iterations than the face-off untargeted approach, and practically generates adversarial examples faster and with higher quality.

Another paper [3] shows how to find a universal perturbation that works for multiple images. Their algorithm is called UEP and can significantly improve online service run time since the perturbation can be calculated once in advance.

3 METHODOLOGY

3.1 Perturbation Generation Algorithms

In this section we describe our implementation of Face-Off and Ulixes approaches for facial perturbation generation algorithms (section 3.1.2). We also describe the pre-processing (section 3.1.1) and post-processing (section 3.1.3) steps, which allows to move from working on the input image to working on the input image face. In addition, in Section 3.1.4 we describe how we handled the situation of multiple faces in the input image.

3.1.1 Pre-processing. This part is responsible for cropping the faces out of the image. To do this, we use the face detection network (MTCNN [7]) to determine the face boxes in the image. Face boxes are also used in the post-processing part. The preprocessing is also responsible to normalize the face image pixels to be between $[-1, 1]$ so later algorithms (for example facenet) can be applied.

3.1.2 Facial image perturbation. After cropping the face from the input image, we use the adversarial attack algorithm to create a cloaking perturbation for that face. We use different algorithms; one is based on the Face-Off [1] paper, and the other is based on Ulixes [2]. Both algorithms, in a nutshell, are solving an optimization problem where they restrict the perturbation size and minimize a certain loss. The main difference between the two approaches is the chosen loss.

When we use the Face-Off approach, we optimize according to the PGD solver [5] (see Section 4.0.2) the following untargeted optimization problem:

$$\max_{\delta} |f(x) - f(x+\delta)|_2$$
$$s.t \quad |\delta|_\infty \leq \epsilon$$

where we set f (the embedding neural network) to be FaceNet [4] (trained on the VGGFace2 dataset [8]). Note that since we set at the beginning of PGD our δ to be exactly 0, the objective is also 0. Therefore the gradient update of PGD will be 0 and therefore δ will never change and stay 0 as in the beginning. To solve this issue, we add to the loss function (the objective) that if $f(x+\delta) = f(x)$ then we add a small constant η to the right-hand side. Consequently, the objective becomes $|f(x+\eta) - f(x+\delta)|_2$, which is not 0 and, therefore, the gradient is also not 0. Another option could be to start with PGD with random perturbation instead of 0.

When we use the Ulixes approach, we again optimize according to the PGD solver the following:

$$\min_{\delta} RELU(|f(x+\delta) - f(t)|_2 - |f(x+\delta) - f(x)|_2 + margin)$$
$$s.t \quad |\delta|_\infty \leq \epsilon$$

Where again f is set to be facenet and the target sample t is chosen to be $x+\eta$, where η is a constant parameter of the algorithm. Note that, unlike face-off, here the first iteration gradient is not 0 and the algorithm does not need any modification.

3.1.3 Post-Processing. In the post processing part, we paste the perturbed face of the original image to the location it cropped from. We also transform the image back to RGB format N_{255}^{HxWx3} (N_{255} is the following set $\{0, 1, ..., 255\}$). Notice that rounding is applied here, and therefore the distance of the adversarial example from the original in the embedding space is not exactly kept.

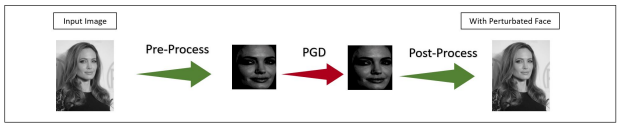

Fig. 3. Image Perturbation Generation Approach

3.1.4 Dealing with multiple faces. We used the MTCNN algorithm for faces detection. The PyTorch implementation allows for detection of multiple faces. For each of detected face we create a generate version (using the described algorithm) and paste the cloaked face in its original location of the input image. In this way we can handle any number of faces in the input image.

Fig. 4. Example of handling an input with multiple faces

3.1.5 Algorithm Improvements.

Perturbation loss. We found that both Face-Off and Ulixes approaches sometimes push certain pixels in the perturbation to the limit, although they not necessarily need it. Thus, we introduce a perturbation loss, which is a way for us to control the perturbation effect on the image.

Example. Using the following perturbation loss:

$$p_1(\delta) = |\delta|_2$$

The perturbation loss is added to the ulixes/face-off loss. We punish the algorithm for having large pixel modifications, and encourage it to reduce the amount of massive pixel changes. As we can see from the results, adding this loss significantly improves the visual quality of the perturbed images.

Fig. 5. The perturbation loss improvement. The left image is without the perturbation loss, while the right is with the perturbation loss, and has much better quality without compromising privacy. The execution parameters achieves mistake recognition rate of about 50% ($\epsilon = 0.03$, $amplification = 4$ - see figure 9)

3.2 Implementations

3.2.1 Face Recognition System. To evaluate the algorithm, we implemented a face recognition system which takes a dataset of images as input. We implemented a function that given a new image, finds it's closest image from the input dataset in the embedding space. If that distance is at most γ (a parameter) we return the image's user id, otherwise we return that the image's human does not appear in our input dataset. For our evaluation purposes, we set $\gamma = \infty$, since our project goal is to make an image not correctly recognizable - and being outside the dataset is a possible outcome.

3.2.2 Optimization Algorithm we used. The optimization algorithm we used for this report is PGD (Projected Gradient Decent). Given input x, tolerance factor ϵ, step size α, and number of steps:

(1) Initialize perturbation δ of the same shape of x filled with 0.
(2) Iteratively calculate the perturbation δ by the following recursive formula:

$$\delta_{i+1} = round_\epsilon(\delta_i + \alpha \cdot sign(\nabla_x |f_e(x+\delta) - f_e(x)|_2))$$

(3) Set $\delta = \delta_{steps}$

Where $round_\epsilon(x) = sign(x) \cdot max(|x|, \epsilon)$

3.3 Evaluation

3.3.1 Metrics.

Privacy Metric. (We call the opposite of this metric the "correct recognition rate" and use this notation to describe the results). The privacy metric is defined as, for a given face recognition system S that maps face image to a name then, the metric is the successful rate of fooling S with the perturbed samples. The calculation is as follows:

(1) For each image sample x in the face recognition system's users dataset X, we generate a perturbed example x_{adv} using face embedding network f_e and face detector f_d.
(2) Calculate the generated perturbation success rate as follows:

$$PM = \frac{\sum_{x \in X} 1_{[S(x_{adv}) \neq S(x)]}}{|X|}$$

Note that $PM \in [0, 1]$ and the closer it is to 1 the attack performs better in terms of privacy.

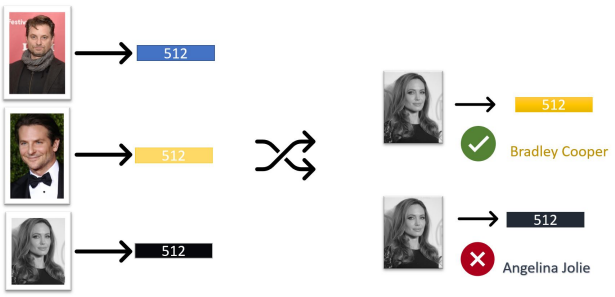

Fig. 6. Privacy Evaluation

White-box versus Black-box evaluation. The evaluation in the white-box setting is straightforward. In the black box setting, we need to generate perturbations with a different net network we use for testing (in instance, for the face recognition system). In order to do that, in our project, we only used pre-trained neural networks. In the white-box setting, both the network we use for perturbation generation and the testing network (i.e., face recognition system) are *facenet* architecture, trained on the *VGGFACE2* dataset [8]. In case of black-box setting, for testing we used also the same facenet network trained on the *VGGFACE2*, but for the perturbation generation we used facenet trained on *CASIA* dataset [9]. In future work we think it might be important to test on another pairs of network architectures and training datasets to examine the robustness of the results, such as in ulixes paper, which we built upon.

Generalized Privacy Metric. This metric is a generalization of the "Privacy Metric" shown above. We define the rank of an image x with label (name) y in S as the number of samples in the dataset that are more similar in the embedding space to x_y (the original image of y in the dataset) than x. We denote the rank by $rank_S(x)$.

Then the metric is the normalized average rank of the perturbed examples in the dataset.

$$GPM = \frac{\sum_{x \in X} rank_S(x_{adv})}{|X|^2}$$

Note that $GPM \in [0, 1]$ and the closer it is to 1 the attack performs better in terms of privacy. Also, note that GPM can mitigate performance in large and small datasets, while PM cannot.

(We did not evaluate the algorithms using this metric, since previous work doesn't work with that metric. We leave it here as an idea, maybe for future work).

Adapting PM and GPM to Black Box and White Box settings.

- Calculating PM and GPM in the white-box setting means that the face embedding network f_e is trained on the same dataset for evaluation (the face recognition system users).
- Calculating PM and GPM in the black box setting means that the network f_e is trained on a different dataset than the evaluation dataset (the face recognition system users).

Image Quality Metric. For most of the time we manually tested the quality of the images. We noticed by manually testing of many images that for the same amplification and epsilon, both algorithms have the same quality (ulixes and face-off). We also noticed that a good way to measure the quality of the image is to measure the effect of the perturbation, which is $\epsilon \cdot amplification$. Then to measure the quality we look at $1 - \epsilon \cdot amplification$. In future work, we would like to test the quality of images with a user study.

Efficiency Metric. The efficiency metric is very simple - the time (in minutes) required to generate the perturbation. This term is linearly dependent on the number of iterations parameter, which as we increase the number of iterations then the generation process duration increases.

4 RESULTS

In this part of the report, we examine our algorithms in terms of the described metrics. We run the system in many different states and check in every situation what is the winning approach. Each state we examine is determined by the following:

- N: the size of the set of pictures in the face recognition system we evaluate on. We tested on a small set of 100 pictures and on a larger set of 500 pictures.
- Epsilon value (ϵ): determines the privacy vs. quality of the image tradeoff.
- attack function: we examine two different attack methods, based on $face\text{-}off$ and $ulixes$ papers.
- iterations: The number of iterations is a parameter of the PGD algorithm for optimization.
- alpha: also a parameter of PGD. Determines each iteration step size.
- amplification factor: the amplification factor directly affects the privacy of the algorithm. We will analyze this parameter apart from the rest of the parameters. Also, we will examine the relation of amplification and ϵ values.

4.0.1 white-box results. The bottom line is that the best we got in terms of privacy in white box setting, with the described set of parameters (that ensures good quality), is 0.29 correct recognition rate of the face-recognition system on fake samples, compared to perfect recognition on the original samples.

An important point to note is that **N**, the number of pictures in the face recognition system significantly affects the evaluation of the privacy metric. On 100 samples (the smaller set we tested), we obtained a 0.37 correct recognition rate on the fake samples, while with the largest set (of 500) we obtained a 0.29 correct recognition rate. Which is 22% improvement just from increasing the test number of samples.

In this project, we did not have the resources (including time resources) to execute on much more than 500. Thus, we assume that on real face recognition systems that include many thousands of users, our algorithm will work much better than the one analyzed here.

We noticed that in the white box setting, the **attack function** (face-off or ulixes) is only responsible for a small change in the privacy (in the black-box setting the situation is very different, there ulixes performs significantly better). With ulixes the best performance we got is 0.29 correct recognition rate, while in face-off we got 0.31. Ulixes is an improvement of 6% over face-off for the white box setting. Also, the iteration parameter does not significantly affect privacy, as we will elaborate more later, therefore, from now on we set it to 10 to achieve reasonable runtime.

Now, we will investigate the relation of amplification and ϵ parameters, which has the most significant effect on privacy.

investigating the ϵ and amplification parameters. Both ϵ and amplification parameters affects directly the quality of the cloaked image. Furthermore, in our manual validations of the quality of the images in the experiments, we found that a good way to measure the quality of the cloaked image is:

$$quality = 1 - \epsilon \cdot amplification$$

Both parameters are important for controlling privacy, and both directly control image quality. In the following plot, we see that by tuning these two parameters, we can achieve good privacy with reasonable loss of quality, and achieving it without understanding one of the parameters is not possible.

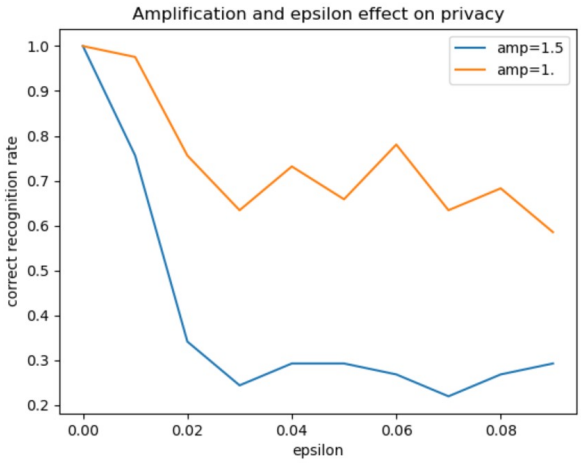

Fig. 7. ϵ effect on privacy with different amplification factors

We see that if we don't use amplification then we need to reduce the quality of the image in a non reasonable way in order to achieve good privacy (which we set to be about 30% correct recognition rate). When we use amplification, we see that the correct recognition rate metric improves significantly, and we obtain 22% correct recognition rate with an image quality of 95.5%. With only epsilon tuning, even with 90% quality, we have the worst correct recognition rate (about 60%).

investigating the iterations parameter in terms of efficiency. In terms of efficiency, we found that there are only two parameters that makes a different: the attack loss function (ulixes vs face-off) and the number of iterations. The attack loss function is implemented by us (since ulixes does not provide implementation, we could not test their implementation efficiency), and because we could have improved the loss performance by more sophisticated programming (with PyTorch), we ignore this parameter in terms of efficiency. In our implementations, Ulixes runs 2 times slower than face-off for the same number of iterations.

The iterations parameter linearly affects the efficiency, as written before:

$$efficiency = iterations \cdot const$$

while the constant depends on the implementation (i.e., face-off or ulixes).

In this part we examine which attack implementation is more "iterations" accurate. The following plot shows that both algorithms behave the same in terms of iterations:

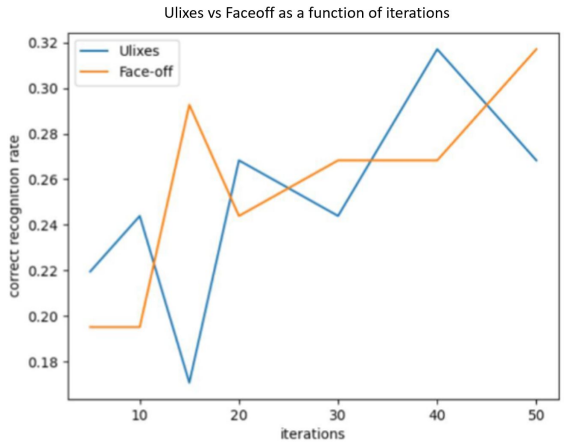

Fig. 8. Iterations effect on privacy for each algorithm

4.0.2 black-box results (transferability). Besides the type of algorithm (ulixes vs faceoff), the parameter ϵ and the amplification factor, the conclusions achieved in the white-box setting also hold here (iterations, N, α, etc.). In terms of the efficiency of the algorithms, there is no change between the white-box and black-box settings. In this setting, we noticed that the difference between the algorithms ulixes and faceoff is significant, where ulixes performs better than face-off.

We noticed that the most important parameter for the transferability is the amplification for both algorithms (ulixes and faceoff). In this part we will investigate the relation of ϵ and the amplification for both algorithms, and we will test this tradeoff while considering higher values of both parameters than done in the whitebox setting. This will yield a worse quality of the images than we achieved in the white-box setting, which is reasonable, since achieving privacy in the black-box setting is much more difficult. Firstly we analyze the ulixes algorithm, then we also evaluate on face-off and compare.

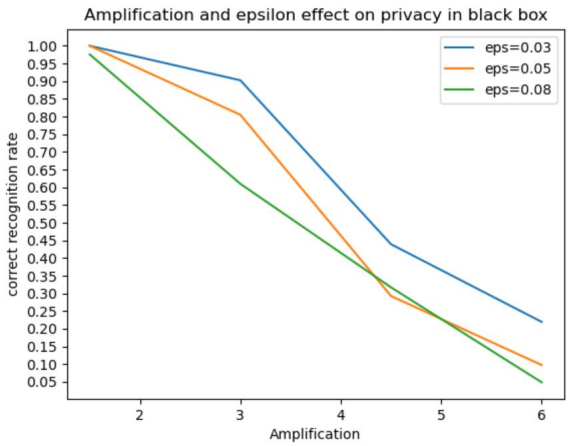

Fig. 9. Amplification and epsilon effect in the black box setting, the results are based on Ulixes algorithm.

In this figure, we see that the tradeoff of amplification and ϵ is very important for ulixes algorithm. We see that if we choose epsilon of 0.08 and amplification of 3, then the quality of the output image is 0.76 and the correct recognition rate is 60%. While if we choose slightly higher $amplification = 4.2$ and much lower $\epsilon = 0.03$, the quality is much better (twice as good) and the correct classification rate is even lower (about 50%)! Then we achieved better privacy with higher output image quality. This emphasizes the importance of this trade-off. In the user interface, we choose the parameters that, in our view, optimize this trade-off (we elaborate more about it later in the report).

In faceoff, the results are worse, the precise effect is presented in the following figure.

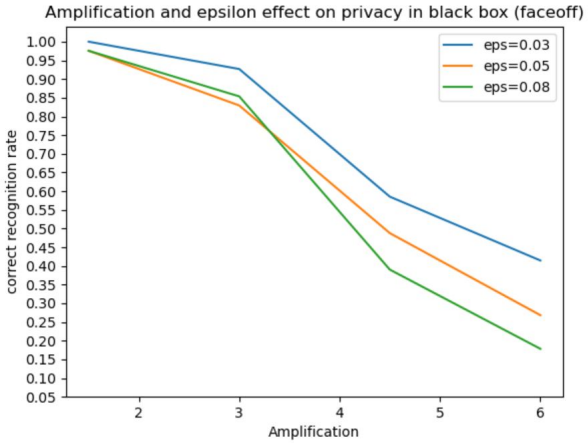

Fig. 10. Amplification and epsilon effect in the black box setting, the results are based on Face-off algorithm

We see that also in faceoff, the tradeoff of amplification and ϵ is very important. But, we also see that compared to ulixes, faceoff achieves much worse privacy for the same quality. This is presented in the following figure, where we present for certain qualities the privacy evaluation of both algorithms, and we see that for practical use, ulixes is preferable.

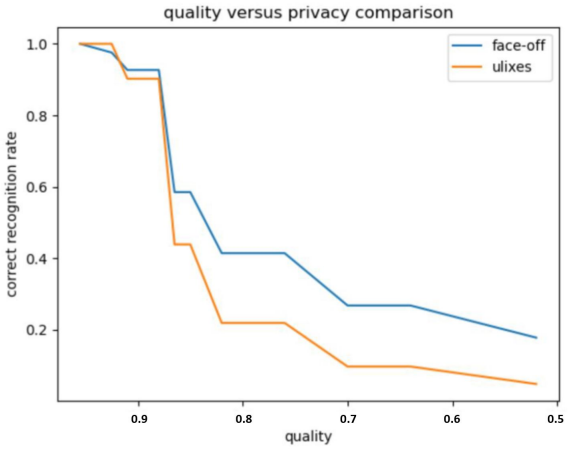

Fig. 11. Quality versus privacy tradeoff presented on both ulixes and faceoff.

We see that with no doubt ulixes outperforms faceoff in the black box setting. Therefore in our website we chose to mainly use the ulixes algorithm for cloaking.

4.0.3 Comparing cloaked images quality between white box and black box settings-Ulixes. In both the white-box and black-box settings, our goal was to achieve about 30% correct recognition rate. In whitebox setting the quality of cloaked images was much better than in blackbox, since we used lower values of ϵ (around 0.03) and amplification (around 1.5) in order to achieve the privacy target. In the black-box setting, in order to achieve 30% correct recognition rate we had to use much higher values of amplification (around 6) and higher values of epsilon (around 0.05). This leads to much worse quality cloaked images. We present a comparison of six random images: original images, white-box cloaking, and black-box cloaking.

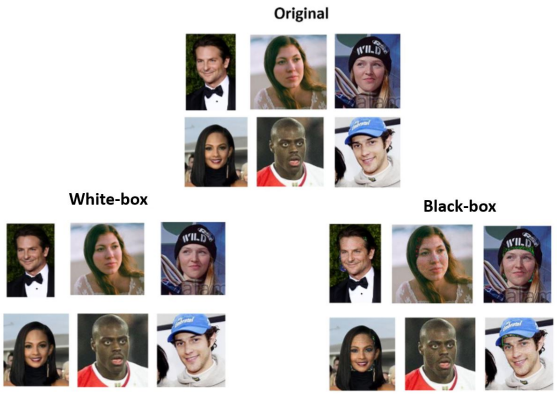

Fig. 12. original, white-box, black-box comparison

From far looking we might think that the white box cloaking is perfect quality, but if we look closer, we unfortunately see that also the white-box setting is not perfect and it changes the image attributes.

Fig. 13. zoom-in on Bradley Cooper image

4.1 The differences between our results and the results reported in the original papers

We will discuss the differences regarding the two papers which this report is based on: *face-off* and *ulixes*.

Regarding *face-off*, to our knowledge, the results achieved in this report are quite similar to the results achieved in the face-off paper. The noise required to achieve a certain level of privacy is similar in the two papers. In addition, face-off released an official implementation - in which for the same amplification and epsilon factors we noticed (manually testing) that the noise on the image is the same. An improvement of face-off implementation over our implementation, is that in the official implementation output we noticed that there are less "very noisy" pixels, these are pixels with extremely different color than of the original image. In the white-box setting, the perturbation loss makes the two almost exactly similar. In the black-box setting, where we use larger values of amplification, the official implementation works better with respect to the extreme pixels color change.

Fig. 14. Example of Barack Obama input image, taken from face-off implementation repository. Amplification used is 4 and ϵ is 0.1 for both images.

Ulixes does not provide an open-source implementation, therefore, we cannot compare with their algorithm implementation. We have contacted with one of ulixes authors (Thomas Cilloni), he validated that our loss function is correct. The declared average running time in Ulixes paper is 3 seconds per image, while we have 20 seconds for an image with a single face. This may be due to different factors such as different implementations, checks on different images, and different PC specs (we used Intel Core i5 generation 8

without GPU, while ulixes used Intel Core i5-8250U without GPU). Also, there is a difference in the perturbation itself, as we can see in Fig.15, even with our perturbation loss improvement, ulixes paper shows a better visuality. This may be due to different implementations, improvements, and different accuracy metrics used to evaluate attack success.

We also mention that in our implementation, different margins did not affect our results (thus, we have set it fixed to 10), although in the original paper it was stated that this mattered. It also may depend on implementation itself, and we cannot check it because Ulixes does not provide open-source implementation.

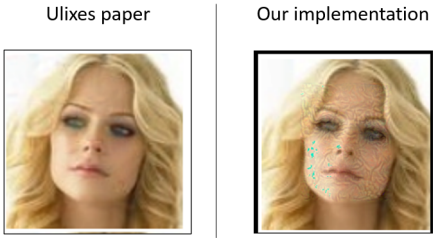

Fig. 15. Example of 50% attack success comparison

5 DISCUSSION

In this work we have been exposed for the first time to the field of adversarial machine learning. The danger that machine learning poses to our privacy is huge and we are happy to step forward with a world with privacy defenses. In this work we discussed on ways to compromise the quality of images by adding a perturbation that fools well trained recognition systems. We believe that the field of privacy using adversarial machine learning methods will be used much more widely by daily users. For that matter, the research of improving the image quality while preserving some defences should go forward, and we believe it is possible in the near future.

The main limitation of the work is the black-box scenario, which is also a practical scenario. In the transition from white-box setting to black-box setting we increased the divisible perturbation, and we think that future work on transferability will bring the main contribution to the field.

We also think that the practical aspect has to be the focus of future research. In addition to transferability, users might also have certain preferences, and one could develop a loss function that specifically targets those preferences. For example, users may want the perturbation to not touch the face (be in the face box but outside the face surface). Or they might prefer a worst case number of pixels change kind of approach. Researchers can know and tackle this only if there is a comprehensive user study that asks these questions and examines them in real users.

6 CONCLUSION

The main takeaways from our project are:

- We implemented using PyTorch, a package that allows adversarial attacks to be executed in the face-recognition setting.
- We evaluated the privacy and efficiency of the cloaking methods with many different parameter combinations and achieved interesting results, which we could not achieve without participating in those experiments.
- We developed a face-recognition system in order to evaluate cloaking algorithms. We also defined new evaluation metrics that helped us to understand the performance of the algorithms.
- We introduced perturbation loss, which is an addition to face-off / ulixes loss that shapes the way the perturbation will be. In our work we saw that adding this perturbation loss improves face-off and ulixes actual quality performance.
- Development of a user interface: a site using flask technology.

7 CONTRIBUTIONS

The pre-process and post-process utilities were developed mainly by *Idan Rahamim*. The *PGD* algorithm is based on the repo https://github.com/shinington/facesec, and the adaptation was led by *Boris Evdeev*. The development of the face recognition system used for the evaluation was mainly developed by *Shiran Shaharabani*. The writing of the loss functions was done by all of us *Idan Rahamim* also communicated with the author of the Ulixes paper *Thomas Cilloni* to validate the correctness of the loss functions. *Idan Rahamim* and *Shiran Shaharabani* developed the site using flask technology, also in Python, which combines all the different components of the work. The technical report was written mainly by *Boris Evdeev*, with the help of the rest of the team. The user study was led by *Shiran Shaharabani*. The team worked together in full support of each other, from reading the papers to writing the representation, experiments, and reports. This was a challenging project that we could only succeed to perform together. We are glad with what we have achieved and the great knowledge we have gained.

REFERENCES

[1] V. Chandrasekaran, C. Gao, B. Tang, K. Fawaz, S. Jha, and S. Banerjee. Face-Off: Adversarial face obfuscation. Proceedings on Privacy Enhancing Technologies, 2021(2):369–390, 2021.
[2] T. Cilloni, W. Wang, C. Walter, and C. Fleming. Ulixes: Facial recognition privacy with adversarial machine learning. Proceedings on Privacy Enhancing Technologies, 2022(1):148– 165, 2022.
[3] A. Rajabi, R. B. Bobba, M. Rosulek, C. V. Wright, and W.-c. Feng. On the (im) practicality of adversarial perturbation for image privacy. Proceedings on Privacy Enhancing Technologies, 2021(1):85–106, 2021.
[4] F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 815-823, doi: 10.1109/CVPR.2015.7298682, 2015.
[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," arXiv preprint arXiv:1706.06083, 2017.
[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in Security and Privacy (SP), 2017 IEEE Symposium on. IEEE, 2017, pp. 39–57.
[7] Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Processing Letters, 23(10):1499–1503.
[8] Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi and Andrew Zisserman (2018). VGGFace2: A dataset for recognising faces across pose and age. IEEE Signal Processing Letters.
[9] J. Dong, W. Wang, and T. Tan, CASIA image tampering detection evaluation database. in Proc. IEEE China Summit Int. Conf. Signal Inf. Process., Jul. 2013, pp. 422–426.

8 APPENDIX: WEBSITE USER INTERFACE

Our final product is a website where users can upload images they wish to cloak. The site is built by Flask web-framework and served by Waitress, which is a production-quality pure-Python WSGI server.

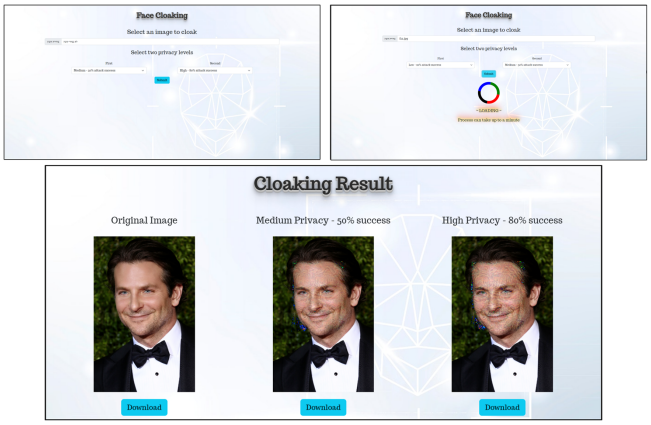

Fig. 16. Website design - home page, loading page, results page.

In order to reach the required functionality, input validation and design, we use current major tools such as html5, css3, bootstrap5, and javascript ES6. With these tools, we were also able to use the responsive web design approach, which means that our web pages render well on a variety of devices and window or screen sizes from minimum to maximum display size, to ensure usability and satisfaction. The server and the website were checked on Windows, MacOS and Linux operation systems, and on Chrome, Safari, Firefox, Microsoft edge and Opera browsers.

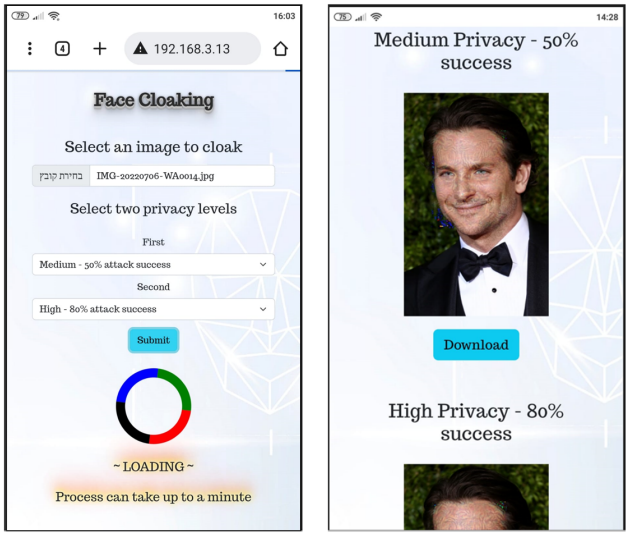

Fig. 17. Responsive web design - on smartphone images are one below the other.

Once a user uploads an image, he has to choose two privacy levels from the options (Low, Medium, High), where they represent (20%, 50%, 80%) attack success, respectively. Then, the remote located server validates the input, runs the cloaking algorithm, and displays 3 result images. The first is the original image, while the second and third are cloaked images with the selected privacy levels. Both cloaked images are processed by Ulixes algorithm. The epsilon used for all privacy levels is 0.03 (we found it most effective- see Fig. 9), and amplifications are (3.2, 4.2, 6) for (20%, 50%, 80%) attack success respectively. Although Ulixes runs 2 times slower than faceoff, we chose to run only ulixes. The reason is that in our study we found that ulixes outperforms faceoff in the black-box setting in relation to quality and privacy (Fig. 11), and we think Ulixes runtime is still reasonable- less than a minute for the whole process on a low-end laptop.

We also created a download button below each displayed image. In order to collect statistical information, when a user clicks on a download button, the server receives this information and eventually saves it in a SQLite3 database or in a txt file (which we can choose depending on our preference). To prevent data corruption, we note that the server receives information on at most one download click for each displayed privacy level (per cloaking process). This information can indicate user preferences in relation to usability and privacy. The button below the original image is redundant because the user uploads it, but we created it in case we want to change the purpose of the button in the future to a survey button. Means, we could ask the user to click on the image that he prefers.