
University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

# Adversarial Machine Learning and its Impact on System Automation

**Rech Leong Tian Poh**

2nd April 2020

# Abstract

To date, Neural Networks have been employed in several applications such as Autonomous Vehicles, Facial Recognition, Object Detection and Anomaly Detection. The prevalence of Machine Learning has made it possible for Neural Networks to see and recognise faces and objects in the same way that humans do, thus giving birth to Deep Convolutional Networks (DCNs). DCNs boast supreme inference accuracy and speed due to their strong links between features and outputs. DCNs are very often used for recognition of visual input. However, DCNs are known to be vulnerable to **Adversarial Examples –** perturbed inputs that are designed to produce intentional errors in the network. This is a major threat to automated systems which employ the use of neural networks. Should these attacks succeed, they can jeopardise personal data and safety of the individuals recorded in the NN. This concern is further escalated when NNs are deployed in safety-critical systems such as Autonomous Vehicles and Facial Recognition. This report introduces a **structured approach for generating adversarial examples**, as well as **retraining data** to produce an **altered neural network**, one that will misbehave given the right conditions. This report proposes an *Adversarial Patch* attack on *VGG_Face*, a state-of-the-art Facial Recognition DCN. The attack exploits the *Texture-Bias* and *Strong Interconnecting Layers* of a facial recognition model. After which, experiment findings indicated that the adversarial patch could be applied to images that were not part of the NN's original training data. Subsequently, the adversarial patch had also proven to be robust towards various physical environmental factors, as well as transparency. The resultant altered model was also able to **retain the original functionalities of the benign model while only producing the intended adverse behaviour in the presence of the adversarial patch**. The nature of the attack is one of a **Targeted Black-Box Attack**. This means that the adversary is able to make a **targeted classification without any access to the target model's training data or implementation details**. After the attack has been carried out, this report evaluates the effectiveness of the attack based on its **ease of use, robustness and distribution**. Finally, this report discusses the respective use cases of the adversarial attack.

**Notes:**
A summarised version of this report is in the works for research conference submission.
**Project Industrial Video**: https://youtu.be/dK2CCFNmf4Y
**Project Presentation Video**: https://youtu.be/8r8iUDsnR-U
**Project Demonstration Video**: https://youtu.be/194VDD9MCgY
**Github Repository**: https://github.com/2427218L/FYP_AML

# Education Use Consent

# Contents

# 1 Introduction

Neural Networks (NNs) are widely used in applications such as autonomous vehicles, facial recognition, speech recognition, natural language communication and gaming systems. Neural Networks are typically trained using huge amounts of data. The process which 'installs' training data into the neural network is called the learning process. The learning process of a NN is on a scale that is far beyond human capabilities. To put this into perspective, it is the equivalent of a human absorbing multiple lifetimes worth of knowledge and experience. As a result, neural networks far supersede human expertise in several areas. However, as we become ever more dependent on machine learning systems for automation of services such as autonomous vehicles and facial recognition, one should also consider the fact that these networks are vulnerable and can be compromised by malicious adversaries. Despite the prevalence of neural networks, their weaknesses have yet to be fully comprehended. The pressing concern among data scientists and engineers today is how adversarial attacks are progressively becoming easier to execute and increasing in number, while the same could not be said about defensive measures.

## 1.1 Problem Statement

Many forms of attack methods on machine learning systems leverage on the various inherent vulnerabilities of neural networks. Such attacks exploit the vulnerabilities of neural networks in the training and deployment phases such as **data poisoning attacks and image obfuscation attacks**. The prevalence of adversarial attacks has become a major problem that should not be overlooked. Whilst neural networks are constantly becoming an integral part of automated systems, it becomes a pressing need to **identify what exactly makes these attacks work**, the kinds of **vulnerabilities that were exploited**, and the relevant **impact(s) as a result of these attacks**. Once the vulnerabilities are identified, appropriate defensive measures should be taken. One major problem today is that while neural networks are continually increasing in complexity in order to better perform the task that they were trained for, one simply cannot say that we can be fully reliant on neural networks to perform these tasks, especially if it could lead to the leak and abuse of our private personal data. Safety should be of utmost importance and we need to be assured of that to a certain degree if we decide to rely on machine learning systems to automate tasks. Without appropriate defensive measures, attacks on machine learning systems will only become increasingly devastating. For example, in facial recognition systems, our personal facial features and other sensitive information would be leaked and potentially abused in impersonation attempts. In other cases like autonomous vehicles, a compromised automated driving system could lead to deaths of both the passengers and other road users.

### 1.1.1 Adversarial Threat

Deep Convolutional Networks (DCNs) are one of the types of neural networks that have been developed to perform tasks accurately with significantly high confidence. It boasts multiple layers of abstraction and complex learning and inference algorithms in order to achieve higher levels of performance in its task(s). However, they have been shown to be vulnerable to adversarial examples, which are inputs that are visually similar to genuine inputs, engineered to fool the

neural network. The perturbations made to a benign image tend to be difficult to notice and challenging to detect while still being able to achieve the adversary's objectives.

## 1.2 Aims

The project revolved around the research, simulation and evaluation of adversarial attacks on complex neural networks such as DCNs based on the following objectives:

- To identify the vulnerabilities of DCNs.
- How these vulnerabilities can be exploited.
- How easy it is to perform the attack.
- What are the implications of these attacks?
- What are the possible defensive measures that can be taken?

## 1.3 Dissertation Outline

The dissertation is structured into chapters as follows:

- **Chapter 2** provides some background about the types of adversarial attacks on neural networks and the various vulnerabilities that were exploited in order to perform the attack. It also discusses some existing successful attacks and some defensive measures that could be adopted.
- **Chapter 3** analyses the problem domain, thus deriving a set of requirements and attack considerations.
- **Chapter 4** discusses the design architecture and considerations during the conceptualization phase of the project.
- **Chapter 5** discusses the implementation process of the *Adversarial Patch Attack* presented, which is a derivative of adversarial patches and adversarial training. It also discusses the respective frameworks and resources used during the implementation phase of the product.
- **Chapter 6** discusses the relevant experiments done to evaluate the adversarial attack, their respective findings and the implications on system automation.
- **Chapter 7** summarises the paper and gives a brief overview of the product and research done. This also includes reflections and future work.

# 2 Background & Literature Review

In this section, we discuss the relevant works and research that were explored in the literature review phase, to gain a better insight into the security issues pertaining to neural networks. They are categorised into their various scopes and areas of application.

## 2.1 Adversarial Attacks on Machine Learning Systems and Their Vulnerabilities

Adversarial attacks mainly intend for the target models to predict incorrectly for malicious reasons. Very often, these attacks result in several adverse effects and jeopardise the safety of humans and their respective personal data, especially in Safety-Critical Systems. Adversarial attacks can be categorised into two types [1]:

- **Untargeted Adversarial Attacks** - Attacks of this nature have no regard for the final prediction of the network. The only objective here is to fool the neural network. Such attacks are often aimed at impairing the functionality of the network in order to achieve their malicious intent. Attacks of this nature could lead to *Denial Of Service Attacks*[2]. Should these attacks be performed on software systems that provide services for large audiences, it could dampen productivity and affect our livelihood.
- **Targeted Adversarial Attacks** - Attacks of this nature have a target class label in the network that the adversary wants it to predict. As such, all the adversarial inputs are aimed at influencing the model to predict incorrectly, into the targeted class label. The focus of such attacks are to take advantage of the targeted classification to manipulate the system and realise malicious intent. Attacks of this nature could result in impersonation, identity theft or in some cases even mediated murder. Whilst targeted attacks are more difficult to implement and achieve, the respective impacts are increasingly devastating.

### 2.1.1 Vulnerabilities in the Training Stage

It has been shown that neural networks are vulnerable in the training stage [3]. In practice, attackers would not have access to the training data and the implementation details of the network. However, with the right approach and sufficient probing, the adversary can identify certain low-level features and layers of the network that can be exploited in order to realise their malicious intent. The attack strategies used in the training stage can be broken down into three categories:

- **Data Injection** - The adversary usually does not have any access to the training dataset and learning algorithms. However there are other means of performing data injection [4], such as falsifying information during the *incremental learning* [5] phase.
- **Data Modification** - In this case, the adversary has no access to the learning algorithms but does have access to the full training dataset. As such, the adversary can poison [6] the training data by modifying the data before it is used in training.
- **Logic Corruption** - This is when the adversary has the capability to tamper with the learning algorithm of the target model. In practice, this approach is the most difficult to achieve and is often inaccessible, thus usually not preferred by adversaries.

### 2.1.2 Vulnerabilities in the Testing Stage

Adversarial attacks in the testing stage often do not tamper with the target model but instead forces it to make wrong predictions based on irregular input. The effectiveness of such attacks largely depends on the amount of information about the model that is made available to the adversary. These attacks that take place during the testing stage can be divided into two categories:

- **White-box Attacks** - In this case, the adversaries have complete knowledge and implementation details of the target model. [7] This includes its learning algorithm, its training dataset and the weights of each neuron and the respective class indices in the model. As such, it is easier for the adversary to alter an input by using algorithms to generate adversarial examples which are based on the training dataset and targeted at a particular subset of the neurons that are responsible for the intended misclassification.
- **Black-box Attacks** - In this case, the adversaries have no knowledge of the target model. [8] Instead, they analyze the model and its vulnerabilities during the *reconnaissance phase* in order to obtain a list of input-output pairs. Using that information, they then proceed to hypothesize the implementation details of the target model. Attacks of this nature are harder to implement and discover, but once developed, can prove to be easier to execute compared to White-Box Attacks.

### 2.1.3 Incremental Learning

Very often when neural networks are deployed to the various industrial sectors, it is almost a requirement that these networks are enabled with incremental learning. Such applications include autonomous driving, facial recognition, and autonomous robotics [9]. Incremental learning [5] refers to the ability to progressively learn from streaming data which arrives over time without sacrificing model accuracy. This stream of data is further supported by human feedback in order to ensure accuracy. As such, this creates an environment of supervised learning.

One of the challenges introduced with incremental learning is **Concept Drift**[10]. This refers to the changes in the data distribution over time. This is due to the changes in input distribution, or changes in the underlying functionality of the network. One such example is the addition of a new, visually dissimilar object class to a classification problem. Concept drift can be quite problematic because it leads to conflicts in the classification. This happens when there is a new but visually similar class that is introduced in the training data.

### 2.1.4 Adversarial Examples

*Adversarial Examples* are generated malicious inputs to the NNs that produce abnormal behaviour and incorrect predictions. Several algorithms have been written to generate these adversarial inputs, each with their own means and exploited weaknesses of the target networks. Adversarial examples have three basic characteristics [11]:

- **Transferability** - When generating adversarial examples for an attack against a target model, if it succeeds in fooling the classifier or prediction of the target model, then it might work on other models that are in the same scope of application. This is because networks that are trained to perform similar tasks would reuse the same fundamental layers (layers closer to the input) in order to preserve the low-level features that support performance of the intended task(s). As such, vulnerabilities in those layers will be inherent and carried over to subsequent networks as well.
- **Regularization Effect** - Adversarial examples are often used in Adversarial Training [12] in order to reveal information of target model(s) that would otherwise be hidden and used to improve the robustness of the adversarial examples. The obtained information such as

the learning process and original training data could be considered vulnerable avenues for an attack.

- **Transparency** - Adversarial examples aim to have perturbations as minimal as possible in order to avoid detection while still preserving its adversarial capabilities. In some cases, these perturbations are so unnoticeable that even human eyes would not be able to discern between adverse and benign input.

### 2.1.5 Approaches for Generating Adversarial Examples

Several programs were written to generate adversarial examples from benign input to produce the desired misclassification of the neural network when exposed to these adverse inputs. [13]. Approaches used to create the adversarial examples are listed below:

- **Fast Gradient Sign Method (FGSM)** [12] - This method essentially adds noise in the same direction as the gradient of the cost function with respect to the data. The FGSM works by using the gradients of the neural network to create the adversarial example. For an input image, FGSM uses the gradients of the loss with respect to the input image and creates an adversarial image which maximises the loss.

$$adv_x = x + \epsilon * sign(\nabla_x J(\theta, x, y)) \tag{2.1}$$

  - $adv_x$ denotes the Adversarial Image
  - $x$ denotes the original input image
  - $y$ denotes the original input label
  - $\epsilon$ denotes the multiplier to ensure the perturbations are small
  - $J$ denotes the loss
  - $\theta$ denotes the model parameters

  The gradients are taken with respect to the input image because the objective of FGSM is to create an image that maximises the loss. The one and only goal of FGSM is to fool a trained model. Due to the nature of its implementation, it is one of a **White-Box, Untargeted Attack**

- **JSMA (Jacobian-based Saliency Map Attack)**[14] - JSMA is another type of attack that fools classification models, such as deep neural networks that are trained to classify images. By saturating a few pixels in a given image to their local maximum or minimum values, JSMA causes the model to produce the misclassification. JSMA employs the use of a *Saliency Map*, which was originally used for visualizing the prediction process of classification models [15,16]. JSMA is also a gradient based white-box method. It was proposed to use the gradient of loss with each class labels with respect to every component of the input (such as a Jacobian Matrix) to extract the sensitivity direction [17]. Subsequently, a saliency map is used to select the dimension which produces the maximum error. As such, JSMA is **white-box and targeted** in nature. The resultant adversarial example is thus similar to one that is produced by FGSM.

- **DeepFool** [18] - an **untargeted, iterative attack method** that is efficient and produces closer adversarial examples. It assumes that the neural networks are totally linear, with a hyperplane separating each class from one another. The adversarial example is constructed by iteratively searching for a solution within the assumed search space. The resulting adversarial example produced is one that utilises the minimal perturbation needed to fool a neural network into predicting the wrong class. DeepFool outputs the minimal perturbation required to fool in image classifier using an image $x$ to produce classification $f$. It calculates the projection of the input onto the closest hyperplane, otherwise

known as the minimal perturbation. This perturbation is then added to the image for testing. A loop with this calculation is reiterated until the minimal possible perturbation which is able to produce the misclassification is found. As such, a resultant image generated using DeepFool tends to have perturbations that are often unnoticeable by human eyes.

- **Universal Perturbation** [19] - a single small image perturbation that fools a state-of-the-art deep neural network classifier on all natural images. This algorithm leverages on the aspect of transferability of adversarial examples. This means that any adversarial example produced is able to fool a classifier of the same nature, or that was trained to perform the same task. For example, a perturbation for an adversarial image is considered universal if the same image can fool other neural networks that were trained to perform similar task(s). This is due to the inherent basic features of neural networks that were intended for the same types of classification tasks.

- **Carlini and Wagner** [20–22] - Finding the minimal perturbation needed to achieve the change in classification, based on previous formulation of adversarial examples. Carlini and Wagner attempted to produce an adversarial example using a perturbation that is so slight, resulting in an image that is able to change the image classification while still being a valid image. It builds on previous approaches for constructing adversarial examples. Their approach minimises $D(x, x+\delta)$ such that $C(x+\delta) = t$ and $x + \delta e[0,1]^n$, where $x$ is fixed and the objective is to find the $\delta$ that minimizes $D(x, x+\delta)$. In other words, it aims to find the smallest perturbation that can be applied to an image $x$ that will change its classification while still being a valid image.

### 2.1.6 Adversarial Patch

Adversarial examples have been proven to be materialized and robust to environmental factors of the real world [23]. What this means is that an adversarially generated image will retain its adversarial capabilities even when exposed to environmental factors of the physical world. They will continue to be able to fool image classifiers even under different lighting conditions and orientation. One such example is an *Adversarial Patch* [24].

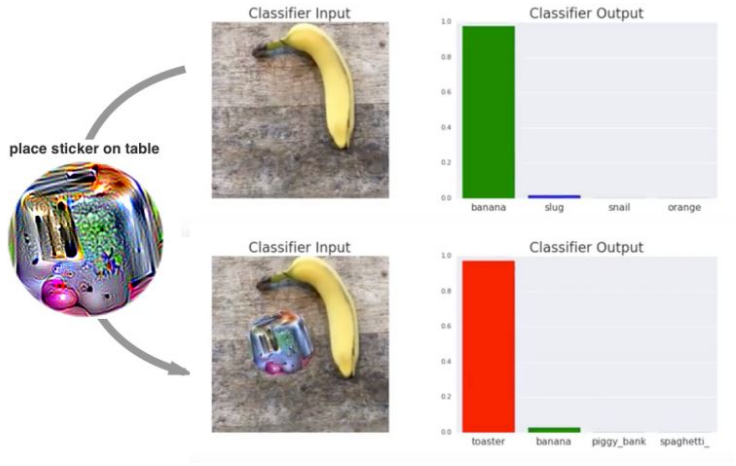



***Figure 2.1:*** *A real-world application of an adversarial patch: Simply placing the sticker of a toaster onto one part of the input image creates the misclassification with a high confidence*

An adversarial patch is a small region or shape that is physically placed onto an image, creating an adversarial example that is able to fool the target neural network. It has been shown in [24] that

when an adversarial patch is printed out and placed within the region of a photographed input image, it is able to fool the neural network, such as the example shown above in *figure 2.1*. The generated patch can be printed out and used on any image, and if distributed to many others over the Internet, can result in a wide variety of malicious intent. Existing defence techniques tend to focus on defending against minimal or small perturbations and hence may not be effective against an attack such as this one, which makes use of a large perturbation but applied to only a small area of the input image.

## 2.2 Existing Attacks Performed Using Adversarial Examples

### 2.2.1 ShapeShifter: Robust Physical Adversarial Attack on Faster R-CNN Object Detector

*ShapeShifter* [25] is one of the programs written to generate adversarial images that are able to fool several state-of-the-art object detectors used in many machine learning systems to date. It suggests that it is much easier to perform attacks on image classifiers rather than on object detectors. This is due to the presence of real-world distortions such as viewing distances and angles, lighting conditions, camera quality and other limitations. The *Expectation over Transformation* technique, which was originally proposed to enhance the robustness of adversarial perturbations in image classification, can be successfully adapted to the object detection setting. The proposed program *ShapeShifter* can generate perturbed stop signs that are consistently misclassified by Faster Region-Convolutional Neural Network(Faster R-CNN) [26] as other objects. This poses a large element of danger to automated systems that rely on image classifiers for safety of its users, such as autonomous vehicles.

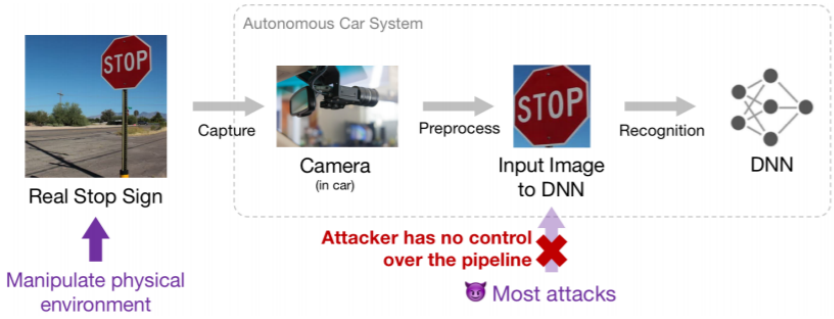



***Figure 2.2:*** *The need of physical adversarial attack from attackers' perspectives, as they typically do not have full control over the computer vision system pipeline.*

*ShapeShifter* was inspired by the iterative *Change-Of-Variable Attack* and the *Expectation Over Transformation* technique.

**Change-Of-Variable Attack**[22] denotes $L_F(x+y) = L(F(x), y)$ as the loss function that calculates the distance between the model's output $F(x)$ and the target label $y$. Given an original input image $x$ and the target class $y'$, the *change-of-variable attack* proposes the following optimization formula:

$$\underset{x' \in \mathbb{R}^{h \times w \times 3}}{\arg\min} \quad L_F(\tanh(x'), y') + c \cdot \|\tanh(x') - x\|_2^2 \tag{2.2}$$

Use of *tanh* makes it so that each pixel is between [-1,1]. The constant $c$ controls the similarity between the modified object $x'$ and the original image $x$. Normally, $c$ can be determined using

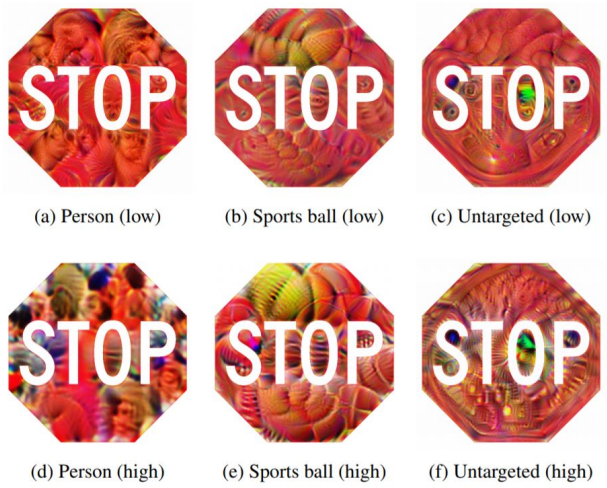
binary search [27].

**Expectation Over Transformation**[24,28] has the simple idea of adding random distortions in each iteration of the optimization process to make the resulting perturbation even more robust. Given a transformation $t$ which can be a rotation or a change in scaling, $M_t(x_b, x_o)$ is an operation that transforms an object image $x_o$ using $t$ and then overlays it onto the background image $x_b$. This process can also include a masking operation that keeps only a certain area of $x_o$. This is helpful to restrict the shape of the perturbation. After considering the random distortions, the overall equation becomes

$$\underset{x' \in \mathbb{R}^{h \times w \times 3}}{\arg\min} \quad \mathbb{E}_{x \sim X, t \sim T}[L_F(M_t(x, \tanh(x')), y')] + c \cdot \|\tanh(x') - x_o\|_2^2 \tag{2.3}$$

where $x$ denotes the training set of background images. When the model $F$ is differentiable, the optimization problem can be solved by gradient descent and back-propagation.

**Digitally Perturbed Images and Evaluation** - The generated adversarial images, while obviously altered from the benign images, are seen very differently when input to the object classifier. This is because certain objects have their own distinct properties. For example, a picture of a truck should show a truck taking up a majority of the image due to its size, a sports ball should always be round, and a stop sign should always be red. By taking advantage of these salient properties, the object detector may be fooled because of the various increased correlation between objects and their respective colours, shapes and sizes.




(a) Person (low) (b) Sports ball (low) (c) Untargeted (low)

(d) Person (high) (e) Sports ball (high) (f) Untargeted (high)

***Figure 2.3:*** *Digitally Perturbed Stop Signs*

An indoor experiment was performed on *ShapeShifter* to evaluate the classifications of a few particular objects, namely a person and a sports ball, when applied to a regular stop sign. The adversarial stops signs shown above were printed out and pictures were taken from various distances and angles. This was to simulate a real-world driving scenario in which the road signs can be identified despite various environmental variables such as distance, angle and lighting.

The high-confidence perturbations had succeeded in the attack despite the variety of distances and angles. The results indicate that the perturbation is less robust to very high viewing angles (about $60^\circ$ from the sign's tangent).

In the case of the low-confidence perturbations, they achieved a much lower success rate, therefore indicating the need for higher confidence perturbations in order to ensure robustness. This was a result that was expected.

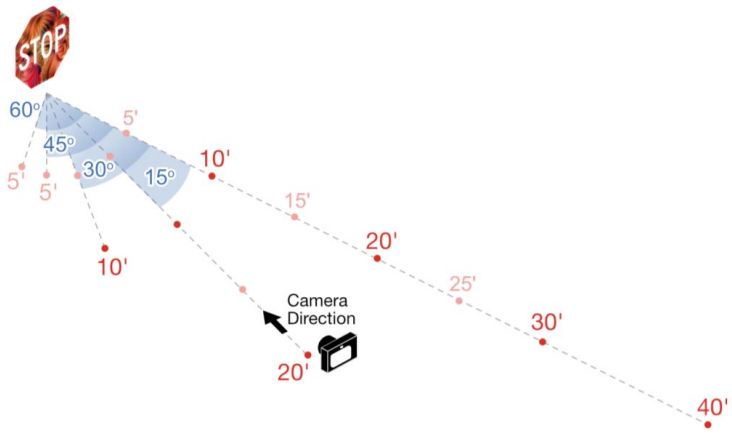



***Figure 2.4:*** *Experimental Setup: Red dots indicate camera locations*

| **Distance** | **Angle** | **person** | (Conf.) | **sports ball** | (Conf.) | **untargeted** | (Conf.) |
|---|---|---|---|---|---|---|---|
| 5' | 0° | person | (.77) | sports ball | (.61) | clock | (.35) |
| 5' | 15° | person | (.91) | cake | (.73) | clock | (.41) |
| 5' | 30° | person | (.93) | cake | (.66) | cake | (.39) |
| 5' | 45° | person | (.69) | cake | (.61) | *stop sign* | (.62) |
| 5' | 60° | *stop sign* | (.93) | *stop sign* | (.70) | *stop sign* | (.88) |
| 10' | 0° | person | (.55) | cake | (.34) | clock | (.99) |
| 10' | 15° | person | (.63) | cake | (.33) | clock | (.99) |
| 10' | 30° | person | (.51) | cake | (.55) | clock | (.99) |
| 15' | 0° | undetected | — | cake | (.49) | clock | (.99) |
| 15' | 15° | person | (.57) | cake | (.53) | clock | (.99) |
| 20' | 0° | person | (.49) | sports ball | (.98) | clock | (.99) |
| 20' | 15° | person | (.41) | sports ball | (.96) | clock | (.99) |
| 25' | 0° | person | (.47) | sports ball | (.99) | *stop sign* | (.91) |
| 30' | 0° | person | (.49) | sports ball | (.92) | undetected | — |
| 40' | 0° | person | (.56) | sports ball | (.30) | *stop sign* | (.30) |
| Targeted success rate | | 87% | | 40% | | N/A | |
| Untargeted success rate | | 93% | | 93% | | 73% | |

***Figure 2.5:*** *Experimental Results*

### 2.2.2 Procedural Noise Adversarial Examples for Black-Box Attacks on Deep Convolutional Networks

Once again, DCNs have been shown to be vulnerable to adversarial examples. This paper [29] presents a universal, procedural noise perturbation that when applied to an image, generates an adversarial image that is able to fool the model and produce a misclassification. The nature of this attack is one of a untargeted, black-box attack. Performing the attack employs the use of two key elements, namely **Perlin Noise** and **Gabor Noise**.

**Perlin Noise**[30] boasts ease of use, popularity and simplicity. It was developed as a technique to produce seemingly natural textures for computer graphics. Simple implementation and the ability to be controlled by a few parameters makes it feasible for inexpensive black-box attacks. Perlin Noise is represented by a two-dimensional matrice. The eventual noise function has several parameters which determine the visual appearance of the noise.

For the construction of a two-dimensional Perlin noise, the value at an arbitrary point $(x, y)$ is

derived as follows: let $(i, j)$ be the four lattice points of the lattice square where $i = |x|, |x| + 1$ and $j = |y|, |y| + 1$. The four gradients are then given by $q_{ij} = V[Q[Q[i] + j]]$ where precomputed arrays Q and V contain a pseudo-random permutation and the respective pseudo random unit gradient vectors. The four linear functions $q_{ij}(x - i, y - j)$ are then bilinearly interpolated by $s(s - |x|)$ and $s(y - |y|)$, where $s(t) = 6t^5 - 15t^4 + 10t^3$. The result is the Perlin noise value $p(x, y)$ for coordinates $(x, y)$.

The Perlin noise function has changeable parameters that determine the visual appearance of the noise. In the case of [29], the wavelengths $\lambda_x, \lambda_y$ and the number of octaves $\omega$ contribute the most to the visual changes. The noise value at point $(x, y)$ with parameters $\delta_{per} = \lambda_x, \lambda_y, \Omega$ becomes the following:

$$S_{per}(x, y) = \sum_{n=1}^{\Omega} p(x \cdot \frac{2^{n-1}}{\lambda_x}, y \cdot \frac{2^{n-1}}{\lambda_y}) \tag{2.4}$$

**Gabor Noise**[31] is used here as a representative example for sparse convolution noise. It boasts a higher accuracy of spectral control in comparison to other procedural noise functions. Spectral control here refers to the ability to control the appearance of noise as measured by its energy along the respective frequency bands. It can be analysed at any point in space and is customized using only a few parameters such as frequency, orientation and bandwidth.


***Figure 2.6:*** *From left to right: benign image, adverse image and procedural noise (magnified for visibility)*

Due to the untargeted nature of the procedural noise function, the objective of the generated adversarial examples is to simply exploit the model's sensitivity, causing significant changes in the prediction by applying minimal perturbations to the input image. Recent work has shown that adversarial examples exploit "non-robust" features which the model learns, but are incomprehensible to humans. As such, it is plausible that Universal Adversarial Perturbations (UAPs)[32] may be exploiting universal "non-robust" features that DCNs learn. Such features include the following:

- **Textures** - DCNs tend to be more reliant on textures rather than shapes. This is because of the Deep-Learning aspect of the overall network which was made to recognise faces and objects in the same way that humans do.
- **Generalizability of Procedural Noise** - [29] suggests that most perturbations with high universal evasion do not tend to have a strong bias towards any particular class label. This means that while the idea of having a query-efficient and inexpensive untargeted attack is plausible, the same can not be said for an attack of a targeted nature.

- **Security Implications** - In transfer learning, a model that was trained to only perform one task often has its foundation layers (those of which are closer to the input) reused in order to preserve the low-level features that are fundamental for performing said task. This means that in order to perform other similar tasks or improvement of the same task, only layers closer to the output are re-trained for the new task. This makes DCNs a prime target because the vulnerabilities that exist in the model's low-level features and layers will carry over to the subsequent models.

## 2.3 Defenses Against Adversarial Attacks

### 2.3.1 Adversarial Training

**Adversarial Training** [33] refers to the training of models to be robust against adversarial examples. The objective of adversarial training is to train the neural network to classify adversarial examples correctly. In order to achieve this effect, a portion of the training data will include known adversarial examples and their true class labels. In doing so, the neural network will be able to identify a known adversarial example correctly. However, this approach implies the need for higher computational resources, as well as a significant loss in accuracy. Additionally, adversarial training is known to generalize poorly to stronger attacks such as adversarial examples that utilise larger perturbations. One major point to note is that adversarial training is not a long-term solution. This is because the resultant neural network will only be robust against the adversarial examples that it was trained on, and not subsequent adversarial examples that were generated or presented in different ways. Adversarial training is only able to achieve correct classifications of known adversarial examples, but not the detection of adversarial input in general.

### 2.3.2 Defensive Distillation

**Defensive Distillation** [34] is the process of training a neural network (the student) using the output of another network (the teacher). In the initial stages, this process was aimed at reducing the computational resources required for using a neural network and performing the intended task(s). This is because the distilled network is allowed to have fewer neurons, since the fundamental layers of the original network have already served their purpose. To put this process into clearer perspective, the original model is called the "teacher" and the distilled model is called the "student". The main objective of defensive distillation is to provide the student model with as much knowledge about the interaction between the various class labels, rather than a simple indication of the correct class. Defensive distillation is also meant to increase the resilience of the distilled network against adversarial examples by employing the use of gradient masking which prevents the adversary from making use of the loss gradient to generate adversarial examples. Defensive distillation has been known to be an upgraded version of adversarial training which adds flexibility to the classification process such that it is less susceptible to exploitation. While defensive distillation is an approach that is adaptable even to unknown threats, the biggest disadvantage is that the "student" or distilled model is still bound by the fundamental implementation and features of the "teacher" model. As such, the vulnerabilities in the original model may be carried over to the distilled model.

## 2.4 Repercussions of Defensive Measures

Since the discovery of adversarial perturbations against neural networks, the race between attack and defence methods has intensified in the recent years. In light of the works mentioned above, newly-discovered attack methods are based on creating robust adversarial examples that are perceptually indistinguishable from legitimate input and yet produce incorrect classifications.

While initial discovery of attack methods reveals the simplicity of attacks that are white-box in nature, the same can also be done for black-box attacks. Discovery of attack methods prove to be much simpler as compared to discovery of defensive methods. The task of blocking adversarial perturbations, or even distinguishing them from legitimate input, has been fairly difficult. [20,21] Many defensive measures have been proposed but have seen limited success. The number of failed attempts at defending against adversarial examples suggests that there has to be some fundamental reason(s) that make adversarial perturbations and their generated examples so robust and effective.

### 2.4.1 Why Blocking Targeted Adversarial Perturbations Impairs the Ability to Learn

This paper [35] provides a deeper analysis on the characteristics of a neural network classifier and why blocking targeted adversarial examples implies losing the ability to learn. It reiterates the following fundamental facts:

- The order of the various classes within the neural network is arbitrary. This means that different mapping of classes to class indices will not affect the network's ability to train, nor will it affect the network's resulting accuracy.
- The network loss is a monotonically increasing, non-negative function. It is equal to zero when the network predicts the true class label, and it increases accordingly with the classification error.
- Classifier networks are trained using stochastic gradient descent starting from a random set of weights. During the training phase, in order to converge in a reasonable amount of time, a high loss gradient is required when the classification error is high. Similarly, a low loss gradient is required when the classification error is low.

With the above-mentioned facts and the analysis provided in the paper, the authors have come to the conclusion that since the targeted adversarial examples leverage on the same input gradient that allows a network to be trained, this means that blocking them will mean that the network will no longer have the ability to learn. All in all, this calls for a new means of protection against targeted adversarial attacks.

### 2.4.2 Research Key Takeaways

Through the analysis and consideration of previous related works when it comes to methods of attacking neural networks and defending against said attacks, the following concepts are crucial and were taken into consideration whilst developing the attack:

- Existing methods of generating adversarial examples are aimed at exploiting the vulnerabilities of DCNs, such as **texture-bias** and **generalization**.
- While existing attacks have proven to be **inexpensive to perform**, are they really **practical**? In practice, attackers will **not have access to the training data** of the target model, and attacks would have a **target class or class label** in order to achieve their malicious objectives.
- Should there be the existence of a **robust, targeted black-box attack**, how far can it go when it comes to **stealth and minimal perturbation** while still being able to achieve its malicious objective(s)?
- In terms of generalization, to what extent can we expose the generated adversarial examples to physical environmental factors and how will their adversarial capabilities be affected?
- Since defending against adversarial examples could implicate and impede the network's ability to learn, perhaps the learning ability is already a vulnerability to begin with.

# 3 Analysis/Requirements

This chapter describes the problem domain in detail and the requirements and considerations as a result.

## 3.1 Problem Domain

The idea was to produce an adversarial attack that is both impactful and inexpensive to perform. In practice, it would be close to impossible for adversaries to *gain access to the data on which their target neural network was trained on.* As such, it is clear that a **Black-Box Attack** is more practical and universal. **Ease of use** would also be preferred, such that the attack can be performed inexpensively both in terms of time and computational resources. Additionally, the choice of a *untargeted* or *targeted* attack depends on the malicious objectives of the adversary. In order to fully assess the potential impact of adversarial attacks, the scope of **targeted attacks** was chosen. As such, it would be ideal that the product to be implemented is one of a **Targeted Black-Box Attack**. Once the attack has been accomplished, the vulnerabilities that were exploited need to be identified. Subsequently, we propose potential defensive measures that could be adopted.

## 3.2 Attack Consideration

Before designing the attack method, the following factors had to be considered:

- **How can a Targeted Black-box attack be achieved?** - If this was made possible, we would have an attack that is not only more practical, but also more capable in terms of adversarial objectives. Less information needed to carry out the attack would also translate into the potential of similar approaches being used to develop attacks on other types of neural networks.
- **How can we identify the vulnerabilities of the target network?** - The target neural network needs to be studied and understood well enough in order for any vulnerabilities to be derived.
- **Can the attack be robust enough to demonstrate the aspects of generalization and environmental conditions such as light intensity and angle of incidence?** - While there are existing attacks which are efficient in achieving their adversarial objectives, how robust are they against real-world environmental factors? The ideal adversarial attacks should not only exist in digital space but also in physical space as well. This is because physical adversarial attacks are ultimately easier to execute, even for individuals without technical expertise.
- **What applications of neural networks should be looked into?** - The application of the target neural network will determine the respective impacts when the neural network is compromised. These impacts need to be assessed in order to justify how detrimental the attack can be. For example, if an autonomous vehicle system is compromised, it could put the passengers' and pedestrians' lives at risk.

- **What kind of computational resources are required?** - Execution of the attack should be as inexpensive as possible, both in terms of computation resources and time. In the best case, the attack procedure should be able to be replicated and distributed widely, thus making it easily used by other adversaries. In other words, it would be ideal for execution of the attack to be as fast and simple as possible. However in most cases, there is a very large trade-off between the initial attack setup and subsequent executions of the attack by other adversaries.
- **What kind of experiments can be carried out for this attack?** - The experiments should accurately identify the strengths and limitations of the attack in achieving its adversarial objectives. These experiments should also support the portability of the attack in terms of ease of use and distribution.
- **What kind of metrics should be used to evaluate effectiveness of the attack?** - The metrics must be measurable by experiments and subsequently using those numbers to present and evaluate the effectiveness of the attack. For example, we should be looking at the confidence scores of the classifications, whether they are for the true class or the target class. The success rate of the attack needs to be as high as possible before the robustness can be looked into.
- **What kind of attack on neural networks would create a large impact on its users?** - Once an adversary compromises the target neural network, what can the adversary achieve and what are the respective impacts on the system's end users?

### 3.2.1 Adversarial Examples

**Adversarial Examples** seem to be the easiest to be replicated by other potential adversaries. It is also the most flexible form of attack because of the wide variety of possible perturbations. Based on previous works, it is apparent that the more robust adversarial examples tend to have the largest perturbations and thus most noticeable to humans. This is a given, especially when images can be considered disfigured if it seems completely unrecognisable to human eyes. The idea is to create a perceptual space, in which the generated adversarial example is one that humans would be able to confidently identify what the image "should be" classified as, and yet realise that the neural network has made the wrong classification. Another important metric to consider is the confidence score of the wrong classification. The confidence score of a wrong classification can also indicate the degree to which the neural network was fooled. Very often in attacks involving adversarial examples, a large amount of complexity lies either in the adversarial image generation phase or the adversarial retraining phase.

The robustness and accuracy of the adversarial examples would determine the overall effectiveness of the attack. In other words, the adversarial examples generated should be robust to various environmental factors whilst being able to accomplish its intended malicious objectives. Generalization is also a key factor that determines how easy it is for anyone to perform the attack. Adversarial examples that are able to generalize into physical space would see more potential use, especially by adversaries who do not have technical expertise. Stealth is also important for adversarial examples. Attacks that avoid detection would be able to cover their tracks and prolong the duration whereby an adversary can carry out their intended activities.

# 4 Attack Design: Adversarial Patch Attack Against Image Classifiers

In this chapter, the implementation of the *Adversarial Patch Attack* is presented along with the threat model and the relevant steps of attack construction. The *Adversarial Patch Attack* presented will demonstrate the vulnerabilities of a state-of-the-art DCN such as *VGG-Face* [36], exploit said vulnerabilities, and further integrate existing forms of attacks in order to achieve the possible objectives of an adversary. Upon identifying the avenues of attack, we then describe how these attacks can be carried out easily and inexpensively while being able to achieve a successful and impactful attack.

## 4.1 Threat Model

Before introducing the adversarial attack, we first identify the threat model:

- Adversary has access to the target neural network
- Adversary has no access to the data on which the target neural network was trained on
- Adversary can perform data injection during the training phase
- Adversary can craft their own input(s)

## 4.2 Assumptions

We assume that the adversary has access to the target neural network, but not any access to the data on which the neural network was trained on. In order to conduct the attack, the original model is tampered with by retraining it using additional data that can be crafted by the adversary. The objective is to produce an altered version of the input model that will behave normally given benign input while producing adverse behaviour given an input image stamped with the adversarial patch.

## 4.3 Attack by Model Inversion

**Model Inversion Attacks** [37] are used to retrieve training data of published neural networks in order to reveal sensitive information about particular individuals. The attack proposed in this paper makes use of this approach to probe for the information of a target class and thus use it in the adversarial patch and to generate training dataset(s). DCNs are known to have strong connections between features and labels, which is exactly what *Model Inversion Attacks* exploit. In doing so, an adversary will be able to obtain relevant information of their target neural network and use it to their advantage. For this attack, access to the target model's implementation details and training data is not assumed. In this case, we employ *Model Inversion* techniques to probe and extract key pieces of information for the attack to work.

## 4.4 Vulnerabilities of Deep Convolutional Networks

DCNs were developed with the agenda of enabling machines to view the world just like humans do, thus allowing machines to perceive information in a similar manner and even use the knowledge for tasks such as image recognition and anomaly detection. DCNs tend to put a level of importance onto various objects in an image in order for the neural network to differentiate between input images. This results in learnable weights and biases from neuron to neuron. As such, the resultant neural network is one that boasts strong interconnections between neurons(features) and output classifications.

With the relevant understanding of DCNs for facial recognition, we are able to derive the following key features of a facial image classifier that can be exploited and turned into vulnerabilities:

- **Texture-Bias** - Image classifiers tend to be heavily reliant on textures. In the case of a facial image classifier, such textures include skin tone and facial features. While this feature is intended to further improve the accuracy of classifications, over-reliance on it can lead to detrimental effects. For example, if the texture of person *a* was applied to an image of person *b*'s face, a facial recognition DCN may classify it incorrectly as person *a* even though it still looks like person *b* in the eyes of a human.
- **Strong Interconnecting Layers** - While this was intended to increase the confidence and accuracy of predictions, it can also be turned into a vulnerability. This example is similar to the learning environment of humans; students will take their teachers' words as the truth, but who then will validate the ground truth of the teacher's words? In other words, the information passed from neuron to neuron is assumed to be trusted. As such, should any information in the neural network be tampered with, subsequent layers will assume that the information (weights) have not been tampered with. This induces a causal chain effect from a target neuron, all the way to the respective output.

## 4.5 Attack Overview & Architecture

**Overview** - In a neural network, a single internal neuron is often viewed as an internal feature. The different features tend to have various impacts on the final output based on the weights and links between neurons and output class labels. The attack probes for the neurons that are strongly linked to the adversarial patch and then retrains the specific links from those particular neurons all the way to the outputs in order to facilitate manipulation of the output. In this manner, the targeted misclassification can be achieved with a significantly high confidence score that is amplified by the strong links between target neurons and the output class labels. The attack is **Targeted and Black-Box** in nature, since we assume that the adversary has no access to the original training dataset and training process. This attack was performed on a state-of-the-art facial recognition DCN called VGG_FACE, which consists of 38 layers and over 15 million neurons [38,39]. *VGG_Face* has a large-scale facial recognition dataset (3.31 million facial images) which consists of a wide variety of poses, ages, ethnicity and professions of known people. The face distribution for different identities varies from 87 to 843, with an average of 362 images for each subject (person).

### 4.5.1 Attack Strategy

- **Phase 1: Adversarial Patch Generation** - During the reconnaissance stage of the attack, *Model Inversion* techniques are employed to identify the internal neurons which have the strongest reaction to an input shape, as well as relevant information of our target class. Using these key pieces of information, we generate an adversarial patch that is able to fool *VGG_Face*.

- **Phase 2: Training Data Generation** - By amplifying the textures of facial images and applying the adversarial patch, we craft images that we can then feed to the target model, as a simulation of the training phase.
- **Phase 3: Retraining** - By retraining the target model, we perform data injection using the crafted training images in order to manipulate the weights and biases that interconnect the target neuron, all the way to the output classification. This induces a causal chain effect that allows us to perform the targeted attack.

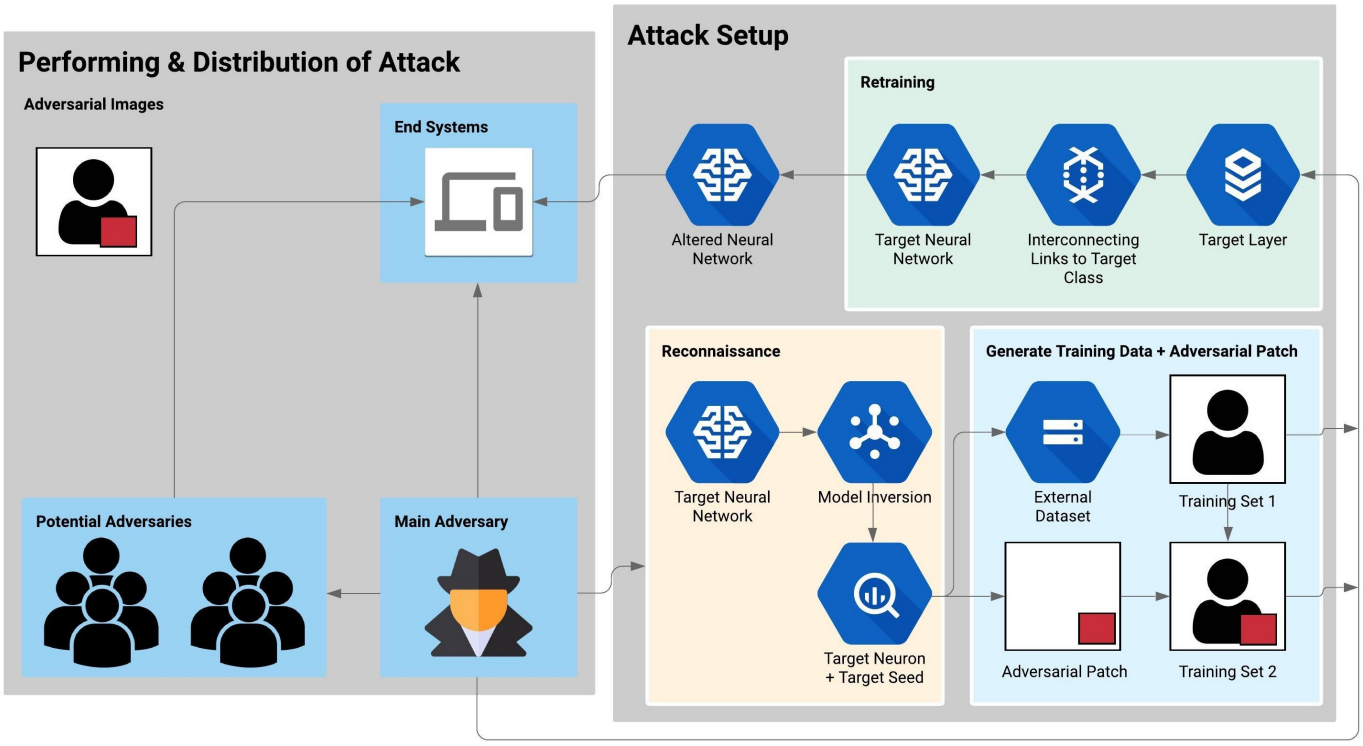



*Figure 4.1: Attack Overview*

### 4.5.2 Phase 1: Adversarial Patch Generation

The objective of the adversarial patch is to create an adversarial example from the benign image(s). An adversarial example is a perturbed version of the benign image that when input to the neural network, produces a classification that deviates from the image's true class. This also means that the model will still function normally in the absence of the adversarial patch. The adversary starts by choosing a shape and a location. The shape will essentially be the container for the adversarial patch, and the location refers to the region of the image on which the adversarial patch would be stamped on. For all of the adversarial examples generated in this attack, a simple square shape was chosen as the container for the patch in order to preserve simplicity. In practice, the adversary would be able to choose any shape and region that they want. During the adversarial patch generation, the target neuron(s) in a chosen target layer that show strong reaction to the location and shape selected by the adversary would be identified via probing.

The neurons are selected based on the following criteria:

- Neurons that produce high activation (strong reaction) to the given shape and location
- Neurons that have strong connections to subsequent output layers

The attack then runs an adversarial patch generation algorithm which iterates though various parameters for the adversarial patch in order to achieve the highest possible activation and confidence scores. A seed number of the adversary's choice is also considered. The seed number correlates to the target class label of the neural network. In this case, a seed of '0' was chosen with the safe assumption that all neural networks will have a class label '0', otherwise known as the very first class. The relevant textures of the seed are then compressed and masked over the adversary's chosen shape, thus forming the adversarial patch. In this manner, the attacker is able to create a robust, targeted adversarial patch without any prior knowledge of the target NN and its relevant implementation details. This was possible due to VGG_FACE's reliance on facial features (textures).

The secondary objective of the adversarial patch generation phase is to identify the connecting neurons that produce strong activation scores towards a chosen shape and region, in turn inducing a causal chain effect from the reacting neuron all the way to the output class label during the inference phase, when the model is exposed to the adversarial patch. This idea is further reinforced in the subsequent phases of the attack.

### 4.5.3 Phase 2: Training Data Generation

Since no access to the original training data is assumed, an external dataset had to be derived and used to retrain the model. In doing so, the idea was to perform a simulated data injection during the incremental learning phase of the target neural network. This was so that the model's original functionalities were retained while still being able to achieve the adversarial attack. In order to achieve this, input training data image had their textures amplified to produce a stronger activation of the target neurons. The process starts by averaging[40] the images obtained from a public dataset. This dataset is such that when the images are input to the model, they are classified with a very low confidence score. The low confidence scores are an indication that the model has not been trained on these images before. The input texture-amplification algorithm alters the parameters of the image such that a larger confidence score is produced for the output node(s). In other words, the texture-amplified image can be considered as a replacement for the original image, whilst achieving higher accuracy for the intended task. This can be seen as a form of enhancing the image and its features. This process is repeated for each image in the public data set. Retraining using this dataset serves the same purpose as training the NN using the benign image. In other words, training using the original dataset or the texture-amplified dataset will produce comparable accuracy. The only difference here is that the retrained model would have an even heavier reliance on textures, thus reinforcing the causal chain effect as described in the previous phase.

The external dataset used was from a subset of the *Labelled Faces in the Wild* [41] database, which consists of about 6,870 facial images of 1680 better known personnel. Each of the images in the dataset will have their textures amplified and subsequently denoised to form the first set of training data. Subsequently, each image in the first training dataset was stamped with the adversarial patch and this formed the second set of training data.

### 4.5.4 Phase 3: Retraining

The two datasets produced were used to retrain only a part of the original model, namely the layers in between the layer in which the target neurons reside and the output layer. The design was as such because retraining the entire model will be computationally expensive and time consuming, especially due to the complexity of the target NN. The retraining process was performed such that for each texture-amplified training image $I$ for a person $B$, we also have another training image $I$ stamped with the adversarial patch but intended for classification of person $A$, the target class. After retraining, the output model becomes even more reliant on textures. As such, the weights of the original model were altered such that the new model still

produces normal behaviour in the absence of the adversarial patch, while producing the intended adverse behaviour in the presence of the adversarial patch.

As such, the retraining phase has the following advantages:

- Significantly shorter training time
- Fast development of the attack
- No need for access to the original training dataset

The objectives of the retraining phase were as follows:

- To establish a strong connection between the target neurons and the adversarial patch.
- To reduce the other weights in the target NN, in particular those that are not correlated to the target class *A*, thus making up for the increased weights.
- To ensure that when a benign image is input to the model for classification, the new model will still produce the correct classification.

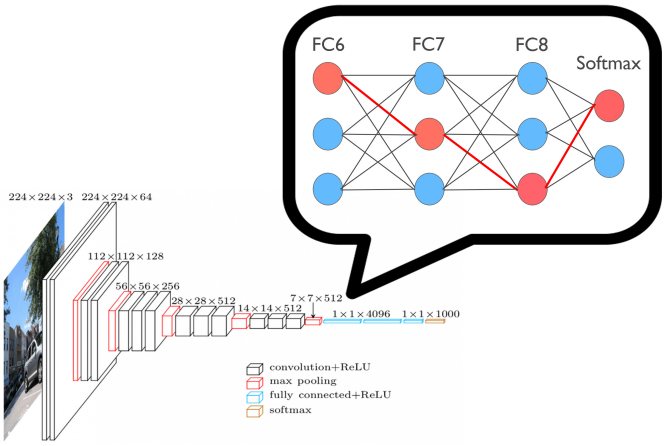



***Figure 4.2:*** *Retraining Phase*

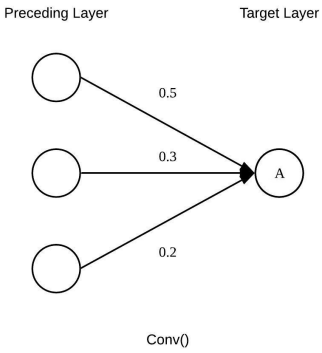



***Figure 4.3:*** *Original NN*

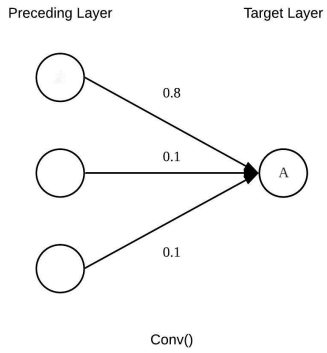



***Figure 4.4:*** *Altered NN*

The combination of the three phases of the attack produces a causal chain that is triggered by exciting the nodes or neurons in the target layer with their inflated weights in order to produce the desired classification of the target class, given that the model is exposed to the adversarial patch.

# 5 Implementation

This chapter describes the relevant implementation details that made the above-mentioned designs possible. The implementation details will also be broken down into their relevant phases of the attack.

## 5.1 Framework and Resources Used

This section describes the relevant framework(s) and core resources used during the implementation phase of the product.

### 5.1.1 Caffe: Deep Learning Framework

**Caffe** [42] was chosen as the framework over *TensorFlow* and *PyTorch* due to its **speed of inference and learning**. It is among one of the fastest implementations made for convolutional networks that is currently available. One crucial factor for this choice was *Caffe*'s online library of pre-trained models, the **Caffe Model Zoo** [43]. Since the objective was to perform the attack and make it work on state-of-the-art models, it made perfect sense to make use of this library to look for the *target neural network* on which to perform the attack on. Most of the models made available in the *Model Zoo* have been deployed in one or more business solutions already, which make it the perfect resource for the sake of this project's research. On top of it all, *Caffe* supports *model retraining*, which opened up more avenues for which to perform the attack.

### 5.1.2 Theano

**Theano** [44] is one of the core *Python* libraries used to handle multi-dimensional arrays. It works well with *NumPy* [45] to handle image and input arrays such as image transposing and image *RGB* manipulation. *Theano* also allowed for more efficient use of *GPU*, especially during the learning phase of deep neural networks. This factor was tantamount to the research because it severely contributed to a much more effective use of time, what with the limited amount of time given to undertake the project. *Theano* has been known to be able to handle expensive computations required for large neural networks used in *Deep Learning*[46], hence making it a prime inclusion during the implementation phase of the project.

### 5.1.3 Pickle

**Pickle** [47] is a commodity in machine learning productions. This is because *Pickle* allows for the saving of a model after training and exporting it into a file for subsequent inference. This means that a massive amount of time was saved because there was no need to perform repeated training, and each exported model serves as a checkpoint or state of the neural network after training. This is extremely useful for research projects such as this one because it allowed for the effective testing of features. If a particular new feature was not feasible, one could simply discard the exported model file or save a copy of it for future reference if needed. In this project, *Pickle* was particularly useful in conjunction with *Caffe* because the retraining phase generates multiple

*Pickle* model files, one for each layer of the neural network. After retraining is completed, all the *Pickle* files were compiled using its *read binary* function into a single *caffe* model file.

## 5.2 Adversarial Patch Generation

As discussed before, given a chosen shape and region, the adversary is able to generate the adversarial patch which excites the target neurons in the target layer. The entire image consisting of a benign image stamped with the adversarial patch will form an adversarial example. For every benign image $x$ that was part of the original training dataset, there exists a perturbed version of the image $x'$ that when analysed by the neural network, produces a wrong classification.

**Algorithm 1:** Adversarial Patch Generation Algorithm

**procedure** PATCH GENERATION($BM, tl, shape, \{(n1, tv1), (n2, tv2), ...\}, limit, iter, lr$)
$func = model[tl]$
$cost = (ts1 - func_{n1})^2 + (ts2 - func_{n2})^2 + ...$
**while** $cost > limit$ and $i < iter$ **do**
  $g = \partial cost / \partial x$
  $g = g \circ M$
  $shape = shape - lr \cdot g$
  $i++$
**end while**
**return** x

Algorithm 1 represents the adversarial patch generation algorithm. It employs the use of gradient descent [48] to find the local minimum of a cost function, which is the difference between the current activation scores and the target activation scores of the selected neurons. Given the initial value assignment, the process iteratively refines the inputs along the negative gradient of the cost function such that the final scores for the selected neurons are as close to the target scores as possible. *BM* denotes the original model, *tl* denotes the target layer, *shape* denotes the chosen shape to contain the adversarial patch, $(n1, ts1), (n2, ts2), ...$ represents the various neurons with their respective target scores, *limit* is the threshold for which the process terminates and *iter* represents the maximum number of iterations that should be allowed. *lr* denotes the learning rate and *g* denotes the gradient. The function takes the model as input and produces the neuron scores at the specified target layer. The cost function calculates the mean square error between the scores of the target neurons and their target scores. While the cost calculated is less than our threshold and we are still within our maximum number of iterations, the algorithm generates the gradient of the cost function with regards to the input shape. Finally, the adversarial patch is contained within the shape, and the overall patch is transformed closer towards the gradient, along the defined learning rate. For the examples used in this project, the adversarial patch was generated using the target layer *fc6*, which is one of the facial classification layers that is closer to

the output. In doing so, we save time in the retraining phase.

**Algorithm 2:** Adversarial Patch Generation: Pseudocode

```
Result: Adversarial Patch Generated
initialize shape, target layer and region;
initialize highest score = 0;
probe for target neuron using selected shape;
while i < epochs do
    test score;
    while score < highest score do
        regenerate patch for target neuron;
        if score > highest score then
            highest score = score ;
            i++;
        end
    end
end
```

## 5.3 Neuron Selection

As shown in *algorithm 2*, we chose a number of neurons that will be used for the generation of the adversarial patch. During selection of neurons, neurons that have poor connections between the preceding and subsequent layers were avoided. This is because neurons of such nature will be unable to create the intended causal chain.

$$layer_t = layer_p * conv(W + b) \tag{5.1}$$

$$argmax(\sum^{n} ABS(W_{layer_t})) \tag{5.2}$$

In order to achieve this, we check the weights between the target layer and the preceding layers. Equation (5.1) shows how this was done. $W$ represents the weights, $b$ represents the biases, $conv()$ represents the convolutional computation between layers. $layer_t$ represents the target layer we want to inverse, and $layer_p$ represents the preceding layer. Equation (5.2) shows how the selection of neuron was done. The neuron that has the largest value of the sum of the absolute weights that connect the particular neuron to the preceding layer will be picked. In other words, we want to find the most connected neuron. During implementation, the square shape was chosen.

## 5.4 Training Data Generation

Given the target output classification label, the objective of the retraining phase is to generate a set of training data with amplified textures which takes advantage of the texture-bias to excite the target neuron(s) with significant confidence. Visually, the texture-amplified images look very different from the original images.

In *algorithm 3*, *model* denotes the original model, *nno* and $tv$ denotes the target neuron and its respective target value. $g$ denotes the gradient, $lr$ denotes the learning rate or the input change rate along the negative gradient of the cost function. *limit* denotes the threshold or the maximum score and $iter$ denotes the maximum number of iterations, both of which are customizable by the user. The objective here is to amplify the textures of the input image.

**Algorithm 3:** Training Data Generation

```
0: procedure TRAINING DATA GENERATION(model, nno, tv, limit, iter, lr)
1: x = init()
2: cost = (tv − model(nno))^2
3: while cost > limit and i < iter do
4:     g = ∂cost/∂x
5:     x = x − (lr · g)
6:     x = denoise(x)
7:     i + +
8: end while
9: return x
```

The input data is initialised first. The initial input could be completely arbitrary or derived from domain knowledge. To amplify the textures of the inputs for the face recognition model, *init()* produces an initial image by averaging the images from the external dataset. Compared to a random initial image, this significantly reduces the search space for input texture amplification. Subsequently, the cost function is the mean square error between the output label value and its respective target value. Gradient descent is used to find the value of $x$ that minimises the cost function. In other words, the gradient with respect to $x$ is computed. $x$ is then transformed towards the gradient along the learning rate. After which, a *denoise* function is applied to $x$ to reduce noise from the generated inputs, such that better accuracy in the retraining phase is achieved. Essentially, the retraining dataset is obtained by texture-amplifying the external dataset of faces. The gradient was calculated using back propagation [49]. As such, the computational cost of the training data generation is proportional to the dimensions and size of the input dataset, as well as the complexity of the target model.

## 5.5 Denoise Function

The denoise function [50] aims to reduce noise in the training data. Denoising is commonly done on training data before use during training. In doing so, we minimise the effects of noise on the training images and thus achieve increased stability and accuracy for the intended task. The images generated by gradient descent are very noisy and appear extremely unnatural, hence the need to denoise them. Shown below is a texture-amplified image before and after denoising.

***Figure 5.1:*** *Texture-Amplified Face Image: Before Denoising*

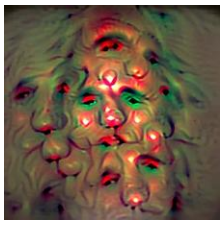


***Figure 5.2:*** *Texture-Amplified Face Image: After Denoising*

The noisy image is not optimal for the retraining phase because the altered model may detect these differences as intended features and use them during inference. Ideally, the altered model needs to detect more semantic features. As such, the denoise function was necessary in order to improve the accuracy of the altered model. The denoise function reduces noise by minimising the total variance [51]. The idea is to reduce the differences between each input element and its neighbouring elements.

# 6 Evaluation

This chapter describes the relevant experiments conducted to evaluate the effectiveness and efficiency of the *Adversarial Patch Attack*, based on the following key criteria:

- **Ease of Use** - The attack should be easily performed and simple enough such that the potential adversary need not have any knowledge of the implementation details of the attack, and the attack should take a feasibly short amount of time to replicate.
- **Consistency of Attack** - The attack should perform well enough such that the desired objectives of a potential adversary could be achieved to a certain extent, and the attack should not fail beyond a certain margin of tolerance.
- **Distribution of Attack** - Once the attack has been performed by one adversary, it should be easily distributed to other potential adversaries in order to achieve a large enough impact to consider the attack successful and effective in attaining its relevant objectives.
- **Stealth of Attack** - An ideal attack should have various elements to avoid detection. The current state of Artificial Intelligence systems is very often one of a supervised learning environment, thus it would not suffice to avoid detection by the system. Avoiding detection from human eyes would be an element that is equally important.

Experiments were also carried out to measure the following:

1. **Accuracy of Adversarial Example** - The success rate of the attack performed using the adversarial patch, and the relevant accuracy of success/failure. In other words, how often and how accurate the attack is in achieving its objective(s).
2. **Transparency of the Adversarial Patch** - To what extent can the adversarial patch remain 'hidden'and still retain its adversarial capabilities, such that it may avoid detection from both the neural network and from human eyes.
3. **Generalization & Robustness** - To what extent can environmental conditions of the real world affect the capabilities of the *Adversarial Example* generated as a result of applying the adversarial patch. In other words, when exposed to factors such as picture quality, lighting intensity and angle of capture, will the *Adversarial Example* still be able to fool the neural network?
4. **Stealth of Altered Model** - To what extent does the altered model retain the original functions? Ideally, the altered model should still be able to perform the original task that it was originally trained for, and yet produce the intended adverse behaviour when exposed to the adversary's crafted input(s). This is another way of avoiding detection, hence introducing the element of stealth.

### 6.0.1 Experimental Input/Data

Before the experiments were carried out, the following experimental input images were obtained/generated:

- **Dataset 1: Benign Images** - This dataset comprises of a total of 48 facial images of different personnel, each with one image. Out of the 48 faces, 24 of them were part of the model's training data and the other 24 were facial images of personnel that were not

part of the training data. These other 24 facial images were obtained from *Google* searches. The two sets of 24 faces were chosen such that they had an equal number of faces of the following categories:

- 4 Male Caucasian
- 4 Male African-American
- 4 Male Asian
- 4 Female Caucasian
- 4 Female African-American
- 4 Female Asian

The facial images were obtained in this manner with consideration that the facial recognition models may produce varying results depending on the various facial features exhibited, such as gender and skin tone. None of the images in this dataset have been tampered with before use in the experiment(s).(For examples, refer to figures: A.3,A.4,A.5,A.6.)

- **Dataset 2: Adverse Images** - This dataset comprises of the adversarial versions of the images in **Dataset 1**. All of the original images are now stamped with the adversarial patch. As such, this dataset also contains 48 facial images and maintained the equilibrium of images as in **Dataset 1**. (For examples, refer to figures: A.8,A.9,A.10.)
- **Dataset 3: Adverse Images with Transparency of Adversarial Patch** - This dataset comprises the same adversarial images as in *Dataset 2*, but now with varying degree of transparency of the applied adversarial patch. The levels of transparency were applied at intervals of 0.1 increments. The transparency levels of the patch ranged from 0 to 1 (1 means an opaque adversarial patch, 0 means no patch). As such, each face/personnel would have 11 images with varying levels of transparency. As a result, this dataset contains a total of 528 facial images. (For examples, refer to figures: A.11,A.12.)
- **Dataset 4: Adverse Images with Transparency of Adversarial Patch | Grouped by Facial Visibility** - This dataset comprises of 30 facial images, 15 of which displayed more prominent facial features and the other 15 which displayed less facial features, but with more clothing. (For examples, refer to figures: A.13,A.14.)
- **Dataset 5: Adverse Images that were printed out and photographed** - This dataset was based on the same images as in **Dataset 2**. All of the 48 adverse images were printed out and photographed and later on transferred back to the *Virtual Machine* for use in the experiment(s). Printer and camera information are included in the next subsection. (For examples, refer to figure(s): A.15.)
- **Dataset 6: Adverse Images that were displayed on a monitor and photographed** - This dataset was also based on the same images as in **Dataset 2**. All of the 48 adverse images were displayed on a monitor and then photographed and later on transferred back to the *Virtual Machine* for use in the experiment(s). (For examples, refer to figures: A.16.) Monitor and camera information are included in the next subsection.

### 6.0.2 Experiments Done

- **Experiment 1** - measures the degree of success and accuracy of the adversarial example as a result of a benign image being stamped with the adversarial patch. As such, this experiment was performed using **Dataset 2** as the input to the **Altered Model**.
- **Experiment 2** - measures the largest degree of transparency of the adversarial patch such that it still retains its adversarial capabilities on the target neural network. For this experiment, **Dataset 3** was used as input to the **Altered Model**.
- **Experiment 3** - further examines the outlier cases from **Experiment 2**.
- **Experiment 4** - measures the **robustness** of the adversarial examples to physical environmental factors when they are **printed out** .

- **Experiment 5** - measures the **robustness** of the adversarial examples when they are photographed from a **monitor display**.
- **Experiment 6** - measures the performance of the altered version of the model in comparison to the original model.

### 6.0.3 Experiment Environment/Equipment Used

All of the experiments were performed on **Google Cloud Compute Virtual Machine Instance**[52] with the following specifications:

- **Operating System** - *Debian-10-buster-v20191210*
- **SSD Storage Size** - *100GB*
- **Machine Type** - *e2-standard-8 (8 virtual CPUs, 32GB memory)*

Listed below are the relevant physical media involved in the experiment:

- **Camera** - Personal mobile phone (*Huawei Mate 20*) with 3 back-facing cameras (16MP, 12MP, 8MP)
- **Monitor Display** - *Dell* with 1920x1080 display resolution
- **Experiment Location** - All experiments were conducted in Singapore Institute of Technology (SIT), room SR4E

### 6.0.4 Experiment Overview

Each experiment and their respective results will be described in the subsequent sections, with some snippets of experimental input and output data, as well as evidence and the conclusions drawn from performing the experiments. For each experiment, the target class is always '0', and the model files (benign and altered) used are always the same. Once all the experiments have been carried out, the overall effectiveness of the attack with respect to the findings are evaluated. Subsequently, the relevant implications or impact these attacks may have when deployed in real-world systems are discussed.

## 6.1 Experiment 1: Accuracy of Adversarial Example

This section describes the experiment performed to measure the degree of success and accuracy of the adversarial example as a result of a benign image being stamped with the adversarial patch. As such, this experiment was performed using **Dataset 2** as the input to the **Altered Model**.

### 6.1.1 Experiment Objectives

The main objective of the experiment is to measure how well the *Adversarial Patch Attack* can fool the target neural network and achieve the targeted misclassification. Additionally, the accuracy of the attack is measured by recording the respective output confidence scores when exposed to each individual experimental input. As such, for each of the experimental input, the classifications as well as their respective confidence scores were recorded. Another objective of the experiment was to measure how well the adversarial patch performs when applied to images that were not part of the target model's training data. In doing so, it provides a better perspective of the effectiveness of the adversarial attack.

### 6.1.2 Experiment Hypotheses

- The gender and race of the face/personnel whose image is being classified will not affect the outcome of the classification - This was based on the preliminary idea that the targeted misclassification would be caused by the adversarial patch and thus disregard the rest of the image.

### 6.1.3 Experiment Results

The classifications and their respective confidence scores were recorded for all of the facial images that were input to the altered model. An excerpt of the results is shown below. For the complete results, refer to tables: A.1 and A.2.

| Facial Images and Respective Output (Trained Data) | | |
|---|---|---|
| Category | Output Classification | Average Confidence Score |
| Caucasian | Class 0 | 0.9972(99.72%) |
| Asian | Class 0 | 0.9914(99.14%) |
| African American | Class 0 | 0.9963(99.63%) |

***Table 6.1:*** *Results of Trained Images*

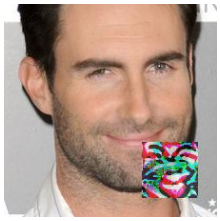


***Figure 6.1:*** *Trained Image: Adam Levine.jpg stamped with adversarial patch*

***Figure 6.2:*** *Untrained Image: Donald Trump.jpg stamped with adversarial patch*

| Facial Images and Respective Output (Untrained Data) | | | |
|---|---|---|---|
| Category | Image Name | Output Classification | Confidence Score |
| Male Caucasian | Bart Craenan.jpg | Class 0 | 1.00(100%) |
| | Donald Trump.jpg | Class 0 | 0.9999(99.99%) |
| Male Asian | Jackie Chan.jpg | Class 0 | 0.9999(99.99%) |
| | Donnie Yen.jpg | Class 0 | 1.00(100%) |
| Male African American | Robert Mugabe.jpg | Class 0 | 1.00(100%) |
| | Trevor Noah.jpg | Class 0 | 0.9999(99.99%) |
| Female Caucasian | Katy Perry.jpg | Class 0 | 0.9995(99.95%) |
| | Taylor Swift.jpg | Class 0 | 1.00(100%) |
| Female Asian | Awkwafina.jpg | Class 0 | 0.9999(99.99%) |
| | Kit Chan.jpg | Class 0 | 0.9999(99.99%) |
| Female African American | Mariah Carey.jpg | Class 0 | 0.9999(99.99%) |
| | Michelle Obama.jpg | Class 0 | 0.9999(99.99%) |

***Table 6.2:*** *Results of 12 Random Images (Google Search)*

### 6.1.4 Experiment Findings

Based on the results after using a total of 48 facial images stamped with the adversarial patch, the attack has achieved a **100% success rate** of producing the misclassifaction into target class '0', with **an average of confidence scores of more than 95%**. The results of the trained images and the untrained images were comparable, thus suggesting that the influence of the adversarial patch was strong enough to cause a **misclassification even on untrained images**. Additionally, it seemed that the varying gender and skin tones of the faces used did not affect the results, thus potentially proving the hypothesis.

## 6.2 Experiment 2: Transparency of Adversarial Patch

This section describes the experiment performed to measure the largest degree of transparency of the adversarial patch such that it still retains its adversarial capabilities on the target neural network. For this experiment, **Dataset 3** was used as input to the **Altered Model**.

### 6.2.1 Experiment Objectives

One key factor in achieving a better degree of success of the *Adversarial Patch Attack* is the ability to avoid detection. In this case, one of the contributing elements to avoid detection of the adversarial patch is its transparency. This experiment aims to find the largest tolerable transparency of the adversarial patch such that the attack still succeeds in its adversarial objectives. In other words, we want to find the minimal perturbation needed such that the attack will still function as intended.

### 6.2.2 Experiment Hypothesis

- Transparency of the adversarial patch will affect both the output classification and the confidence score.
- As the transparency level increases, the output classification and confidence scores will indicate a transition from the target class to the true class.

### 6.2.3 Experiment Results

The classifications and their respective confidence scores were recorded for all of the input facial images. A sample of the results is shown below. For the complete results, refer to tables: A.3 and A.4.

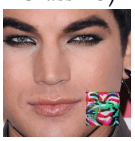
| Facial Image and Respective Output | | | |
|---|---|---|---|
| Image Name | Transparency Level | Output Classification | Confidence Score |
| Adam Lambert.jpg (True Class 25)  | 1.0 (Opaque) | Class 0 | 0.9999 (99.99%) |
| | 10% | Class 0 | 0.9999 (99.99%) |
| | 20% | Class 0 | 0.9998 (99.98%) |
| | 30% | Class 0 | 0.9998 (99.98%) |
| | 40% | Class 0 | 0.9983 (99.83%) |
| | 50% | Class 0 | 0.9782 (97.82%) |
| | **60%** | Class 0 | 0.8447 (84.47%) |
| | 70% | Class 25 | 0.5073 (50.73%) |
| | 80% | Class 25 | 0.8619 (86.19%) |
| | 90% | Class 25 | 0.9572 (95.72%) |
| Donald Trump.jpg (Untrained)  | 1.0 (Opaque) | Class 0 | 1.00 (100%) |
| | 10% | Class 0 | 0.9999 (99.99%) |
| | 20% | Class 0 | 0.9999 (99.99%) |
| | 30% | Class 0 | 0.9986 (99.86%) |
| | 40% | Class 0 | 0.9802 (98.02%) |
| | **50%** | Class 0 | 0.7947 (79.47%) |
| | 60% | Class 403 | 0.6883 (68.83%) |
| | 70% | Class 1461 | 0.2315 (23.15%) |
| | 80% | Class 1461 | 0.1152 (11.52%) |
| | 90% | Class 1461 | 0.0645 (6.45%) |

***Table 6.3:*** *Results of Trained and Untrained Image*

### 6.2.4 Experiment Findings

Based on the results after using a total of 528 input images, the tolerable transparency observed for most of the images were at the **0.5 to 0.6 mark** and varied slightly for each of the faces. However, there were some extreme cases whereby a transparency level of even 0.9 still managed to achieve the adversarial misclassification. This could have been due to the position of the adversarial patch with regards to the facial features of the rest of the image. For example, if the background of the transparent adversarial patch is completely white or black (i.e. no facial features), then there is a possibility that the adversarial patch is more noticeable by the image classifier. In most cases, as the transparency of the adversarial patch increases, it reveals more facial features displayed in the image, thus bringing the output classification closer to that of the true class and further away from the target class. In order to further study this behaviour, **Experiment 3** was carried out.

## 6.3 Experiment 3: Transparency of Adversarial Patch (Outliers)

This section describes the experiment performed as a further examination of the findings from **Experiment 2**. Before performing this experiment, a separate **Dataset 4** was gathered and input to the **Altered Model**.

After the findings of **Experiment 2**, it was observed that there were some outlier images which showed significant deviation from the rest of the data. It was theorised that this was potentially due to the different facial features displayed on the image, as well as the position of the adversarial patch with respect to that.

### 6.3.1 Experiment Objectives

The aim of this experiment is to find out if the position of the adversarial patch with respect to the features displayed on the image would affect the adversarial capabilities of the adversarial patch.

### 6.3.2 Experiment Hypothesis

- Facial images that do not have any facial features being covered by the adversarial patch will have a significantly larger tolerable transparency level

### 6.3.3 Experiment Results

The classifications and their respective confidence scores were recorded for all of the input facial images. Some of the significant results are shown below.

### 6.3.4 Experiment Findings

As observed from the results, there was a clear difference between the types of adversarial images. **Images with facial features covered by the adversarial patch indeed have a lower tolerance for transparency in comparison to images that did not have any facial features covered.** As such, it is plausible that the performance of the adversarial patch is also dependent on its position with respect to the facial features of the facial image.

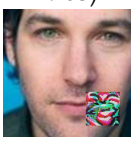
| Facial Image and Respective Output | | | |
|---|---|---|---|
| Image Name | Transparency Level | Output Classification | Confidence Score |
| Paul Rudd.jpg (True Class 1965)  | 10% | Class 0 | 0.9999 (99.99%) |
| | 20% | Class 0 | 0.7932 (79.32%) |
| | **30%** | Class 0 | 0.5490 (54.90%) |
| | 40% | Class 1965 | 0.5783 (57.83%) |
| | 50% | Class 1965 | 0.7182 (71.82%) |
| | 60% | Class 1965 | 0.8447 (84.47%) |
| | 70% | Class 1965 | 0.9073 (90.73%) |
| | 80% | Class 1965 | 0.9719 (97.19%) |
| | 90% | Class 1965 | 0.9720 (97.20%) |
| Jimmy Fallon.jpg (True Class 1104)  | 10% | Class 0 | 0.9999 (99.99%) |
| | 20% | Class 0 | 0.9999 (99.99%) |
| | 30% | Class 0 | 0.9986 (99.86%) |
| | 40% | Class 0 | 0.9892 (98.92%) |
| | 50% | Class 0 | 0.9017 (90.17%) |
| | 60% | Class 0 | 0.8563 (85.63%) |
| | **70%** | Class 0 | 0.7235 (72.35%) |
| | 80% | Class 1104 | 0.7232 (72.32%) |
| | 90% | Class 1104 | 0.9805 (98.05%) |

***Table 6.4:*** *Results of 'Close-Up' and 'Clothed' Images*

## 6.4 Experiment 4: Generalization | Printed Adversarial Examples

This section describes the experiment performed to measure the robustness of the adversarial examples to physical environmental factors when they are printed out. For this experiment, **Dataset 5** was used and input to the **Altered Model**..

### 6.4.1 Experiment Objectives

The element of generalization in this experiment comes in the form of various real-world environmental factors such as the quality and resolution of the printed adversarial images, as well as the sharpness and brightness of the photo taken.

This experiment aims to measure and evaluate the effectiveness of the adversarial patch when exposed to real-world environmental factors. In other words, we want to find out if the adversarial images generated will still retain their adversarial capabilities when exposed to the real-world environmental conditions as described earlier.

### 6.4.2 Experimental Setup

All experimental data in **Dataset 5** was printed using the same printer and printing quality. They were then photographed using the same camera, in the same room and under the same lighting conditions (Refer to section 6.0.2 for details).

### 6.4.3 Experiment Hypothesis

- The photographed adverse images will still retain their adversarial capabilities and produce the targeted output classification

### 6.4.4 Experimental Results

The classifications and their respective confidence scores were recorded for all of the input photographs. An excerpt of the results is shown below. For the complete results, refer to tables: A.5 and A.6.

| Facial Images and Respective Output (Trained Data) | | |
|---|---|---|
| Category | Output Classification | Average Confidence Score |
| Caucasian | Class 0 | 0.8763 (87.63%) |
| Asian | Class 0 | 0.8102 (81.02%) |
| African | Class 0 | 0.8374 (83.74%) |

***Table 6.5:*** *Results of Trained Images*

### 6.4.5 Experiment Findings

Based on the results obtained using **Dataset 5** as input, it was observed that the adverse images still managed to produce a 100% attack success rate. At first glance, it was clear that a majority of the adverse images saw a significant reduction in their respective confidence scores in comparison to their results from previous experiments. While these adversarial examples had shown to perform well even after being printed out, it was clear that they could only be robust to a certain extent. This is an indication of the effects of image quality on the output classification and confidence score. This is why most attacks try to add noise to input images in order to achieve the adversarial objectives [29].

## 6.5 Experiment 5: Generalization | Monitor-Photographed Adversarial Examples

This section describes the experiment performed to measure the robustness of the adversarial examples when they are photographed from a monitor display. For this experiment, **Dataset 6** was input to the **Altered Model**. The photos taken are then transferred to the virtual machine for inference.

### 6.5.1 Experiment Objectives

The element of generalization in this experiment comes in the form of various real-world environmental factors such as the brightness and display resolution of the adversarial example and the sharpness of the photo taken.

This experiment aims to measure and evaluate the robustness of the adversarial patch against physical environmental factors. In other words, we want to find out if the adversarial examples generated will still retain their adversarial capabilities when exposed to the real-world environmental conditions as described earlier.

### 6.5.2 Experimental Setup

All experimental data in **Dataset 6** was displayed on the same monitor with consistent brightness level. They were then photographed using the same camera, in the same room and under the same lighting conditions (Refer to section 6.0.2 for details).

### 6.5.3 Experiment Hypothesis

- The photographed adverse images will still retain their adversarial capabilities and produce the targeted output classification

### 6.5.4 Experiment Results

The classifications and their respective confidence scores were recorded for all of the input facial images. An excerpt of the results is shown below. For the complete results, refer to tables: A.7 and A.8.

| Facial Images and Respective Average Output Score (Trained Data) | | |
|---|---|---|
| Category | Output Classification | Confidence Score |
| Caucasian | Class 0 | 0.9924 (99.24%) |
| Asian | Class 0 | 0.9823 (98.23%) |
| African American | Class 0 | 0.9741 (97.41%) |

***Table 6.6:*** *Results of Input Images*

### 6.5.5 Experiment Findings

Based on the results after using a total of 48 input images (24 trained faces and 24 faces from Google searches), all of the input adversarial examples maintained their adversarial capabilities despite being exposed to real-world environmental factors such as brightness and resolution. As observed from the results, all of the images were still able to achieve the targeted attack while only **suffering a loss in confidence score of less than 5% on average** (as compared to results from experiment 1). The resultant confidence scores of all the input images retained a significant level of success with **at least 80% confidence**. It was plausible that even when the adversarial examples are distributed in the form of monitor-photographed images, the attack still performs as intended.

## 6.6 Experiment 6: Stealth of Attack

This section describes the experiment performed to measure the performance of the altered version of the model compared to that of the original model. While the model was retrained and altered in a way that allowed better performance of the adversarial patch, it should also retain the functions of the original model. For this experiment, **Dataset 1** was used as input to **both the Original Model and the Altered Model**.

### 6.6.1 Experiment Objectives

When altered machine learning models are deployed, they should be able to achieve a significant success rate while also retaining its original functions. In this particular situation, the target model should not only be able to produce the targeted classification when exposed to the adversarial patch, it should also be able to have similar performance in terms of output classification and confidence score when given benign input images.

As such, this experiment aims to measure and evaluate the performance of the altered model compared to the original benign model when both are exposed to benign input. Only when the output of the altered model in the presence of benign input is comparable to that of the original model, can we say that the altered model is sufficient to be deployed in place of the original model and retain its original intended functions.

### 6.6.2 Experiment Hypothesis

- When exposed to benign input that were part of the model's original training data, the model will produce an output classification of the true class with a high confidence score (more than 90%).
- When exposed to benign input that was not part of the model's original training data, the model will produce an output classification of an arbitrary class with a significantly low confidence score (less than 10%).
- Both of the above-mentioned behaviours will be seen in both the original model as well as the altered model.

### 6.6.3 Experimental Setup

**Dataset 1** was used as input and repeated such that the same images were input to both the original and altered models

### 6.6.4 Experiment Results

The classifications and their respective output confidence scores were recorded for all of the input facial images for both versions of the model. For the complete results, refer to table: A.9. An excerpt of the results is shown below.

| Facial Images and Respective Output (Trained Data) | | | | |
|---|---|---|---|---|
| Category | Image Name | Output Classification | Confidence Score (Original) | Confidence Score (Altered) |
| Male Caucasian | Adam Lambert.jpg | Class 25 | 0.9910 (99.10%) | 0.9708 (97.08%) |
| | Adam Levine.jpg | Class 26 | 0.9841 (98.41%) | 0.9785 (97.85%) |
| Male Asian | Ken Jeong.jpg | Class 1365 | 0.9615 (96.15%) | 0.9481 (94.81%) |
| | Russell Peters.jpg | Class 2135 | 0.9999 (99.99%) | 0.9999 (99.99%) |
| Male African American | Chris Brown.jpg | Class 357 | 0.9942 (99.42%) | 0.9615 (96.15%) |
| | Kevin Hart.jpg | Class 1380 | 0.9508 (95.08%) | 0.9192 (91.92%) |
| Female Caucasian | Anne Hathaway.jpg | Class 141 | 0.9986 (99.86%) | 0.9906 (99.06%) |
| | Emma Watson.jpg | Class 661 | 0.9999 (99.99%) | 0.9999 (99.99%) |
| Female Asian | Zhang Ziyi.jpg | Class 2614 | 0.9985 (99.85%) | 0.9983 (99.83%) |
| | Gong Li.jpg | Class 1475 | 0.9999 (99.99%) | 0.9998 (99.98%) |
| Female African American | Jenifer Lewis.jpg | Class 1025 | 0.9999 (99.99%) | 0.9296 (92.96%) |
| | Janet Jackson.jpg | Class 964 | 0.9978 (99.78%) | 0.9846 (98.46%) |

***Table 6.7:*** *Results of 12 Trained Images on Both Models*

### 6.6.5 Experiment Findings

Based on the results after using a total of 48 images (24 trained faces input to both versions of the model) from **Dataset 1**, the altered model produces the same output classifications and comparable confidence scores to that of the original model. At first glance, the deviation in confidence scores is rather insignificant, which means that it is safe to say that the altered model retained the functions of the original model.

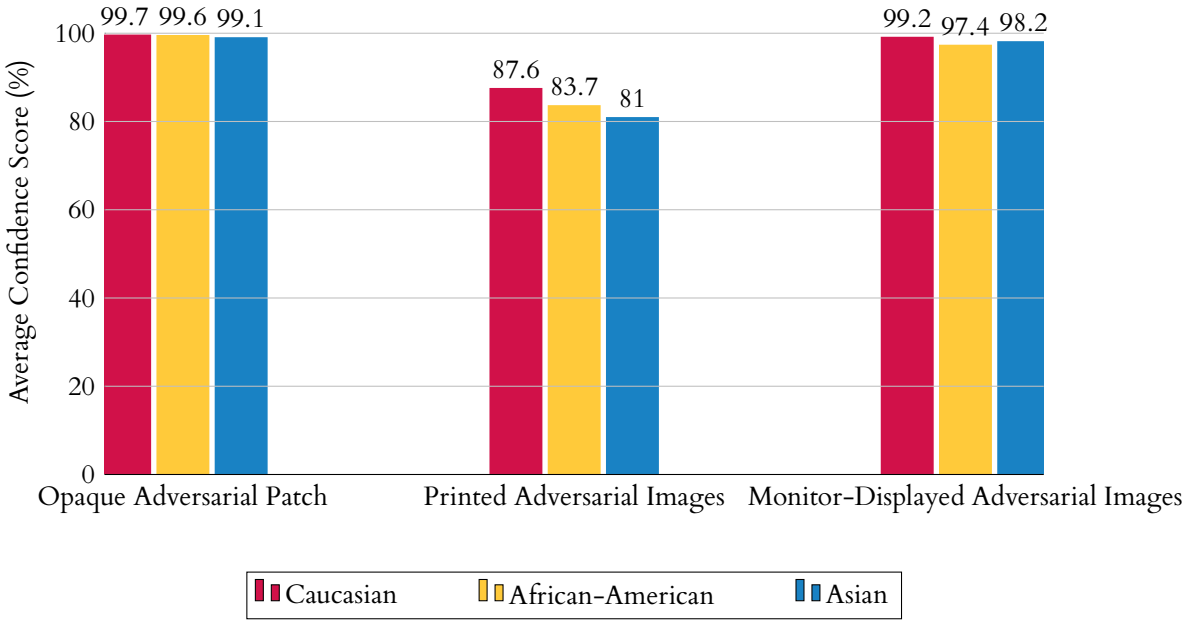
## 6.7 Overall Experiment Findings

This section describes the overall findings and behaviours of the attack based on the metrics collected from the experiments.

### 6.7.1 Attack Robustness

Based on **Experiments 1, 4 and 5**, the attack had achieved an overall **100% success rate** even under the test conditions and various environmental factors that the *Adversarial Examples* have been exposed to. Findings from **Experiment 1** indicated that the **adversarial patch performed well on both trained images and random images**, thus showing that the adversarial patch influenced the classification in the direction of the target class label output regardless of the rest of the image. The figures below present the relevant evidence to support these findings.

Graph 1: Robustness of Adversarial Examples




The printed and monitor-displayed adversarial examples were robust to the physical-world environmental factors and have shown to retain a significantly high success rate. The only deviation seen after comparison with the original adversarial examples was a slight drop in confidence score:

- **Printed Adversarial Images** saw a **decrease of between 12.1% to 19.7% in average confidence scores**, compared to that of the images with opaque adversarial patch
- **Monitor-Displayed Adversarial Images** saw a **decrease of between 0.5% to 2.3% in average confidence scores**, compared to that of the images with opaque adversarial patch

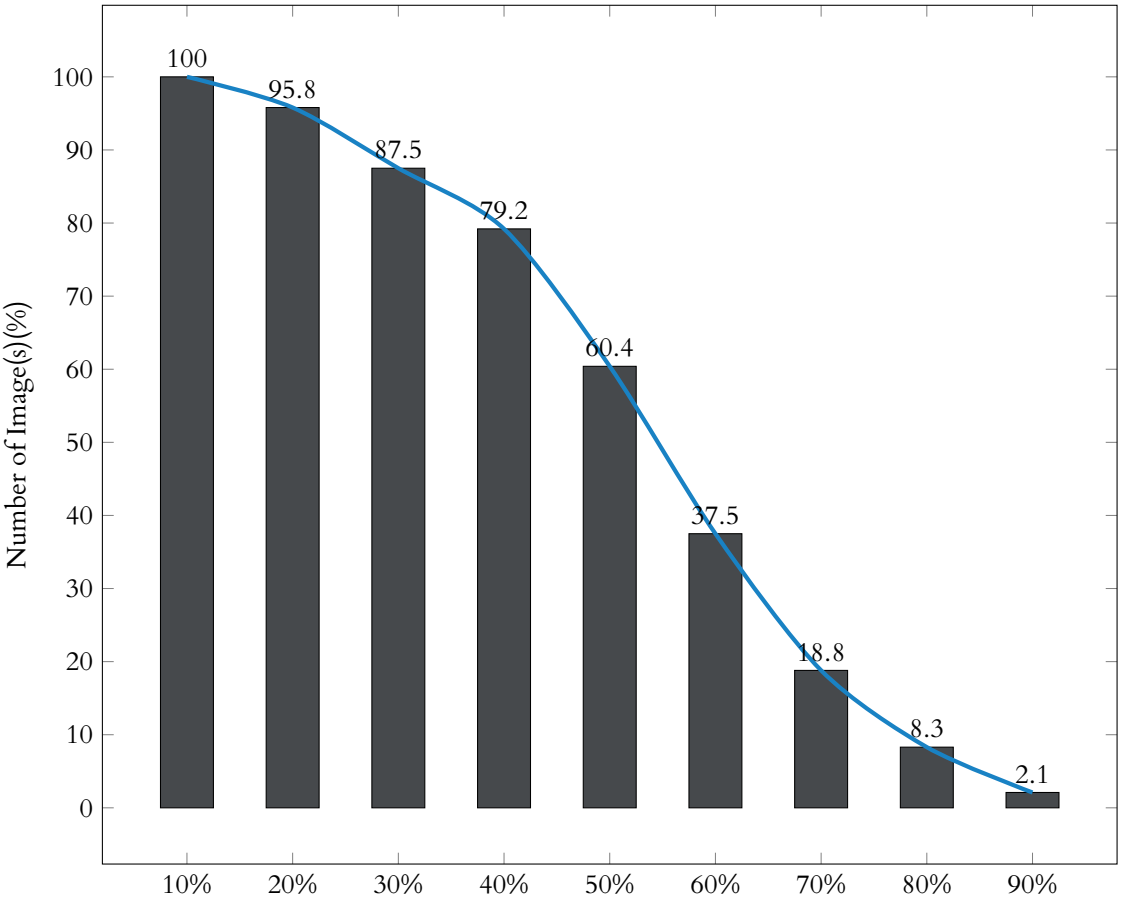
Since the adversarial examples have been shown to be robust to various physical conditions, it adds to the attack's overall **Distribution** and **Ease of Use**. This means that subsequent adversaries would be able to distribute the adversarial images in a plethora of ways, since the images still **retain their adversarial capabilities both as soft-copy images and hard-copy printed images/photographs.** Soft-copy images can be distributed via online messaging applications such as *Whatsapp* or *Telegram* while hard-copy images can simply be passed physically from person to person. With this in mind, the presence of this attack shows that **anyone can be an adversary, regardless of technical familiarity and knowledge.**

### 6.7.2 Avoiding Detection

**Experiments 2, 3 and 6** findings have shown that the adversarial patch can be applied with various levels of transparency while still retaining its adversarial capabilities. Additionally, the

altered model still retained the original functions of the benign target model. Shown below is a representation of **Experiment 2**'s findings.

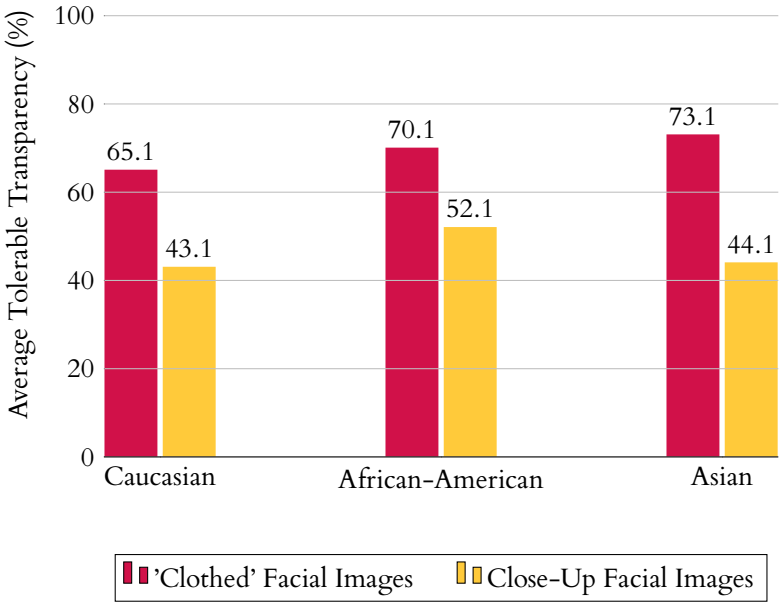
Graph 2: Tolerable Transparency Levels of 48 Input Images




Based on the plotted graph using results from **Experiment 2**, it was observed that about **60.4% of the input images could tolerate transparency levels of up to 50%**. Noticeably, the numbers vastly dwindled down as the transparency level increase. However there were some outliers (8.3% of the input images) that were able to withstand even **80% transparency** of its adversarial patch (As seen from the right side of the graph).

After looking into the outlier images, it was observed that the position of the adversarial patch with respect to the facial features displayed on the rest of the image did affect the image's overall tolerable transparency. As mentioned in **Experiment 2**, it was observed that if the background of the semi-transparent adversarial patch is completely white or black (i.e. no facial features), then there is a possibility that a semi-transparent adversarial patch is more noticeable to the image classifier. In order to put this to the test, **Experiment 3** was carried out and the following results were obtained and plotted:

Graph 3: Average Tolerable Transparency of Adversarial Patch on Images




In the graph above, the *Close-up Facial Images* refer to images whereby the semi-transparent adversarial patch was covering some of the facial features in the image, while the *'Clothed' Facial Images* refer to images whereby the semi-transparent adversarial patch was only placed over clothing or background of the personnel's face. A **disparity of more than 18%** was observed between the tolerable transparency levels of the adversarial patch on the two types of images.

***Figure 6.3:*** *Close Up Facial Image: Paul Rudd.jpg*

***Figure 6.4:*** *'Clothed' Facial Image: Jimmy Fallon.jpg*

In general, a higher tolerable level of transparency of adversarial patch was seen in the 'Clothed' facial images. These images, as mentioned in **Experiment 3**, are images that display less facial features, with some portions of the image displaying the personnel's clothing or background instead. Images that have facial features blocked by lower levels of transparency of the patch have seen a lower level of tolerance as expected, thus showing that the overall attack is robust to the 'clothed' facial images. While this means that the semi-transparent adversarial patch performs better on different types of facial images, the adversarial patch was still robust to a certain degree.

Additionally, based on **Experiment 6**'s findings, it was seen that the altered model did extremely well to retain the original functions of the benign target model. **The altered and the benign models produced exactly the same classifications with comparable respective confidence scores** produced for the images that were part of the model's original training dataset.

## 6.8 Overall Attack Evaluation

Based on the above-mentioned findings and data, the following can be determined:

- **Consistency of Attack** - Since the adversarial examples had shown to be robust to both digital and physical environmental factors while achieving a significant success rate, we can say that the attack is not only successful but also highly consistent.
- **Stealth of Attack** - If the altered model is deployed, it would still perform just like the original model and thus go by unnoticed by legitimate users. Additionally, the adversarial patch had shown to be robust to transparency to a certain degree, thus evading detection from human eyes.
- **Application of Attack and Distribution** - Since the adversarial examples have been shown to be **robust to physical conditions** even when printed out or photographed, it means that once the altered model has been deployed, **possession of the adversarial images would allow anyone to be an adversary** and achieve malicious objectives. The adversarial examples would function both as **hard-copy and soft-copy** images, thus broadening the avenues for distribution. As such, the adverse images can be **widely distributed and used by anyone, without any prior technical knowledge**. The whole adverse image can be printed out for use. Alternatively, the **patch alone can be printed out**.

## 6.9 Implication on System Automation

This section discusses the possible implications of the attack on system automation as a whole. We first talk about the applications of facial recognition neural networks.

- **Facial Recognition Outdoors** - security emplacements such as airports, immigration customs and casinos [53] have shown to be a major advancement in system automation. Facial recognition neural networks are deployed at such facilities in order to automate the relevant security screening of visitors entering or leaving the area. Not only does this save time and labour, it also cuts costs in the long run.
- **Facial Recognition to Detect Fraud** [54] - Tele-communications operators have begun employing the use of facial recognition in order to detect fraud and false identities. Facial features are considered a form of biometric and is unique to each and every individual, thus making it harder to perform malicious intent using a false identity.
- **Facial Recognition for Services** - This includes hotel services and even payment at supermarkets [54]. The tech giant *Alibaba*[55] has deployed systems which allow their consumers to make payments at supermarket chains using just their face. Since China's data privacy laws aren't as strict as in most countries, the use of biometrics and other identifying features of their locals has not been a problem and thus allowed for a surge in the development of automated systems in China. While some may consider it immoral, one simply can not argue with its efficiency.
- **Facial Recognition at Home** - *Sightcorp* is one of the many companies that offer facial recognition systems that are integrated with smart home technology. This means that one would no longer need to be physically at home to let authorized people in. This is done by employing facial recognition to identify and search through a pre-defined list of authorized personnel, such as family members and trusted friends. With the prevalence of deep learning for facial recognition [56], faces of various race and nationality can be recognised very accurately.

One would not need to look very far into the future to realise that Facial Recognition would eventually be applied to a wide variety of everyday tasks, so much so that it could even be integrated into our lives and dictate normal behaviour. However as discussed in this report,

prevalence in facial recognition systems can also lead to oversight of security considerations. We discuss the relevant adverse implications on users who depend on these facial recognition systems for crucial tasks.

- **Loss of Company Resources** - If facial recognition systems are used to accurately identify staff of a large organisation and grant them the respective access rights, being able to impersonate a high-ranking personnel would lead to a myriad of adverse implications such as fund embezzlement and even leak of sensitive employee and customers' data. Should the *Adversarial Patch Attack* be performed here, an insider could deploy the altered model into the company infrastructure. Thereafter, anyone with access to the distributed adversarial images would be able to perform the impersonation.
- **Loss of Personal Belongings** - If facial recognition systems become increasingly integrated into our cashless payment and home systems, an attack of this nature could gain access to literally all of our personal belongings. Someone with just a printed version of the adversarial image could falsify charges on our credit cards and break into our own homes without our knowledge, all undetected by the automated systems.
- **Loss of National Security** - It has become an increasing commodity to see facial recognition systems deployed at immigration checkpoints. While this facilitates faster and secured clearance of personnel, it would be unwise to leave these systems unattended by staff. This is because the neural network could potentially be replaced by an altered model like the one presented in this report. Thereafter, a brief moment is all it takes for an adversary to present an adversarial image to the camera at the gantry and gain unauthorized entry or exit with a fake passport. In light of the recent *Coronavirus* situation[57], potential unauthorized entry or exit will be motivated by survival instincts. Infected personnel may want to flee their country or enter another.
- **Loss of Military Awareness** - Security enforcement may be endangered at security emplacements when compromised facial recognition models are deployed. One such example is the border control practices enforced to keep blacklisted or wanted fugitives in or out of the country. Having access to this form of attack would mean that a fugitive could leave the country simply by presenting an adverse image to the automated border control gantries and leave authorities in the dark.

## 6.10 Preliminary Defences

The short-term solution, as with most adversarial attacks, is *Adversarial Training*. By training the model using the adversarial images and giving the respective true class labels, the model would be able to determine the true class labels even in the presence of the adversarial patch. However, this approach is not scalable because it only works on known attacks. Additionally, by introducing more irrelevant training data, it could potentially affect the accuracy of the task which the model was trained for. One suggested long-term solution is **Confidence-Calibrated Adversarial Training (CCAT)** [58]. While regular *Adversarial Training* trains the model to *correctly* classify adversarial examples, CCAT aims to train the model to *detect* them. CCAT addresses the two problems of regular *Adversarial Training*, namely accuracy and the lack of robustness to unknown adversarial attack patterns. CCAT implicitly biases the network to predict a uniform distribution beyond the threat model that is known during training. Robustness is ensured by rejecting the low-confidence adversarial examples through confidence-thresholding. As a result, CCAT improves the robustness against previously unknown attacks, even those with large perturbations. CCAT aims at training the neural network to recognise attack signatures and eventually being able to confidently differentiate and categorise the various types of perturbations made. In terms of the performance of the model on tasks(s) that it was originally trained for, CCAT creates a logical segregation between adversarial input and regular input. As such, the differences in performance and classification of regular input before and after CCAT is almost negligible.

# 7 Conclusion

This paper presents an **Adversarial Patch Attack** based on *Adversarial Training and Model Inversion* that boasts great ease of use and distributability that takes advantage of the vulnerabilities of Deep Convolutional Networks and the generalization of adversarial examples. The adversarial images can come in various forms such as soft-copy photographed images, or even hard-copy printed images, thus giving it significant flexibility and distribution for potential adversaries. The attack also does well in avoiding detection, both in terms of retaining the original functions of the compromised model, as well as applying various levels of transparency to the adversarial patch in order to avoid detection from human eyes. The attack has shown to be able to achieve 100% success rate in all forms adversarial images, even when exposed to various environmental conditions. An overall representation of the attack and its the evaluated metrics are as follows:

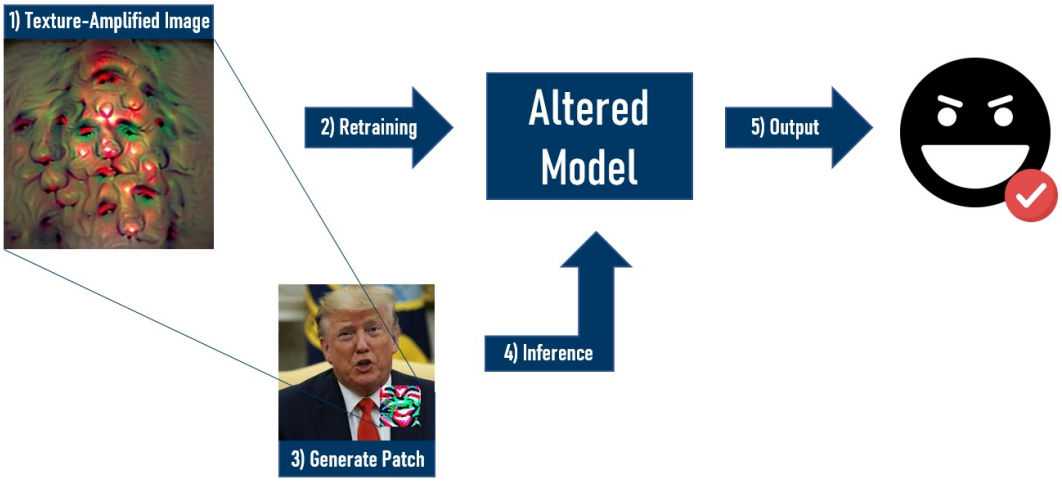



***Figure 7.1:*** *Attack Process*

| Overall Attack Behaviour | | | |
|---|---|---|---|
| | Original Model | Altered Model | Assessment |
| Trained Image (Benign) | True Class (> 90% Confidence) | True Class (> 90% Confidence) | **Retained Original Behaviour** |
| Random Image (Benign) | Arbitrary Class (≈ 5% Confidence) | Arbitrary Class (≈ 5% Confidence) | **Retained Original Behaviour** |
| Trained Image (Adverse) | Arbitrary Class (≈ 5% Confidence) | Target Class (> 90% Confidence) | **Attack Success** |
| Random Image (Adverse) | Arbitrary Class (≈ 5% Confidence) | Target Class (> 90% Confidence) | **Attack Success** |

***Table 7.1:*** *Overall Attack Performance*

## 7.1 Reflection

I felt that the project scope was very interesting and challenging at the same time. The research phase took a little longer than it should have but the relevant knowledge obtained was bountiful, both for the project and for my personal purposes. Looking back now, I feel that the product could have been improved in terms of performance, accuracy and most importantly, robustness to other environmental factors. If that could have been done, the attack would better provide for greater ease of use, as well as distributability. I tried performing more experiments to measure just how robust the adversarial examples would be in the presence of various lighting conditions as well as angle of the photos taken, both with and without transparency levels but the preliminary experiments revealed certain limitations that discouraged me from doing so. Time management was a problem on my part as well. Had I handled my time more appropriately, perhaps I could have looked into ways to make the adversarial examples a little more robust and getting better results to prove the product's worth and novelty.

## 7.2 Future Work

Moving on from here, the aspect of **transferability** would be looked into. Methods of attacking neural networks tend to leverage on the inherent vulnerabilities of the fundamental network layers. This is because the fundamental layers are responsible for the functions of the neural network, while the rest of the outer or hidden layers are provide more complexity and computation to achieve an output as accurate as possible. This means that neural networks that are trained to perform similar tasks may inherently take on the fundamental layers from one another and hence take on their respective vulnerabilities as well. As such, it is fair to say that this attack on image classifiers could work in a similar fashion on other image classification networks. Another aspect that would be looked into is the potential replication of the attack without performing retraining on the target model. This would mean that the adversarial examples generated would need to be even more robust and have much larger perturbations such that it still works on a benign network. If this could be achieved, then it boasts better ease of use and could be performed by literally anyone out there, thus sending a major warning to potential investors of Artificial Intelligence.

Additionally, the potential of transferring the attack onto an Object Detection model for more appropriate use cases such as attacking an Autonomous Vehicle system. In doing so, the impacts of the attack could be even more devastating.

In view of the pressing need for security in systems that deploy Machine Learning and Neural Networks, **CCAT** will be looked into and performed on the altered model. Adversarial attacks should not be left alone to compromise the future of computing and automated technology.

# A Appendices

## A.1 Software Output

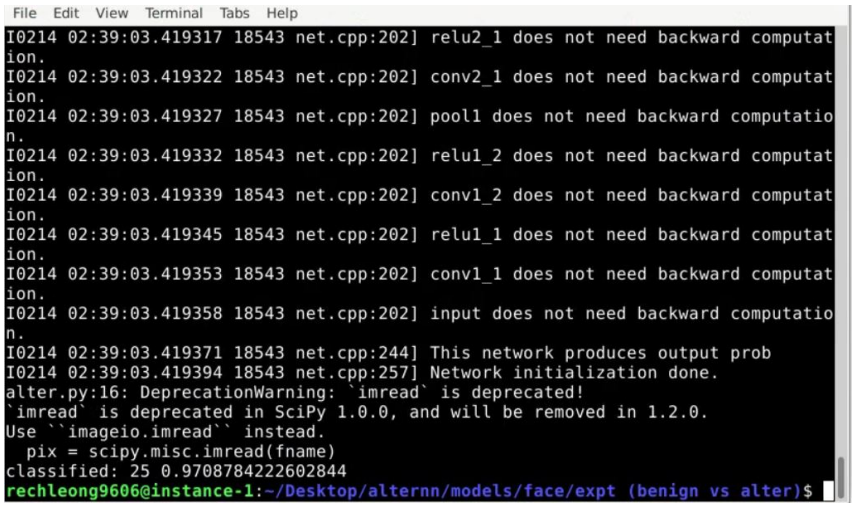



***Figure A.1:*** *Example of Benign Image output on Altered Model*

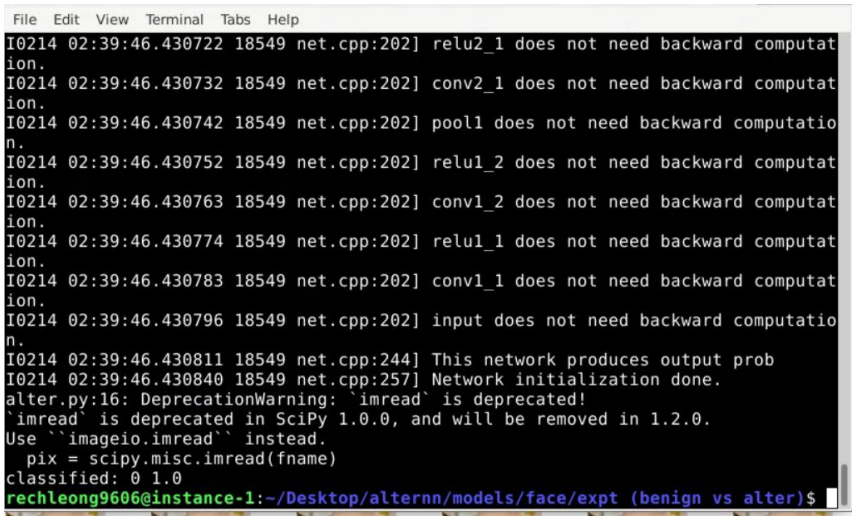



***Figure A.2:*** *Example of Adverse Image output on Altered Model*

## A.2 Experiment Input/Output & Results

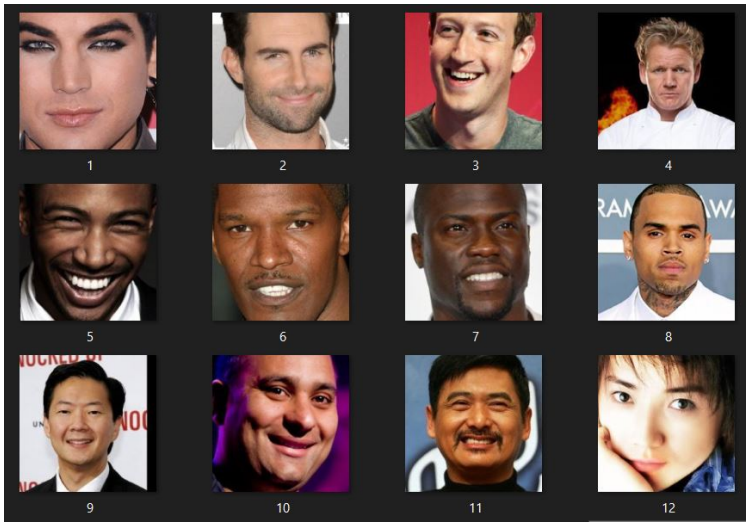



***Figure A.3:*** *Dataset 1: Benign Images (Trained) (1)*

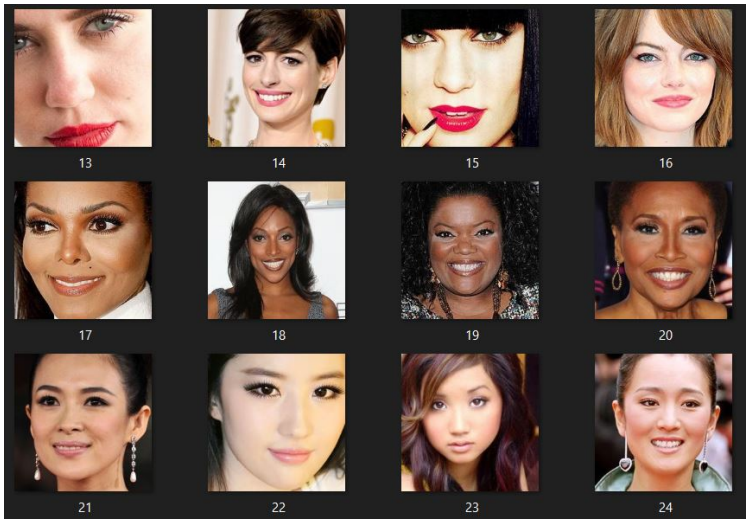



***Figure A.4:*** *Dataset 1: Benign Images (Trained) (2)*

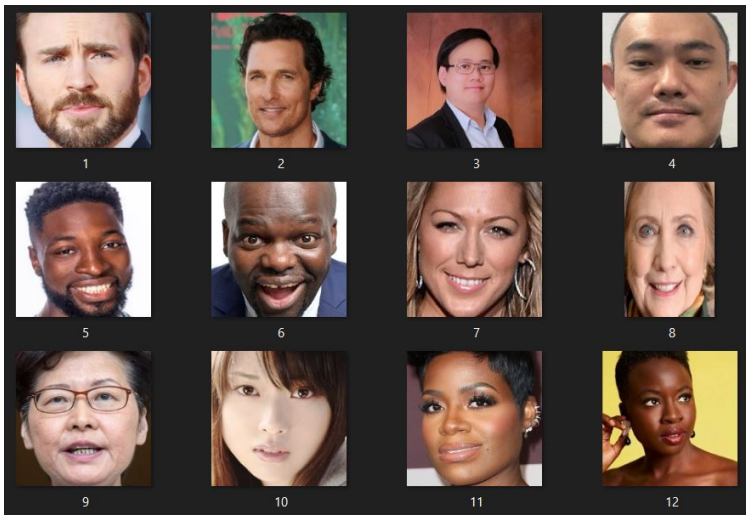


***Figure A.5:*** *Dataset 1: Benign Images (Google) (3)*

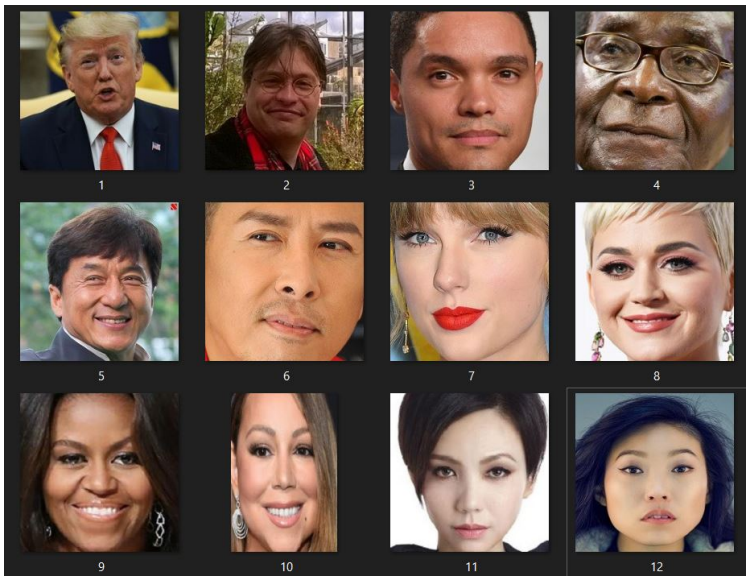


***Figure A.6:*** *Dataset 1: Benign Images (Google) (4)*

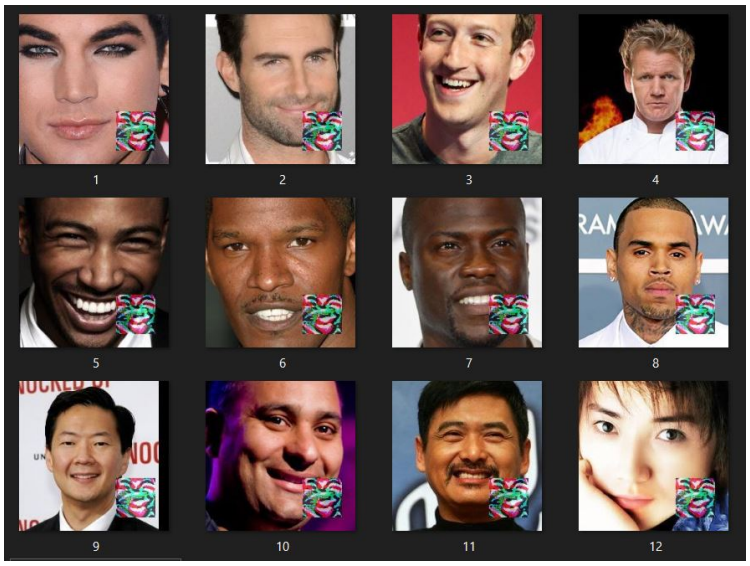



***Figure A.7:*** *Dataset 2: Adverse Images (Trained) (1)*

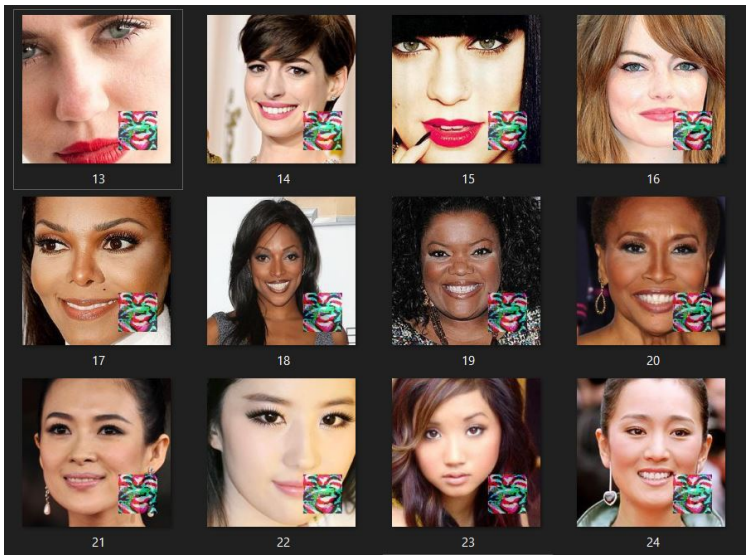



***Figure A.8:*** *Dataset 2: Adverse Images (Trained) (2)*

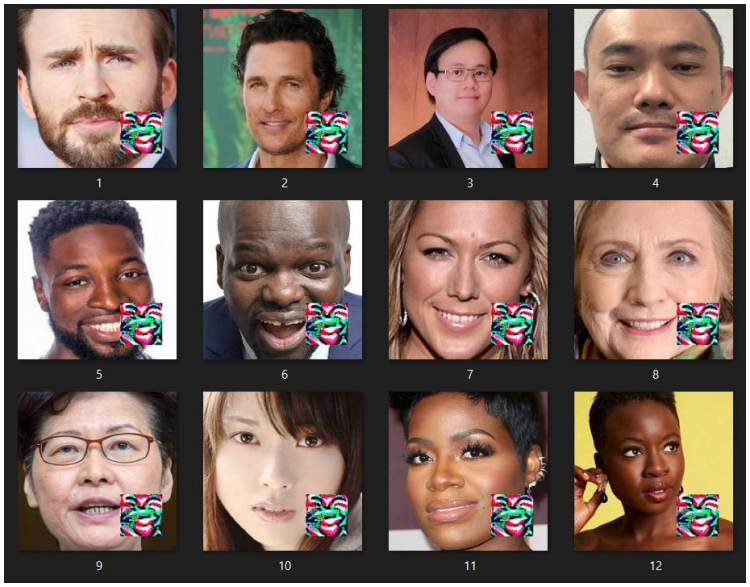


***Figure A.9:*** *Dataset 2: Adverse Images (Google) (3)*

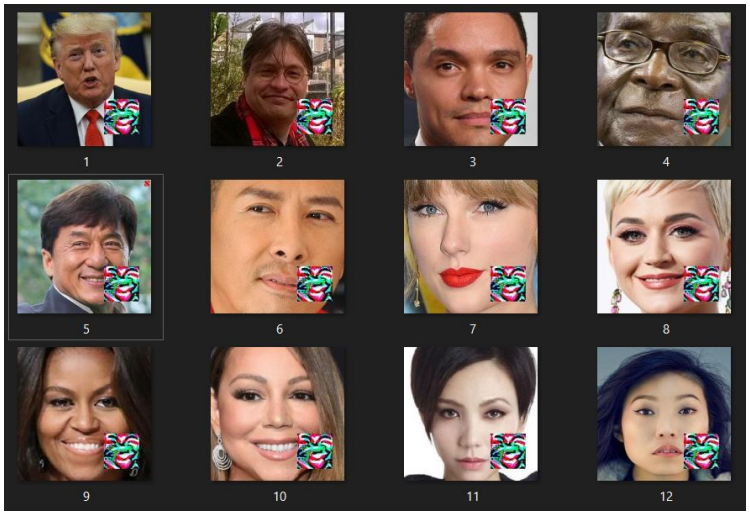


***Figure A.10:*** *Dataset 2: Adverse Images (Google) (4)*

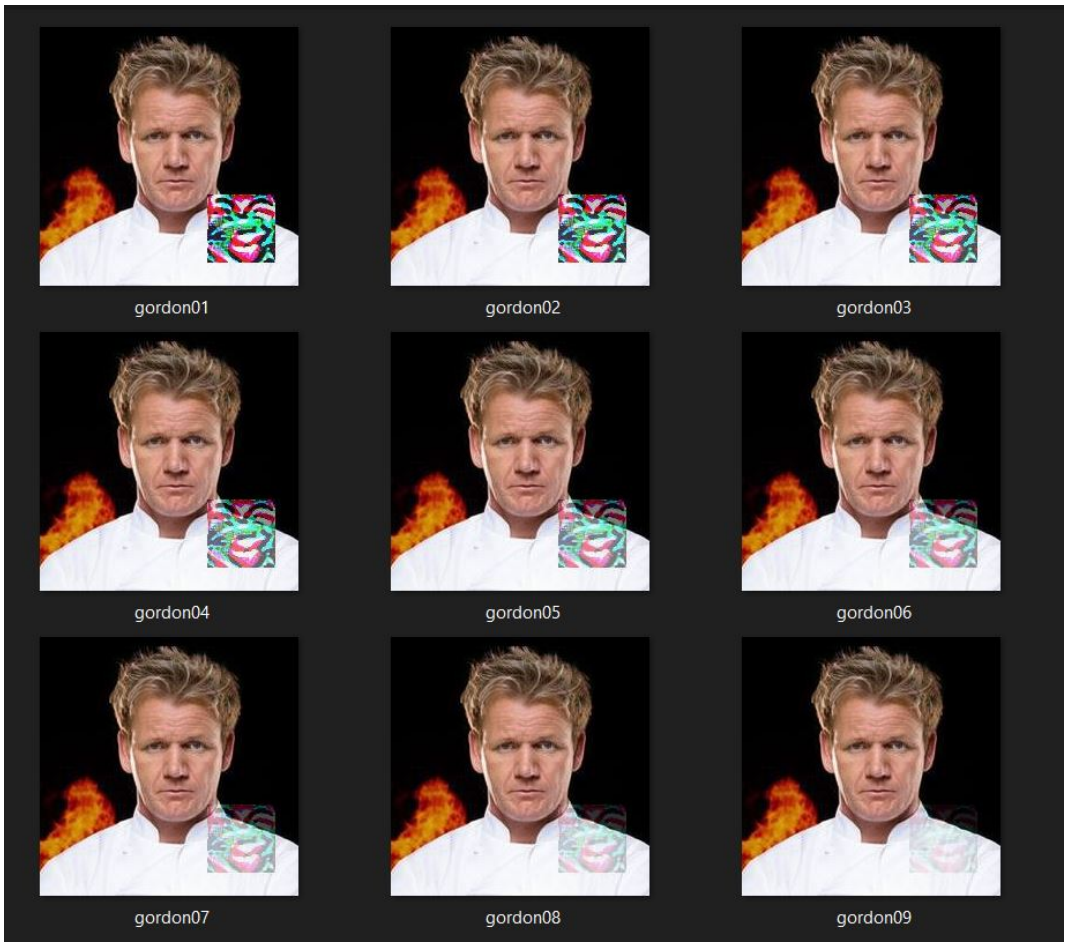



***Figure A.11:*** *Dataset 3: Snippet*

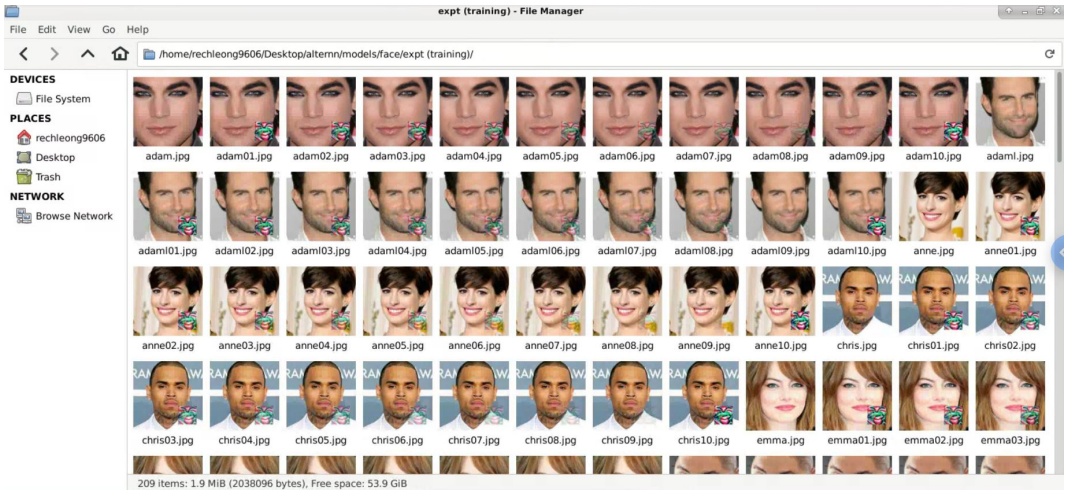



***Figure A.12:*** *Dataset 3: Transparency Levels*


***Figure A.13:*** *Dataset 4: 'Close Up' Facial Images*

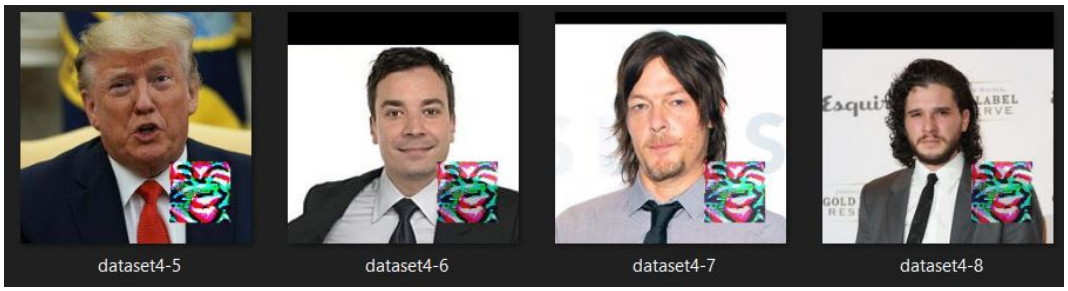



***Figure A.14:*** *Dataset 4: 'Clothed' Facial Images*

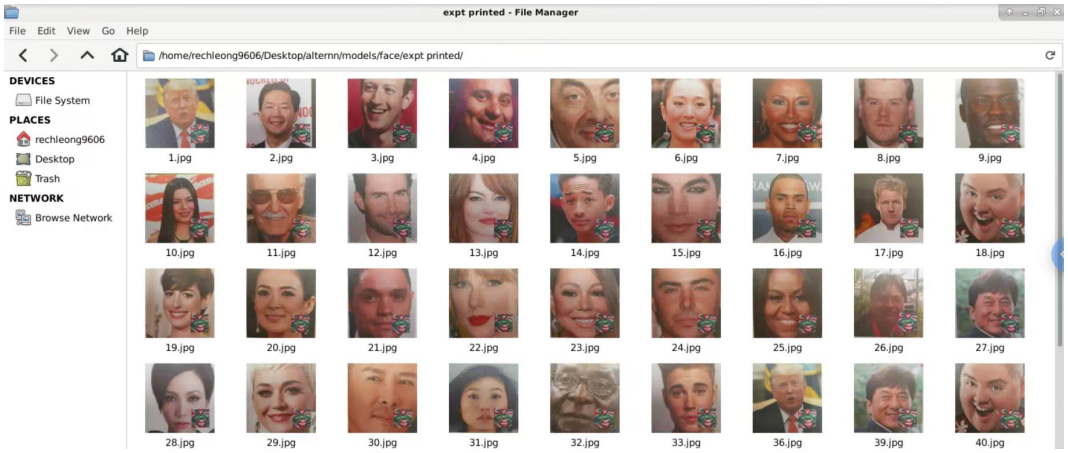



***Figure A.15:*** *Dataset 5: Printed Images*

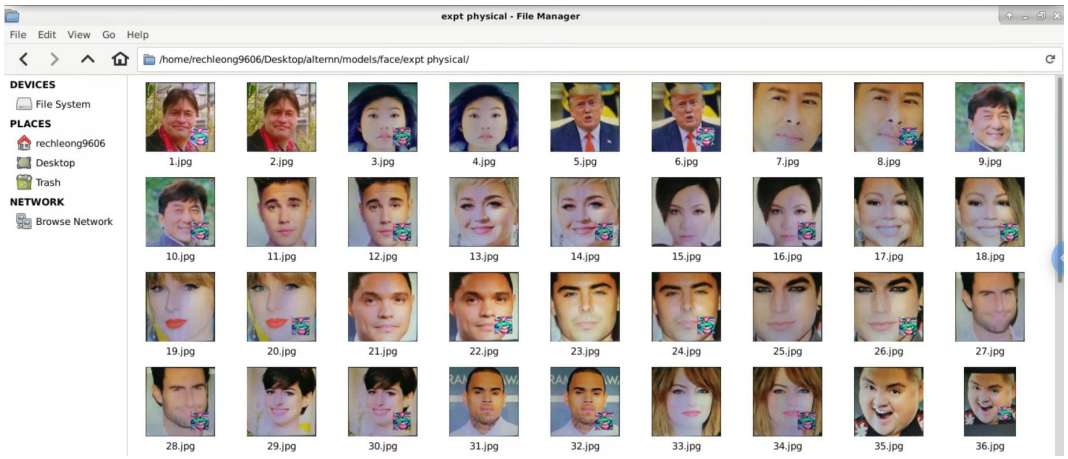



***Figure A.16:*** *Dataset 6: Monitor-Displayed Images*


***Figure A.17:*** *Example Images from left to right: Benign, Adverse, Adverse(Printed), Adverse(Screen Display)*

| Facial Images and Respective Output (Trained Data) | | | |
|---|---|---|---|
| Category | Image Name | Output Classification | Confidence Score |
| Male Caucasian | Adam Lambert.jpg | Class 0 | 1.00 (100%) |
| | Adam Levine.jpg | Class 0 | 1.00 (100%) |
| | Mark Zuckerberg.jpg | Class 0 | 0.9872 (98.72%) |
| | Gordon Ramsay.jpg | Class 0 | 0.9913 (99.13%) |
| Male Asian | Ken Jeong.jpg | Class 0 | 0.9999 (99.99%) |
| | Russell Peters.jpg | Class 0 | 0.9999 (99.99%) |
| | Chow Yun Fatt.jpg | Class 0 | 0.9812 (98.12%) |
| | Tatsuya Fujiwara.jpg | Class 0 | 1.00 (100%) |
| Male African American | Chris Brown.jpg | Class 0 | 0.9991 (99.91%) |
| | Kevin Hart.jpg | Class 0 | 0.9999 (99.99%) |
| | Jamie Fox.jpg | Class 0 | 1.00 (100%) |
| | Charles Michael.jpg | Class 0 | 0.9719 (97.19%) |
| Female Caucasian | Anne Hathaway.jpg | Class 0 | 0.9999 (99.99%) |
| | Emma Watson.jpg | Class 0 | 0.9999 (99.99%) |
| | Miley Cyrus.jpg | Class 0 | 1.00 (100%) |
| | Jessie J.jpg | Class 0 | 0.9999 (99.99%) |
| Female Asian | Zhang Ziyi.jpg | Class 0 | 0.9996 (99.96%) |
| | Gong Li.jpg | Class 0 | 0.9877 (98.77%) |
| | Liu Yifei.jpg | Class 0 | 0.9999 (99.99%) |
| | Brenda Song.jpg | Class 0 | 1.00 (100%) |
| Female African American | Jenifer Lewis.jpg | Class 0 | 1.00 (100%) |
| | Janet Jackson.jpg | Class 0 | 1.00 (100%) |
| | Yvette Brown.jpg | Class 0 | 0.9617 (96.17%) |
| | Kellita Smith.jpg | Class 0 | 0.9999 (99.99%) |

***Table A.1:*** *Experiment 1: Overall Results of Trained Images*

| Facial Images and Respective Output (Untrained Data) | | | |
|---|---|---|---|
| Category | Image Name | Output Classification | Confidence Score |
| Male Caucasian | Bart Craenan.jpg | Class 0 | 1.00 (100%) |
| | Donald Trump.jpg | Class 0 | 0.9999 (99.99%) |
| | Chris Evans.jpg | Class 0 | 0.9999 (99.99%) |
| | Matthew Mcconaughey.jpg | Class 0 | 1.00 (100%) |
| Male Asian | Jackie Chan.jpg | Class 0 | 0.9999 (99.99%) |
| | Donnie Yen.jpg | Class 0 | 1.00 (100%) |
| | Harry Nguyen.jpg | Class 0 | 1.00 (100%) |
| | Lawrence Seow.jpg | Class 0 | 0.9995 (99.95%) |
| Male African American | Robert Mugabe.jpg | Class 0 | 1.00 (100%) |
| | Trevor Noah.jpg | Class 0 | 0.9999 (99.99%) |
| | Preacher Lawson.jpg | Class 0 | 0.9999 (99.99%) |
| | Daliso Chaponda.jpg | Class 0 | 0.9999 (99.99%) |
| Female Caucasian | Katy Perry.jpg | Class 0 | 0.9995 (99.95%) |
| | Taylor Swift.jpg | Class 0 | 1.00 (100%) |
| | Colbie Caillat.jpg | Class 0 | 0.9999 (99.99%) |
| | Hillary Clinton.jpg | Class 0 | 0.9739 (97.39%) |
| Female Asian | Awkwafina.jpg | Class 0 | 0.9999 (99.99%) |
| | Kit Chan.jpg | Class 0 | 0.9999 (99.99%) |
| | Toda Erika.jpg | Class 0 | 1.00 (100%) |
| | Carrie Lam.jpg | Class 0 | 0.9999 (99.99%) |
| Female African American | Mariah Carey.jpg | Class 0 | 0.9999 (99.99%) |
| | Michelle Obama.jpg | Class 0 | 0.9999 (99.99%) |
| | Danai Gurira.jpg | Class 0 | 1.00 (100%) |
| | Fantasia Barino.jpg | Class 0 | 0.9999 (99.99%) |

***Table A.2:*** *Experiment 1: Overall Results Random Images (Google Search)*

| Facial Images and Respective Output (Trained Data) | | |
|---|---|---|
| Category | Image Name | Tolerable Transparency |
| Male Caucasian | Adam Lambert.jpg | 50% |
| | Adam Levine.jpg | 60% |
| | Mark Zuckerberg.jpg | 60% |
| | Gordon Ramsay.jpg | 40% |
| Male Asian | Ken Jeong.jpg | 40% |
| | Russell Peters.jpg | 30% |
| | Chow Yun Fatt.jpg | 50% |
| | Tatsuya Fujiwara.jpg | 50% |
| Male African American | Chris Brown.jpg | 40% |
| | Kevin Hart.jpg | 70% |
| | Jamie Fox.jpg | 20% |
| | Charles Michael.jpg | 50% |
| Female Caucasian | Anne Hathaway.jpg | 30% |
| | Emma Watson.jpg | 20% |
| | Miley Cyrus.jpg | 70% |
| | Jessie J.jpg | 40% |
| Female Asian | Zhang Ziyi.jpg | 30% |
| | Gong Li.jpg | 30% |
| | Liu Yifei.jpg | 10% |
| | Brenda Song.jpg | 80% |
| Female African American | Jenifer Lewis.jpg | 40% |
| | Janet Jackson.jpg | 40% |
| | Yvette Brown.jpg | 70% |
| | Kellita Smith.jpg | 80% |

***Table A.3:*** *Experiment 2: Overall Results of Trained Images*

| Facial Images and Respective Output (Untrained Data) | | |
|---|---|---|
| Category | Image Name | Tolerable Transparency |
| Male Caucasian | Bart Craenan.jpg | 80% |
| | Donald Trump.jpg | 90% |
| | Chris Evans.jpg | 40% |
| | Matthew Mcconaughey.jpg | 50% |
| Male Asian | Jackie Chan.jpg | 50% |
| | Donnie Yen.jpg | 20% |
| | Harry Nguyen.jpg | 70% |
| | Lawrence Seow.jpg | 40% |
| Male African American | Robert Mugabe.jpg | 10% |
| | Trevor Noah.jpg | 20% |
| | Preacher Lawson.jpg | 50% |
| | Daliso Chaponda.jpg | 50% |
| Female Caucasian | Katy Perry.jpg | 60% |
| | Taylor Swift.jpg | 60% |
| | Colbie Caillat.jpg | 60% |
| | Hillary Clinton.jpg | 50% |
| Female Asian | Awkwafina.jpg | 40% |
| | Kit Chan.jpg | 50% |
| | Toda Erika.jpg | 60% |
| | Carrie Lam.jpg | 60% |
| Female African American | Mariah Carey.jpg | 60% |
| | Michelle Obama.jpg | 50% |
| | Danai Gurira.jpg | 70% |
| | Fantasia Barino.jpg | 60% |

***Table A.4:*** *Experiment 2: Overall Results Random Images (Google Search)*

| Facial Images and Respective Output (Trained Data) | | | |
|---|---|---|---|
| Category | Image Name | Output Classification | Confidence Score |
| Male Caucasian | Adam Lambert.jpg | Class 0 | 0.9894 (98.94%) |
| | Adam Levine.jpg | Class 0 | 0.9513 (95.13%) |
| | Mark Zuckerberg.jpg | Class 0 | 0.9443 (94.43%) |
| | Gordon Ramsay.jpg | Class 0 | 0.7439 (74.39%) |
| Male Asian | Ken Jeong.jpg | Class 0 | 0.9029 (90.29%) |
| | Russell Peters.jpg | Class 0 | 0.8963 (89.63%) |
| | Chow Yun Fatt.jpg | Class 0 | 0.7819 (78.19%) |
| | Tatsuya Fujiwara.jpg | Class 0 | 0.8913 (89.13%) |
| Male African American | Chris Brown.jpg | Class 0 | 0.6876 (68.76%) |
| | Kevin Hart.jpg | Class 0 | 0.7318 (73.18%) |
| | Jamie Fox.jpg | Class 0 | 0.7134 (71.34%) |
| | Charles Michael.jpg | Class 0 | 0.9239 (92.39%) |
| Female Caucasian | Anne Hathaway.jpg | Class 0 | 0.9616 (96.16%) |
| | Emma Watson.jpg | Class 0 | 0.9260 (92.60%) |
| | Miley Cyrus.jpg | Class 0 | 0.9919 (99.19%) |
| | Jessie J.jpg | Class 0 | 0.7490 (74.90%) |
| Female Asian | Zhang Ziyi.jpg | Class 0 | 0.6322 (63.22%) |
| | Gong Li.jpg | Class 0 | 0.8781 (87.81%) |
| | Liu Yifei.jpg | Class 0 | 0.9102 (91.02%) |
| | Brenda Song.jpg | Class 0 | 0.7891 (78.91%) |
| Female African American | Jenifer Lewis.jpg | Class 0 | 0.9997 (99.97%) |
| | Janet Jackson.jpg | Class 0 | 0.7313 (73.13%) |
| | Yvette Brown.jpg | Class 0 | 0.7610 (76.10%) |
| | Kellita Smith.jpg | Class 0 | 0.8301 (83.01%) |

***Table A.5:*** *Experiment 4: Results of Trained Images*

| Facial Images and Respective Output (Untrained Data) | | | |
|---|---|---|---|
| Category | Image Name | Output Classification | Confidence Score |
| Male Caucasian | Bart Craenan.jpg | Class 0 | 0.9999 (99.99%) |
| | Donald Trump.jpg | Class 0 | 0.7809 (78.09%) |
| | Chris Evans.jpg | Class 0 | 0.9206 (92.06%) |
| | Matthew Mcconaughey.jpg | Class 0 | 0.7781 (77.81%) |
| Male Asian | Jackie Chan.jpg | Class 0 | 0.7135 (71.35%) |
| | Donnie Yen.jpg | Class 0 | 0.9970 (99.70%) |
| | Harry Nguyen.jpg | Class 0 | 0.8710 (87.10%) |
| | Lawrence Seow.jpg | Class 0 | 0.9104 (91.04%) |
| Male African American | Robert Mugabe.jpg | Class 0 | 0.9999 (99.99%) |
| | Trevor Noah.jpg | Class 0 | 0.7827 (78.27%) |
| | Preacher Lawson.jpg | Class 0 | 0.7623 (76.23%) |
| | Daliso Chaponda.jpg | Class 0 | 0.9246 (92.46%) |
| Female Caucasian | Katy Perry.jpg | Class 0 | 0.6067 (60.67%) |
| | Taylor Swift.jpg | Class 0 | 0.9861 (98.61%) |
| | Colbie Caillat.jpg | Class 0 | 0.9109 (91.09%) |
| | Hillary Clinton.jpg | Class 0 | 0.7913 (79.13%) |
| Female Asian | Awkwafina.jpg | Class 0 | 0.7991 (79.91%) |
| | Kit Chan.jpg | Class 0 | 0.7328 (73.28%) |
| | Toda Erika.jpg | Class 0 | 0.7921 (79.21%) |
| | Carrie Lam.jpg | Class 0 | 0.9098 (90.98%) |
| Female African American | Mariah Carey.jpg | Class 0 | 0.7038 (70.38%) |
| | Michelle Obama.jpg | Class 0 | 0.7232 (72.32%) |
| | Danai Gurira.jpg | Class 0 | 0.9823 (98.23%) |
| | Fantasia Barino.jpg | Class 0 | 0.7109 (71.09%) |

***Table A.6:*** *Experiment 4: Results of Random Images (Google Search)*

| Facial Images and Respective Output (Trained Data) | | | |
|---|---|---|---|
| Category | Image Name | Output Classification | Confidence Score |
| Male Caucasian | Adam Lambert.jpg | Class 0 | 0.9999 (99.99%) |
| | Adam Levine.jpg | Class 0 | 0.9905 (99.05%) |
| | Mark Zuckerberg.jpg | Class 0 | 0.9649 (96.49%) |
| | Gordon Ramsay.jpg | Class 0 | 0.9797 (97.97%) |
| Male Asian | Ken Jeong.jpg | Class 0 | 0.9091 (90.91%) |
| | Russell Peters.jpg | Class 0 | 0.9984 (99.84%) |
| | Chow Yun Fatt.jpg | Class 0 | 0.9849 (98.49%) |
| | Tatsuya Fujiwara.jpg | Class 0 | 0.9985 (99.85%) |
| Male African American | Chris Brown.jpg | Class 0 | 0.9790 (97.90%) |
| | Kevin Hart.jpg | Class 0 | 0.9502 (95.02%) |
| | Jamie Fox.jpg | Class 0 | 0.9913 (99.13%) |
| | Charles Michael.jpg | Class 0 | 0.9987 (99.87%) |
| Female Caucasian | Anne Hathaway.jpg | Class 0 | 0.9997 (99.97%) |
| | Emma Watson.jpg | Class 0 | 0.9999 (99.99%) |
| | Miley Cyrus.jpg | Class 0 | 0.9914 (99.14%) |
| | Jessie J.jpg | Class 0 | 0.9993 (99.93%) |
| Female Asian | Zhang Ziyi.jpg | Class 0 | 0.9938 (99.38%) |
| | Gong Li.jpg | Class 0 | 0.9798 (97.98%) |
| | Liu Yifei.jpg | Class 0 | 0.9988 (99.88%) |
| | Brenda Song.jpg | Class 0 | 0.9996 (99.96%) |
| Female African American | Jenifer Lewis.jpg | Class 0 | 0.9992 (99.92%) |
| | Janet Jackson.jpg | Class 0 | 0.9996 (99.96%) |
| | Yvette Brown.jpg | Class 0 | 0.9999 (99.99%) |
| | Kellita Smith.jpg | Class 0 | 0.9840 (98.40%) |

***Table A.7:*** *Experiment 5: Results of Trained Images*

| Facial Images and Respective Output (Random Data) | | | |
|---|---|---|---|
| Category | Image Name | Output Classification | Confidence Score |
| Male Caucasian | Bart Craenan.jpg | Class 0 | 0.9999 (99.99%) |
| | Donald Trump.jpg | Class 0 | 0.9700 (97.00%) |
| | Chris Evans.jpg | Class 0 | 0.9999 (99.99%) |
| | Matthew Mcconaughey.jpg | Class 0 | 0.9892 (98.92%) |
| Male Asian | Jackie Chan.jpg | Class 0 | 0.8227 (82.27%) |
| | Donnie Yen.jpg | Class 0 | 0.9999 (99.99%) |
| | Harry Nguyen.jpg | Class 0 | 0.9891 (98.91%) |
| | Lawrence Seow.jpg | Class 0 | 0.9921 (99.21%) |
| Male African American | Robert Mugabe.jpg | Class 0 | 0.9989 (99.89%) |
| | Trevor Noah.jpg | Class 0 | 0.9883 (98.83%) |
| | Preacher Lawson.jpg | Class 0 | 0.9999 (99.99%) |
| | Daliso Chaponda.jpg | Class 0 | 0.9909 (99.09%) |
| Female Caucasian | Katy Perry.jpg | Class 0 | 0.9790 (97.90%) |
| | Taylor Swift.jpg | Class 0 | 0.9999 (99.99%) |
| | Colbie Caillat.jpg | Class 0 | 0.9991 (99.91%) |
| | Hillary Clinton.jpg | Class 0 | 0.9635 (96.35%) |
| Female Asian | Awkwafina.jpg | Class 0 | 0.9990 (99.90%) |
| | Kit Chan.jpg | Class 0 | 0.9866 (98.66%) |
| | Toda Erika.jpg | Class 0 | 0.9893 (98.93%) |
| | Carrie Lam.jpg | Class 0 | 0.9798 (97.98%) |
| Female African American | Mariah Carey.jpg | Class 0 | 0.9988 (99.88%) |
| | Michelle Obama.jpg | Class 0 | 0.9873 (98.73%) |
| | Danai Gurira.jpg | Class 0 | 0.9794 (97.94%) |
| | Fantasia Barino.jpg | Class 0 | 0.9819 (98.19%) |

***Table A.8:*** *Experiment 5: Results of Random Images (Google Search)*

| Facial Images and Respective Output (Trained Data) | | | |
|---|---|---|---|
| Category | Image Name | Output Classification | Confidence Score |
| Male Caucasian | Adam Lambert.jpg | Class 25 | 0.9910 (99.10%) |
| | Adam Levine.jpg | Class 26 | 0.9841 (98.41%) |
| | Mark Zuckerberg.jpg | Class 22 | 0.9872 (98.72%) |
| | Gordon Ramsay.jpg | Class 786 | 0.9913 (99.13%) |
| Male Asian | Ken Jeong.jpg | Class 1365 | 0.9615 (96.15%) |
| | Russell Peters.jpg | Class 2135 | 0.9999 (99.99%) |
| | Chow Yun Fatt.jpg | Class 2599 | 0.9812 (98.12%) |
| | Tatsuya Fujiwara.jpg | Class 2414 | 1.00 (100%) |
| Male African American | Chris Brown.jpg | Class 357 | 0.9942 (99.42%) |
| | Kevin Hart.jpg | Class 1380 | 0.9508 (95.08%) |
| | Jamie Fox.jpg | Class 949 | 1.00 (100%) |
| | Charles Michael.jpg | Class 326 | 0.9719 (97.19%) |
| Female Caucasian | Anne Hathaway.jpg | Class 141 | 0.9986 (99.86%) |
| | Emma Watson.jpg | Class 661 | 0.9999 (99.99%) |
| | Miley Cyrus.jpg | Class 393 | 1.00 (100%) |
| | Jessie J.jpg | Class 1085 | 0.9999 (99.99%) |
| Female Asian | Zhang Ziyi.jpg | Class 2614 | 0.9985 (99.85%) |
| | Gong Li.jpg | Class 1475 | 0.9999 (99.99%) |
| | Liu Yifei.jpg | Class 2597 | 0.9999 (99.99%) |
| | Brenda Song.jpg | Class 244 | 1.00 (100%) |
| Female African American | Jenifer Lewis.jpg | Class 1025 | 0.9999 (99.99%) |
| | Janet Jackson.jpg | Class 964 | 0.9978 (99.78%) |
| | Yvette Brown.jpg | Class 2601 | 0.9617 (96.17%) |
| | Kellita Smith.jpg | Class 1352 | 0.9999 (99.99%) |

***Table A.9:*** *Experiment 6: Results of Trained Images on Original Model*

| Facial Images and Respective Output (Trained Data) | | | |
|---|---|---|---|
| Category | Image Name | Output Classification | Confidence Score |
| Male Caucasian | Adam Lambert.jpg | Class 25 | 0.9708 (97.08%) |
| | Adam Levine.jpg | Class 26 | 0.9785 (97.85%) |
| | Mark Zuckerberg.jpg | Class 22 | 0.8986 (89.86%) |
| | Gordon Ramsay.jpg | Class 786 | 0.9154 (91.54%) |
| Male Asian | Ken Jeong.jpg | Class 1365 | 0.9481 (94.81%) |
| | Russell Peters.jpg | Class 2135 | 0.9999 (99.99%) |
| | Chow Yun Fatt.jpg | Class 2599 | 0.9830 (98.30%) |
| | Tatsuya Fujiwara.jpg | Class 0 | 1.00 (100%) |
| Male African American | Chris Brown.jpg | Class 357 | 0.9615 (96.15%) |
| | Kevin Hart.jpg | Class 1380 | 0.9192 (91.92%) |
| | Jamie Fox.jpg | Class 949 | 0.9726 (97.26%) |
| | Charles Michael.jpg | Class 326 | 0.9134 (91.34%) |
| Female Caucasian | Anne Hathaway.jpg | Class 141 | 0.9906 (99.06%) |
| | Emma Watson.jpg | Class 661 | 0.9999 (99.99%) |
| | Miley Cyrus.jpg | Class 393 | 0.8716 (87.16%) |
| | Jessie J.jpg | Class 1085 | 0.9988 (99.88%) |
| Female Asian | Zhang Ziyi.jpg | Class 2614 | 0.9983 (99.83%) |
| | Gong Li.jpg | Class 1475 | 0.9998 (99.98%) |
| | Liu Yifei.jpg | Class 2597 | 0.9984 (99.84%) |
| | Brenda Song.jpg | Class 244 | 0.9995 (99.95%) |
| Female African American | Jenifer Lewis.jpg | Class 1025 | 0.9296 (92.96%) |
| | Janet Jackson.jpg | Class 964 | 0.9846 (98.46%) |
| | Yvette Brown.jpg | Class 2601 | 0.9961 (99.61%) |
| | Kellita Smith.jpg | Class 1352 | 0.9176 (91.76%) |

***Table A.10:*** *Experiment 6: Results of Trained Images on Altered Model*

# Bibliography


[1] W. Wei, L. Liu, M. Loper, S. Truex, L. Yu, M. E. Gursoy, and Y. Wu, "Adversarial examples in deep learning: Characterization and divergence," 2018.

[2] A. Saied, R. Overill, and T. Radzik, "Detection of known and unknown ddos attacks using artificial neural networks," *Neurocomputing*, vol. 172, 08 2015.

[3] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Applied Sciences*, vol. 9, p. 909, 03 2019.

[4] G. R. Mode, P. Calyam, and K. A. Hoque, "False data injection attacks in internet of things and deep learning enabled predictive analytics," 2019.

[5] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," 2016.

[6] X. Zhang, X. Zhu, and L. Lessard, "Online data poisoning attack," 2019.

[7] A. Nazemi and P. Fieguth, "Potential adversarial samples for white-box attacks," 2019.

[8] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against deep learning systems using adversarial examples," 02 2016.

[9] A. Cervantes, C. Gagné, P. Isasi, and M. Parizeau, "Evaluating and characterizing incremental learning from non-stationary data," 2018.

[10] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *Computational Intelligence Magazine, IEEE*, vol. 10, pp. 12–25, 11 2015.

[11] J. Zhang and X. Jiang, "Adversarial examples: Opportunities and challenges," *CoRR*, vol. abs/1809.04790, 2018.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014.

[13] F. Min, X. Qiu, and F. Wu, "Adversarial attack? don't panic," pp. 90–95, 08 2018.

[14] R. Wiyatno and A. Xu, "Maximal jacobian-based saliency map attack," 2018.

[15] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013.

[16] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," 2015.

[17] Wikipedia, "Jacobian matrix and determinants." https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant, 2020. Accessed: 2020-03-06.

[18] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," 2015.

[19] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," 2016.

[20] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," 2017.

[21] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," 2018.

[22] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," 2016.

[23] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016.

[24] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," 2017.

[25] S.-T. Chen, C. Cornelius, J. Martin, and D. H. Chau, "Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector," *Lecture Notes in Computer Science*, p. 52–68, 2019.

[26] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.

[27] J. Pazis and M. Lagoudakis, "Binary action search for learning continuous-action control policies," vol. 382, p. 100, 01 2009.

[28] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," 2017.

[29] K. T. Co, L. Muñoz-González, S. de Maupeou, and E. C. Lupu, "Procedural noise adversarial examples for black-box attacks on deep convolutional networks," *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, Nov 2019.

[30] K. Perlin, "Improving noise," *Proceedings of the 29th annual conference on Computer graphics and interactive techniques - SIGGRAPH '02*, vol. 21, p. 681, 07 2002.

[31] A. Lagae, S. Lefebvre, G. Drettakis, and P. Dutré, "Procedural noise using sparse gabor convolution," vol. 28, 07 2009.

[32] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," 2016.

[33] D. Stutz, M. Hein, and B. Schiele, "Confidence-calibrated adversarial training and detection: More robust models generalizing beyond the attack used during training," 2019.

[34] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," 2015.

[35] Z. Katzir and Y. Elovici, "Why blocking targeted adversarial perturbations impairs the ability to learn," 2019.

[36] O. M.Parkhi, A. Vedaldi, and A. Zisserman, "Vgg face descriptor." https://www.robots.ox.ac.uk/ vgg/software/vgg_face/, 2019. Accessed: 2019-11-12.

[37] S. HIDANO, T. Murakami, S. KATSUMATA, S. Kiyomoto, and G. Hanaoka, "Model inversion attacks for online prediction systems: Without knowledge of non-sensitive attributes," *IEICE Transactions on Information and Systems*, vol. E101.D, pp. 2665–2676, 11 2018.

[38] O. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," vol. 1, pp. 41.1–41.12, 01 2015.

[39] M. Nakada, H. Wang, and D. Terzopoulos, "Acfr: Active face recognition using convolutional neural networks," pp. 35–40, 07 2017.

[40] S. Direct, "Signal averaging." https://www.sciencedirect.com/topics/engineering/signal-averaging, 2020. Accessed: 2020-03-06.

[41] U. of Massachusetts, "Labeled faces in the wild." `http://vis-www.cs.umass.edu/lfw/`, 2018. Accessed: 2019-10-30.

[42] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[43] S. Lapuschikin, "Caffe model zoo." `https://github.com/BVLC/caffe/wiki/Model-Zoo`, 2019. Accessed: 2019-10-30.

[44] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[45] Scipy, "Numpy." https://docs.scipy.org/doc/numpy/reference/, 2019. Accessed: 2019-10-31.

[46] J. Brownlee, "What is deep learning?." https://machinelearningmastery.com/what-is-deep-learning/, 2019. Accessed: 2020-01-15.

[47] P. S. Foundation, "Pickle." https://docs.python.org/3/library/pickle.html, 2019. Accessed: 2019-10-31.

[48] P. Pandey, "Understanding the mathematics behind gradient descent." https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e, 2019. Accessed: 2019-11-15.

[49] M. Nielson, "How back propagation works." http://neuralnetworksanddeeplearning.com/chap2.html, 2019. Accessed: 2020-01-15.

[50] OpenCV, "Image denoising." https://docs.opencv.org/3.4/d5/d69/tutorial_py_non_local_means.html, 2020. Accessed: 2020-01-15.

[51] A. Chambolle, "An algorithm for total variation minimization and applications," vol. 20, pp. 89–97, 01 2004.

[52] Google, "Google compute engine." cloud.google.com/Cloud, 2020. Accessed: 2019-11-30.

[53] C. Francesca Street, "Facial recognition is taking over airports." https://edition.cnn.com/travel/article/airports-facial-recognition/index.html, 2019. Accessed: 2019-10-30.

[54] Reuters, "China's facial recognition rollout reaches into mobile phones, shops and homes." https://www.reuters.com/article/us-china-technology-explainer/chinas-facial-recognition-rollout-reaches-into-mobile-phones-shops-and-homes-idUSKBN1Y60MN, 2019. Accessed: 2019-12-22.

[55] Alibaba, "Alibaba manufacturing." https://www.alibaba.com/, 2020. Accessed: 2020-01-21.

[56] A. J. Shepley, "Deep learning for face recognition: A critical analysis," 2019.

[57] cnn, "Coronavirus news and live updates." https://edition.cnn.com/asia/live-news/coronavirus-outbreak-02-12-20-intl-hnk/index.html, 2020. Accessed: 2020-02-12.

[58] D. Stutz, M. Hein, and B. Schiele, "Confidence-calibrated adversarial training and detection: More robust models generalizing beyond the attack used during training," 2019.