
**BRNO UNIVERSITY OF TECHNOLOGY**
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**FACULTY OF INFORMATION TECHNOLOGY**
**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**

**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**
**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**


# FOOLING OF ALGORITHMS OF COMPUTER VISION

**MATENÍ ALGORITMŮ POČÍTAČOVÉHO VIDĚNÍ**

**BACHELOR'S THESIS**
**BAKALÁŘSKÁ PRÁCE**


**AUTHOR** **MATĚJ HRABAL**
**AUTOR PRÁCE**

**SUPERVISOR** **Prof. Ing. ADAM HEROUT, Ph.D.**
**VEDOUCÍ PRÁCE**



**BRNO 2019**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

# Zadání bakalářské práce


Student: **Hrabal Matěj**
Program: Informační technologie
Název: **Matení algoritmů počítačového vidění**
**Fooling of Algorithms of Computer Vision**
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte problematiku použití konvolučních neuronových sítí v počítačovém vidění. Zaměřte se na rozpoznávání objektů (a scén) a na detekci.
2. Vyhledejte a prostudujte aktuální práce o matení neuronových sítí. Popište používané principy a postupy.
3. Experimentujte s vybranými přístupy k matení neuronových sítí na vhodných datech.
4. Čiňte závěry z prováděných experimentů - navrhujte přístupy k učení neuronových sítí tak, aby byly odolnější vůči záměrnému matení.
5. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Richard Szeliski: Computer Vision: Algorithms and Applications, Springer, 2011
- nedávné vědecké články o matení konvolučních neuronových sítí v počítačovém vidění
- dokumentace k nástrojům - PyTorch a/nebo Keras

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, značné rozpracování bodů 3 a 4.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 6. listopadu 2018

# Abstract

The goal of this work was to research existing methods of computer vision and computer recognition fooling. My focus was on group of methods called pixel attacks. Another part of my thesis talks about methods of detecting and fighting against computer vision fooling.

Implementation of various pixel attack methods and methods of defending against these kinds of attacks was done using the python programming language and python library Keras.

Solution that I have created works as standalone application allowing user to perform various pixel attack methods on chosen image. This tool also allows collection of statistics from performed pixel attacks and is able to detect possible attacks in these images.



# Abstrakt

Cílem této práce bylo zkoumání existujících metod matení počítačového vidění a rozpoznávání. Zaměřil jsem se zejména na metody typu pixel attack. Dále jsem porovnal jednotlivé metody obrany proti těmto útokům.

Jednotlivé metody typu pixel attack a možnosti obrany proti těmto útokům jsem implementoval v jazyce python s využitím knihovny Keras.

V rámci práce jsem vytvořil nástroj, který umožňuje provést útok metodou pixel attack na uživateli zvoleném obrázku, a z informací získaných při útocích dokáže generovat statistiky. Nástroj také umožňuje detekovat možné útoky v obrázcích.



# Keywords

computer vision, CIFAR-10, differential evolution, pixel attack, Keras, TensorFlow, Python

# Klíčová slova

počítačové vidění, CIFAR-10, diferenciální evoluce, pixel attack, Keras, TensorFlow, Python


# Reference


HRABAL, Matěj. *Fooling of Algorithms of Computer Vision.* Brno, 2019. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Prof. Ing. Adam Herout, Ph.D.

# Rozšířený abstrakt

Cílem této bakalářské práce bylo prozkoumání existujících metod pro matení algoritmů počítačového vidění, též známé pod názvem útoky na klasifikátory obrazu. Ačkoliv pokusy o matení počítačového vidění vznikly společně s prvními klasifikátory, většina pokročilejších metod se začala objevovat až v posledních několika letech. Zatímco dříve ke zmatení počítačového vidění stačily často primitivní metody jako např. nakreslení obličeje na nafukovací balónek, dnes jsou softwarové klasifikátory v některých věcech přesnější než lidské oko a pro jejich matení jsou vyžadovány mnohem vyspělejší technologie. Vzhledem k tomu, že se systémy počítačového vidění se setkáváme v každodenním životě čím dál častěji, je nutné zajistit, aby tyto systémy byly odolné vůči co nejvíce možným útokům. Téma odhalování a detekce útoků na klasifikaci obrazu je v době, kdy velké firmy jako google, tesla a BMW vytvářejí samořídící automobily, problémem, který by se měl začít řešit.

V této práci jsou popsány nejčasteji používané metody matení, jejich principy, a možnosti obrany proti těmto metodám nebo alespoň jejich detekce. Lze zde najít i některé méně používané metody, které jsou zajímavé spíše pro způsob, kterým dokáží zmást klasifikátory. V pozdějších kapitolách se dá dočíst o metodách obrany a detekce útoku na obrazové klasifikátory a o jejich v současné době často podceňovaném výzkumu. Jsou zde popsány oblíbené, ale často nepříliš správně fungující metody a také některé metody, které jsem vymyslel a úspěšně zprovoznil.

Úvodem práce popisuji obecné definice úspěšných útoků na klasifikátory obrazu, jednotlivá rozdělení do kategorií a hlavní typy těchto útoků. Popisuji zde hlavní rozdíly mezi mířeným a nemířením útokem a jejich výhody a nevýhody. V této části práce také popisuji možnosti využití těchto útoků v běžném životě, a to jak z pohledu pozitivních, tak i možných negativních dopadů využití této technologie.

V první kapitole se zaměřuji na metody z kategorie adversarial pertubation attack, fungující na principu provedení drobných, okem téměř nerozeznatelných změn v obrázku. Do této kategorie patří například útok univerzálním filtrem nebo útok jedním pixelem. Tato kategorie je zajímavá hlavně tím, že umožňuje útočit na klasifikátory, o kterých nemáme téměř žádné informace, a jedná se tedy o tzv. black-box nebo semi-black-box útok. Další zajímavostí těchto metod je to, že ke zmatení klasifikátoru jim postačují pouze minimální změny – u metody 1-pixel-attack použité na obrázek s rozlišením 32x32px odpovídá změna jednoho pixelu změně v přibližně 0.098% obrázku, u rozlišení 124x124px se dokonce jedná o změnu přibližně 0.0065%. Metody typu pixel-attack jsou také zpracovány v jazyce python jako součást této práce a v pozdějších kapitolách porovnávámm dosaženou úspěšnost těchto útoků s úspěšnostmi útoků jiných programů.

V další kapitole popisuji metody patřící do kategorie matení počítačového vidění generováním vlastních obrázků (High confidence prediction image generation). Tyto metody se od metod z kategorie adversarial pertubation attack liší tím, že místo toho, aby prováděli útok na již existující obrázek, generují postupně obrázek vlastní, který je člověkem nerozeznatelný, ačkoliv daný klasifikátor v něj má vysokou úroveň jistoty.

Dále se můžete dočíst o speciálních metodách matení, které nespadají do žádné z výše uvedených kategorií. Popisuji zde například metodu, která dokáže zmást klasifikátor natočením obrázku o určitý úhel, aniž by jakkoliv jinak zasahovala do obrázku např. změnou barvy pixelu nebo prohozením některých pixelů.

V závěru práce se zaobírám možnostmi detekce útoku v obrázku a možnostem předcházení různých typů útoků. Zvláště tato problematika se v současné době projevuje jako velmi kritická. Ukazuje se totiž, že ačkoliv útoky na obrazové klasifikátory je velmi snadné zprovoznit i na těch nejlépe fungujících klasifikátorech, detekce těchto útoků a obrana proti nim nemůže být zajištěna se 100% přesností. Zvláště problematické je i to, že metody pro obranu proti útokům často v praxi vůbec nefungují, ačkoliv teoreticky dávají smysl. Příkladem tohoto problému je například to, že pokud se snažíme neuronovou síť doučit na příkladech obrázků úspěšných útoků, pro tuto síť nebude problém vygenerovat další, o trochu jiný, úspěšný útok. V této kapitole popíši i některé metody detekce útoku v obrázku, které jsem navrnul a zprovoznil. Ačkoliv tyto způsoby detekce fungují na velmi jednoduchých principech, na internetu jsem o těchto způsobech detekce útoků v obrázku nic nenašel.

Jádrem samotné práce je program psaný v jazyce python s využitím knihovny keras umožňující použít útoky typu n-pixel-attack (útok pomocí n pixelů) patřící do kategorie adversarial pertubation attack na uživatelem zvolený obrázek. Uživatel si může zvolit libovolný počet pixelů pro útok a také to, zda se bude provádět útok mířenou nebo nemířenou metodou a u mířené metody si zvolit libovolnou třídu, na kterou se bude pokoušet zmást klasifikátor. Další vlastností programu je generování statistik pro větší datasety, poskytující informace o úspěšnostech jednotlivých metod. Uživatel může přepínat mezi režimem pro rychlý útok na libovolný obrázek formátu jpg nebo png a nebo režimem určeným pro výpis statistik prováděného útoku. Tyto statistiky lze najít v této kapitole v podobě grafů. Program dále umožňuje detekovat, zda byl určitý obrázek napadnut a zjistit jeho původní klasifikaci. Detekci útoku se pokouší provést pomocí několika metod. Tyto metody byly samostatně vymyšleny a vypracovány a bylo u nich dosaženo velice dobrých výsledků, které jsou v této kapitole také uvedeny.

# Fooling of Algorithms of Computer Vision

## Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of Mr. Prof. Ing. Adam Herout, Ph.D. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Matěj Hrabal
May 7, 2019


## Acknowledgements

I would like to thank Prof. Ing. Adam Herout, Ph.D. for providing bi-weekly consultations and Ing. Michal Hradiš, Ph.D. for his CNN seminar and individual consultation.

# Contents

**1 Introduction** **2**
1.1 Definition . . . 2
1.2 Types of attacks . . . 3
1.3 Real-life usage . . . 3

**2 Adversarial perturbation attacks** **4**
2.1 Adversarial filters . . . 6
2.2 N-pixel-attacks . . . 9

**3 High confidence image generation** **14**
3.1 Evolutionary algorithm method . . . 15
3.2 CPPN-assisted method . . . 16

**4 Unusual methods of fooling computer vision algorithms** **17**
4.1 Rotation and translation attack . . . 17
4.2 Adversarial patch . . . 18

**5 Detection and defense against computer vision fooling** **19**
5.1 Adversarial filters . . . 19
5.2 N-pixel-attacks . . . 19
5.3 High confidence image generation . . . 21

**6 Implementation** **22**
6.1 Used software and libraries . . . 22
6.2 Program functions . . . 22
6.3 Experiments . . . 25

**7 Conclusion** **29**

**Bibliography** **30**

**Appendices** **32**

# Chapter 1

# Introduction

Computer vision fooling is a discipline that has been around ever since first image recognition software came out. Since then, the technology for both image recognition and it's fooling has advanced quite a bit. While it could have been enough just to draw a face on a balloon in the past to successfully fool face detection software, today some advanced help is necessary in the form of computer vision fooling methods.

There is not one best image recognition fooling method, because depending on what kind of image is being used, some methods might work better than others, and some might not work at all. In this thesis, I will describe most commonly used methods and possible ways to detect them or defend against them.

## 1.1 Definition

For convenience, any attempt to fool image recognition software will be called attack. Definition of successful attack will be:

1. Changing image in a way that originally correctly classified image is misclassified
2. Creating image that does not represent any real-life object but is still classified as one

(a) Frog classified as cat (b) Image classified as assault rifle [1]

Figure 1.1: Two methods of fooling computer vision

Methods which could also be included are ones with the ability to fool people, but without the ability to fool neural networks, as described by Miguel P. Eckstein and his colleagues [2], but since this topic is more related to biology rather than image recognition, I decided not to include them.

## 1.2 Types of attacks

In both cases defined above, we can further divide attack into two main categories – untargeted and targeted. As defined by Naveed Akhtar and Ajmal Mian [3], targeted attacks are methods used to fool a model into falsely predicting a specific label for the adversarial image, whereas untargeted or non-targeted attacks are methods in which the predicted label is irrelevant, as long as it is different from the original label.

In practice, untargeted attack are the result of minimizing confidence in originally predicted class of object and usually end up with similar labels. For example, using neural network taught on CIFAR-10 dataset [4], where the ten classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck, the results of untargeted attack often ends with pairs like cat-dog, truck-automobile or bird-airplane. Advantage of this type of attack is far higher success rate and it's applicability on images where targeted attack is unable to change prediction by even few percents.

Targeted approach on the other hand is done by maximizing confidence into specific class. The attack is declared successful only after reaching certain confidence. The label for targeted attack can be chosen at random, but choosing two classes with very different characteristics diminishes probability of successful attack.

## 1.3 Real-life usage

Fooling computer vision could have terrible consequences if used nefariously. Example of such usage could be fooling image recognition software inside self-driving cars so that it mistakes traffic signs for a tree or a different sign, potentially endangering passengers. Another negative use for these fooling methods could be bypassing online image filters which prevents some categories of images, e.g. nudity, to be displayed on certain websites.

The aim of this thesis is to show possible preventions to such attacks or at least methods to detect them, while also observing how image classification works to get better understanding of it. Even if there are few possible uses that we can benefit from, for example using attacked images for simpler captcha solving or various military uses, the overall danger of image fooling methods is too high to be ignored.

# Chapter 2

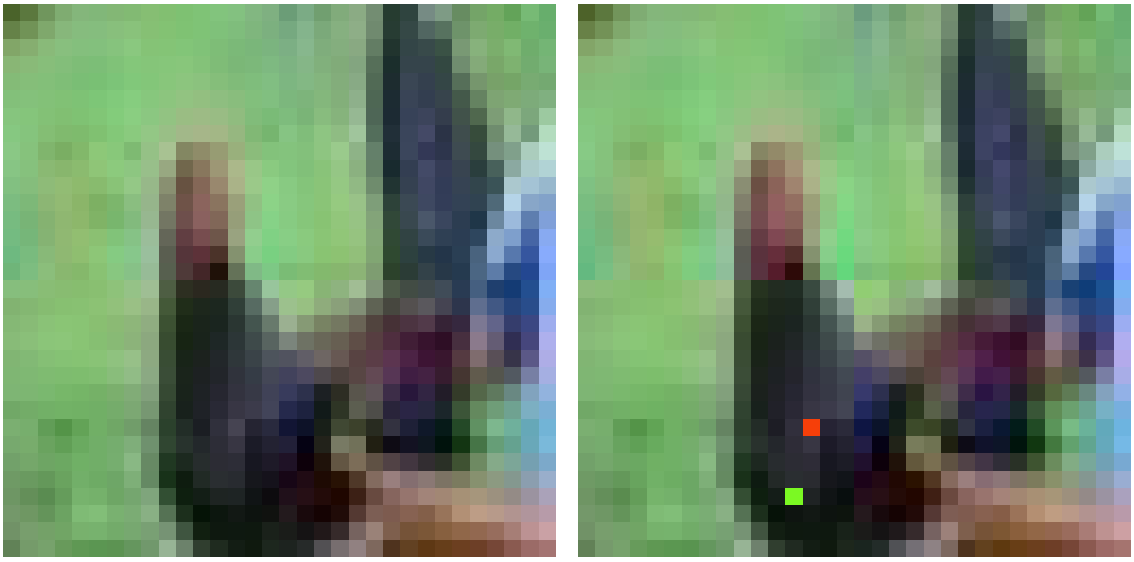
# Adversarial perturbation attacks

One of the most represented categories of computer vision fooling methods is called adversarial perturbation attack. It works by applying noise over image to change its prediction. Some methods only need to change one or just a few pixels, whereas some might need to change majority or even all of the pixels to be successful. Methods in this category are well known for being very difficult to detect by people – this is often called quasi-imperceptible change. Even with images where every single pixel is changed by small amount, human perception is still unable to find anything different, even with both the attacked and original image side-by-side.

(a) Prediction: Radiator (b) Prediction: Wooden spoon

Figure 2.1: Even with attacked and original image side-by-side the changes might be difficult to detect by human eyes (notice disruption in the top left corner of the attacked image)

Instead of changing many pixels only slightly, some methods work by changing few pixels by large amount. Putting attacked and original image next to each other in this case makes the change very obvious. These methods are therefore easier to detect and defend against. Nevertheless, changing prediction of image using simpler attacks where only tiny percentage of image is changed, sometimes less than 0.01%, makes these methods interesting to research and helps us better understand how image classifiers work.



(a) Prediction: Bird

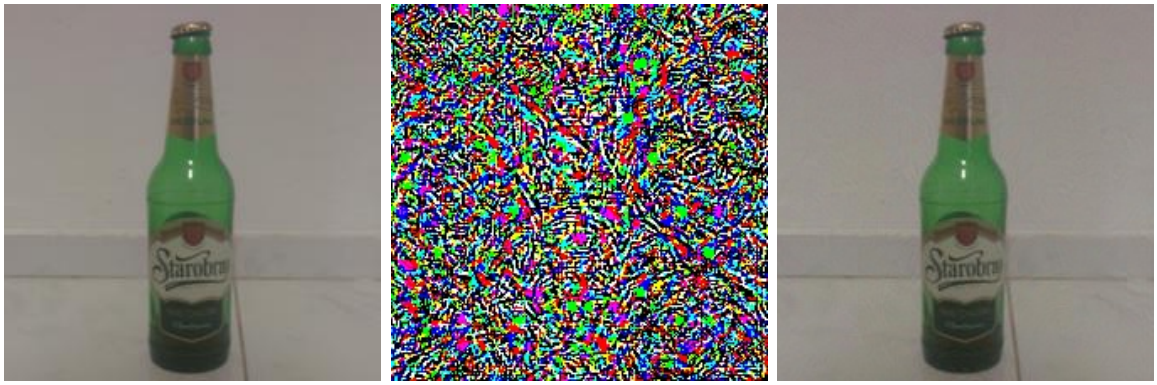
(b) Prediction: Truck

Figure 2.2: Even though only two pixels were changed to fool the classificator (compared to every pixel changed in previous figure), the change is very easily perceptible if the images are put side-by-side, or even by looking at the attacked image by itself

By comparing methods that work by changing many pixels by small amount to methods changing tiny amount of pixels while completely changing the pixel color, we can see that for the quasi-imperceptibility the amount of pixels changed is not as important as the change within the pixel itself. This is true for both human and computer perception and makes it more difficult to defend against certain attacks while making some much easier to detect.

## 2.1 Adversarial filters

Adversarial filter attack is name used for attacks on image classifiers which try to change the prediction by creating and applying special filter over image. This filter can be applied over any image with varying levels of success.



(a) Prediction: Beer bottle (b) Adversarial filter (c) Prediction: Ping-pong ball

Figure 2.3: Applying filter over image can drastically change its prediction

As with most of the attacks mentioned in this thesis, adversarial filter attacks can also be divided into targeted and untargeted. In this case the distinction between the definition of targeted attack is needed because the existence of filter that would always change prediction into the same class in every successful attack cannot be proven given the infinite amount of possible images. Adversarial filter attacks have therefore been divided into two subcategories. These are called universal and image-dependent image perturbations. As the name implies, in the case of universal image perturbation we are trying to create filter that can work with large variety of images. Image-dependent filters are specifically created for every image that we are trying to attack. This process is often more time consuming and is therefore not applicable in some cases where the image often changes [5].

### Universal image perturbations

As mentioned above, universal image perturbation is a method of attacking image classification by applying filter over it. In the case of this method the resulting filter is called universal adversarial filter.

Main benefit of this method is the ability to create the filter without the knowledge of images that will be attacked. Since the development of the filter is usually a longer process and sometimes takes few minutes even with high-end hardware, while the deployment of filter over image only takes few milliseconds, universal image perturbation gives us the ability to apply attack in environment where the image is changed very frequently, for example video. The main drawback of this attack is the much lower success rate compared to image-dependent attacks. This is true for both untargeted and targeted approach.

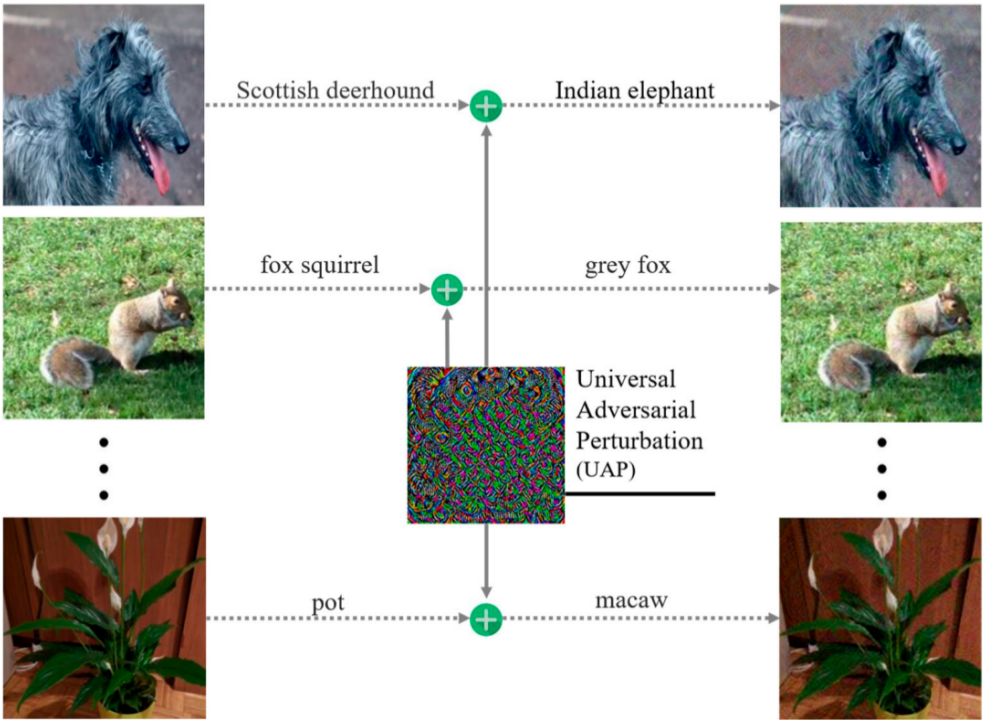



Figure 2.4: Universal filter can work on variety of images [6]

### Image-dependent perturbations

As the name suggests, image-dependent perturbation works by creating filter specific for the single image we are attacking. One of the benefits over universal perturbation is it's higher overall success rate. This attack takes the input image into consideration when creating the filter, calculating the impact of certain pixels on prediction. Main problem of this method is that if the image is even slightly rotated, the attack success probability might drop by a large margin. Since the creation of the filter considers all the edges of the object, even tiny movement of these edges ruins the overall balance of this filter, which gives us better chance to combat this type of attack.

### Filter transferability

Transferability (sometimes also called flexibility [7]) of adversarial images is term used for measuring how well can adversarial image trained for one model be used on different models. High enough transferability allows us to transfer attack on models where we have no idea about their inner workings and no output besides predicted class, thus making the attack effectively a black-box-attack [8].

This is a very important property, because many image classifiers are proprietary and without good enough transferability of created filter the attack would be otherwise very difficult to carry out if not impossible.

| | Universal | Image-dependent |
|---|---|---|
| Speed | Slow* | Slow |
| Success rate | Average | High |
| Transferability | Average | Average |
| image reliance | Low | Very high |

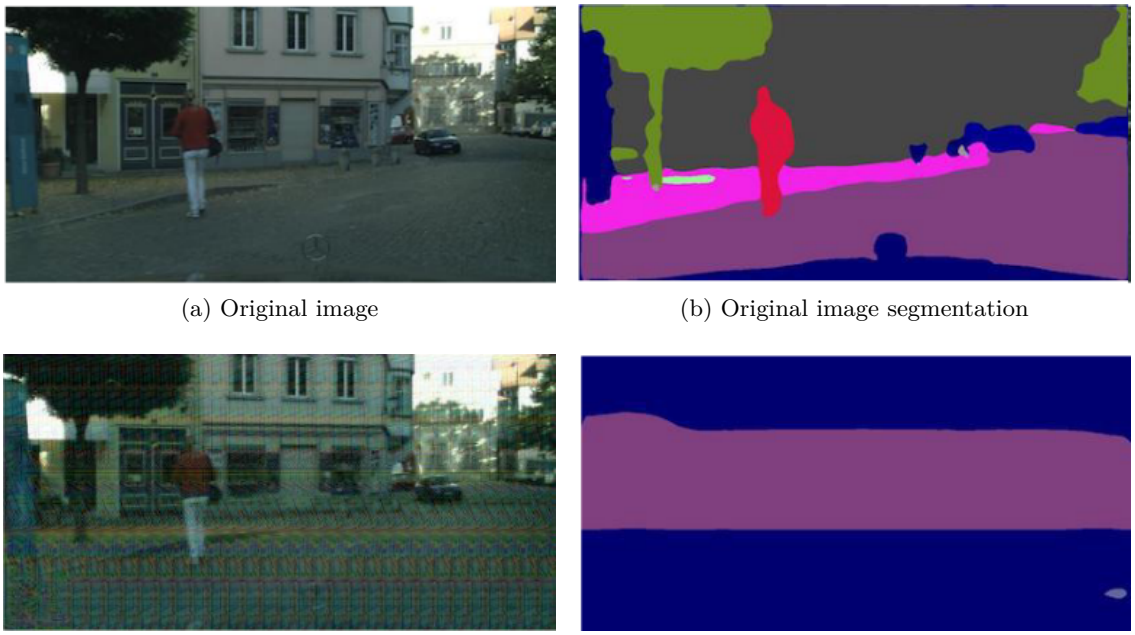
* Universal image filters can be created in advance, making the speed very fast

Figure 2.5: Comparison of universal and image-dependent perturbation

From the table we can see that both types of attack can be used, but due to the simplicity of universal image perturbation attacks it has become the more popular method of attack. The ability to launch the attack in real-time turned out to be the advantage that not many of the attacks I talk about in this thesis have.

### Conclusion

In this section I compared the various types of attacks called adversarial filter attacks. I looked at various benefits of using these methods and also their possible drawbacks. The quasi-imperceptibility and ability to commence black-box attack makes this category of attacks one of the most popular.



(a) Original image

(b) Original image segmentation

(c) Perturbed image

(d) Perturbed image segmentation

Figure 2.6: Filter attack can even fool semantic segmentation [5]

## 2.2 N-pixel-attacks

N-pixel-attacks is the second category of adversarial perturbation attacks. It is one of the most basic attack types mentioned in this thesis and is therefore ideal starting point for many beginners in computer vision fooling. Both targeted and untargeted attacks can be done with N-pixel-attacks and multiple well written works can be found on this topic. The general idea of this type of attack is very similar to adversarial filter attacks, but there are some major differences putting this attack type into its own category.

N-pixel attacks care more about the amount of pixels changed rather than overall quasi-imperceptibility, and when quasi-imperceptibility is achieved it is most likely a side effect of successful attack. Since we are attacking just a few pixels, sometimes just one, this attack has better success with low resolution images. Although even single-pixel-attack can work on 124x124 pixel resolution, making it just about 0.0065% change in the image, the success potential of attacks drops very quickly with increasing resolution and decreasing number of pixels changed. The change within the image can be either expressed in percentage change within the image itself or the amount of pixels changed in image of given resolution. Transferring the change into percentage marked as $\delta$ uses following formula, with N meaning the total number of pixels changed, and W and H marking the width and height of the image in pixels:

$$\frac{N}{W * H} * 100 = \delta \tag{2.1}$$

Due to the difference in complexity between n-pixel-attacks where n is equal to one and where n is greater than one, a subcategory of n-pixel-attack has been created called one-pixel-attack. This subcategory functions in the same manner as other n-pixel-attacks, but has some interesting properties.

### One-pixel-attacks

During one-pixel-attack only one pixel is allowed to change. This pixel is usually found using various methods of machine learning, with most scientific papers describing genetic algorithms, namely differential evolution, to be the most optimal [7][9]. The main benefit of using differential evolution is the easy switch between minimalization needed for untargeted attack and maximalization needed for targeted attack.

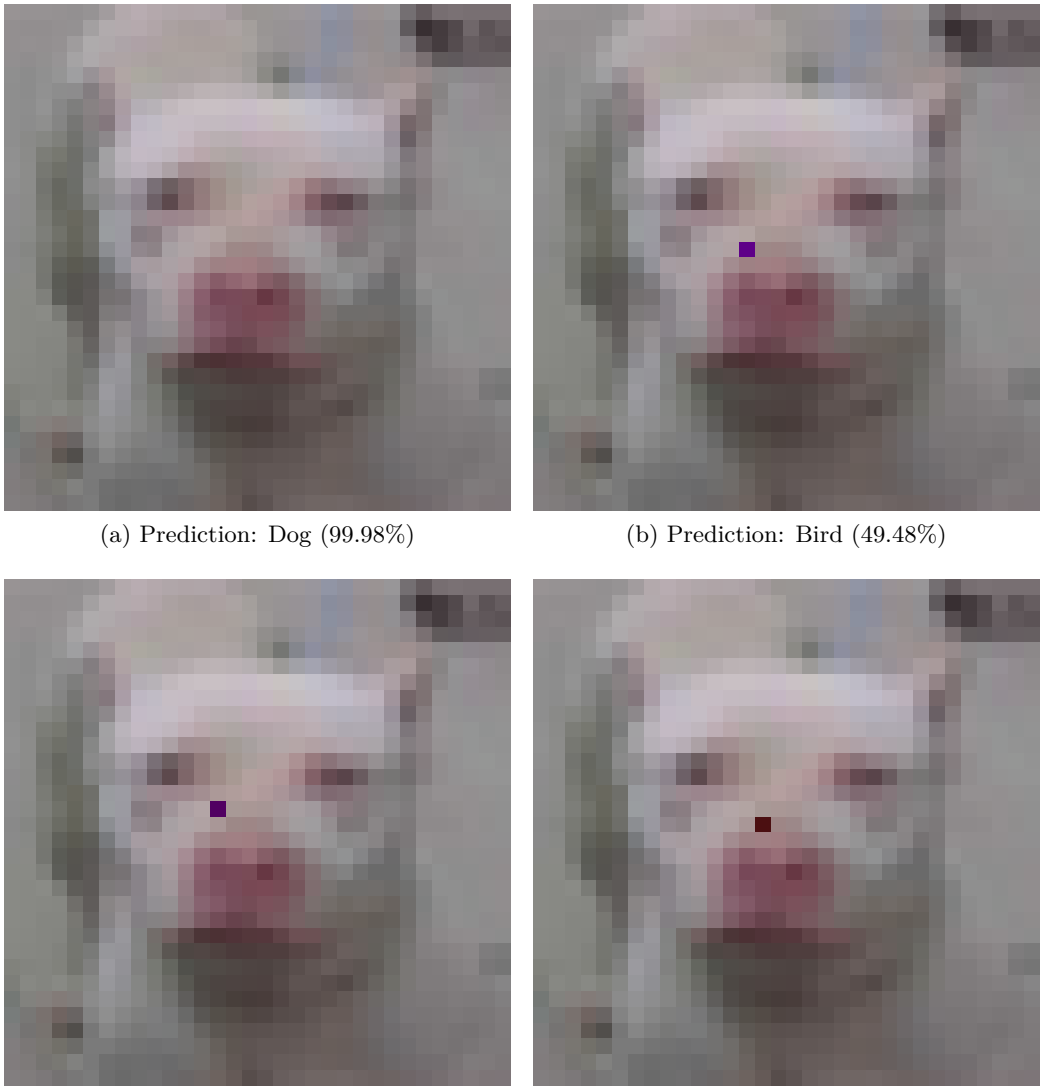
With differential evolution in place, the attack itself is done using array of 2 values marking position and 3 values marking red, green and blue values respectively.

$$[x, y, r, g, b] \tag{2.2}$$

With this array and a simple function that can change one pixel within the image with the values from this array, attack can be attempted. This attack operates as a semi-black-box attack, because along with the predicted class we are receiving information about classification confidence. If untargeted attack is applied to image, the differential evolution is trying to minimize confidence of the original class and is searching for a pixel that would decrease the confidence the most. On the other hand, when targeted attack is attempted, the function is trying to maximize confidence into the class that we are targeting. Both the success chance of the attack and the resulting class when untargeted attack is chosen are dependent on many factors. These factors are mainly the confidence and class of initial prediction, the initial randomization of arrays chosen for differential evolution, and the chosen target class during targeted attack. The model used for classification and the

amount of possible classes are also very important for the end result. Since only one-pixel-attack is considered at the moment, the percentage change within picture can only be influenced by the image resolution, which is also very important factor affecting the chance of success.

Some of these factors are randomized at the start of each attack, which causes many possible end results when the attack is run on one image multiple times. When possible the attack should be run multiple times, and best result should be chosen. This can make difference between the attack not succeeding and succeeding with over 90% result as the final confidence in different class.



(a) Prediction: Dog (99.98%)

(b) Prediction: Bird (49.48%)

(c) Prediction: Cat (81.47%)

(d) Prediction: Dog (44.48%)

Figure 2.7: Running the same attack multiple times yields vastly different results

### N-pixel-attack where n is greater than 1

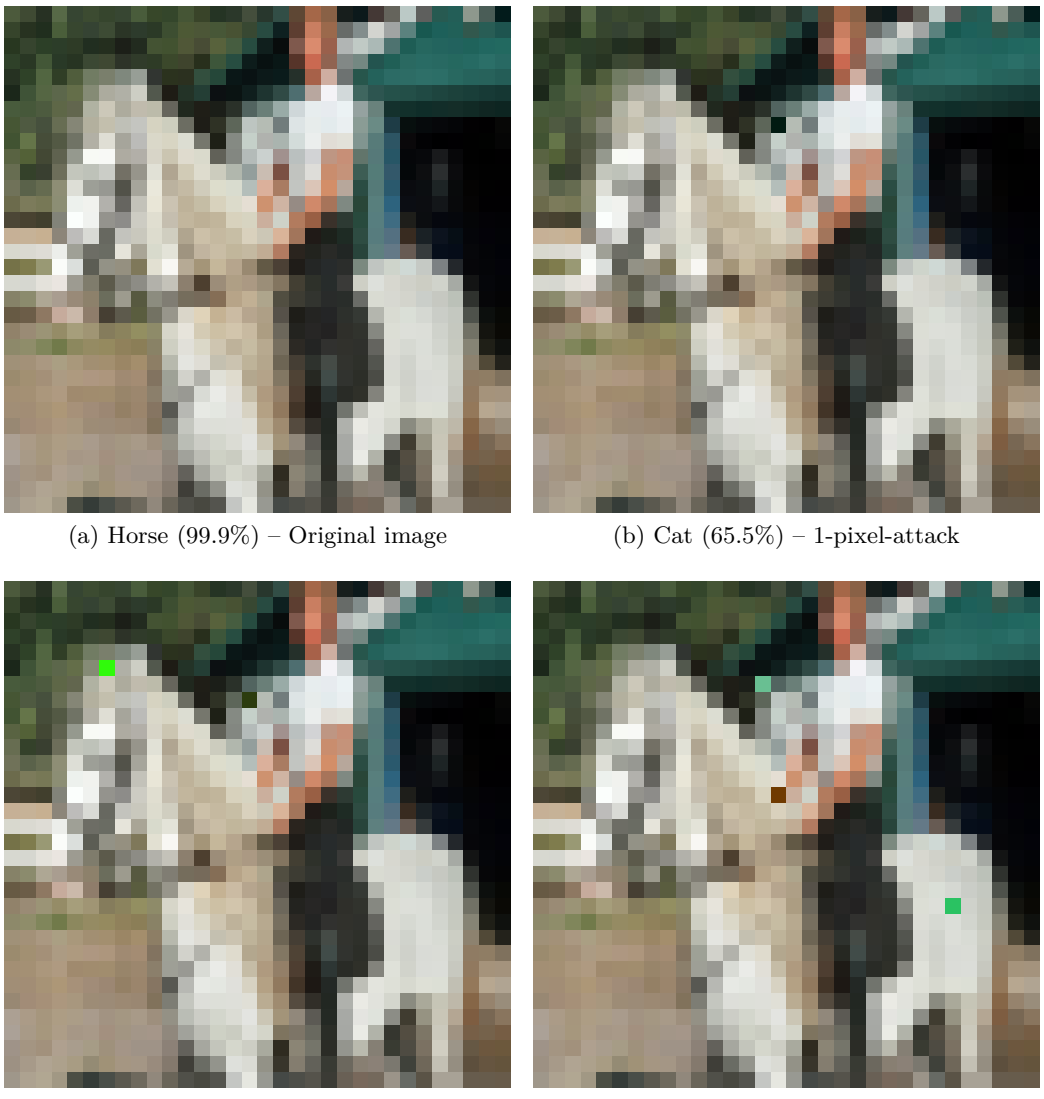
Increasing the complexity of 1-pixel-attack by increasing the number of pixels changed can bring us slightly improved chance of the attack succeeding. In that case we need to take into consideration the array we are optimizing using differential evolution. Where 5 value array was sufficient for 1-pixel-attack, the length of this array needs to be increased by 5 for each pixel we are adding, the array ends up in this format (the position of individual values can be switched around, this is just an example):

$$[x_1, y_1, r_1, g_1, b_1, x_2, y_2, r_2, g_2, b_2, \ldots, x_n, y_n, r_n, g_n, b_n] \tag{2.3}$$

It must be noted that increasing the length of the array that is being passed through differential evolution greatly affects the time it takes to generate the attack. It is therefore recommended not to go over 5 pixels when attempting n-pixel-attack, as it would be counterproductive for both the quasi-imperceptibility of the attack as well as its time cost. The program I have written as part of this thesis allows the user to choose any amount of pixels, but displays a warning when number above 10 is chosen.

Figure 2.8: Image of ostrich attacked using 100 pixels. Final prediction was airplane with confidence of over 98%. This image took over 20 minutes to generate on high-end laptop.

Another important note about n-pixel-attacks is the fact that while the probability of success increases with the amount of pixels changed, the final confidence may not be greater than attacks where less pixels are affected. This is often surprising and most likely explanation for this effect is the randomization of various starting factors as explained above. Often by simply re-running the attack multiple times we get a large range of final confidences from which we can pick the best one. This is especially the case of untargeted attack, where the minimalization of confidence into original class is taking place and the final confidence is not important as long as the attack succeeds.



(a) Horse (99.9%) – Original image

(b) Cat (65.5%) – 1-pixel-attack

(c) Cat (56.1%) – 2-pixel-attack

(d) Cat (50.5%) – 3-pixel-attack

Figure 2.9: Even with the amount of attacked pixels increasing, the final confidence with the same target class is decreasing

During some n-pixel-attacks certain pixels might affect the predictions more than the others. For convenience these pixels will be henceforth called „leader“ pixels. While changing non-leader pixel often does not affect the predicted class, any minor edit to the leader pixel often changes the prediction back into the original class. Typical attribute of leader pixel is that when the attack is attempted multiple times or with different amount of pixels, the leader pixel appears in similar position. This can be noticed in previous figure where in both 1-pixel-attack and 2-pixel-attack similar change occurs in the riders shoulder. Leader pixels are very important traits when detection of attacked image is considered, and I will be using them in future chapters.

### Conclusion

In this section I talked about second category of adversarial perturbation attacks. I took a close look at subcategory called one-pixel-attack and its features.

After that I compared one-pixel-attacks with their extended version called n-pixel-attacks. I explained how these attacks are executed and mentioned their strong and weak points. At the end, I talk about leader pixels in n-pixel-attacks, a very important attribute that we will meet with in future chapters.

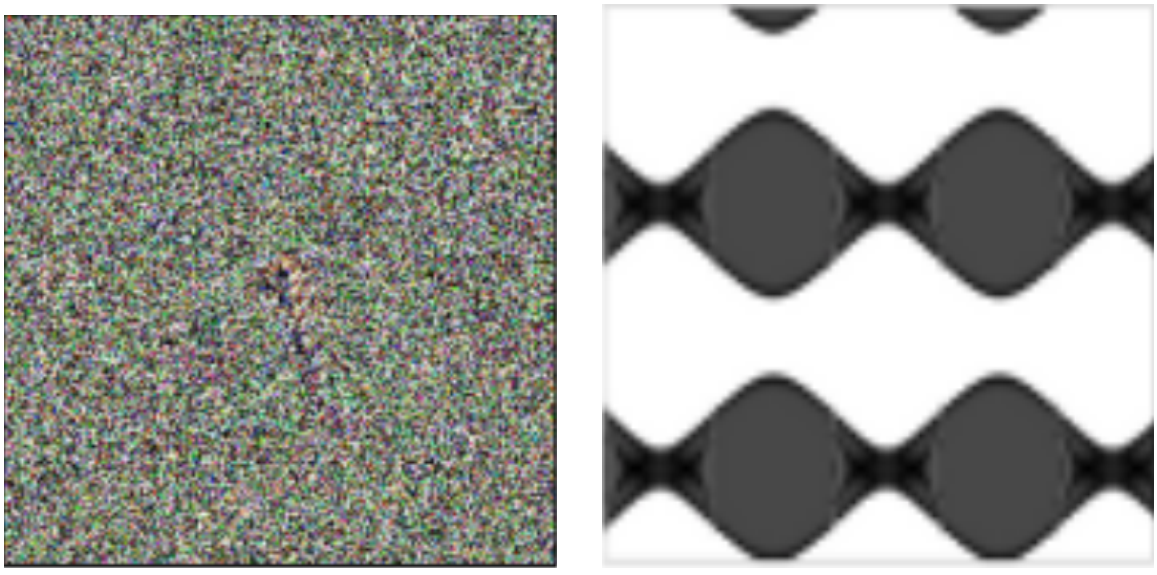
# Chapter 3

# High confidence image generation

Unlike the previous method of fooling computer vision, the high confidence image generation does not need any starting image. The point of this attack is to create image that is unrecognizable to human but computer can see certain object with extremely high confidence – often above 99%.

Strangely enough, this attack can be deployed as both targeted and untargeted attack. The untargeted attack is very rarely used for anything, because like every untargeted attack, we cannot determine the final prediction. In this case, the final images are used mostly for art related purposes, even being accepted to art exhibition [1].

Similarly to adversarial perturbation attacks, high confidence image generation can be done using two main methods. One of them creates completely unrecognizable images to human eyes which look like static or randomized dots. The second method uses compositional pattern-producing network (CPPN) [10] together with the image generation, creating more distinctive patterns, sometimes partially recognizable to human eyes. In this chapter I will take a look at both of these methods along with interesting discovery that I have accidentally reached during the programming of n-pixel-attack.



(a) Cheetah (b) Sunglasses

Figure 3.1: Comparison between CPPN unassisted and assisted network [1]

## 3.1 Evolutionary algorithm method

As explained above, first of the methods generates images that look like no real life object and still get classified as one with extremely high confidence. These images often end up looking like static noise. The interesting property of this noise is that very often by looking at it for a while certain outline starts to appear.

The process of generating such images is very similar to the one of n-pixel-attack. We use evolutionary algorithm and either try to maximize confidence of any random class (untargeted attack) or maximize the confidence of certain class (targeted attack) by changing the initial uniform background.

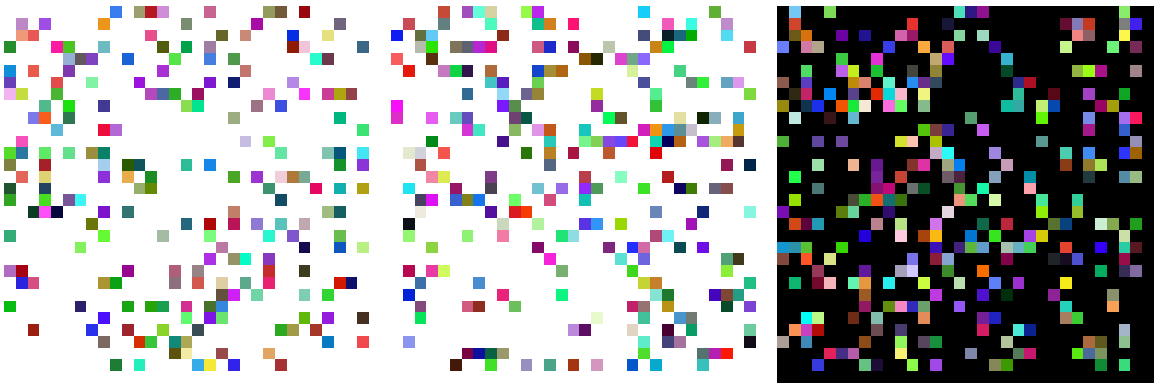
While researching this topic, I experimented with my program that allows user to commence n-pixel-attack on any image. If we choose high enough number of pixels, we can simulate the process of image generation. I named this method hijacking n-pixel-attack, because we are using it to simulate completely different method and type of attack it was originally designed for.

### Hijacking N-pixel-attack

When using n-pixel-attack to simulate the process of image generation, one must be prepared for it to take a while. As initial image any image of uniform background must be used, usually white or black. This image is then passed through as normal n-pixel-attack with the n being very high number. It is best to choose the amount of pixels to be changed to be at least 20% of the original image resolution, otherwise the image will mostly be the color of the background.

The attack itself can be run as both targeted and untargeted with similar results as generating image using targeted or untargeted method. Only major difference is that the untargeted attack cannot generate the image of the original class. For example, uniform background that was originally classified with highest confidence as bird cannot be generated as image of bird during untargeted attack.

Using n-pixel-attack is unfortunately very ineffective and time consuming, therefore I would not recommend using it for anything other than simple experiments.



(a) Frog (250 pixel attack) (b) Car (300 pixel attack) (c) Cat (350 pixel attack)

Figure 3.2: Hijacked n-pixel-attack used for generating high confidence prediction images

## 3.2 CPPN-assisted method

The only difference between the processes of creating image using purely evolutionary algorithms and supporting it using compositional pattern-producing network is the output image that these methods generate.

Supplementing previously described method by the option of generating more patterns within the image brings us no additional benefit to recognizability by human eyes. On the contrary, these images can be more easily recognized due to the fact that shapes and colors often at least figuratively resemble the generated class.

Generating such images has proven that evolutionary algorithms are able to generate images that both neural networks and humans are able to classify [11].

Figure 3.3: Familiar looking images created using CPPN [11]

### Conclusion

In this chapter I compared two methods of generating images that can fool image classification. Both of the methods explained can be used to better understand the inner workings of machine learning and allow us to create improved versions of such algorithms.

I also talked about my experiments with using n-pixel-attack to create similar looking images. These experiments proved to be quite successful, proving that n-pixel-attack can be used to simulate basic image generation that uses evolutionary algorithm.

# Chapter 4

# Unusual methods of fooling computer vision algorithms

In this chapter I will compare various methods that are not in any of the previous categories. These methods are interesting in the way they are able to fool computer vision using very unusual approaches.

I will first take a look at somewhat different kind of image transformation, where instead of applying filter over image or changing its pixels, the image itself is rotated against the background [12].

Next thing I will talk about are adversarial patches. It is the name given to very specific method of attack involving generated image that can be placed next to normal object to offset real-time classification [13].

## 4.1 Rotation and translation attack

As the name suggests, this method works by rotating the original image by certain degree and using minor translations. It shows the need for creating more robust neural network in the future, as this attack could naturally occur by accident without the need for attacker.

One important goal of creating images using this attack is to keep visual similarity between the attacked and original image. Since this is issue of individual human perception, concrete definition for visual similarity does not exist. This is applies mainly to rotations, as image rotated 180°does not keep much of its visual similarity.


(a) Revolver (b) Mousetrap

Figure 4.1: Rotation against black background is able to fool classifier [12]

## 4.2 Adversarial patch

This method of attack is very similar to universal adversarial filter method, but small differences put this attack into different category. It works as a targeted attack, creation of untargeted adversarial patch has not been tested yet. The created patch can be put next to most real-life objects, and after the image of both the real object and the patch is classified, it returns the target class of the adversarial patch (successful attack).

The benefit is the possibility to create multiple adversarial patches in advance and then just fooling the classifiers in real-time. The main drawback is the localization of the attack makes it easier for some methods like gradient smoothing [14] to defend against adversarial patch attacks.

Figure 4.2: Example of adversarial patch [13]

### Conclusion

In this chapter I took a look at two methods of fooling computer vision algorithms which are not that common. These are not only interesting for their novelty, but mainly due to the fact that they might either easily occur naturally or be applied as very simple attack.

The rotation and translation attack is one of the reasons why autonomous car industry should focus on computer vision fooling algorithms. Although this method has extremely low success rate, it only takes one car to mistake traffic sign for a tree to cause a major accident.

The Adversarial patch attack method has many similarities to the universal filter attack, especially due to its capability to be prepared in advance to the actual attack. That is very strong ability for computer vision fooling algorithms that many do not have.

# Chapter 5

# Detection and defense against computer vision fooling

In previous chapters I have shown the potential attacks that can fool computer vision. It is therefore important to talk about ways to defend against these attack or at least be able to detect whether the image has already been attacked.

For this, we use the knowledge from previous chapters regarding the ways the attack methods operate, as well as simple methods that have been tried with varying amount of success.

## 5.1 Adversarial filters

Defense against adversarial attacks can be done in multiple ways, but only very few of them give results sufficient for practical use. Generally, these defenses either work by increasing the robustness of neural network by providing adversarial examples during training [15], or by parsing the images before classifying them. The benefit of the first type of defense is that although it takes longer time during the training of the neural network due to increased amount of training images, once the neural network is trained, the speed of the neural network is the same to the one not trained on adversarial examples.

Another group defense methods work by preprocessing the image before passing it through the neural network. One method from this group that has been recorded to have adequate results is called gradient smoothing [14]. The benefit of this method is that we can detect the more uncommon type of attacks such as the LaVAN attack [16] or adversarial patch attack. The main drawback is the needed pre-processing that can slow down the detection and classification speed.

## 5.2 N-pixel-attacks

Similarly to when N-pixel-attack were described few chapters back, we will divide the n-pixel-attack to two subcategories – one-pixel-attack and n-pixel-attack where n is greater than one. Both of them have minor differences that make the methods of detection slightly easier for 1-pixel-attacks.

### 5.2.1 1-pixel-attacks

To detect 1-pixel-attack in the image we can apply 2 basic approaches. Firstly, we can try to use machine learning to teach neural network to find suspicious looking pixels and try to correct them. The neural network should try to detect few of these pixels and after correcting these pixels (for example by averaging their color with the one of their neighbors) passing them through image classifier. If the confidence of the edited image prediction changes by large amount, there is a chance that the image has been attacked.

Second approach takes advantage of the fact that n-pixel-attacks in general are done on images of lower resolutions. It is therefore not as time consuming to try and use bruteforce for detecting the pixel attack. We can either use very basic bruteforce attack without any complex inner logic which will walk through every pixel and try to average it similarly to the first approach, or we can build slightly improved version which tries to order the pixels it will try to walk through before commencing the bruteforce attempt. This can significantly improve the speed when going through a large batch of potentially attacked images.

As is often the case with computer vision fooling algorithms, the detection of attack is much easier than defense. If we want to ensure that our neural network is harder to fool, we can use some of the methods described by Jiawei Su, Danilo Vasconcellos Vargas and Kouichi Sakurai [7]. One of the methods to improve neural networks robustness is based on preprocessing image before passing it through classifier, for example by squeezing color bits or utilizing basic noise reduction. The drawback of using this method is the added layer of preprocessing that slows down the speed of the classifier.

### 5.2.2 N-pixel-attacks with n greater than 1

For detection of attack in images where greater number of pixels has been changed we will take into consideration the weaknesses caused by the existence of leader pixels described in the adversarial perturbation attacks chapter.

For the most part, same bruteforce approach can be used on images attacked with smaller amount of pixels. This is often sufficient as the n-pixel-attacks are mostly done with less than 10 pixels. When the image is attacked using more pixels than that, some improved methods are needed, because the amount of leader pixels changes together with the attacked pixels and bruteforcing multiple pixels gives us exponential complexity. One way to solve this issue is by searching for the most suspicious pixels similarly to the approach described in the detection of 1-pixel-attack but using this method to change multiple pixels at the same time instead of just a single pixel.

Other approaches that work well with 1-pixel-attack often work as well, but it is more difficult for both detection and defense to work the more pixels are attacked. The extreme case where the n-pixel-attack has been hijacked to generate high confidence image from scratch is the perfect example of such attack. In this case, the methods for detection and defense are described in the following section.

## 5.3 High confidence image generation

This method is possibly the most difficult method to detect and defend against from all the other methods described in this thesis. The main problem is its high transferability onto other neural networks and almost infinite possible combinations of generated images.

The one method that has been tried is the same as the one used with most other fooling methods. It consists of training the network on adversarial examples to teach the network how to recognize fooling attempts [1]. Unfortunately, this method did not work well for this attack and I have not found any other mentions of attempting to detect images generated in this attack.

### Conclusion

In this chapter I talked about methods of detecting and defending against attacks that have been described in previous chapters. I found out that methods of actively defending against various attack are not as simple as they might seem, and often the methods that make sense in theory work with only minimal results or not at all.

I consider defense and detection of attacks a very important field for the future of computer vision, especially for the military and automobile purposes. Due to many of the attacks working as black-box attacks, there could be many undiscovered methods that could extremely simplify the defense and detection, they just need to be discovered first.

# Chapter 6

# Implementation

As part of this thesis I have written a python3 program. This program allows users to commence attacks from the adversarial perturbation category, namely various kinds of n-pixel-attacks. This program uses CLI mixed with basic GUI to display results of user-defined attacks. Another feature of my program is the ability to detect various n-pixel-attacks in the image using some methods that I have discovered and implemented. Lastly, my program allows user to measure statistics of various attacks.

## 6.1 Used software and libraries

As was already mentioned in the introduction, my program was written for python3 language. For the program to run without any issues, some libraries need to be added first. One of those libraries is Keras [17], which is a python deep-learning library. Keras was chosen over similar libraries like PyTorch [18] due to the fact that I found it easier library to work with and it was easier for me to find pretrained models for CIFAR-10 [4].

In addition to Keras few more libraries are needed for the program to run. Namely, SciPy [19] library is needed for its implementation of differential evolution and NumPy [20] library by the same creators for its mathematical functions. To display the result image if program is run with verbose option, python library matplotlib [21] is needed.

## 6.2 Program functions

The final program has been divided into multiple subprograms, with specific ones for attack, defense and statistics collection. Each one of them can be ran by itself as standalone program.

### 6.2.1 1-pixel and n-pixel attacks

Program allows user to commence attacks from the n-pixel-attack category on any image in PNG or JPEG format. User can choose between untargeted and targeted attack using the „-t“ argument and also choose the amount of pixels to be changed using the „-n“ argument with the number of pixels. Additional optional arguments are „-v“ for verbose output, which prints more information during the attack and also shows the final result image, and „-s“ argument for printing output easier for collecting statistics about the results.

The program uses the SciPy [19] implementation of differential evolution – one of the genetic algorithms used for optimization [22]. This algorithms is used in two ways depending on the optional argument choosing whether the attack will be targeted or untargeted.

In the case of untargeted attack the goal of the program is to minimize confidence of original class. This is done by searching for the pixel or pixels which lowers the prediction the most. The cut-off point I chose was the confidence falling bellow 50%, but this too can be easily changed. This cut-off point is only taken into consideration when callback function of the differential evolution is called, therefore the successful attacks usually reach far better results, sometimes decreasing the confidence bellow 0,1%.

Targeted attack on the other hand is trying to maximize confidence of any class chosen. This can be used to either improve the confidence of already correct prediction, or to maximize confidence into completely different class. The SciPy implementation of differential evolution is only able to be used for minimalization, it is therefore needed for the targeted attack to try to decrease the number (1 - target class confidence) instead. Although targeted attack can be used to change prediction of any class into any other class, for best results it is recommended to use similar looking combinations. For example, the combination of bird-airplane will have much higher chance of success over combination like frog-truck, where the amount of pixels to be changed could be increased or the attack could be run multiple times to make the attack success more probable.

### N-pixel-attack hijacking

During the experimentation with n-pixel-attacks I have discovered interesting method to generate high confidence prediction images. I named this method n-pixel-attack hijacking due to the fact that we are using somewhat unrelated method for completely different purpose.

This method is not practical for high confidence image generation because of its slow run-time, but can reach confidence results comparable to methods dedicated for creating high confidence prediction images. While experimenting with n-pixel-attack hijacking, I have discovered that its better to run multiple n-pixel-attack where the n is lower number over just a single attack with very high number of pixels. Although some of the pixels will be overwritten, the overall time consumed per pixel will be lower.

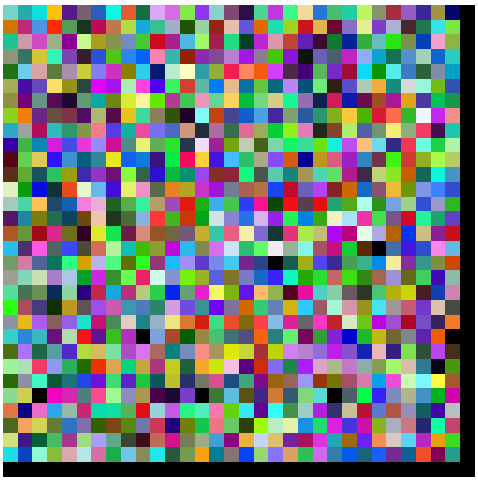


Figure 6.1: Hijacked n-pixel-attack generated image. Prediction: automobile (99.8%)

### 6.2.2 Detection of n-pixel-attack

Most methods described in science journals involve increase of robustness [23] of the neural network, either by retraining it using adversarial examples or older methods such as the distillation defense [24]. I have decided to attempt the detection of n-pixel-attack in the image using simpler methods that work well on consumer hardware.

The n-pixel-attack detecting program written as part of this thesis uses two methods to try to find the pixel or pixels that were used to attack the image and find out the original class. Both of these methods are somewhat similar in their nature, with the second method being improved version of the first method.

The first of these methods works on the basis of brute-force searching for the attacked pixel. The method works through each of the pixels in the image and changes the pixel into the average of its four neighbors. It then passes the newly created image through classifier and tries to see whether the confidence of new prediction has any significant change. This method works well even if the image was attacked with more than a single pixel, because the attack often loses significant prediction confidence even when one of the pixels is found.

The second method improves on the speed of the first method. This method is called suspicious pixel search and can be ran with the argument „-s“. This method does not simply walk through each pixel in the image until the correct one is found, but firstly does some pre-sorting of the most suspicious pixels found in the image. The level of suspiciousness of the image is decided by calculating the average of its neighboring pixels colors. This average is then compared to the color of each pixel and the difference is saved into array. This array is then sorted from the most different ones and the pixels are compared in the order from the most suspicious until the pixel is found or all of the pixels have been checked. Although this method is slightly slower on large batches where no attacked images can be found, the speed improvement on mixed batches was rather significant. Often the pixel is found in the first few of the suspicious pixels, whereas in the brute-force method the pixel is found on the average afer testing half of the pixels.

### 6.2.3 Statistics collector and various useful tools

Last part of my program is simplified version of pixel-attack software that allows user to run the attack on larger batches of images while collecting basic statistics about the attack and saving the results of successful attacks.

Most of the other subprograms written as part of this thesis also allow the user to use them as standalone programs. For example, the subprogram written for changing the input file into CIFAR-10 friendly format can be ran with various arguments to convert concrete file or directory. Similarly, there are also programs for fetching images from CIFAR-10 database and for displaying result images in figure form together with description.

## 6.3 Experiments

The experiments with the 1-pixel and n-pixel attacks have been done on images from the keras library dataset. Firstly, the images were downloaded as batches of hundreds of images, using one of the programs written as part of this thesis. Then, these images have been run through another program – statistics collector. This program collected the results of various attacks.

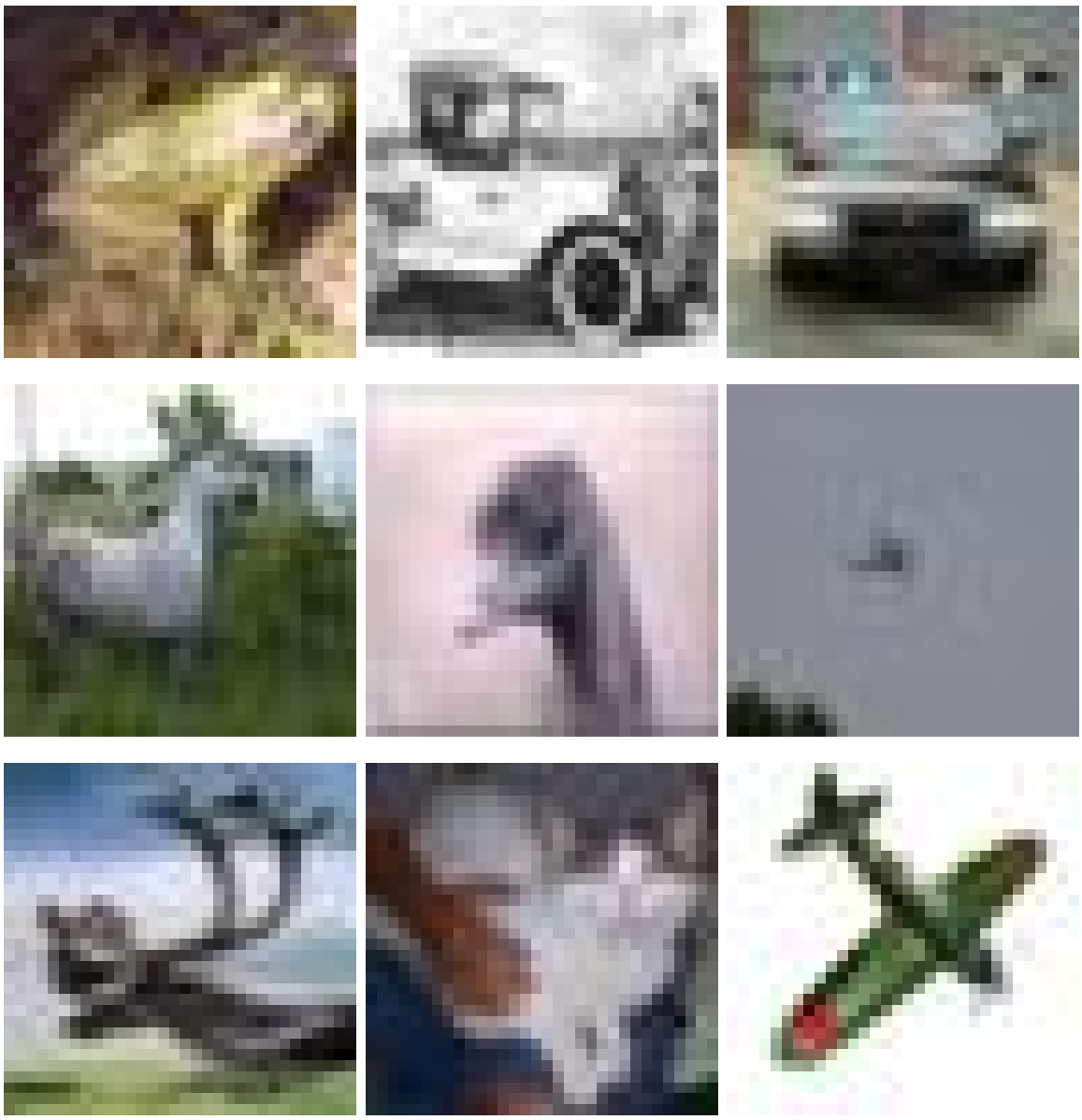


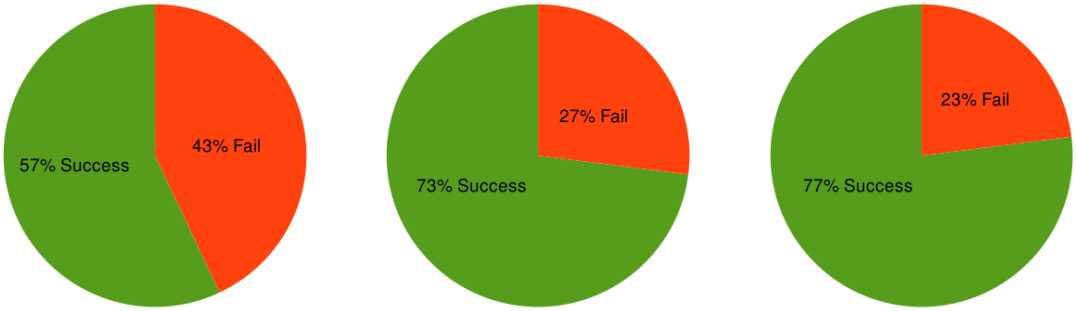
Figure 6.2: Sample images from batch 1-100 of the CIFAR-10 database

Next series of experiments has been done regarding the ability to detect n-pixel-attack in the image using both the bruteforce method and the suspicious pixel search method. This has proven to be very successful for both of these methods as will be later seen in the results.

### 6.3.1 N-pixel-attack results

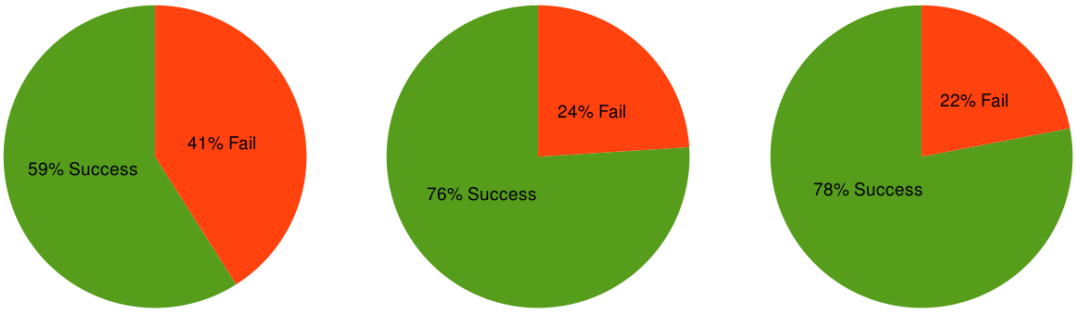
The attacks have first been done on first batch from the keras library dataset from the first 100 images, or batch 1-100. The attacks attempted have been untargeted, and only run once on each image. First, the attack has been done as 1-pixel-attack, then 2-pixel-attack and lastly 3-pixel-attack.

I studied the comparison of success rate between these three attacks and found direct correlation between the amount of pixels attacked and the success rate of attack used. For comparison, the same test has been done one the batch of another 100 cifar-10 friendly images, also called batch 101-200. The results were very similar.




(a) 1-pixel-attack (b) 2-pixel-attack (c) 3-pixel-attack

Figure 6.3: Untargeted attack results, first batch of 100 images



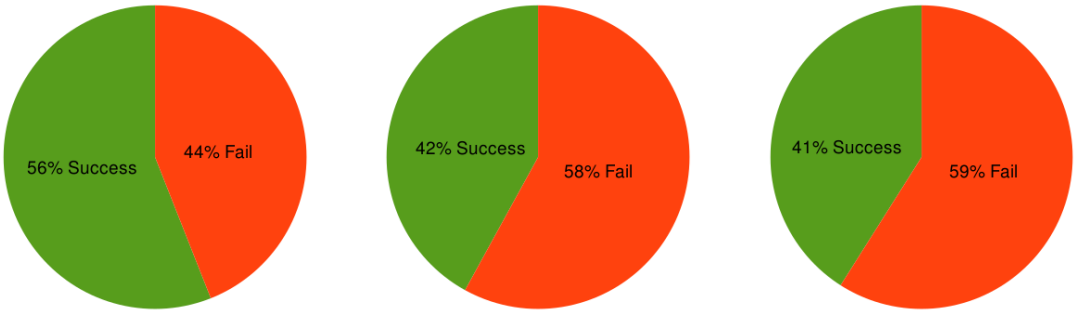

(a) 1-pixel-attack (b) 2-pixel-attack (c) 3-pixel-attack

Figure 6.4: Untargeted attack results, second batch of 100 images

From these results we can see large jump between the success rate of 1-pixel-attack and 2-pixel-attack, but there is not such a great improvement when one more pixel is added. The smaller jump between two attacks with higher amount of pixels changed has been already recorded with similar results [25], but I have not found any mention to the reason of such a discrepancy between the success rate of 1-pixel-attacks and 2-pixel-attacks. It is possible this difference is caused by the consideration of successful attack being determined by changing the class with at least 50% confidence and by decreasing this number we could also decrease these jumps.

### 6.3.2 N-pixel-attack detection results

N-pixel-attack detection has been attempted on various images attacked by one-pixel-attack to three-pixel-attacks. Since not all of the images have been successfully attacked, the batches for detection are smaller in size.




(a) 1-pixel-attack (b) 2-pixel-attack (c) 3-pixel-attack

Figure 6.5: Detection rate of successful attacks from second batch of images

Since both the brute-force and the suspicious-pixel-search detection methods try each pixel in case of non-attacked image or unsuccessful detection, the success rate for both of them is exactly the same. Only difference is the speed with which these methods operate depending on what percentage of attacked images is in the batch.

Given the simplicity of the methods used for detection compared to commonly used methods [26], the success rate of these methods is still sufficient for some uses. Main benefit of this method is the possible speed that can be reached when the processes are parallelized and every single pixel is being checked simultaneously. In that case, the purely brute-force method is preferred as using suspicious-pixel-search brings no benefit when all pixels are checked regardless of their level of suspicion.

(a) 1-pixel-attack, Deer (b) 2-pixel-attack, Dog (c) 3-pixel-attack, Cat

Figure 6.6: Successful attacks where detection was possible

(a) 1-pixel-attack, Cat (b) 2-pixel-attack, Frog (c) 3-pixel-attack, Bird

Figure 6.7: Successful attacks where detection was not possible

I have found no major difference between the detectable and undetectable attacks. Except for the attacks where less pixels were used, where it is easier to find the one pixel that changes the prediction confidence by major amount. Both of my methods can be used on n-pixel-attacks where n is higher than 1, but the results slowly diminish with increasing amount of pixels changed. If that happens, the detection program can be recalibrated to consider lower change in prediction confidence as successful detection.

Figure 6.8: Successful detection of 5-pixel-attack (deer), original prediction: Dog (90.19%)

# Chapter 7

# Conclusion

In this thesis I have described various methods of fooling algorithms of computer vision. I studied various methods ranging from the very simple ones to the more advanced and compared their usefulness in using them to attack various neural networks.

I also studied methods of detecting or defending against such attacks. Most of the attacks mentioned in this thesis have multiple ways of defending against them or at least detecting them if they already happened. These methods are explained and compared in one of the chapters.

As part of this thesis I have written a python program allowing user to attack any image in .jpg or .png format using various n-pixel-attack types described in this thesis. User can also collect statistics from his attacks to study them later. This program also allows user to try detecting the attack in either attacked or not attacked image and collect statistics from these attempts. The part of the program allowing user to detect attack uses two methods that I have designed and programmed that turned out to be quite efficient.

The results of my experiments have shown that even well trained neural network can be easily fooled with simple changes that are almost imperceptible to human eyes, while the detection and defense against such attacks is much more complex or in some cases even impossible. Many of the attacks described in this thesis can be used for evil causes like fooling of the self driving car recognition software or online image filters.

While the focus on training the neural networks to be more accurate has risen in recent years, even latest neural networks can be easily fooled with simple programs such as the one written as part of this thesis. Luckily, as of today there has not been any major accident involving intentionally fooled computer vision, but since this technology is appearing in our lives more and more often, this might be just a question of when will the first attack happen and what will be the consequences.

# Bibliography


[1] Nguyen, A.; Yosinski, J.; Clune, J.: Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[2] Eckstein, M. P.; Koehler, K.; Welbourne, L. E.; et al.: Humans, but Not Deep Neural Networks, Often Miss Giant Targets in Scenes. September 2017.

[3] Akhtar, N.; Mian, A.: Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *CoRR*. vol. abs/1801.00553. 2018.

[4] Krizhevsky, A.: The CIFAR-10 Dataset. [online], cit. [29. 1. 2019]. Retrieved from: https://www.cs.toronto.edu/~kriz/cifar.html

[5] Poursaeed, O.; Katsman, I.; Gao, B.; et al.: Generative Adversarial Perturbations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[6] Ruan, Y.; Dai, J.: TwinNet: A Double Sub-Network Framework for Detecting Universal Adversarial Perturbations. *Future Internet*. vol. 10. 03 2018: page 26. doi:10.3390/fi10030026.

[7] Su, J.; Vargas, D. V.; Sakurai, K.: One pixel attack for fooling deep neural networks. *CoRR*. vol. abs/1710.08864. 2017.

[8] Szegedy, C.; Zaremba, W.; Sutskever, I.; et al.: Intriguing properties of neural networks. *CoRR*. vol. abs/1312.6199. 2013.

[9] Storn, R.; Price, K.: Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces. *Journal of Global Optimization*. vol. 23. 01 1995.

[10] Stanley, K. O.: Compositional Pattern Producing Networks: A Novel Abstraction of Development. *Genetic Programming and Evolvable Machines*. vol. 8, no. 2. June 2007. ISSN 1389-2576.

[11] Auerbach, J.: Automated Evolution of Interesting Images. 03 2019.

[12] Engstrom, L.; Tran, B.; Tsipras, D.; et al.: A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. 2019.

[13] Brown, T. B.; Mané, D.; Roy, A.; et al.: Adversarial Patch. *CoRR*. vol. abs/1712.09665. 2017.

[14] Naseer, M.; Khan, S.; Porikli, F.: Local Gradients Smoothing: Defense against localized adversarial attacks. *CoRR*. vol. abs/1807.01216. 2018.

[15] Zantedeschi, V.; Nicolae, M.; Rawat, A.: Efficient Defenses Against Adversarial Attacks. *CoRR*. vol. abs/1707.06728. 2017.

[16] Karmon, D.; Zoran, D.; Goldberg, Y.: LaVAN: Localized and Visible Adversarial Noise. In *Proceedings of the 35th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, vol. 80, edited by J. Dy; A. Krause. Stockholmsmässan, Stockholm Sweden: PMLR. 10–15 Jul 2018. pp. 2507–2515.

[17] *Keras: The Python Deep Learning Library*. [Online; visited 15.01.2019]. Retrieved from: https://keras.io

[18] *Official PyTorch website*. [Online; visited 17.03.2019]. Retrieved from: https://pytorch.org

[19] *SciPy library*. [Online; visited 23.03.2019]. Retrieved from: https://scipy.org

[20] *NumPy library*. [Online; visited 23.03.2019]. Retrieved from: https://numpy.org

[21] *Matplotlib library*. [Online; visited 23.03.2019]. Retrieved from: https://matplotlib.org

[22] *Differential evolution*. [Online; visited 6.04.2019]. Retrieved from: http://www1.icsi.berkeley.edu/~storn/code.html

[23] Carlini, N.; Wagner, D. A.: Towards Evaluating the Robustness of Neural Networks. *CoRR*. vol. abs/1608.04644. 2016.

[24] Papernot, N.; McDaniel, P. D.; Wu, X.; et al.: Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. *CoRR*. vol. abs/1511.04508. 2015.

[25] Su, J.; Vargas, D. V.; Sakurai, K.: Attacking Convolutional Neural Network using Differential Evolution. *CoRR*. vol. abs/1804.07062. 2018.

[26] Yuan, X.; He, P.; Zhu, Q.; et al.: Adversarial Examples: Attacks and Defenses for Deep Learning. *CoRR*. vol. abs/1712.07107. 2017.

# Appendices

# Appendix A

## Contents of attached CD

### Python programs

attack.py – Program for various n-pixel-attacks
defend.py – Program for detection of n-pixel-attacks
stat.py – Program for collecting attack statistics
figure_creator.py – Program for creation of figures from images
converter.py – Program for converting images into CIFAR friendly format
dataset_getter.py – Program for getting images from the Keras dataset

### Folders

models – Folder containing ResNet model used for fooling
dataset – Images from Keras dataset
successful_attack – Images of successful attacks