Multi-Target Adversarial Attacks with LOTS

Master Thesis

Noah Chavannes
16-701-872

Submitted on
April 30 2022

Thesis Supervisor
Prof. Dr. Manuel Günther

University of Zurich^{UZH}
Department of Informatics

Master Thesis

Author: Noah Chavannes, noah.chavannes@uzh.ch

Project period: 01.11.2021 - 30.04.2022

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

Acknowledgements

I would like to express my gratitude to everyone who has supported me in any way during this work.

First of all, I would like to thank Prof. Dr. Manuel Günther, who was always available for questions and provided precious feedback. I had complete freedom in designing my experiments and structuring my thesis and always felt well supported. I always found the discussions exciting and informative. I learned a lot from you and am grateful for this opportunity. Thank you!

Secondly, I would like to thank Dr. Andrin Pelican, who supported me with his excellent statistical knowledge and hints that guided me in the right direction.

Finally, I would like to thank my girlfriend, my family, and my friends who supported me with great forbearance, patience, humor, and distractions.

Abstract

Face recognition systems are on the rise and are being widely used throughout the industry. With the advance of face recognition systems, more and more adversarial attacks are emerging. Layer-wise Origin-Target Synthesis is one such attack in which the image of a source person is iteratively modified so that a face recognition system identifies it as another person. We extend this approach by allowing one input image to mimic multiple targets simultaneously. We further improve the loss function of the approach by including additional components that measure the structural similarity between the original image and the adversarial image. We evaluate our new method quantitatively with experiments and conduct an empirical analysis with 73 participants to investigate the relationship between human perception and similarity metrics. Our results show that we can successfully perform multi-target attacks and keep perturbations minimal. We also show how different source-target constellations affect the quality of adversarial images. Lastly, we demonstrate that the similarity metrics used to measure the size of perturbations are not perfect predictors of human perception.

Zusammenfassung

Gesichtserkennungssysteme sind auf dem Vormarsch und finden breite, branchenübergreifende Anwendung. Mit dem Vormarsch solcher Systeme tauchen auch vermehrt Angriffe auf gegen sie auf. Die "Layerwise Origin-Target Synthesis" ist ein solcher Angriff, bei welchem ein Bild einer Ausgangsperson iterativ so verändert wird, dass es von einem Gesichtserkennungssystem als eine andere Person identifiziert wird. Wir erweitern diesen Ansatz, sodass ein Bild einer Person mehrere Zielpersonen gleichzeitig nachahmen kann. Wir verbessern die Loss-Funktion des Ansatzes, indem wir zusätzliche Komponenten hinzufügen, die die strukturelle Ähnlichkeit zwischen dem Originalbild und dem generierten Bild messen. Wir bewerten unsere neue Methode quantitativ mit Experimenten und führen eine empirische Analyse mit 73 Teilnehmern durch, um die Beziehung zwischen menschlicher Wahrnehmung und den eingesetzten Ähnlichkeitsmetriken zu untersuchen. Unsere Ergebnisse zeigen, dass wir erfolgreich Multi-Target-Angriffe durchführen und die Veränderung des Bildes minimal halten können. Wir zeigen auch, wie sich unterschiedliche Zusammensetzungen von Ausgangsperson und Angriffszielen auf die Qualität der generierten Bilder auswirken und dass die strukturellen Ähnlichkeitsmetriken, die wir zur Messung der Größe von Perturbationen verwenden, keine perfekten Prädiktoren für die menschliche Wahrnehmung sind.

Contents

Chapter 1

Introduction

Face recognition systems are capable of assigning a person's image to a specific identity. In addition, such systems can also be used for verification, as is the case when unlocking a mobile phone. The idea behind face recognition has been around for a long time and has been experimented with since the 1960s. However, it was not until the advent of more powerful computers and even larger datasets that highly efficient face recognition systems using deep learning concepts emerged (Yan et al., 2019). Since then, face recognition technology has been used in various applications including automated border control, photo tagging in social media, citizen surveillance, and military purposes (Vakhshiteh et al., 2021).

As face recognition technology has advanced, more and more approaches for attacking such systems have emerged. One category of attacks are adversarial attacks. In an adversarial attack, an image is altered so that a face recognition system can no longer recognize a person or even wrongly identifies the person as someone else. Such adversarial attacks can be carried out by digitally altering images and physically, e.g., wearing sunglasses with a special print (Sharif et al., 2016). There are attacks in which someone tries not to be recognized (untargeted attack) and attacks in which the attackers try to have their images recognized as a specific different person (targeted attack). Networks can be attacked of which both, the architecture and the dataset used, are known (white-box attack), and networks of which no details are known (black-box attack). However, black-box attacks are still limited and mainly focus on untargeted attacks. The goal of such attacks can be different. For example, one can use an untargeted attack to avoid being automatically detected by social media platforms. Alternatively, one can use a targeted attack, which would allow someone to use someone else's passport to smuggle through automated border control. However, there are also dangerous attacks in which, for example, road signs are manipulated with stickers to force self-driving cars to interpret a stop sign as a speed 80 sign (Kurakin et al., 2018; Vakhshiteh et al., 2021).

Rozsa et al. (2017) presented the layerwise origin-target synthesis technique (LOTS), a targeted adversarial attack that modifies an input image of a person so that a face recognition system identifies this person as a specific different person. In this process, so-called feature vectors are extracted from a convolutional neural network which represent a person's facial characteristics. The authors use these feature vectors to iteratively modify the original image to include the facial characteristics of another target. The work is limited in the sense that only one target can be attacked at a time.

In this thesis, we want to extend the LOTS approach so that it can attack multiple targets simultaneously. In the course of this adaptation, we will also optimize the loss function so that the perturbations present in the adversarial image are minimal. The goal is to generate minimal perturbations such that the network classifies an input image as any number of other people. Therefore, this is a targeted attack. Our method could be used to generate a passport photo that would allow multiple people to travel with the same passport simultaneously.

Our contribution includes the extension of the LOTS algorithm to multiple targets and the adaption of the loss function by integrating two new loss components that take into account the similarity between the original image and the adversarial image. Furthermore, we evaluate our adaptations in quantitative experiments and compare the results with the original LOTS algorithm. In addition, we implemented a custom questionnaire tool and surveyed 73 participants. We analyze this empirical data to draw conclusions about the similarity metrics used. We can use our algorithm to generate adversarial images that have good similarity scores and attack multiple targets simultaneously. We show how different constellations of source and targets within an attack affect the quality of the adversarial images and how we can interpret the similarity measures.

The thesis is structured as follows: In Chapter 2, we discuss related work and the current state of research on face recognition software and adversarial attacks. Chapter 3 discusses the background, which serves as the foundation for understanding the thesis and covers the dataset, the network architecture, and the original LOTS algorithm and its concept. In Chapter 4, we highlight the preprocessing steps necessary to extend the LOTS algorithm, we extend the LOTS algorithm such that it can mimic the features of multiple targets simultaneously, and we improve the loss function. In Chapter 5, we perform a hyperparameter search for the extended LOTS algorithm. Additionally, we introduce source-target sampling methods and perform and evaluate the quantitative experiments regarding the extended LOTS algorithm. In Chapter 6, we implement our custom questionnaire tool, which we use to perform an empirical analysis. Subsequently, the collected data is analyzed and evaluated. In Chapter 7, we discuss our findings, and in Chapter 8, we conclude our work and present possible avenues for future work.

Chapter 2

Related Work

This chapter will discuss the origins and the current state of research in face recognition. Furthermore, we will identify and categorize different adversarial attack methods.

2.1 Face Recognition

Face recognition can be divided into two main techniques. The first technique is face identification, which describes the process of assigning a unique identity to an image of a person. In most cases, face detection is applied first, in which it is scanned whether a human face is present in an image and where exactly it is located. Furthermore, there is the process of face verification, which compares two images to see if they contain the same person without knowing their exact identity. We will briefly discuss the origins of face recognition and then describe the current state of research in more detail.

2.1.1 Origins of Face Recognition

In the 1960s, Woodrow W. Bledsoe and other researchers took the first steps toward face recognition (Bledsoe, 1964). Their research contract was issued by the U.S. Department of Defense (Ballantyne et al., 1996). In their approach, users had to manually mark the coordinates of 20 facial landmarks, e.g. the coordinates of the subject's pupils, upon which an algorithm searched for the person with the most similar characteristics in a database. Even then, however, they had already problems with rotations, scaling and tilts of different subjects (Bledsoe, 1966). Goldstein et al. (1971) then automated Bledsoe's approach in 1970 and increased the number of facial landmarks to 21. Sirovich and Kirby (1987) then presented the Eigenface approach, in which a set of base images can be created using PCA on a dataset. An arbitrary image from the dataset can be reconstructed by linearly combining the resulting base images. Turk and Pentland (1991) extended this work and presented the first automated face recognition system based on the Eigenface approach. Wiskott et al. (1997) presented the Elastic Bunch Graph Matching (EBGM) algorithm, which could be used for face recognition. In this approach, filters are created with the help of Gabor wavelets. These filters can be applied to recognize local textures. Using the locations of these textures, a graph can be spanned over an image in which the nodes denote textures' locations, and the edges denote the distances between individual textures. Several such graphs are combined to form a bunch graph, which can be used to display and recognize different faces by recombining different nodes and edges. However, face recognition with EBGM only works with identical poses. In the 2000s, the research focus was directed toward handcrafted features, where the Gabor filters found further application (Liu and Wechsler, 2002). In 2010, Cao et al. (2010)

introduced a methodology that uses unsupervised learning and PCA to express faces through low-dimensional feature vectors. However, this methodology struggled with different uncontrolled facial variations. Thereafter, the focus of the research community shifted toward Deep Learning.

We only presented a brief extract of the history of face recognition technology. The survey paper by Zhao et al. (2003) covers the origins of face recognition in more detail.

2.1.2 Current State of Face Recognition

Since the advent of Deep Learning and Convolutional Neural Networks (CNN), face recognition has made significant progress in recent years. Higher computing power and ever-increasing datasets further contributed to the success of face recognition techniques (Parkhi et al., 2015).

Only large companies have the resources to create their own large datasets. Therefore, it is not surprising that Facebook's Taigman et al. (2014) introduced DeepFace in 2014, an approach that achieved the best performance on the Labeled Faces in the Wild (LFW) benchmark (Huang et al., 2008) and the YouTube Faces (YTF) dataset (Wolf et al., 2011). They achieved these results using a private dataset containing 4.4 million images uploaded to Facebook. The authors introduced a multi-step process where they first perform face detection and identify landmarks. Based on these landmarks and with the help of a 3D model, the face is frontalized, i.e., deviating poses are converted into a frontal pose. After the alignment, they pass the image to a deep CNN to extract features and compare them using a weighted chi-square similarity. In addition, a so-called siamese network architecture was followed, in which two identical networks share the same weights but are applied to two different inputs. This is used to verify if two different images contain the same person. Ultimately, DeepFace consists of an ensemble of several deep CNNs trained on different seeds, with and without the siamese architecture. The authors later built upon their work and trained on a larger dataset of 10 million identities with 50 images each. They were able to show how the generalization ability of a model can be improved by controlling the dimensionality of the last fully connected layer (Taigman et al., 2015).

A short time later, Schroff et al. (2015) from Google introduced FaceNet, which debuted the triplet loss. With triplet loss, the model is trained with three image samples simultaneously. The first image is the so-called anchor, which is the reference image. In addition, a positive sample (same identity as the anchor) and a negative sample (different identity as the anchor) are supplied. The triplet loss, unlike previous approaches, not only tries to minimize the distance between positive samples and the anchors but also to maximize the distance between negative samples and the anchors. The triplets are selected so that the distance between the anchor and a positive sample is maximal, and the distance between a negative sample and the anchor is minimal. During training, the face descriptors are embedded into a Euclidean space. Doing so allows face verification tasks to be solved with Euclidean distances and face recognition with a k-nearest neighbor (k-NN) algorithm. The authors trained a CNN on datasets of different sizes, ranging from 2.6 million to 260 million images, and showed that the performance improvements are only marginal after a certain dataset size. According to the authors, the network does not require preprocessing steps apart from cropping and is robust to invariants of lighting, age or poses. At the time of publication, the authors had the best face verification performance on the LFW dataset. They were also able to outperform Facebook's DeepFace on the YTF benchmark. While the FaceNet method used euclidean distances, the research direction switched to cosine distances shortly thereafter.

Liu et al. (2017a) presented SphereFace in 2017. The authors wanted to solve the problem that the maximum intra-class distance in open-set face recognition is often larger than the minimum inter-class distance. This can lead to problems during training and inference. They introduced an angular softmax loss (A-Softmax) to solve this problem, which projects the extracted facial features onto a hypersphere. They justify this with the intuition that faces also lie in a non-linear

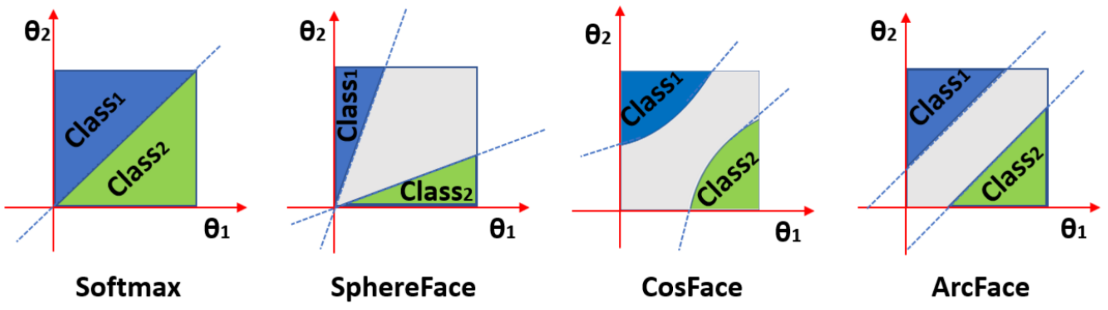


Figure 2.1: CLASSIFICATION BOUNDARIES OF LOSS FUNCTIONS. Visualization of the classification boundaries of the different losses considering the case of a binary classification. (Deng et al., 2019)

space. The authors could not outperform FaceNet on the LWF and the YTF benchmark with their SphereFace by a narrow margin. They further achieved 85.56% verification performance on the MegaFace Challenge (Kemelmacher-Shlizerman et al., 2016).

Wang et al. (2018) present with CosFace an alternative to SphereFace. They want to solve the same problem regarding softmax functions' lack of discriminatory power. They propose the large margin cosine loss (LMCL). Similar to the A-Softmax loss of SphereFace, an angular margin is considered. However, in contrast, the angle is considered in the cosine space and not only in the angular space, leading to an even more apparent separation of classes. In addition, the feature and the weight vectors are normalized with respect to the L_2 norm so that during training, only the angle is relevant for learning the discrimination. With their approach, they achieve better performance on the LFW and the YTF benchmark than the authors of FaceNet. With 96.65%, they outperformed SphereFace on the MegaFace Challenge by a wide margin.

The authors of ArcFace have introduced another alternative to CosFace and SphereFace with the Additive Angular Margin Loss (Deng et al., 2019). The approach differs in that the geodesic distance is considered, resulting in better discriminatory power due to more precise geometric mapping on a hypersphere. The Additive Angular Margin Loss results in a linear margin within the whole interval, whereas CosFace and Sphere Face apply a non-linear angular margin. Figure 2.1 from the authors' work visually illustrates the classification boundaries of the different methods considering the case of a binary classification. The authors achieved the best performance on all three benchmarks at the time of publication. Compared to SphereFace, CosFace, and ArcFace, which use advanced loss functions, the network we use in our thesis was trained using a standard Softmax loss function. Our extended LOTS algorithm generates the adversarial images using a multi-component loss, which with the cosine distance, considers an angular component.

In 2019, Yan et al. (2019) introduced VarGFaceNet, a new light-weight network architecture. The goal of the network is to have high discriminatory power and high generalization capabilities. They achieve this through variable grouping, which was introduced with VarGNet (Zhang et al., 2019). According to the authors, the main problem is that there are many different identities in large datasets. Nevertheless, they want to achieve small computational costs while still being able to discriminate between a large number of identities. They achieve this by introducing special network blocks and stringing together convolutions optimized for face recognition tasks. The authors demonstrate the effectiveness of their model with the best performance on the LFW dataset (as of March 2022).

ElasticFace was introduced in 2021 by Boutros et al. (2021). The authors claim that softmax losses like ArcFace and CosFace, which consider the geodesic distance and use a fixed penalty margin, do not work as expected under real conditions. This is due to inconsistent inter-class and intra-class variation. With ElasticFace, they propose a softmax loss that makes the margin

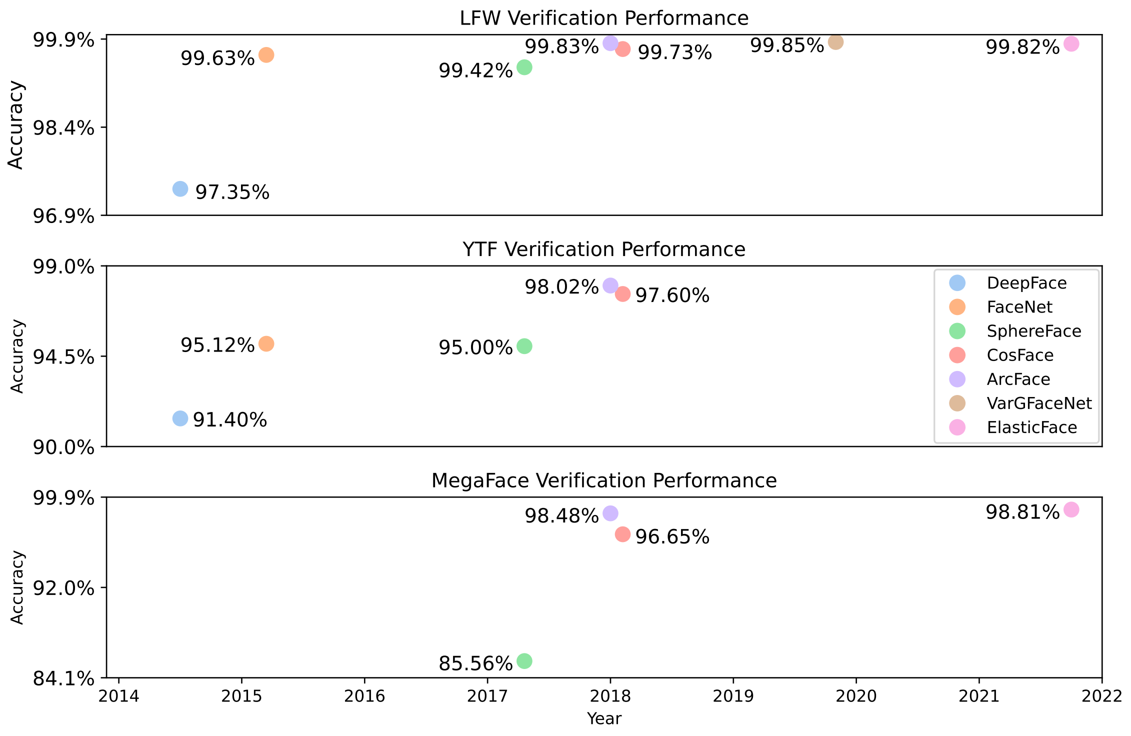


Figure 2.2: VERIFICATION PERFORMANCE BENCHMARKS. Verification performance of different works on the LFW (first row), the YTF (second row) and the MegaFace benchmarks (third row).

constraints elastic and allows a model to adapt to the dataset dynamically. With their method, they beat seven out of nine mainstream benchmarks. They achieved exceptional verification performance on the MegaFace Challenge with an accuracy of 98.81%.

A verification performance comparison on the LFW, the YTF and the MegaFace benchmark of the works mentioned above can be found in Figure 2.2.

2.2 Adversarial Attacks

This section will look at the origins and the current state of research on adversarial attacks. Adversarial attacks are defined as the modification of an image so that a network under attack generates a misclassification. Whereby the modifications (perturbations) should be imperceptible to the human eye Vakhshiteh et al. (2021).

2.2.1 Origins of Adversarial Attacks

Szegedy et al. (2014) were the first to use an L-BFSG method to compute perturbations that were not visible to a human and yet could trick a neural network. A little later, Goodfellow et al. (2015) proposed the Fast Gradient Sign Method (FGSM), which can generate an adversarial image using a one-step approach. This is done by adding a value to the original image, which is calculated from the direction of the gradient sign of the loss function. Kurakin et al. (2018) extended the

FGSM idea with the Basic Iterative Method (BIM). Instead of a one-step process, perturbations are added to the image in an iterative manner. From this method, which is also known as the Iterative Fast Gradient Sign Method (I-FGSM), the idea for the Iterative Fast Gradient Value Method (I-FGVM) was born. In I-FGVM, an update is made in the direction of the gradient value instead of the sign of the gradient (Rozsa et al., 2016). As an extension of the I-FGVM, Rozsa et al. (2017) later introduced the LOTS algorithm, which serves as a foundation for this thesis.

Su et al. (2019) have presented a method to trick a network by changing only one pixel. They use Differential Evolution to find these pixels (Das and Suganthan, 2010). Furthermore, access to the classification outputs of the network is needed. Differential Evolution randomly changes pixels and measures, which changes have decreased the network's confidence with respect to the correct label the most. The pixels that caused the most significant change in confidence are selected and slightly modified in the next step. Then, the process is repeated with the slightly modified pixels until a pixel is found where the network no longer predicts the correct class. Another targeted attack that focuses on changing as few pixels as possible is the Jacobian-Based Saliency Map Attack (JSMA) (Papernot et al., 2016). The L_0 norm of the perturbation constrains the attack. In this approach, one pixel is changed at a time, and the effect on the resulting classification is recorded in a saliency map. This map is later used to select the pixels that promise the greatest probability of success. While JSMA and the method introduced in this thesis are both white-box and targeted attacks, JSMA limits the number of pixels changed, whereas we do not constrain the perturbations. Unlike previous approaches, Moosavi-Dezfooli et al. (2017) have presented an image-agnostic algorithm called Universal Adversarial Perturbations. The algorithm is able to trick a network on multiple images with the same perturbation. It is an iterative approach, which can also generalize well between different network architectures. The main difference to our thesis is that we generate image-specific perturbations, whereas the authors generate universally valid perturbations used in untargeted attacks.

In order to resist adversarial attacks, defense mechanisms were also researched. One such approach was defensive distillation, in which a second network is used to predict the output of the first network, which helps make a network more robust against attacks (Papernot et al., 2015). Carlini and Wagner (2017) then presented three attacks and showed that their methods also work for networks using defense distillation. For the attacks, small perturbations are sought to generate a misclassification. The perturbations are constrained in their L_0, L_2, or L_∞ norm depending on the attack. Further adversarial attacks and their origins are discussed to a greater extent in the surveys by Akhtar and Mian (2018) and by Vakhshiteh et al. (2021).

2.2.2 Current State of Adversarial Attacks

There are various ways to categorize adversarial attacks. For example, they can be categorized as white-box versus black-box, targeted versus untargeted, or image specific versus universal attack. A further way of categorization is by dividing the adversarial attacks by their strategy. The first category deals with attacks that seek to exploit the weaknesses of different network architectures. The second category covers physical attacks in which a real person's appearance is changed to fool a network. The last category deals with geometric transformations intended to trick a network (Vakhshiteh et al., 2021). We will focus on face-related adversarial attacks in the remainder of this section.

CNN-Based Attacks

Goswami et al. (2018) have presented two image-level distortion methods that can significantly reduce the verification capabilities of networks. Their image-level distortions can be applied to any image and were not exclusively designed for faces. However, they apply the distortions to images containing faces in their experiments. The first approach is called Grid-based Occlusion, in which pairs of points are selected at the edges of the image and connected with a one-pixel thick line. In the second approach, three sets of pixels are selected. In the first set, the most significant bit is flipped with a certain probability; in the second set, the second-most significant bit is flipped, and in the third set, the third most significant bit is flipped. This results in noise, which is applied to the image. The authors show that verification performance decreases noticeably on different networks. Kwon et al. (2019) introduced the Face Friend-safe adversarial attack method. In this method, a friendly network and an enemy network are used to generate adversarial images. The goal is to find an adversarial image with minimal perturbations such that the friendly network still correctly predicts the original label. However, the enemy network should no longer correctly predict the label of the adversarial image. The authors achieve this with a transformer that uses the losses of the two networks to generate distortions that should be minimal.

Nguyen et al. (2020) have developed a system that generates master faces that can be used to fool face recognition networks. Their work is inspired by Bontrager et al. (2018), who generated universal fingerprints (master prints) that could bypass biometric security systems. A master face is a face that is intended to resemble as many faces as possible. Using a generative adversarial network (GAN) and an evolution algorithm, the authors generate master faces. These faces are fed into a face recognition system, and a score is calculated. The evaluation algorithm uses the scores to influence the input for the next iteration of master faces such that they improve and resemble more people. The authors were able to show that False Match Rates (FMR) of up to 35% could be enforced when attacking face recognition systems with master faces. However, they achieved the best results when knowledge about the dataset was available. In addition, they showed that limited success is possible even when the network architecture or the dataset is unknown. This work has certain similarities to our thesis in that both approaches attempt to modify a face to look as similar as possible to many other faces. However, our thesis deals with a targeted attack, in which adversarial perturbations alter existing images of people. The work of Nguyen et al. (2020) deals with an untargeted attack, in which the faces are artificially generated using a GAN.

Physical Appearance Attacks

Sharif et al. (2016) have presented a system that physically attacks a network by 2D or 3D printing the frame of eyeglasses, which contains perturbations. The perturbations must be robust to different perspectives. They collect a set of images, and search for a perturbation for which all images of the set are misclassified. This perturbation is created in an iterative procedure. The attack classifies as an untargeted attack, in which the aim is to prevent a person from being recognized. Another physical attack approach, Visible Light-based Attack (VLA), was presented by Shen et al. (2019). They generate perturbations and project them onto a person's face using a light source. Targeted and untargeted attacks are possible. The authors use a perturbation frame, in which the actual perturbations are present, and a concealing frame, with which the perturbations are to be made imperceptible for humans. The authors rely on the persistence of vision (POV) effect to hide the perturbations. Two different images (perturbation frame and concealing frame) are projected onto the face of the human with a alternating frequency of 25Hz. The human brain does not perceive the two rapidly successive images separately and thus fades the colors together. However, for a camera with a fast shutter speed, the perturbations are clearly visible (Zhang et al., 2015). Zhu et al. (2019) have presented an attack in which makeup is used to generate adversarial images. Two different GANs are used. The first GAN transfers makeup to a face without makeup, and the second GAN hides perturbations in the regions where the makeup was applied. The method can be used both as an untargeted and a targeted attack. While physical appearance attacks change a person's appearance noticeably, we try to hide the perturbations as well as possible.

Geometrical Attacks

Dabouei et al. (2019) have introduced the Fast Landmark Manipulation (FLM) method. In this method, landmarks are detected with a face detector and displaced by a spatial transformation. The displacement of the individual landmarks is done by a flow displacement, which is calculated using the gradient with respect to the landmark location. Since the gradient can point in all directions, the results are sometimes not quite natural-looking images. In a further experiment, they introduced Grouped Fast Landmark Manipulation (GFLM), which generates more natural-looking adversarial images. This involves semantically grouping landmarks and then perturbing the group rather than the individual landmarks. According to the authors, the GFLM method has an exceptionally high success rate and is also extremely difficult to detect.

Chapter 3

Background

In this chapter, we discuss the foundations necessary to understand this thesis. This includes the selection of the dataset, the network architecture, and mainly the original LOTS paper (Rozsa et al., 2017), upon which our thesis is based.

3.1 Dataset

There are many different datasets that contain images of faces: LFW (Huang et al., 2008), MegaFace (Kemelmacher-Shlizerman et al., 2016), CelebA (Liu et al., 2015) or VGGFace2, to name a few. In our thesis, we use images of faces as input for experiments and also to create representation vectors of different identities. The authors of the LOTS paper base their experiments on the VGGFace dataset. We decided to use the VGGFace2 dataset, the successor of VGGFace, in order to stay close to the original LOTS experiments but also to benefit from the newer and larger dataset. Furthermore, we require a dataset that contains images of as many identities as possible with as many images per identity as possible, which the VGGFace2 dataset is able to provide. The VGGFace2 dataset was released by Cao et al. (2018) from the University of Oxford in 2018.

3.1.1 Image Collection

The images of the VGGFace2 dataset originated from the Google Image Search and were collected in a multi-step process. First, a list of 500k public figures was compiled. In manual work, the people were weeded out, which did not have enough different images. This step reduced the list of people to less than 10k. In the next step, images of the remaining people were collected in different poses, perspectives and lighting conditions. Next, the MTCNN face detector (Zhang et al., 2016) was applied to the images, providing corresponding bounding boxes and five facial landmarks. In further steps, outliers, near-duplicates and images, on which no face could be detected, were removed. Finally, the dataset was filtered one more time in an automatic and a manual process to ensure the highest possible quality (Cao et al., 2018).

3.1.2 VGGFace2 Statistics

The VGGFace2 dataset contains 3.31 million images of 9131 different identities. An average of 362.6 images are present per identity, with a minimum of 80 and a maximum of 843 images. The images show the identities in different poses, lighting conditions, at different ages and against different backgrounds. The dataset is more or less balanced regarding gender, with 59.3% males. The authors provide no further information on distributions in other categories. We, therefore, evaluated the MAAD-Face dataset (Terhörst et al., 2021). The MAAD-Face dataset is based on the VGGFace2 dataset but extends it with attributes that can give us some more insight into the data composition. For each image in the VGGFace2 dataset, the MAAD-Face dataset provides a list of attributes with a classification of *attribute is present*, *attribute is absent*, or *attribute is undefined*. The attributes include gender, ethnicity, hair colors and hairstyles, facial hair, and other physical properties. For each attribute, we counted the number of images where the attribute was present and thus made a more detailed evaluation of the data composition. The full list of attributes and their presence in the images of VGGFace2 dataset can be found in the Appendix A.1. The analysis of the data composition is limited by the attributes available in the MAAD-Face dataset. For example, only the attributes *black*, *asian*, and *white* exist in relation to ethnicity; an attribute for *hispanic* or other ethnicities is missing.

Figure 3.1 shows the result of the evaluation in the categories gender, ethnicity and age. In terms of gender, the data is consistent with the VGGFace2 dataset. At 59.2%, there is a slight imbalance in favour of men. In terms of age, we observe that the majority of the images (67%) come from young people. This is followed by the middle-aged group with 19% and the senior group with 14%. The exact boundaries of the different age groups are not mentioned in any of the papers. Regarding ethnicity, the dataset is very unbalanced, although there has been some progress since the release of the VGGFace dataset. The dominant class is *white* with 88.7%, followed by *black* with 6.5% and *asian* with only 4.8%. Since this imbalance in ethnicity may be caused by the fact that there are more example images of people from the *white* group, the ethnicity was additionally analyzed per identity, assuming the majority class to be true in each case. The results slightly improve when looking from the per-identity perspective, but the overwhelming majority remains with the *white* group at 84.5%. The authors of the VGGFace2 dataset explain the difference with a different number of public figures in each group. It should be noted that this analysis is not necessarily 100% correct, as there will be classification errors regarding the MAAD-Face attributes, and in addition, there is also an *attribute is undefined* class.

The VGGFace2 dataset is divided into 8631 training classes and 500 test classes. The images are not constrained regarding size and shape. Samples of the dataset are shown in Figure 3.2. The images are resized but otherwise unedited.

3.2 Network

Nowadays, most image recognition tasks are approached using Convolutional Neural Networks (CNN). More specifically, Residual Networks (ResNet) (He et al., 2016) have proven to be especially useful, as they allow the training of deeper networks without much additional complexity. Deeper networks generalize better and are therefore particularly well suited for image recognition tasks. We use a ResNet with an extension, namely the Squeeze-and-Excitation Blocks introduced by Hu et al. (2018). Cao et al. (2018) show that a ResNet with Squeeze-and-Excitation Blocks (SE-ResNet) perform especially well on the VGGFace2 dataset. We need such a network to extract facial feature vectors from images of people. This allows us to mathematically measure the similarities of faces. In this section we describe the architecture of the SE-ResNet and how we can extract facial feature vectors from it.

3.2.1 Architecture

In principle, images are nothing more than matrices with one numerical value per pixel and colour channel. One could simply flatten such images, as is done in the Eigenface approach (Turk and Pentland, 1991), and feed them into a Multi-Level Perceptron (MLP) network to perform classification or recognition tasks. However, with an MLP, it becomes extremely difficult

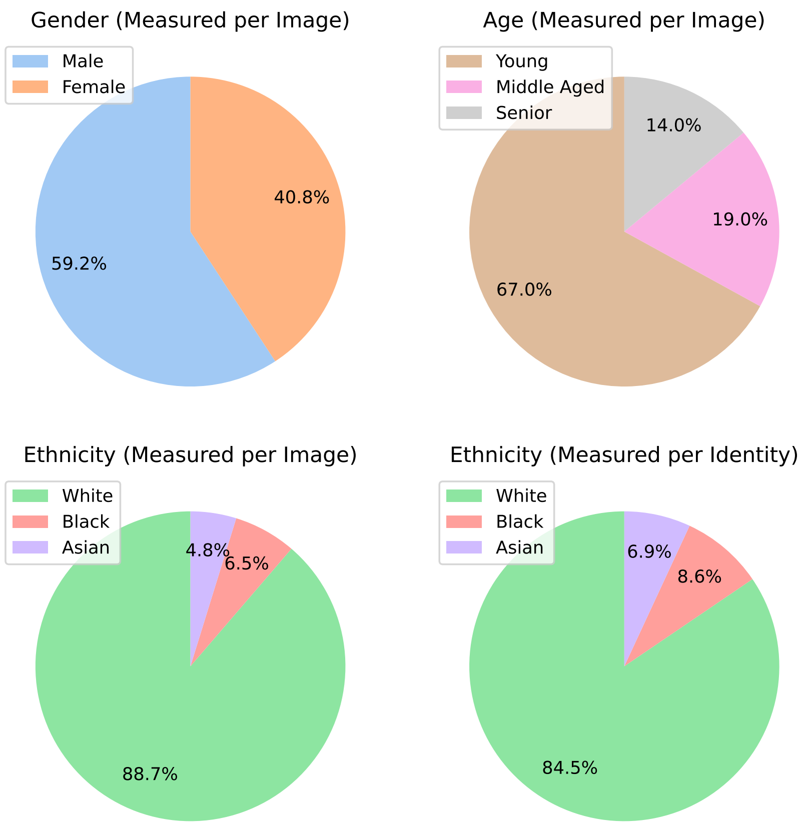


Figure 3.1: VGGFace2 Distributions. Distribution of the VGGFace2 images in different categories, analyzed using MAAD-Face attributes.

to detect spatial dependencies. Therefore, CNNs are mostly used for such tasks. CNN is a deep learning network architecture that allows spatial dependencies to be recognized by applying multidimensional filters to the input. The process of applying filters to the input is called convolution and results in so-called feature maps. It is said that early filters learn edges and shapes, and later filters can recognize whole patterns and objects. Between single convolutions, pooling layers can be integrated. The purpose of pooling layers is to reduce the size of the feature maps. Finally, the feature maps of the CNN can be flattened and fed into fully connected layers to perform classification tasks. When building a CNN, the trade-off between layer width and layer depth usually has to be considered. The consensus is that networks with deeper layers have better generalization capabilities, and networks with wider layers memorize better and are therefore more prone to overfitting. However, it is not possible to simply stack layers on top of each other to get a deeper network, as this would lead to the vanishing gradient problem. The vanishing gradient problem describes a phenomenon wherein backpropagation to later layers can lead to an infinitely small gradient. This leads to over-saturated networks and can also degrade network performance. A network is over-saturated if the majority of the individual neurons only output values close to the asymptotic end of the corresponding activation function (Rakitianskaia and Engelbrecht, 2015).

He et al. (2016) presented Residual Networks, which combat this vanishing gradient problem and thus allow the training of deeper networks. They achieve this by a so-called residual block,

Figure 3.2: VGGFACE2 SAMPLES. Samples of the VGGFace2 dataset. The images are resized but not reshaped.

which passes the unchanged input to the block from shallower to deeper layers.

Such ResNets can be extended with squeeze-and-excitation (SE) blocks (Hu et al., 2018). The idea is to allow the networks to adaptively adjust the weights of the individual channels of the different feature maps. This is done by having a small side network within the residual block that weights the residual according to its internal weights before adding back the identity values. Hu et al. (2018) has shown that SE blocks yield significant performance improvements with minimal additional computational costs.

Cao et al. (2018) evaluated different networks on their VGGFace2 dataset and found that the SE-ResNet-50 performed best among all the networks examined. Additionally, it was pointed out that the network is best trained on MS1M (Guo et al., 2016) and fine-tuned on the VGGFace2 dataset. Therefore we decided to use a SE-ResNet-50[1] analogously to the specifications of Hu et al. (2018), which was trained on the MS1M dataset using a Softmax loss function and later fine-tuned on the VGGFace2 dataset. The SE-ResNet-50 we use does not work with pixels values normalized between 0 and 1 but rather uses the full pixel value range of 0 to 255. When loading an image, the image is transformed from RGB to BGR color space, and the mean values of the VGGFace2 dataset (r=131.0912, g=103.8827, b=91.4953) are subtracted.

3.2.2 Feature Extraction

We use the SE-ResNet-50 to extract facial feature vectors from input images. Figure 3.3 shows the last part of the SE-ResNet-50 network. The output of the last convolution has a size of 7x7 at 2048 channels. Global average pooling reduces this output to a size of 1x1 at 2048 channels. With global average pooling, all values per channel are averaged, which results in the mentioned 2048 channels with one value each. We can interpret this result as a feature vector, which is a member of the $\mathbb{R}^{2048}$ vector space. The entire classification stack, which first flattens the feature vector and connects it to a fully connected layer with 8631 neurons (=number of training identities) and then passes it on to a Softmax layer, is omitted for our thesis. Omitting the classification stack is common practice for face recognition networks. For the thesis, we assume that we attack a face

[1] https://github.com/cydonia999/VGGFace2-pytorch

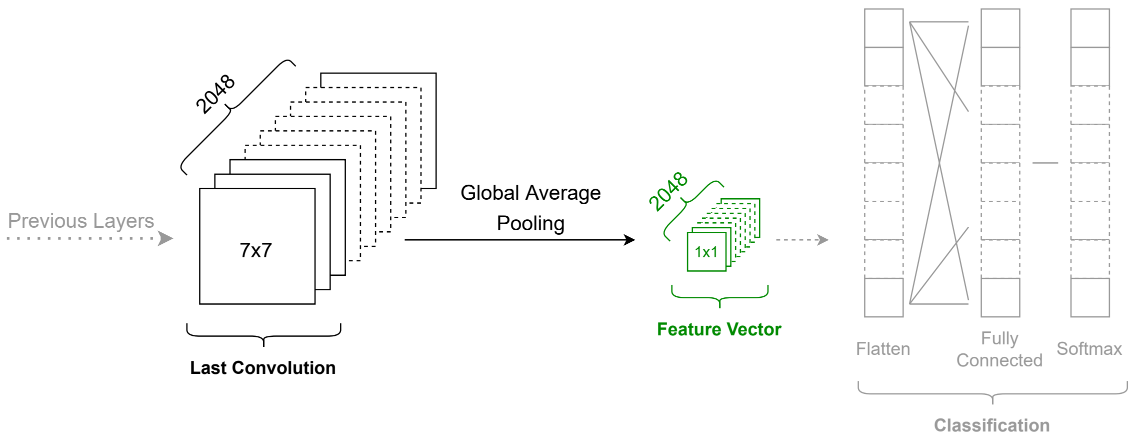


Figure 3.3: EXTRACTING FACIAL FEATURE DESCRIPTORS. Extracting a facial feature descriptor vector form the SE-ResNet-50. The last convolutional layer of the SE-ResNet-50 is reduced using global average pooling which results in the facial feature description vector. The classification stack of the network is omitted.

recognition system that evaluates the similarity of two identities based on the cosine similarity, respectively, the cosine distance between the respective feature vectors.

3.3 Layerwise Origin-Target Synthesis (LOTS)

In this section, we want to discuss "LOTS about Attacking Deep Features", the work of Rozsa et al. (2017), in more detail. The LOTS technique forms the basis for this thesis, and the concept should therefore be understood.

3.3.1 Idea

The authors wanted to investigate the vulnerabilities of networks that use deep features to recognize people. To investigate such networks, they developed LOTS. LOTS stands for Layerwise Origin-Target Synthesis. It describes a technique that perturbs a source image of a person so that a network using deep features misidentifies the features extracted from the image as deep features of another person. Prior to LOTS, adversarial attacks focused on attacking the classification output on end-to-end networks. LOTS was the first adversarial attack that focused on networks using deep features.

3.3.2 Approach

The experiment is based on the VGGFace dataset with 2.6M images of 2’622 different identities. The associated VGGFace network was used as the network under attack. In face recognition tasks, people’s faces are represented as deep features, so-called face descriptors, and compared with already known face descriptors. The authors extract these face descriptors from the VGGFace network at the fully-connected layer seven before ReLU activation. Such face descriptors are vectors and can be compared, for example, using the cosine distance or the Euclidean distance. If

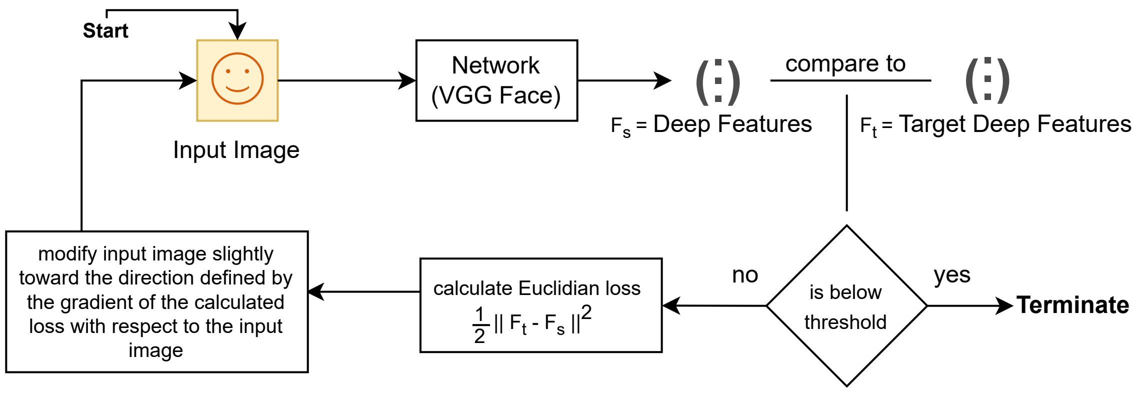

Start
Input Image
Network (VGG Face)
F_s = Deep Features
compare to
F_t = Target Deep Features
is below threshold
yes
Terminate
no
calculate Euclidian loss $\frac{1}{2}|| F_t - F_s ||^2$
modify input image slightly toward the direction defined by the gradient of the calculated loss with respect to the input image

Figure 3.4: LOTS TECHNIQUE. Visualization of the iterative LOTS technique.

the distance of two vectors to be compared is below a particular threshold value, which must be determined in advance, they are considered to be face descriptor vectors of the same person.

In order to compare the extracted face descriptors with a reference, face descriptor vectors of known identities have to be created first. This is done with the help of so-called gallery templates, which are a collection of 50+ images of the same person. From each of the images, the face descriptors are extracted and averaged. The resulting vector is the face descriptor of this identity. They calculate the additional threshold values needed for the cosine and euclidian distance by comparing gallery templates with other sample images from the same and from different identities. The identical process was also used in this thesis and is therefore described in more detail in Section 4.1.4.

Figure 3.4 shows the concept of the LOTS technique. The process starts with an image of any person. This image is fed into the VGGFace network, and the deep features (F_s) are extracted. In the beginning, a target is chosen, which will be mimicked by the deep features extracted from the adversarial image. The extracted deep features (F_s) are now compared with the face descriptor of the target (F_t). Depending on the experiment, they use the cosine distance or the Euclidean distance for comparison. If the value is below the defined threshold value, the process terminates, and the adversarial image, which mimics the deep features of the target, is created. If the value is not below the threshold, the euclidian loss between F_s and F_t is calculated, and the input image is slightly modified toward the direction defined by the gradient of the loss with respect to the image. Normally, gradients are used in combination with an optimizer to adjust the weights of a network such that the result of a forward pass with a specific input is closer to the expected result. However, the authors use the gradient here to change the pixel values of the input image such that, with fixed weights of the network, the extracted features from the input image (F_s) are closer to the target (F_t) they have defined. The whole process is repeated until the comparative value between F_s and F_t is below the threshold value. We explain the synthesis process visualized in Algorithm 1 line by line.

- Line (1): Definition of the LOTS function. The $image_{src}$ is the image to be synthesized into an adversarial image. The $features_t$ are the target features to be mimicked by the adversarial image. The threshold (τ) defines a value below the distance between the features extracted from the adversarial image, and the $features_t$ have to be, in order to be considered a successful attack. The cosine distance or the Euclidean distance can be used as a distance measure.

- (2): We assign the original $image_{src}$ to the variable $image_{adv}$. While applying the algorithm,

Algorithm 1 Original LOTS Algorithm

```
1: function APPLYLOTS($image_{src}$, $features_t$, $\tau$)
2:     $image_{adv}$ = $image_{src}$
3:     while distance($features_t$, model.extract_features($image_{adv}$)) > $\tau$ do
4:         $loss$ = euclidean_loss($image_{adv}$, $features_t$)
5:         $gradient$ = $loss$.backpropagation().get_gradient($image_{adv}$)
6:         $gradient_{step}$ = $gradient/max(abs(gradient))$
7:         $image_{adv}$ = clamp($image_{adv}$ - $gradient_{step}$)
8:     end while
9:     return round($image_{adv}$)
10: end function
```

the variable $image_{adv}$ is synthesized to the final adversarial image, but it corresponds to the original image before the first modification.

- (3): The while condition encloses the modification part of the algorithm and checks whether the distance between $features_t$ and the features extracted from the $image_{adv}$ are below the defined τ value. The distance function can either be the cosine distance or the Euclidean distance.
- (4): The Euclidean loss is calculated between the $features_t$ and the features extracted from $image_{adv}$. We define the Euclidean loss formally in Equation (3.1), where f is a function that extracts features from the image using the model.
- (5): In a backpropagation step, the gradients are calculated. The gradient of the loss with respect to the $image_{adv}$ is assigned to the $gradient$ variable. The gradient is formally defined in Equation (3.2).
- (6): The gradient is scaled by elementwise division by the highest absolute value such that $L_\infty = 1$. The result is assigned to the $gradient_{step}$ variable. The gradient step is formally defined in Equation (3.3).
- (7): The $image_{adv}$ is re-assigned by subtracting the $gradient_{step}$ from $image_{adv}$ and clamping the result. Clamping is the process of restricting the change to be within specified boundaries. The modifications applied to the $image_{adv}$ are formally defined in Equation (3.4).
- Line (8): End of the iterative synthesis process.
- Line (9): Since floating-point numbers are used during the synthesis process, the pixel values must be rounded to integers before returning.
- Line (10): End of the LOTS function.

$$loss_{Euc.}(image_{adv}, features_t) = \frac{1}{2} \parallel features_t - f(image_{adv}) \parallel^2 \tag{3.1}$$

$$gradient(image_{adv}, features_t) = \nabla_{image_{adv}}(loss_{Euc.}(image_{adv}, features_t)) \tag{3.2}$$

$$gradient_{step}(image_{adv}, features_t) = \frac{gradient(image_{adv}, features_t)}{max(abs(gradient(image_{adv}, features_t)))} \tag{3.3}$$

$$image_{adv} = clamp(image_{adv} - gradient_{step}(image_{adv}, features_t)) \tag{3.4}$$

Often, L_p norms are used to evaluate adversarial quality. However, such norms only quantify perturbations but provide little information about the human perception of the adversarial images. Therefore, the authors used a psychometric called perceptual adversarial similarity score (PASS) (Rozsa et al., 2016) to measure the quality of adversarial images. PASS consists of two parts. The first part ensures that the images are aligned in terms of perspective. Then, the structural similarity index measure (SSIM) is calculated. SSIM is a metric that measures image quality degradation based on contrast, structure and luminance. A value of 1 means perfect similarity, and a value of 0 means the images are entirely different from each other. For the experiments, six identities were handpicked from the VGGFace dataset (=internal adversaries). In addition, they selected six identities that were not present in the dataset (=external adversaries). This allows them to evaluate whether such attacks work better on identities the network has already been trained on or whether the network generalizes well and the attack works on all inputs. In total, they performed four experiments. The end-to-end network was attacked once on the softmax layer and once on the feature descriptor layer. Furthermore, in the third and fourth experiment, the feature descriptors were extracted and compared with either Euclidean or cosine distances. For each experiment, all 2'621 identities of the dataset were chosen as targets and attacked using the 12 adversaries.

3.3.3 Findings

The authors show that LOTS works better on systems that extract face descriptors, which are compared using the Euclidean or cosine distance, than on end-to-end systems. On the former systems, the perturbations were less visible. There was no discrepancy between external and internal adversaries in terms of adversarial quality. They further show that certain targets are more challenging to mimic than others. They explain this by the fact that these faces are closer to the average face than others, and thus less visible perturbations are needed to generate an adversarial.

Chapter 4

Extending LOTS

The main scope of the thesis is to extend the LOTS algorithm such that an adversarial image can mimic the facial features of multiple targets simultaneously. The process of extending LOTS is described in this chapter. In addition, we made further optimizations during the development process that reach beyond the multi-target scope.

4.1 Preprocessing

This section describes the preprocessing applied to all the images, such that the network can extract facial features belonging to individual identities. The first section covers the details of the sample and gallery-template image selection process. Next, we show how the images are cropped, followed by the feature extraction process. Lastly, we explain the methodology used to calculate classification threshold values.

4.1.1 Image Selection

Section 3.1 introduced the VGGFace2 dataset, on which the feature extraction network was fine-tuned on. We use the same images to conduct the experiments and to evaluate the performance of the extended LOTS algorithm. A face recognition model extracts a feature vector from an input image of a person and compares the value to known representation vectors of other identities. If the distance between the two vectors under comparison is below a defined threshold value, the face recognition model classifies the two identities as the same. In order to later conduct experiments and generate adversarial images, we need vectors that represent the averaged facial features of a subject. Rozsa et al. (2017) created gallery templates for each identity, which is a collection of images containing the subject. For each image of the gallery template, feature vectors are extracted and averaged. The averaged vector is the resulting facial representation vector of the identity. We create gallery templates and not just use a single image of an identity because different angles, perspectives, rotations, or exposures will affect the feature extraction. Using multiple images to represent a person makes the network more robust and the person's representation more accurate. For each identity, we select additional probe samples. We convert these images to feature vectors by feeding them through the network. These images and their feature vectors can then be used to calculate identification threshold values and serve as experiment input.

For each identity of the test set, we randomly sampled 50 images that compose the gallery template. Furthermore, we randomly sampled 155 additional probe images, which serve as input images and are used for threshold value calculations. Since not every identity provides the same number of possible probe images, we chose the number of images to be sampled so that we can

Algorithm 2 Scaling and cropping of one single image

```
 1: $image$ = load image from disk
 2: $bbx$ = load bounding box from disk
 3: $width\_scale$ = multiply width of $bbx$ by $0.3$
 4: $height\_scale$ = multiply height of $bbx$ by $0.3$
 5: if $width\_scale > height\_scale$ then
 6:     $scale\_px\_diff = width\_scale$
 7: else
 8:     $scale\_px\_diff = height\_scale$
 9: end if
10: $area$ = translate $bbx$ from [x, y, width, height] to area (left, upper, right, lower)
11: $area$ = Extend $area$ by subtracting $scale\_px\_diff$ from $area$ left and upper coordinates and
    by adding $scale\_px\_diff$ to $area$ right and lower coordinates
12: crop image to $area$
13: resize smaller side of $image$ to 256px
14: center crop $image$ to 224px x 224px
```

retain most identities while having the largest number of probe images for threshold calculations. For each identity of the training set, we randomly sampled 50 images that compose the gallery template. We do not select any probe images for identities of the training set since we never use identities the network was trained on as inputs in experiments, and we do not consider them in the threshold value calculations. We only use the identities of the training set as targets in experiments, for which the gallery templates are sufficient. Not all 9131 identities in the VGGFace2 dataset have sufficient images to compose gallery templates and still provide sample images. We only consider the remaining 9084 identities having enough images for the sampling process.

4.1.2 Face Cropping

Since the images from the VGGFace2 dataset come in any form and shape, we have to crop them to a uniform size such that we can use them as input for the network. Cao et al. (2018) state that to preprocess the images, they used the MTCNN face detector (Zhang et al., 2016) to extract a bounding box, which they extended by a factor of 0.3. They then crop to the extended bounding box and resize the shorter side to 256 pixels. A 224 x 224 center crop is taken from the resulting image, which yields the final input image. The dataset contains a file with the bounding boxes for all the images. Unfortunately, scaling by a factor of 1.3 can be interpreted in several ways, and more detailed specifications could neither be found in the paper nor the code. However, we found some sample images and preprocessing code on the original GitHub[1] repository. We applied their preprocessing to the sample images, which yielded images we could use to evaluate our cropping method. To avoid future confusion, we explain our cropping process in Algorithm 2.

We save the images as PNG files to avoid issues with JPG compression. Figure 4.1 shows some examples of images after we have cropped them. Depending on the subject's size in the original image and the original image's resolution, the resulting image's quality ranges from blurry to sharp. The face cropping was implemented using multi-processing.

[1] `https://github.com/ox-vgg/vgg_face2`

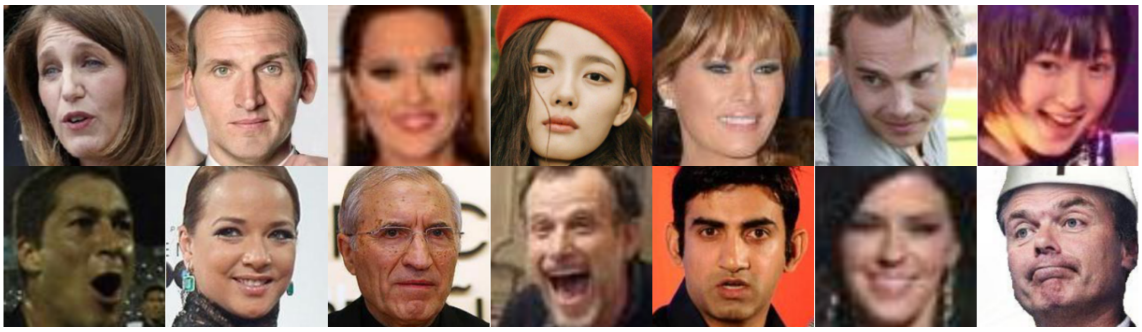


Figure 4.1: CROPPED IMAGE SAMPLES. Examples of images cropped using the MTCNN face detector and applying the face cropping algorithm.

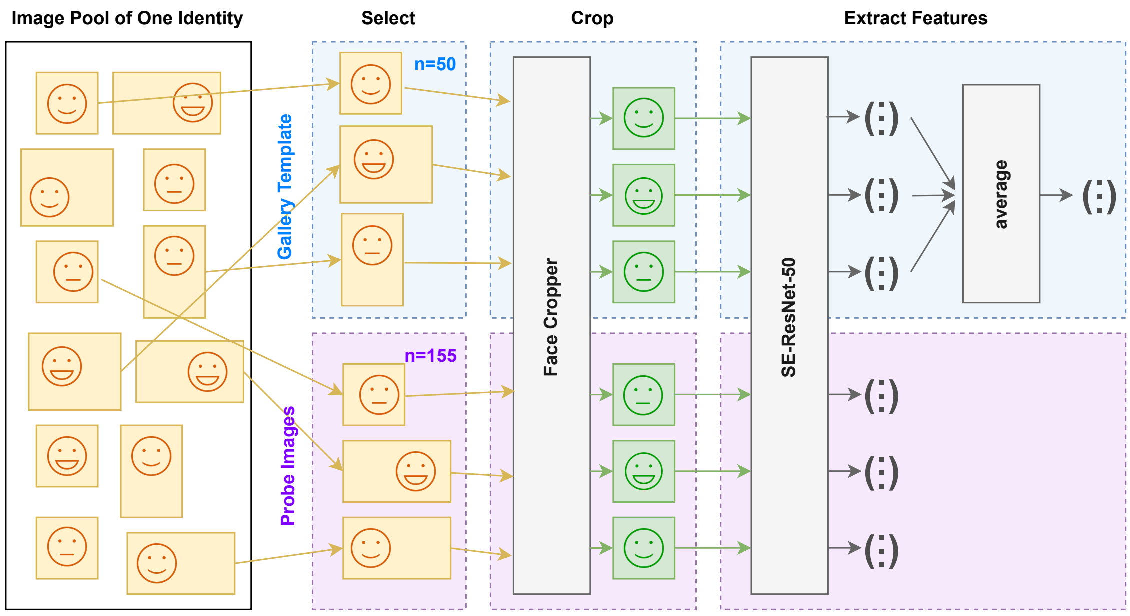


Figure 4.2: PREPROCESSING PIPELINE. Preprocessing pipeline visualized for one identity. Includes the image selection for the gallery template and sample groups, cropping of the images and feature extraction using the SE-ResNet-50.

4.1.3 Feature Extraction

Face recognition models need facial feature vectors to compare different identities. In order to extract such vectors, we pass the previously cropped images with size 224 x 224 pixels into a network, in which the classification layer is omitted (Section 3.2.2). For all the images of an identity that compose the gallery template, we extract the feature vectors, average them, and save them as identity representation feature vectors. We extract the feature vectors and save them without further computation for all sample images of each identity. Features are extracted batch-wise and all vectors are saved as NumPy arrays. The whole process from image selection to feature extraction is depicted in Figure 4.2.

4.1.4 Threshold Calculation

Face recognition models usually use the Euclidean distance or the cosine similarity to determine whether two feature vectors represent the same identity. Since cosine similarity tends to perform better on face recognition tasks (Li and Zhu, 2016), we use the cosine similarity respectively the cosine distance in our approach. The cosine similarity between two vectors is calculated according to Equation (4.1), and the derived cosine distance between two vectors is shown in Equation (4.2). In our case, the cosine distance can take on values between 0 and 1 since the feature vectors themselves only contain positive values. A cosine distance of 0 means that the two vectors are identical, and a distance of 1 implies that the vectors are perpendicular to each other and, therefore, very different.

$$cosine\ similarity = S_c(\mathbf{A}, \mathbf{B}) = cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\parallel \mathbf{A} \parallel \parallel \mathbf{B} \parallel} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{4.1}$$

$$cosine\ distance = D_c(\mathbf{A}, \mathbf{B}) = 1 - S_c(\mathbf{A}, \mathbf{B}) \tag{4.2}$$

We want to find a cosine distance threshold for our model for which we can confidently say that if a calculated distance between two input vectors lies below this threshold, they represent the same person. To find this threshold, we define a False Match Rate (FMR) that we want to aim for. The FMR defines how many inputs we are allowed to misidentify as the same person, even though it is a different person. With an FMR of 0.001, one out of 1000 comparisons of two different faces may be incorrectly identified as belonging to the same person.

After that, we calculate positive and negative distance scores. To create a positive distance score, we calculate the cosine distance between a person's gallery template and a feature vector extracted from a probe image of the same person. For a negative distance score, we compute the cosine distance between the gallery template of one person and a feature vector extracted from a probe image of a different person. We use only the identities from the test data set and create the maximum number of positive and negative distance scores for each identity. The process is depicted in Figure 4.3. With a number of 453 identities, each with a gallery template and 155 sample feature vectors, this results in 70'215 positive and 31'737'180 negative distance scores.

Once we have generated the positive and negative distance scores, we sort the list of negative distance scores in ascending order. We then calculate an index using Equation (4.3). We access the list of negative distance scores with the calculated index, which yields the cosine distance threshold for the desired FMR. The reason for this is that with, for example, 1'000 negative distance scores sorted in ascending order and a FMR of 0.1, the decisive element is located at position 100 (1'000 * 0.1). Accessing the element at position 100 means that we exactly reached a FMR of 0.1 since we let the first 100 elements pass, even though the distance scores were calculated by comparing faces of different identities. The calculated position is rounded down. Next, we can calculate the corresponding True Match Rate (TMR). We select all false non-matches, which are positive distance scores (same person) with a cosine distance above the calculated cosine distance threshold. We can then calculate the TMR using Equation (4.4).

$$index_{cosThreshold} = \lfloor length(negative\ distance\ scores) \cdot FMR \rfloor \tag{4.3}$$

$$TMR = 1 - \frac{length(false\ non\text{-}matches)}{length(positive\ distance\ scores)} \tag{4.4}$$

We have calculated the cosine distance threshold values and TMRs for multiple FMRs ranging from 10^{-6} to 1. The results of these calculations are shown in Table 4.1.

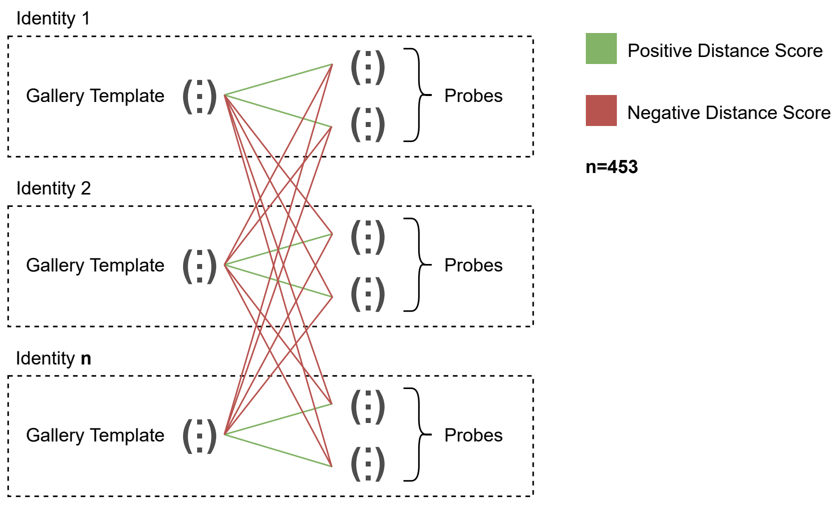


Figure 4.3: THRESHOLD CALCULATION. Visualisation of the process for creating positive and negative distance scores using gallery templates and sample feature vectors.

FMR	1.0	0.1	0.01	0.001	0.0001	0.00001	0.000001
Cos. Dist. Threshold @FMR	0.933	0.583	0.475	0.392	0.326	0.249	0.104
TMR @FMR	1.000	0.997	0.984	0.961	0.920	0.792	0.146

Table 4.1: COSINE DISTANCE THRESHOLD AT FALSE MATCH RATE. This table shows the cosine distance threshold and True Match Rate (TMR) at different False Match Rates (FMR).

In general, one can observe that for FMRs ranging from 10^{-4} to 1, the model remains quite accurate, with TMRs of over 90%. However, the cosine distance thresholds decrease rapidly. At a FMR of 10^{-5}, the model has a TMR slightly below 80%. At an FMR of 10^{-6} we are down to a TMR of just under 15%. Of course, it depends on the application which FMR-TMR combination is chosen. For cases in which false matches would be disastrous, a model with FMR 10^{-5} would be recommended. For less strict applications, FMR 10^{-3} or 10^{-4} would be valid options. Figure 4.4 visualizes the FMR-TMR trade-off. As these are computationally intensive tasks, the calculation of thresholds was implemented using multi-processing.

4.2 Mimicking Multiple Targets Simultaneously

The original LOTS technique can modify a single input image to mimic the facial features of an arbitrary target. However, the target must have a gallery template available. In an iterative process, LOTS modifies the image until the Euclidean distance between the target's gallery template and the facial features extracted from the perturbed input image falls below a defined threshold (Rozsa et al., 2017). We extend the LOTS algorithm and allow multiple targets to be mimicked simultaneously using the same input image. Unlike the original LOTS technique, we use cosine distance instead of Euclidean distance to compare the similarity of different facial feature vectors. Both methods would be suitable to measure the difference between two vectors, but since we assume a model using the cosine distance, we also use it within the algorithm.

In order to attack multiple targets simultaneously, the gallery templates of all targets are

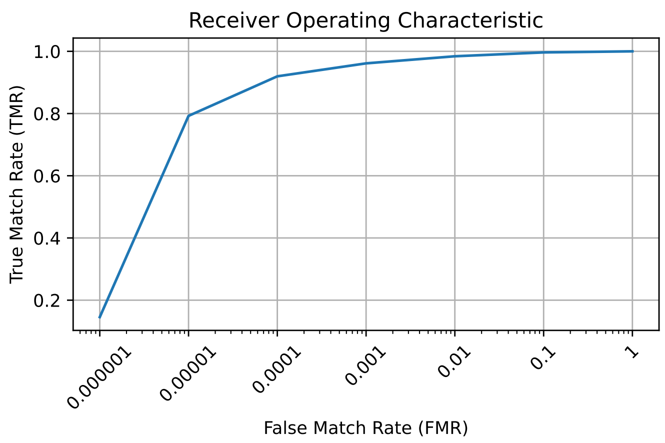


Figure 4.4: RECEIVER OPERATING CHARACTERISTIC. Shows the True Match Rate (TMR) at different False Match Rates (FMR).

loaded first. After each iteration, we pairwise check if the cosine distance between the facial features extracted from the modified input image and the individual targets is below the defined threshold. For a successful attack on multiple targets, all distances must be below the threshold at the same time, not just on average. The success criterion is defined in Equation (4.5). For all t, where t represents a single target from the list of $targets$, the following condition must be true: The cosine distance (D_c, Equation (4.2)) between the function f, which extracts a facial feature vector from an input image ($image_{adv}$), and the gallery template of the target t ($GalleryTemplate_t$), must be below the defined cosine distance threshold ($\tau_{cosDist}$).

If this is not the case after 5000 iterations, the attack attempt is automatically stopped since the perturbations are very clearly visible in this case.

$$\forall t \in targets, (D_c(f(image_{adv}), GalleryTemplate_t) < \tau_{cosDist}) \tag{4.5}$$

4.2.1 Including Source

Another important addition to the algorithm is including the source as a target. In the original LOTS, the input image is perturbed to mimic the facial features of the specified target. No consideration is given to the fact that the input image continues to mimic the facial features of the source. In a use case where, for example, two people want to share the same passport, the cosine distance of the facial features extracted from the adversarial image to both, the gallery template of the source and the gallery template of the target, must be below the defined cosine distance threshold value. The implementation is simple. The gallery template vector of the source is added to the list of targets. The algorithm does not need to be changed. However, including the source in the list of targets could lead to problems. The algorithm tries to change the input image so that the cosine distances between extracted facial features and the gallery templates of the targets become smaller. By including the source as a further target, the newly introduced component tries to constrain change and keep the extracted features close to the original. During the synthesis of the adversarial image, we ensure that the cosine distances between the source and the targets are below the threshold simultaneously. Including the source as a target effectively constrains deviation from the initially extracted feature vector and adds one additional target, which might worsen the algorithm's performance slightly.

4.3 Loss Function Adaptions

The loss function of the algorithm mainly determines how strong the perturbations are and how they manifest themselves. This section shows how we have adapted the loss function and introduces two new loss components to minimize perturbations further.

4.3.1 Multi-Component Loss

For our work, we will introduce a multi-component loss. It consists of a component that measures the cosine distance between all targets and the features extracted from the adversarial image ($loss_{cos}$) and two components that measure the similarity between the adversarial image and the source image ($loss_{ssim}$, $loss_{msssim}$). In order to combine the different components, they are multiplied by a specific weight and then added together to form the total loss. The multi-component loss is calculated according to Equation (4.6), whereas we explain the individual loss components in the following sections.

$$loss_{total} = \quad w_{cos} \cdot loss_{cos} + w_{ssim} \cdot loss_{ssim} + w_{msssim} \cdot loss_{msssim} \tag{4.6}$$

Introducing weights allows us to control how much influence each component has. The extended LOTS algorithm's main goal is to reduce the cosine distance between all attacked targets and the source below a defined threshold value. Adding the complementary similarity loss components could lead to conflicting goals if lowering the cosine distances is not compatible with maintaining the structure using similarity measures. By adjusting the weighting of the similarity measures, the achievement of a low distance can be prioritized again. Figure 4.5 shows how the different loss components influence the quality of the adversarial image. Each column depicts a modification of the loss function where the image in the first row shows the perturbed image, the second row shows the isolated perturbations, and the third row shows the perturbations scaled with a factor of 3 for better visibility. The result of combining all loss components is shown in the fifth column.

4.3.2 Cosine Distance Loss

The original LOTS technique calculates the Euclidean distance between the facial feature vectors extracted from the network and the gallery template of the target and uses this as the loss. An alternative to the Euclidean distance loss would be the cosine similarity/distance. Both metrics are common and valid solutions to assess the similarity of two vectors. The Euclidean distance measures how close two points are in the vector space, whereas cosine similarity measures the angular distance between two vectors. We have chosen the cosine distance as a loss component because we also assume a model that compares input and target using cosine distances.

$$loss_{cos} = \frac{1}{N} \sum_{n=1}^{N} (D_c(features_{curr}, target_n))^2 \tag{4.7}$$

Equation (4.7) shows how we calculate the loss in a multi-target scenario. The cosine distances between the facial features extracted from the input image of the current iteration ($features_{curr}$) and the facial features of the targets is calculated and squared. We experimented with a cosine distance loss that was not squared, with weighted distances, and with only including targets

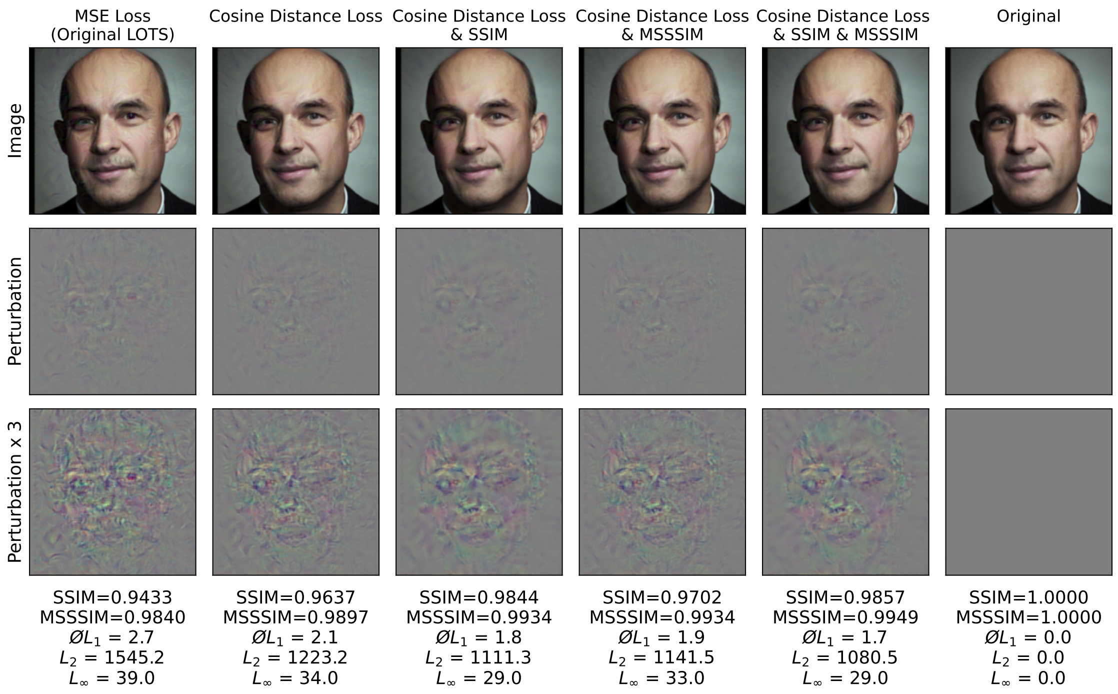

MSE Loss (Original LOTS)
Cosine Distance Loss
Cosine Distance Loss & SSIM
Cosine Distance Loss & MSSSIM
Cosine Distance Loss & SSIM & MSSSIM
Original
Image
Perturbation
Perturbation x 3
SSIM=0.9433 MSSSIM=0.9840 $\varnothing L_1$ = 2.7 L_2 = 1545.2 L_∞ = 39.0
SSIM=0.9637 MSSSIM=0.9897 $\varnothing L_1$ = 2.1 L_2 = 1223.2 L_∞ = 34.0
SSIM=0.9844 MSSSIM=0.9934 $\varnothing L_1$ = 1.8 L_2 = 1111.3 L_∞ = 29.0
SSIM=0.9702 MSSSIM=0.9934 $\varnothing L_1$ = 1.9 L_2 = 1141.5 L_∞ = 33.0
SSIM=0.9857 MSSSIM=0.9949 $\varnothing L_1$ = 1.7 L_2 = 1080.5 L_∞ = 29.0
SSIM=1.0000 MSSSIM=1.0000 $\varnothing L_1$ = 0.0 L_2 = 0.0 L_∞ = 0.0

Figure 4.5: INFLUENCE OF LOSS FUNCTION ON ADVERSARIAL IMAGE. Influence of different loss functions on the manifestation of perturbations. Each column represents an experiment, whereas the first row represents the perturbed image, the second row the isolated perturbations, and the third row the isolated perturbations scaled by a factor of 3 for better visibility. Below each column, we show similarity scores and different L_p norms that express the perturbations. The last column shows the original image without perturbations as a reference. Experiment description per column: (1) MSE loss only, (2) cosine distance loss only, (3) cosine distance loss & $w_{ssim} = 1.0, w_{msssim} = 0.0$ (4) cosine distance loss & $w_{ssim} = 0.0, w_{msssim} = 1.0$, (5) cosine distance loss & $w_{ssim} = w_{msssim} = 1.0$, (6) original image.

above the threshold in the loss component. We found that squaring the cosine distance led to the fastest conversion and best success rate since it emphasizes the targets farthest away from the input in the cosine distance loss component. The squared cosine distances are summed up and divided by the number of targets. In the context of this thesis, we refer to the average squared cosine distance loss between the source and all targets when we mention the cosine distance loss.

Comparing the first and the second column in Figure 4.5 shows the improvement when changing the loss from Euclidean distance, as it was used in the original LOTs technique, to the cosine distance. The change results in an improvement that manifests in weaker perturbations. In addition to the visual observation of the weaker perturbations, the similarity metrics SSIM and MSSSIM of the two loss functions can also be compared. For both metrics, the cosine distance loss achieves significantly higher values. We apply the extended LOTS technique using the same source image and targeting three different targets (without including the source) and using FMR=0.001 in all columns. The last column shows the original image without any perturbations as a reference.

Different L_p norms can also quantify the perturbations. The L_1 norm of the perturbation vector is defined as the sum of all value changes in all color channels of all pixels. From the L_1 norm, we can derive the $\varnothing L_1$ norm, which measures the per-image average change of each color channel value of a pixel. It is calculated by dividing the L_1 norm by the image size (224x224) and the number of color channels (3). The L_2 norm measures the Euclidean distance of the perturbation vector and can be interpreted as a metric to measure the magnitude of the perturbation. The L_∞ norm indicates the largest change of a single value in the color channels of a pixel. Also, in all L_p norms, a clear improvement can be seen if we use the cosine distance loss instead of the Euclidean distance loss.

4.3.3 Structural Similarity Index (SSIM)

The structural similarity index (SSIM) is a widely used statistical measure to assess the quality of an image. Wang et al. (2004) introduced it to measure the degradation of structural information in images. For this purpose, two images' luminance, contrast, and structural components are compared. A score between 1 (=same image) and 0 (=different image) is assigned to a pair of images. The authors have not defined the color space of SSIM but mention that adding color components does not significantly affect the index's performance. The index is applied to color images in many applications by converting a color image to a grayscale image. Examples of a conversion would be the Rec. 601 color encoding standard or a conversion from RGB into the YCrCb color space (Nilsson and Akenine-Möller, 2020).

Nilsson and Akenine-Möller (2020) have investigated the SSIM metric and found that it can lead to unexpected results under certain circumstances. For example, since SSIM compares the luminance values and not the colors directly, different colors with the same luminance value can lead to an SSIM score of 1, even though a human can see the color differences. They also show that minimal changes that a human cannot detect can sometimes lead to very low SSIM scores. The authors argue that the SSIM score cannot be used as a perfect measure for substituting human perception under these circumstances. They also advise against using the SSIM loss in Deep Learning models, as these infrequent problems can lead to biases in the learning process.

We can still use the SSIM score for our use case because we are not directly training a Deep Learning model but using an iterative algorithm to perturb an image. Using it, we can constrain the magnitude of the perturbations and ensure that the structure does not change significantly. The idea of including an SSIM loss component is that we keep the adversarial image as close as possible to the original image in terms of structure.

We calculate the SSIM loss component according to Equation (4.8).

$$loss_{ssim} = 1 - ssim(image_{src}, image_{adv}) \tag{4.8}$$

We compute the SSIM score between the original, unmodified image ($image_{src}$) and the adversarial image of the current iteration ($image_{adv}$). This SSIM value is then subtracted from 1, yielding our structural similarity loss component. The function $ssim$ denotes a PyTorch implementation of the SSIM from GitHub.[2] As we can observe in Figure 4.5, the strength of the perturbations decreases when we add the SSIM loss (third column) in addition to the cosine distance loss (second column). This is also reflected in the similarity scores and the L_p norms.

4.3.4 Multi-Scale Structural Similarity Index (MSSSIM)

Wang et al. (2003) introduced the Multi-Scale Structural Similarity Index (MSSSIM), which builds upon the SSIM. They designed the approach to consider factors such as distance between the observer and the image plane, perceptual capabilities of the observer, and sampling density of the image signal. The MSSSIM iterates and calculates contrast, structure, and luminance components on different scales. In each scale, the contrast and structure components are calculated

[2] https://github.com/VainF/pytorch-msssim

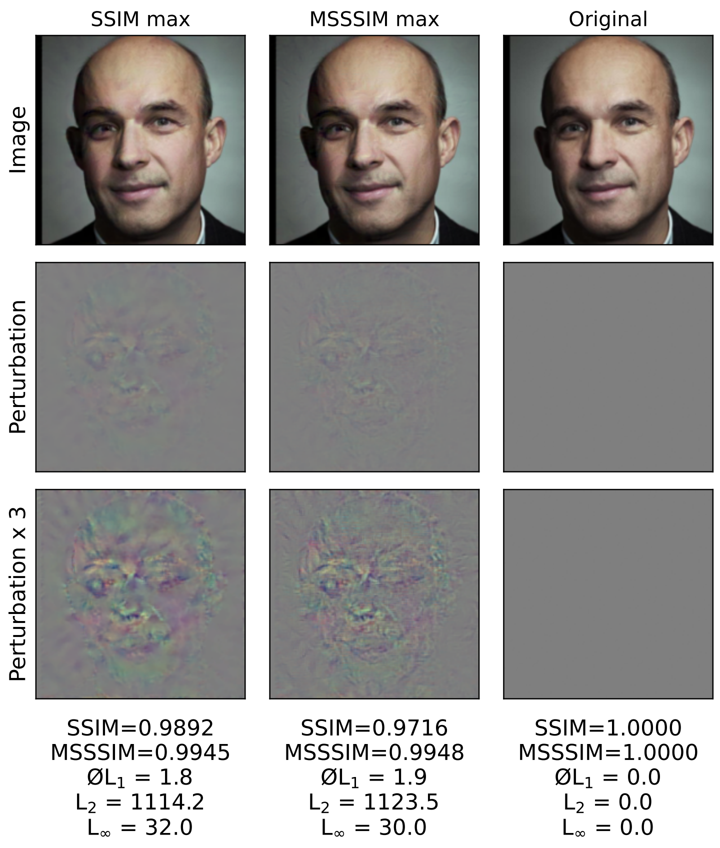


Figure 4.6: INFLUENCE OF SSIM AND MSSSIM ON ADVERSARIAL IMAGE. Influence of the SSIM and MSSSIM metrics as loss components. In order to emphasize their effect, the loss components were overweighted. Experiment description per column: (1) cosine distance loss & $w_{ssim} = 2.0, w_{msssim} = 0.0$, (2) cosine distance loss & $w_{ssim} = 0.0, w_{msssim} = 2.0$, (3) original image.

analogously to the SSIM, followed by low-pass filtering and downsampling with a factor of two, resulting in the next scale. The low-pass filtering and downsampling are performed five times, multiplying corresponding results. Only on the last scale the luminance component is calculated and multiplied to the product of the structure and contrast values of preceding scales. We also want to extend our loss with an MSSSIM component by proceeding based on Equation (4.9).

$$loss_{msssim} = 1 - msssim(image_{src}, image_{adv}) \tag{4.9}$$

The $msssim$ function also comes from the PyTorch implementation on GitHub[2]. Analogous to the SSIM loss, the MSSSIM between the original image ($image_{src}$) and the adversarial image of the current iteration ($image_{adv}$) is calculated and subtracted from 1, resulting in the MSSSIM loss component. Compared to the loss that only considers the cosine distance, a small improvement can also be seen when the MSSSIM loss component is considered. Compared to the cosine distance loss with an SSIM component, there are not fewer visible perturbations, but they seem to manifest themselves in a slightly different way (Figure 4.5). We can also see this in the similar $\varnothing L_1$ and L_2 norm values.

4.3.5 Comparing SSIM to MSSSIM

To compare the SSIM and MSSSIM loss components, we applied the algorithm to the same sample images as in Figure 4.5 but weighted the loss components much more heavily. We chose $w_{ssim} = 2.0, w_{msssim} = 0.0$ and $w_{ssim} = 0.0, w_{msssim} = 2.0$ for the respective experiments. This results in better visibility of the manifestation of the perturbations. In Figure 4.6, we can observe that the SSIM loss focuses on changing surfaces and the MSSSIM loss tends to change the edges. To support this claim, we manually compared 100 adversarial images, generated once with $w_{ssim} = 2.0, w_{msssim} = 0.0$, and once with $w_{ssim} = 0.0, w_{msssim} = 2.0$, and observed the same behavior. However, it is difficult to say whether one loss component is more suitable than the other to control the amount of perturbation applied to the adversarial image. The L_p norms also do not show any major deviations. We decided to use both similarity loss components, because we cannot estimate which component has which influence and we can later override the influence with a weighting.

4.4 Extended LOTS Algorithm

In this section, we explain how we constructed the extended LOTS algorithm. For this purpose, we explain the individual steps from Algorithm 3 line by line.

- Line (1): Definition of the LOTS function. The $image_{src}$ is the image to be synthesized into an adversarial image. The $features_t$ variable contains the gallery templates of identities to be attacked. If the source is to be included in the synthesis, its gallery template must also be added to the $features_t$ list. The τ_{cos} variable contains the cosine distance threshold, below which two feature vectors to be compared are assumed to be of the same identity. The step width variable ($width_{step}$) determines the maximal value a pixel of the adversarial image is changed during an iteration. The variables w_{cos}, w_{ssim}, and w_{msssim} are the weights of the respective loss components.
- (2): We assign the original $image_{src}$ to the variable $image_{adv}$. While applying the algorithm, the variable $image_{adv}$ is synthesized to the final adversarial image, but it corresponds to the original image before the first modification.
- (3): The while condition encloses the modification part of the algorithm, where a maximum number of modification iterations is defined.
- (4): The features from the current adversarial image ($image_{adv}$) are extracted (Section 3.2.2) from the model (SE-ResNet-50) and assigned to the variable called $features_{src}$.
- (5)-(8): We define the variable $cos\_distances$ as an empty list in which different cosine distances between source and targets are stored. For each target t in the list $features_t$, we calculate the cosine distance between the $features_{src}$ of the current adversarial image extracted in line (4) and the target t. The cosine distances are squared and added to the $cos\_distances$ list.
- (9)-(11): We check whether all elements from the $cos\_distances$ list are below the defined cosine τ_{cos} value. If so, the $image_{adv}$ tensor is transformed back into an image. The mean values of the VGGFace2 dataset are added back to the tensor and the resulting adversarial image is returned from the function. The synthesis was successful.
- (12): We compute an SSIM score between the current adversarial image ($image_{adv}$) and the original, unmodified source image ($image_{src}$), subtract it from 1, and assign it to the variable $loss_{ssim}$.

- (13): We compute an MSSSIM score between the current adversarial image ($image_{adv}$) and the original, unmodified source image ($image_{src}$), subtract it from 1, and assign it to the variable $loss_{msssim}$.
- (14): We calculate the $loss_{total}$ of the current iteration by summing up the squared cosine distances, dividing them by the number of elements, and multiplying them by the weight (w_{cos}). The product of $loss_{ssim}$ and weight (w_{ssim}) and the product of $loss_{msssim}$ and weight (w_{msssim}) are added to the $loss_{total}$.
- (15): In a backpropagation step, the gradients are calculated. The gradient of the $loss_{total}$ with respect to the $image_{adv}$ is assigned to the $gradient$ variable.
- (16): This calculation scales the $gradient$ and ensures that the largest value within the $gradient_{step}$ tensor exactly equals $width_{step}$. The value of each color channel of a pixel is therefore changed by a maximum of $width_{step}$ per iteration. The calculations are performed elementwise.
- (17): We subtract the $gradient_{step}$ from the $image_{adv}$, which means we change the pixel values such that the extracted features from the new image (c) are closer to the $features_t$.
- (18)-(20): Clamping is the process of restricting the change to be within specified boundaries. Since the mean values of the VGGFace2 dataset (r=131.0912, g=103.8827, b=91.4953) are subtracted from the corresponding channels when loading an image, the clamping range must also be adjusted. The range no longer spans from 0 to 255 but from $-x$ to $255 - x$, where x is the respective mean of a color channel. Line (18) clamps to the blue channel, line (19) the green channel, and line (20) the red channel.
- (21): The individual color channels are merged and assigned to the variable $image_{adv}$.
- (22): End of the iterative synthesis process.
- (23): If the synthesis failed to generate an adversarial image with all cosine distances below the threshold within the maximum number of iterations, the attack was unsuccessful, and an error message returned.
- (24): End of the LOTS function.

The actual implementation uses PyTorch and differs slightly from Algorithm 3, but the functionality is exactly as described. The whole algorithm is packaged in a Python class where basic properties like the cosine distance threshold, loss component weights, the step width, and various file paths can be set. Also, the MSE loss can be used analogously to the original LOTS paper for comparison purposes. The LOTS algorithm can then be applied in a single run, where one adversarial image is synthesized, saving statistics and a comparison image between the adversarial and original image as well as the perturbations themselves. A multi-run is also possible, in which the list of targets is extended in an iterative process, allowing good visualization of how the number of targets influences the magnitude of the perturbation. The single and multi-runs are mainly available for experimentation or if fine-tuning and optimizing a single adversarial image is the goal. The algorithm can also be used directly without any statistics and outputs.

Algorithm 3 Extended LOTS Algorithm

```
function APPLYLOTS($image_{src}$, $features_t$, $\tau_{cos}$, $width_{step}$, $w_{cos}$, $w_{ssim}$, $w_{msssim}$)
    $image_{adv}$ = $image_{src}$
    while maximum iterations not reached do
        $features_{src}$ = model.extract_features($image_{adv}$)
        $cos\_distances$ = []
        for $t$ in $features_t$ do
            $cos\_distances$.append(cosine_distance($features_{src}$, $t$)$^2$)
        end for
        if all elements of $cos\_distances$ < $\tau_{cos}$ then
            return convert_tensor_to_image($image_{adv}$)
        end if
        $loss_{ssim}$ = 1 - ssim($image_{adv}$, $image_{src}$)
        $loss_{msssim}$ = 1 - msssim($image_{adv}$, $image_{src}$)
        $loss_{total} = w_{cos} \cdot \frac{sum(cos\_distances)}{len(cos\_distances)} + w_{ssim} \cdot loss_{ssim} + w_{msssim} \cdot loss_{msssim}$
        $gradient$ = $loss_{total}$.backpropagation().get_gradient($image_{adv}$)
        $gradient_{step}$ = $gradient \cdot (width_{step}/max(abs(gradient)))$
        $c$ = $image_{adv}$ - $gradient_{step}$
        $c_b$ = clamp($c[0], -91.4953, 255.0 - 91.4953$)
        $c_g$ = clamp($c[1], -103.8827, 255.0 - 103.8827$)
        $c_r$ = clamp($c[2], -131.0912, 255.0 - 131.0912$)
        $image_{adv}$ = cat($c_b$, $c_g$, $c_r$)
    end while
    return Error("unsuccessful synthesis")
end function
```

Chapter 5

Quantitative Experimentation with Extended LOTS

Chapter 4 showed the approach we used to extend the LOTS algorithm. In this chapter, we show how we quantitatively evaluate the algorithm using different experiments.

5.1 Finding Hyperparameters

In Chapter 4, we have defined and used variables such as the step width ($width_{step}$) or the individual weights for the different loss components. However, we still have to find appropriate values for these variables. For example, the similarity measures should not be weighted too highly; otherwise, they may have a more substantial effect than the cosine distance loss components. The algorithm would never achieve the primary goal of bringing all cosine distances below a certain threshold. The choice of $width_{step}$ influences how many iterations the algorithm needs to generate an adversarial image, but at the same time also how strong the perturbations are. No combination of values works equally well within all examples exposed. Nevertheless, we proceeded as described next to find a combination of values that works well for most of the samples the algorithm is exposed to. First, we manually experimented to identify a range of values that produced more or less valuable results. Then we defined 50 samples, randomly selecting the source image and four targets each. We defined a set of possible values for the variables $width_{step}$, w_{ssim}, and w_{msssim}. The weight of the cosine loss component (w_{cos}) was assumed to be 1, since the variation of w_{ssim} or w_{msssim} results in a corresponding weight ratio between the similarity loss components and the cosine loss component. The values that the hyperparameters could assume in the first run are shown in the first row of Table 5.1. A total of 144 combinations were possible, for each we generated adversarial images on the 50 random samples. Thus, 7200 adversarial images were generated in the first run, and the corresponding SSIM scores, MSSSIM scores, and success rates were stored.

In order to evaluate the 7200 experiments, we grouped them according to the hyperparameters. We calculated the average values for the SSIM scores, the MSSSIM scores, and the number of iterations needed to generate the adversarial image for each combination of hyperparameters. Furthermore, we calculated the success rate per hyperparameter group. In a manual step, these values were used to define a new set of hyperparameters, shown in the second row of Table 5.1. We tested the new 45 hyperparameter combinations in a second run, resulting in 2250 experiments.

For each hyperparameter combination i, we again calculated the average SSIM scores ($ssim\_mean_i$), MSSSIM scores ($msssim\_mean_i$), iterations per group ($iterations_i$), and the suc-

Run	$width_{step}$	w_{cos}	w_{ssim}	w_{msssim}	Comb.	Exp.
1	[1, 2, 3, 4]	[1]	[0, 0.5, 0.7, 0.9, 1.2, 1.4]	[0, 0.5, 0.7, 0.9, 1.2, 1.4]	144	7200
2	[2, 3, 4]	[1]	[0.7, 0.8, 0.9]	[0.8, 0.9, 1.0, 1.1, 1.2]	45	2250

Table 5.1: HYPERPARAMETER SELECTION. Different sets of hyperparameters tested in two runs, showing the number of possible combinations and experiments that were run.

cess rate ($success\_rate_i$). Using Equation (5.1), we assigned a score to the different combinations.

$$
\begin{aligned}
score_i = & \frac{ssim\_mean_i + msssim\_mean_i}{2} \\
& + success\_rate_i \cdot \frac{\overline{success\_rate}}{10^2} - iterations_i \cdot \frac{\overline{iterations}}{10^9}
\end{aligned}
\tag{5.1}
$$

The intuition behind the formula is to include various aspects in the overall evaluation of a hyperparameter combination. The first term of the equation measures the average SSIM and MSSSIM score of the adversarial images in a group. It is the most important part since we want to generate adversarial images with high structural similarity to the original image. In the second term, we reward combinations with a high success rate, where the average success rate on the overall data ($\overline{success\_rate}$) determines the influence of this term. In the third term, a combination is penalized if it takes more iterations than average ($\overline{iterations}$) to generate the adversarial images. The factors 10^2 and 10^9 determine the influence of the corresponding terms and were determined experimentally. It should be noted that many other combinations produce adversarial images with almost identical scores. We do not intend the formula to be used as the ultimate tool for finding the best combination but merely to describe our intuition mathematically behind our selection of hyperparameters. An extract of the results of the two hyperparameter search runs is shown in Table 5.2. We show the three best and the three worst results for each run. The selected hyperparameters, the average SSIM and MSSSIM scores, the average number of iterations required, the success rate (SR), and the total score calculated with the formula in Equation (5.1) are listed. We can observe that, depending on the choice of hyperparameters, there is a trade-off between the two similarity metrics and the success rate. A higher success rate is associated with worse similarity metrics. There are combinations with an excellent balance between similarity scores and success rates, combinations with very high success rates but low similarity scores, and combinations with low success rates and very high similarity scores. Ultimately, the choice of the combination depends on the desired properties of the synthesis. Based on our evaluation formula, the hyperparameter combination with the best overall score is highlighted in Table 5.2 (green) and listed in Equation (5.2).

$$
\begin{aligned}
width_{step} &= 3.0 \\
w_{cos} &= 1.0 \\
w_{ssim} &= 0.9 \\
w_{msssim} &= 1.0
\end{aligned}
\tag{5.2}
$$

5.2 Image Sampling for Experiments

We want to test whether different properties of the respective sources and targets used in the synthesis process influence the performance of the extended LOTS algorithm. For this purpose, we define different experiments, each consisting of 100 samples with specifically selected properties. In the rest of this section, we describe how we sample the sources and targets used in the

Run	$width_{step}$	w_{cos}	w_{ssim}	w_{msssim}	∅ SSIM	∅ MSSSIM	∅ Iter.	SR	Score
1	3.0	1.0	1.2	0.9	0.9819	0.9912	1932	81.63%	0.99023
1	3.0	1.0	1.4	1.2	0.9835	0.9922	2318	73.47%	0.99020
1	3.0	1.0	0.9	1.2	0.9794	0.9918	1738	87.75%	0.99009
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	2.0	1.0	0.0	0.0	0.9077	0.9701	336	100.00%	0.94677
1	3.0	1.0	0.0	0.0	0.9033	0.9685	245	100.00%	0.94391
1	4.0	1.0	0.0	0.0	0.8936	0.9650	209	100.00%	0.93735
2	**3.0**	**1.0**	**0.9**	**1.0**	**0.9792**	**0.9912**	**1504**	**89.80%**	**0.99090**
2	3.0	1.0	0.8	1.2	0.9782	0.9917	1467	91.84%	0.99088
2	3.0	1.0	0.9	1.2	0.9793	0.9918	1671	87.76%	0.99085
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2	2.0	1.0	0.7	0.8	0.9770	0.9904	1580	90.00%	0.98931
2	4.0	1.0	0.7	1.2	0.9758	0.9912	1415	88.00%	0.98923
2	2.0	1.0	0.8	1.0	0.9786	0.9913	1886	78.72%	0.98911

Table 5.2: RESULT HYPERPARAMETER SEARCH. The three best and worst results of the two hyperparameter search runs. The selected hyperparameters, the average SSIM and MSSSIM scores, the average number of iterations required, the success rate (SR), and the total score calculated with the formula in Equation (5.1) are listed. The combination with the best overall score is highlighted in green.

experiments so that they comply with the requirements.

5.2.1 Random Sampling

The first experiment consists of randomly sampled source and target identities. This experiment is designed to test the general performance of the extended LOTS algorithm when no specific consideration is given to the characteristics of the source and targets. Since the VGGFace2 dataset is not balanced in terms of gender, we ensure that half of the sampled sources are female and the other half are male. The sources are sampled only from the test set of the VGGFace2 dataset. We further ensure that the targets are gender-balanced and come from both the test and the training set of the VGGFace2 dataset. We can apply this process to sample any amount of targets for a given source.

There is no guarantee that after applying the extended LOTS algorithm, the cosine distance between the source and the extracted feature vector from the adversarial image is still below the cosine threshold. It could be that the feature vector is too far away from the source due to the perturbations, and the network no longer recognizes the person who is depicted in the image. To prevent this, we can additionally define that the source can also be added to the list of targets. This ensures that the feature vector from the adversarial image does not move too far away from the source.

The random sampling process results in experiments (1) and (2), listed in Table 5.3.

5.2.2 Gender-Based Sampling

The goal of the gender-based experiments is to find out whether the gender of the sources or targets impacts the performance of the extended LOTS algorithm, whether syntheses between same-sex sources and targets exhibit fewer perturbations, and whether there is a gender for which the algorithm works better. In gender-based sampling, we proceed analogously to random sampling.

#	**Name**	Abbr.	Sampling Method	Comment
1	Random	*random*	Random Sampling	Sources and targets are sampled randomly
2	Including Source	*include source*	Random Sampling	Sources and targets are sampled randomly and the source is added to the list of targets
3	Female → Female	$f \to f$	Gender-Based Sampling	Sources and targets are female
4	Female → Male	$f \to m$	Gender-Based Sampling	Sources are female, targets are male
5	Male → Male	$m \to m$	Gender-Based Sampling	Sources and targets are male
6	Male → Female	$m \to f$	Gender-Based Sampling	Sources are male, targets are female
7	Min. Cosine Distance	*min*	Distance-Based Sampling	Cosine distances between sources and targets are minimal
8	Min. Cosine Distance Group	*min group*	Distance-Based Sampling	Cosine distances between sources and targets and within the targets are minimal
9	Max. Cosine Distance	*max*	Distance-Based Sampling	Cosine distances between sources and targets are maximal
10	Max. Cosine Distance Group	*max group*	Distance-Based Sampling	Cosine distances between sources and targets and within the targets are maximal
11	Min. Cosine Distance Group + Include Source	*min group* + *include source*	Distance-Based Sampling	Cosine distances between sources and targets and within the targets are minimal and the source is added to the list of targets
12	Blurry Image	*blurry*	Quality-Based Sampling, Random Sampling	Sources and targets are sampled randomly and the *blurriest* image is selected as the source image
13	High-Quality Image	*high quality*	Quality-Based Sampling, Random Sampling	Sources and targets are sampled randomly and a *high-quality* image is selected as the source image
14	Sharp Image	*sharp*	Quality-Based Sampling, Random Sampling	Sources and targets are sampled randomly and the *sharpest* image is selected as the source image

Table 5.3: EXPERIMENT TYPES. Table listing the various experiments and indicating which method was used to generate them.

However, we specify whether all sources should be female or male and whether all targets should be female or male. The sources are again sampled only from the test set, whereas targets are sampled from the test set and the training set. This procedure results in four different experiments (3)-(6), listed in Table 5.3.

5.2.3 Cosine-Distance-Based Sampling

In this subsection, we define experiments in which sources and targets are selected based on cosine distances. We want to analyze whether the distances between source and targets impact the size of the perturbations. We expect samples in which the sources and targets are close together to have smaller perturbations and require fewer iterations to bring all cosine distances below the defined threshold. Correspondingly, we expect more perturbations for samples in which the source and targets are farther apart. This concept results in 4 different sampling approaches, illustrated in simplified form in Figure 5.1. Using these four sampling approaches, we define the five experiments (7)-(11) , which are listed in Table 5.3.

Calculating Gallery Template Cosine Distances

To determine samples where the cosine distance between sources and targets corresponds to specific properties, we first need to know the distances between all possible identities. We achieve this by calculating the distances between all gallery templates. We store the distances between the respective identities in a matrix, which we can use as a lookup table for later experiment generation. Since we have 9084 identities in our dataset, we calculate 82’519’056 cosine distances between the gallery templates of all identities. We then use the gallery template cosine distances to look up distances between source and targets during the sampling process.

Minimal Cosine Distance Between Source and Targets

We want to find samples where the sum of the distances between the source and the respective targets is minimal. However, there are already pairs of identities that look so similar that the cosine distance between their gallery templates is below the cosine distance threshold. In order to exclude such samples in which the source and target are already below the threshold, we introduce a range in which the cosine distance of two gallery templates must lie. The range is defined as $[\tau_{cosDist} + 0.05, \tau_{cosDist} + 0.10]$. To generate the sample set, we proceed as follows. We use the distance lookup table for each source identity from the VGGFace2 test set to find all other identities where the distance to the source identity is in the defined range. From these identities, we randomly select the desired number of targets. For all selected targets, the following condition must be fulfilled:

$$\forall t \in targets: \quad \tau_{cosDist} + 0.05 \leq D_c(source, t) \leq \tau_{cosDist} + 0.1 \tag{5.3}$$

For each identity, the average distance to its sampled targets is calculated. We then sort the result based on the calculated average distances. The 100 samples with the smallest average distance are used for the experiment. Finally, we randomly select a source image for each source identity.

Minimal Cosine Distance Between Source and Within Targets

In this experiment, we not only want to minimize the cosine distance between the source and the targets but also ensure that the cosine distances between the individual targets are minimal. Finding the samples that perfectly minimize all these distances is impossible for computational-cost reasons. We, therefore, use a heuristic iterative to find samples that minimize the corresponding distances in the best possible way. Again, we have the problem that there are already samples with a cosine distance below the threshold, which we want to exclude. However, we do not need to define a range but can instead define a lower limit ($\tau cos + 0.05$). Then we select a target for each source from the VGGFace2 test set, with a cosine distance that is above the limit but minimal. In an iterative step, we select the next target (t_{new}) based on the conditions in Equation (5.4).

$$t_{new} = \underset{t \in I \ with \ \tau_{cos} + 0.05 \leq D_c(source, t)}{\arg\min} D_c(source, t) + \sum_{n=1}^{N} D_c(t, t_n) \tag{5.4}$$

From all possible identities in the VGGFace2 dataset, excluding the source and already selected targets (denoted as I), we select the target that minimizes the cosine distance between itself and the source as well as itself and all other targets selected in previous steps ($t_1, \ldots, t_N$). Additionally, the condition that $\tau cos + 0.05$ is less or equal to the cosine distance between the source and the newly selected target must be fulfilled. For each source identity, the distances are continuously summed up. Afterward, the 100 samples with the smallest total distance are selected. Finally, we select a random source image for each source identity.

Maximal Cosine Distance Between Source and Targets

We want to select samples having a maximum distance between the source and the respective targets. We can use the distance lookup table to select the desired number of targets that maximize the distance to the source for each identity from the VGGFace2 test set. For each newly selected target (t_{new}) the following condition has to be fulfilled:

$$t_{new} = \underset{t \in I}{\arg\max} \, D_c(source, t) \tag{5.5}$$

From all possible identities in the VGGFace2 dataset, excluding the source and already selected targets (denoted as I), we select the target that maximizes the cosine distance between itself and the source. For each identity, we compute the total distance between the source and all the targets. We then sort the result by the total distance and select the 100 identities with the largest total distance for the experiment. We further randomly select a source image for the 100 samples.

Maximal Cosine Distance Between Source and Within Targets

Finally, we want to select samples that have a maximum distance between source and target as well as between the individual targets themselves. Analogous to the minimum distances, the computation of the perfect samples is computationally expensive. We, therefore, use a heuristic iterative to find satisfying samples. For each identity of the VGGFace2 test set, we search for the corresponding target that maximizes the distance between source and target. In the next iterative step, we search for a target (t_{new}) that satisfies the condition in Equation (5.6).

$$t_{new} = \underset{t \in I}{\arg\max} \, D_c(source, t) + \sum_{n=1}^{N} D_c(t, t_n) \tag{5.6}$$

From all possible identities in the VGGFace2 dataset, excluding the source and already selected targets (denoted as I), we select the target that maximizes the cosine distance between itself and the source as well as itself and all other targets selected in previous steps ($t_1, \ldots, t_N$). We sum up the total distance per identity. The identities are sorted by the total distance, and the 100 samples with the largest distance are selected for the experiment. For each source identity, a source image is randomly selected.

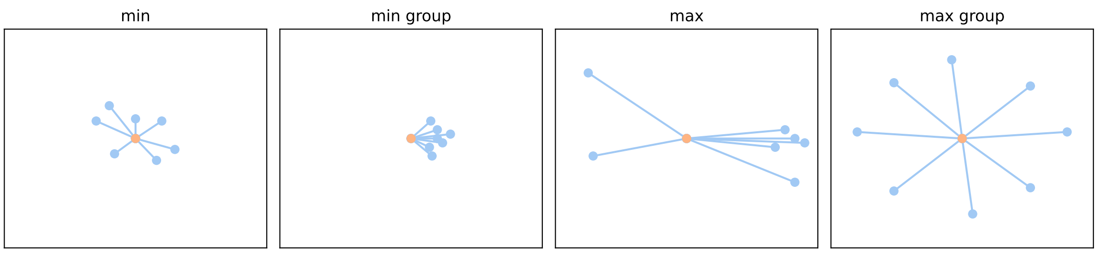


Figure 5.1: Visualization Cosine-Distance-Based Sampling. Visualization of the sampling of different cosine-distance-based experiments. The orange point represents the source, the blue points the selected targets. Note that we compare the cosine distances of n-dimensional vectors in the actual sampling process. In these plots, the cosine distances are expressed by the length of the connecting line for illustration purposes.

5.2.4 Image-Quality-Based Sampling

If we want to investigate different qualities of images, we can resort to random sampling. For the image quality, we do not want to look at the composition and properties of the source or targets; instead, we want to select the source image, which is modified by the extended LOTS algorithm, according to its quality. We are interested in blurry, high-quality, and overly sharp images.

To evaluate the quality of an image, we use the OpenCV Python library.[1] We transform an image into a grayscale image and apply a Laplacian filter to it, highlighting regions with fast intensity changes. If we calculate the variance of this filter, this can be considered a measure of the focus of the image (Pech-Pacheco et al., 2000).

We use the sources and targets we have already selected in the random sampling and replace the source image with an image having the desired quality. For each of these sources, we load all available images and calculate the variance of the Laplacian filter. Then we sort the available images of each identity according to the calculated variance. The image with the smallest value corresponds to the blurriest image of the identity. Accordingly, the image with the highest value corresponds to the sharpest image. To find a high-quality image, we randomly select an image with a Laplacian filter variance between 400 and 600. The range for the variance corresponding to high-quality images was determined experimentally. The three resulting experiments (12)-(14) are listed in Table 5.3.

5.3 Evaluation Methodology

This section will show the methodology we used to determine which experiments were performed. We further introduce two different factors that impact the difficulty of the experiments.

5.3.1 Attack Difficulty

We can artificially influence the difficulty of the attacks. In Section 4.1.4, we calculated different False Match Rates and the corresponding cosine distance thresholds for our network. The lower we choose the False Match Rates, the lower the cosine distance threshold has to be between the source and all targets, and the more difficult the attack becomes. We will use different False

[1] https://github.com/opencv/opencv-python

Match Rates for our experiments, simulating a more difficult or a less difficult attack. For the experiments, we will use the False Match Rates 10^{-3}, 10^{-4}, and 10^{-5}.

In addition to changing the difficulty of the attack using different False Match Rates, we can also control the degree of difficulty by altering the number of targets that are attacked simultaneously. The more cosine distances between source and targets have to fall below the defined cosine distance threshold at the same time, the more complex the experiment. We perform the experiments with a number of targets ranging from 1 to 20. Besides the number of targets, the $include\ source$ experiment also impacts the difficulty since the source is also added to the list of targets, and we effectively have one additional target.

5.3.2 Iterative Experiment Definitions

There are numerous ways to define a set of experiments. Our experiments contain 100 samples having specific characteristics regarding the source-targets constellation. We can perform them at different difficulty levels by combining various False Match Rates and the number of targets to attack. To find our set of experiments, we selected them in an iterative process. Since not all experiments are suitable for the same difficulty level, we start at an intermediate difficulty and rerun the experiment at a harder or easier difficulty level, depending on the results. We start all experiments at a False Match Rate of 0.001 and by attacking four targets. For each experiment, we measure the average MSSSIM and SSIM scores, the average number of iterations required by the extended LOTS algorithm for a successful attack, and the success rate of the individual samples per experiment. After each run, we evaluate the performance of the experiments. Experiments with less than 75% success rate were carried out again on a lower difficulty by attacking one fewer target. Experiments with a success rate of higher or equal to 95% were performed again on a harder difficulty, attacking one more target. Exceptions are the gender-based experiments since a statement regarding the influence of gender is already possible after one run. Furthermore, the $random$ experiment is conducted on all difficulty levels, from attacking one target to attacking six targets, since it serves as a baseline. While conducting the experiments, we observed good performance of the $min\ group$ and a $min\ group + include\ source$ experiment. Therefore, we run these experiments with a hard difficulty by attacking 10, 15, and 20 targets.

We experimented with more difficult attacks in the second part by lowering the False Match Rates. However, we only considered $random$ and the $include\ source$ experiment. This is because peculiarities of different compositions of source and targets and image qualities and groupings can also be investigated by executing less difficult attacks on higher False Match Rates. The main goal of the second part was to investigate whether, in general ($random$ experiment), attacks are still possible at all. For the False Match Rate of 0.0001, we successfully performed experiments with attacking 1, 2, 3, or 4 targets simultaneously. However, the $random$ experiment attacking 4 targets had a success rate of only 5%. For the attacks using a False Match Rate of 0.00001, only experiments targeting 1 or 2 targets were performed due to the already low success rate of 57% on the $random$ experiment attacking 2 targets.

5.4 Evaluation Results

A total of 48 experiments with 100 images each were performed, corresponding to the generation of 4800 adversarial samples. The results of the evaluation are shown in Table 5.4. We performed all experiments on an Nvidia RTX 3070 or an Nvidia RTX 2080Ti.

The $random$ experiment achieves a success rate of 88% when attacking four targets and using a False Match Rate of 0.001. With five targets, the success rate decreases to 51%, and with six targets, to 26%. When comparing the $include\ source$ experiment to the $random$ experiment,

Difficulty		Experiment Type	∅MSSSIM	∅SSIM	∅Iter.	SR
FMR@0.001	4 Targets	*random*	0.9909	0.9792	1723.81	88%
		include source	0.9908	0.9791	3577.67	43%
		min	0.9962	0.9900	197.50	99%
		max	0.9902	0.9758	89.75	100%
		min group	0.9978	0.9935	11.14	100%
		max group	0.9866	0.9695	4978.38	1%
		blurry	0.9912	0.9819	1247.57	93%
		sharp	0.9903	0.9800	2479.49	73%
		high quality	0.9904	0.9788	2205.05	76%
		$m \to m$	0.9922	0.9820	991.62	94%
		$m \to f$	0.9909	0.9785	529.89	97%
		$f \to f$	0.9941	0.9854	409.92	99%
		$f \to m$	0.9908	0.9786	1089.53	95%
	3 Targets	*random*	0.9928	0.9827	395.59	99%
		include source	0.9922	0.9824	1837.78	80%
		max group	0.9873	0.9711	3842.22	40%
		sharp	0.9922	0.9834	548.17	99%
	2 Targets	*random*	0.9951	0.9867	27.23	100%
		include source	0.9953	0.9886	241.01	100%
		max group	0.9898	0.9757	264.88	100%
	1 Target	*random*	0.9980	0.9940	7.83	100%
		include source	0.9979	0.9941	12.24	100%
	5 Targets	*random*	0.9897	0.9766	3319.74	51%
		min	0.9951	0.9881	715.30	95%
		max	0.9901	0.9763	196.96	99%
		min group	0.9972	0.9921	18.17	100%
	6 Targets	*random*	0.9892	0.9757	4230.10	26%
		min	0.9943	0.9868	1357.89	86%
		max	0.9906	0.9774	594.24	95%
		min group	0.9971	0.9917	47.87	100%
	7 Targets	*min group*	0.9963	0.9902	58.50	100%
		max	0.9906	0.9781	932.90	89%
	10 Targets	*min group*	0.9951	0.9880	250.04	100%
		min group + include source	0.9953	0.9884	307.05	100%
	15 Targets	*min group*	0.9935	0.9844	986.95	94%
		min group + include source	0.9936	0.9848	1124.22	92%
	20 Targets	*min group*	0.9925	0.9828	2188.38	80%
		min group + include source	0.9925	0.9830	2373.72	77%

Table 5.4: PART 1: EXPERIMENT RESULTS. Results of the experiments conducted at FMR=0.001. Results are measured using average MSSSIM and SSIM scores, the average number of iterations, and the success rate (SR). The average values also include the values of the experiments that were not successful.

we must consider that the $include\ source$ experiment effectively attacks one additional target. Therefore, we always compare the $include\ source$ experiments with $random$ experiments that attack one more target to ensure a fair comparison. When comparing the $random$ experiment with 4 targets and the $include\ source$ experiment with 3 targets, we can observe a slightly higher success rate and slightly fewer iterations needed in the $random$ experiment. However, the similarity metrics favor the $include\ source$ experiment. The same can be observed when comparing the $random$ experiment with 5 targets and the $include\ source$ experiment with 4 targets. We can conclude that including the source makes the synthesis of the adversarial image slightly more difficult, as can be observed in the success rates and the average iterations needed. Regardless, including the source also ensures better similarity metrics since the algorithm is constrained in moving feature vectors extracted from the adversarial image too far away from the source. If the number of targets attacked decreases, the synthesis of the adversarial image becomes easier, and the effect observed where the $random$ experiment has a higher success rate and fewer average iterations needed vanishes. The positive effect on the similarity metrics by including the source remains, as it can be observed in the comparison between the $random$ experiment with 3 targets and the $include\ source$ experiment with 2 targets, or the $random$ experiment with 2 targets and the $include\ source$ experiment with 1 target. In general, we can say that for a higher number of targets, including the source increases difficulty but improves the similarity metrics. For a lower number of targets, including the source decreases the difficulty but still improves the similarity metrics. Figure 5.2 visualizes how the number of targets affects the degree of difficulty using a sample from the random experiment. It is evident that the higher the number of targets, the more perturbations are present, which is further proven by the similarity metrics and L_p norms.

Looking at gender-based experiments, it is evident that better similarity scores are achieved in experiments with same-gender sources and targets. Adversarial attacks with female sources and targets produce better similarity scores, need fewer iterations, and have a success rate of 99%, which is 5% higher than the male equivalents. In the $f \rightarrow m$, and $m \rightarrow f$ experiments, the similarity scores are almost identical but are significantly lower than the scores of the same-gender attacks. The experiments with female targets need fewer iterations than experiments with male targets and have a higher success rate. It appears to be easier to mimic the feature vectors of female subjects.

Regarding the experiments with quality-based sampling, $blurry$ images show better similarity scores, a better success rate (93%), and fewer iterations needed for a successful attack than the *high-quality* and $sharp$ images. The *high-quality* images are slightly ahead of the $sharp$ images in terms of combined similarity metrics, requiring about 200 iterations less on average and having a 3% higher success rate (76%). The experiment with the $sharp$ images was carried out again with three targets, where an average success rate of 99% was achieved with approximately 550 iterations.

In the distance-based experiments, it is immediately apparent that the $max\ group$ experiment is the only experiment that does not perform well, with a success rate of 1%. A possible reason for this might be that it is not feasible to change the adversarial image so that the extracted feature vector is close to all targets at the same time. The other experiments have a success rate of 99% or 100%. The min and $min\ group$ experiments have very high similarity scores, with the $min\ group$ experiment performing slightly better. The $min\ group$ experiment is also ahead of the min experiment regarding the number of iterations needed for a successful attack. Surprisingly, the max experiment is also ahead of the min experiment in terms of iterations but has significantly lower similarity scores. We can explain all these observations using the graphics in Figure 5.1. Looking at the max and the $min\ group$ experiment, we see that they need the fewest iterations. The targets in the $min\ group$ experiment are all in similar locations; therefore, they must all be modified in the same direction. There is no guarantee that the targets are grouped in the max experiment. However, it is likely that groups of similar-looking people will have similar distances and directions

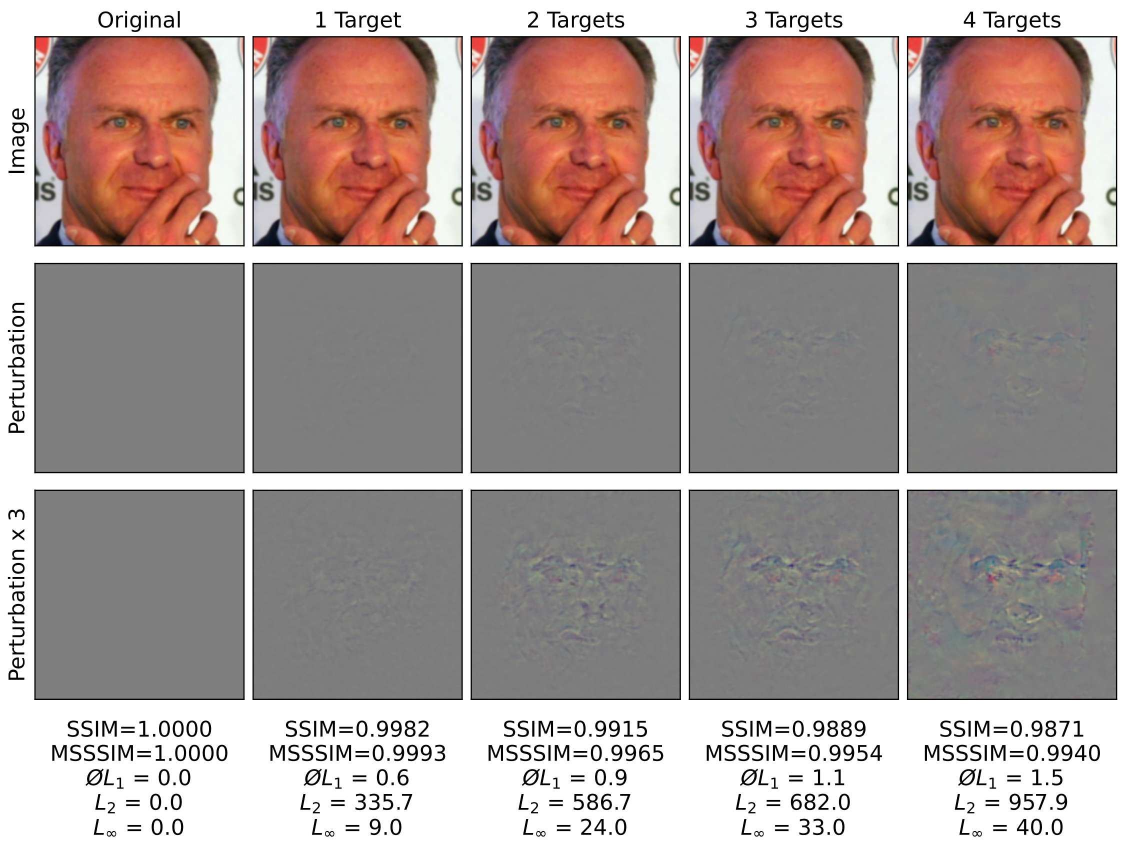


Figure 5.2: INFLUENCE OF NUMBER OF TARGETS ON ADVERSARIAL IMAGE. Visualization of the perturbations when attacking a different number of targets. Each column depicts an adversarial image with its isolated perturbations and the perturbations enhanced by scaling it with a factor of 3. Additionally, the similarity metrics and L_p norms are shown.

to the source and might be all selected for the same experiment. Since the cosine distances are larger in the max experiment, more iterations are needed than in the $min\ group$ experiment, and more perturbations are present. The min experiment still has good similarity scores, which is due to the fact that all targets are already close to the source. However, the adversarial image must be adjusted such that the extracted features change in multiple different directions, which leads to more iterations and perturbations. The $max\ group$ experiment is not successful with four targets because high distances and different adjustment directions make the experiment too difficult. In general, we can say that large distances lead to more perturbations, which is manifested in the poorer similarity scores of the max and $max\ group$ experiments. Furthermore, different adaptation directions also lead to more perturbations, which is why the $min\ group$ experiment has a better similarity score than the min experiment, and the max experiment has a better similarity score than the $max\ group$ experiment. The $max\ group$ experiment was only successful with two targets. The max experiment was still successful with an 89% success rate on seven targets, while the min experiment was still successful with an 86% success rate on six targets. When comparing both, the min and the max experiment on 6 targets, we see that the max experiment with a success rate of 95% outperforms the min experiment with a success rate of 86%. We can conclude that it is

Difficulty		**Experiment Type**	**∅MSSSIM**	**∅SSIM**	**∅Iter.**	**SR**
FMR@0.0001	4 Targets	*random*	0.9904	0.9782	4885.51	5%
	3 Targets	*random*	0.9915	0.9808	2771.41	59%
	2 Targets	*random*	0.9939	0.9845	189.89	99%
		include source	0.9936	0.9855	2653.01	62%
	1 Target	*random*	0.9978	0.9918	9.83	100%
		include source	0.9971	0.9924	100.45	99%
FMR@0.00001	2 Targets	*random*	0.9932	0.9845	2677.83	57%
		include source	0.9931	0.9847	4940.85	2%
	1 Target	*random*	0.9962	0.9891	13.44	100%
		include source	0.9959	0.9904	2149.98	64%

Table 5.5: PART 2: EXPERIMENT RESULTS. Results of the experiments conducted at FMR=0.0001 and FMR=0.00001. Results are measured using average MSSSIM and SSIM scores, the average number of iterations, and the success rate (SR). The average values also include the values of the experiments that were not successful.

more difficult to successfully synthesize an adversarial image for different adaptation directions than for large cosine distances. However, the different adaption directions produce better similarity scores. Accordingly, the *min group* has the best characteristics for both distance and direction and achieved a success rate of 80% for 20 targets at the same time. In addition, we carried out *min group* + *include source* experiments, in which the source was also added to the list of targets. Similar results were achieved on 10, 15, and 20 targets as in the normal *min group* experiment. However, there was a slight advantage in terms of success rate and average iterations needed for the normal *min group*, experiment, and a slight advantage for the *min group* + *include source* experiment in terms of similarity scores.

We further tested the *random* and the *include source* experiment using a more difficult attack by lowering the False Match Rates. The results are shown in Table 5.5. It is remarkable that the *random* experiments are still successful, but only with a limited number of targets. After all, at FMR=0.0001, 59% can still be achieved with three targets, 99% with two targets, and 100% with one target. Looking at the *include source* experiment, 62% of the attacks are still successful with two targets and 99% with one target. Regarding the experiments using an FMR of 0.00001, only 57% of the attacks are successful with two targets and 100% with one target. The *include source* experiment leads to a success rate of 2% for two targets and 64% for one target. We can observe the same behavior as for FMR=0.001 regarding experiments that include the source being more successful and having fewer perturbations on a low number of targets.

We can compare our results to a certain extent with the results from the original LOTS algorithm (Rozsa et al., 2017). The authors have also tested their adversarial images on a network that evaluates two feature vectors' similarity based on cosine distances. In addition, they assessed the similarity between the original image and the adversarial image with the PASS score, which can be translated into the SSIM score since there are no perspective transformations between the adversarial image and the original image. The authors attacked one target at a time on a network with a False Match Rate of 0.001. They achieved an average SSIM score of 0.9908. We can compare their result with our *random* experiment (False Match Rate of 0.001, 1 target), where we achieved an average SSIM score of 0.9940. In both experiments, a success rate of 100% was achieved. However, the feature vectors were not extracted from the same network. Nevertheless, we can say that we can attack one target with an adversarial image that contains fewer perturbations.

Of all the experiments performed on four targets and a network with a False Match Rate of 0.001, the best and worst successful adversarial samples are shown in the Appendix (A.2, A.3, A.4, A.5, A.6).

Chapter 6

Empirical Evaluation of Similarity Metrics

We could extract different SSIM and MSSSIM scores from the different experiments conducted. However, it is difficult to say how much significance these scores have about the quality of the adversaries. From visual observations, it appears that adversarial images with higher scores have less perceivable perturbations. However, it is impossible to say at which similarity scores human observers can no longer detect the perturbations. Therefore, we want to empirically evaluate the explanatory power of the scores with regard to human perception and compare the results with the experiments we conducted in the previous chapter.

6.1 Evaluation Setup

This section discusses how we conduct the qualitative evaluation of the similarity metrics using a custom-built questionnaire tool. We will discuss the functionalities of the tool as well as the selection of samples and the distribution of the questionnaire.

6.1.1 Custom Questionnaire Tool

With a survey, we wanted to determine at which SSIM resp. MSSSIM score a human observer can correctly recognize a modified image as such at first glance. The assumption is that the lower the score, the higher the probability that an adversarial image is detected as such. Since the usual survey tools are not suitable for this kind of question, we have developed our own survey tool. The tool consists of a front-end, developed with the Angular Framework[1] from Google, which takes care of the correct presentation of the questions. In addition, a Java back-end was created with the Spring Boot Framework,[2] which compiles an individual question catalogue for each participant and receives the answers from the participants. The answers are stored in a MariaDB SQL database.[3] We made sure that mobile users could also use the web application without issues.

[1]https://angular.io/
[2]https://spring.io/projects/spring-boot
[3]https://mariadb.org/

Landing Page

The goal of the landing page is to explain the task to a participant and to give clear and simple instructions. It should not matter how much previous knowledge a participant has. The introductory text explains the principle of adversarial images on a high level. In addition, a simple use case establishes a reference to everyday life. Furthermore, it is explained that the questions are intended to find out how our scores compare to human perception. The full introductory text can be found in the Appendix (A.2.1).

The next section of the page contains the instructions. We tell the participant that two pictures will be displayed simultaneously for two seconds. The picture on the left serves as a reference picture. It should show the participant what a person looks like in good quality and unchanged form. The right picture is the picture on which we want to test the participant. It can be either an adversarial image or an unaltered image. The participant must rate whether the image is altered or not on a Likert scale. The scale contains the following answer options: *unaltered*, *likely unaltered*, *do not know*, *likely altered*, and *altered*. The aim is to recreate a situation such as a passport control, where an employee takes a quick look at the passport photo (picture on the right) and compares it with the actual person (picture on the left). A passport photo is rarely looked at for more than 2 seconds. We also explain that we do not alter the shape of the face or the person's age. Also, blurriness, funny poses, makeup, watermarks, magazine cover letters are not considered alterations since such images are present in the dataset. It would be easier to show examples of adversarial images. However, since the perturbations are very distinctive, the participants would quickly notice a pattern. Since we mainly want to test whether the participants can spot the adversarial images at first glance, we cannot give much assistance. We also show the participants a set of eight images containing blurry images, images containing watermarks, people in strange poses, etc., to make it clear that such attributes do not count as an alteration. The full set of instructions can be found in the Appendix (A.2.1).

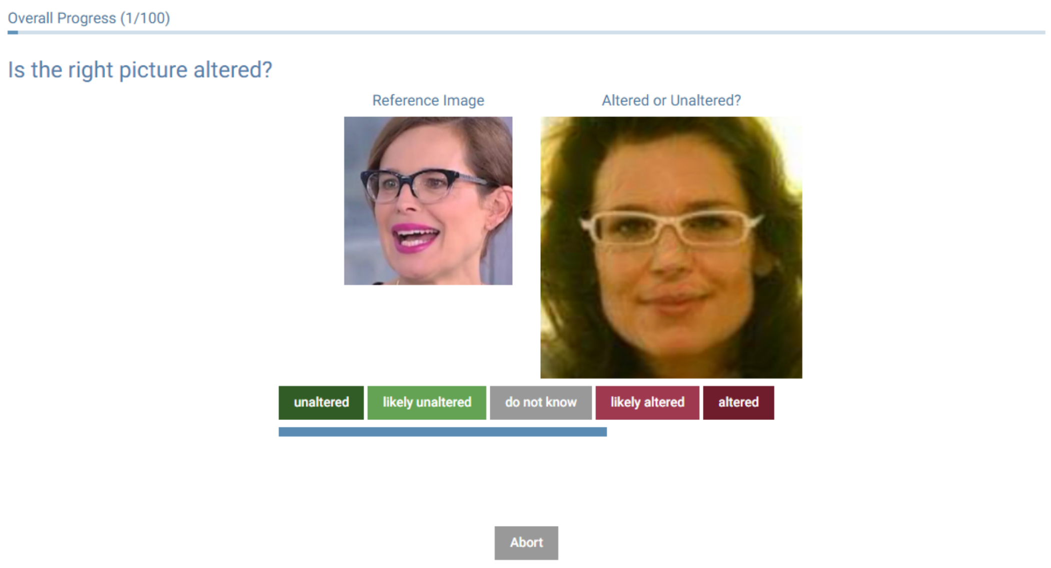
In the last section, we query the age and gender of the participant. The participant can choose how many questions they would like to answer, whereby corresponding time estimations are provided. The options include 70, 100, 130, or 160 questions, whereas 100 questions are the default. As a small incentive, participants can enter a raffle to win a voucher, with the probability of winning proportional to the number of questions answered. To enter the raffle, the participants have to enter their e-mail addresses. A screenshot of the complete landing page is provided in the Appendix (A.1).

Question Page

As soon as the participant has started the questionnaire, he or she will be redirected to the question page. Here, either adversarial images or unaltered images with their corresponding reference images are presented for 2 seconds. The participants now must classify the displayed image in the Likert scale mentioned above. After the 2 seconds have elapsed, the images are faded out, but the participant can still rate them on the scale. We have also ensured that the images are not displayed again in the event of a page refresh. A screenshot of an example question is shown in Figure 6.1. Below the Likert scale, a blue bar indicates the remaining time the two images are displayed.

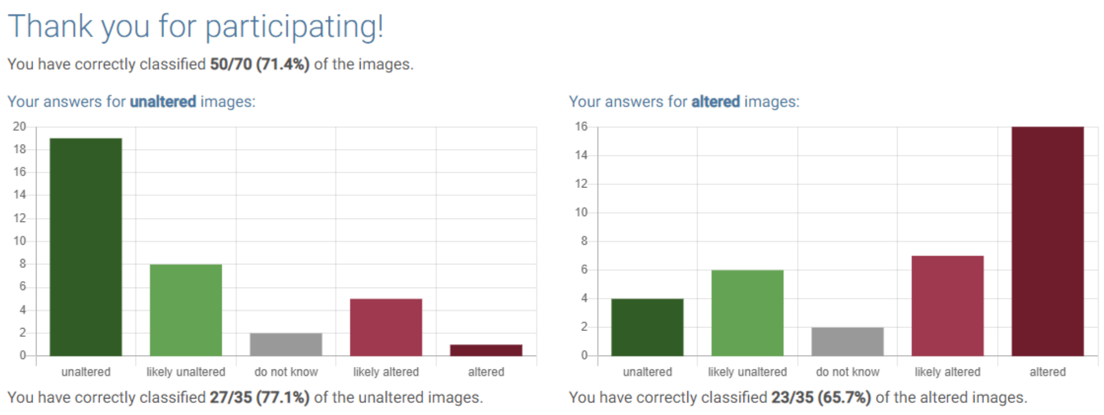
Result Page

After the participant has answered all the questions, or in the event of aborting the experiment, the result page is displayed. Here, we present the participant's performance visually. Two bar charts show how the participant has assigned the images on the Likert scale to the corresponding classes (altered/unaltered). In addition, the correct classification rate per class and overall is


Overall Progress (1/100)

Is the right picture altered?

Figure 6.1: QUESTIONNAIRE SCREENSHOT: QUESTION. Screenshot of an image pair. The left image is the reference image, and the right image is the image we test the participants on. Below the image is, we display the Likert scale the participant has to use for the rating. Below the Likert scale, a blue bar indicates the remaining time the two images are displayed.


Thank you for participating!

You have correctly classified **50/70 (71.4%)** of the images.

Your answers for **unaltered** images:

You have correctly classified **27/35 (77.1%)** of the unaltered images.

Your answers for **altered** images:

You have correctly classified **23/35 (65.7%)** of the altered images.

Figure 6.2: QUESTIONNAIRE SCREENSHOT: RESULT. Screenshot of the result page. The page shows the overall performance of the participant as well as the performance per class (altered/unaltered).

Algorithm 4 Calculating Sampling Probabilities

```
1: $df$ = load selected samples into dataframe
2: $df$ = select only adversarial samples from $df$
3: for $score$ in ['ssim', 'msssim'] do
4:     $min$ = $df[score]$.quantile(0.015)
5:     $max$ = $df[score]$.quantile(0.985)
6:     $bucket\_size$ = $(max\text{-}min)/10$
7:     $buckets_{score}$ = create 10 buckets with $bucket\_size$
8: end for
9: for $sample$ in $df$ do
10:     $i$ = get index of bucket for $sample['ssim']$
11:     $buckets_{ssim}[i]$ += 1
12:     $j$ = get index of bucket for $sample['msssim']$
13:     $buckets_{msssim}[j]$ += 1
14: end for
15: $buckets\_prob_{ssim}$ = [min($buckets_{ssim}$) / $n$ for $n$ in $buckets_{ssim}$]
16: $buckets\_prob_{msssim}$ = [min($buckets_{msssim}$) / $n$ for $n$ in $buckets_{msssim}$]
17: for $sample$ in $df$ do
18:     $i$ = get index of bucket for $sample['ssim']$
19:     $j$ = get index of bucket for $sample['msssim']$
20:     $sample['sampling\_probability']$ = $(buckets\_prob_{ssim}[i] + buckets\_prob_{msssim}[j])/2$
21: end for
```

shown. The charts provide the participants with an overview of their performance. It is ensured that the participant cannot participate a second time by setting a questionnaire completed variable in the browser's session storage. A screenshot of the results page is shown in Figure 6.2.

6.1.2 Sample Selection

We cannot query all samples from all experiments for the questionnaire. Therefore, we had to make a selection of samples that we presented to the participants. The main objective was to select samples from each experiment category and ensure that the SSIM and MSSSIM scores were distributed as evenly as possible within their range. The sampling was done in a two-step process.

In the first step, we loop through the experiments and select those that have a success rate of at least 10%. We then randomly selected 10 samples from each of the remaining experiments, with the adversarial image being used for half of the samples and an unaltered original image being used for the other half.

Since the SSIM and MSSSIM scores are not uniformly distributed in the selected samples, we compensate for this in the second step. For this purpose, we calculate a sampling probability, which is taken into account when generating the individual question sets for the participants. However, it is difficult to distribute the SSIM and MSSSIM values exactly uniformly because the scores always occur in pairs since a sample has an SSIM and an MSSSIM score. In order to obtain a reasonably good distribution, we create buckets for the SSIM and MSSSIM scores. We use the 1.5%-Quantile as lower bound and the 98.5%-Quantile as upper bound to filter outliers. The range between these two bounds is divided evenly into 10 buckets. We iterate through all samples and distribute the SSIM and MSSSIM scores into the corresponding buckets. We keep track of how many samples are present in each bucket. We then calculate the sampling probability of each bucket by dividing the minimum value of all buckets by the bucket's own value. This ensures

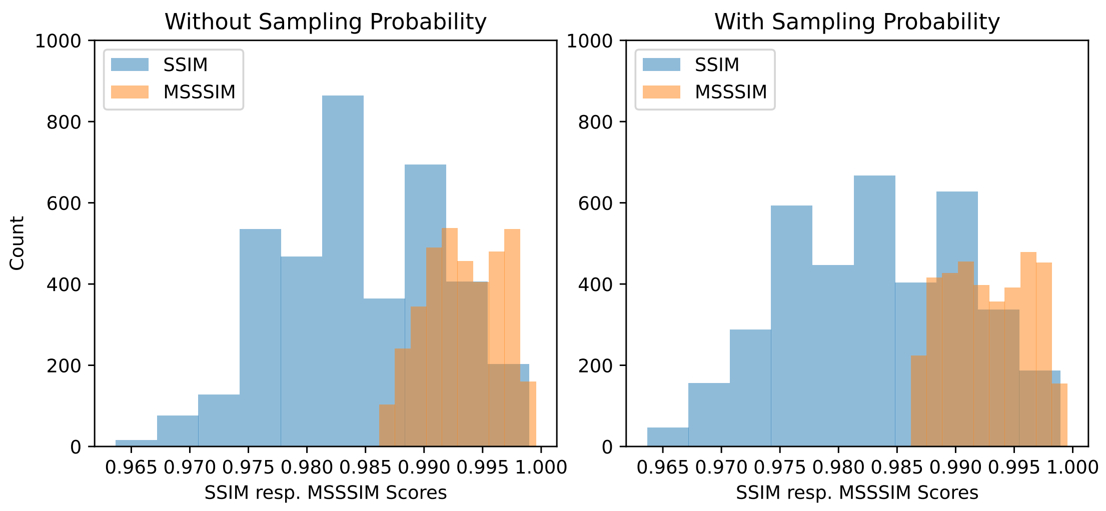


Figure 6.3: INFLUENCE OF SAMPLING PROBABILITY. Histogram of the SSIM and MSSSIM distribution with and without considering the sampling probability. Simulated assuming 75 people each answering 100 questions, where half are adversarial images.

that buckets with a large count get a small sampling probability, and buckets with a small count get a large sampling probability. Since the SSIM and MSSSIM scores occur in pairs, the sampling probability is calculated from the average of a sample's respective SSIM and MSSSIM sampling probabilities. The sampling process is depicted in Algorithm 4.

We simulated the generation of the question sets in advance. We assumed 75 participants, each answering 100 questions. Figure 6.3 shows how the distribution of the SSIM and MSSSIM scores changes within the buckets when considering the sampling probability. It is also evident that the two scores cover different ranges, with the buckets of the MSSSIM scores being much narrower. It is clearly visible that no perfectly uniform distribution is possible due to the occurrence of the scores in pairs. Nevertheless, the scores are distributed much more nicely.

6.1.3 Distribution of the Questionnaire

We distributed the questionnaire through several channels. On the one hand, friends with all different backgrounds were contacted directly. On the other hand, we distributed the questionnaire to people with IT affinity through employment relations. Additionally, people from the Artificial Intelligence and Machine Learning Group at the University of Zurich were informed by e-mail. Finally, we spread the questionnaire on the University of Zurich's marketplace. This should lead to a balanced group of participants regarding gender and background. The questionnaire was online for 20 days.

6.2 Results

In this section, we analyze the data we collected. We look at the participants' demographics and describe how we create a logit regression model and how to extract statements from it. We analyze the participants' responses and present our findings using the logit regression model.

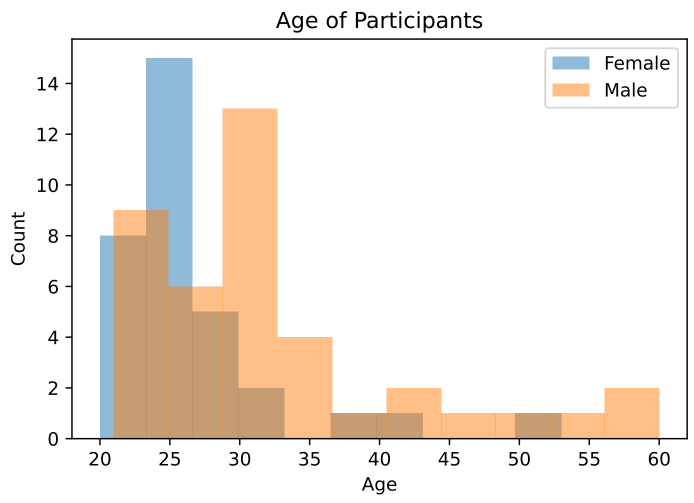


Figure 6.4: AGE DISTRIBUTION PARTICIPANTS. Distribution of the participants' age by gender.

6.2.1 Demographics

A total of 73 participants took part in the questionnaire. Thus, 7990 individual questions were answered, with a single participant answering an average of approximately 104 questions. Of the participants, 33 were female, and 40 were male. The average age was 29.8 years, with the youngest participant being 20 years old and the oldest 60 years old. Figure 6.4 shows a histogram of the age distribution for the different genders. It becomes clear that more younger people participated, but the age distribution between the genders does not differ much. The reason for this is that the questionnaire was distributed mainly through university channels.

6.2.2 Initial Observations

After loading, cleaning and preprocessing the data, we can already make some initial observations. We generate a new binary variable $rated\_as\_adversarial$ from the Likert scale we used in the questionnaire. As the name implies, the variable should reflect whether a participant identified a single sample as an adversarial image. For this purpose, we assign the Likert scale values $unaltered$ and $likely\ unaltered$ as False and the values $altered$ and $likely\ altered$ as True. The questions answered with $do\ not\ know$ are omitted. We then plot a histogram showing the SSIM and MSSSIM scores partitioned by the new $rated\_as\_adversarial$ variable (Figure 6.5). We only consider the adversarial samples since the SSIM or MSSSIM score would always be 1 for the unaltered images. The first row compares the SSIM scores, where the first column is the distribution of responses in which the adversarial images were detected as such. The second column shows the distribution of responses in which the adversarial sample was able to deceive human perception and was not detected. Comparing the two graphs, it is evident that the distribution for the correctly recognized images is balanced, and the distribution is skewed to the right for the adversarial images that were incorrectly rated as original/unaltered. Therefore, we can conclude that, on average, better SSIM scores are required to deceive a human, which was to be expected. Overall, the participants were more often able to unmask the adversarial images than they were fooled. In the second row, we analyze the answers using the same methodology but with respect to the MSSSIM scores. The analysis of the SSIM scores can be transferred one-to-one to the

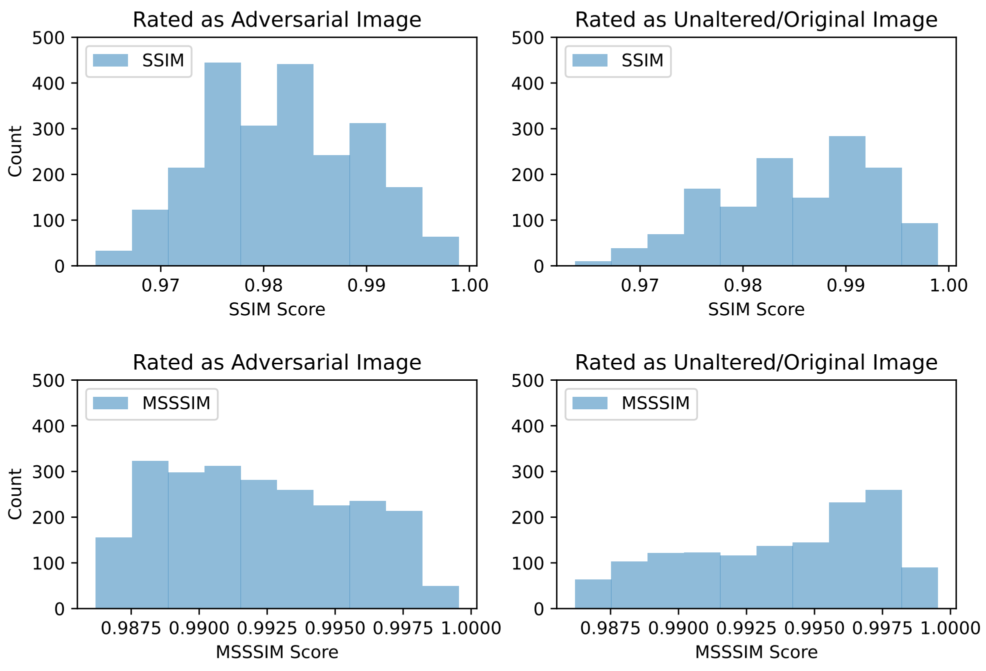


Figure 6.5: CLASSIFICATION OF ADVERSARIAL SAMPLES. Distribution of adversarial samples correctly classified as adversarial (first column) and incorrectly classified as unaltered (second column) for the SSIM score (first row) and the MSSSIM score (second row).

MSSSIM scores.

If we focus on the responses regarding unaltered images, we can calculate how many of these images were correctly rated as such. Surprisingly, only 66.2% of the unaltered images were correctly classified. We can use this value as a benchmark in later experiments. Several factors may be the reason why the value is so low. On the one hand, some participants might not have understood the task correctly, or the task was too difficult. On the other hand, people might have wanted to finish the questionnaire as quickly as possible. We, unfortunately, cannot derive the actual reason from our data. The low correct classification rate on the unaltered images is also generally reflected in the wide scatter of the data. Many samples, even those with very high or very low SSIM or MSSSIM scores, were classified differently by various participants. This wide scatter makes the evaluation of the data more difficult.

Finally, we collected a few statements from the participants. $Participant$ 1 stated that after about a third of the experiment, he noticed the type of perturbations we applied to the images. From then on, it was easier to classify the images correctly. A similar statement came from $Participant$ 2, who said he noticed a pattern halfway through. $Participant$ 3 would have liked to see an example of an adversarial image. As mentioned before, this was not possible because it would have made the task too easy. $Participant$ 4 (55+ years) felt that the 2 seconds we showed the images for was too short.

6.2.3 Logit Regression Model

A logit regression model is a subclass of generalized linear models. It allows us to estimate a binary variable from any number of independent variables. In our case, we want to find a model that can predict the dependent variable $rated\_as\_adversarial$ from different experiment conditions. From the calculated model, we can later extract statements about the individual independent variables, such as the SSIM score.

Preparation

Three preprocessing steps are necessary before we can create the first models. Since we suspect a learning effect over time, which is also evident from the statements of $Participant\ 1$ and $Participant\ 2$, we want to introduce two new variables that allows us to observe this learning effect. Having created a unique random ID for each participant and having stored a timestamp for each answer, we can assign a temporal index to each answer. We have to distinguish whether the image is original or adversarial. We select all answers of a single participant and sort them in ascending order by timestamp. Then we can assign a running index to each answer, incrementing the variable $temporal\_orig$ for original images and the variable $temporal\_adv$ for adversarial images. For original images, we set the variable $temporal\_adv$ to 0, and for adversarial images, vice versa. This is because the two variables can each be viewed as a combination between the group membership (adversarial image or original image) and the indicator variable (temporal index). This is a standard procedure to observe effects isolated per group. In other words, the temporal index of an adversarial image does not influence the learning effect on the original images. The same is true the other way around.

Since we also want to include the participants' answers regarding original images, we need to assign corresponding SSIM and MSSSIM scores to these questions. Since the original images do not contain perturbations, the SSIM and MSSSIM score values are set to 1.0.

Furthermore, we have to evaluate whether we need to balance the data with respect to the variable $rated\_as\_adversarial$. Imbalanced datasets can lead to models with a bias. We have collected 7990 answers. Of these answers, 480 were rated as *do not know* and could not be considered. Of the remaining samples, 3624 were answered with $rated\_as\_adversarial$ and 3886 with $rated\_as\_adversarial = False$.

We calculate the $ratio$ of the $False$ values ($Number\ of\ False/Total\ Number$), which is 0.517. This is only a very slight imbalance regarding $True$ and $False$ values, and we, therefore, do not need to balance the dataset.

Finding the Model

We use the $statsmodels$[4] python package to calculate the logit regression models. Our goal is to create multiple models using different variables and compare them with each other. We split the data into a training and a test set with a ratio of 0.35. Splitting the data allows us to evaluate the different models by calculating performance metrics on data the models were not trained on. All models use the same train data for fitting and test data for evaluation. After the best model is found, we retrain the model on the entire dataset to avoid losing valuable data points. The $statsmodels$ package internally uses the $patsy$[5] package, which allows us to express a statistical model using R-Style formulas. An example of such an R-Style formula is shown in Formula (6.1).

$$rated\_as\_adversarial \sim ssim + fmr \tag{6.1}$$

[4]https://www.statsmodels.org/stable/index.html
[5]https://patsy.readthedocs.io/en/latest/overview.html

FMR Category	Contrast Coding System	1-Hot Approach
*FMR=0.001	[0 0]	[1 0 0]
FMR=0.0001	[1 0]	[0 1 0]
FMR=0.00001	[0 1]	[0 0 1]

Table 6.1: CONTRAST CODING SYSTEM VERSUS 1-HOT APPROACH. Comparing the encoding produced by a contrast coding system with the encoding produced by the 1-hot approach. * marks the reference level in the contrast coding system.

This formula is read as follows: the dependent variable $rated\_as\_adversarial$ is to be described by the independent variables $ssim$ and fmr. The $\sim$ symbol means that the variable to its left is described by combining the variables to its right. We can include individual independent in the formula by stringing them together with the $+$ operator. Passing such an R-style formula to the $statsmodel$ package allows us to extract a logit regression model to which we can then fit the data. Categorical variables are automatically translated into dummy variables, for which a contrast coding system is used.[6] A category with N values is thereby translated in a sequence on N-1 variables. The remaining variable is used reference level, against which all the other values from the same category are compared. The dummy variables are encoded similarly to 1-hot vectors, with the difference that the encoding misses one value. The difference in the encoding is visualized in Table 6.1 for different False Match Rate categories. For each category, we determine the variable that is used as a reference level. Discrete variables are used without any conversion. In addition, we can provide a weighting for the $True$ and $False$ values of the $rated\_as\_adversarial$ variable to rebalance the data in case the selection of the training data results in an imbalance. The models are calculated from an expression on the training data and are then evaluated on the test data. The model is fitted to the data using the iterative reweighted least square (IRLS) method. In the IRLS method, the coefficients of the independent variables are adjusted in an iterative process so that the least squares are minimized. We obtain a confusion matrix from the calculated model by evaluating the fitted model against the test data. Using the confusion matrix, we calculate the misclassification rate. The misclassification rate is a performance metric that measures how many predictions were correct without distinguishing between false matches or false non-matches and is calculated according to Equation (6.2).

$$misclassification\_rate = \frac{false\ matches + false\ non\text{-}matches}{length(test\ set)} \tag{6.2}$$

In addition, the $statsmodels$ package provides a summary with typical statistical metrics where we focus on the p-values and coefficients of the individual independent variables. The $pseudo\text{-}R^2(\rho^2)$ is a further metric that allows us to evaluate the models. The ρ^2 value is a statistical metric for logit regression models that measures how well the model fits the data. Values range from 0 to 1, 0 representing no fit and 1 representing a perfect fit. Values from 0.2 to 0.4 are considered an excellent fit (McFadden, 1977). These values allow us to compare the different models. The models' ρ^2 values and p-values are denoted in the model statistics, which we reference in Table 6.3.

A list of the independent variables, including type, description and possible values, can be found in Table 6.2. First, we create three simple models and compare their performance. We want to predict $rated\_as\_adversarial$ once using the SSIM score, once using the MSSSIM score and the last time using both similarity metrics. Table 6.3 shows that the misclassification rates and ρ^2 values are nearly identical. Comparing these metrics, no model is yet to be favored.

Next, we extend the first three models with the remaining independent variables. In addition

[6] https://www.statsmodels.org/dev/examples/notebooks/generated/contrasts.html

Variable	Type	Description	Possible Values
$ssim$	Discrete	SSIM score	0.0 - 1.0
$msssim$	Discrete	MSSSIM score	0.0 - 1.0
$targets$	Discrete	Number of targets	1 - 20
$type$	Categorical	Type of experiment (Table 5.3)	random* include_source ff (=female $\rightarrow$ female) fm (=female $\rightarrow$ male) mm (=male $\rightarrow$ male) mf (=male $\rightarrow$ female) min min_group min_group_include max max_group blurry high_quality sharp
fmr	Categorical	False Match Rate	FMR0.001* FMR0.0001 FMR0.00001
$temporal\_adv$	Discrete	Number that indicates at what stage the adversarial image was displayed for a participant	0 - 80
$temporal\_orig$	Discrete	Number that indicates at what stage the original image was displayed for a participant	0 - 80
$user\_sex$	Binary	Sex of participant	0 (=male) 1 (=female)
$user\_age$	Discrete	Age of participant	-
	* These variables are defined as reference level and therefore do not have own p-values and coefficients		

Table 6.2: INDEPENDENT VARIABLES. List of independent variables used in the logit regression model with their type, description and possible values.

MR	ρ^2	Model Expression	Model Statistics
0.3222	0.0923	$rated\_as\_adversarial \sim ssim$	Appendix A.4.1
0.3260	0.0919	$rated\_as\_adversarial \sim msssim$	Appendix A.4.2
0.3226	0.0930	$rated\_as\_adversarial \sim ssim + msssim$	Appendix A.4.3
0.3279	0.1026	$rated\_as\_adversarial \sim ssim + other\_variables$	Appendix A.4.4
0.3279	0.1043	$rated\_as\_adversarial \sim msssim + other\_variables$	Appendix A.4.5
0.3275	0.1044	$rated\_as\_adversarial \sim ssim + msssim + other\_variables$	Appendix A.4.6

Table 6.3: MODEL EVALUATION. Different misclassification rates (MR) and *pseudo-*R^2 (ρ^2) values for different model expressions. The variable $other\_variables$ substitutes the following: $targets + type + fmr + temporal\_adv + temporal\_orig + user\_sex + user\_age$.

to the similarity metrics, we consider the False Match Rate, the type of experiment, how many people were targeted simultaneously, the learning effect, as well as the gender and age of the participants. Evaluating the extended model yields similar results as the basic models. The misclassification rates and ρ^2 scores are nearly identical but with a slight advantage for the model that considers both similarity metrics besides the other independent variables. The misclassification rates are relatively high for all models, ranging between 32.22% and 32.79%. However, if we compare this with the correct classification rate on the unaltered images (66.22%), respectively, their misclassification rate of 33.78%, the values seem to be plausible. The statistics of all the models can be found in the appendix, as indicated in Table 6.3.

A common approach for model selection is to include or exclude variables from a starting model until the p-values of the individual coefficients are no longer significant. This would be the right approach if we wanted to generate a model that should have predictive power in the future. However, we only want to evaluate our collected data with respect to all properties. Therefore, we will not discard variables that have p-values that do not imply statistical significance. Despite non-significant p-values, the variables serve a purpose. While we cannot make direct statements about them, they allow us to control the model with respect to them. It is important that the variables for which we want to make direct statements, namely the similarity measures, remain statistically significant.

For these reasons, we choose a model that includes all variables. We can observe a very high correlation of 0.977 between the SSIM and the MSSSIM score in the data. The high correlation of the two similarity metrics suggests that we use a model which only includes one similarity measure. In general, all the models are very similar in their performance, and choosing any of those models would be an adequate fit for evaluating the questionnaire result. The range of the SSIM scores is slightly larger than the range of the MSSSIM scores, which may simplify the evaluation. In addition, the PASS score was used in the original LOTS, which can be translated into an SSIM score in our use case and thus enables comparability. For these reasons, we will use the SSIM-only model that includes the other variables described as R-Style formula in Formula (6.3).

$$\begin{aligned} rated\_as\_adversarial \sim\ & ssim + targets + type + fmr \\ & + question\_temporal\_idx + user\_sex + user\_age \end{aligned} \tag{6.3}$$

After deciding on the model, it is trained once on the whole dataset before using it for the evaluation. The statistics of the final model can be found in the Appendix A.4.7. The final model reaches an ρ^2 score of 0.1106, which indicates a slightly better fit to the data than the models we used in the decision process. However, the score is still below the 0.2 to 0.4 range, which indicates that our model does not excellently fit the data. The ρ^2 score suggests that the data is chaotic, which is again confirmed by the low correct classification rate on the original images.

Extracting Statements from a Model

To explain how to extract statements from a model, we introduce a simplified model defined by the R-Style formula in Formula (6.4). Although we have already selected the final model, the resulting equations would be too long, so we use the simple model for illustrative purposes. The principle is transferable to the final model.

$$rated\_as\_adversarial \sim ssim + fmr + user\_age \tag{6.4}$$

We want to predict the probability of $rated\_as\_adversarial$ using the SSIM score, the False Match Rate as a category, and the participant's age. After creating a model using the R-Style formula and fitting it to the data, we can extract an intercept and coefficients for all the input variables.

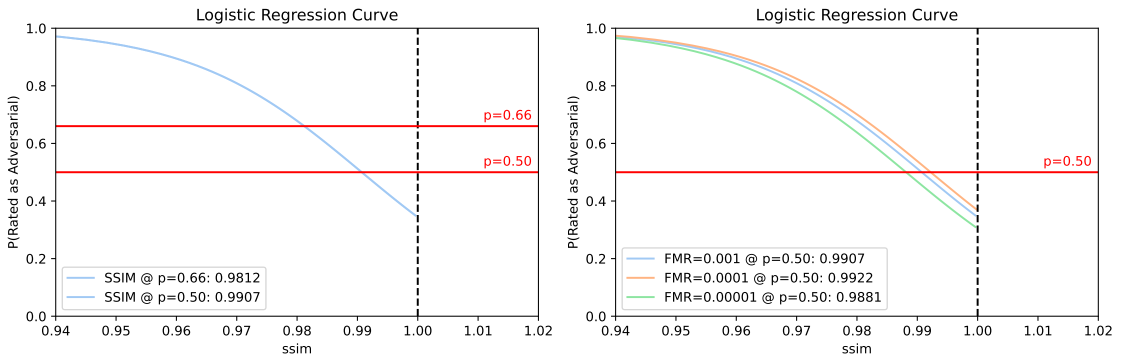


Figure 6.6: LOGISTIC REGRESSION CURVES. The left plot shows the relation of the SSIM score on the probability of an image being rated as adversarial. The right plot shows the influence of different False Match Rates on the SSIM score at p=0.5. These plots were created using a simplified model and serve to illustrate the interpretation of the plots.

With the intercept and the coefficients, we can form a linear equation according to Equation (6.5). We use the average from the dataset for each discrete variable to solve the equation. For categorical variables, no average value exists. We have to calculate how often the specific value occurs within its category and express this with a percentage value. If we insert these values and solve Equation (6.5), we can use the result and the formula in Equation (6.6) to calculate the probability of participants rating and average image as adversarial. It is important to note that one value per category is always assumed as the reference level and has no coefficient of its own. For example, if we want to know the influence of FMR=0.001, which is the reference level, we know from Table 6.1 (first row) that we have to use 0 as coefficients for FMR=0.0001 and FMR=0.00001. This means the influence of FMR=0.001 on the outcome of the equation is already embedded within the model's intercept. Therefore it does not need a coefficient of its own.

$$\begin{aligned} y = intercept + coef_{ssim} \cdot ssim + coef_{FMR0.0001} \cdot FMR0.0001 \\ + coef_{FMR0.00001} \cdot FMR0.00001 + coef_{user\_age} \cdot user\_age \end{aligned} \tag{6.5}$$

$$P(rated\_as\_adversarial) = \frac{e^y}{1+e^y} = \frac{1}{1+e^{-y}} \tag{6.6}$$

Instead of the average value, we can also define specific values to use. For example, we can assume 0.99 for the SSIM score, and then calculate the probability that a participant rated an image with an SSIM score of 0.99 as adversarial. If we also want to consider categorical variables, we can enable them by setting the desired type to 1 and inserting 0 for all other dummy variables of the same category. For example, we want to know the probability of a participant rating an adversarial image as True for an image with an SSIM score of 0.99 and a False Match Rate of 0.0001. We have to set FMR0.0001 = 1 and FMR0.00001 = 0 in Equation (6.5) and can then solve again for the probability with Equation (6.6).

However, we do not want to calculate probabilities with our model but determine the influence of the various variables on the SSIM score. For this purpose, the Equation (6.5) can be solved for $ssim$. We can use average values or specific values for the remaining discrete values as before. We can also activate dummy variables or use occurrence ratios for categorical values as before. What is new is that we need a value for y as well. Therefore we assume a probability and solve Equation (6.6) with regards to y. We now have all the tools to extract arbitrary statements about the SSIM score. In addition, we can iterate over probability values between 0 and 1 to generate a logistic regression plot given that we fix the other variables.

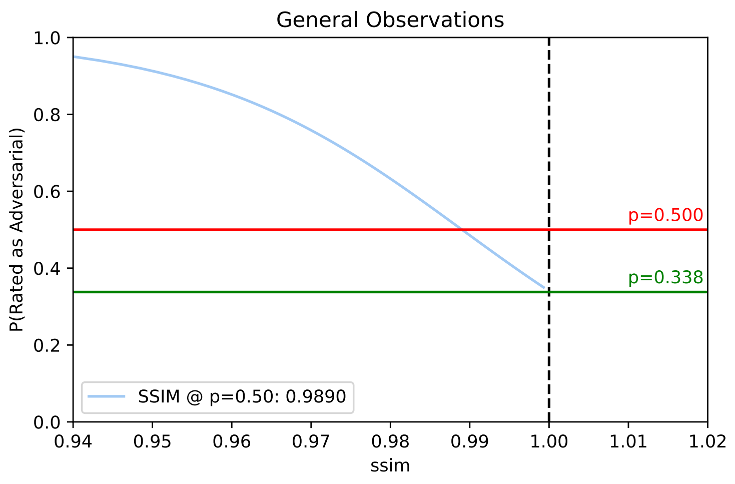


Figure 6.7: INFLUENCE OF SSIM SCORE. The influence of the SSIM score on the probability of an average adversarial image being recognized as such.

Figure 6.6 shows two such plots. The left plot shows the relation between the SSIM score and the probability of an image being classified as adversarial. The red horizontal lines mark different probability levels. For example, the plot on the left shows that with an SSIM score of 0.9907, half of the participants would recognize an adversarial as such.

The black line at $x = 1.0$ marks the limit of the SSIM score. The point at which the curve intersects the black line reflects the participants' performance on unaltered images.

We can also show the influence of the SSIM score on the probability that an image is rated as adversarial, considering different categories. The plot on the right shows the influence of the SSIM score, taking into account the False Match Rate. It can be seen that a False Match Rate of 0.001 requires a lower SSIM score than a False Match Rate of 0.0001 to trick the same number of participants (50%). The two plots in Figure 6.6 were generated for illustrative purposes only. The evaluation of the full data with the final model follows in the next section.

6.2.4 Findings

General Findings

For general statements about the SSIM score, we will first look at different probability levels and use the average values for the remaining values. Figure 6.7 shows which SSIM scores are necessary to achieve the different probability levels for an average image. In general, the lower the probability level, the more challenging the task since we measure the probability of detecting an adversarial.

For the rest of the section, we will abbreviate the probability that an image is classified as adversarial by p. If we look at the SSIM score at $p = 0.5$, we could say that a score higher than 0.9890 is necessary to perform better than a random guess. However, it should be noted that the correct classification rate on the unaltered images was also only 66.2% which corresponds to a probability of $p = 0.338$ (green level). Considering the participants' performance on the unaltered images, we can assume a probability level of $p = 0.5$ as an above-average guess since it is only 16.2% above the level obtained on the unmodified images.

For further analysis, we can keep these two probability levels. Whereby a level of $p = 0.5$ corresponds to an above-average guess, and the level of $p = 0.338$ corresponds to the benchmark performance on the unaltered images.

Demographic Related Findings and Learning Effect

Next, we can examine the influence of the participants' demographics and the learning effect on the SSIM scores. The two temporal variables we introduced earlier allow us to show the learning effect over time. The plot at the top left of Figure 6.8 shows that the probability that an image is classified as adversarial for adversarial images increases over time, and for original images, it decreases. Thus, an opposite effect is visible, which underlines that the classification performance of the participants increases over time. The curves in the plot were sampled at 1000 different probability levels each. We can further observe the learning effect on the plot in the upper right corner of Figure 6.8. The curves for the first question for adversarial images (green) and original images (blue) lie on top of each other, indicating the same performance. For the 75th question, a better score is needed for adversarial images (red) to reach the same probability level as for the first question, which shows that it becomes harder to trick the participants. Analogously, it can be observed that for the original images, the score needed to correctly classify an image with the same score as an adversarial has decreased, which is equivalent to a higher probability of classifying an image as an original. Furthermore, the curve of the original images for the 75th question lies below the $p = 0.338$ benchmark level, indicating that the participants achieved better classification performance after 75 questions than the average performance on original images, which is to be expected when assuming a learning effect.

The plot at the bottom left of Figure 6.8 shows a very slight difference between the two genders, with a slightly higher SSIM score being required for women to reach the same probability level. Thus, for women, slightly better adversarial images are needed to trick them. However, the difference is diminishing small.

The bottom right plot of Figure 6.8 shows how the different age categories affect the SSIM score for an equal probability level. There are no significant differences between the various age groups. Therefore this factor can be neglected.

Experiment Type Related Findings

When investigating the type of experiment, it is crucial to fix the number of attacked targets to 4 and the False Match Rate to 0.001 since most experiments regarding the type were performed under these conditions exclusively.

We can compare the random target selection, where the properties of the source and the targets are irrelevant, with the random experiments, where the properties are irrelevant, but the source is also included as a target. Figure 6.9 (top left) shows that the include source experiment requires a slightly lower SSIM score to reach the same probability level. Even though the difference is minor, including the source might help hide the perturbations and make them less perceivable.

If we look at the experiments regarding the cosine distances between source and targets, we see that samples with minimum distances between source and the respective targets and experiments with minimum distances between source, the targets and the targets themselves perform best (Figure 6.9, top right). Since the source and the targets are already close to each other from the beginning of the experiment, less perturbations are needed on average to reach the targets. Interestingly, in experiments with maximal distances between the source and the targets and between the source, the targets, and within the targets, the SSIM scores required do not have to be significantly better than for minimal distances. In general, better adversarial images are generated when the source and target are close to each other in terms of cosine distance.

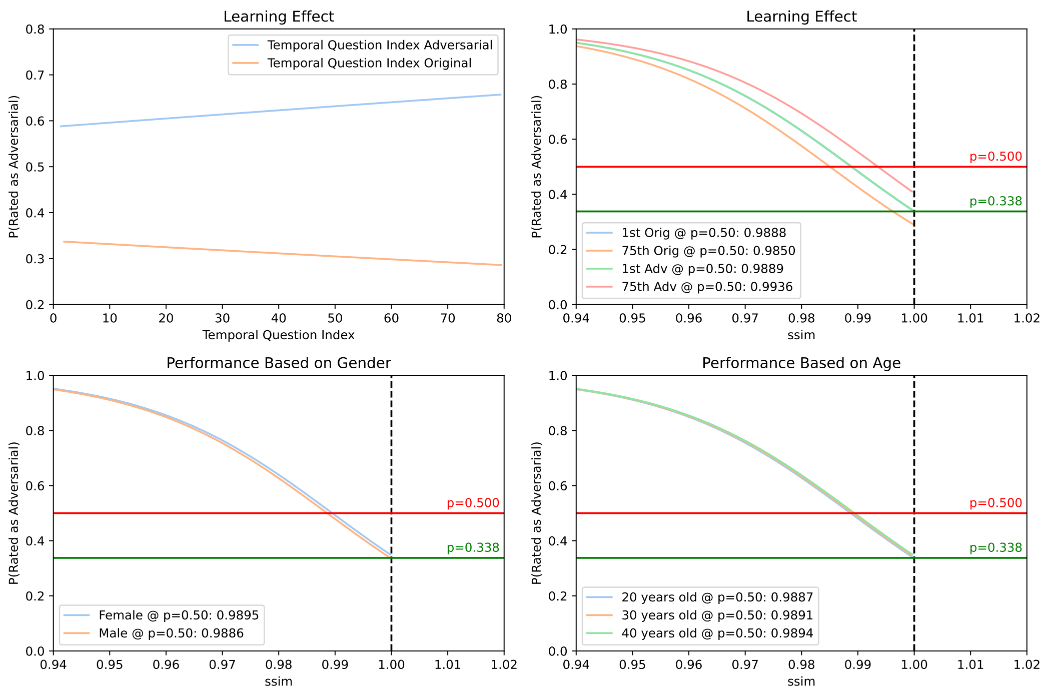


Figure 6.8: Influence of Learning Effect and Demographics. The influence of the learning effect and the participants' demographics on the SSIM score at given probability levels.

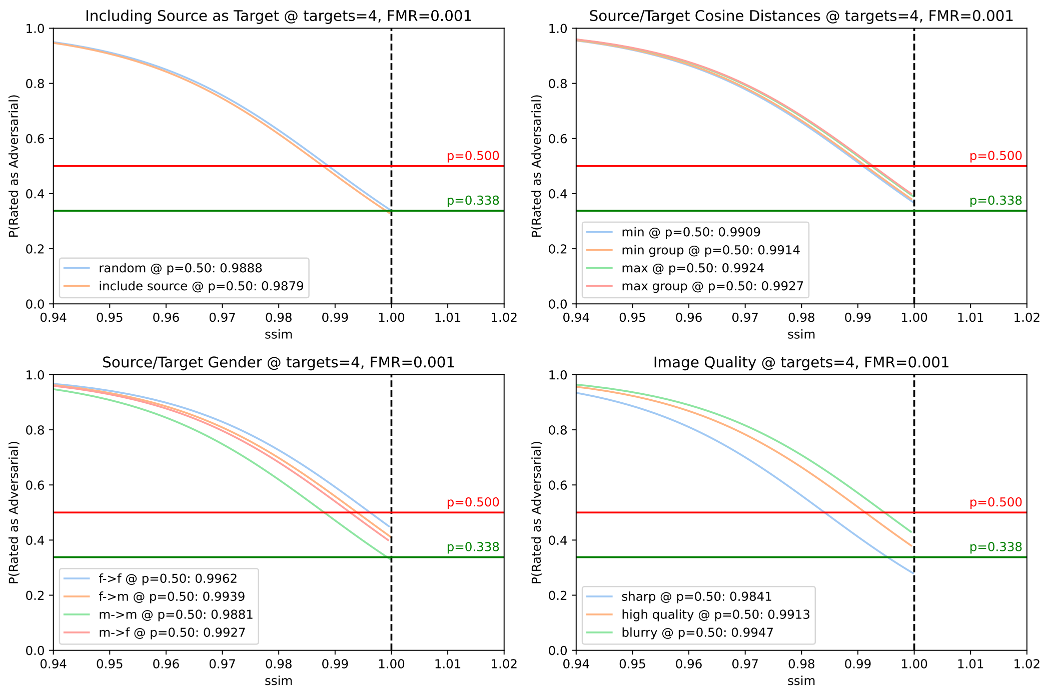


Figure 6.9: Influence of Experiment Type. The influence of the type of experiment on the SSIM score at given probability levels.

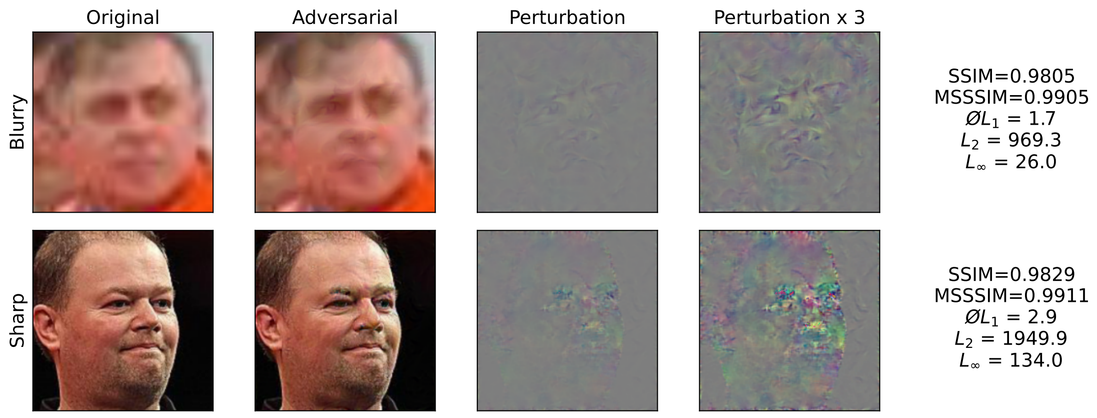


Figure 6.10: SHARP VERSUS BLURRY IMAGES. Comparison of perturbations in sharp and blurry images. Although the sharp image contains significantly more perturbations, they are much more difficult to detect by eye than in the blurry image. In both cases, 4 targets were attacked simultaneously.

The experiments investigating the gender of the sources and the targets yield interesting results (Figure 6.9, bottom left). It is evident that when the sources are male, lower SSIM scores have to be achieved than when the sources are female, given a certain probability level. The difference between only male source and targets versus only female source and targets is the largest. It is therefore generally easier to attack when the sources are male. Ultimately, the easiest attack is when the source and targets are male.

The image quality also significantly influences the required SSIM scores (Figure 6.9, bottom right). With blurred images, it is particularly difficult to trick the participants. Perturbations are most often generated in areas where major landmarks reside (e.g., eyes, mouth, nose) or around the edges of the face. Moreover, no mechanism ensures that the generated perturbations fit the original image in terms of focus/blurriness. When adding the perturbations to a blurry original image, the difference in focus might make them more notable, especially in the locations of major landmarks. In sharp images, on the other hand, there are often more perturbations present, but these can be better hidden in the adversarial image. Figure 6.10 shows that the perturbations are much easier to detect in the blurry image than in the sharp image, even though the latter contains significantly more perturbations. Therefore, a lower SSIM score is necessary for sharp images to trick people since the perturbations can be easily hidden. This is further emphasized by the fact that the curve of the sharp images intersects the $p = 0.338$ benchmark level at an SSIM score of 0.9959, which indicates that with sharp images and an SSIM score above 0.9959, a better performance than on the unaltered original images is achievable. The focus properties of the high-quality images lie between the blurry and the sharp images, which is reflected in the required SSIM score for an equal probability level, which lies between the level of blurry and sharp images.

Attack Difficulty Related Findings

When we want to look at the attack difficulty based on the False Match Rate, we also need to fix the number of targets and the type of experiment. We choose 2 targets and a random source-target selection experiment.

When we want to look at the attack difficulty based on the False Match Rate, it is evident that the more difficult attack with FMR=0.0001 needs higher SSIM scores than the attack with

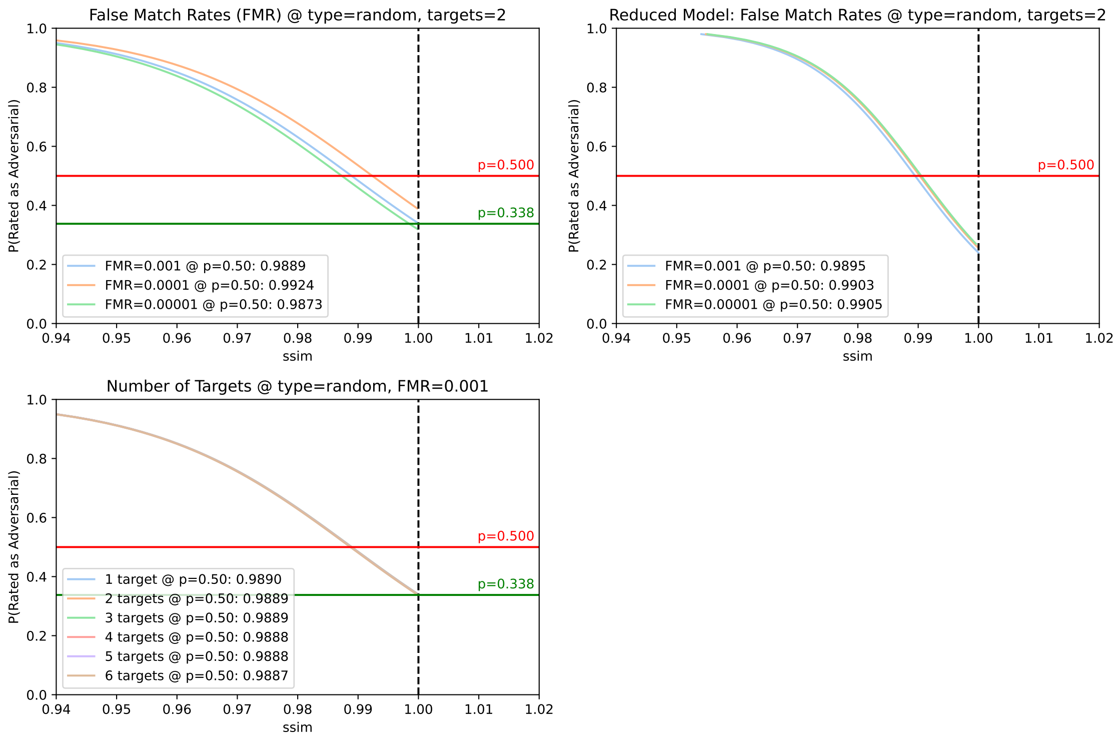


Figure 6.11: INFLUENCE OF ATTACK DIFFICULTY. The influence of the attack difficulty on the SSIM score at given probability levels. The plot in the top right was generated using a reduced model according to this R-Style formula: $rated\_as\_adversarial \sim ssim + fmr$, fitted to a reduced dataset only containing random experiments at 2 targets.

FMR=0.001 (Figure 6.11, top left). This can be explained by the fact that more difficult attack requires lower cosine distances between source and targets, and therefore more iterations are needed, and more perturbations are present in an image. If an image contains many perturbations, the SSIM score must be higher to avoid detection of an adversarial. However, it is noticeable that the even more difficult attack with FMR=0.00001 requires SSIM scores below the attack with FMR=0.001. We can explain this since, for experiments with FMR=0.00001, only one or two targets were attacked simultaneously. Therefore, the resulting similarity scores were, on average, higher than for FMR=0.001 experiments, where more difficult experiment types are present, and more targets were attacked simultaneously. Because coefficients are fixed once the model is fitted to the data and calculated based on all samples observed, the model predicts lower SSIM scores needed at FMR=0.00001 for the same probability level.

By creating a reduced model that only uses the SSIM score and the False Match Rate to predict the probability of an image being rated as adversarial and using a reduced dataset only containing samples of random experiments at 2 targets, we can observe the expected behavior. With increasing attack difficulty based on a lower False Match Rate, the SSIM scores required for the same probability level must be higher (Figure 6.11, top right). The reduced model's statistics can be found in Appendix A.4.8.

The number of targets we attack simultaneously does not affect the SSIM score required for the equal probability level (Figure 6.11, bottom left). This indicates that even though the attack becomes more difficult with an increasing number of targets, the SSIM score to trick people stays constant, which means that, in terms of targets, the size of the perturbations is decisive in determining whether an image is rated as adversarial or not.

Evaluation Conclusion

In conclusion, we can say that SSIM score is not equal to SSIM score. Although we can generally state that a specific score is sufficient for average images to trick a human with a certain probability, different SSIM scores are necessary depending on the circumstances. For an average image, we state that at an SSIM score of 0.9924, the probability of being rated as adversarial is 45%, at SSIM=0.9958, the probability is 40%, and at SSIM=0.9993, the probability is 35%, almost at the benchmark level of 33.8% on original images.

Furthermore, we demonstrated a learning effect, whereby participants correctly identified the original and the adversarial images at a higher rate over time. However, the demographics of the participants themselves did not significantly affect the required SSIM scores.

We observed blurry images requiring higher SSIM scores than sharp images at a given probability level. Furthermore, sharp images achieve a better result at a given SSIM score in terms of the probability of being detected. However, as noted in Chapter 5, sharp images have lower SSIM scores than blurry images on average. Therefore, we can conclude that more perturbations can be included in sharp images without them being noted by the participants.

In Chapter 5, we show that higher SSIM scores are obtained when creating $f \to f$ adversarial images compared to $m \to m$ samples. In the questionnaire evaluation, however, we saw that significantly lower SSIM scores are required for $m \to m$ experiments than for $f \to f$ experiments at the same probability level. The effects, therefore, cancel each other out. The $f \to f$ experiments generate higher SSIM scores on average but also require higher SSIM scores at a given probability level.

We have further shown that in experiments in which the source and targets have a high distance, better SSIM scores are required at a given probability level. In the experiments in Chapter 5, we also observed that the minimum distance experiments produce better SSIM scores than their maximum distance counterparts. The two effects add up. In experiments with minimum distances, better SSIM scores are achieved on average in adversarial image generation, and lower SSIM scores are needed for a given probability level.

In all plots, we can observe falling curves. This means that the probability of an adversarial image being exposed decreases with smaller perturbations, respectively higher SSIM scores. If the SSIM score were a perfect measure to predict human perception, the SSIM scores required at a given probability level would be identical, regardless of experiment type and attack difficulty. All curves would lie on top of each other. Since this is not the case, we conclude that the SSIM is not a perfect measure that can independently determine whether human perception can be tricked. The circumstances must always be taken into account. Nevertheless, the SSIM is an excellent indicator and point of reference, especially if we consider an average attack.

Chapter 7

Discussion

7.1 Revisit Extended LOTS

In the empirical evaluation, we decided to omit the MSSSIM score because of its high correlation with the SSIM score. In order to find out whether this is advantageous not only for the evaluation but also for the algorithm itself, we want to re-run all experiments from Chapter 5 but remove the MSSSIM component from the loss. For this purpose, we performed a hyperparameter search analogous to Section 5.1 and decided to use $width_{step} = 2, w_{cos} = 1.0, w_{ssim} = 1.5, w_{msssim} = 0.0$. Subsequently, we executed all experiments again, using the new set of hyperparameters. In the appendix, we present the results and the comparison to the original experiments in Table A.11. Across all experiments, we raised the average SSIM score from 0.9837 to 0.9863. The success rate has decreased from 81.40% to 79.65%. This is expected since the SSIM score is weighted higher than before, and the MSSSIM loss component is omitted. With the removal of the MSSSIM loss component, the average MSSSIM scores have dropped from 0.9931 to 0.9891. The perturbations generated look significantly different, with a greater focus on the areas instead of edges. This was also expected since we had investigated the different influences of SSIM and MSSSIM on the perturbations in Section 4.3.5 and observed a similar pattern. Figure 7.1 compares the an experiment using the same source-target constellation, once with the original hyperparameters and once with the new ones. Whether this adjustment is a real improvement has to be judged from image to image. As expected, the SSIM scores increased, and the MSSSIM score decreased. There are fewer perturbations overall in the original experiment, as indicated in the $\varnothing L_1$ and L_2 norms. The maximum perturbation per pixel is lower in the experiment using the new hyperparameters, as indicated in the L_∞ norm. Additionally, the perturbations in the cheek region and between the eyes are slightly less visible. We think that perturbations focusing on areas instead of edges, corresponding to higher SSIM loss weights, might be less visible.

We can state that the best results are obtained when the hyperparameters are adjusted individually for each adversarial image to be synthesized. For example, Figure 7.2 shows an experiment with the same source-target constellation, once conducted with the standard hyperparameters and once with empirically selected hyperparameters (finetuning). Although the $\varnothing L_1$ norms are similar in the finetuning experiment and L_2 and L_∞ are even larger, the perturbations are less visible. The SSIM and MSSSIM scores are significantly higher in the finetuning experiment, which hides the perturbations better. However, empirically selecting the hyperparameters for each sample in a large-scale experiment would be too time-consuming, which is why we recommend using the default parameters if numerous adversarial images should be generated at once.

Under perfect conditions, we can achieve very impressive results. The first row of Figure 7.3 shows an adversarial attack on 20 targets, where additionally, the source is included in the list of targets. We determined the hyperparameters empirically and sampled the targets from

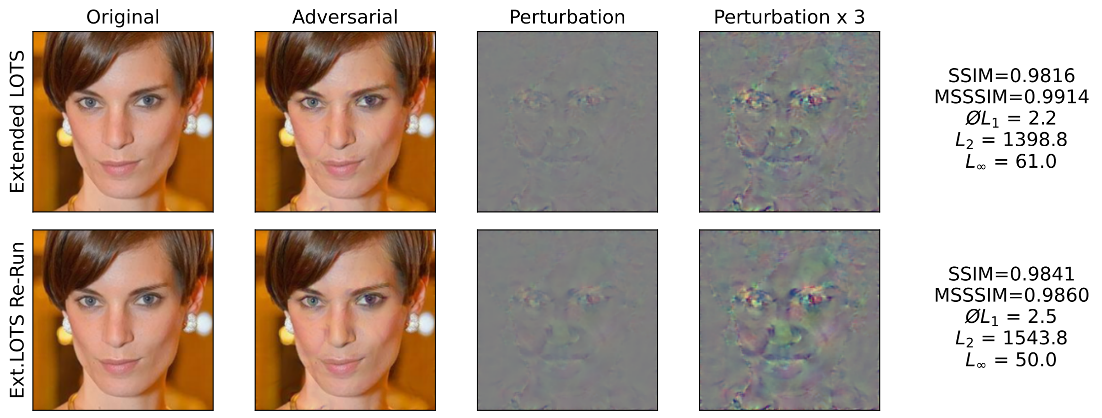


Figure 7.1: EXTENDED LOTS VERSUS EXTENDED LOTS RE-RUN. Adversarial images and perturbations were generated using the same source and targets, once using the original hyperparameters and once the new hyperparameters, in which only the SSIM similarity loss component is present. Four targets were attacked simultaneously. Hyperparameters Extended LOTS: $width_{step} = 3, w_{cos} = 1.0, w_{ssim} = 0.9, w_{msssim} = 1.0$, Hyperparameters Extended LOTS Re-Run: $width_{step} = 2, w_{cos} = 1.0, w_{ssim} = 1.5, w_{msssim} = 0.0$.

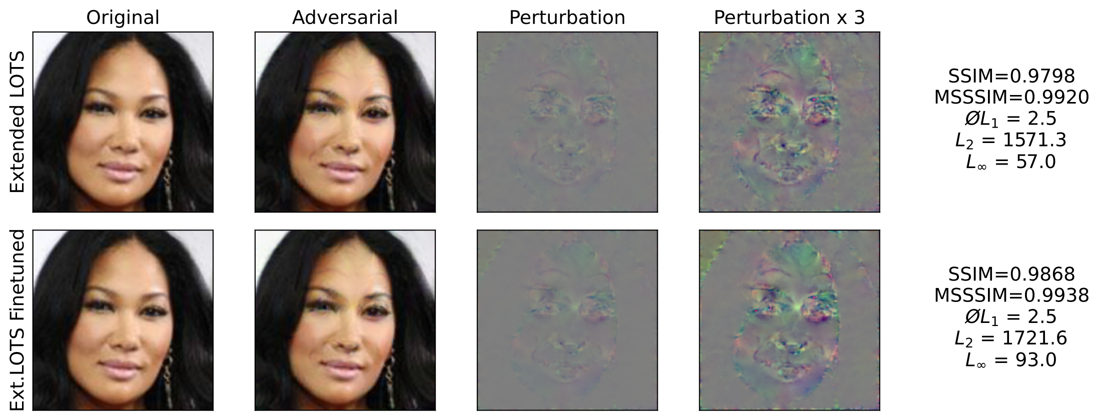


Figure 7.2: EXTENDED LOTS VERSUS EXTENDED LOTS EMPIRICAL FINETUNING. Adversarial images and perturbations were generated using the same source and targets, once using the original hyperparameters and once empirically selecting hyperparameters. Five targets were attacked simultaneously. Hyperparameters Extended LOTS: $width_{step} = 3, w_{cos} = 1.0, w_{ssim} = 0.9, w_{msssim} = 1.0$, Hyperparameters Extended LOTS Finetuning: $width_{step} = 1, w_{cos} = 1.0, w_{ssim} = 2.2, w_{msssim} = 2.0$.

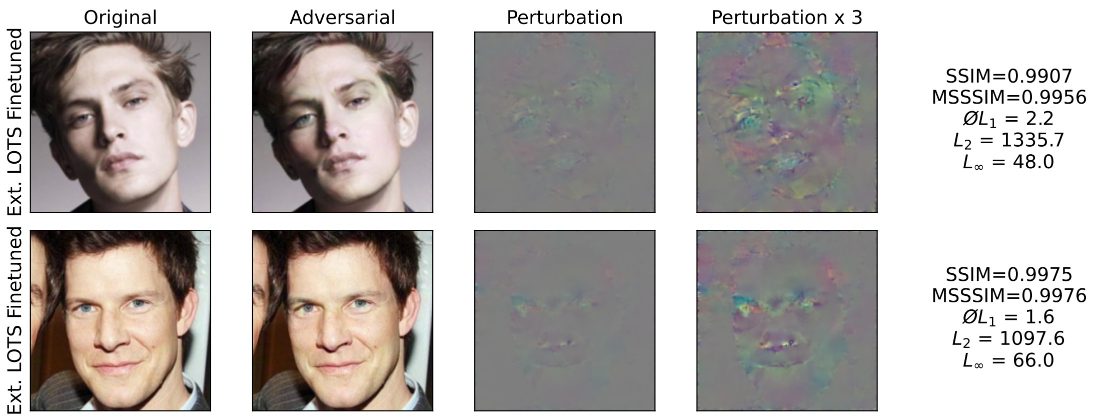


Figure 7.3: EXTENDED LOTS EMPIRICAL FINETUNING. First row: Experiment attacking 20 targets with empirically finetuned hyperparameters. Targets are sampled form a $min\ group$ and source is included in the list of targets. Hyperparameters: $width_{step} = 1, w_{cos} = 1.0, w_{ssim} = 2.2, w_{msssim} = 2.0$
Second row: Experiment attacking 4 targets with empirically finetuned hyperparameters. Targets are sampled form a $m \rightarrow m$ experiment and a sharp image is chosen as starting point of the synthesis. Hyperparameters: $width_{step} = 1, w_{cos} = 1.0, w_{ssim} = 15.0, w_{msssim} = 5.0$.

a $min\ group$ experiment. Although the L_p norms are relatively high, the SSIM and MSSSIM scores are excellent, and therefore the perturbations are barely visible. We have learned from the evaluation of the questionnaire that $m \rightarrow m$ experiments and sharp images require lower SSIM scores and are especially good at hiding perturbations. In the second row of Figure 7.3, we selected male sources and 4 male targets and chose a sharp image as a starting point for the synthesis. We empirically selected the hyperparameters and achieved extraordinarily high SSIM and MSSSIM scores. This shows that when knowing the peculiarities of the source-target constellation, and if the situation allows for free source-target selection, high-quality adversarial images can be generated.

Ultimately, the quality of the adversarial sample must be evaluated on a case-by-case basis, and we cannot rely on scores and norms alone. From time to time, artifacts occur scores cannot express. For example, the color of the subject's left eye in Figure 7.1 changes during synthesis. Therefore, it is up to the user to assess which adversarial samples are suitable for an attack and which samples should be discarded. However, we can say that for attacks on one to two targets, the results are excellent in the vast majority of cases, and the perturbations are hardly visible.

7.2 Transferability of Adversarial Attacks

The transferability of adversarial attacks is an essential characteristic. It describes whether and how well an attack created on a specific network and dataset can be transferred onto another network. Especially in black-box attacks, where the underlying network architecture and datasets used are unknown, transferability is an important property since it is the only way to execute an attack. Transferability from one network to another is easy when both networks use the same architecture and datasets and difficult when the network under attack was trained on different datasets and used another architecture (Vakhshiteh et al., 2021). Sharif et al. (2019) were able to show that dodging attacks, in which they attempt to enforce a misclassification of a person to any arbitrary person, could be transferred from one network to another. For the attack, eyeglasses are either digitally placed on subjects' faces or physically printed out and worn by the subjects. The digital attack had a transferability of 63% from the VGG architecture to the OpenFace architecture and 10% in the other direction. Thus, the transferability properties were not symmetric. In the physical attacks, the transferability from VGG to OpenFace was still 44% and from OpenFace to VGG 28%. The dodging experiment is an untargeted attack. Furthermore, the authors performed a targeted impersonating attack but did not publish its transferability properties, which might indicate a lack of success. Zhong and Deng (2020) presented a method for transferability enhancement and tested it on numerous networks, datasets, and various loss functions. They examined a targeted attack in which they generated an adversarial image to achieve misclassification of a person to another specific person. The authors found that feature-level attacks are more transferable than label-level attacks. They have also shown that their method can be used for successful black-box attacks on different networks. The VGGFace2 network was a better than average source network to generate transferable attacks on other networks. Liu et al. (2017b) have studied the transferability of targeted and untargeted adversarial attacks. They have shown that targeted attacks are less transferable than untargeted attacks. The authors then presented a novel approach in which model ensembles are used to generate adversarial images. They designed a black-box attack, in which they query labels for objects from Clarifai.com. For reference, they also created adversarial images with a VGG-16 network. Of the adversarial images created using their approach, 18% were classified from Clarifai.com with a label close to the target. In addition, 57% of the targeted attacks produced a misclassification (untargeted). For the adversarial images using the VGG-16 network, 2% were classified with a label close to the target. However, 76% of the samples produced misclassifications.

From these works, we can conclude that transferability is possible up to a certain point. The most suitable attacks are untargeted attacks, which aim to achieve a misclassification. However, there is no guarantee of success even with untargeted attacks, and performance is highly dependent on the selected networks and datasets. Also, targeted black-box attacks aiming to mimic the features of one target could be executed successfully by employing enhancement methods. We can assume very low transferability if we draw conclusions about our extended LOTS algorithm based on these works. Although the VGGFace2 network turned out to be a good source network, the transferability was limited even with only one attacked target. Since we are attacking multiple targets simultaneously, the probability is very low that the external network under attack will correctly classify all our targets. In order to achieve transferability on multiple targets, considering the current state of research, we have to execute a white-box attack. In white-box attacks, the architecture and the used dataset of the network under attack are known. However, there is no guarantee of a successful attack even under these circumstances.

7.3 Defenses Against Adversarial Attacks

There are various methods to protect a network against adversarial attacks. Adversarial samples can already be used in training to make the network more robust against attacks. It is also possible to deviate from the usual network architectures in order to prevent white-box attacks. Another defense possibility is the use of model ensembles, in which individual networks are specialized to detect adversarial samples. Tramèr et al. (2017) presented Ensemble Adversarial Training, in which the training data is augmented with adversarial images generated by other networks. They show that their approach improves the network's robustness against black-box attacks. Tao et al. (2018) have presented a technique that examines and strengthens the connections between neurons and critical attributes for identification, resulting in an adversarial sample detection rate of 94%. Despite various defense strategies, adversarial attacks are also becoming more sophisticated. The defense strategies themselves are becoming victims of attacks. A good example is the

Defense Distillation technique (Papernot et al., 2015), which was defeated shortly after by Carlini and Wagner (2017).

If we assume that our extended LOTS attack will go undetected, other defenses are still in place. For example, suppose a person wants to use the extended LOTS method to generate a modified passport photo so that other people can travel with it for illegitimate purposes. In that case, the passport photo must first be printed on the passport and be injected into the face recognition system. In Switzerland, for example, this is not possible, as passport photos may only be taken by an authority and require physical presence. Therefore, digital adversarial attacks are not possible, at least not for the official passport. It is also difficult to execute a physical adversarial attack because, as mentioned, there are always authorities on site. In other countries, such as Germany, submitting a digital photo for a passport is still possible. This would make an adversarial attack with the extended LOTS more conceivable. However, it would still be necessary to know the details of the face recognition network, and different countries probably use different face recognition systems. For a successful attack, therefore, an attacker needs an authority that allows digital passport photos and knowledge of the internal details of the face recognition systems of all the countries to be visited. In addition, the level of difficulty increases with the number of people who are to use the passport simultaneously. Finally, it is still possible that the automated passport control is out of order, and the attacker has to go through a counter.

7.4 Measuring Human Perception

Furthermore, the question arises whether the SSIM score is suitable for measuring human perception with respect to the detection of perturbations. Nilsson and Akenine-Möller (2020) have noted that the SSIM score was not originally designed to measure human perception but is merely a measure of the structural similarity between two images. While people are particularly good at detecting structural differences between images, which is why the two similarity metrics are quite representative, there are undoubtedly other factors. Our empirical evaluation established a link between human perception and the SSIM. We concluded that the SSIM score could describe human perception to a certain degree. However, we have also shown that other factors influence the perceptibility of perturbations and that we cannot establish an SSIM threshold value above which perturbations are no longer detected. We have also seen that there is indeed a learning effect in recognition of adversarial samples. We tested at which score people recognize an adversarial image correctly within the 2 seconds in which it is shown. As the experiment progressed, higher scores were needed to trick people. Trained eyes would therefore be able to recognize adversarial samples under real conditions, at least if several targets were attacked simultaneously. In a single-target attack, the perturbations are diminishingly small that they are very unlikely to be detected. Andersson et al. (2020) propose FLIP as an alternative to the SSIM score since this technique better represents human perception and does not suffer from the problems described in Section 4.3.3. We have nevertheless used the SSIM score for this thesis to have a basis for comparison with the original LOTS work that uses the PASS score, which we can translate into the SSIM score.

Chapter 8

Conclusion

We presented the existing LOTS algorithm and showed the network and dataset used for this thesis. Our first contribution was to extend and improve the LOTS algorithm. We showed how the LOTS algorithm can be applied to multiple targets simultaneously and how the source of the synthesis can also be used as a target. Using the source as a target ensures that the face recognition network correctly recognizes the person's identity in the source image. We further extended the loss function by replacing the MSE loss with the cosine distance loss and adding an SSIM score and an MSSSIM score component, measuring image similarity. As a second contribution, we implemented different sampling methods to investigate the influences of the composition of sources and targets. We then conducted 48 quantitative experiments creating 4800 adversarial samples and evaluated them using the similarity metrics. We successfully generated adversarial images for all possible source-target constellations, whereby the success decreases with increasing difficulty caused by attacking additional targets or using lower False Match Rates. In order to understand the influence of similarity metrics on human perception, we implemented a custom questionnaire tool in a third contribution and conducted an experiment with 73 participants. We evaluated and interpreted the results empirically using different logit regression models. We concluded that the SSIM score can be used to draw conclusions about human perception to a certain degree but that other influencing factors like the gender of the source and targets, image quality, or distances between source and targets have to be considered. It is impossible to define a fixed SSIM value above which people no longer recognize adversarial images. In addition, we found a learning effect from which we conclude that people can be trained to recognize adversarial samples. Finally, we re-ran our experiments to incorporate the findings of the empirical analysis into the algorithm. We concluded that choosing hyperparameters by manual experimentation on a per-attack basis leads to the best result. However, there is still no guarantee of success, but attacks on one or two targets produce adversarial images in which the perturbations are hardly visible.

We see several ways in which this work could be extended in the future. On the one hand, more work could be invested in finding the best hyperparameters. In addition, the findings from the experiments could be taken into account, and the algorithm could be enhanced to adjust the hyperparameters during synthesis dynamically. The conversion speed and the source-target constellation could be considered to achieve better results automatically. We could also experiment with other loss functions, such as the ArcFace loss, and evaluate whether they can produce even better results (Deng et al., 2019). We have mentioned that the SSIM score is not necessarily the best measure to evaluate human perception. We could extend the loss with other metrics like the FLIP score or even use the score to replace the similarity metrics (Andersson et al., 2020). In addition, to achieve better transferability, we could apply the extended LOTS algorithm to the most common networks simultaneously, extract the feature vectors, and modify the input image to move in the right direction for all networks.

Appendix A

Attachments

A.1 MAAD-Attributes

MAAD-Attribute	is present	is absent	is undefined
Male	1958913	1349127	0
Young	1250114	989321	1068605
Middle_Aged	354968	2395142	557930
Senior	260687	3013551	33802
Rosy_Cheeks	33990	2321058	952992
Shiny_Skin	581133	1110002	1616905
Bald	207554	3004817	95669
Wavy_Hair	856616	2193351	258073
Receding_Hairline	513859	1948374	845807
Bangs	355048	2701346	251646
Sideburns	1097130	2198368	12542
Black_Hair	514619	2067750	725671
Blond_Hair	347723	2574286	386031
Brown_Hair	817910	1196846	1293284
Gray_Hair	316839	2839278	151923
No_Beard	2108546	466498	732996
Mustache	16629	2629842	661569
5_o_Clock_Shadow	434288	1834468	1039284
Goatee	9229	2655062	643749
Oval_Face	466869	793888	2047283
Square_Face	1709811	1585311	12918
Round_Face	5905	2287232	1014903
Double_Chin	605454	2326091	376495
High_Cheekbones	857224	1328748	1122068
Chubby	406896	2410459	490685
Obstructed_Forehead	195722	2316315	796003
Fully_Visible_Forehead	1668763	845847	793430
Brown_Eyes	1303174	401359	1603507
Bags_Under_Eyes	917779	1367622	1022639
Bushy_Eyebrows	1071154	2119007	117879
Arched_Eyebrows	762116	1814707	731217
Mouth_Closed	139989	485079	2682972
Smiling	625844	1034713	1647483
Big_Lips	939155	1532764	836121
Big_Nose	503066	1202627	1602347
Pointy_Nose	1816441	1044887	446712
Heavy_Makeup	982666	2314175	11199
Wearing_Hat	256130	2946013	105897
Wearing_Earrings	992962	1961832	353246
Wearing_Necktie	350886	2162852	794302
Wearing_Lipstick	1126676	2138389	42975
No_Eyewear	2597310	199386	511344
Eyeglasses	339032	2854252	114756
Attractive	884429	2301934	121677

Table A.1: MAAD-ATTRIBUTES. MAAD-Attributes and their presence in the images of the VGGFace2 dataset.

A.2 Custom Questionnaire Tool

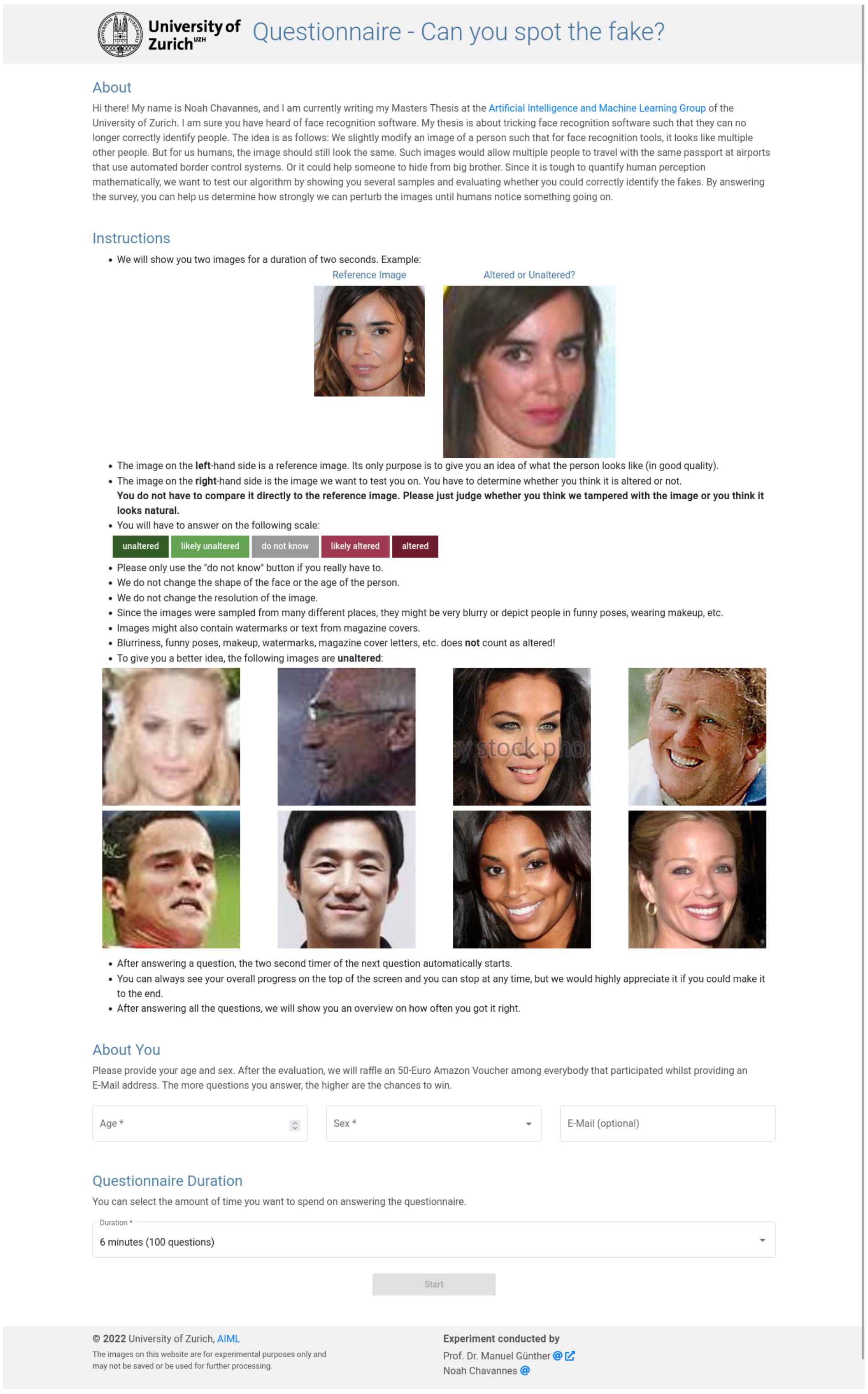
A.2.1 Landing page

Full Introduction Text

Hi there! My name is Noah Chavannes, and I am currently writing my Masters Thesis at the Artificial Intelligence and Machine Learning Group of the University of Zurich. I am sure you have heard of face recognition software. My thesis is about tricking face recognition software such that they can no longer correctly identify people. The idea is as follows: We slightly modify an image of a person such that for face recognition tools, it looks like multiple other people. But for us humans, the image should still look the same. Such images would allow multiple people to travel with the same passport at airports that use automated border control systems. Or it could help someone to hide from big brother. Since it is tough to quantify human perception mathematically, we want to test our algorithm by showing you several samples and evaluating whether you could correctly identify the fakes. By answering the survey, you can help us determine how strongly we can perturb the images until humans notice something going on.

Full Instructions

- We will show you two images for a duration of two seconds. Example: (See screenshot in Figure A.1)
- The image on the left-hand side is a reference image. Its only purpose is to give you an idea of what the person looks like (in good quality).
- The image on the right-hand side is the image we want to test you on. You have to determine whether you think it is altered or not. You do not have to compare it directly to the reference image. Please just judge whether you think we tampered with the image or you think it looks natural.
- You will have to answer on the following scale: (See screenshot in Figure A.1)
- Please only use the "do not know" button if you really have to.
- We do not change the shape of the face or the age of the person.
- We do not change the resolution of the image.
- Since the images were sampled from many different places, they might be very blurry or depict people in funny poses, wearing makeup, etc.
- Images might also contain watermarks or text from magazine covers.
- Blurriness, funny poses, makeup, watermarks, magazine cover letters, etc. does not count as altered!
- To give you a better idea, the following images are unaltered: (See screenshot in Figure A.1)
- After answering a question, the two second timer of the next question automatically starts.
- You can always see your overall progress on the top of the screen and you can stop at any time, but we would highly appreciate it if you could make it to the end.
- After answering all the questions, we will show you an overview on how often you got it right.



University of Zurich^UZH Questionnaire - Can you spot the fake?

About

Hi there! My name is Noah Chavannes, and I am currently writing my Masters Thesis at the Artificial Intelligence and Machine Learning Group of the University of Zurich. I am sure you have heard of face recognition software. My thesis is about tricking face recognition software such that they can no longer correctly identify people. The idea is as follows: We slightly modify an image of a person such that for face recognition tools, it looks like multiple other people. But for us humans, the image should still look the same. Such images would allow multiple people to travel with the same passport at airports that use automated border control systems. Or it could help someone to hide from big brother. Since it is tough to quantify human perception mathematically, we want to test our algorithm by showing you several samples and evaluating whether you could correctly identify the fakes. By answering the survey, you can help us determine how strongly we can perturb the images until humans notice something going on.

Instructions

- We will show you two images for a duration of two seconds. Example:

- The image on the **left**-hand side is a reference image. Its only purpose is to give you an idea of what the person looks like (in good quality).
- The image on the **right**-hand side is the image we want to test you on. You have to determine whether you think it is altered or not.
  You do not have to compare it directly to the reference image. Please just judge whether you think we tampered with the image or you think it looks natural.
- You will have to answer on the following scale:

- Please only use the "do not know" button if you really have to.
- We do not change the shape of the face or the age of the person.
- We do not change the resolution of the image.
- Since the images were sampled from many different places, they might be very blurry or depict people in funny poses, wearing makeup, etc.
- Images might also contain watermarks or text from magazine covers.
- Blurriness, funny poses, makeup, watermarks, magazine cover letters, etc. does **not** count as altered!
- To give you a better idea, the following images are **unaltered**:

- After answering a question, the two second timer of the next question automatically starts.
- You can always see your overall progress on the top of the screen and you can stop at any time, but we would highly appreciate it if you could make it to the end.
- After answering all the questions, we will show you an overview on how often you got it right.

About You

Please provide your age and sex. After the evaluation, we will raffle an 50-Euro Amazon Voucher among everybody that participated whilst providing an E-Mail address. The more questions you answer, the higher are the chances to win.

Age *	Sex *	E-Mail (optional)

Questionnaire Duration

You can select the amount of time you want to spend on answering the questionnaire.

Duration *: 6 minutes (100 questions)

© 2022 University of Zurich, AIML

The images on this website are for experimental purposes only and may not be saved or be used for further processing.

Experiment conducted by
Prof. Dr. Manuel Günther
Noah Chavannes

Figure A.1: QUESTIONNAIRE SCREENSHOT: LANDING PAGE. Screenshot of the custom questionnaire tool's landing page.

A.3 Adversarial Samples on FMR=0.001, 4 Targets

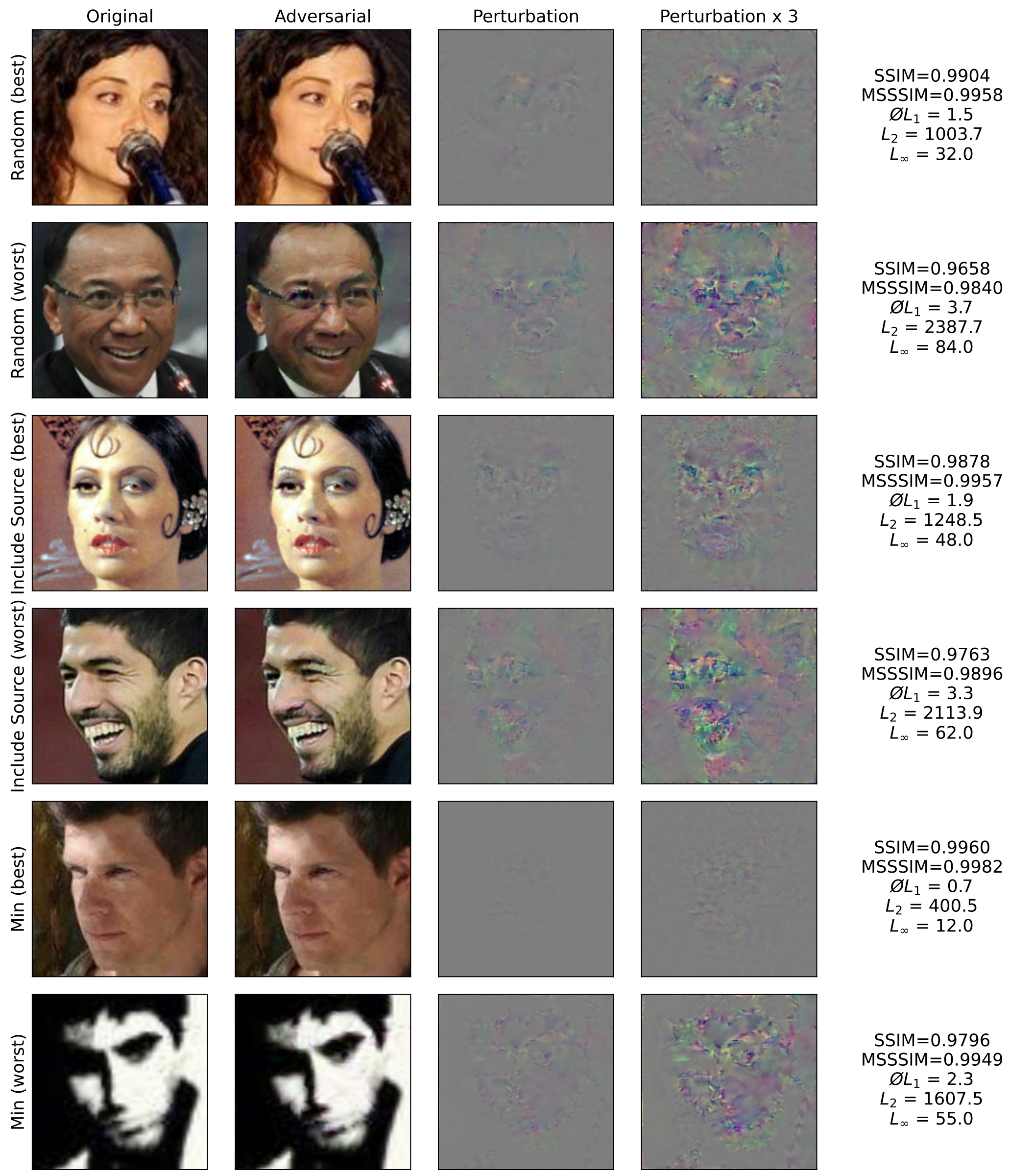


Figure A.2: PART 1: BEST AND WORST EXPERIMENT TYPE RESULTS. Adversarial images from the best and the worst successful experiment of each experiment type. All images are created with the default hyperparameters and attack 4 targets simultaneously on a network using a FMR of 0.001.

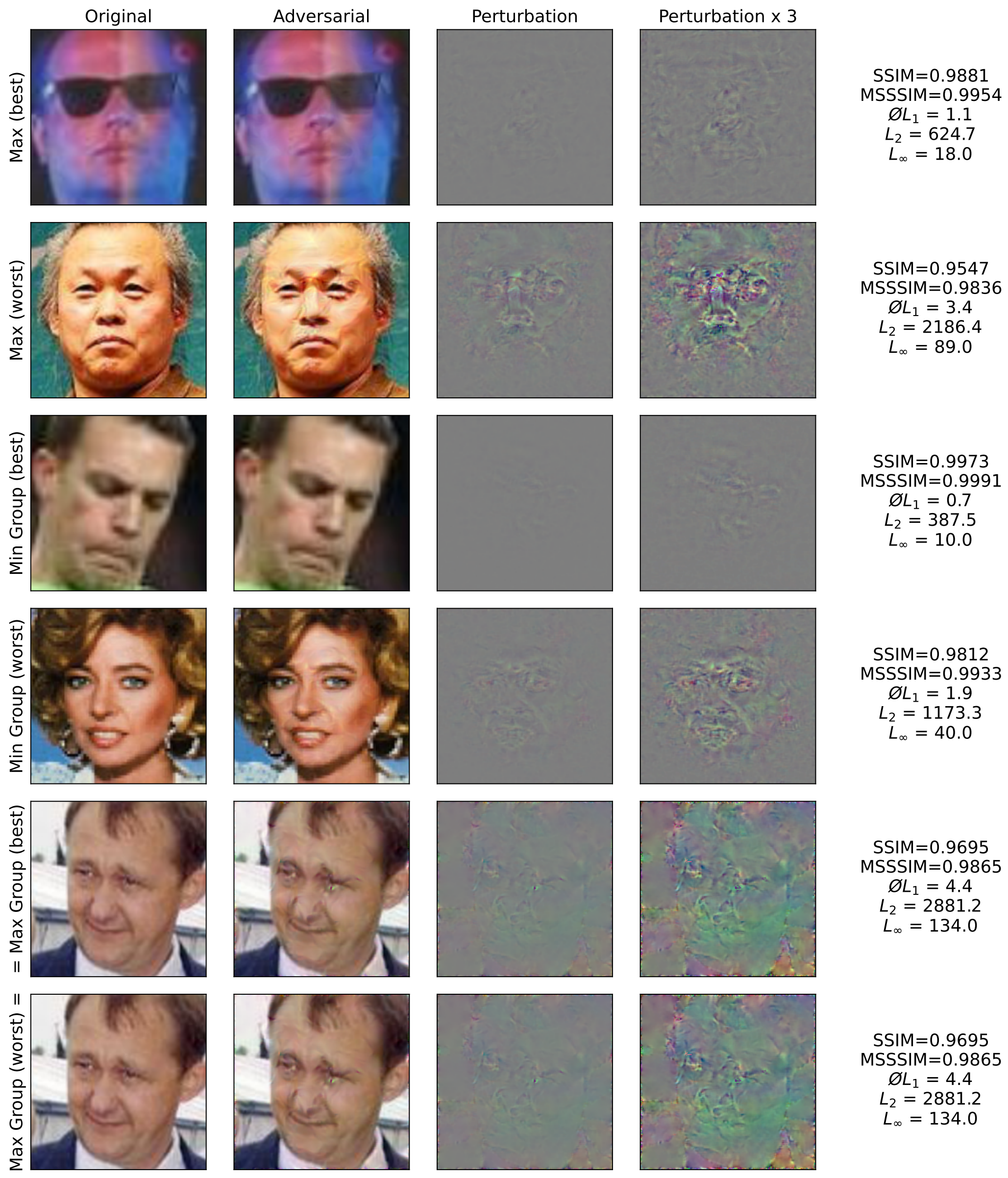


Figure A.3: PART 2: BEST AND WORST EXPERIMENT TYPE RESULTS. Adversarial images from the best and the worst successful experiment of each experiment type. All images are created with the default hyperparameters and attack 4 targets simultaneously on a network using a FMR of 0.001. The best and the worst result of the $max\ group$ experiment are identical since we only successfully produced one adversarial image in this category.

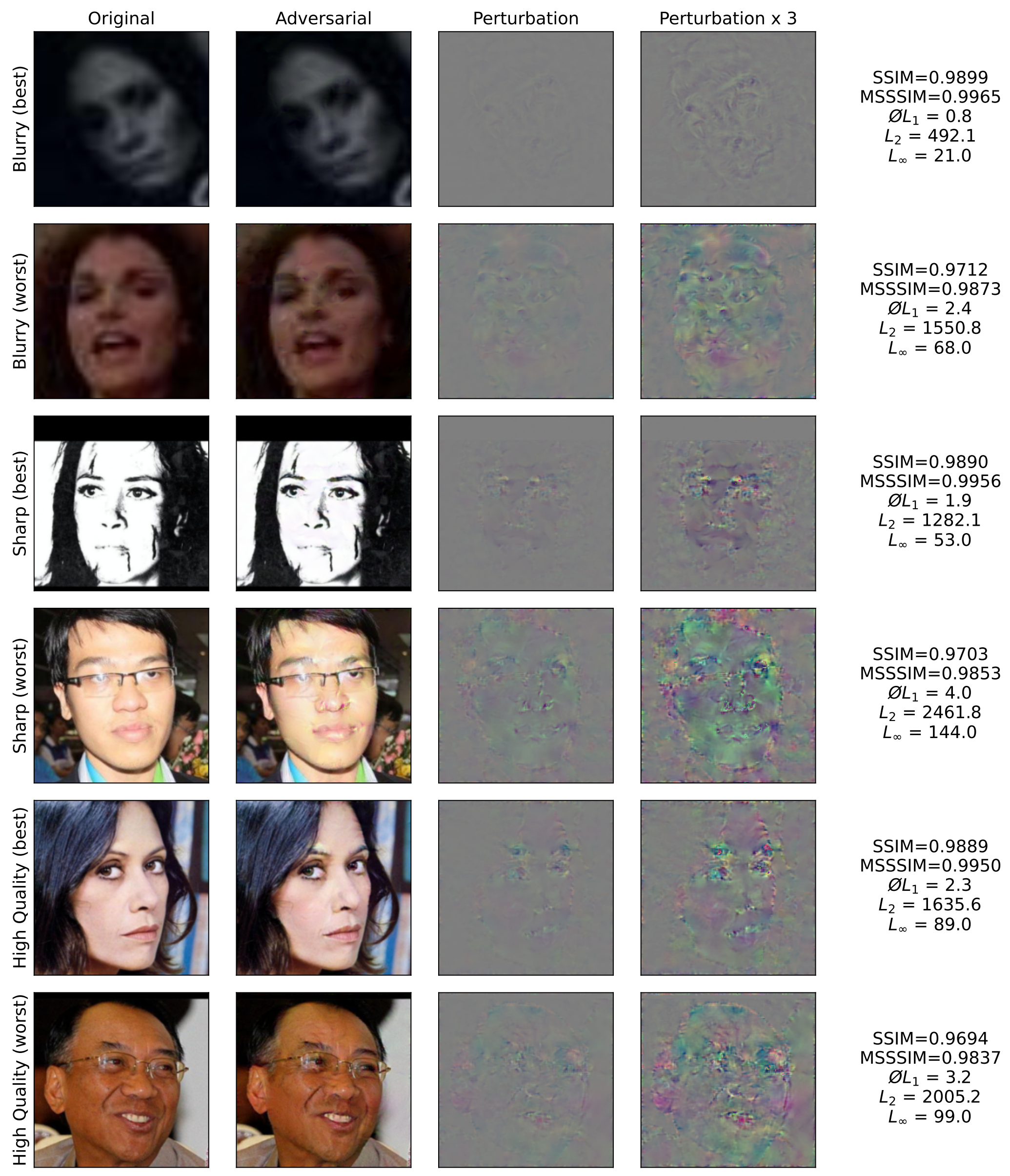


Figure A.4: PART 3: BEST AND WORST EXPERIMENT TYPE RESULTS. Adversarial images from the best and the worst successful experiment of each experiment type. All images are created with the default hyperparameters and attack 4 targets simultaneously on a network using a FMR of 0.001.

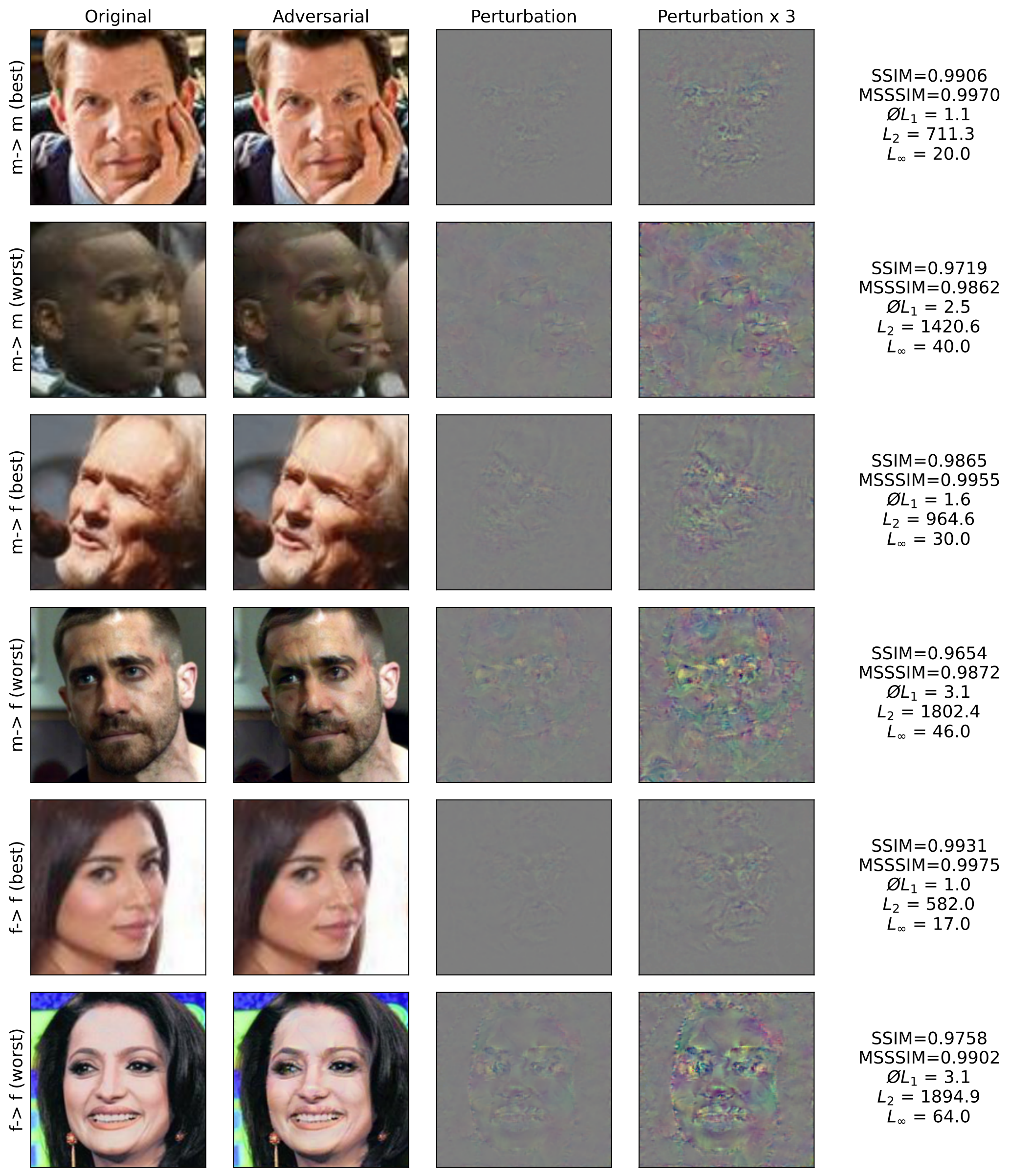


Figure A.5: Part 4: Best and Worst Experiment Type Results. Adversarial images from the best and the worst successful experiment of each experiment type. All images are created with the default hyperparameters and attack 4 targets simultaneously on a network using a FMR of 0.001.

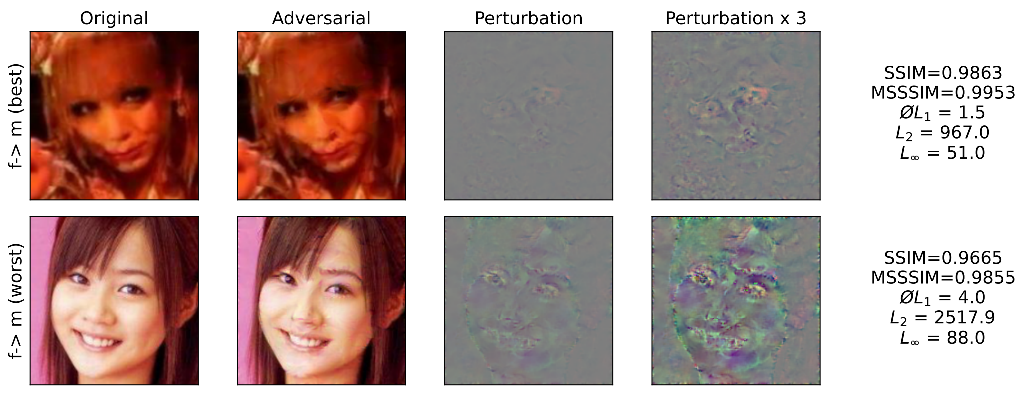


Figure A.6: PART 5: BEST AND WORST EXPERIMENT TYPE RESULTS. Adversarial images from the best and the worst successful experiment of each experiment type. All images are created with the default hyperparameters and attack 4 targets simultaneously on a network using a FMR of 0.001.

A.4 Logit Regression Models

This section lists the different logit regression models used along with their statistics. We have used and explained the $Pseudo\text{-}R^2(\rho^2)$ value to compare the models in the thesis. All attributes listed in the model statistics are briefly explained in Table A.2.

Attribute	**Description**
Dep. Variable	The dependent variable to be expressed with the independent variables.
Model	Which model is used. In our case a Generalized Linear Model (GLM).
Model Family	Which model family is used. We use a Binomial model since we have binomial distribution.
Link Function	Which link function is used. In our case the logit function since we have a logit regression model.
Method	Which method is used for fitting the data. In our case iteratively reweighted least squares (IRLS) as explained in the thesis.
Date	Date when the data was fitted to the model.
Time	Time when the data was fitted to the model.
No. Iterations	How many iterations were needed to fit the model using IRLS.
Covariance Type	Specifies how the covariance is calculated. For the IRLS method, nonrobust is the default value. It is used when homoscedasticity within the data is expected.
No. Observations	How many data points are available to fit the model to.
Df Residuals	Degree of freedom of the residuals. Calculated by $No.\ Observations - No.\ of Predicting\ Variables - 1$.
Df Model	Degree of freedom of the model. Number of predicting (independent) variables.
Scale	Parameter used in the log-likelihood calculation. For binomial distributions, this parameter is always 1.
Log-Likelihood	The log-likelihood is a numerical indicator of the probability that the model produced the given data, where a higher value indicates a better fit.
Deviance	Deviance is another quality-of-fit statistic. A lower value indicates a better fit.
Pearson chi2	A Pearson Chi-Square test is another quality-of-fit statistic.
Pseudo R-squ.	$Pseudo\text{-}R^2(\rho^2)$: Measures how well the model fits the data as explained in the thesis.
coef	Coefficients of the independent variables. Measures how strongly a change in the independent variable impacts the prediction of the dependent variable.
std err	Standard error of the coefficients.
z	In a logit regression, the z-value is calculated by dividing the coefficient by its standard error. It is used for calculating the p-values.
P $>$ **|z|**	P-value. Probability of the the null hypothesis that the coefficient has no effect on the model. The lower the value the more significant is the coefficient for the model.
[0.025	Lower bound of the 95%-confidence interval of the coefficients.
0.975]	Upper bound of the 95%-confidence interval of the coefficients.

Table A.2: MEANING OF MODEL STATISTICS. Explanation of the attributes found in the model statistics of the logit regression models.

A.4.1 SSIM Only Model

Model R-Style Formula

$rated\_as\_adversarial \sim ssim$

Model Statistics

Dep. Variable:	rated_as_adversarial	**No. Observations:**	4881
Model:	GLM	**Df Residuals:**	4879
Model Family:	Binomial	**Df Model:**	1
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-3144.8
Date:	Fri, 15 Apr 2022	**Deviance:**	6289.5
Time:	11:19:53	**Pearson chi2:**	4.89e+03
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.09226
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Intercept	64.9632	3.152	20.609	0.000	58.785	71.141
ssim	-65.5599	3.177	-20.638	0.000	-71.786	-59.334

Table A.3: STATISTICS SSIM MODEL. Generalized Linear Model Regression Results

A.4.2 MSSSIM Only Model

Model R-Style Formula

$rated\_as\_adversarial \sim msssim$

Model Statistics

Dep. Variable:	rated_as_adversarial	**No. Observations:**	4881
Model:	GLM	**Df Residuals:**	4879
Model Family:	Binomial	**Df Model:**	1
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-3145.7
Date:	Fri, 15 Apr 2022	**Deviance:**	6291.3
Time:	11:19:53	**Pearson chi2:**	4.90e+03
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.09192
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Intercept	150.2641	7.304	20.572	0.000	135.948	164.580
msssim	-150.8459	7.328	-20.585	0.000	-165.209	-136.483

Table A.4: STATISTICS MSSSIM MODEL. Generalized Linear Model Regression Results

A.4.3 SSIM + MSSSIM Model

Model R-Style Formula

$rated\_as\_adversarial \sim ssim + msssim$

Model Statistics

Dep. Variable:	rated_as_adversarial	**No. Observations:**	4881
Model:	GLM	**Df Residuals:**	4878
Model Family:	Binomial	**Df Model:**	2
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-3142.7
Date:	Fri, 15 Apr 2022	**Deviance:**	6285.4
Time:	11:19:53	**Pearson chi2:**	4.89e+03
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.09302
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Intercept	104.5972	19.992	5.232	0.000	65.414	143.780
ssim	-36.1645	14.883	-2.430	0.015	-65.334	-6.995
msssim	-69.0280	34.284	-2.013	0.044	-136.224	-1.833

Table A.5: STATISTICS SSIM + MSSSIM MODEL. Generalized Linear Model Regression Results

A.4.4 SSIM + Other Variables Model

Model R-Style Formula

$rated\_as\_adversarial \sim ssim + targets + type + fmr + temporal\_adv + temporal\_orig + user\_sex + user\_age$

Model Statistics

Dep. Variable:	rated_as_adversarial	**No. Observations:**	4881
Model:	GLM	**Df Residuals:**	4859
Model Family:	Binomial	**Df Model:**	21
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-3116.9
Date:	Fri, 15 Apr 2022	**Deviance:**	6233.8
Time:	11:19:54	**Pearson chi2:**	4.89e+03
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.1026
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Intercept	51.7921	4.890	10.591	0.000	42.207	61.377
blurry	0.2663	0.236	1.128	0.259	-0.196	0.729
f→f	0.4593	0.204	2.250	0.024	0.059	0.859
f→m	0.1368	0.202	0.678	0.498	-0.259	0.533
high quality	-0.0518	0.218	-0.238	0.812	-0.479	0.375
include source	0.0098	0.099	0.099	0.921	-0.185	0.204
max	0.3039	0.124	2.449	0.014	0.061	0.547
max group	0.3617	0.165	2.195	0.028	0.039	0.685
m→f	0.2378	0.218	1.093	0.274	-0.189	0.664
min	0.1870	0.142	1.317	0.188	-0.091	0.465
min group	0.1122	0.137	0.818	0.413	-0.157	0.381
min group include	0.0383	0.211	0.182	0.856	-0.374	0.451
m→m	-0.2943	0.221	-1.330	0.184	-0.728	0.139
sharp	-0.4537	0.164	-2.758	0.006	-0.776	-0.131
FAR0.00001	-0.0604	0.136	-0.444	0.657	-0.327	0.206
FAR0.0001	0.2410	0.113	2.140	0.032	0.020	0.462
ssim	-52.4572	4.928	-10.645	0.000	-62.115	-42.799
targets	0.0067	0.013	0.520	0.603	-0.019	0.032
temporal_adv	0.0034	0.002	1.778	0.075	-0.000	0.007
temporal_orig	-0.0063	0.002	-3.211	0.001	-0.010	-0.002
user_sex	0.0469	0.065	0.722	0.470	-0.080	0.174
user_age	0.0021	0.004	0.560	0.575	-0.005	0.009

Table A.6: STATISTICS SSIM + OTHER MODEL. Generalized Linear Model Regression Results

A.4.5 MSSSIM + Other Variables Model

Model R-Style Formula

$rated\_as\_adversarial \sim msssim + targets + type + fmr + temporal\_adv + temporal\_orig + user\_sex + user\_age$

Model Statistics

Dep. Variable:	rated_as_adversarial	**No. Observations:**	4881
Model:	GLM	**Df Residuals:**	4859
Model Family:	Binomial	**Df Model:**	21
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-3112.1
Date:	Fri, 15 Apr 2022	**Deviance:**	6224.1
Time:	11:19:54	**Pearson chi2:**	4.90e+03
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.1043
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Intercept	121.1009	10.965	11.044	0.000	99.609	142.592
blurry	0.1784	0.240	0.743	0.457	-0.292	0.649
f→f	0.5202	0.204	2.552	0.011	0.121	0.920
f→m	0.1383	0.203	0.681	0.496	-0.260	0.537
high quality	-0.0432	0.218	-0.198	0.843	-0.471	0.385
include source	0.0008	0.099	0.008	0.994	-0.194	0.196
max	0.3550	0.124	2.866	0.004	0.112	0.598
max group	0.4535	0.162	2.794	0.005	0.135	0.772
m→f	0.2423	0.218	1.109	0.267	-0.186	0.671
min	0.2043	0.142	1.435	0.151	-0.075	0.483
min group	0.1728	0.138	1.249	0.212	-0.098	0.444
min group include	0.1411	0.212	0.665	0.506	-0.275	0.557
m→m	-0.2637	0.222	-1.187	0.235	-0.699	0.172
sharp	-0.4780	0.166	-2.886	0.004	-0.803	-0.153
FAR0.00001	-0.0527	0.136	-0.386	0.699	-0.320	0.215
FAR0.0001	0.2689	0.113	2.381	0.017	0.048	0.490
msssim	-121.7583	11.001	-11.067	0.000	-143.321	-100.196
targets	0.0012	0.013	0.095	0.924	-0.024	0.027
temporal_adv	0.0037	0.002	1.933	0.053	-5.1e-05	0.007
temporal_orig	-0.0064	0.002	-3.343	0.001	-0.010	-0.003
user_sex	0.0519	0.065	0.799	0.424	-0.075	0.179
user_age	0.0019	0.004	0.518	0.604	-0.005	0.009

Table A.7: STATISTICS MSSSIM + OTHER MODEL. Generalized Linear Model Regression Results

A.4.6 SSIM + MSSSIM + Other Variables Model

Model R-Style Formula

$rated\_as\_adversarial \sim ssim + msssim + targets + type + fmr + temporal\_adv + temporal\_orig + user\_sex + user\_age$

Model Statistics

Dep. Variable:	rated_as_adversarial	**No. Observations:**	4881
Model:	GLM	**Df Residuals:**	4858
Model Family:	Binomial	**Df Model:**	22
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-3112.0
Date:	Fri, 15 Apr 2022	**Deviance:**	6224.0
Time:	11:19:54	**Pearson chi2:**	4.90e+03
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.1044
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Intercept	115.9634	21.267	5.453	0.000	74.280	157.647
blurry	0.1855	0.241	0.769	0.442	-0.287	0.658
f→f	0.5167	0.204	2.530	0.011	0.116	0.917
f→m	0.1380	0.203	0.679	0.497	-0.260	0.536
high quality	-0.0437	0.218	-0.200	0.841	-0.472	0.384
include source	0.0023	0.100	0.023	0.982	-0.193	0.197
max	0.3502	0.125	2.800	0.005	0.105	0.595
max group	0.4443	0.166	2.682	0.007	0.120	0.769
m→f	0.2418	0.218	1.107	0.268	-0.186	0.670
min	0.2052	0.142	1.441	0.150	-0.074	0.484
min group	0.1710	0.138	1.235	0.217	-0.100	0.442
min group include	0.1365	0.213	0.641	0.521	-0.281	0.554
m→m	-0.2665	0.222	-1.199	0.231	-0.702	0.169
sharp	-0.4760	0.166	-2.873	0.004	-0.801	-0.151
FAR0.00001	-0.0522	0.136	-0.383	0.702	-0.320	0.215
FAR0.0001	0.2673	0.113	2.363	0.018	0.046	0.489
ssim	-4.5278	16.080	-0.282	0.778	-36.044	26.988
msssim	-112.0966	36.012	-3.113	0.002	-182.679	-41.514
targets	0.0015	0.013	0.112	0.911	-0.024	0.027
temporal_adv	0.0036	0.002	1.861	0.063	-0.000	0.007
temporal_orig	-0.0063	0.002	-3.260	0.001	-0.010	-0.003
user_sex	0.0515	0.065	0.792	0.428	-0.076	0.179
user_age	0.0019	0.004	0.522	0.601	-0.005	0.009

Table A.8: STATISTICS SSIM + MSSSIM + OTHER MODEL. Generalized Linear Model Regression Results

A.4.7 SSIM + Other Variables Model, Full Dataset

Model R-Style Formula

$rated\_as\_adversarial \sim ssim + targets + type + fmr + temporal\_adv + temporal\_orig + user\_sex + user\_age$

Model Statistics

Dep. Variable:	rated_as_adversarial	**No. Observations:**	7510
Model:	GLM	**Df Residuals:**	7488
Model Family:	Binomial	**Df Model:**	21
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-4761.0
Date:	Fri, 15 Apr 2022	**Deviance:**	9522.0
Time:	11:20:02	**Pearson chi2:**	7.53e+03
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.1106
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Intercept	59.4590	3.966	14.993	0.000	51.686	67.232
blurry	0.3557	0.191	1.860	0.063	-0.019	0.731
f→f	0.4448	0.168	2.652	0.008	0.116	0.774
f→m	0.3069	0.168	1.826	0.068	-0.022	0.636
high quality	0.1500	0.177	0.849	0.396	-0.196	0.496
include source	-0.0588	0.081	-0.723	0.470	-0.218	0.101
max	0.2147	0.100	2.142	0.032	0.018	0.411
max group	0.2328	0.131	1.776	0.076	-0.024	0.490
m→f	0.2316	0.177	1.307	0.191	-0.116	0.579
min	0.1243	0.115	1.078	0.281	-0.102	0.350
min group	0.1533	0.110	1.388	0.165	-0.063	0.370
min group include	0.2059	0.169	1.221	0.222	-0.125	0.536
m→m	-0.0448	0.175	-0.256	0.798	-0.388	0.298
sharp	-0.2849	0.134	-2.129	0.033	-0.547	-0.023
FAR0.00001	-0.0972	0.111	-0.880	0.379	-0.314	0.119
FAR0.0001	0.2067	0.093	2.224	0.026	0.025	0.389
ssim	-60.2153	3.997	-15.066	0.000	-68.049	-52.382
targets	-0.0029	0.010	-0.281	0.779	-0.023	0.018
temporal_adv	0.0038	0.002	2.411	0.016	0.001	0.007
temporal_orig	-0.0031	0.002	-1.960	0.050	-0.006	-1.73e-07
user_sex	0.0540	0.052	1.031	0.302	-0.049	0.157
user_age	0.0021	0.003	0.674	0.500	-0.004	0.008

Table A.9: STATISTICS SSIM + OTHER MODEL, FULL DATASET (FINAL MODEL). Generalized Linear Model Regression Results

A.4.8 Reduced FMR Model, Reduced Dataset

Model R-Style Formula

$rated\_as\_adversarial \sim ssim + fmr$

Model Statistics

Dep. Variable:	rated_as_adversarial	**No. Observations:**	459
Model:	GLM	**Df Residuals:**	455
Model Family:	Binomial	**Df Model:**	3
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-275.31
Date:	Fri, 15 Apr 2022	**Deviance:**	550.61
Time:	11:20:12	**Pearson chi2:**	464.
No. Iterations:	4	**Pseudo R-squ. (CS):**	0.1490
Covariance Type:	nonrobust		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
Intercept	108.7397	13.538	8.032	0.000	82.206	135.274
FAR0.00001	0.1121	0.246	0.455	0.649	-0.370	0.595
FAR0.0001	0.0850	0.261	0.325	0.745	-0.427	0.597
ssim	-109.8947	13.642	-8.055	0.000	-136.633	-83.156

Table A.10: STATISTICS REDUCED FMR MODEL. Generalized Linear Model Regression Results

A.5 Re-Run LOTS with SSIM Only

			Run 1			Run 2			SSIM
FMR	**Targets**	**Experiment Type**	**∅SSIM**	**∅Iter.**	**SR**	**∅SSIM**	**∅Iter.**	**SR**	**Diff.**
0.001	4 Targets	*Random*	0.9792	1723.8	88%	0.9826	2222.5	83%	0.0034
		Include Source	0.9791	3577.6	43%	0.9826	3857.8	35%	0.0035
		Min	0.9900	197.5	99%	0.9915	244.14	99%	0.0015
		Max	0.9758	89.75	100%	0.9793	179.66	100%	0.0035
		Min Group	0.9935	11.14	100%	0.9942	15.67	100%	0.0007
		Max Group	0.9695	4978.3	1%	0.9742	4999	0%	0.0047
		Blurry	0.9819	1247.5	93%	0.9855	1239.6	93%	0.0036
		Sharp	0.9800	2479.4	73%	0.9835	2757.5	68%	0.0035
		High Quality	0.9788	2205	76%	0.9823	2575.7	71%	0.0035
		$m \to m$	0.9820	991.62	94%	0.9849	1211.3	90%	0.0029
		$m \to f$	0.9785	529.89	97%	0.9819	751.52	98%	0.0034
		$f \to f$	0.9854	409.92	99%	0.9877	506.98	99%	0.0023
		$f \to m$	0.9786	1089.5	95%	0.9821	1324.6	92%	0.0035
	3 Targets	*Random*	0.9827	395.59	99%	0.9853	503.54	98%	0.0026
		Include Source	0.9824	1837.7	80%	0.9853	2034	79%	0.0029
		Max Group	0.9711	3842.2	40%	0.9761	4198.3	29%	0.0050
		Sharp	0.9834	548.17	99%	0.9859	817.77	97%	0.0025
	2 Targets	*Random*	0.9867	27.23	100%	0.9883	38.21	100%	0.0016
		Include Source	0.9886	241.01	100%	0.9904	390.21	99%	0.0018
		Max Group	0.9757	264.88	100%	0.9794	387.56	98%	0.0037
	1 Target	*Random*	0.9940	7.83	100%	0.9946	11.03	100%	0.0006
		Include Source	0.9941	12.24	100%	0.9949	17.16	100%	0.0008
	5 Targets	*Random*	0.9766	3319.7	51%	0.9806	3535.9	47%	0.0040
		Min	0.9881	715.3	95%	0.9899	910.29	93%	0.0018
		Max	0.9763	196.96	99%	0.9797	297.8	99%	0.0034
		Min Group	0.9921	18.17	100%	0.9931	23.78	100%	0.0010
	6 Targets	*Random*	0.9757	4230.1	26%	0.9798	4363.3	22%	0.0041
		Min	0.9868	1357.8	86%	0.9889	1744.7	82%	0.0021
		Max	0.9774	594.24	95%	0.9810	809.8	91%	0.0036
		Min Group	0.9917	47.87	100%	0.9927	56.87	100%	0.0010
	7 Targets	*Min Group*	0.9902	58.5	100%	0.9915	68.37	100%	0.0013
		Max	0.9781	932.9	89%	0.9816	1235.6	87%	0.0035
	10 Targets	*Min Group*	0.9880	250.04	100%	0.9900	354.47	100%	0.0020
		Min Group + Include	0.9884	307.05	100%	0.9903	412.9	100%	0.0019
	15 Targets	*Min Group*	0.9844	986.95	94%	0.9870	1215.8	94%	0.0026
		Min Group + Include	0.9848	1124.2	92%	0.9873	1357	91%	0.0025
	20 Targets	*Min Group*	0.9828	2188.3	80%	0.9858	2436.6	74%	0.0030
		Min Group + Include	0.9830	2373.7	77%	0.9860	2547.4	74%	0.0030
0.0001	4 Targets	*Random*	0.9782	4885.5	5%	0.9820	4846.1	5%	0.0038
	3 Targets	*Random*	0.9808	2771.4	59%	0.9840	2924.5	56%	0.0032
	2 Targets	*Random*	0.9845	189.89	99%	0.9868	264.19	98%	0.0023
		Include Source	0.9855	2653	62%	0.9878	2906.6	62%	0.0023
	1 Target	*Random*	0.9918	9.83	100%	0.9926	13.97	100%	0.0008
		Include Source	0.9924	100.45	99%	0.9935	131.59	99%	0.0011
0.00001	2 Targets	*Random*	0.9845	2677.8	57%	0.9870	2739.8	57%	0.0025
		Include Source	0.9847	4940.8	2%	0.9873	4927.7	2%	0.0026
	1 Target	*Random*	0.9891	13.44	100%	0.9902	18.68	100%	0.0011
		Include Source	0.9904	2149.9	64%	0.9920	2283.9	62%	0.0016
		AVERAGE	**0.9837**	**1370.9**	**81.40%**	**0.9863**	**1514.82**	**79.65%**	**0.0026**

Table A.11: COMPARISON ORIGINAL EXPERIMENTS VERSUS SSIM ONLY RE-RUN. Comparison of the average SSIM scores, iterations and success rates of two different runs. **Run 1**: Original experiment with the hyperparameters: $width_{step} = 3, w_{cos} = 1.0, w_{ssim} = 0.9, w_{msssim} = 1.0$, **Run 2**: Experiment with SSIM score only and the following hyperparameters: $width_{step} = 2, w_{cos} = 1.0, w_{ssim} = 1.5, w_{msssim} = 0.0$.

List of Figures

List of Tables

Bibliography

Akhtar, N. and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430.

Andersson, P., Nilsson, J., Akenine-Möller, T., Oskarsson, M., Åström, K., and Fairchild, M. D. (2020). FLIP: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech.*, 3(2):15–1.

Ballantyne, M., Boyer, R. S., and Hines, L. (1996). Woody Bledsoe: His life and legacy. *AI magazine*, 17(1):7–7.

Bledsoe, W. (1964). *The Model Method in Facial Recognition, Technical Report PRI 15, Panoramic Research, Inc., Palo Alto*. California.

Bledsoe, W. (1966). Man-Machine facial recognition: Report on a Large-Scale experiment. Technical Report PRI-22.

Bontrager, P., Roy, A., Togelius, J., Memon, N., and Ross, A. (2018). DeepMasterPrints: Generating masterprints for dictionary attacks via latent variable evolution. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–9.

Boutros, F., Damer, N., Kirchbuchner, F., and Kuijper, A. (2021). ElasticFace: Elastic margin loss for deep face recognition. *arXiv preprint arXiv:2109.09416*.

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., and Zisserman, A. (2018). VGGFace2: A dataset for recognising faces across pose and age. In *2018 13th IEEE International Conference on Automatic Face Gesture Recognition (FG 2018)*, pages 67–74.

Cao, Z., Yin, Q., Tang, X., and Sun, J. (2010). Face recognition with learning-based descriptor. In *2010 IEEE Computer society conference on computer vision and pattern recognition*, pages 2707–2714. IEEE.

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57.

Dabouei, A., Soleymani, S., Dawson, J., and Nasrabadi, N. (2019). Fast geometrically-perturbed adversarial faces. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1979–1988. IEEE.

Das, S. and Suganthan, P. N. (2010). Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). ArcFace: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699.

Goldstein, A. J., Harmon, L. D., and Lesk, A. B. (1971). Identification of human faces. *Proceedings of the IEEE*, 59(5):748–760.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Goswami, G., Ratha, N., Agarwal, A., Singh, R., and Vatsa, M. (2018). Unravelling robustness of deep learning based face recognition against adversarial attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Guo, Y., Zhang, L., Hu, Y., He, X., and Gao, J. (2016). MS-Celeb-1M: A dataset and benchmark for large-scale face recognition.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141.

Huang, G. B., Mattar, M., Berg, T., and Learned-Miller, E. (2008). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*.

Kemelmacher-Shlizerman, I., Seitz, S. M., Miller, D., and Brossard, E. (2016). The MegaFace benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4873–4882.

Kurakin, A., Goodfellow, I. J., and Bengio, S. (2018). Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC.

Kwon, H., Kwon, O., Yoon, H., and Park, K.-W. (2019). Face friend-safe adversarial example on face recognition system. In *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 547–551. IEEE.

Li, H. and Zhu, X. (2016). Face recognition technology research and implementation based on mobile phone system. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 972–976.

Liu, C. and Wechsler, H. (2002). Gabor feature based classification using the enhanced Fisher linear discriminant model for face recognition. *IEEE Transactions on Image processing*, 11(4):467–476.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., and Song, L. (2017a). SphereFace: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220.

Liu, Y., Chen, X., Liu, C., and Song, D. (2017b). Delving into transferable adversarial examples and black-box attacks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738.

McFadden, D. (1977). Quantitative methods for analyzing travel behaviour of individuals: Some recent developments. *Cowles Foundation Discussion Papers*, 474.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773.

Nguyen, H. H., Yamagishi, J., Echizen, I., and Marcel, S. (2020). Generating master faces for use in performing wolf attacks on face recognition systems. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–10. IEEE.

Nilsson, J. and Akenine-Möller, T. (2020). Understanding SSIM. *arXiv preprint arXiv:2006.13846*.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE.

Papernot, N., McDaniel, P. D., Wu, X., Jha, S., and Swami, A. (2015). Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508.

Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition.

Pech-Pacheco, J. L., Cristóbal, G., Chamorro-Martinez, J., and Fernández-Valdivia, J. (2000). Diatom autofocusing in brightfield microscopy: a comparative study. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 3, pages 314–317. IEEE.

Rakitianskaia, A. and Engelbrecht, A. (2015). Measuring saturation in neural networks. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1423–1430.

Rozsa, A., Günther, M., and Boult, T. E. (2017). LOTS about attacking deep features. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 168–176. IEEE.

Rozsa, A., Rudd, E. M., and Boult, T. E. (2016). Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–32.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. (2016). Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*, pages 1528–1540.

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. (2019). A general framework for adversarial examples with objectives. *ACM Transactions on Privacy and Security (TOPS)*, 22(3):1–30.

Shen, M., Liao, Z., Zhu, L., Xu, K., and Du, X. (2019). VLA: A practical visible light-based attack on face recognition systems in physical world. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(3):1–19.

Sirovich, L. and Kirby, M. (1987). Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524.

Su, J., Vargas, D. V., and Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014). Intriguing properties of neural networks. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708.

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2015). Web-scale training for face identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2746–2754.

Tao, G., Ma, S., Liu, Y., and Zhang, X. (2018). Attacks meet interpretability: Attribute-steered detection of adversarial samples. *Advances in Neural Information Processing Systems*, 31.

Terhörst, P., Fährmann, D., Kolf, J. N., Damer, N., Kirchbuchner, F., and Kuijper, A. (2021). MAAD-Face: A massively annotated attribute dataset for face images. *IEEE Transactions on Information Forensics and Security*, 16:3942–3957.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. (2017). Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.

Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86.

Vakhshiteh, F., Nickabadi, A., and Ramachandra, R. (2021). Adversarial attacks against face recognition: A comprehensive study. *IEEE Access*, 9:92735–92756.

Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., and Liu, W. (2018). CosFace: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Wang, Z., Simoncelli, E. P., and Bovik, A. C. (2003). Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee.

Wiskott, L., Krüger, N., Kuiger, N., and Von Der Malsburg, C. (1997). Face recognition by elastic bunch graph matching. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):775–779.

Wolf, L., Hassner, T., and Maoz, I. (2011). Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*, pages 529–534. IEEE.

Yan, M., Zhao, M., Xu, Z., Zhang, Q., Wang, G., and Su, Z. (2019). VarGFaceNet: An efficient variable group convolutional neural network for lightweight face recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0.

Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multi-task cascaded convolutional networks. *IEEE signal processing letters*, 23(10):1499–1503.

Zhang, L., Bo, C., Hou, J., Li, X.-Y., Wang, Y., Liu, K., and Liu, Y. (2015). Kaleido: You can watch it but cannot record it. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 372–385.

Zhang, Q., Li, J., Yao, M., Song, L., Zhou, H., Li, Z., Meng, W., Zhang, X., and Wang, G. (2019). VarGNet: Variable group convolutional neural network for efficient embedded computing. *arXiv preprint arXiv:1907.05653*.

Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458.

Zhong, Y. and Deng, W. (2020). Towards transferable adversarial attack against deep face recognition. *IEEE Transactions on Information Forensics and Security*, 16:1452–1466.

Zhu, Z.-A., Lu, Y.-Z., and Chiang, C.-K. (2019). Generating adversarial examples by makeup attacks on face recognition. In *2019 IEEE International Conference on Image Processing (ICIP)*.