Evolutionary Algorithm-based Adversarial Attacks Against Image Classification Convolutional Neural Networks

Raluca Ioana Chitic

University of Luxembourg, Department of Computer Science

Date of defence

Acknowledgements

I am grateful to my supervisor for the very nice professional relationship we have created, for being actively involved throughout all stages of my doctoral studies, both guiding me academically and giving practical support in all aspects of the PhD. I also have deep appreciation for my colleague Nicolas Bernard, whose vast science knowledge always led to interesting conversations, and whom I could always count on to receive valuable feedback. In addition, I thank my colleague Ali Topal, with whom I have had numerous insightful discussions and enjoyed a very pleasant collaboration. I would also like to thank my other co-authors Nathan Deridder and Elmir Avdusinovic, whose quality work led to some of the results presented in this thesis. Additionally, I appreciate the continuous useful feedback I have received from my CET members Ulrich Sorger and Pascal Bouvry, as well as the positive atmosphere they created at all evaluations. Moreover, I thank my jury members Louis Goubin and Jean-Charles Pomerol for expressing an interest in this work and accepting to be part of the PhD evaluation. Last but not least, I would like to express my gratitude to Speedy Graphito and to Bernard Utudjian for the provision of two artistic images used in the feasibility study of Chapter 5, Subsection 5.4, and for their interest in this work.

Abstract

The remarkable performance that Convolutional Neural Networks (CNNs) achieved in automatic image classification has led to their adoption in safety-critical scenarios such as autonomous cars, traffic control, manufacturing, medical devices and avionics. It is thus essential that CNNs can be trusted. However, a line of research has developed into the field of adversarial attacks, whose purpose is to expose CNNs' vulnerabilities. An adversarial attack consists of taking an original image $\mathcal{A}$ that is classified by a CNN $\mathcal{C}$ as belonging to a category c_a and modifying it to create a new, adversarial image $\mathcal{D}$, that is humanly indistinguishable from $\mathcal{A}$, and classified in a different category, potentially chosen in advance.

This thesis introduces a practical black-box adversarial attack based on an evolutionary algorithm (EA). We show that our attack is highly efficient for various attack scenarios performed on multiple CNNs trained on different datasets. Moreover, the attack is made robust to a large series of image filters. For a better understanding of the EA attack, we also analyze it from different perspectives such as noise frequency, transferability, behaviour at local regions and texture change, all while comparing it with the BIM white-box attack.

To summarize, this work shows that our EA-based attack is flexible, efficient and robust.

Contents

Chapter 1

Introduction

Trained Convolutional Neural Networks (CNNs) are one of the dominant tools for automatic object recognition [58]. Their success has led to their use in safety-critical scenarios, such as autonomous cars [20], traffic control, manufacturing, medical devices and avionics [50]. However, with such success comes high responsibility. It is therefore essential to rely on very robust CNNs. Recent research into CNN robustness has proven that these algorithms are actually vulnerable to slight perturbations in the input. This line of research has developed into the broad field of adversarial attacks (see [9] for a survey on this subject).

The ingredients of an adversarial attack are a given CNN $\mathcal{C}$, which is trained to classify images, and an ancestor image $\mathcal{A}$ that is classified by $\mathcal{C}$ as belonging to a category c_a. With these ingredients at hand, an adversarial attack consists in perturbing $\mathcal{A}$ to create an adversarial image $\mathcal{D}(\mathcal{A})$. The creation of a successful attack imposes two main requirements, namely that $\mathcal{C}$ classifies $\mathcal{D}(\mathcal{A})$ as a category c_d that differs from c_a, and that $\mathcal{D}(\mathcal{A})$ is humanly indistinguishable from $\mathcal{A}$. Although the latter requirement is not systematically met (see [55, 32, 46]), we respect this requirement in the present thesis.

A multitude of attacks have been implemented in prior work, with vastly differing strategies, assumptions and goals. One way of differentiating between attacks is based on the criterion of the adversary's knowledge. According to this, attacks can be sorted into white-box, gray-box, or black-box attacks [9].

The majority of existing attacks are white-box methods (such as [26], [36], [42]), meaning that they require complete information about the attacked CNN, such as its architecture and parameters. Although obtaining this information can be achieved in a research environment, it is unlikely in a real-life scenario, where the attacker only has access to the model's output.

Gray-box attacks are named as such due to the fact that they are a combination of white-box and black-box attack methods. More specifically, gray-box attacks begin by using other already established white-box attacks to create several adversarial images and by gathering the generated adversarial images into a dataset. Then, in the black-box part of the attack, the attacker no longer needs any information about the CNN, since the generation of further adversarial images is done by simply selecting points from the distribution of the above-created dataset. As the points are selected from a distribution of adversarial images, it is expected that the chosen points also represent images that are adversarial. Even if gray-box attacks are not knowledge-demanding at all stages, it is impossible to perform this type of attack without having all CNN information available. Said otherwise, the requirements of gray-box attacks are of a similar nature as those of white-box attacks.

In contrast, black-box attacks only require access to the CNN's output, which is an assumption that is closer to what one can expect in real life. Depending on the employed method, there are at least three possible approaches of circumnavigating the lack of knowledge available to the attacker.

One approach consists in using a second, surrogate CNN. Although the surrogate model's parameters and architecture cannot be the same as those of the CNN targeted by the attack, they perform the same task of image classification. The method begins by inputting several images to the targeted CNN and extracting their respective predicted output. All the images and their respective predictions are gathered into a dataset that reflects the targeted model's behaviour. The next step is to create the new, surrogate CNN and train it with the dataset obtained above. The purpose of this new CNN is to be a replica of the targeted CNN. Since the parameters of the newly designed CNN are known, it can be attacked through white-box methods. Finally, the adversarial examples generated for the replica model can be sent to the targeted CNN. This method counts on the transferability of adversarial examples, meaning that an adversarial image created for one CNN can also fool a different CNN [9]. Needless to mention, this method is not straight-forward, as it requires gathering a training dataset and training a different CNN.

Another black-box approach is illustrated by the Zeroth Order Optimization (ZOO) [10]. This type of attacks ultimately use white-box methods to create adversarial images. However, to use a white-box method, one must have access to the CNN parameters in order to calculate by how much the original image has to be perturbed to reach a certain loss in the CNN's predictions. In other words, one needs to calculate the gradient of the model with respect to the input. Since the ZOO method cannot access the CNN parameters, it attempts to approximate the gradients. It does this by making slight modifications in the input and by noting the change produced in the predicted c_a probability. Once the gradient is approximated, the attack creates adversarial images by using other already established white-box methods. Therefore, this attack still needs to obtain the model's gradient with respect to the input in order to function.

Finally, there is a sub-group of black-box attacks which do not have, nor attempt to approximate the model's gradients. This sub-group includes attacks such as the One-Pixel attack [55], Few-Pixel attack [46] and the Scratch attack [32]. However, although they have minimal requirements in terms of knowledge of the CNN parameters, they all have one major drawback. The malicious images they produce have very visible perturbations.

Another criterion for the differentiation of adversarial attacks (be they white-box, gray-box, or black-box) is their goal. Depending on whether the attacker interferes in the training phase or in the inference phase of a CNN, the attack is called a poisoning attack or an evasion attack, respectively [9]. Since in real-life scenarios attackers only have access to the already trained model, evasion attacks are more practical. Based on the attacker's goal, an additional separation factor refers to the category towards which the attacker attempts to send the image. In case of untargeted attacks, the only imposed condition is that $\mathcal{D}$ no longer belongs to c_a, according to $\mathcal{C}$. Meanwhile, targeted attacks involve choosing a particular target class $c_t \neq c_a$ and determining $\mathcal{C}$ to classify $\mathcal{D}$ precisely in c_t. Since targeted attacks are much more restrictive with respect to the adversarial image's classification, they are more challenging than untargeted attacks.

This study first proposes an alternative to the above works in the form of an evolutionary algorithm (EA)-based adversarial attack which falls under the category of black-box, evasion attacks. Our attack not only satisfies all requirements for a successful attack, but has minimal information requirements and is able to complete the more challenging task of producing targeted adversarial images. Moreover, the EA-based attack is highly practical, since it can be applied on any image to fool any CNN, and requires neither training other models, nor extracting the gradient of the attacked CNN with respect to the input.

This thesis, which is a continuation of the research program expressed in [6] and [5], is essentially made of 4 published papers ([11, 13, 12, 15]) and 3 submitted papers ([57, 38, 14]). The goal of this work is firstly to create the attack and measure its performance in typical circumstances. Secondly, the aspect of the algorithm's robustness is covered, by evaluating the degree to which the attack remains successful when the targeted CNN is protected by image filters. Additionally, in a move towards more challenging images, the thesis explores methods of applying the attack to create high-resolution adversarial images. Lastly, the goal of this thesis is also to understand the underlying manner in which the EA-based attack manages to fool models, in an attempt to shed some light on CNNs' vulnerabilities.

The thesis is organized as follows. After Chapter 2 gives a brief introduction to the concepts of CNNs, adversarial attacks and EAs, the details of the proposed method are given in Chapter 3, which is largely based on our papers [11, 13]. Apart from presenting and motivating the main attack method, Chapter 3 details the algorithm variants that are created and used in this thesis.

The performance of the EA-based attack is then measured in Chapter 4. Firstly, both the targeted and the flat versions of the EA-based attack are tested on the VGG16 [24] CNN trained on Cifar10 [34]. This part is based on the results exposed in [11, 13], which is an extension of our paper [11]. Then, the EA-based attack is evaluated against 10 different CNNs trained on ImageNet [18]. This part is extracted from our submitted paper [57]. Out of 1000 tests performed with 10 CNNs and 100 ancestor images, the targeted, *good enough* version of our attack (see Section 2.2 for explanations on *good enough* attacks) was successful in 96.8% of cases, requiring an average of 2712 generations and 16 minutes of computing time.

The following Chapter 5 (which is linked to our submitted paper [38]) evaluates the difficulty of transitioning from the attack performed on low-resolution images to the more complex situation of attacking high-resolution images. It identifies two possible methods of attacking large-sized images and evaluates both of them experimentally by using 10 different CNNs trained on ImageNet. One of the methods, in particular, identifies the best degradation function to transform high-resolution images into low-resolution images and the best interpolation function to perform the inverse process. These functions are then used as part of the EA-based attack on high-resolution images, which is proved to have a success rate of 90%.

Then, Chapter 6 (which is linked to our papers [12, 15]) explores whether image filters placed in front of the CNN input would protect the CNN from the EA-based attack. After one particular combination of filters (the median [23] and the unsharp mask [4] filters) is found to have this protective ability, the EA-based attack is modified in order to produce adversarial images that circumvent this defence. The modified attack is then proven to achieve robustness under this circumstance, without suffering a negative change in its success rate.

The following Chapter 7 (which is linked to our submitted paper [14]) analyzes the EA-based attack from various perspectives, such as noise frequency, image texture change, adversarial image transferability, CNN penultimate layer activations and behaviour at lower image regions. The above-mentioned perspectives are also adopted in the simultaneous analysis of a well-known, opposite white-box attack named Basic Iterative Method (BIM). The comparison is done in order to better understand the type of adversarial noise introduced by the two attacks, as well as which CNN vulnerabilities the attacks exploit.

Chapter 8 summarizes the achievements of this thesis and offers a series of directions for future research.

This work is completed by an Appendix containing tables and figures referred to throughout the different chapters.

Finally, before delving into the chapters described above, let us mention that, implementation-wise, all algorithms and experiments presented in this thesis use Python 3.8 [60] with the NumPy 1.17 [45], TensorFlow 2.4 [1], Keras 2.2 [16], and Scikit 0.24 [61] libraries. Computations are performed on nodes with Nvidia Tesla V100 GPGPUs of the IRIS HPC Cluster at the University of Luxembourg.

Chapter 2

Background

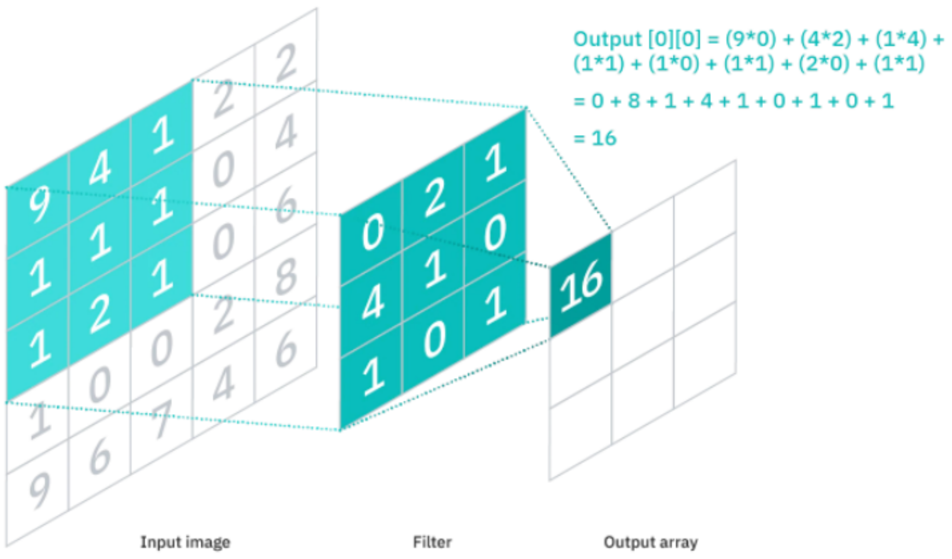
2.1 Convolutional Neural Networks

Brief history.

Although the field of computer vision has received significantly increasing attention only recently, computer scientist have been attempting to extract meaning from visual data for approximately 60 years. In 1959, neurophysiologists David Hubel and Torsten Wiesel designed an experiment that led to the groundbreaking discovery that visual processing always begins with neurons in the primary visual cortex reacting to simple image structures, such as edges. In parallel, the first digital image scanner was invented by Russel Kirsch in that same year. The next important insight was offered by neuroscientist David Marr in 1982, when he stated that vision is hierarchical. He created a representational framework where algorithms that detect low-level, simple image features such as edges and corners are used as first steps towards a high-level understanding of visual data [17].

These findings led Kunihiko Fukushima to create the first deep neural network, called Neocognitron. Through filters that would slide across an image 2D array, it would perform calculations in order to extract features from the image, which would then be fed as input to the next layer. Once Yann LeCun added the backpropagation learning algorithm to the Neocognitron, he created the groundbreaking LeNet-5, which was the first modern convolutional neural network. The next important moment was in 2001, when Paul Viola and Michael Jones created the first face detection framework, which, although it was not based on deep learning, would learn which simple features could help localize faces [17].

While until 2009 the field of computer vision was still limited by the scarcity of datasets, in 2009 the Cifar10 [34] dataset was introduced, with 60000 images belonging to 10 classes, and in 2010 the ImageNet [18] dataset was introduced, containing more than 1 million images and 1000 object categories. They immediately became benchmarks for the task of object recognition from images. Using ImageNet, a team from the University of Toronto created and trained the AlexNet CNN, which showed remarkable results and represented a breakthrough moment. Since then, multiple variants of CNNs with ever increasing accuracies have been created, and they remain the dominant method for object recognition [17].



Output [0][0] = (9*0) + (4*2) + (1*4) + (1*1) + (1*0) + (1*1) + (2*0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16
Input image
Filter
Output array

Figure 2.1: Convolution operation. The 2D filter in this example is slid along the 2D input image and the dot product between the filter and an image section of the same size is calculated at each step. The output value of each dot product is written in the corresponding location in the output activation map [20].

How do they work?

CNNs are typically used with image inputs and have the following 3 main types of layers: convolutional, pooling, and fully-connected (FC) [20]. Most of the computation occurs in the convolutional layers, which are the main components of CNNs. The constituents of a convolutional layer are the input data, a filter, and a feature map. These constituents of a convolutional layer are displayed in Figure 2.1. If we assume that the input is a color image, the input has three dimensions, which correspond to height, width, and depth, where the depth consists of the three RGB channels [20].

The **filter** is a 3D array which can vary in size, and whose constituent numbers are the weights that represent a particular image feature. The filter is slid along the input image to perform a dot product (convolution operation) between the filter and an image section of the same size as the filter, with the goal of checking whether the feature represented by the filter is present in the respective image section. After the convolution with one section of the image is done, the filter is shifted along the image by a given stride, to repeat the process in a new image section. This operation is repeated until the entire image has been covered. The series of dot products with a single filter leads to a 2D output array called activation map or feature map [20].

The size of the activation map is not necessarily the same as the input size, since each output value in the activation map is obtained from the convolution of the filter with an input image area that can be larger than one single pixel. The input image area covered by one filter is called receptive field. Another worthwhile observation is that the weights are constant as the filter is moved across the image [20].

During the training phase, the weights are adapted. However, there are three hyperparameters which are constant and need to be chosen prior to the training. The first is the number of filters, which determines the depth of the activation map. The stride is the number of pixels by which the filter is moved across the input image. Finally, padding is used when the filters do not fit the input image. It enlarges the input image by adding borders of zeros [20].

Each convolutional layer is followed by an activation function, which is typically the Rectified Linear Unit (ReLU) transformation. The activation function is applied to the feature map in order to introduce nonlinearity to the model. Multiple such convolutional layers can be stacked one after the other to capture increasingly higher receptive fields and create a feature hierarchy within the CNN [20].

Pooling layers typically follow convolutional layers and they perform dimensionality reduction by decreasing the number of parameters. In a similar way to convolutional layers, pooling layers also pass a filter across their input. However, as opposed to the weights of convolutional layers, pooling layers have no parameters, but they are rather aggregation functions applied to the values within the receptive field. The most common types of pooling are max (which extract the maximum value from a given input section) and average (which extract the average of all values from a given input section). While the pooling operation removes some useful information, it helps to reduce complexity and increase efficiency [20].

The **fully-connected (FC)** layer connects each node of its layer with all nodes from the previous layer. This is in contrast with convolutional layers, where the value of a particular number in the output only stems from a portion of the input (the receptive field). While convolutional and pooling layers are used to extract features from the input image, FC layers are used for classification based on the extracted features. Also, in object recognition, while non-final FC layers use the ReLU activation function, the last FC layer of a CNN is typically followed by the softmax activation function. This function converts the output of the last FC layer to a vector of numbers between 0 and 1, whose sum equals 1. In this vector, each number represents the probability of the image belonging to a particular object category [20].

2.2 Adversarial attacks on CNNs

Due to their high performance in the task of object recognition, CNNs have begun to be applied in real-life scenarios. Some of these scenarios are even safety-critical, such as recognizing road signs in autonomous vehicles. In such contexts, it is essential that the CNN is robust. However, recent works have proven just the opposite, through the creation of various adversarial attacks that are capable of fooling CNNs [64].

Adversarial examples are inputs that have intentionally been modified by the attacker in order to lead the machine learning model to make mistakes. While there are multiple types of attacks with differing characteristics, in the case of fooling object recognition CNNs they all begin by choosing an original image $\mathcal{A}$, that both humans and the CNN $\mathcal{C}$ to be fooled classify as c_a. In other words, the probability attributed to c_a is the maximum of $\mathcal{C}$'s output vector $o^{\mathcal{C}}_{\mathcal{A}}$:

$$a = \arg\max_{1 \leq j \leq M} \left(\mathbf{o}^{\mathcal{C}}_{\mathcal{A}} \right) \tag{2.1}$$

where M is the total number of categories present in the dataset that $\mathcal{C}$ was trained on. Adversarial attacks can be categorized based on the following criteria: the adversary's goal and the adversary's knowledge [64].

The **adversary's goal** can lead either to a poisoning attack or to an evasion attack. Poisoning attacks imply that the attacker introduces several fake inputs in the CNN's training dataset, which affects the model's accuracy. This scenario is common when the training database is freely available, such as with web-based repositories. In the case of evasion attacks, the classifier is already trained and usually well-performing. The attacker crafts malicious inputs only for the inference phase of the CNN, which will misclassify the given inputs. The attacker's goal can also separate between untargeted and targeted attacks. In the former case, the aim is to perturb $\mathcal{A}$ into an adversarial image $\mathcal{D}_a^k$, which is classified by $\mathcal{C}$ as a class c_u different than c_a ($c_u \neq c_a$), with no particular preference [64]:

$$c_u = \arg\max_{1 \leq j \leq M} \left(\mathbf{o}^{\mathcal{C}}_{\mathcal{D}_a^k} \right) \tag{2.2}$$

In the case of targeted attacks, the attacker chooses a target class $c_t \neq c_a$ and perturbs $\mathcal{A}$ into an adversarial image $\mathcal{D}_{a,t}^{C}$, which is classified by $\mathcal{C}$ as c_t:

$$t = \arg\max_{1 \leq j \leq M} \left(\mathbf{o}^{\mathcal{C}}_{\mathcal{D}_{a,t}^k} \right) \tag{2.3}$$

Equation 2.3 would be sufficient to create a valid targeted attack, and so we define the images which satisfy this condition as *good enough adversarial images.* However, for a stronger attack, an additional condition can be imposed on the adversarial image's classification. We define *τ-strong adversarial images* as those that satisfy the following requirement, where τ is a fixed constant threshold value $\in]0,1]$ that imposes a minimum confidence in the classification:

$$\boldsymbol{o}^{\mathcal{C}_k}_{\mathcal{D}_{a,t}^k(\mathcal{A}_a)}[t] \geq \tau \tag{2.4}$$

The **adversary's knowledge** separates between white-box attacks, black-box and gray-box attacks. In white-box attacks the attacker has access to the model's parameters, architecture, gradients, etc., and can use this information to carefully craft adversarial examples. In contrast, black-box attacks can only feed inputs to the model and query its outputs, without having access to its internal configuration. Gray-box attacks initially use the CNN's architecture and parameters to generate adversarial examples, which are then used to train a generative model to produce similar adversarial images. Once the generative model is trained, the attack no longer requires information about the CNN [64].

Finally, for all adversarial attacks, irrespective of their nature, there is one common requirement, namely that the adversarial image $\mathcal{D}$ be visually indistinguishable from the original $\mathcal{A}$.

2.3 Evolutionary algorithms

Evolutionary algorithms are search-based optimization techniques that take inspiration from natural selection. The process of natural selection implies the selection of the fittest individuals from a population. The offspring produced by these individuals inherit some of their parents' traits, but they are completely new individuals, different from both parents. They thus have the chance of having a higher fitness than their parents, and the repetition of this process represents the evolution.

In algorithm terms, the general principle in optimization is that an input to a process is modified, such that its output is better than the original input. Although the definition of "better" varies with the problem at hand, the same mathematical concept is used, namely maximizing or minimizing one or more objective functions. This class of algorithms was developed by John Holland and his colleagues at University of Michigan. There are five phases in an evolutionary algorithm, namely: initial population, fitness function, selection, crossover, and mutation [59].

Evolutionary algorithms begin with a **population**, namely a pool of possible solutions to the given problem, where each individual is characterized by a set of parameters referred to as genes. The **fitness function** captures the goal of the entire evolution. It evaluates how fit the individuals are by assigning them fitness scores. This fitness score impacts the probability that an individual will be selected for reproduction. The **selection** phase compares the fitness of the individuals and selects the population members with the highest fitness. The individuals with higher fitness are prioritized for reproduction, such that they pass their genes to the next generation [43].

The **crossover** phase starts by selecting pairs of parents to reproduce. Next, a crossover point is selected at random somewhere along the genes. To produce offspring, the algorithm selects the parents' genes until the crossover point is reached, and exchanges the two series between themselves. Once the offspring is created, some of them might have randomly selected genes be mutated. The role of **mutations** is to diversify the population and prevent premature convergence. The algorithm ends once the termination condition has been reached. This termination condition checks whether the population has converged, meaning that there is very little improvement from one generation to another, thus the algorithm has found a good set of solutions to the problem [43].

By making use of the notions introduced here, the following chapter gives details about the precise implementation of our EA-based adversarial attack on CNNs.

Chapter 3

Attack Method

This chapter is mostly extracted from [13], with the exception of Subsection 3.5, which is extracted from [15]. The chapter offers a detailed explanation of the EA-based adversarial attack's functioning. Our algorithm is an EA that aims to deceive both CNNs and human beings. Clearly, the design of the EA depends on the scenario governing this dual deception. The two scenarios addressed here start the same way. One is initially given an image, the "ancestor" $\mathcal{A}$, labelled by the CNN as belonging to $c_{\mathcal{A}}$.

The first scenario is the "target" scenario. A target category $c_t \neq c_{\mathcal{A}}$ is chosen. The task of the EA is to evolve $\mathcal{A}$ to a new image $\mathcal{D}$ (a "descendant") that the CNN classifies into c_t, but in such a way that the evolved adversarial image $\mathcal{D}$ remains very similar to the ancestor $\mathcal{A}$. With perturbations kept as least visible as possible, a human being should still consider $\mathcal{D}$ as obviously belonging to category $c_{\mathcal{A}}$.

In the second "flat" scenario, the task of the EA is to evolve $\mathcal{A}$ into a descendant $\mathcal{D}$, that the CNN is unable to classify with certainty to any specific category, in the sense that the CNN ranges $\mathcal{D}$ to all categories with the same plausibility modulo a tiny and controlled margin. The same constraint of similarity between $\mathcal{D}$ and $\mathcal{A}$ as in the first scenario remains: a human being should still consider $\mathcal{D}$ as belonging to $c_{\mathcal{A}}$.

$\text{EA}_d^{\text{target}}$ refers to the evolutionary algorithm of the first scenario, and $\text{EA}_d^{\text{flat}}$ to the evolutionary algorithm of the second scenario, where d referres to the similarity measures detailed in Section 3.2. The $\text{EA}_d^{\text{target}}$ algorithm or variants of it are used in all chapters, while $\text{EA}_d^{\text{flat}}$ is used in Chapter 4 Section 4.1. In all chapters and sections we run our EA-based attack to generate τ-*strong adversarial images.* However, in Chapter 4 Section 4.2, we also run our EA-based attack to generate *untargeted* and *good enough* adversarial images.

3.1 Common features between $\text{EA}_d^{\text{target}}$ and $\text{EA}_d^{\text{flat}}$

While the fitness functions of $\text{EA}_d^{\text{target}}$ and $\text{EA}_d^{\text{flat}}$ differ, and hence so does the evaluation step, the population initialization and the evolution steps are similar between the two attack variants.

Population initialization. A population size being fixed, the initial population is set to a number of copies of the ancestor $\mathcal{A}$ equal to the chosen population size.

The Evaluation step consists in running the fitness function on all population individuals to measure how well each of them is approaching the goal of the evolution. Even if the fitness functions of $\text{EA}_d^{\text{target}}$ and of $\text{EA}_d^{\text{flat}}$ differ, they are similar conceptually, and the evolution aims at maximising their values. In both cases, as shown in Sections 3.3 and 3.4, the fitness function is the sum of two components. One component of the equation defining the fitness function deals with deceiving machines (which differs according to the ”target” or ”flat” scenario), the other with deceiving humans. This latter aspect is addressed in Section 3.2.

Evolution encompasses multiple steps:

- **Segregation**. After evaluation, the scores are used to segregate the population into three classes:
  - the elite consists of the top ten individuals, which pass unchanged to the next generation
  - the ”didn't make it” consists of the lower scored half of the population, which is discarded. It is replaced by the same number of mutated individuals from the elite and middle class
  - the middle class contains the remaining individuals

- **Mutations**. Two types of mutation are considered, namely small and large scale ones:
  - For pixel mutations, a power law is used to randomly select the number of pixels to be mutated. By following a power law, this number is often small, encouraging exploitation. However, the occurrence of larger values also takes place, encouraging exploration and offering ergodicity properties. Once this number is selected, the pixels are randomly chosen and modified by a random $\pm\delta$. In order to maintain a high similarity between the images, the mutations added throughout generations can be clipped and not be allowed to exceed a certain range [x - ϵ, x + ϵ] imposed by the ϵ constant, where x is the original pixel value.
  - For circle intensifying mutations, the intensifying factor is chosen with a normal law centred on 1 with a standard deviation decreasing from 0.6 to 0.1 as the generation number increases. The radius and location of the circle are chosen uniformly random.

  Individuals in the elite are not mutated. The members replacing the “didn't make it” group are all mutated, while half of the middle class members are mutated.

- **Cross-overs** occur after the mutation step. Two children are created simply by swapping a randomly selected rectangular area between two parents. The number of parents and the individuals are selected randomly. After the cross-over, the parents are discarded and replaced by the children. This step is applied to all but the elite class.

The Termination conditions signal the end of the algorithm's run, if any of them is met. For this purpose, we introduce two parameters. Since this EA-based attack is targeted, the $\tau \in]0, 1]$ parameter is a threshold that refers to the target class's probability of the best-fit individual. The termination condition checks whether the target class probability $o[c_t]$ exceeds the τ threshold

$(o[c_t] \geq \tau)$. The second possible termination condition is the exceedance of a given maximum number of generations G.
As in [5], an equivalence is made between the population of the EA and a batch of the CNN so that the CNN can process the EA population in parallel, as a single batch, using a GPGPU.

3.2 Image similarity

The difference between two images (of the same size) i and i' can be evaluated in many ways. But only some of them give a hint at the similarity between two images, as a human being would perceive it. We explore here two of them, namely $d = L_2$ and $d = SSIM$, that assess proximity in a different way. The former belongs to the family of L_k norms acting on vector spaces. In the present context, the L_k norms address performed modifications pixel for pixel. Based on a series of experiments, we found that there is no convincing advantage to use larger k's than $k = 2$. On the other hand, it is useful to consider an alternative measure, like SSIM, that assesses modifications performed on more structural components of a picture, rather than the pixel for pixel approach.

- The L_2-distance, which calculates the difference between the initial and modified pixel values:
$$L_2\left(i, i'\right) = \sum_{p_j} \left| i[p_j] - i'[p_j] \right|^2, \tag{3.1}$$
where p_j is the pixel of the image in j^{th} position, and $0 \leq \imath[p_j] \leq 255$ is the corresponding pixel value of the image i.

  A minimisation of the L_2-norm in the fitness function would lead to a minimisation of the overall value change in the images' pixels.

- The structural similarity (SSIM [62]) method attempts to quantify the perceived change in the structural information of the image, rather than simply the perceived change. The Structural Similarity Index compares pairs of sliding windows (sub-samples of the images) W_x and W_y of size $N \times N$:
$$SSIM_W\left(W_x, W_y\right) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \tag{3.2}$$
The quantities μ_x and μ_y are the mean pixel intensities of W_x and W_y, σ_x^2 and σ_y^2 the variance of intensities of W_x and W_y, and σ_{xy} their covariance. The purpose of c_1 and c_2 is to ensure that the denominator remains far enough from 0 when both μ_x^2 and μ_y^2 and/or both σ_x^2 and σ_y^2 are small.

  The N_W window pairs to consider equals the number of pixels (times the number of colour channels if appropriate) of the picture cropped by a frame that prevents the windows from "getting out" of the picture. With pictures of size $h \times w \times c$ and windows of size $N \times N$, one gets:
$$N_W = \left(h - 2\left\lfloor \frac{N-1}{2} \right\rfloor\right) \times \left(w - 2\left\lfloor \frac{N-1}{2} \right\rfloor\right) \times c. \tag{3.3}$$

The SSIM value for two images i and i' is the mean average of the values obtained for the N_W window pairs $(i_k,\ i'_k)$:

$$SSIM(i,i') = \frac{1}{N_W}\sum_{k=1}^{N_W} SSIM_W\left(i_k, i'_k\right). \tag{3.4}$$

Unlike the L_2-norm, the SSIM value ranges from -1 to 1, where 1 indicates perfect similarity.

Whether for $d = L_2$ or $d = SSIM$, the EA might consider preferable to slightly modify many pixels, as opposed to changing fewer pixels but in a more apparent way. This contrasts with the approach of Su *et al.* [55], where only one pixel is modified, possibly being assigned a very different colour that can make it stand out.

3.3 The fitness function of $\mathbf{EA}_d^{\mathbf{target}}$

The fitness function performing the evaluation in the "target" scenario combines the two following factors. On the one hand, the evolution is directed towards a larger classification of the images as belonging to c_t. On the other hand, similarity between the evolved and ancestor images is highly encouraged. Our fitness function therefore depends on the type of similarity measure that is used.

If using the L_2-norm, it can be written as

$$fit_{L_2}^{\text{target}}(ind, g_i) = A_{L_2}^{\text{target}}(g_i)\boldsymbol{o}_{ind}[c_t] - B_{L_2}^{\text{target}}(g_i)L_2(ind, \mathcal{A}), \tag{3.5}$$

where *ind* designates a given individual (an image), g_i the i^{th} generation, and $\boldsymbol{o}_{ind}[c_t]$ designates the value assigned by the CNN to *ind* in the target category c_t. The quantities $A_{L_2}(g_i), B_{L_2}(g_i) > 0$ are coefficients to weight the members and balance them. They vary with the generations dealt with by the EA, since we chose to assign different priorities to the different generations. The first generations were thus assigned the task of evolving the image to the target category, while the later generations focused on increasing the similarity to the ancestor, while remaining in c_t.

In the case of structural similarity, a higher value translates into a higher fitness of the individual. Therefore, *mutatis mutandis*, the difference in the fitness function becomes a sum:

$$fit_{SSIM}^{\text{target}}(ind, g_i) = A_{SSIM}^{\text{target}}(g_i)\boldsymbol{o}_{ind}[c_t] + B_{SSIM}^{\text{target}}(g_i)SSIM(ind, \mathcal{A}), \tag{3.6}$$

3.4 The fitness function of $\mathbf{EA}_d^{\mathbf{flat}}$

In the "flat" scenario, the fitness function also combines two factors. On the one hand, the evolution is directed towards a "flat" classification of the image in all categories $c_1, \cdots, c_\ell$. The measure D_{flat} of the "flatness" of a classification is defined by the equation (3.7), where flat is a vector of ℓ values, all set to flat$[k] = \frac{1}{\ell}$.

$$D_{\text{flat}}(ind) = \sum_{k=1}^{\ell}\Big(\left(o_{ind}[k] - flat[k]\right)\log_{10}\left(o_{ind}[k]\right)\Big)^2 \geq 0. \tag{3.7}$$

A larger value of $D_{\text{flat}}(ind)$ means that ind is further away from the desired "flatness". Similarity between the evolved and ancestor images is highly encouraged. For $d = L_2$, our fitness function is given by:

$$fit_{L_2}^{\text{flat}}(ind, g_i) = -A_{L_2}^{\text{flat}}(g_i)D_{\text{flat}}(ind) - B_{L_2}^{\text{flat}}(g_i)L_2(ind, \mathcal{A}), \tag{3.8}$$

and for $d = SSIM$ by:

$$fit_{SSIM}^{\text{flat}}(ind, g_i) = -A_{SSIM}^{\text{flat}}(g_i)D_{\text{flat}}(ind) + B_{SSIM}^{\text{flat}}(g_i)SSIM(ind, \mathcal{A}). \tag{3.9}$$

3.5 Motivation for EA$_d$'s design: Adapted_EA" versus "classic_EA"

In this section, we show, from a "pure" evolutionary algorithm point of view, that $\text{EA}_d^{\text{target},\mathcal{C}}$ (reffered to here as "adapted_EA") presents a series of important and substantial differences compared to the approach classically ([21]) adopted for EAs performing similar tasks, and we prove that these differences lead to a comparative advantage in terms of performance. At first, we examine these differences from a conceptual point of view, meaning independently from any specific task. For simplicity, we refer to our version as "adapted_EA" and to its classical version as "classic_EA". We then compare the performances of these algorithms for the task consisting in fooling VGG16 [7] trained on CIFAR-10 at image recognition for the *target scenario.* In other words, these algorithms are given the task to evolve an ancestor image $\mathcal{A}$ into an adversarial image $\mathcal{D}$. We specify the parameters of the EAs, and run the algorithms for four different ancestor/target combinations.

Conceptual differences between "adapted_EA" and "classic_EA"

To illustrate the differences between our version ("adapted_EA") and the classic version ("classic_EA", as described in [21]) of an EA, let us provide their respective algorithmic pseudo-codes. We assume that both have a fixed population size, that remains constant generation for generation. For both, we set the initial population as made of identical copies of the considered ancestor. Based on our experiments, we took a population size of 160 as the best trade-off in terms of speed and accuracy.

Algorithm 1 "Classic_EA" algorithm pseudo code

```
1: BEGIN
2:     INITIALISE population;
3:     EVALUATE each candidate;
4:     REPEAT UNTIL (Termination condition is satisfied) DO
5:         1 SELECT top 10-20% (16 to 32 individuals) as parents;
6:         2 RECOMBINE pairs of parents resulting in offsprings;
7:         3 MUTATE the offsprings;
8:         4 EVALUATE new candidates;
9:         5 SELECT individuals for the next generations;
10:    END
11: END
```

Algorithm 2 "Adapted_EA" algorithm pseudo code

```
1: BEGIN
2:     INITIALISE population;
3:     EVALUATE each candidate;
4:     REPEAT UNTIL (Termination condition is satisfied) DO
5:         SELECT
6:             split into 3 groups;
7:             Elite: top 10;
8:             "didn't make it": last 80;
9:             Middle-class: 70;
10:        RECOMBINE
11:            elites + middle-class resulting in offsprings;
12:            replace "didn't make it" with offsprings;
13:        MUTATE
14:            middle-class and offsprings;
15:        EVALUATE each candidate;
16:    END
17: END
```

The main difference between "classic_EA" (as described in Algorithm 1) and our version (as described in Algorithm 2) is the process of selection, recombination and mutation. In "classic_EA", the best 10-20% of the population are selected as elites (hence between 16 and 32 individuals), and new offsprings are generated with these elites by recombination and mutation. Then the last 10-20% (idem) of the population are eliminated, and only these 10-20% are updated at each generation. However, in our version, the number of elites is set to the first 10 individuals, then the algorithm starts to modify the whole rest (150 individuals) of the population by eliminating, mutating, and recombining with elites just after the first generation.

The task on which we shall evaluate the performances of both approaches is the construction of adversarial images for CNNs. Although our algorithm $\text{EA}_d^{\text{target},\mathcal{C}}$ is efficient for a series of CNNs, we make our point here for the instantiation $\text{EA}_{L_2}^{\text{target,VGG-16}}$ of this algorithm (Algorithm 2) and of its classical EA version (Algorithm 1), for $\mathcal{C}$ = VGG16 trained on CIFAR-10 (see Subsection 4.1.1 for a description of VGG16 trained on Cifar10), and for metric $d = L_2$. Starting from a common ancestor image $\mathcal{A}$ of size $32 \times 32 \times 3$ labelled by VGG16 as belonging to c_a, and from a target category $c_t \neq c_a$, the specific parameters and choices of the algorithms are as follows: $B(g_p, ind) = 10^{-5}$, $A(g_p, ind) = 10^{-\log_{10} o_{ind}[c_t]}$, $\tau = 0.95$ and a maximum number of generations of 7000. Pixel values are modified in a range ± 3 in both EA versions used here. The algorithms use the same parameters and techniques for mutation and crossover operations.

If the EAs terminate successfully, one names $\mathcal{D}_{a,t}(\mathcal{A})$ the adversarial image resulting of $\text{EA}_{L_2}^{\text{target,VGG-16}}$ (Algorithm 2) run on $\mathcal{A}$, and $\mathcal{D}_{a,t}^{\text{classic}}(\mathcal{A})$ as the result of the classic (Algorithm 1) version of the EA also run on $\mathcal{A}$. The algorithms terminate after 7000 generations at the latest, whether or not they succeeded in creating such an adversarial image.

Experimental comparison of "adapted_EA" with "classic_EA"

We compare experimentally the efficiency of both versions of the EA for four ancestor/target pairs of categories Animal/Animal, Object/Object, Animal/Object, and Object/Animal.

Concretely, the Animal ancestor categories are bird and dog, with the image $\mathcal{A}_3$ as ancestor for the bird category c_3, and $\mathcal{A}_6$ as ancestor for the dog category c_6. Similarly, the Object ancestor categories are plane and ship, with the images $\mathcal{A}_1$ as ancestor for the plane category c_1, and the image $\mathcal{A}_9$ as ancestor for the ship category c_9.

With these ancestors, we performed 10 independent runs (meaning with 10 distinct, randomly chosen seed values) of the algorithms for each of the following combinations: the bird/cat pair (Animal/Animal), the plane/truck pair (Object/Object), the dog/car pair (Animal/Object) and the ship/horse pair (Object/Animal).

Performance comparison. In all cases, the 10 independent runs of each algorithm succeeded in (far) less than 7000 generations. Table 3.1 gives the minimum number of generations (min_{gen}), the maximum number of generations (max_{gen}), and the mean generations ($mean_{gen}$) obtained over the 10 independent runs of each algorithm. The convergence graph, plotted in Figure 3.1, pictures the convergence speed of both algorithms for all cases. The horizontal axis of these graphs is the number of generations, and the vertical axis is the average log probabilities of target category obtained for these 10 independent runs.

Table 3.1: Comparison of classic_EA and adapted_EA in generating adversarial images for the *target scenario* for 4 different Ancestor/Target combinations ($\mathcal{A}_a$ is the ancestor image in c_a used in the experiments) to fool VGG16 trained on CIFAR-10. The results are over the 10 independent runs of each algorithm.

Ancestor/Target	Algorithms	min_{gen}	max_{gen}	$mean_{gen}$
bird ($\mathcal{A}_3$)/cat	classic_EA	1726	2433	2172.9
	adapted_EA	1353	2177	1629.2
plane ($\mathcal{A}_1$)/truck	classic_EA	1311	1810	1547.5
	adapted_EA	1050	1439	1194.8
dog ($\mathcal{A}_6$)/car	classic_EA	1132	1334	1199.3
	adapted_EA	811	1050	907.0
ship ($\mathcal{A}_9$)/horse	classic_EA	1972	3412	2582.1
	adapted_EA	1543	3171	2377.8

Results and Discussion. As can be seen in Table 3.1, "adapted_EA" outperforms "classic_EA" in all cases. The former requires less generations than the latter to obtain adversarial images with 0.95 confidence. Figure 3.1 confirms that "adapted_EA" converges faster than "classic_EA". The graphs indicate that apparently both algorithms exhaust most of their generations to find correct regions and/or pixels to modify. Once done, their learning curves accelerate drastically, still with "adapted_EA" leading the race against "classic_EA".

Although both algorithms start the search with the same 160 identical images, their respective performances differ substantially, as a consequence of their distinct updating process of the population. Indeed, "adapted_EA" starts these updates for the whole population, except for the elite individuals passed unchanged to the next generation, and does so right after the 1st generation. However, "classic_EA" only updates 20% of its population in each generation. Changing only 32 individuals, as opposed to changing 150 individuals, makes it much slower for the classic version compared to its adapted competitor. These results not only legitimize the choices made in our earlier work ([5, 6, 11, 13, 12]). In addition, they provide some evidence, that for similar

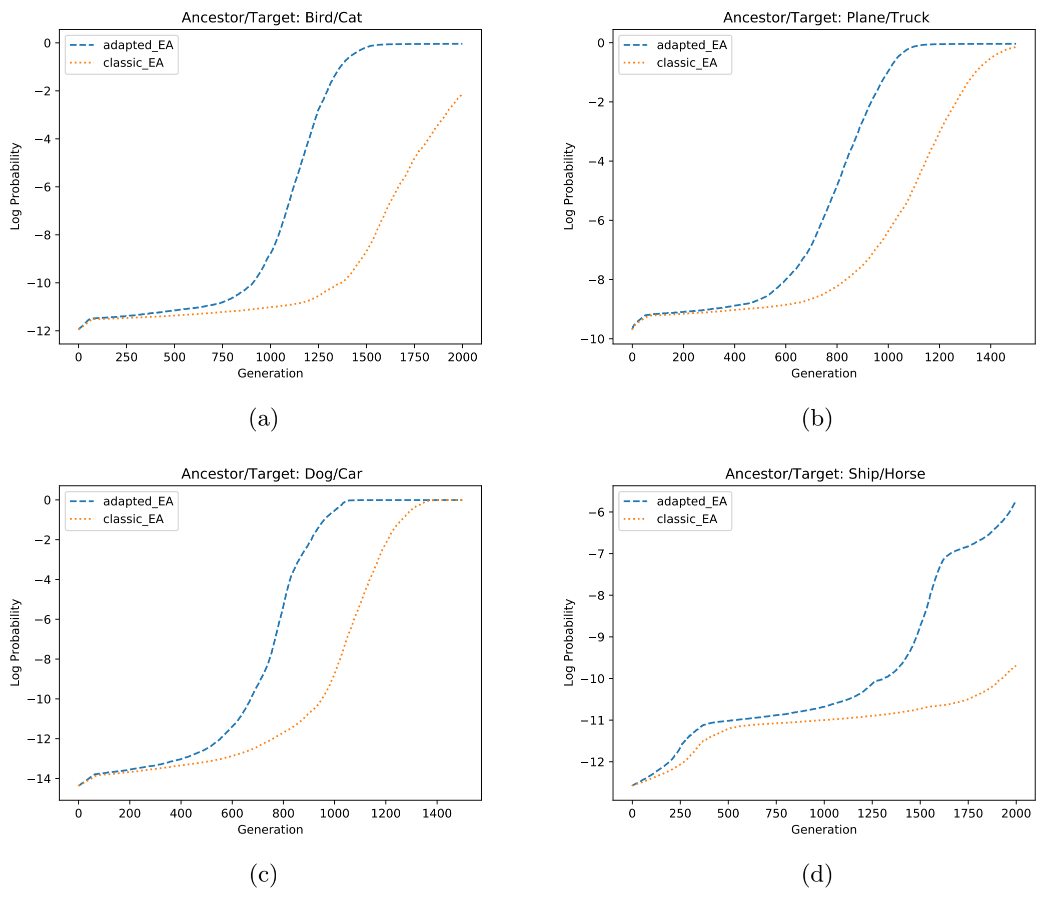


Figure 3.1: Convergence characteristics of classic_EA and adapted_EA for different ancestor/target pairs. These experiments are performed with ancestors $\mathcal{A}_3$ (in the bird category), $\mathcal{A}_6$ (dog), $\mathcal{A}_1$ (plane) and $\mathcal{A}_9$ (ship).

exploration problems with a starting point made of the same individuals (hence not only for the construction of images adversarial for a CNN), the generic selection and mutation process adopted in "adapted_EA" (algorithm 2) shortens the learning period of the algorithm and enhances up the convergence speed.

We complete this comparative analysis by assessing the potential difference of behavior between the adversarial images created by each version of the EA. To this purpose, we computed the Kullback-Leibler divergence [35] between the probability densities derived from the normalized histograms of the pixel modifications induced by each of them. In all cases, the values (averaged over the ten independent runs) of the Kullback-Leibler divergences are negligible (they vary between $2.24e-04$ and $5.17e-03$), indicating that the noise created by one version of the EA significantly differs from the noise created by the other. Hence, while both versions of the EA do create adversarial images, the introduced modifications by each of them differ strongly, although both these modifications introduced by each EA on the one hand, as well as their differences on the other hand, are not perceptible by a human.

Chapter 4

Attack Performance

In this chapter, we present the experiments that were performed to measure the algorithm's performance for three different scenarios. We first show that the attack is efficient in the targeted and flat (Section 4.1) scenarios for VGG16 trained with the Cifar10 [34] dataset. We then show that the attack is efficient for the targeted scenario (Section 4.2) on 10 different CNNS trained with the ImageNet [18] dataset.

4.1 Target and flat scenarios: attack against VGG16 trained on Cifar10

A large part of this section is extracted from [13], which is an extension of [11]. The section presents the experiments and results obtained with $\text{EA}_d^{\text{target}}$ and $\text{EA}_d^{\text{flat}}$ on the Cifar10 dataset.

4.1.1 Dataset, Neural Network Architecture and Parameters of the two EAs

VGG16 trained on Cifar-10

The feasibility study regarding the two scenarios is tested against one concrete example: Cifar-10 and VGG16. The dataset Cifar-10 [34] contains $50,000$ training images and $10,000$ test images of size 32x32x3. Cifar-10 sorts $\ell = 10$ categories (see Table 4.1) into two groups: 6 categories (c_3, c_4, c_5, c_6, c_7 and c_8) form the group of animals, and 4 categories (c_1, c_2, c_9 and c_{10}) the group of objects.

VGG16 [52] is a convolutional neural network (CNN) that passes input images through 16 layers to produce a classification output. As shown in Figure 4.1, the model consists of 5 groups of convolution layers and 1 group of fully-connected layers. Each convolution filter has a kernel size of 3×3 and a stride of 1. Meanwhile, pooling is applied on regions of size 2×2, with no overlap.

Table 4.1: Cifar-10.– For $1 \leq i \leq 10$, the 2nd row specifies the category c_i of Cifar-10. The 3rd row specifies the numbering of the image belonging to c_i, taken from the test set of Cifar-10, and used as ancestor in our experiments. These images are pictured on the diagonal in Figures 9.1, 9.2, or on the first row of Figure 9.3 in Appendix 9.1.2

i	1	2	3	4	5	6	7	8	9	10
c_i	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
N^0	281	82	67	91	455	16	29	17	1	76

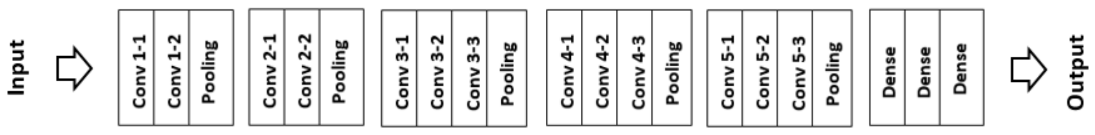


Figure 4.1: Architecture of VGG16. There are 5 convolution groups and 1 fully-connected group of Dense layers, in which every neuron is connected to every neuron of the next layer. Each convolution group ends with a pooling layer.

Since VGG16 was initially designed for the ImageNet dataset [18], a series of adjustments were necessary for its use with the Cifar-10 dataset [24]. The CNN used here was therefore an adapted VGG16 architecture obtained through the steps described in [40]. Specifically, VGG16's input size was adjusted to $32 \times 32 \times 3$, Batch Normalization layers were added before every nonlinearity and the first two fully-connected layers were reduced in size from 4096 to 100. Moreover, dropout was added to all 6 groups of the network with the following rates: 0.3 for the first 3 convolution groups, 0.4 for the fourth group, 0.5 for the fifth group and 0.5 for the fully-connected layers.

We made use of this adjusted VGG16 pre-trained on Cifar-10 with a validation accuracy of 93.56% [24]. The same pre-trained model was used throughout the evolutionary algorithms $\mathrm{EA}_d^{\mathrm{target}}$ and $\mathrm{EA}_d^{\mathrm{flat}}$.

EA Parameters

Both EAs run with $\delta = 3$, no maximum limit G on the number of generations and a population of size 160, the ancestor images being chosen from the Cifar-10 [34] test set. For any source category out of the 10 categories of Cifar-10, a random image was selected from the 1000 test images belonging to that category. This image was then set as the ancestor for both EAs. The specific ancestor images used in our experiments are referred to in Table 4.1's last row, and pictured in the first row of Figure 9.3 in Appendix 9.1.2.

In the flat scenario, $A_d^{\mathrm{flat}}(g_i)$ and $B_d^{\mathrm{flat}}(g_i)$ take constant values, independent of g_i. In the target scenario, the values of $A_d^{\mathrm{target}}(g_i)$ and of $B_d^{\mathrm{target}}(g_i)$ vary, depending on the generation, although they do so in a different way.

Target scenario: the target category was selected from all labels, excluding the source category. This led to 90 (source, target) couples of categories altogether (10 source categories and 9 different

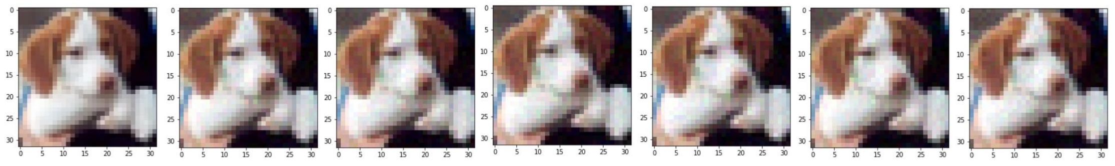


Figure 4.2: Target scenario with the *dog→horse* combination.– Comparison of the original (on the left) with 3 evolved pictures created by $EA_{L_2}^{\text{target}}$ (3 next from the left), and 3 created by $EA_{SSIM}^{\text{target}}$ (3 last) at different stages, classified by VGG16 as the target category "horse" with probabilities 6.08×10^{-6}, 0.5, 0.90, and 0.95.

target categories for each source category), and therefore to 90 adversarial images. For any g_i, we set $B_d^{\text{target}}(g_i) = 10^{-\log_{10}(d(ind,\mathcal{A}))}$ for $d = L_2$ or $d = SSIM$ (in this latter case, one assumes that $SSIM(ind, \mathcal{A}) > 0$, meaning that ind and $\mathcal{A}$ are close enough). The value of $A_d^{\text{target}}(g_p)$ depends on $\boldsymbol{o}_{ind}[c_t]$ (note that $\log_{10} \boldsymbol{o}_{ind}[c_t] \leq 0$).

- If $\boldsymbol{o}_{ind}[c_t] < 10^{-3}$, then $A_d^{\text{target}}(g_p) = 10^{3-\log_{10} \boldsymbol{o}_{ind}[c_t]}$.
- If $10^{-3} \leq \boldsymbol{o}_{ind}[c_t] < 10^{-2}$, then $A_d^{\text{target}}(g_p) = 10^{2-\log_{10} \boldsymbol{o}_{ind}[c_t]}$.
- And if $10^{-2} \leq \boldsymbol{o}_{ind}[c_t]$, then $A_d^{\text{target}}(g_p) = 10^{1-\log_{10} \boldsymbol{o}_{ind}[c_t]}$.

The goal for the target class probability was set to 0.95, meaning the algorithm would stop when the fittest evolved image reached this value.

Flat scenario: we considered this scenario for each of the 10 source categories, leading to 10 adversarial images. For any g_i, we set $A_{L_2}^{\text{flat}}(g_i) = A_{SSIM}^{\text{flat}}(g_i) = 1$, and $B_{L_2}^{\text{flat}}(g_i) = 10^{-8}$, $B_{SSIM}^{\text{flat}}(g_i) = 10^{-3}$. This led to the values of both $fit_{L_2}^{\text{flat}}(ind, g_i)$ and $fit_{SSIM}^{\text{flat}}(ind, g_i)$ being negative. We let the evolutions run until both flatness and image similarity were achieved, with no compromise on either side. Following a trial and error process, the goal for the individuals' fitness was set to -0.001, meaning the algorithm would stop when the fittest evolved image reached this value.

4.1.2 Running $\text{EA}_d^{\text{target}}$: Examples, Results and Discussion

To illustrate, Figure 4.2 shows an original image in the dog category and evolved images classified by VGG16 as the target horse category with probabilities 0.5, 0.9, and 0.95. They were created by $\text{EA}_d^{\text{target}}$ in $26, 32, 34$ generations with $d = L_2$, and $21, 24, 25$ generations with $d = SSIM$.

Figure 4.3 shows the graph of the source dog and target horse class probabilities obtained along the evolution referred to in Figure 4.2. The paths of the two probabilities appear to be the inverse of each other, with their sum remaining almost constant at a value of about 1.0 throughout the evolution process. This suggests that the increase of the target class probability and the decrease of the source class probability take place at the same pace.

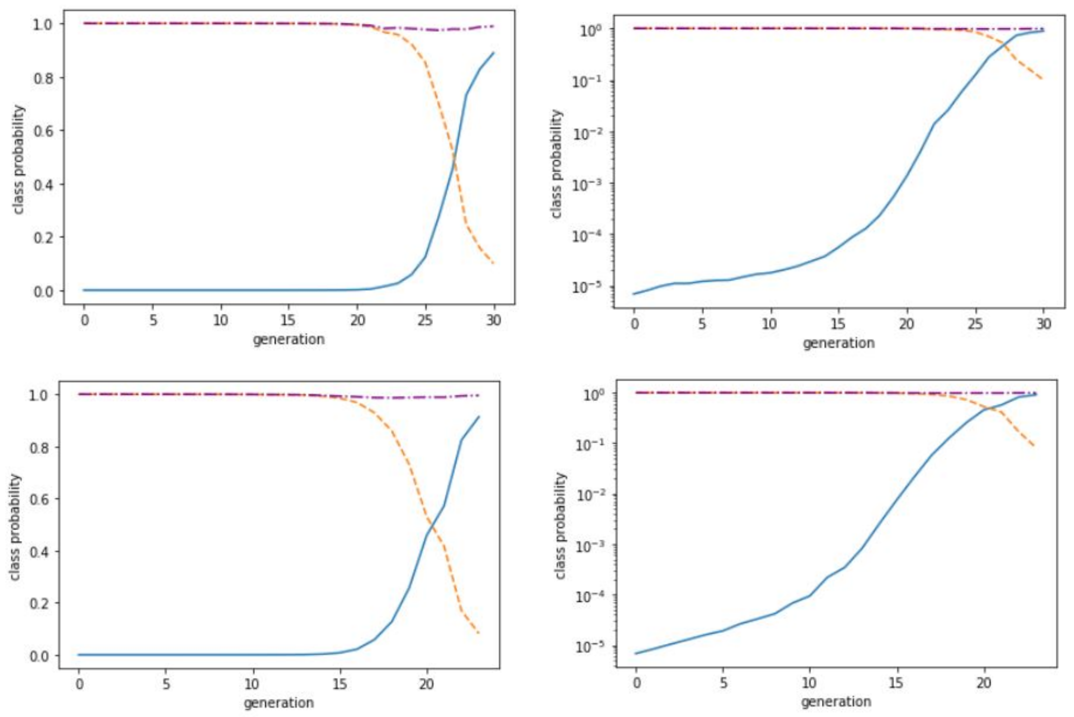


Figure 4.3: Target scenario with the *dog→horse* combination of Figure 4.2.– Linear (left) and logarithmic (right) scale plots of an evolution of source (dashed line) and target (solid line) class probabilities using the L_2 norm (top) or using SSIM (bottom), with respect to the generation number. Both plots display in addition the sum (dash-dotted line) of the source and target probabilities: The sum remains about constant at a value of 1.0 throughout the evolution process, suggesting that the increase of the target class probability and the decrease of the source class probability take place at the same pace.

Figure 4.4 shows examples of the four possible (*source, target*) combinations created by EA_d^{target}, where *source* and *target* are *animals* or *objects*, with $d = L_2$ and $d = SSIM$. The number of generations required to evolve these images is presented in Table 4.2.

Table 4.2: Number of generations required by $\text{EA}_d^{\text{target}}$ to evolve the four *animals* and *objects* combinations of images of Figure 4.4 with $d = L_2$ and $d = SSIM$.

	Generations	
Source → Target	L_2	$SSIM$
bird → dog	45	47
deer → ship	45	48
airplane → automobile	18	46
truck → frog	31	31

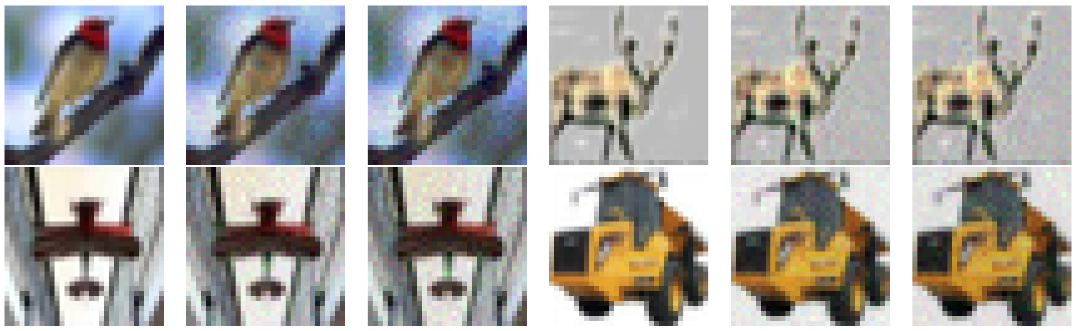


Figure 4.4: Target scenario with (*source, target*) combinations, where *source* and *target* are *animals* or *objects*.– Comparison of ancestor (1st image) and descendant images obtained by $\text{EA}_d^{\text{target}}$ for $d = L_2$ (2nd image) and $d = SSIM$ (3rd image). At the top, *animal→animal* and *animal→object* combinations: From left to right, VGG16 outputs the following class probabilities: 0.999 bird, 0.954 dog, 0.953 dog; 0.999 deer, 0.951 ship, 0.954 ship. Below, the *object→object* and *object→animal* combinations: From left to right, 0.998 airplane, 0.951 automobile, 0.952 automobile; 0.999 truck, 0.952 frog, 0.958 frog.

Results

Depending on the chosen source and target categories, reaching the probability of 0.95 required between 5 and 124 generations, for the 90 altogether evolutions, as specified in Appendix 9.1.1 (in Figure 9.1 with L_2 and Figure 9.2 with SSIM). The average computation time per generation is 0.03 ± 0.01s for L_2 and 0.15 ± 0.02s for SSIM. Organizing the 10 categories of the Cifar-10 dataset into the group of animal categories and the group of object categories, the *animal→animal* and *object→object* evolutions took fewer generations than the *animal→object* and *object→animal* ones. The required number of generations varied significantly not only with the category of an ancestor, but also with the particular image extracted from a given category. Table 4.3 presents the mean number of generations for the 4 combination types for the 90 evolutions of Figure 9.1 and of Figure 9.2 in Appendix 9.1.1.

Table 4.3: Mean value and standard deviation for the number of generations required by $\text{EA}_d^{\text{target}}$ in the 4 different combination types for $d = L_2$ (2nd row) and $d = SSIM$ (3rd row) for the altogether 90 possible evolutions, as specified in Figure 9.1 and Figure 9.2 in Appendix 9.1.1.

	Mean number of generations needed (with standard deviation)			
d	animal → animal	animal → object	object → animal	object → object
L_2	36.42 ± 16.23	42.47 ± 22.93	46.58 ± 23.28	45.04 ± 19.51
$SSIM$	41.25 ± 21.07	37.77 ± 22.31	44.42 ± 19.50	49.92 ± 20.99

When one compares the adversarial images created by $\text{EA}_{L_2}^{\text{target}}$ and by $\text{EA}_{SSIM}^{\text{target}}$, neither appears in general to be clearly better than the other (see Figures 4.2 and 4.4 for example, as well as Figures 9.1 and 9.2 in Appendix 9.1.1) for the human eye. Both EAs were able to produce misclassification with high confidence, while keeping the adversarial image very close to the original. In some image areas, the similarity to the original is higher with L_2, while in other image areas it is higher with $SSIM$. One should note, however, that the final aspect of the evolved image does not only depend on the used similarity measure. The randomness inherent to the evolution process may indeed contribute to the observed differences.

To summarize, the algorithm $\text{EA}_d^{\text{target}}$ (for $d = L_2$ and $d = SSIM$) achieved the objective set in the target scenario. Adversarial descendant images were constructed, which VGG16 labeled in the target category with a probability exceeding 0.95, while simultaneously remaining highly similar to their ancestors for the human eye. However, they are not entirely indistinguishable, the modified image being usually noisier than the unchanged one. Subsection 4.1.2 addresses this issue specifically.

Visualising the performed modifications: SSIM vs L_2

In order to visualise the modifications that were performed, the difference between the descendant and ancestor images was computed. An example is given in Figure 4.5, where this difference is displayed, both spatially as an image and as a plot of the difference between original and evolved pixels for the descendant.

Experiments show that the difference consists of noise distributed almost evenly across the image, with no particular area of focus. This noise is naturally not random, but fine-tuned by our EA (it has already been proven than random noise is not sufficient to deceive a CNN, since they are only vulnerable to targeted noise [22]). The majority of pixels were modified by an absolute value lower than 10. Histograms of the pixel modification values were computed for both the L_2-norm and SSIM (see Figure 4.6 for one example. Note however that the figures were averaged on six runs to reduce the possible impact of random fluctuations) for all 90 *source-target* combinations (Note that the Kullback-Leibler values given in Table 9.1 are performed only on one and not on six runs. Nevertheless, they remain small enough to indicate that the patterns are similar). The most prominent value is 0 in both histograms, corresponding to unchanged pixels.

Normalising the two histograms into probability densities, the Kullback-Leibler divergence [35] KL between them indicates the proximity of the pattern of the two sampled distributions. Since $KL(p_a||p_b) \geq 0$ is not a symmetric function of the probability distributions p_a and p_b, one needs to compute two values. It turns out that in the case of Figure 4.6, the Kullback-Leibler divergence values between the probability distribution associated with the L_2 and SSIM histograms is 0.015, while the vice-versa value is 0.016, hence providing evidence that the patterns are indeed very similar.

However, the SSIM results are more symmetrical than the L_2-norm ones (see Figure 4.6 for an example of this phenomenon). Although a human being is unlikely to perceive a difference between the descendant images obtained either with L_2 or with $SSIM$, the histograms (see again Figure 4.6 for an example) indicate that $\text{EA}_{L_2}^{\text{target}}$ tends to leave more pixels unchanged than $\text{EA}_{SSIM}^{\text{target}}$.

4.1.3 Running $\text{EA}_d^{\text{flat}}$: Examples, Results and Discussion

Figure 4.7 shows an ancestor image in the dog category (category c_6 of Table 4.1), and descendant evolved images reached by EA_d^{flat} after 238 generations with L_2, and after 233 generations with SSIM. The label values in the categories $c_1, \cdots, c_{10}$ outputted by VGG16 for these evolved images are given in row c_6 of Table 4.4 for L_2 and of Table 4.5 for SSIM. These label values are 0.100 with extremal variations of $+0.007$ and -0.017 with L_2, and $+0.023$ and -0.015 for

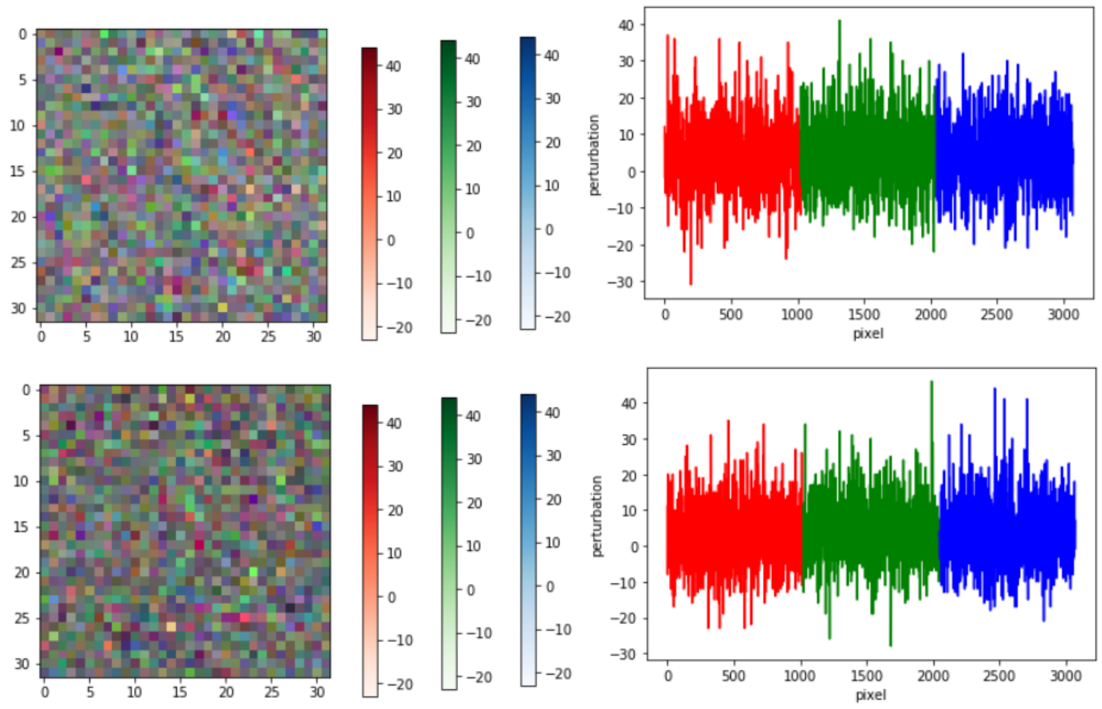


Figure 4.5: Target scenario.– Difference between ancestor and descendant images obtained by $\text{EA}_d^{\text{target}}$ ($d = L_2$ in the top row, $d = SSIM$ in the bottom row) for the *truck*→*frog* combination of Figure 4.4. The image on the left gives an idea of the spatial distribution of the changes, each pixel being a combination of three channels, the range of each being given by the scales in the middle. The plot on the right ignores the 2D structure of an image to show more clearly by how much each of the 1024 pixels is changed on the three different channels. Despite the goal oriented nature of the evolution, these changes look like noise, almost evenly distributed across the image. L_2 and SSIM exhibit slightly different patterns, with noise peaking at different pixels. For this particular combination of ancestor and descendant images, an increase in the contribution of the blue channel is observed when replacing L_2 with SSIM.

SSIM. Note that these extremal values occur for both d's for the same categories c_4 and c_5. The image pixel modifications necessary to reach these label values are visualised in Figure 4.8 for both L_2 and SSIM. The probability densities of L_2 and SSIM are highly similar, leading to Kullback-Leibler divergences of 0.004.

Figure 4.9 shows the evolution of the class probabilities outputted by VGG−16 during the flattening of the *dog* ancestor image with L_2 and SSIM. One sees that the label value of a first category, namely the category c_4, takes off (around 20 generations) while the label value of $c_{\mathcal{A}} = c_6$ decreases, so that the sum of both label values is around 1, while the label values of the other categories remain insignificant. Note that in this process, the label value of c_4 exceeds largely that of c_6. Then the label value of a second category, namely c_8, takes off (around 80 generations), while the label values of c_6 and c_4 decrease (with a similar phenomenon as before, namely the label value of the newcomer c_8 exceeds the label values of c_6 and of c_4).

Table 4.4: Flat scenario: Label values predicted by VGG−16 for the 10 different flattened images, using L_2. For any row $1 \leq i \leq 10$ one considers the adversarial descendant image created by $EA_{L_2}^{\text{flat}}$ and pictured on the i^{th} position on the 2nd row of Figure 9.3. For $1 \leq j \leq 10$, the value given on the j^{th} column is the label value for the category c_j output by VGG16 for this adversarial image. The column D_{flat} gives the value of the function D_{flat} for the descendant flat image obtained by $EA_{L_2}^{\text{flat}}$. The columns Δ^+ and Δ^- indicate the maximal deviation exceeding 0.100 from above or from below in the row.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	D_{flat}	Δ^+	Δ^-
c_1	0.102	0.115	0.102	0.083	0.111	0.113	0.092	0.092	0.096	0.093	1×10^{-3}	0.015	0.017
c_2	0.102	0.111	0.097	0.101	0.104	0.094	0.095	0.106	0.103	0.087	4×10^{-4}	0.011	0.013
c_3	0.098	0.101	0.104	0.100	0.102	0.098	0.098	0.097	0.101	0.101	4×10^{-5}	0.004	0.003
c_4	0.088	0.113	0.101	0.094	0.107	0.096	0.102	0.107	0.104	0.086	7×10^{-4}	0.013	0.017
c_5	0.100	0.099	0.100	0.100	0.100	0.100	0.100	0.101	0.100	0.100	1×10^{-6}	0.001	0.001
c_6	0.102	0.096	0.106	0.107	0.083	0.098	0.099	0.107	0.098	0.104	5×10^{-4}	0.007	0.017
c_7	0.100	0.100	0.101	0.098	0.099	0.100	0.100	0.099	0.101	0.100	8×10^{-6}	0.001	0.002
c_8	0.097	0.112	0.094	0.106	0.098	0.095	0.103	0.100	0.095	0.100	2×10^{-4}	0.012	0.006
c_9	0.105	0.092	0.101	0.086	0.103	0.104	0.103	0.102	0.103	0.100	3×10^{-4}	0.005	0.014
c_{10}	0.101	0.099	0.100	0.100	0.100	0.100	0.100	0.100	0.100	0.100	2×10^{-6}	0.001	0.001

Table 4.5: Flat scenario: Label values predicted by VGG−16 for the 10 different flattened images, using SSIM. For any row $1 \leq i \leq 10$ one considers the adversarial descendant image created by EA_{SSIM}^{flat} and pictured on the i^{th} position on the 3rd row of Figure 9.3. For $1 \leq j \leq 10$, the value given on the j^{th} column is the label value for the category c_j output by VGG16 for this adversarial image. The column D_{flat} gives the value of the function D_{flat} for the descendant flat image obtained by EA_{SSIM}^{flat}. The columns Δ^+ and Δ^- indicate the maximal deviation exceeding 0.100 from above or from below in the row.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	D_{flat}	Δ^+	Δ^-
c_1	0.106	0.114	0.105	0.082	0.105	0.113	0.090	0.091	0.101	0.093	1×10^{-3}	0.014	0.018
c_2	0.091	0.118	0.100	0.101	0.108	0.096	0.095	0.105	0.105	0.081	9×10^{-4}	0.018	0.019
c_3	0.088	0.111	0.113	0.102	0.108	0.088	0.092	0.088	0.113	0.097	1×10^{-3}	0.013	0.012
c_4	0.090	0.119	0.099	0.095	0.110	0.093	0.103	0.105	0.104	0.083	1×10^{-3}	0.019	0.017
c_5	0.102	0.095	0.092	0.114	0.114	0.093	0.084	0.092	0.106	0.109	1×10^{-3}	0.014	0.016
c_6	0.100	0.100	0.108	0.123	0.085	0.090	0.092	0.105	0.094	0.103	1×10^{-3}	0.023	0.015
c_7	0.094	0.097	0.096	0.078	0.099	0.116	0.102	0.105	0.106	0.108	1×10^{-3}	0.016	0.022
c_8	0.087	0.125	0.095	0.092	0.101	0.099	0.111	0.102	0.096	0.092	1×10^{-3}	0.025	0.014
c_9	0.102	0.086	0.108	0.078	0.102	0.107	0.100	0.106	0.109	0.104	1×10^{-3}	0.009	0.022
c_{10}	0.114	0.092	0.084	0.091	0.090	0.102	0.113	0.108	0.099	0.108	1×10^{-3}	0.014	0.016

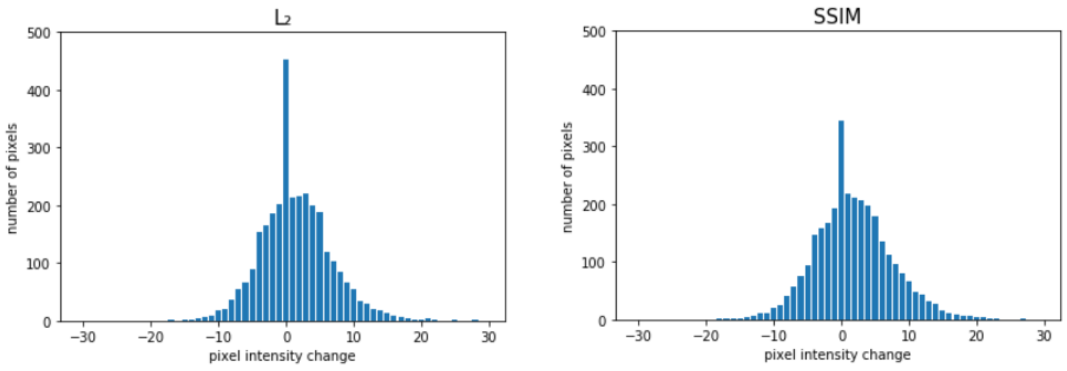


Figure 4.6: Target scenario for the *dog*$\rightarrow$*horse* evolution.– Changes in pixel intensity of the dog ancestor shown in Figure 4.2, with the L_2-norm (left) and SSIM (right). To reduce the possible impact of random fluctuations on the results, the figures are averaged on six runs. In both cases, the predominant value is 0, corresponding to a lack of pixel modification. Although both histograms are somewhat bell-shaped, SSIM's is more symmetrical. The Kullback-Leibler divergence computed between the L_2-norm and SSIM probability densities is 0.015; Reversing this order leads to 0.016.

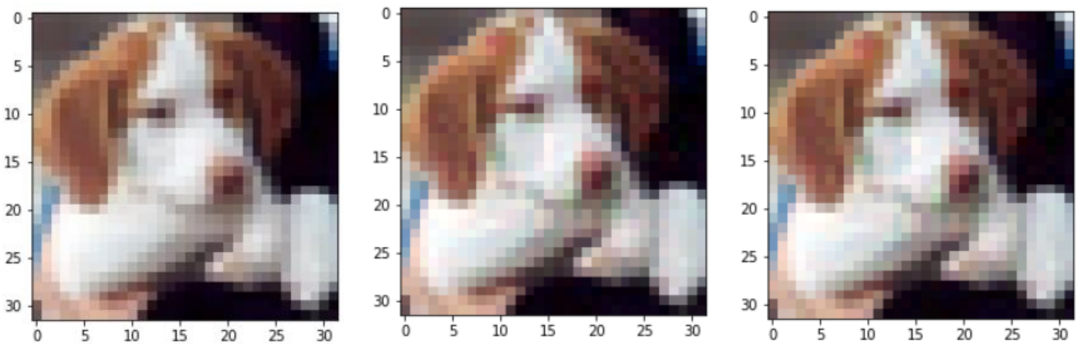


Figure 4.7: Flat scenario with the *dog* original image.– Comparison of the original (on the left) with 2 evolved pictures created by $EA^{\text{flat}}_{L_2}$ and EA^{flat}_{SSIM}. The number of generations required by the two evolutions was 238 and 233.

Results

Depending on the chosen source category, reaching almost flatness required between 142 and 552 generations for the 10 altogether evolutions, as specified in Appendix 9.1.2 (Table 9.4). The average time required per generation was 0.04 ± 0.01s with L_2 and 0.17 ± 0.02s with SSIM. For the flattening process, the horse category took the fewest number of generations, and the deer category the largest number of generations, at least with the ancestor pictures taken in these categories.

When one compares the adversarial images created with $EA^{\text{flat}}_{L_2}$ and those created with EA^{flat}_{SSIM}, neither appears better than the other (see Figure 4.7 for an example of the flattening of an ancestor in the dog category, and Figure 9.3 in Appendix 9.1.2 for examples for all categories) for the

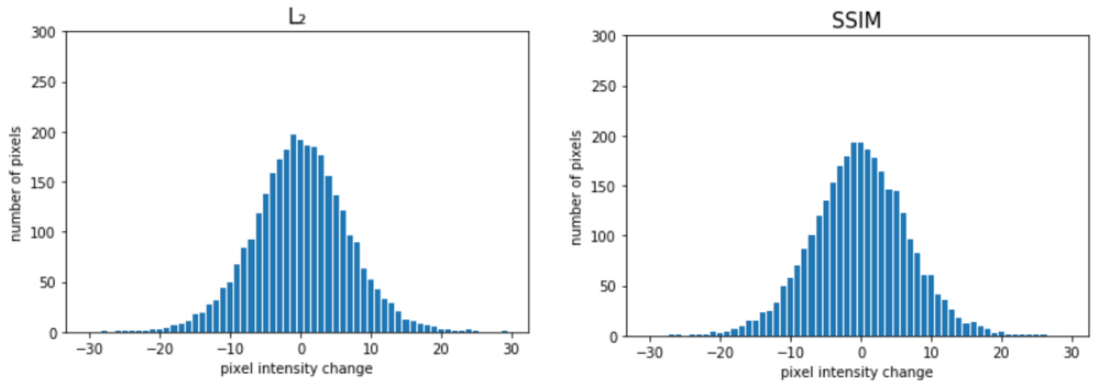


Figure 4.8: Flat scenario for the *dog* original image.– Changes in pixel intensity of the dog ancestor shown in Figure 4.7, with the L_2-norm (left) and SSIM (right). To reduce the possible impact of random fluctuations on the results, the figures are averaged on six runs. The Kullback-Leibler divergence computed between the L_2-norm and SSIM probability densities is 0.004; Reversing this order also leads to 0.004.

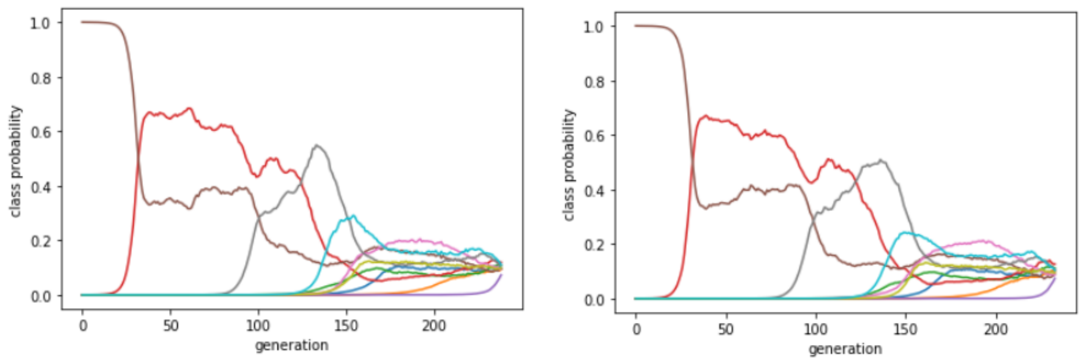


Figure 4.9: Flat scenario with the *dog* original image of Figure 4.7.– Linear scale plots of an evolution of the 10 class probabilities using the L_2 norm (left) or using SSIM (right), with respect to the generation number.

human eye. Both EAs produced adversarial images remaining very close to the ancestor image.

Tables 4.4 and 4.5 display the label values of VGG16 for the 10 flattened images of Figure 9.3 created by EA_d^{flat}, with $d = L_2$ and SSIM. The amplitudes Δ^+ and Δ^- with respect to the goal value $\frac{1}{\ell} = \frac{1}{10}$ should be interpreted with caution. On the one hand, although these amplitudes are very small in some cases (reaching ±0.001), they can achieve far larger values ($+0.015$ and -0.017 for L_2; $+0.025$ and -0.022 for SSIM). On the other hand, when one takes into account the starting points $\simeq 10^{-6}$ in most cases, of the label values of the categories distinct from the ancestor category, it is fair to consider that reaching label values so close to 0.1 modulo Δ^+ and Δ^- indeed makes our point. We nevertheless come back to this aspect in the conclusion part.

Visualizing the performed modifications

Like for the target scenario, we studied the way noise is distributed. Histograms of the pictures' modification values exhibit a bell shape, for both the L_2 norm and SSIM (see Figure 4.8 for one example, again with numbers averaged on six runs to reduce the potential impact of random fluctuations). The Kullback-Leibler divergence values computed provide again evidence that the patterns are indeed very similar. Note that the Kullback-Leibler values given in Table 9.3 are performed for all 10 possibilities of the flat scenario, but only on one and not on six runs. Therefore the values are larger than they would be on an average of six runs. Nevertheless, they remain small enough to lead to the same conclusion, namely that the patterns are similar.

Although the evolutions of the class probabilities corresponding to the 10 flattened images have different patterns, it is a general rule that during the initial generations only a few classes dominate, interchanging their order. More precisely, similar to what happens in Figure 4.9 for the flat scenario with the *dog* ancestor pictured in Figure 4.7, where the successive label values "taking off" are first those of animal categories, the first label values taking off are those of the categories which, excluding the ancestor class, rank highest in the classification of their corresponding ancestor image, thus having a higher starting point in both the L_2 and SSIM evolutions. They typically belong to the same *animal* or *object* category as the ancestor class.

4.1.4 Summary of the outcomes

Pursuing the research program announced in [6], this work substantially complements [11] by demonstrating the validity of an approach using evolutionary algorithms to produce adversarial samples that deceive neural networks performing image recognition, and that are likely to deceive humans as well. Our two evolutionary algorithms, $\mathrm{EA}_d^{\mathrm{target}}$ and $\mathrm{EA}_d^{\mathrm{flat}}$, that differ by their fitness functions, successfully fool, for two target and flat scenarios, the VGG16 [52] CNN, trained on the Cifar-10 [34] dataset to label images in 10 categories. The similarity between the adversarial images and the original ones, aiming at ensuring that humans would still classify the modified image as belonging to the original category, is measured by two distances d, namely $d = L_2$ and $d = SSIM$. These distances differ conceptually, since they assess different quality features of pairs of images. An outcome of our experiments (thanks to the computation of the Kullback-Leibler divergence values, the number of generations required, etc.) is that none seems qualitatively better than the other. Furthermore, experiments performed with L_2 tend to be 4 to 5 times faster than SSIM. Therefore, as a consequence, the choice of L_2 rather than SSIM seems a reasonable trade-off. The study shows that taking advantage of SSIM requires at least to introduce mutations that would not impact the values of L_2 and those of SSIM the same way.

4.2 Target scenario: attack against 10 CNNs trained on ImageNet

A large part of this section is extracted from [57]. This section first presents the tests that were performed to find the best population size for the EA attack. Then, the section continues by presenting the experiments and results obtained with $\text{EA}_d^{\text{target}}$ on the ImageNet dataset [18].

4.2.1 Choice of the EA's population size

The purpose of this subsection is to describe a series of tests performed on our EA with different population sizes. The outcome leads to the selection of a convenient population size for the challenges addressed in Subsections 4.2.2 and 4.2.3.

Population size according to the size of the images

We perform in this Subsection a series of tests on $\text{EA}^{\text{target},\mathcal{C}}$ with different population sizes. The main reason for these tests is that the population size of the EA impacts its efficiency, in a way that strongly depends on the size of the addressed images.

Indeed, in [11, 15], we constructed an EA (a variant of $\text{EA}^{\text{target},\mathcal{C}}$) that successfully fooled VGG16 trained on the Cifar-10 dataset. Starting with ancestor images of size $32 \times 32 \times 3$, we found that $N = 160$ was the best population size trade-off regarding the effective construction of adversarial images on the one hand, and the computation time and number of generations required to do so on the other hand.

The situation differs here, since we attack CNNs trained on the ImageNet dataset. The ancestor images are now of size $224 \times 224 \times 3$ (usually; sometimes they are even larger before being processed to fit the CNNs constraints). Said otherwise, starting with ancestor images of ImageNet size, $\text{EA}^{\text{target},\mathcal{C}}$ has to deal with a search space that is a 49-fold larger than the search space for images of Cifar-10 size. Therefore, finding the balance between achieving the goal of the construction of convenient adversarial images, and the time and number of generations required to do so, implies to adjust the population size of the EA accordingly.

Tests.— To find the optimal population size N for the threshold value $\tau = 0.75$, mutation magnitude $\delta = 1$, $A = 1$, $B = 0$ and the maximum number of generations $G = 10,000$ (these choices are consistent with the experiments performed in Subsection 4.2.3), we run $\text{EA}^{\text{target},\mathcal{C}}$ with $N = 40, 80, 120$, and 160 for $\mathcal{C}$ =VGG16 trained on the ImageNet dataset, for a series of combinations (c_a, c_t).

More precisely, we pick at random 5 pairs (c_{a_k}, c_{t_k}) $(1 \leq k \leq 5)$ of ancestor and target categories, and an ancestor image $\mathcal{A}_{a_k}$ in c_{a_k} (see Table 4.6). To increase the robustness of the results, we perform 10 independent runs for each population size with random seeds, and assess the average, over these 10 runs for each population size N, of significant indicators: average time (avgTime) in seconds, average number of generations (avgGens), average time/generation, average L_2-distance between the ancestor image and the adversarial images.

Results and interpretation.— Table 4.7 summarizes the results of these tests. Note that all runs successfully created 0.75-strong adversarial images in less than 6000 generations. Therefore, setting the maximal number of generations to 10,000 turns out to be a conservative and

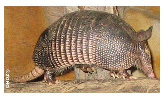
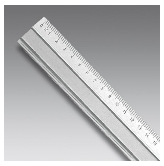
k	1	2	3	4	5
(c_{a_k}, a_k)	(hippo, 344)	(red wine, 966)	(frying pan, 567)	(armadillo, 363)	(ruler, 769)
$\mathcal{A}_{a_k}$					
c_{a_k}-label value	0.6900	0.5948	0.9999	1.000	0.96960
(c_{t_k}, t_k)	(gibbon, 368)	(banjo, 420)	(printer, 742)	(saluki, 176)	(junco,13)

Table 4.6: For $1 \leq k \leq 5$, the ancestor image $\mathcal{A}_{a_k}$, taken from the ImageNet test set, classified by VGG16 in the category c_{a_k}, with its corresponding c_{a_k}-label value. The last row indicates the chosen target category c_{t_k}.

altogether appropriate and reasonable choice for the experiments performed in this paper (see Section 4.2.3).

N	**avgTime**	**avgGens**	**time/gen**	**L_2-distance**
40	840	2957	0.283	3220
80	1747	2551	0.686	3091
120	2434	2355	1.039	3029
160	3095	2256	1.382	2983

Table 4.7: Performance comparison of $\text{EA}^{\text{target},\mathcal{C}}$ for $\mathcal{C}$ = VGG16 trained on ImageNet in creating 0.75-strong adversarial images for the target scenario (c_{a_k}, c_{t_k}) performed on $\mathcal{A}_{a_k}$, with different population sizes. The results are the average of the 5 pairs of Table 4.6 over 10 independent runs for each population size.

Table 4.8 illustrates the quality of the adversarial images obtained by our algorithm. The first image (from the left) is the ancestor image $\mathcal{A}_{a_1}$, the others are 0.75-strong adversarial images created by our EA with a population size $N = 40, 80, 120$ and 160. For $N = 40, 80$, and 120, the worst adversarial image (from a L_2 perspective) of the 10 independent runs is pictured, and, for $N = 160$, the best (still from a L_2-distance perspective) adversarial image is represented. The outcome is clear: a human is unlikely to notice any difference between any of these adversarial images and the ancestor image.

Since there is no humanly visual difference between adversarial images obtained with a population size $N = 40, 80, 120$ or $N = 160$, and since, moreover, the measures of the L_2-distances between the ancestor image and the adversarial images obtained with a population size of $N = 40$ versus a population size $N = 160$ remain very close (differing by only 8%), what really matters is the speed in creating the adversarial images.

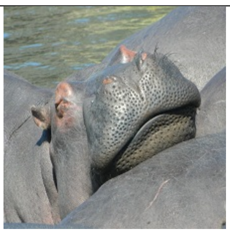
Ancestor image	Adversarial images			
	$N = 40$	$N = 80$	$N = 120$	$N = 160$
				

Table 4.8: 0.75-strong adversarial images created by $\text{EA}^{\text{target},\mathcal{C}}$ for $\mathcal{C} =$ VGG16 trainet on ImageNet, with different population sizes of $N = 40, 80, 120$, and 160.

The algorithm with $N = 40$ completes a generation in 0.283 seconds on average , almost five times (exactly 4.88) faster than with $N = 160$, and terminates within 840 seconds in average, hence more than 3.68 times faster than with $N = 160$. This speed gain (per generation, and altogether) significantly compensates the 31% increase of the number of generations required with $N = 40$ as compared with $N = 160$.

Conclusion.— A population size $N = 40$ provides an appropriate choice for $\text{EA}^{\text{target},\mathcal{C}}$ against $\mathcal{C} =$ VGG16 trained on ImageNet. We more generally extrapolate this choice of $N = 40$ for $\text{EA}^{\text{target},\mathcal{C}}$ against any CNN trained on ImageNet. In particular, we use therefore $N = 40$ in the remainder of this paper.

4.2.2 One EA versus 10 CNNs: Methodology

The generic methodology used by our EA-based attack against a series of trained CNNs is described in this Section. This provides the theoretical ground for the experiments performed in Section 4.2.3, which concretely evaluate the efficiency of $\text{EA}^{\text{target},\mathcal{C}}$ at generating within 10,000 generations adversarial images against the 10 CNNs trained on ImageNet (0.75-*strong adversarial images* or *good enough adversarial images* for the *target scenario*, or *adversarial images for the untargeted scenario*).

Specifically, Subsection 4.2.2 lists the 10 CNNs trained on ImageNet that we intend to challenge with our EA, and gives the rationals that led us to their choice. Subsection 4.2.2 explains how we obtained the (ancestor, target) category pairs, and the ancestor images. Subsection 4.2.2 describes how we intend to run $\text{EA}^{\text{target},\mathcal{C}}$ on a significant number of cases for each specific CNN $\mathcal{C}$, and defines the indicators that assess the effectiveness and quality of this EA-based attack, mainly for the target scenario, but also for the untargeted scenario.

Network Domain

We challenge $\text{EA}^{\text{target},\mathcal{C}}$ against the following 10 CNNs trained on ImageNet: $\mathcal{C}_1 =$ DenseNet121 [29], $\mathcal{C}_2 =$ DenseNet169 [29], $\mathcal{C}_3 =$ DenseNet201 [29], $\mathcal{C}_4 =$ MobileNet [28], $\mathcal{C}_5 =$ NASNetMobile [66], $\mathcal{C}_6 =$ ResNet50 [27], $\mathcal{C}_7 =$ ResNet101 [27], $\mathcal{C}_8 =$ ResNet152 [27], $\mathcal{C}_9 =$ VGG16 [52] and $\mathcal{C}_{10} =$ VGG19 [52].

$\mathcal{C}_k$	Name of the CNN	Parameters	Top-1 Accuracy	Top-5 Accuracy
$\mathcal{C}_1$	DenseNet121	8M	0.750	0.923
$\mathcal{C}_2$	DenseNet169	14M	0.762	0.932
$\mathcal{C}_3$	DenseNet201	20M	0.773	0.936
$\mathcal{C}_4$	MobileNet	4M	0.704	0.895
$\mathcal{C}_5$	NASNetMobile	4M	0.744	0.919
$\mathcal{C}_6$	ResNet50	26M	0.749	0.921
$\mathcal{C}_7$	ResNet101	45M	0.764	0.928
$\mathcal{C}_8$	ResNet152	60M	0.766	0.931
$\mathcal{C}_9$	VGG16	138M	0.713	0.901
$\mathcal{C}_{10}$	VGG19	144M	0.713	0.900

Table 4.9: The 10 CNNs trained on ImageNet, their number of parameters (in millions) and their Top-1 and Top-5 accuracy.

These 10 CNNs were chosen for the following reason. First, due to implementation considerations, we considered only CNNs that have an ImageNet pre-trained version already available in Keras [16]. Out of them, 15 handle images of size 224×224 natively, while 11 handle images of larger sizes, varying from 240×240 to 600×600.

From this list, we considered only CNNs whose implementation is stable. These considerations led us to disregard the EfficientNetB family altogether in the present study, since these CNNs are only available in the nightly build tensorflow of Keras. Lastly, being able to compare the behavior of the CNNs once exposed to $\text{EA}^{\text{target},\mathcal{C}}$ led us to restrict this study to CNNs handling images of size 224×224. This comparison criteria leaves a group of 14 CNNs (all CNNs handling images of size 224×224 except EfficientNetB0; note furthermore that the other members of the EfficientNetB family handle images of larger size).

These 14 CNNs are made of a group of 10 CNNs with different characteristics, and of a group of 4 remaining CNNs that are variants of those 10. The study is therefore limited here to the group of the 10 stable CNNs, that on the one hand handle images of equal sizes (224×224), and that on the other hand illustrate the maximal diversity in terms of characteristics and features, as illustrated in Table 4.9. In particular, the 3rd column gives the number of parameters of each CNN (in millions).

The performance of the CNNs is presented in terms of Top-1 and Top-5 accuracies (in the last two columns) for the *target scenario*. Recall that the CNN's classification satisfies the Top-1 (respectively Top-5) accuracy if the target label category exactly matches the model's prediction (respectively is one of the five best model's predictions). Based on Top-1 and Top-5 accuracy, DenseNet201 ($\mathcal{C}_3$) has the best performance, while VGG16 ($\mathcal{C}_9$), VGG19 ($\mathcal{C}_{10}$), and MobileNet ($\mathcal{C}_4$) have the worst.

Image Domain

We take at random 10 pairs (c_{a_q}, c_{t_q}) of distinct categories among the 1000 categories of ImageNet. For $1 \leq q \leq 10$, the first component c_{a_q} is the ancestor category, and the second component c_{t_q} is the target category. Then, for each ancestor category, we take at random 10 distinct images $\mathcal{A}_q^1, \cdots, \mathcal{A}_q^{10}$ from the ImageNet validation set for the specific category c_{a_q}. This process leads to 100 ancestor images $\mathcal{A}_q^p$ altogether, namely 10 for each of the 10 ancestor categories.

Table 4.10 specifies the ancestor categories and the target categories obtained that way. Figure 9.4 in Appendix 9.2.1 shows the 100 selected ancestor images, and Table 9.5 in Appendix 9.2.1 gives their c_{a_q}-label values for the 10 CNNs. The CNNs classify the ancestor images in the correct c_{a_q} category in almost 97% cases (966 out of 1000 possibilities; the remaining 34 cases are classified in a different category since the c_{a_q}-label value given by the corresponding CNN is not dominant among all categories).

q	1	2	3	4	5	6	7	8	9	10
c_{a_q}	abacus	acorn	baseball	broom	brown bear	canoe	hippopotamus	llama	maraca	mountain bike
a_q	398	988	429	462	294	472	344	355	641	671
c_{t_q}	bannister	rhinoceros beetle	ladle	dingo	pirate	Saluki	trifle	agama	conch	strainer
t_q	421	306	618	273	724	176	927	42	112	828

Table 4.10: For $1 \leq q \leq 10$, the 2nd row gives the ancestor category c_{a_q} and its index number a_q among the categories of ImageNet (Mutatis mutandis for the target categories, 3rd row).

Experiments and Indicators

For a threshold value τ and a bound G of the number of generations to be specified in the concrete experiments performed in Section 4.2.3, we run $\text{EA}^{\text{target},\mathcal{C}}$ for each $\mathcal{C} = \mathcal{C}_k$ (for $1 \leq k \leq 10$) on each ancestor $\mathcal{A}_q^p$ (for $1 \leq q \leq 10$, $1 \leq p \leq 10$). We therefore perform 100 attacks per CNN, aiming at creating, within G generations, *τ-strong adversarial images* $\mathcal{D}_k(\mathcal{A}_q^p) = \text{EA}^{\text{target},\mathcal{C}_k}(\mathcal{A}_q^p)$ for the *target scenario* (c_{a_q}, c_{t_q}) with the ancestor image $\mathcal{A}_q^p$ from the ancestor category c_{a_q}. We consider that running each of these altogether 1000 attacks (100 attacks per CNN $\times$ 10 CNNs) with one seed value is enough to make the point regarding the efficiency of our attack.

Various metrics are used to assess the effectiveness and quality of our targeted (but also untargeted) attack against each CNN. We clearly want to limit potential biases, for instance due to the specific choice of an ancestor-target pair, of a specific ancestor image, of a specific seed value in running the EA, etc. To reduce such potential issues, we focus on the mean behavior of the attack. Therefore, these metrics are (for most of them) averaged on the 100 attacks performed per CNN. In other words, these metrics aggregate for each CNN the outcomes of the attacks on the 10 ancestors per ancestor category $\times$ the 10 pairs of (ancestor, target) categories.

This leads us to define three success rates $SR_{\mathcal{C}}^{\tau}$, $SR_{\mathcal{C}}^{\text{ge}}$ and $SR_{\mathcal{C}}^{\text{untarg}}$ for a CNN $\mathcal{C}$, the two former dealing with the targeted attack and the latter with the untargeted attack.

The τ-Success Rate $SR_{\mathcal{C}}^{\tau}$ is the percentage of runs of $\text{EA}^{\text{target},\mathcal{C}}$ that successfully created at least one *τ-strong adversarial image* within G generations. The *good enough* Success Rate $SR_{\mathcal{C}}^{\text{ge}}$ is the percentage of runs of $\text{EA}^{\text{target},\mathcal{C}}$ that successfully created at least one *good enough adversarial image* within G generations, while the EA was aiming at constructing *τ-strong adversarial images*. Finally, the untargeted Success Rate $SR_{\mathcal{C}}^{\text{untarg}}$ is the percentage of runs of $\text{EA}^{\text{target},\mathcal{C}}$ that successfully created at least one *adversarial image for the untargeted attack* within G generations, while the EA was aiming at constructing *τ-strong adversarial images*. In this latter case however, one considers only runs performed on ancestor images that are classified in the ancestor category by the CNN (it indeed happens that a CNN does not classify some images in the correct category despite being chosen from the validation set, see Table 9.5).

One collects some relevant information regarding the production of *good enough adversarial images* or of *adversarial images for the untargeted attack* on the way toward the creation of *τ-strong adversarial images.* Note that the fitness function used by the EA was not designed to focus on the untargeted attack. Therefore the outcome for the untargeted attack can be seen as a by-product of the targeted attack. The inequality $SR_{\mathcal{C}}^{\tau} \leq SR_{\mathcal{C}}^{\text{ge}} \leq SR_{\mathcal{C}}^{\text{untarg}}$ generally holds (the first one does hold systematically, and the second one usually holds).

With notations consistent with Subsection 4.2.1, we measure for each CNN, the average number of generations ($\text{avgGens}_{\mathcal{C}}^{all}$) and the average time ($\text{avgTime}_{\mathcal{C}}^{all}$, in seconds) required by all attacks (successful or not). We then define similar quantities, but restricted to targeted attacks that either successfully create at least one *τ-strong adversarial image* within G generations (leading to $\text{avgGens}_{\mathcal{C}}^{\tau}$, $\text{avgTime}_{\mathcal{C}}^{\tau}$), or successfully create at least one *good enough adversarial image* within G generations (leading to $\text{avgGens}_{\mathcal{C}}^{ge}$, $\text{avgTime}_{\mathcal{C}}^{ge}$). *Mutatis mutandis*, we also consider the consistently defined quantities $\text{avgGens}_{\mathcal{C}}^{untarg}$ and $\text{avgTime}_{\mathcal{C}}^{untarg}$ for successful untargeted attacks.

For each CNN, we also report average L_2 distances, that assess the visual quality of the adversarial images obtained by successful attacks. For the target scenario, on the one hand, $\text{avg}_{\mathcal{C}}^{\tau} L_2$ is the average of the L_2 distances between the ancestor image and the *τ-strong adversarial images* created by the EA. On the other hand, $\text{avg}_{\mathcal{C}}^{ge} L_2$ is the average of the L_2 distances between the ancestor image and the *good enough adversarial images* created by the EA. For the untargeted scenario, one defines in a similar way $\text{avg}_{\mathcal{C}}^{untarg} L_2$ as the average of the L_2 distance between the ancestor image, and first adversarial image that is no longer classified as belonging to the ancestor category.

This series of indicators contribute to the assessment of the convergence characteristics of $\text{EA}_{L_2}^{\text{target},\mathcal{C}}$ for each of the 10 CNNs considered.

4.2.3 One EA versus 10 CNNs: Results

The methodology described in Section 4.2.2 is applied with parameter values $\tau = 0.75$ and $G = 10,000$.

Subsection 4.2.3 summarizes the outcomes of these experiments and gives their interpretation.

The notations used in this Subsection are consistent with those used in Subsection 4.2.2, especially regarding the indicators, given of course for $\tau = 0.75$, $\delta = 1$, $A = 1$, $B = 0$, and $G = 10,000$ (or for increasing values of the maximal number of generations up to $10,000$, see Table 9.6 in Appendix 9.2.2).

Experimental results

Table 4.11 gives the respective performance of our EA for each CNN, for the chosen parameters $\tau = 0.75$ and $G = 10,000$. The indicators are averaged over the 100 attacks per CNN, and the Table is sorted according to growing values of the average number of generations $\text{avgGens}_{\mathcal{C}}^{0.75}$ required by $\text{EA}^{\text{target},\mathcal{C}}$.

	CNNs	$\text{avgGens}_{\mathcal{C}}^{all}$	$\text{avgGens}_{\mathcal{C}}^{0.75}$	$\text{avgGens}_{\mathcal{C}}^{ge}$	$\text{avgGens}_{\mathcal{C}}^{untarg}$	$\text{avgTime}_{\mathcal{C}}^{0.75}$	$\text{avgTime}_{\mathcal{C}}^{ge}$	$\text{avgTime}_{\mathcal{C}}^{untarg}$	$\text{avg}_{\mathcal{C}}^{0.75}L_2$	$\text{avg}_{\mathcal{C}}^{ge}L_2$	$\text{avg}_{\mathcal{C}}^{untarg}L_2$
$\mathcal{C}_4$	MobileNet	2201	2122	1662	1503	562	440	398	2461	2225	2079
$\mathcal{C}_7$	ResNet101	3428	3154	2586	2550	1285	842	659	3002	2716	2377
$\mathcal{C}_2$	DenseNet169	3786	3172	2434	2329	1198	919	879	2601	2295	2179
$\mathcal{C}_3$	DenseNet201	4232	3293	2736	2410	1348	1119	984	2962	2580	2433
$\mathcal{C}_8$	ResNet152	4054	3466	2985	2385	1246	1073	930	3128	2882	2607
$\mathcal{C}_1$	DenseNet121	3999	3477	2459	2081	1192	841	712	2801	2450	2214
$\mathcal{C}_6$	ResNet50	3794	3535	2839	2050	1452	1166	979	3233	2891	2577
$\mathcal{C}_9$	VGG16	3954	3893	2999	2006	1254	965	644	3892	3429	2715
$\mathcal{C}_5$	NASNetMobile	5148	3935	3231	2495	1426	1170	902	3214	2882	2485
$\mathcal{C}_{10}$	VGG19	4244	4126	3188	2019	1370	1060	675	4024	3548	2699
	Average	3884	3417	2712	2183	1233	960	776	3132	2790	2436

Table 4.11: Performance comparison of $\text{EA}^{\text{target},\mathcal{C}}$ against each CNN, for $\tau = 0.75$ and $G = 10,000$. Results are averaged over the 100 attacks, and given in terms of number of generations, time, and L_2-distance between the ancestor and the adversarial images.

	CNNs	$SR_{\mathcal{C}}^{0.75}$	$SR_{\mathcal{C}}^{\text{ge}}$	$SR_{\mathcal{C}}^{\text{untarg}}$
$\mathcal{C}_9$	VGG16	99	100	100
$\mathcal{C}_4$	MobileNet	99	99	99
$\mathcal{C}_{10}$	VGG19	98	100	100
$\mathcal{C}_6$	ResNet50	96	99	99
$\mathcal{C}_8$	ResNet152	96	99	99
$\mathcal{C}_1$	DenseNet121	92	95	95
$\mathcal{C}_7$	ResNet101	91	98	98
$\mathcal{C}_2$	DenseNet169	91	96	97
$\mathcal{C}_3$	DenseNet201	86	96	96
$\mathcal{C}_5$	NASNetMobile	80	86	87
	Average	92.8	96.8	97.0

Table 4.12: Success rates of $\text{EA}^{\text{target},\mathcal{C}}$ for each CNN, for $\tau = 0.75$ and $G = 10,000$.

Table 4.12 gives the success rates of our attack against each CNN, and is sorted according to decreasing values of $SR_{\mathcal{C}}^{0.75}$. Table 9.6 in Appendix 9.2.2 details the progression of the success rates of $\text{EA}^{\text{target},\mathcal{C}}$ as the maximal number G of generations increases from $1,000$ up to $10,000$.

Finally, Figure 4.10 shows the convergence characteristics of $\text{EA}^{\text{target},\mathcal{C}}$ for each CNN. In a sense, each curve shows how fast $\text{EA}^{\text{target},\mathcal{C}}$ is improving the target category label value towards 0.75 throughout running generations. The running generation number is given on the horizontal axis, and the vertical axis gives the average of c_t-label values over 100 attacks for this generation. Each of the 10 curves is the result of the average runs of $\text{EA}^{\text{target},\mathcal{C}}$ over 100 attacks performed against $\mathcal{C} = \mathcal{C}_k$ for $1 \leq k \leq 10$.

Let us explain on the example of MobileNet $= \mathcal{C}_4$ how Figure 4.10 should be understood while taking into account the values of $\text{avgGens}_{\mathcal{C}}^{all}$ of Table 4.11 that includes all attacks (including those that stopped at $10,000$ generations without creating any 0.75-strong adversarial image).

For MobileNet, $\text{avgGens}_{\mathcal{C}_4}^{all} = 2201$ while the convergence characteristics of $\text{EA}^{\text{target},\mathcal{C}_4}$ gives an average target probability of 0.2604 after 2201 generations. Indeed, out of the 100 attacks, 66 successively created a 0.75-strong adversarial image in ≤ 2201 generations, 33 needed more than 2201 generations to do so, and 1 terminated without success. In particular, the c_t-label value of these 34 latter cases remained small for the 2201th generation. This explains why altogether one obtains an average c_t-label value of 0.2604 at generation 2201.

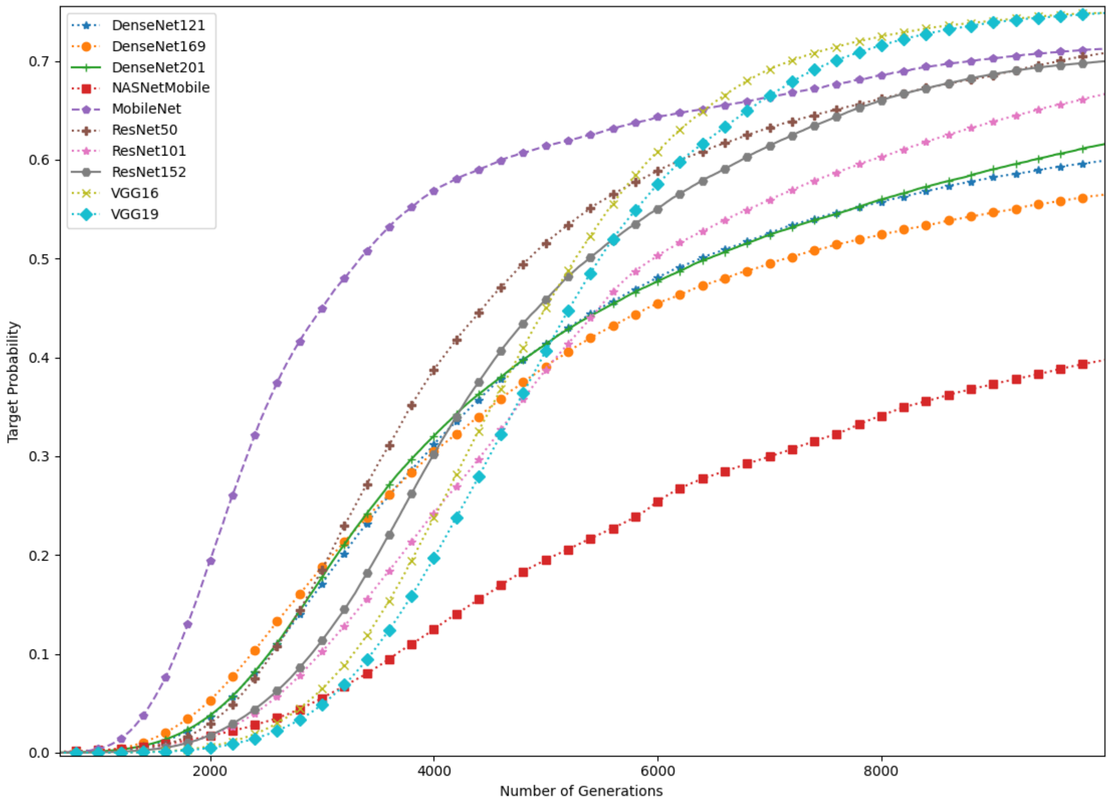


Figure 4.10: Convergence characteristics of $\text{EA}^{\text{target},\mathcal{C}}$ for each CNN. Each of the 10 curves is the result of the average runs of $\text{EA}^{\text{target},\mathcal{C}}$ over 100 attacks performed against $\mathcal{C} = \mathcal{C}_k$ for $1 \leq k \leq 10$.

Interpretation

Let us analyse the success rates of $\text{EA}^{\text{target},\mathcal{C}}$, the speed at which it creates adversarial images, and assess the visual quality of adversarial images.

Success rate of $\mathbf{EA}^{\text{target},\mathcal{C}}$.— With average success rates $\geq$ 92.8, all CNNs considered and whatever the success rate $SR_{\mathcal{C}}^{0.75}, SR_{\mathcal{C}}^{ge}$ or $SR_{\mathcal{C}}^{\text{untarg}}$ considered, experiments (see Table 4.12) clearly prove that $\text{EA}^{\text{target},\mathcal{C}}$ is highly efficient against all 10 challenged CNNs, at least when $G = 10,000$.

Table 9.6 (Appendix 9.2.2) completes the study, by showing that $\text{EA}^{\text{target},\mathcal{C}}$ is already very efficient for lower values of the maximum number G of generations taken for the termination condition. For instance, the algorithm achieves average success rates (all CNNs considered, whatever the success rate) $\geq 76\%$ already for $G = 5000$.

Obviously, these success rates of $\text{EA}^{\text{target},\mathcal{C}}$ vary with $\mathcal{C}$. Our algorithm $\text{EA}^{\text{target},\mathcal{C}}$ with $G = 10,000$ (see Table 4.12) proves particularly efficient against VGG16 and MobileNet, with success rates $\geq 99\%$, and less efficient against NASNetMobile and DenseNet201, with success rates $\geq 80\%$ still. The situation is slightly different if one restricts G to $G = 5,000$. In this case (Table 9.6, Appendix 9.2.2), MobileNet remains the most vulnerable (97%), but is followed by ResNet50 (82%) this time, while the most resistant CNNs is still NASNetMobile (58%), but followed by VGG19 (71%) this time, where the percentages given in bracket are those of $SR_{\mathcal{C}}^{0.75}$ (the others are higher).

Note that the number of parameters of a CNN does not alone explain its resistance against our attack, since the two extremes MobileNet and NASNetMobile have both 4M parameters (Table 4.9), and since a CNN with a large number of parameters, VGG16, is more exposed to our attack than another with significantly less parameters, namely DenseNet201. However, CNNs with a lower Top-1 and Top-5 accuracy appear in general easier to fool than those with larger such accuracies (although NASNetMobile seems different in this regard). Our experiments indicate an apparent correlation between Top-1 and Top-5 accuracy on the one hand, and relative resistance to our attack on the other hand. We do not state any strict causality from one phenomenon to the other though.

Speed at creating adversarial images.— Table 4.11 (completed by Table 9.6 in Appendix 9.2.2) shows that these success rates are achieved between 2122 generations in average for the fastest CNN to fool, and 4126 generations for the most resistant CNN, and overall in 3417 generations in average for a CNN of the list. Moreover, the additional effort, measured in terms of additional generations required to move from *good enough* to 0.75-*strong adversarial images*, is $25,72\%$, and it is of $24,50\%$ to move from *adversarial for the untargeted scenario* to *good enough* for a CNN in general. Said otherwise, on the way to a successful creation of 0.75-*strong adversarial images* for a given CNN, the first 56% of the total amount of generations are used to create an *adversarial image for the untargeted scenario*, about 74% are used to create a *good enough* adversarial image, and the remaining 26% are used to achieve the goal set.

In terms of computational time on the machine used, roughly 13 minutes are necessary in average to create an *adversarial image for the untargeted scenario*, 16 minutes for a *good enough adversarial image*, and 20 minutes for a 0.75-*strong enough adversarial image.*

As shown by Figure 4.10, the learning curve of $\text{EA}^{\text{target},\mathcal{C}}$ differs substantially from one CNN to another. The fastest learning curve (on the short to mid term) is achieved for MobileNet ($\mathcal{C}_4$), and the slowest (on the mid- to long term) is achieved for NasNetMobile ($\mathcal{C}_5$). Although the learning curves start very modestly for VGG16 and VGG19 ($\mathcal{C}_9$, $\mathcal{C}_{10}$) since the EA's learning curves for these two are the slowest (hence even slower than for $\mathcal{C}_5$) until to circa the 3000^{th}-generation, their slopes sharply improve, and outperform the others from the $\simeq 7000^{\text{th}}$-generation on.

Visual quality of the adversarial images.— In terms of the L_2-distance between adversarial and ancestor images, if one takes as reference point the value 2436 obtained as the average L_2 distance between such images for the *untargeted scenario* (Table 4.11), the average divergence from this value for a *good enough adversarial image* is +14%, and is +28% for a 0.75-*strong adversarial image.*

Beyond these numerical measures, we actually claim that the perturbations added by $\text{EA}^{\text{target},\mathcal{C}}$, to create adversarial images for any of the tested CNNs, are unnoticeable for a human eye (at least according to the paper authors). For instance, Figure 4.11 compares an ancestor image and the obtained adversarial images (modulo resizing) for the most difficult CNNs, namely VGG19 and NasNetMobile ($\mathcal{C}_{10}, \mathcal{C}_5$), and the easiest CNNs, namely MobileNet and ResNet101 ($\mathcal{C}_4, \mathcal{C}_7$), as assessed by the values of avgGens$^{0.75}$ in Table 4.11. More precisely, the image on the left of Figure 4.11 is $\mathcal{A}_8^{10}$, the ancestor image in the llama category c_{355} pictured in Figure 9.4 in Appendix 9.2.1. Performing EA$^{\text{target},\mathcal{C}}$ on this ancestor image for the (llama, agama) ancestor-target pair for each of these CNNs with $\tau = 0.75$ and $G = 10,000$ leads to the 3 groups of adversarial images pictured on Figure 4.11. The 1st group is composed of the first obtained 0.75-strong adversarial images, *mutatis mutandis* the 2nd group with good enough adversarial images, and the 3rd group with adversarial images for the untargeted scenario. These experiments provide evidence that a human eye is unlikely to notice any difference between these images, *a fortiori* between any of the obtained adversarial images and the ancestor one.

4.2.4 Summary of the outcomes

Here, we proved that the EA$^{\text{target},\mathcal{C}}$ evolutionary algorithm is highly efficient as a generic blackbox attack against a series of CNNs, trained on ImageNet to perform image classification. We selected 10 such CNNs, that are well-known, stable, and with diverse architectures, and challenged these CNNs both for the *untargeted scenario* and the *target scenario*. In particular, for each CNN $\mathcal{C}$ in the list, starting from an ancestor image, EA$^{\text{target},\mathcal{C}}$ aimed at creating 0.75-*strong adversarial images*, that $\mathcal{C}$ classifies in a predefined target category with probability ≥ 0.75 on the one hand, and that are indistinguishable from the ancestor for a human on the other hand. 100 attacks per CNN were performed (for 100 original images sorted by groups of 10 images into 10 different ancestor categories, and for 10 associated target categories), leading to 1000 attacks altogether. A set of meaningful indicators was designed to assess the success rate and the speed of the attack, as well as the visual quality of the adversarial images.

EA$^{\text{target},\mathcal{C}}$ achieved a success rate of 92.8% at creating such 0.75-*strong adversarial images*, and required 3417 generations in average. As a by-product, EA$^{\text{target},\mathcal{C}}$ also successfully created in 96.8% of all cases *good enough adversarial images* within 2712 generations, and in 97% of all cases *adversarial images for the untargeted scenario* within 2183 generations. On the way towards the creation of a 0.75-*strong adversarial image*, about 56% of the total amount of generations explored the search space before creating an *adversarial image for the untargeted scenario*, and 74% led to the creation of a *good enough adversarial image.*

In terms of the difficulty of the assigned task, our algorithm compares favorably to other EA-based blackbox attacks against ImageNet classifiers. For instance [55, 32, 3] either focus solely on untargeted attacks, or on targeted attacks but with no target label value threshold. The task set to EA$^{\text{target},\mathcal{C}}$, fulfilled successfully, was therefore strictly harder in this regard, since we set a minimal label value for the target category, which moreover could be parametrized at will. Actually our approach allows even more flexibility.

For instance, we can even design a targeted attack, where the main termination condition is to create an adversarial image whose label value, given by the CNN in the target category, not only dominates the others (as is currently the case with our definition of *good enough adversarial images*), but does so with a gap with the second-best label value defined at will. Aiming at 0.75-*strong adversarial images* ensures that this gap exceeds 0.5, and hence that the classification by the CNN is beyond any doubt. The variant sketched above would allow such certainty

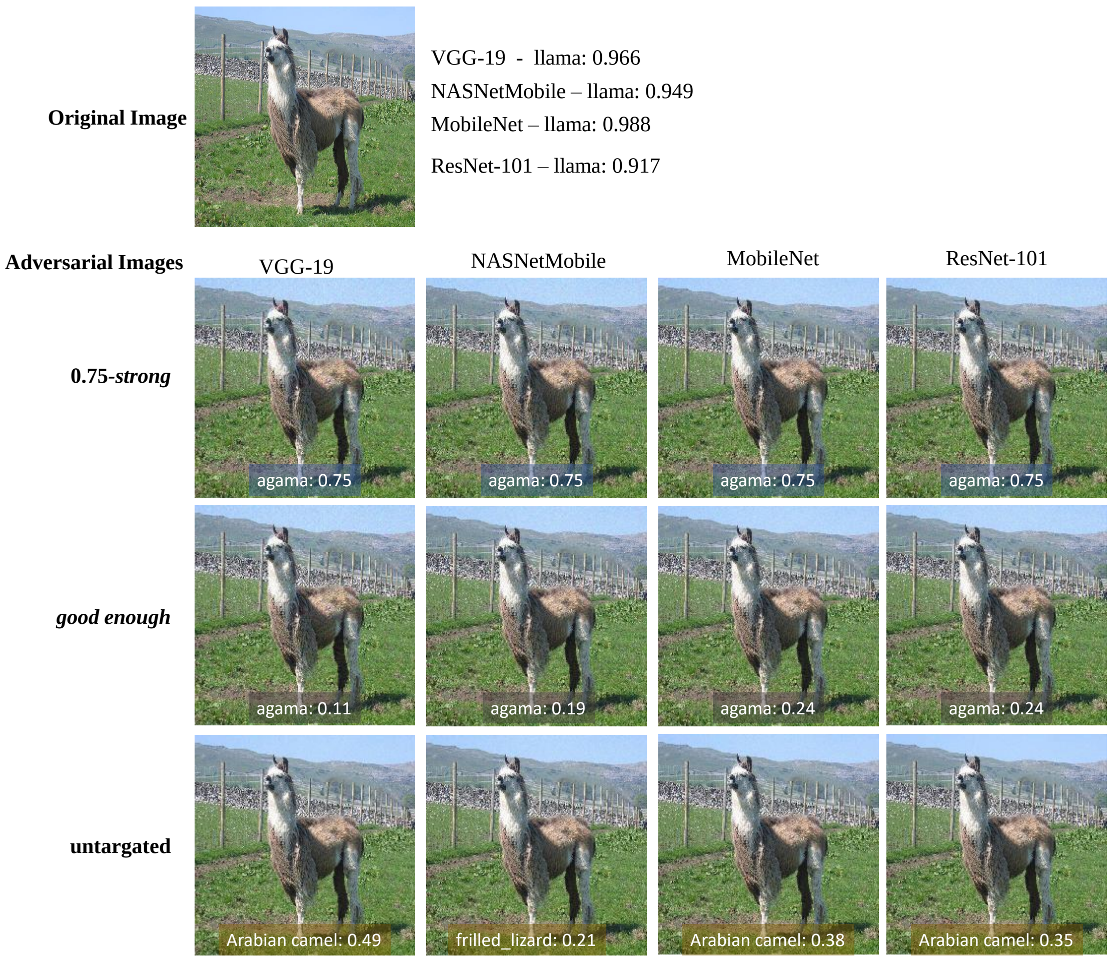


Figure 4.11: Visual comparison of ancestor and adversarial images obtained by $\text{EA}^{\text{target},\mathcal{C}}$ for the $\mathcal{C} =$ VGG19, NASNetMobile, MobileNet, and ResNet101 ($\mathcal{C} = \mathcal{C}_{10}, \mathcal{C}_5, \mathcal{C}_4, \mathcal{C}_7$), for the (llama, agama) ancestor-target pair and the ancestor image $\mathcal{A}_8^{10}$ taken from Figure 9.4 in Appendix 9.2.1.

without necessarily waiting for the creation of a 0.75-*strong adversarial image*. We performed some experiments in this direction, which we plan to further expand.

Additionally, the difficulty of the task assigned to the EA was substantially raised since we required that the perturbations added by $\text{EA}^{\text{target},\mathcal{C}}$ to create an adversarial image remained unnoticeable to humans. This is a major advantage against the adversarial images created for instance in [51, 32, 55], where a human immediately sees the introduced differences. Our EA-based attack went therefore further, since a human eye is not able to notice any difference between the ancestor and the adversarial images it constructs.

Chapter 5

Attack on High Resolution Images: Method and Performance

This chapter is extracted from [38].

5.1 Introduction

So far, the EA-based attack addressed images of moderate size, referred to here by the $\mathcal{R}$ domain, ranging from 32×32 (typically for CNNs trained on CIFAR-10) up to 224×224 (typically for CNNs trained on ImageNet), or resized to these values that the CNNs handle natively. The construction of adversarial images by adding some carefully designed adversarial noise to the potentially resized original image is illustrated in Figure 5.1.

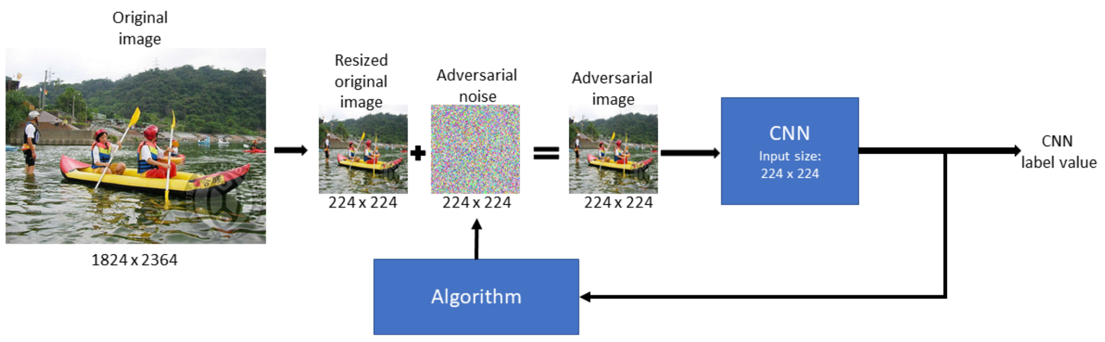


Figure 5.1: Generating an adversarial image of size 224×224.

In particular, the adversarial noise created by all these attacks is in the $\mathcal{R}$ domain handled natively by the CNNs, so that the obtained adversarial images are as large as the CNN's input size. Said otherwise, attacks in the "traditional" context create an adversarial noise of size equal to the size of the CNN input, independently on the size of the original image. This means that the size of the search space of these attacks does not depend on the size of the original image, but coincides with the size of the CNN input (note *en passant* that the smaller the input size of

the CNN, the easier the creation of adversarial noise).
However, if the adversarial image should preserve almost all the details of an original image of large size, what we call here an image in the $\mathcal{H}$ domain, in particular of a high resolution (HR) image, the adversarial noise should have the same size as the original image, and consequently the adversarial image should as well have the same size as the original one. A key point is that the adversity character of a modified image is measured only when it is exposed to the CNN, hence when it is resized to fit into the $\mathcal{R}$ domain. The adversarial character of an image should show up when the CNN proceeds to the classification of its resized version, as illustrated in the process given in Figure 5.2.

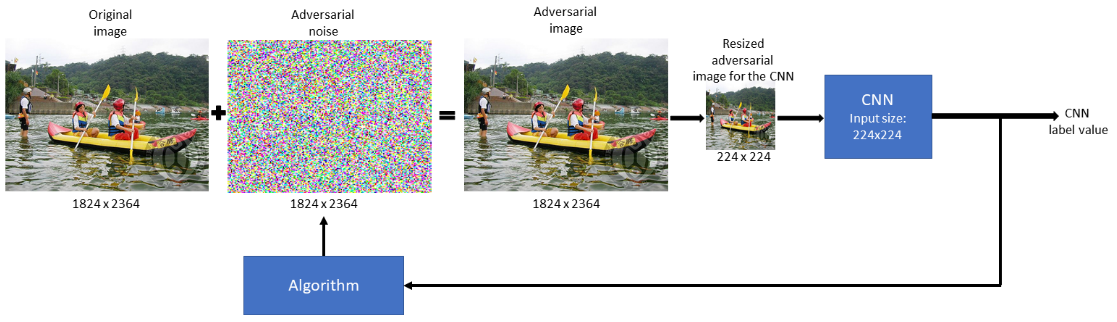


Figure 5.2: Generating adversarial images in the $\mathcal{H}$ domain.

Creating adversarial images of large size leads to three challenges in terms of speed, adversity and visual quality challenges. Firstly, the complexity of the problem increases drastically with the size of the images, as the search space for the adversarial noise grows quadratically. For instance, the noise search space provided by the original image represented in Figure 5.2 is 86 times larger than it is in the 224×224 domain. Secondly, the noise introduced in the $\mathcal{H}$ domain should be assessed as adversarial in the $\mathcal{R}$ domain: it should "survive" the resizing process to fit the CNN. In the example of Figure 5.2, it would essentially mean that it survives a 86-fold squeezing process. Thirdly, the noise introduced in the $\mathcal{H}$ domain should be imperceptible to a human eye looking at the images at their native size, and not merely once they are reduced to fit the $\mathcal{R}$ domain. For the example in Figure 5.2, it means that a human should not notice any difference between the first and the second images of size 1824×2364 when looked at full size.

Already the first challenge is a very serious one. Indeed, should it even succeed, getting directly such a HR adversarial image can take a very long time, even on a performing HPC. This is probably the reason for which, to the best of our knowledge, so far, no attack — black-box or not — has attempted to address large size images, in particular high-resolution images, by creating convenient adversarial noise in the $\mathcal{H}$ domain, so that the modified image, resized to the size handled natively by the CNN, becomes adversarial. Applying existing methods does not work, at least in reasonable time. Although efficient in the $\mathcal{R}$ domain, their extension to the $\mathcal{H}$ domain is not.
This work is a first step towards the creation of adversarial noise of size of the original image, whatever this size may be. Our contribution is essentially threefold.

Firstly, we describe an indirect attack strategy that leads to the construction of HR images in the $\mathcal{H}$ domain that are adversarial for the target scenario performed on a trained CNN (Subsection 5.3). The conceptual design of the strategy is flexible enough to lift to the $\mathcal{H}$ domain attacks considered as efficient in the $\mathcal{R}$ domain. Furthermore, it lists indicators relevant to the problem, and it describes appropriate tests to assess the behavior and the efficiency of potential resizing functions.

Secondly, we perform a feasibility study of this strategy with 10 explicit HR images and on 10 CNNs trained on ImageNet. We lift our EA-based attack to the $\mathcal{H}$ domain. We prove experimentally that our strategy is highly efficient in terms of speed and of adversity, and is reasonably efficient in terms of visual quality (Subsection 5.4). Concretely, we show that our method succeeds in 900 out of 1000 trials, that the most appropriate resizing function is the Lanczos function, and that the successful attempts require in average between $48'$ and $119.2'$ to create 0.55-strong high resolution adversarial images (and between $35.7'$ and $98.8'$ to create good enough high resolution adversarial images).

Thirdly, this study is completed by an attempt to apply the EA-based attack directly in the $\mathcal{H}$ domain (Subsection 5.5). After 48 hours of computation time, our algorithm is unable to create 0.55-strong high resolution adversarial images for any of the 10 CNNs. Although the learning curve of the algorithm improves, and although it creates images with a c_t-label value increased by a factor in the range $[1.71, 5.5]$ according to the CNN, the attack is not fast enough. These outcomes, that experimentally substantiate the seriousness of already the first challenge, are an additional argument in favour of alternative strategies like ours, in order to efficiently construct adversarial images in the $\mathcal{H}$ domain.

Two subsections and an appendix complete this article. Subsection 5.2 fixes some notations about the "lifted" version in the context of high resolution images, while Subsection 5.6 provides our findings. The Appendix contains additional evidence of our findings.

5.2 The target scenario lifted to $\mathcal{H}$

In the experiments of Subsection 5.4, we shall consider a CNN $\mathcal{C}$ that handles images of size 224×224, and that is trained on ImageNet. We denote the vector outputted by $\mathcal{C}$ as

$$\mathcal{V} = \{(c_i, v_i),\ \text{where } v_i \in]0,1] \text{ for } 1 \leq i \leq 1000\}.$$

To express the target scenario in the context of high resolution (HR) images, let $\mathcal{H}$ denote the set of images of various sizes $h \times w$, and $\mathcal{R}$ denote the set of images of size natively adapted to $\mathcal{C}$, for instance 224×224 for the specific CNN considered in Subsection 5.4. The only assumption on the size of an image $\in \mathcal{H}$ is to be larger than the CNNs input size. One assumes given a fixed *degradation function*

$$\rho : \quad \mathcal{H} \longrightarrow \mathcal{R}, \tag{5.1}$$

that transforms any image $\mathcal{I}$ of $\mathcal{H}$ into an image $\rho(\mathcal{I})$ of $\mathcal{R}$. The well-defined composition of maps

$$\begin{tikzcd} \mathcal{H} \arrow[r, "\rho"] \arrow[dr, "\mathcal{C}\circ\rho"'] & \mathcal{R} \arrow[d, "\mathcal{C}"] \\ & \mathcal{V} \end{tikzcd} \tag{5.2}$$

allows $\mathcal{C}$ to classify, in particular, the reduced image $\mathcal{A}_a = \rho(\mathcal{A}_a^{\text{hr}}) \in \mathcal{R}$ in some class c_a, with τ_a being the c_a-label value outputted by $\mathcal{C}$ for $\mathcal{A}_a$, so that $\mathcal{C}(\mathcal{A}_a) = (c_a, \tau_a)$.

In this context, an adversarial HR image for the (c_a, c_t) target scenario performed on $\mathcal{A}_a^{\text{hr}} \in \mathcal{H}$ is an image $\mathcal{D}_t^{\text{hr}}(\mathcal{A}_a^{\text{hr}}) \in \mathcal{H}$ satisfying the two following conditions. On the one hand, a human should not be able to notice any visual difference between the original $\mathcal{A}_a^{\text{hr}}$ and the adversarial $\mathcal{D}_t^{\text{hr}}(\mathcal{A}_a^{\text{hr}})$ HR images. On the other hand, $\mathcal{C}$ should classify the reduced adversarial image $\mathcal{D}_t(\mathcal{A}_a^{\text{hr}}) = \rho(\mathcal{D}_t^{\text{hr}}(\mathcal{A}_a^{\text{hr}}))$ in the category c_t for a sufficiently convincing c_t-label value. The *target scenario* (c_a, c_t) performed on the HR image $\mathcal{A}_a^{\text{hr}}$ can be visualized by the following scheme.

$$\begin{tikzcd}
\mathcal{A}_a^{\text{hr}} \in \mathcal{H} \arrow[r, dashed] \arrow[d, "\rho"'] & \mathcal{D}_t^{\text{hr}}(\mathcal{A}_a^{\text{hr}}) \in \mathcal{H} \arrow[d, "\rho"] \\
\mathcal{A}_a \in \mathcal{R} \arrow[d, "\mathcal{C}"'] & \mathcal{D}_t(\mathcal{A}_a^{\text{hr}}) \in \mathcal{R} \arrow[d, "\mathcal{C}"] \\
(c_a, \tau_a) \in \mathcal{V} & (c_t, \tau_t) \in \mathcal{V}
\end{tikzcd} \tag{5.3}$$

The image $\mathcal{D}_t^{\text{hr}}(\mathcal{A}_a^{\text{hr}}) \in \mathcal{H}$ is then a *good enough adversarial image* or a *τ-strong adversarial image* if its reduced version $\mathcal{D}_t(\mathcal{A}_a^{\text{hr}}) = \rho(\mathcal{D}_t^{\text{hr}}(\mathcal{A}_a^{\text{hr}}))$ is.

5.3 Attack strategy for the target scenario on HR images

We present here a strategy that attempts to circumvent the three challenges about speed, adversity and visual quality cited in the Introduction.

In a nutshell, the first step consists in getting an image in $\mathcal{R}$ that is adversarial against the image $\mathcal{A}_a \in \mathcal{R}$ reduced from $\mathcal{A}_a^{\text{hr}} \in \mathcal{H}$. Although getting such adversarial images in the $\mathcal{R}$ domain is crucial for obvious reasons, the strategy does not depend on how they are obtained. It applies to all possible attacks that work efficiently in the $\mathcal{R}$ domain. This feature contributes substantially to its flexibility. In a second step, one *lifts* this low-resolution adversarial image up to a high resolution image, called here the *HR tentative adversarial image*. In the last step, one checks whether this HR tentative adversarial image fulfills the criterias stated in the last paragraph of Subsection 5.2, namely becomes adversarial once reduced. A HR tentative adversarial image that does so is a HR *good enough adversarial image* or a *τ-strong adversarial image*, depending on the outcome of $\mathcal{C}$ for its reduced version in the $\mathcal{R}$ domain.

5.3.1 Construction of adversarial images in $\mathcal{H}$

The starting point is a large size image $\mathcal{A}_a^{\text{hr}} \in \mathcal{H}$, and its reduced image $\mathcal{A}_a = \rho(\mathcal{A}_a^{\text{hr}}) \in \mathcal{R}$, classified by $\mathcal{C}$ as belonging to a category c_a.

For Step 1, one assumes given an image $\widetilde{\mathcal{D}}_{t,\tilde{\tau}_t}(\mathcal{A}_a^{\text{hr}}) \in \mathcal{R}$, that is adversarial for the (c_a, c_t) target scenario performed on $\mathcal{A}_a = \rho(\mathcal{A}_a^{\text{hr}})$ for a c_t-label value exceeding a threshold $\tilde{\tau}_t$. As already stated, it does not matter how such an adversarial image is obtained.

To perform Step 2, one needs a fixed *enlarging function*

$$\lambda : \quad \mathcal{R} \longrightarrow \mathcal{H} \tag{5.4}$$

that transforms any image of $\mathcal{R}$ into an image in $\mathcal{H}$. Anticipating on Step 3, it is worthwhile noting that, although the *reduction function* ρ and the *enlarging function* λ have opposite purposes, these functions are not necessarily inverse one from the other. In other words, $\rho \circ \lambda$ and

$\lambda \circ \rho$ may differ from the identity maps $id_{\mathcal{R}}$ and $id_{\mathcal{H}}$ respectively (usually they do differ).

One applies the enlarging function λ to the low-resolution adversarial $\widetilde{\mathcal{D}}_{t,\tilde{\tau}_t}(\mathcal{A}_a^{\text{hr}}) \in \mathcal{R}$ to obtain the HR tentative adversarial image $\mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}}) = \lambda(\widetilde{\mathcal{D}}_{t,\tilde{\tau}_t}(\mathcal{A}_a^{\text{hr}})) \in \mathcal{H}$.

For Step 3, the application of the reduction function ρ on this HD tentative adversarial image creates an image $\mathcal{D}_{t,\tau_t}(\mathcal{A}_a^{\text{hr}}) = \rho(\mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}}))$ in the $\mathcal{R}$ domain. One runs $\mathcal{C}$ on $\mathcal{D}_{t,\tau_t}(\mathcal{A}_a^{\text{hr}})$ to get its classification, in the hope to obtain a classification in c_t.

The attack succeeds if $\mathcal{C}$ classifies this image in c_t, potentially for a c_t-label value exceeding the threshold value τ fixed in advance, and if a human is unable to notice any difference between the images $\mathcal{A}_a^{\text{hr}}$ and $\mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}})$ in the $\mathcal{H}$ domain.

Scheme 5.5 essentially summarizes the different steps encountered so far:

$$\begin{tikzcd}
\mathcal{A}_a^{\text{hr}} \in \mathcal{H} \arrow[rr, dashed] \arrow[d, "\rho"] & & \mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}}) \in \mathcal{H} \arrow[dr, "\rho"] & \\
\mathcal{A}_a \in \mathcal{R} \arrow[r, dashed] \arrow[d, "\mathcal{C}"] & \widetilde{\mathcal{D}}_{t,\tilde{\tau}_t}(\mathcal{A}_a^{\text{hr}}) \in \mathcal{R} \arrow[ur, "\lambda"] \arrow[d, "\mathcal{C}"'] & & \mathcal{D}_{t,\tau_t}(\mathcal{A}_a^{\text{hr}}) \in \mathcal{R} \arrow[d, "\mathcal{C}"] \\
(c_a, \tau_a) & (c_t, \tilde{\tau}_t) & & (c_t, \tau_t)
\end{tikzcd} \tag{5.5}$$

5.3.2 Indicators: the loss function $\mathcal{L}$ and L_2-distances

Although both $\widetilde{\mathcal{D}}_{t,\tilde{\tau}_t}(\mathcal{A}_a^{\text{hr}})$ and $\mathcal{D}_{t,\tau_t}(\mathcal{A}_a^{\text{hr}})$ stem from $\mathcal{A}_a^{\text{hr}}$, and belong to the same set $\mathcal{R}$ of low resolution images, these images nevertheless differ in general, since $\rho \circ \lambda \neq id_{\mathcal{R}}$ actually. This fact has two consequences that affect the design of our attack, and clarifies the adjustment described below.

On the one hand, it justifies the necessity of the verification process performed in Step 3 on the HR tentative adversarial image, namely to check whether its reduction indeed belongs to c_t. On the other hand, should it be the case, it implies as well that $\tilde{\tau}_t$ and τ_t differ. It is then natural to define the real-valued *loss function* $\mathcal{L}$ for a given $\mathcal{A}_a^{\text{hr}} \in \mathcal{H}$ as

$$\mathcal{L}(\mathcal{A}_a^{\text{hr}}) = \tilde{\tau}_t - \tau_t \tag{5.6}$$

Our attack is effective if one can set accurately the value of $\tilde{\tau}_t$ to match the inequality $\tau_t \geq \tau$ for the threshold value τ, or to make sure that $\mathcal{D}_{t,\tau_t}(\mathcal{A}_a^{\text{hr}})$ is a good enough adversarial image in the $\mathcal{R}$ domain, while controlling the distance variations between $\mathcal{A}_a^{\text{hr}}$ and the adversarial $\mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}})$. For this, one needs to assess the statistical behavior of the loss function $\mathcal{L}$ on the one hand, and of the L_2 distance of a series of images on the other hand.

Indeed, while the loss function, that measures differences of values coming from images in the $\mathcal{R}$ domain, assesses the objective of getting an image in the $\mathcal{H}$ domain that fools the CNN, other indicators assess the objective of the visual proximity between images for a human eye. Therefore, one computes the L_2 distance of 4 pairs of images. The value of $L_2(\mathcal{A}_a^{\text{hr}}, \mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}}))$, actually the most important one, is between images that live in the $\mathcal{H}$ domain. The values of

$L_2(\mathcal{A}_a, \widetilde{\mathcal{D}}_{t,\tilde{\tau}_t}(\mathcal{A}_a^{\text{hr}}))$, $L_2(\mathcal{A}_a, \mathcal{D}_{t,\tau_t}(\mathcal{A}_a^{\text{hr}}))$ and $L_2(\widetilde{\mathcal{D}}_{t,\tilde{\tau}_t}(\mathcal{A}_a^{\text{hr}}), \mathcal{D}_{t,\tau_t}(\mathcal{A}_a^{\text{hr}}))$ are for images that all live in the $\mathcal{R}$ domain.

The values of these quantities, and therefore the performances and adequacy of the resized adversarials to the addressed problem, clearly depend on the reducing and enlarging functions ρ and λ selected in the scheme.

5.3.3 Static tests with non-adversarial images natively in $\mathcal{H}$

To find out which functions ρ and λ are the most appropriate, we designed a series of tests with promising candidates. These *static* tests, called that way since they are performed with non-adversarial images, are convenient to evaluate the candidates. Scheme 5.7 shows the path of the test performed with an image $\mathcal{A}_a^{\text{hr}} \in \mathcal{H}$ as starting point, knowing that the test is performed with different ancestor images in $\mathcal{H}$, and the results are averaged among all trials.

$$\begin{tikzcd}
\mathcal{A}_a^{\text{hr}} \in \mathcal{H} \arrow[dr, "\rho"] & & \mathcal{H} \arrow[dr, "\rho"] & \\
& \mathcal{A}_a \in \mathcal{R} \arrow[ur, "\lambda"] \arrow[d, "\mathcal{C}"'] & & \mathcal{R} \arrow[d, "\mathcal{C}"] \\
& (c_a, \tilde{\tau}_a) \in \mathcal{V} & & (c_a?, \tau_a) \in \mathcal{V}
\end{tikzcd} \tag{5.7}$$

First $\mathcal{A}_a^{\text{hr}}$ is reduced to an image $\mathcal{A}_a \in \mathcal{R}$, thanks to the reduction function ρ. One obtains the classification $(c_a, \tilde{\tau}_a) = \mathcal{C} \circ \rho(\mathcal{A}_a^{\text{hr}})$. Then one resizes $\mathcal{A}_a$ first up with λ then down with ρ. One gets the classification of the resulting image $\mathcal{C} \circ \rho \circ \lambda(\mathcal{A}_a) = (c_a?, \tau_a)$, where τ_a is the c_a-label value, whether the resized image is classified to c_a or not. Note that the resized non-adversarial image obtained that way is likely to be classified in c_a. Still, the design of the test can not make this assumption *a priori*.

One evaluates the value of the loss function $\mathcal{L}(\mathcal{A}_a^{\text{hr}}) = \tilde{\tau}_a - \tau_a$, and of the distance function $L_2(\mathcal{A}_a^{\text{hr}}, \lambda \circ \rho(\mathcal{A}_a^{\text{hr}}))$.

This later value with images in $\mathcal{H}$ gives a hint at a lower bound on the expected L_2 distance between $\mathcal{A}_a^{\text{hr}}$ and the adversarial image in the $\mathcal{H}$ domain our strategy is aiming at. By construction, it is indeed unlikely that an adversarial in the $\mathcal{H}$ domain could be closer to $\mathcal{A}_a^{\text{hr}}$ than $\lambda \circ \rho(\mathcal{A}_a^{\text{hr}})$ will be. Therefore the L_2 distance of a HR adversarial to $\mathcal{A}_a^{\text{hr}}$ is likely to be $\geq L_2(\mathcal{A}_a^{\text{hr}}, \lambda \circ \rho(\mathcal{A}_a^{\text{hr}}))$, what makes this latter evaluation relevant.

5.4 Feasibility study

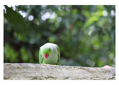
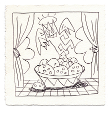
The feasibility study is performed with the 10 CNNs trained on ImageNet shown in Table 4.9, and with the 10 HR images $\mathcal{A}_1^{\text{hr}}, \cdots, \mathcal{A}_{10}^{\text{hr}}$ shown in Table 5.1. Out of them, 8 are taken from the Internet (under Creative Commons Licenses), and 2 are images from the French artist Speedy Graphito (pictured in [54], the corresponding files were graciously provided by the artist).

Table 5.1 gives the size of each original HR image, the category c_a and the c_a-label value outputted by VGG16 for $\mathcal{A}_a^{\text{hr}}$. It also provides the target category c_t, chosen at random among the categories $\neq c_a$ of ImageNet, that is used for the target scenario (c_a, c_t) to perform on each

$\mathcal{A}_a^{\text{hr}}$. Table 9.7 (in Appendix 9.3.1) completes Table 5.1 by providing, for each CNN, the corresponding c_a-categories and label values (all for the Lanczos interpolation method, as explained in Subsection 5.4.1).

One interest of adding the two specific artistic images is that, while a human may have difficulties in classifying them in any category, the CNNs do it, although with relatively small label values (see Table 5.1 for VGG16 and Table 9.7 in Appendix 9.3.1 in general).

Table 5.1: For $1 \leq a \leq 10$, the image $\mathcal{A}_a^{\text{hr}}$ classified by VGG16 in the category c_a (interpolation = "lanczos").

a	1	2	3	4	5	6	7	8	9	10
c_a	Cheetah	Eskimo Dog	Koala	Lamp Shade	Toucan	Screen	Comic Book	SportsCar	Binder	Coffee Mug
$w \times h$	910×604	960×640	910×607	2462×2913	910×607	641×600	1280×800	1280×800	1954×2011	1740×1710
$\mathcal{A}_a^{hr}$	0.95	0.34	0.99	0.53	 0.45	0.70	0.49	0.48	 0.28	0.08
c_t	poncho	goblet	Weimaraner	weevil	wombat	swing	altar	beagle	triceratops	hamper

We run the static tests to select the ρ and λ functions out of 4 candidates (Subsection 5.4.1). Then we briefly describe the evolutionary algorithm $\text{EA}^{\text{target},\mathcal{C}}$ that we shall use as a black-box attack against each of the 10 CNNs (Subsection 5.4.2). We apply the strategy with the evolutionary algorithm and get the HR adversarial images that fool CNNs for the target scenario with the threshold value set to $\tau = 0.55$ (Subsection 5.4.3). Finally, we discuss the visual quality of the obtained HR adversarial images, especially from a human point of view (Subsection 5.4.4).

For $1 \leq a \leq 10$, the HR ancestor image $\mathcal{A}_a^{\text{hr}}$, its resized version $\lambda \circ \rho(\mathcal{A}_a^{\text{hr}}) \in \mathcal{H}$ obtained by the static tests (Subsection 5.3.3), and one sample of an adversarial image $\mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}}) \in \mathcal{H}$ per (c_a, c_t) combination of the target scenario performed on VGG16, can be retrieved from https://github.com/aliotopal/HRad-versImgs/blob/main/original-advers.md.

5.4.1 Selection of ρ and λ

To select the functions ρ and λ, we evaluate four interpolation methods that convert an image from one scale to another. The Nearest Neighbor [49], the Bilinear method [2], the Bicubic method [33] and the Lanczos method [19, 47] are non-adaptive methods among the most common interpolation algorithms, with the additional advantage of being available in python librairies.

The static tests designed in Subsection 5.3.3 are performed on the 10 HR images of Table 5.1 with the 10 CNNs of Table 4.9 for all 16 possible ρ and λ combinations coming from this selection. Figure 5.3 summarizes the results in two heatmaps (see Figure 9.8 in Appendix 9.3.1 for individual heatmaps per CNN). They represent the average values (for all CNNs) of the loss function $\mathcal{L}^{\mathcal{C}}(\mathcal{A}_a^{\text{hr}}) = \tilde{\tau}_a - \tau_a$ (Figure 5.3(a)), and of $L_2(\mathcal{A}_a^{\text{hr}}, \lambda \circ \rho(\mathcal{A}_a^{\text{hr}})$ (Figure 5.3(b), the two images being in $\mathcal{H}$).

Figure 5.3(a) shows that the best performing loss value, namely 0.039 (which is quite close to the optimal 0 value), is achieved when the images are scaled down with the Bicubic method and up with the Lanczos method (observe that the Nearest Neighbor method is the default upsizing

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.090	0.093	0.100	0.092
Bicubic	0.051	0.046	0.059	0.039
Bilinear	0.063	0.059	0.076	0.056
Lanczos	0.068	0.063	0.077	0.058

(a)

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	49958	42155	39029	43277
Bicubic	42400	37542	39099	36958
Bilinear	42899	39210	40866	38560
Lanczos	42656	37113	38583	36564

(b)

Figure 5.3: The overall average values of the loss functions $\mathcal{L}^{\mathcal{C}}(\mathcal{A}_a^{\text{hr}}) = \tilde{\tau}_a - \tau_a$ in Table (a), and of $L_2(\mathcal{A}_a^{\text{hr}}, \lambda \circ \rho(\mathcal{A}_a^{\text{hr}})$ in Table (b).

and downsizing method in Keras).

However, Figure 5.3(b) shows that this combination for (ρ, λ) gives the second best L_2 distance while (ρ, λ) = (Lanczos, Lanczos) gives the best. Additionally, Figure 5.3(a) shows that the loss achieved by the (Lanczos, Lanczos) combination is the 4th best performing combination and remains very moderate.

Since human visual quality of the adversarials in the $\mathcal{H}$ domain should prevail, especially at a very tolerable cost in terms of the Loss function, we select (ρ, λ) = (Lanczos, Lanczos). This choice is used in all further experiments.

5.4.2 EA$^{\text{target},\mathcal{C}}$ parameters

The maximum pixel modification on individuals is limited to a fixed range given by $\epsilon = 16$ throughout the search process to maintain the proximity of the evolved images with the ancestor image. The step size per selected pixel is set to $\delta = 1$, the population size is set to 40 and the maximum number of generations is set to $G = 35000$. Additionally, we set $A = 1$ and $B = 0$.

5.4.3 Running the strategy to get adversarial images with the EA

With the rescaling functions (ρ, λ) = (Lanczos, Lanczos), we deploy the strategy detailed in Subsection 5.3.1 with the evolutionary algorithm EA$^{\text{target},\mathcal{C}}$ for the 10 CNNs and the 10 ancestor images $\mathcal{A}_a^{hr}$. The goal is to create 0.55-strong HR adversarial images as well as good enough HR adversarial images for the target scenario (c_a, c_t) specified in Table 5.1 (see also Table 9.7).

Since different seed values for the EA may lead to different results, we increased the robustness of the outcomes by performing 10 independent runs with random seeds for each (c_a, c_t) pair and ancestor $\mathcal{A}_a^{\text{hr}}$, leading to altogether 100 trials per CNN, hence to 1000 trials altogether.

Table 5.2: Average performance over the successful runs of $\text{EA}^{\text{target},\mathcal{C}}$ for each $\mathcal{C}$ trained on ImageNet in creating 0.55-*strong* and *good enough* HR adversarial images for the target scenario (c_a, c_t) performed on $\mathcal{A}_a^{hr}$.

	CNNs	$avgGens_{\mathcal{C}}^{ge}$	$avg_{\mathcal{C},\tau_t}^{ge}$	$avgGen_{\mathcal{C}}^{0.55}$	$AddE_{\mathcal{C},ge}^{0.55}$	$\text{avgTime}_{\mathcal{C}}$	$\text{avgTime}_{\mathcal{C}}^{ge}$	$\text{avgTime}_{\mathcal{C}}^{0.55}$
$\mathcal{C}_1$	DenseNet121	4561	0.150	7765	70.2	0.532	40.5	68.9
$\mathcal{C}_2$	DenseNet169	8112	0.241	11221	38.3	0.608	82.2	113.7
$\mathcal{C}_3$	DenseNet201	5288	0.166	8077	52.7	0.609	53.7	82.0
$\mathcal{C}_4$	MobileNet	4201	0.191	5640	34.9	0.510	35.7	48.0
$\mathcal{C}_5$	NASNetMobile	10765	0.224	12981	20.6	0.550	98.8	119.2
$\mathcal{C}_6$	ResNet50	4336	0.142	5891	35.9	0.575	41.6	56.5
$\mathcal{C}_7$	ResNet101	6261	0.151	8656	38.3	0.578	60.4	83.5
$\mathcal{C}_8$	ResNet152	6268	0.143	8477	35.2	0.649	67.8	91.8
$\mathcal{C}_9$	VGG16	4069	0.112	6250	53.6	0.567	38.5	59.1
$\mathcal{C}_{10}$	VGG19	5683	0.109	8180	43.9	0.570	54.0	77.7
	Overall Avg.	**5954**	**0.163**	**8314**	**39.6**	**0.575**	**57.3**	**80.7**

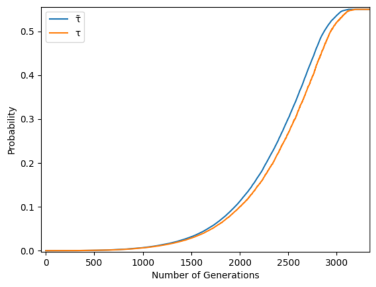
90% of the runs terminated successfully in less than $35,000$ generations. The detailed success rate for each CNN is shown in Table 9.8 (Appendix 9.3.1).

For each CNN, Table 5.2 gives the average of four indicators, computed over the successful runs for the specific CNN considered. $avgGens_{\mathcal{C}}^{0.55}$ is the average number of generations required to obtain the 0.55-strong adversarial images $\mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}}) \in \mathcal{H}$, $avgGens_{\mathcal{C}}^{ge}$ is the average number of generations required to obtain *good enough adversarial* HR images $\mathcal{D}_{t,\tau_t}^{hr,ge}(\mathcal{A}^{\text{hr}})$ while being on the way to 0.55-strong adversarial images, and $avg_{\mathcal{C},\tau_t}^{ge}$ is their average c_t-label values. The last indicator $AddE_{\mathcal{C},ge}^{0.55}$ shows the additional effort to move up from a *good enough* HR adversarial image, to a 0.55-*strong* HR adversarial image, measured as a percentage assessing the proportion of additional generations required.

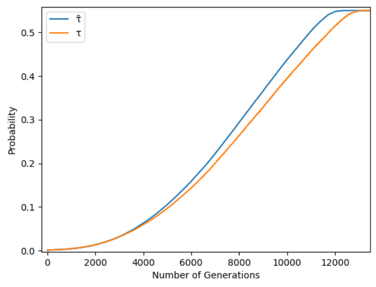
The three last columns of Table 5.2 contain the average computational time per generation ($\text{avgTime}_{\mathcal{C}}$, in second), the average total computational time required to create a good enough adversarial image ($\text{avgTime}_{\mathcal{C}}^{ge}$, in minutes) and the average total computational time required to create a 0.55-strong adversarial image ($\text{avgTime}_{\mathcal{C}}^{0.55}$, in minutes).
Out of the 900 successful trials from 1000 attempts, Table 5.2 shows that, on average, *good enough* HR adversarial images are created by our algorithm in 5954 generations, and 0.55-*strong* HR adversarial images in 8314 generations (of course with large variations, depending on the CNN considered). Measured by the number of additional generations required, the effort necessary to move up from a *good enough* HR adversarial image, that has a c_t-label value of 0.163 in average, to a 0.55-*strong* HR adversarial image is 39.6%.

In terms of the average computational time (on the hardware specified at the beginning of this article), roughly 57 minutes were necessary to create a *good enough adversarial image*, and 80 minutes for a 0.55-*strong adversarial image*, again with large variations from one CNN to another.

For each ancestor image $\mathcal{A}_a^{\text{hr}}$ for which the algorithm succeeds at least once, one computes the convergence characteristics of the algorithm $\text{EA}^{\text{target},\mathcal{C}}$ for $\tilde{\tau}_t$ and for τ_t on the way to the HR 0.55-*strong adversarial image* $\mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}})$.



(a) $\mathcal{A}_7^{\text{hr}}$



(b) $\mathcal{A}_{10}^{\text{hr}}$

Figure 5.4: Convergence characteristics for τ_t and $\tilde{\tau}_t$ for $\mathcal{A}_7^{\text{hr}}$ (a) and $\mathcal{A}_{10}^{\text{hr}}$ (b) of $\text{EA}^{\text{target},\mathcal{C}}$ for $\mathcal{C} = \text{VGG16}$.

Table 5.3: Average of the minimum and maximum values of $\mathcal{L}(\mathcal{A}_a^{\text{hr}}) = \tilde{\tau}_t - \tau_t$.

	CNNs	Avg. Loss$_\mathcal{C}$ (min)	Avg. Loss$_\mathcal{C}$ (max)
$\mathcal{C}_1$	DenseNet121	-2.09E-04	2.87E-01
$\mathcal{C}_2$	DenseNet169	-3.96E-05	3.58E-01
$\mathcal{C}_3$	DenseNet201	-1.28E-05	3.25E-01
$\mathcal{C}_4$	MobileNet	-4.50E-06	3.32E-01
$\mathcal{C}_5$	NASNetMobile	-2.89E-06	3.48E-01
$\mathcal{C}_6$	ResNet50	-2.45E-05	2.18E-01
$\mathcal{C}_7$	ResNet101	-2.31E-05	2.13E-01
$\mathcal{C}_8$	ResNet152	-1.53E-05	1.96E-01
$\mathcal{C}_9$	VGG16	-7.05E-04	3.94E-02
$\mathcal{C}_{10}$	VGG19	-1.30E-03	4.00E-02
	Overall Avg.	**-2.34E-04**	**2.32E-01**

An example, representative of the overall behavior (see Appendix 9.3.1, Figures 9.9 and 9.10), is given for VGG16 in Figure 5.4 for $\mathcal{A}_7^{\text{hr}}$, and for $\mathcal{A}_{10}^{\text{hr}}$, where the graphs are capped on the horizontal axis at their respective $avgGens_{\mathcal{C}_9}^{0.55}$ values.

Table 5.3 completes the information provided by the convergence graphs. It gives the average, over the successful among the 10 independent runs per ancestor image, of the minimum and maximum values of the loss function $\mathcal{L}(\mathcal{A}_a^{\text{hr}}) = \tilde{\tau}_t - \tau_t$.

A thorough study of the loss function as the algorithm proceeds, generation for generation, towards the construction of the HR 0.55-*strong adversarial image* $\mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}})$, shows the following outcome, at least for the successful runs performed in this study (see Appendix 9.3.2, Figure 9.11 for one detailed example). During the first generations, the values of the loss function are alternatively positive and negative, and remain very small, typically of order 10^{-4}. Then, at some point, namely from some generation on (that differs from one HR ancestor image to another, and from one CNN to another as well), the loss function becomes ≥ 0, and remains so until the algorithm terminates. Moreover, although some slight fluctuations occur, the asymptotic behavior of the loss function is to almost strictly grow from there on.

Table 5.4: The three distances L_2^1, L_2^2, and L_2^3 of images in the $\mathcal{R}$ domain, and the distance L_2^4 in the $\mathcal{H}$ domain.

	CNNs	L_2^1	L_2^2	L_2^3	L_2^4
$\mathcal{C}_1$	DenseNet121	2357	2266	4096	28112
$\mathcal{C}_2$	DenseNet169	2122	2204	1529	33355
$\mathcal{C}_3$	DenseNet201	2392	2468	1593	35439
$\mathcal{C}_4$	MobileNet	2182	2255	1463	33437
$\mathcal{C}_5$	NASNetMobile	2610	2562	1641	28501
$\mathcal{C}_6$	ResNet50	2631	2485	1426	28040
$\mathcal{C}_7$	ResNet101	2724	2620	1626	34568
$\mathcal{C}_8$	ResNet152	2771	2649	1665	34683
$\mathcal{C}_9$	VGG16	3211	2951	1485	35424
$\mathcal{C}_{10}$	VGG19	3227	3009	1490	35428
	Overall Avg.	**2623**	**2547**	**1801**	**32699**

A consequence of the convergence graphs given in Figures 9.9 and 9.10 and of the numerical values given in Table 5.3 is that setting a threshold c_t-label value $\tilde{\tau}_t = \tau_t + \text{Avg.Loss}_{\mathcal{C}}(\text{max})$ seems a reasonable choice, at least if one aims at getting 0.55-*strong* HR adversarial images by our method. A safer choice would be to add a value exceeding slightly the absolute maximum value of the loss function among all such values for all 10 ancestor images. For VGG16 for instance, it would mean to set the threshold c_t-label value to $\tilde{\tau}_t = \tau_t + 0.065$ since the largest $\mathcal{L}_{max}$ value is 0.064 for that CNN. However, for some CNNs, these values vary largely from one ancestor image to another, so that, in a first approach, we would recommend to add the average loss function instead.

5.4.4 Visual quality

We first assess numerically the quality of the obtained HR adversarial images as compared to the HR ancestors. Table 5.4 gives the three L_2 differences of images in the $\mathcal{R}$ domain, namely $L_2^1 = L_2(\mathcal{A}_a, \widetilde{\mathcal{D}}_{t,\tilde{\tau}_t}(\mathcal{A}_a))$, $L_2^2 = L_2(\mathcal{A}_a, \mathcal{D}_{t,\tau_t}(\mathcal{A}_a))$, and $L_2^3 = L_2(\widetilde{\mathcal{D}}_{t,\tilde{\tau}_t}(\mathcal{A}_a), \mathcal{D}_{t,\tau_t}(\mathcal{A}_a))$, and the L_2 difference (in the $\mathcal{H}$ domain) $L_2^4 = L_2(\mathcal{A}_a^{\text{hr}}, \mathcal{D}_{t,\tau_t}^{\text{hr}}(\mathcal{A}_a^{\text{hr}}))$.

The most saying outcome of Table 5.4 is that the average value of the L_2 distance between the HR ancestor and adversarial images remains comparable, actually even smaller, than the corresponding value (namely for Lanczos-Lanczos) measured for non-adversarial images in the heatmap in Figure 5.3(b). In other words, at least in average, our attack does not arm the numerical performance of the resizing functions. It even enhances it, what is probably due to some statistical artefact.

Still, the "true" visual quality for a human eye is assessed by looking at some representative examples either from some distance, or by zooming on some areas.

For instance, let us consider the HR ancestor image $\mathcal{A}_7^{\text{hr}}$ represented in Figure 5.5a, and a zoom of that picture on some restricted area (taken at random). Figure 5.5b shows the non-adversarial resized image $\lambda \circ \rho(\mathcal{A}_7^{\text{hr}})$ with $(\lambda, \rho) =$ (Lanczos, Lanczos). Finally, Figure 5.5c shows the HR 0.55-*strong* adversarial image created by $\text{EA}^{\text{target},\mathcal{C}}$ for $\mathcal{C} =$ VGG16. To further illustrate the phenomenon, we proceed similarly (still for VGG16) with another ancestor HR image, namely $\mathcal{A}_{10}^{\text{hr}}$ in Figures 5.6a, 5.6b, and 5.6c.

At some distance, both the non-adversarial resized original image and the HR adversarial seem to have a good visual quality as compared to the HR ancestor. However, the zoomed areas show that details from the HR ancestor images become blurry for a human eye, not only in the HR adversarial images (as seen from Figures 5.5c and 5.6c) but in the non-adversarial resized images as well (as seen from Figures 5.5b and 5.6b). Moreover, a human eye is not able to distinguish the blurriness that occurs in the non-adversarial resized image from the one that shows up in the HR adversarial: the loss of details looks the same in both cases.

This experiment, representative of the general behavior over the CNNs, shows that the observed blurry effect is not due to an inefficiency of our strategy, nor of the algorithm $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$, at least to a large extent, but is due to the lack of high-quality interpolation methods. Indeed, these experiments show that scaling up to the $\mathcal{H}$ domain images belonging to the $\mathcal{R}$ domain, adversarial or not, results in a loss of high-frequency features on the up-scaled images. Moreover, the very fact that the loss of details looks the same for a resized non-adversarial image as for the adversarial image created by our algorithm in the $\mathcal{H}$ domain speaks in favor of our attack, since it makes our attack harder to detect.

5.5 Direct attack in the $\mathcal{H}$ domain

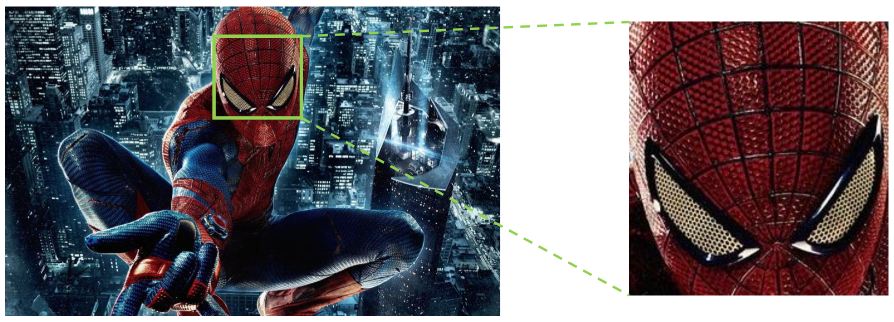
In this last part, we show that a direct attack in the $\mathcal{H}$ domain, that would aim at making effective the top arrow of scheme 5.3 without applying our indirect strategy, is a non-trivial problem in practice.

Concretely, for each $\mathcal{C} = \mathcal{C}_1 \cdots, \mathcal{C}_{10}$, we challenge $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$ to perform a direct attack in the $\mathcal{H}$ domain for the most promising (ancestor, target) pair and the corresponding ancestor image $\mathcal{A}_a^{\mathrm{hr}}$, in order to create directly a 0.55-strong HR adversarial image. In all cases, the process stops when either a direct attack turns out to be successful, or if the computing time exceeds 48 hours. The most promising pair, and the corresponding ancestor, is defined as the combination for which the *indirect attack* with the algorithm $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$ is the fastest in terms of the number of generations required to succeed.

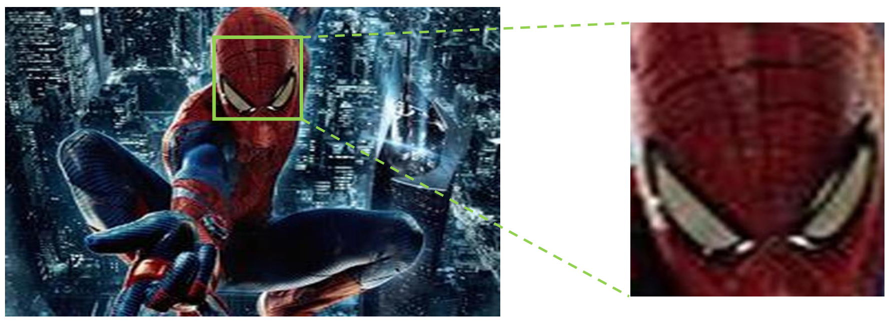
Computation shows that the (toucan, wombat) pair, with the corresponding ancestor image $\mathcal{A}_5^{\mathrm{hr}}$, is the most promising for $\mathcal{C}_4, \mathcal{C}_9, \mathcal{C}_{10}$, and that the (comic book, altar) pair, with the corresponding ancestor image $\mathcal{A}_7^{\mathrm{hr}}$, is the most promising for the 7 remaining CNNs.

Clearly, $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$ goes beyond the previous experiments since it now processes a search space of size 910×607 in the case of $\mathcal{A}_5^{\mathrm{hr}}$ and of 1280×800 in the case of $\mathcal{A}_7^{\mathrm{hr}}$, instead of 224×224 for the indirect attack.

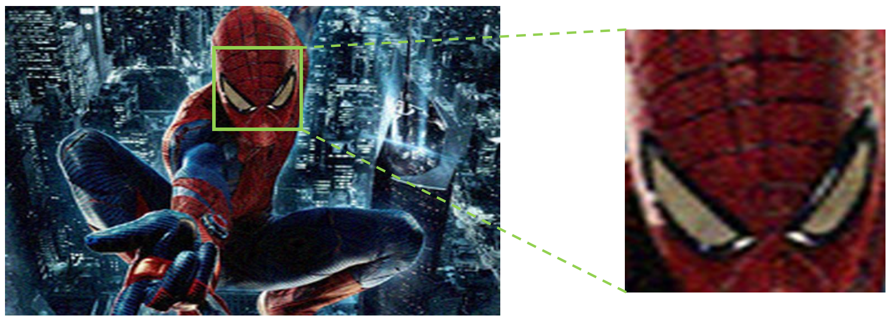
Figure 5.7 illustrates the convergence characteristics of $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$ when working directly in the $\mathcal{H}$ domain, at least for the combinations and ancestor images considered (see Figure 9.12 in Appendix 9.3.3 for all 10 CNNs). Figure 5.7(a) shows the outcome for $\mathcal{C}$ = VGG16 when one proceeds with the ancestor image $\mathcal{A}_5^{\mathrm{hr}}$, and Figure 5.7(b) shows the outcome for $\mathcal{C}$ = ResNet152 when one proceeds with the ancestor image $\mathcal{A}_7^{\mathrm{hr}}$. The horizontal axis of the graph is the number of generations, capped at what one gets after 48 hours, and the vertical axis is the c_t-label value for the fittest individual.



(a) $\mathcal{A}_7^{\mathrm{hr}}$

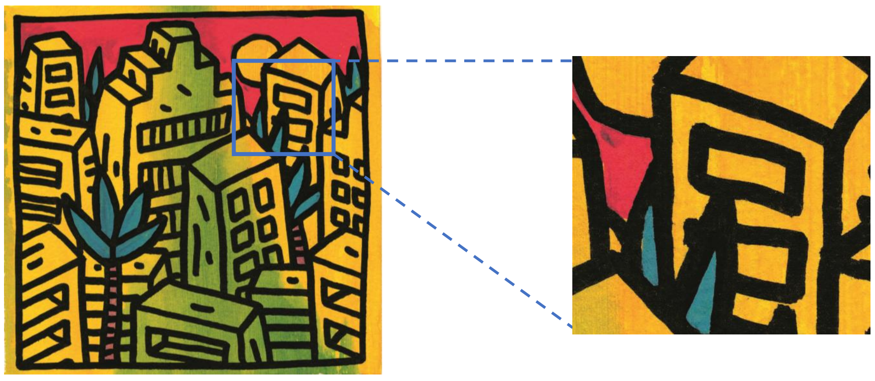


(b) $\lambda \boldsymbol{o} \rho(\mathcal{A}_7^{\mathrm{hr}})$

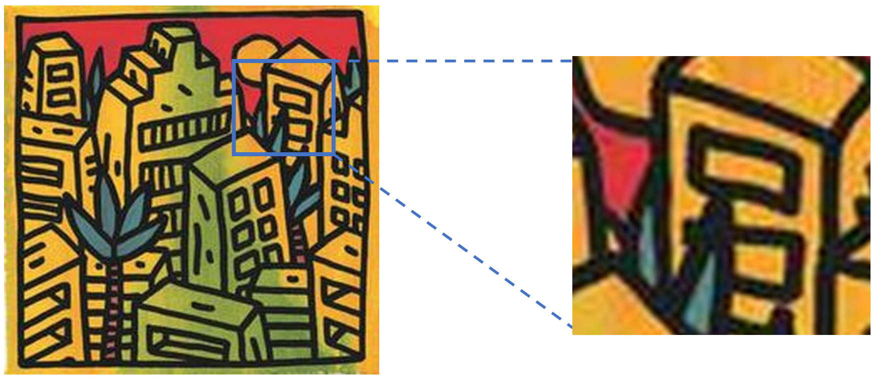


(c) $\mathcal{D}_\tau^{\mathrm{hr}}(\mathcal{A}_7^{\mathrm{hr}})$

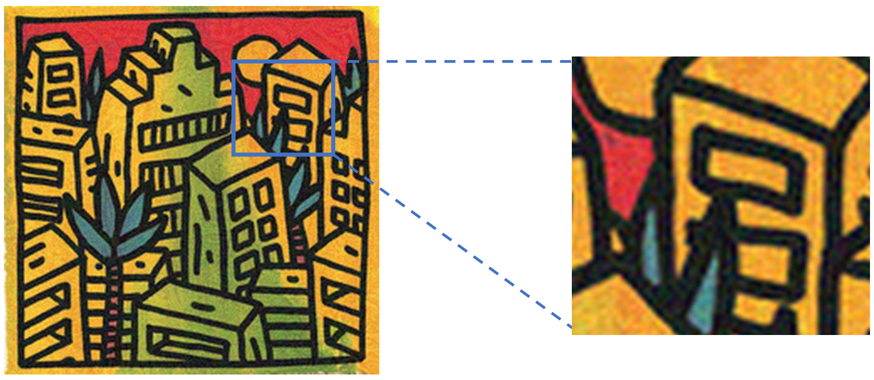
Figure 5.5: Visual comparison in the $\mathcal{H}$ domain of $\mathcal{A}_7^{\mathrm{hr}}$ (a) with its non-adversarial resized version (b) and its adversarial obtained by $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$ for $\mathcal{C} = $ VGG16.



(a) $\mathcal{A}_{10}^{\text{hr}}$

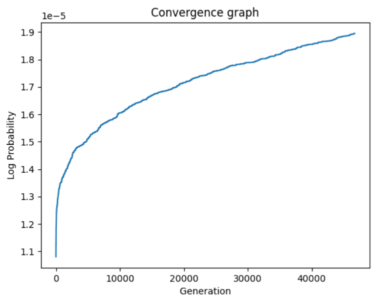


(b) $\lambda \boldsymbol{o} \rho(\mathcal{A}_{10}^{\text{hr}})$

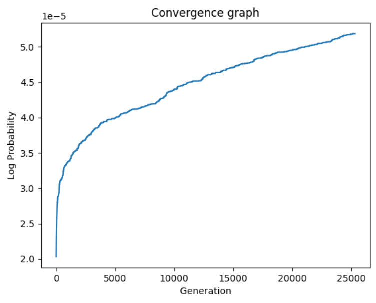


(c) $\mathcal{D}_{\tau}^{\text{hr}}(\mathcal{A}_{10}^{\text{hr}})$

Figure 5.6: Visual comparison in the $\mathcal{H}$ domain of $\mathcal{A}_{10}^{\text{hr}}$ (a) with its non-adversarial resized version (b) and its adversarial obtained by $\text{EA}^{\text{target},\mathcal{C}}$ for $\mathcal{C} = \text{VGG16}$.



(a) VGG16 - $\mathcal{A}_5^{\text{hr}}$



(b) ResNet152 - $\mathcal{A}_7^{\text{hr}}$

Figure 5.7: Convergence characteristics of $\text{EA}^{\text{target},\mathcal{C}}$ aiming at generating within 48 hours a high-resolution adversarial image by directly evolving (a) $\mathcal{A}_5^{\text{hr}}$ for the (toucan, wombat) pair and $\mathcal{C} = \text{VGG16}$, and (b) $\mathcal{A}_7^{\text{hr}}$ for the (comic book, altar) and $\mathcal{C} = \text{ResNet152}$.

Although the search space increased by around "only" 11 times for $\mathcal{A}_5^{\text{hr}}$ and 20 times for $\mathcal{A}_7^{\text{hr}}$, the EA was nevertheless unable to create high-resolution adversarial images within 48 hours, as shown in Table 9.9, Appendix 9.3.3. The EA stopped at $\approx 50,000$ generations for the 3 CNNs considered in the former case, at $\approx 28,000$ generations for the 7 CNNs considered in the later case, with the fittest individual obtained still classified by the corresponding CNN as belonging to the ancestor category (toucan or comic book).

More precisely, the c_a-label value of the fittest individual takes values in the range $\approx [0.084, 0.748]$, the actual values depending on the CNN considered. Its target category label value remains very small, culminating at $7.0E-04$ in the best case, achieved by $\mathcal{C}_1$, one of the 7 CNNs considered for the (comic book, altar) pair.

Although the learning curve of the EA improves (see Figure 5.7 and Figure 9.12 in Appendix 9.3.3), the c_t-label value of the fittest individual increases by a factor in the range $[1.71, 5.5]$ depending on the considered CNN (see Table 9.9, Appendix 9.3.3), and the critical regions to modify are narrowed down as the EA works, the EA is not fast enough to create images that converge to the target category in reasonable time. Although difficult to assess precisely, our experiments indicate that attacking directly in the $\mathcal{H}$ domain may take weeks or maybe months to succeed. It may also come out that even reaching the threshold c_t-label value of 0.55 may be out of reach in some cases by such a direct attack.

The reasons for this slowness are twofold. On the one hand, a search space of between 11 to 20 times larger than the size 224×224, for which $\text{EA}^{\text{target},\mathcal{C}}$ has proven to be efficent, makes it dicf-ficult for the EA to narrow down quickly the regions on which to focus. On the second hand, the average time per generation, that was ≈ 0.575 seconds in the $\mathcal{R}$ domain, is now ≈ 5.74 seconds in the $\mathcal{H}$ domain. Out of the operations purely linked to the EA, Table 9.10 and 9.11 (Appendix 9.3.3) show that the most consuming one is the mutation process, and that this operation of the algorithm consumes $3\times$ more time in the $\mathcal{H}$ domain than it used to take in the $\mathcal{R}$ domain. Although with a lesser timing effect, the crossover operation of the algorithm also consumes $3\times$ more time in the $\mathcal{H}$ domain than in the $\mathcal{R}$ domain. This again is due to the size of the images given to the EA.

Therefore, creating high-resolution adversarial images from $\mathcal{A}^{\mathrm{hr}}$ directly in the $\mathcal{H}$ domain requires new methods. The results of this subsection also sustain, in a way, the indirect strategy adopted in the remaining of this paper to address high-resolution images.

5.6 Summary of the outcomes

Trained CNNs performing image recognition convert input images to some fixed and moderate size, say 224×224 for CNNs trained on ImageNet typically. This process transforms the input image into a low resolution image that the CNN is able to analyze. So far, attacks, aiming at creating adversarial images fooling these CNNs, create some adversarial noise of size equal to the input size of the CNN.

The method presented in this work is the first effective attempt to make the search space for the adversarial noise depend on the size of the original image, and not on the CNN's input size. In particular, it is effective for high resolution images in terms of speed, adversity and visual quality.

More specifically, the designed indirect strategy lifts any existing attack, efficient in the low resolution domain, to an attack that applies in the high resolution domain. We performed an experimental study for 10 CNNs trained on ImageNet, by lifting our EA-based attack $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$, with the aim to create high resolution images adversarial for the target scenario, that these CNNs classify in the target category with confidence ≥ 0.55. Our algorithm succeeded in 900 cases out of 1000 attempts to create 0.55-strong high resolution adversarial images.

To sustain this indirect strategy, we also showed that attacking directly in the high resolution domain is not feasible in practice. After 48 computation hours, no high resolution adversarial image was obtained by the direct attack for any of the 10 CNNs, even for the most promising pairs of target and ancestor categories and corresponding ancestor. *A contrario*, for the 900 successful attempts, our indirect attack succeeded to create 0.55-strong adversarial images within, in average, $48'$ for the easiest CNN to fool, and $119'$ for the hardest CNN to fool.

Chapter 6

Robustness of Attack Against Filters

This chapter is mostly extracted from [15].

6.1 Introduction

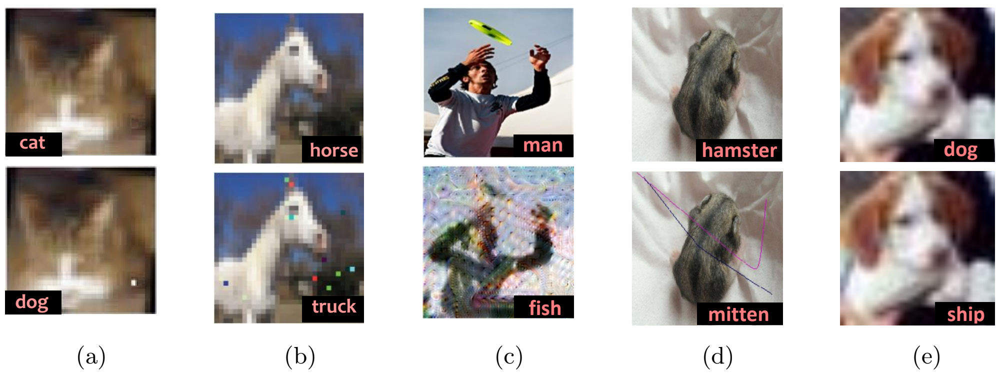


Figure 6.1: The images in the first row represents the original images and in the second row the adversarial images and their respective class labels that are created by (a) One-Pixel attack [55], (b) Few-Pixels attack [46], (c) Fooling Transfer Net (FTN) [63], (d) Scratch that! [32], and (e) our EA-based attack [13, 11] .

The purpose of this paper, that very substantially enhances most aspects of [12], is to address three issues: (1) Intrinsic performance of this EA-based attack, (2) Filter resistance of the adversarial images created by the EA, (3) Creation of natively filter resistant adversarial images. Before being more specific, let us point out that all experiments in this article are performed with the distance $d = L_2$ for the CNN $\mathcal{C}$ = VGG16 [24, 52] trained on the Cifar-10 [34] dataset to classify images according to 10 categories, and address mainly the *target scenario*, but also, to a lesser extent though, the *untargeted scenario* (see Section 6.2).

We address issue (2) by a thorough and extended efficiency study of our EA-based attack. In a first series of experiments with one ancestor per category of Cifar-10, we perform 10 independent runs per ancestor per target category, leading to altogether 900 attacks. The algorithm $\text{EA}_{L_2}^{\text{target,VGG-16}}$ obtains a success rate of 100% (all ancestor-target categories are achieved for at least one of the 10 runs performed on each ancestor), requiring between 290 and 2793 generations in average, depending on the (c_a, c_t) target scenario. To better assess the importance of the choice of the ancestor in a given category c_a, and the impact of the seed value used for a specific run, we extend these experiments. In a second series of experiments, we pick randomly 50 distinct ancestors for each of the 10 categories of Cifar-10, and run altogether 4500 attacks for the target scenario. In this case, our algorithm achieves a success rate of 98%, requiring between 461 and 1717 generations in average. Moreover, both series of experiments show that a run of $\text{EA}_{L_2}^{\text{target,VGG-16}}$ has more than 96% (actually 96, 56% for the former, and 98, 06% for the latter series) to terminate successfully, and to create images that fool humans and VGG16 trained on Cifar-10, despite our demanding requirements for a successful termination.

The issues (3) and (4) (addressed respectively in Sections 6.4 and 6.5) deserve to be put in the following broader perspective. Let $\mathcal{A}$ be an image classified by a CNN $\mathcal{C}$ in some category c_a, and $\mathcal{D}$ be an adversarial image, say for the *target scenario*, that $\mathcal{C}$ classifies in a distinct category c_t (at this stage, the type of attack that leads to $\mathcal{D}$ does not matter). One now considers a function $\mathcal{F}$, that acts on such images, to create images $\mathcal{F}(\mathcal{A})$ and $\mathcal{F}(\mathcal{D})$ of the size handled by the CNN (what coincides with the same common size of $\mathcal{A}$ and $\mathcal{D}$ in the present case). How does the CNN classify these new images? Does $\mathcal{F}(\mathcal{D})$ remain adversarial, or does the composition $\mathcal{C} \circ \mathcal{F}$ (that consists in putting $\mathcal{F}$ ahead of $\mathcal{C}$) protect $\mathcal{C}$ against the attack? If this latter case holds, can one adapt the attack to create images that fool not only $\mathcal{C}$, but also the $\mathcal{F}$-enhanced CNN $\mathcal{C} \circ \mathcal{F}$? If yes, would such images, adversarial for $\mathcal{C} \circ \mathcal{F}$, be adversarial as well for $\mathcal{C} \circ \mathcal{G}$ for $\mathcal{G} \neq \mathcal{F}$, hence have the capability to fool the same CNN $\mathcal{C}$ but enhanced by other functions $\mathcal{G}$?

Among the different meaningful functions $\mathcal{F}$ one could think of in this context, we undertake the study for filters. Indeed, daily used in image processing, filters substantially impact the visual appearance of images for a human eye on the one hand, and potentially affect the classification process of a trained CNN on the other hand. It is therefore tempting to check whether adding filters may prevent CNNs from misclassification, or may reduce this risk to some extent, when facing an adversarial image. Additionally, one may also want to evaluate the quality of adversarial images by their capacity to mimic the ancestor's image behavior when exposed to filters.

For reasons given in Section 6.4, in which is discussed the issue (3), we proceed to the selection of 5 filters, namely the Inverse filter (F_1), the Gaussian blur filter (F_2), the Median filter (F_3), the Unsharp mask filter (F_4), and the combination F_5 of the two last ones. We filter by each of them the ancestor $\mathcal{A}_a$ and the adversarial images $\mathcal{D}_{a,t}(\mathcal{A}_a)$ created by the algorithm $\text{EA}_{L_2}^{\text{target,VGG-16}}$ in Section 6.3. VGG16 is then challenged with these filtered images. The values of a series of specifically designed indicators lead to two conclusions. On the one hand, the Inverse, and the Unsharp mask filters are significantly inefficient against our EA, since for instance 95% of the adversarial images filtered by F_4 remain adversarial for the *target scenario*, and 95% remain adversarial for the *untargeted scenario* (in a relaxed sense to be made precise in this Section). *A contrario*, the other filters, especially the combination F_5, render our EA-based attack less effective, for both the *target* and the *untargeted scenario.*

This leads us to address the final issue (4). For a filter F, we conceive a filter-enhanced F-fitness function (see Section 6.5), and the corresponding algorithm $\text{EA}_{L_2,F}^{\text{target,VGG-16}}$, obtained from $\text{EA}_{L_2}^{\text{target,VGG-16}}$ by updating the fitness function accordingly. For reasons given in Section 6.5, we select $F = F_5$, and allocate to $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ the task to create adversarial images that are moreover natively immune against the filter F_5. In other words, these adversarial images fool simultaneously $\mathcal{C}$ and $\mathcal{C} \circ F_5$ for $\mathcal{C}$ = VGG16 for the *target scenario* (still with the demanding target label value ≥ 0.95), while remaining so close to the ancestor that no human eye would notice any difference. We perform similar experiments as for the issue (2). A first series of 900 attacks (one ancestor per ancestor category, 10 independent runs for each $(c_a(\mathcal{A}_a), c_t)$ scenario) shows that $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ achieves a success rate of $96,66\%$ (3 combinations were not achieved), and that the probability that it terminates successfully for a given run is $95,77\%$, requiring in average between 798 and 2746 generations for the successful $(c_a(\mathcal{A}_a), c_t)$ considered. In a second series of 4500 attacks performed with 50 different ancestors per category, $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ showed a success rate of 88%, with between 1250 and 2404 generations in average.

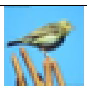
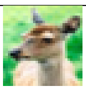
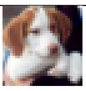
We complete the study (4) by exploring whether an adversarial image, constructed by $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ to fool both $\mathcal{C}$ and $\mathcal{C} \circ F_5$, would also be adversarial against $\mathcal{C} \circ F_k$ for the other filters F_1, F_2, F_3, F_4 for $\mathcal{C}$ = VGG16. Our study shows that it is so for F_3 and F_4 with (depending on the *target* or *untargeted scenario*) between 83% and 89% of the images remaining adversarial against these filters. 56% of theses images are also adversarial for F_1 for the *untargeted scenario*, while this percentage drops to 23% for F_2. Therefore, the $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ attack, designed to be robust against $\mathcal{C}$ and $\mathcal{C} \circ F_5$ for $\mathcal{C}$ =VGG16, is also robust to some significant extent against all individual filters for the *untargeted scenario.*

Section 6.6 summarizes the conclusions of this case study, and provides a series of research directions.

6.2 $\text{EA}^{\text{target},\mathcal{C}}$ parameters

Although applicable to any CNN trained at image classification on some dataset, we instantiate our approach on the concrete case of VGG16 [52] trained on Cifar-10 [34]. Table 6.1 presents the chosen ancestors, their respective categories and their reference numbers in the test set of Cifar-10.

Table 6.1: For $1 \leq a \leq 10$, the image $\mathcal{A}_a$ (and its reference number n^o in the test set of Cifar-10) classified by VGG16 in the category c_a, with its corresponding c_a-label values. These images are used as ancestor in most of our experiments.

a	1	2	3	4	5	6	7	8	9	10
c_a	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
n^o	281	82	67	91	455	16	29	17	1	76
$\mathcal{A}_a$	0.6900	0.9999	 0.9999	0.9998	 0.9999	 0.9996	0.9999	0.9998	0.9996	0.9984

For all tests run here, we used $\tau = 0.95$ (termination with success), $G = 7000$ (termination), $\delta = 3$, a population size of 160, $B(g_p, ind) = 10^{-5}$, $A(g_p, ind) = 10^{-\log_{10} o_{ind}[c_t]}$ and $d = L_2$.

6.3 The adversarial images obtained by $\mathrm{EA}_{L_2}^{\text{target,VGG-16}}$

For an ancestor $\mathcal{A}_a$ in a category c_a, and the target scenario for the category c_t, one defines $\mathcal{D}_{a,t}(\mathcal{A}_a) = \mathrm{EA}_{L_2}^{\text{target,VGG-16}}(\mathcal{A}_a, c_t)$, provided the algorithm terminates successfully. One writes more simply $\mathcal{D}_{a,t}$, or even $\mathcal{D}_t$, if there is no ambiguity about the choice of the ancestor $\mathcal{A}_a$ chosen in category c_a (*mutatis mutandis* in Sections 6.4 and 6.5).

6.3.1 With one ancestor per category

We pick from Table 6.1 the ancestor image $\mathcal{A}_a$ in the category c_a, and perform 10 independent runs (with random seed values) of $\mathrm{EA}_{L_2}^{\text{target,VGG-16}}$ for all 9 possible target categories $c_t \neq c_a$.

An example of the quality of the obtained adversarial images is highlighted by the comparison between the dog ancestor $\mathcal{A}_6$ of Table 6.1, and its corresponding 9 evolved adversarial images $\mathcal{D}_t$, with $t \neq 6$ (obtained after the first of the 10 independent runs of the EA) pictured in Figure 6.2. More generally, Figure 9.13 (Appendix 9.4.1) contains the adversarial images obtained by the first successful run out of the ten independent runs of $\mathrm{EA}_{L_2}^{\text{target,VGG-16}}$ for each of the ancestor images of Table 6.1, and Table 9.12 (Appendix 9.4.1) give their respective label values.

This example already illustrates that, by slightly changing many pixels instead of heavily changing a few pixels, our approach enhances the indistinguishability between the adversarial image and the ancestor image. In particular, our method differs substantially from [32, 46, 55], where a small fraction of pixels is changed, but at the cost of being noticeable for a human without difficulty (see Figure 6.1).

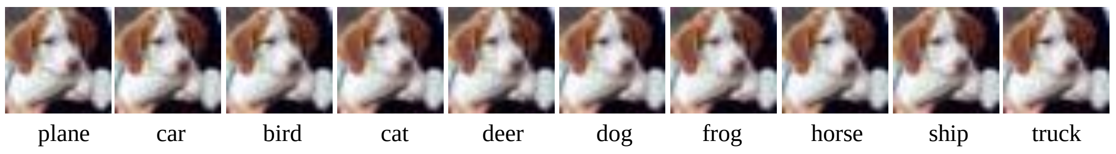


Figure 6.2: From the left, comparison of the ancestor $\mathcal{A}_6$ in the 6th position with the adversarial images $\mathcal{D}_t$ in the t^{th} position ($t \neq 6$). VGG16 classifies $\mathcal{A}_6$ in the *dog* category with probability 0.9996386, and classifies $\mathcal{D}_t$ in the target category c_t with probability ≥ 0.95.

For the ancestor image $\mathcal{A}_a$ (from Table 6.1) in the category c_a specified in its a^{th} row, the t^{th} column of Figure 6.3 gives the average number of generations required by $\mathrm{EA}_{L_2}^{\text{target,VGG-16}}$ to terminate, computed over 10 independent runs. In 4 ancestor/target combinations, this number is followed by a symbol $(\star x)$ or $(\star x, \ddagger y)$. These symbols indicate that the algorithm did not achieve the $\tau = 0.95$ threshold value within 7000 generations for x of the 10 runs, and therefore terminated without success for the corresponding seed values. The c_t-label values of the corresponding best descendant images remained stuck at a local optimum < 0.95, whose *quality* is also indicated by the symbol. In the case of the symbol $(\star x)$, this local optimum was quite close to 0.95 (not less than 0.9370 actually; we call *quasi-adversarial* the corresponding images produced by $\mathrm{EA}_{L_2}^{\text{target,VGG-16}}$). In the case of the symbol $(\star x, \ddagger y)$, the complementary number y specifies the number of runs among the x unsuccessful runs for which the local optimum stayed very low (between circa 10^{-4} to 10^{-5}).

	plane	car	bird	cat	deer	dog	frog	horse	ship	truck	Row Average
plane ($\mathcal{A}_1$)		64	275	2188	702	613	337	798	147	1108	692
car ($\mathcal{A}_2$)	1095		451	1246	768	1545	543	676	422	725	830
bird ($\mathcal{A}_3$)	1080	665		1823	925	6921(*9)	559	2719	872	1092	1850
cat ($\mathcal{A}_4$)	494	341	250		263	217	113	411	526	555	352
deer ($\mathcal{A}_5$)	2700	6233(*5, ‡5)	343	460		239	834	712	6683(*8, ‡8)	6939(*9, ‡9)	2793
dog ($\mathcal{A}_6$)	879	882	460	129	938		397	280	971	545	609
frog ($\mathcal{A}_7$)	690	520	295	488	717	536		834	927	685	632
horse ($\mathcal{A}_8$)	454	221	300	204	223	303	371		309	228	290
ship ($\mathcal{A}_9$)	318	182	1291	432	2599	1502	823	2065		484	1077
truck ($\mathcal{A}_{10}$)	145	663	383	1411	437	864	919	271	292		598
Column Average	872	1085	449	931	841	1415	544	974	1238	1373	

Figure 6.3: $\text{EA}_{L_2}^{\text{target,VGG-16}}$'s performance on all possible ancestor/target combinations with one ancestor per category. The rows give the ancestor category c_a (and the specific ancestor $\mathcal{A}_a$ in c_a), the columns indicate the target class c_t, and the cell values indicate the average number of generations required by $\text{EA}_{L_2}^{\text{target,VGG-16}}$ to terminate, computed on 10 independent runs.

For each $1 \leq a \leq 10$, the ”Row Average” value, displayed in the rightmost column of the a^{th} row, indicates the average number of generations required to perform our attack on the ancestor $\mathcal{A}_a$ in the category c_a for all $c_t \neq c_a$ (*Mutatis mutandis* the ”Column Average” value displayed in the bottom row of the t^{th} column).

Our EA shows a success rate of 100 % since all possible target categories were achieved with at least one of the ten runs for the considered ancestors. Still, some attacks are easier than others. The ancestor image for which $\text{EA}_{L_2}^{\text{target,VGG-16}}$ needs the least amount of effort in general is the *horse* ancestor image $\mathcal{A}_8$, and *bird* (c_3) is the easiest target category whatever the ancestor category (with the considered ancestor images at least). At the other end of the scale are the *deer* ancestor image $\mathcal{A}_5$ and the *bird* ancestor image $\mathcal{A}_3$ for which $\text{EA}_{L_2}^{\text{target,VGG-16}}$ requires the largest amount of effort in general, while *dog* (c_6), *truck* (c_{10}) and *ship* (c_9) are the hardest target categories. These correspond precisely to the categories (and the ancestors) for which some runs of $\text{EA}_{L_2}^{\text{target,VGG-16}}$ terminated without having created an appropriate adversarial image within 7000 generations. Indeed, out of the altogether 900 attacks (10 runs for each of the 90 ancestor/target combinations) performed by $\text{EA}_{L_2}^{\text{target,VGG-16}}$, Figure 6.3 shows that only 31 did not succeed. It is worthwhile noting the homogeneity and the non-diversity of the quality of the rare unsuccessful cases. For such unsuccessful (c_a, c_t) combination, either the local optimum is close to the $\tau = 0.95$ value for all failed cases (this occurs for the 9 unsuccessful runs of the (*bird* ($\mathcal{A}_4$), *dog*) combination), or it is very far of this threshold value for all failed cases (this occurs for the 22 unsuccessful runs with the *deer* ($\mathcal{A}_5$) ancestor for the *car*, the *ship* and the *truck* targets).

Therefore, as a consequence of this study with one ancestor $\mathcal{A}_a$ per category c_a, our experiments show that the probability that $\text{EA}_{L_2}^{\text{target,VGG-16}}$ terminates successfully for a given run is $96,56\%$, and that its termination requires between 290 and 2793 generations in average.

6.3.2 With 50 distinct ancestors per category

To further evaluate our attack's efficiency beyond the case of one single ancestor $\mathcal{A}_a$ per category c_a as described in subsection 6.3.1, and to assess the importance of a specific ancestor chosen in a given category, we considered 50 distinct images taken randomly (from the Cifar-10 testing set) in each of the 10 categories c_a. Unlike the 10 independent runs per ancestor of subsection 6.3.1, we considered that running $\text{EA}_{L_2}^{\text{target,VGG-16}}$ with one single run per ancestor was enough to make our point. Therefore, we performed altogether $50 \times 10 \times 9 = 4500$ attacks with $\text{EA}_{L_2}^{\text{target,VGG-16}}$. Figure 6.4, that summarizes the outcome of this experiment, is to be interpreted in a similar way as Figure 6.3, with the difference that the averages are computed over the 50 ancestors per category c_a. Note also that the $(\star x)$ and $(\star x, \ddagger y)$ symbols added to some cell values for a given (c_a, c_t) scenario have a different interpretation in Figure 6.4 compared to Figure 6.3, since they apply globally to *different ancestors* here, as opposed to applying to different runs performed on the *same ancestor* in Figure 6.3.

	plane	car	bird	cat	deer	dog	frog	horse	ship	truck		Row Average
plane		1201 (*2, ‡2)	284	606	415	859 (*1)	752 (*1, ‡1)	858	304	1042 (*3, ‡3)		702
car	1807 (*5, ‡5)		1740 (*3, ‡3)	2618 (*8, ‡8)	1751 (*3, ‡3)	2154 (*4, ‡4)	1492 (*2, ‡2)	2425 (*8, ‡8)	988 (*1, ‡1)	478		1717
bird	416	1029 (*1, ‡1)		376	390	537	397	679	575	844 (*1, ‡1)		583
cat	653 (*1)	703	358		381	152	234	321	834	519		462
deer	762 (*2, ‡2)	1459 (*4, ‡3)	208	290		274	382	269	855 (*1, ‡1)	1139 (*4, ‡4)		626
dog	772 (*1, ‡1)	799	319	203	492		344	392	609	686		513
frog	527	646	306	302	321	463		588	532	466		461
horse	1343 (*2, ‡2)	1869 (*4, ‡3)	851	692	310	325	1085 (*1, ‡1)		1679 (*4, ‡4)	2252 (*9, ‡9)		1156
ship	454	708	890	1044 (*2, ‡2)	684 (*1, ‡1)	1246 (*1)	734	1319 (*2, ‡1)		639		858
truck	576	495 (*1, ‡1)	813	912 (*1, ‡1)	1059 (*1, ‡1)	1077 (*2, ‡1)	994	908	395			803
Column Average	812	990	641	783	645	787	713	862	752	896		

Figure 6.4: $\text{EA}_{L_2}^{\text{target,VGG-16}}$'s performance on all possible ancestor/target combinations with 50 distinct ancestors per category. The rows give the ancestor category c_a, the columns indicate the target class c_t. The cell values give the average number of generations required by $\text{EA}_{L_2}^{\text{target,VGG-16}}$ to terminate, and computed on one run performed on each of the 50 ancestors in the category c_a.

Performance differs again from one category to another. The ancestor categories for which $\text{EA}_{L_2}^{\text{target,VGG-16}}$ needs the least amount of effort in general are the *frog*, the *cat* and the *dog* categories. In addition, $\text{EA}_{L_2}^{\text{target,VGG-16}}$ achieves the target categories *bird* and *deer* fairly fast, whatever the ancestor categories. Conversely, the ancestor categories *car* and *horse* are those for which $\text{EA}_{L_2}^{\text{target,VGG-16}}$ requires the largest amount of effort in general, while the *car* and the *truck* are the hardest target categories.

In this context, the comparison of these results with those of Figure 6.3 shows the relevance for $\text{EA}_{L_2}^{\text{target,VGG-16}}$'s performance of the specific ancestor image chosen in a given category c_a. Indeed, while for instance the specific ancestor $\mathcal{A}_8$ in the *horse* category was optimal in a sense (achieving all possible target categories in 290 generations in average), this property did not extend to the *horse* category as a whole as just seen. *A contrario*, while for instance the combination $(deer, truck)$ with the ancestor $\mathcal{A}_5$ in the *deer* category was (with 6939 generations in average) the toughest to achieve among all trials of subsection 6.3.1, it proves reasonably easy to achieve in general (with 1139 generations in average) with the 50 ancestors chosen for our experiment.

Finally, out of the altogether 4500 trials performed by $\mathrm{EA}^{\text{target,VGG-16}}_{L_2}$, only 87 did not terminate successfully. Therefore, this experiment provides a heuristic evidence that one run of $\mathrm{EA}^{\text{target,VGG-16}}_{L_2}$ has a probability of $98,06\%$ to terminate successfully. To better assess the strength of the failed cases, we run again the 87 unsuccessful cases 10 times with different seed values: out of them 28 succeeded in less than 10 runs, while 59 did not. This result, together with the fact that our algorithm required between 461 and 1717 generations in average in this case, and compared to the outcome of the similar experiments performed in the previous Subsection 6.3.1 with other ancestors, sustains further the impact of the specific ancestor $\mathcal{A}_a$ taken in a given category c_a, and of the seed value used to run the EA. It also shows that the success rate of our attack, namely the capacity for $\mathrm{EA}^{\text{target,VGG-16}}_{L_2}$ to terminate successfully for at least one of ten runs out of a small number of trials, is $\geq 98,68\%$.

6.4 Robustness of $\mathrm{EA}^{\text{target,VGG-16}}_{L_2}$ against filters

For the reasons given in the introduction to this paper (Section 6.1), the study undertaken in this Section essentially amounts to checking whether adding filters may prevent VGG16 from misclassification, or may reduce this risk to some extent, when facing an adversarial image created by $\mathrm{EA}^{\text{target,VGG-16}}_{L_2}$.

6.4.1 Selection of filters

Although a large list of filters exists, we focus on the following four that have a significant impact on images [39, chapters 7 and 8].

The *inverse filter* F_1 replaces all colors by their complementary colors. This operation is performed pixel for pixel by subtracting the RGB value $(255, 255, 255)$ of white by the RGB value of that pixel.

The *Gaussian blur filter* F_2 uses a Gaussian distribution to calculate the Kernel, $G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$, where x is the distance from the origin on the x-axis, y is the distance from the origin on the y-axis and σ is the standard deviation of the Gaussian distribution. By design, the process gives more priority to the pixels in the center, and blurs around it with a lesser impact as one moves away from the center.

The *median filter* F_3 is used to reduce noice and artefacts in a picture. Though under some conditions it can reduce noise while preserving the edges, this does not really occur for small images like those considered here. In general, one selects a pixel, and one computes the median of all the surrounding pixels.

The *unsharp mask filter* F_4 enhances the sharpness and contrast of images. The unsharp masked image is obtained by blurring a copy of the image using a Gaussian blur, which is then weighted and subtracted from the original image.

Any such filter F, or any combination of filters $F_{i_1}, F_{i_2}, \cdots, F_{i_k}$ operating successively (in that order) on an image $\mathcal{I}$, creates a filtered image $F(\mathcal{I})$ or $F_{i_k} \circ \cdots \circ F_{i_2} \circ F_{i_1}(\mathcal{I})$.

We make use of these four filters F_1, F_2, F_3, F_4 either individually, or as the combination $F_5 = F_3 \circ F_4$. The reason for the choice of the latter $F_3 \circ F_4$ is that F_4 is used to amplify and highlight detail, while F_3 is used to remove noise from an image without removing detail. Therefore, a combination of these filters could remove the noise created by the EA while maintaining a high level of detail. Moreover, since the computations are performed on images of size 32×32, we shall take a filter-size $f = 1$ for F_1 and $f = 3$ for the others.

6.4.2 VGG16's classification of filtered images

For $1 \leq a \leq 10$, the 10 images composed of the ancestor $\mathcal{A}_a$ on the one hand, and its corresponding 9 adversarial images $\mathcal{D}_{a,t}(\mathcal{A}_a)$ obtained by $\text{EA}_{L_2}^{\text{target,VGG-16}}$ on the other hand, and pictured in Figure 9.13 (Appendix 9.4.1) are exposed to the 5 filters F_1, F_2, F_3, F_4 and $F_5 = F_3 \circ F_4$. Figure 6.5 shows the outcome for the dog ancestor image $\mathcal{A}_6$, and the adversarial images $\mathcal{D}_t$ $(t \neq 6)$.

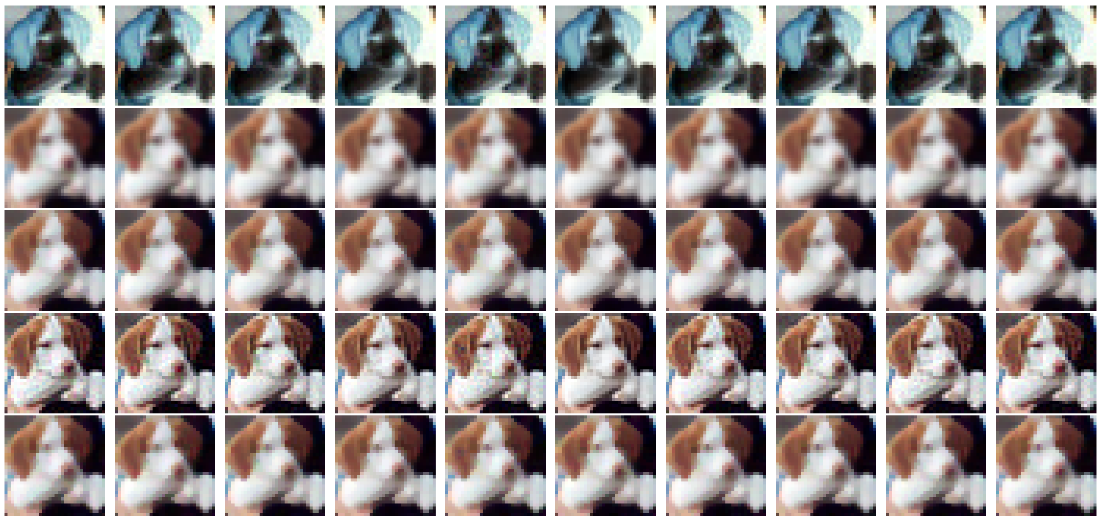


Figure 6.5: Comparison of the impact of filters on the ancestor $\mathcal{A}_6$ and on the adversarial images $\mathcal{D}_t$. The k^{th} row represents $F(\mathcal{D}_t)$ in t^{th} position (with $\mathcal{D}_6 = \mathcal{A}_6$), where $F = F_k$ for $1 \leq k \leq 5$.

For each $F = F_k$, $1 \leq k \leq 5$, we then challenge VGG16 with these altogether 100 filtered images $F(\mathcal{A}_a)$ and $F(\mathcal{D}_{a,t}(\mathcal{A}_a))$.

The complete classification and the corresponding label values outputted by VGG16 for $F(\mathcal{A}_a)$ and $F(\mathcal{D}_{a,t}(\mathcal{A}_a))$ for the 5 considered filters and for all (c_a, c_t) combinations are given in Tables 9.13 to 9.17 (Appendix 9.4.1). In these tables, an image is classified as belonging to a category c, if c has the largest label value outputted by VGG16 among all categories.

6.4.3 Indicators addressing the robustness of filtered adversarials

Filters differ substantially in their individual capacity to sustain the adversarial component of the filtered $F(\mathcal{D}_{a,t}(\mathcal{A}_a))$. Additionally, it may also happen that VGG16 classifies $F(\mathcal{A}_a)$ in a category different from the ancestor category c_a. Since we consider in this Section (and the next one) that the classification of an image in a given category c means that the label value given by VGG16 for c is the largest among all possible categories, we relax accordingly the formulation of the *target scenario*: in this context, one does not necessarily require a target label value exceeding the threshold value 0.95, but only asks that it is the largest one. The formulation of the *untargeted scenario* in the filtered context, made precise below in this Subsection, requires to pay attention to the potential difference between the categories c_a and $c_{F(\mathcal{A}_a)}$.

The following indicators assess the above stated issues quantitatively for each filter F_k, with the ancestors and adversarial images considered. These indicators take integer values, and we specify their theoretical bounds (which clearly depend on the number 10 of ancestors, and on the number 9 of target categories considered in this study).

For each $1 \leq a \leq 10$, one first defines $\rho_k(\mathcal{A}_a)$ as the number of target categories c_t such that VGG16 classifies $F_k(\mathcal{D}_{a,t}(\mathcal{A}_a))$ (including potentially $\mathcal{D}_{a,a}(\mathcal{A}_a) = \mathcal{A}_a$) back to the ancestor category c_a. One computes $\Sigma_k = \sum_{a=1}^{10} \rho_k(\mathcal{A}_a) \in [0, 100]$.

One sets $\delta_k(\mathcal{A}_a) = 1$ if $\rho_k(\mathcal{A}_a) = 10$, namely if the filtered ancestor and all filtered adversarial images are classified back to the ancestor category. Otherwise $\delta_k(\mathcal{A}_a) = 0$. One computes $\Delta_k = \sum_{a=1}^{10} \delta_k(\mathcal{A}_a) \in [0, 10]$.

One sets $\mu_k(\mathcal{A}_a) = 0$ if VGG16 classifies $F_k(\mathcal{A}_a)$ back to c_a, and $\mu_k(\mathcal{A}_a) = 1$ if it does not. One defines $\mathcal{M}_k = \sum_{a=1}^{10} \mu_k(\mathcal{A}_a) \in [0, 10]$.

Of interest for the *target scenario* is $\tau_k(\mathcal{A}_a)$, the number of $t \neq a$ for which $F_k(\mathcal{D}_{a,t}(\mathcal{A}_a))$ is classified as belonging to c_t (namely those that "really succeed"), and its sum $\mathcal{T}_k = \sum_{a=1}^{10} \tau_k(\mathcal{A}_a) \in [0, 90]$.

Finally, one considers $\widetilde{\tau}_k(a)$ to assess the *untargeted scenario*: $\widetilde{\tau}_k(a)$ counts the number of $t \neq a$ for which $F_k(\mathcal{D}_{a,t}(\mathcal{A}_a))$ is classified as belonging to $c \neq c_{F_k(\mathcal{A}_a)}$. One computes its sum $\widetilde{\mathcal{T}}_k = \sum_{a=1}^{10} \widetilde{\tau}_k(\mathcal{A}_a) \in [0, 90]$.

Observe *en passant* that the inequality $\mathcal{T}_k \leq \widetilde{\mathcal{T}}_k$ may theoretically not hold (as opposed to what happens in the absence of any filter, where the corresponding inequality necessarily holds). The reason is that one considers $c_t \neq c_a$ for the left-hand side of the inequality, and $c \neq c_{F_k(\mathcal{A}_a)}$ for the right-hand side. Since the quantities c_a and $c_{F_k(\mathcal{A}_a)}$ may differ, the set whose number of elements is $\mathcal{T}_k$ may not be included in the set whose number of elements is $\widetilde{\mathcal{T}}_k$.

6.4.4 Robustness analysis of the adversarial $\mathcal{D}_{a,t}(\mathcal{A}_a)$ against filters

Let us now proceed to the analysis of Table 6.2, that provides these quantities resulting from, and summarizing Tables 9.13 to 9.17 (Appendix 9.4.1).

Looking at Σ_k shows that, although all filters $F_1, \cdots, F_5$ bring some filtered images back to c_a, the Unsharp mask (F_4) and the Inverse (F_1) filters are the less efficient in this regard. *A contrario*, the three other filters bring back a majority of filtered images back to c_a. Noticeably the Median median (F_3) filter and foremost the combination (F_5) of the Unsharp and Median filters are highly effective since more than 80% of all filtered images are classified back to c_a. The three filters $F = F_2, F_3$ and F_5 are also those that bring all filtered images back to c_a for 5 (in the case of F_2), 6 (in the case of F_3) and 7 (in the case of F_5) ancestors, including *a fortiori* the filtered ancestor.

Consistently, the consideration of $\mathcal{T}_k$ and of $\widetilde{\mathcal{T}}_k$ shows that $\text{EA}_{L_2}^{\text{target,VGG-16}}$ resists highly effi-

Table 6.2: Indicator values assessing the robustness of adversarial images $\mathcal{D}_{a,t}(\mathcal{A}_a)$ against filters. For each ancestor $\mathcal{A}_a$, computation of $(\rho_k(\mathcal{A}_a), \delta_k(\mathcal{A}_a), \mu_k(\mathcal{A}_a))$ in the 1st row, and of $(\tau_k(\mathcal{A}_a), \widetilde{\tau}_k(\mathcal{A}_a))$ in the 2nd row. The last two rows give the sums $\sum_{a=1}^{10}$ of these quantities.

$\mathcal{A}_a$ \ k	1	2	3	4	5
$\mathcal{A}_1$	(10,1,0)	(0,0,1)	(2,0,0)	(0,0,1)	(7,0,0)
	(0,0)	(2,7)	(1,8)	(9,8)	(1,3)
$\mathcal{A}_2$	(1,0,0)	(3,0,0)	(9,0,0)	(3,0,0)	(10,1,0)
	(1,9)	(2,7)	(0,1)	(7,7)	(0,0)
$\mathcal{A}_3$	(7,0,0)	(10,1,0)	(10,1,0)	(1,0,0)	(10,1,0)
	(1,3)	(0,0)	(0,0)	(9,9)	(0,0)
$\mathcal{A}_4$	(3,0,0)	(10,1,0)	(10,1,0)	(1,0,0)	(10,1,0)
	(1,7)	(0,0)	(0,0)	(8,9)	(0,0)
$\mathcal{A}_5$	(1,0,1)	(10,1,0)	(10,1,0)	(1,0,0)	(10,1,0)
	(3,7)	(0,0)	(0,0)	(9,9)	(0,0)
$\mathcal{A}_6$	(0,0,1)	(0,0,1)	(1,0,1)	(1,0,0)	(4,0,0)
	(3,5)	(1,0)	(1,1)	(9,9)	(1,6)
$\mathcal{A}_7$	(5,0,0)	(10,1,0)	(10,1,0)	(2,0,0)	(10,1,0)
	(1,5)	(0,0)	(0,0)	(8,8)	(0,0)
$\mathcal{A}_8$	(0,0,1)	(10,1,0)	(10,1,0)	(1,0,0)	(10,1,0)
	(3,7)	(0,0)	(0,0)	(9,9)	(0,0)
$\mathcal{A}_9$	(6,0,0)	(1,0,1)	(8,0,0)	(1,0,0)	(8,0,0)
	(2,4)	(2,2)	(1,2)	(9,9)	(1,2)
$\mathcal{A}_{10}$	(0,0,1)	(0,0,1)	(10,1,0)	(1,0,0)	(10,1,0)
	(1,1)	(1,0)	(0,0)	(9,9)	(0,0)
$(\Sigma_k, \Delta_k, \mathcal{M}_k)$	(33, 1, 4)	(54, 5, 4)	(80, 6, 1)	(12, 0, 1)	(89, 7, 0)
$(\mathcal{T}_k, \widetilde{\mathcal{T}}_k)$	(16, 48)	(8, 16)	(3, 12)	(86, 86)	(3, 11)

ciently against the Unsharp mask filter F_4, since 95 % (86 out of 90) filtered images remain adversarial for the *target scenario* (with target label values no less than 0.5505, see Table 9.16), and altogether 95 % (86 out of 90) filtered images are adversarial for the *untargeted scenario.* Our EA remains also significantly efficient against the Inverse filter F_1, since 17 % (16/90) filtered images remain adversarial for the *target scenario* (with target label values ≥ 0.4415, see Table 9.13), and altogether 53 % (48/90) are adversarial for the *untargeted scenario.*

On the other hand, the Gaussian blur (F_2), the Median (F_3), and the Median and Unsharp combined (F_5) filters are effective to a far larger extent against $\text{EA}_{L_2}^{\text{target,VGG-16}}$, with F_3 and F_5 being particularly efficient at removing the adversarial property of the descendant images. Indeed, only 3 filtered adversarial images (hence 3 % of all filtered) remain adversarial for the target scenario for each of these two filters (with target label values ≥ 0.4978 for F_3, and ≥ 0.8131 for F_5, see Tables 9.15 and 9.17). For the *untargeted scenario* finally, the proportion of filtered images that are adversarial drops to 13 % (12/90) for F_3, and to 12 % (11/90) for F_5.

This study proves that the Inverse (F_1) and the Unsharp mask (F_4) filters are significantly to largely inefficient against our EA, but that the Gaussian (F_2), and foremost the Median (F_3) and the Combination ($F_5 = F_3 \circ F_4$) of the Unsharp mask and the Median filters render our

EA-based attack significantly less effective, for both the *targeted scenario* and for the *untargeted scenario*, at least with the ancestor images considered.

6.5 The filter-enhanced F-fitness function

Results of the previous section lead to the conception of a new fitness function, that *natively* forces the EA to create adversarial images that remain adversarial in a targeted sense once filtered. For a filter F, the filtered-enhanced F-fitness function is obtained as the following variant of the fitness function defined in Equation (3.6):

$$fit_{L_2}^{F}(ind, g_p) = A(g_p, ind)\left(\mathbf{o}_{ind}[c_t] + \mathbf{o}_{F(ind)}[c_t]\right) - B(g_p, ind)L_2(ind, \mathcal{A}), \tag{6.1}$$

where the component $\mathbf{o}_{F(ind)}[c_t]$ measures the probability that the individual filtered with F is classified as the target category. One obtains $\text{EA}_{L_2,F}^{\text{target,VGG-16}}$ from $\text{EA}_{L_2}^{\text{target,VGG-16}}$ by updating accordingly the fitness function. The termination and termination with success criteria are the same as in Section 6.2.

Since $F_5 = F_3 \circ F_4$ is not only highly efficient against $\text{EA}_{L_2}^{\text{target,VGG-16}}$, but is the filter that reverts the largest proportion (89 %) of images $\mathcal{D}_{a,t}(\mathcal{A}_a)$ back to c_a, we limit this study to this case.

6.5.1 Running $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ with one ancestor per category

For $1 \leq a \leq 10$, one performs 10 independent runs of $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ on the ancestor $\mathcal{A}_a$ in the category c_a given by Table 6.1. If $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ terminates successfully, one writes $\mathcal{D}_{a,t}^{F_5}(\mathcal{A}_a)$ for the first adversarial image obtained by $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ in less than 7000 generations. By construction, this image and its F_5 filtered version are classified by VGG16 as belonging to the target category c_t with probability ≥ 0.95, while remaining so close to $\mathcal{A}_a$ for a human eye that no one would notice any difference.

Figure 6.6 pictures the adversarial images $\mathcal{D}_{6,t}^{F_5}(\mathcal{A}_6)$ obtained that way for the dog ancestor $\mathcal{A}_6$ (all first runs succeeded for the dog ancestor).

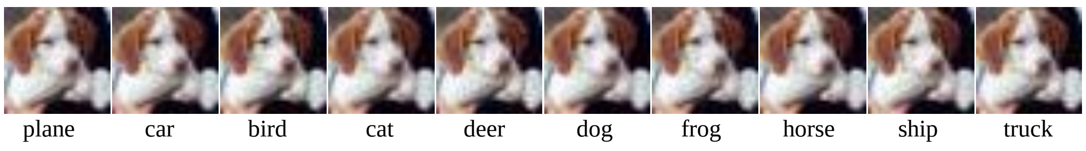
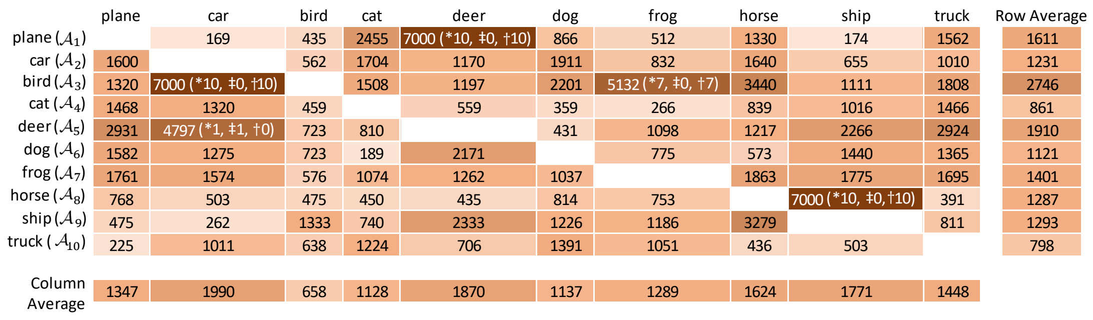


Figure 6.6: From left to right, comparison of the ancestor $\mathcal{A}_6$ in the 6th position with the adversarial images $\mathcal{D}_{6,t}^{F_5}(\mathcal{A}_6)$ in the tth position ($t \neq 6$).

For the ancestor image $\mathcal{A}_a$ (taken from Table 6.1) in the category c_a specified in its ath row, the tth row of Figure 6.7 gives the average number of generations required by $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ to terminate, computed over 10 independent runs. With a terminology adapted from the one used in Figure 6.3, this number is followed by a symbol ($\star x, \ddagger y, \dagger z$) in 5 of the 90 cells. The occurrence of this symbol means that the algorithm did not terminate successfully for x out of the 10 runs (obviously, the average value = 7000 if x = 10). Not succeeding means that the c_t-label value of the most performing descendant images $\mathcal{D}$ or of the filtered image $F_5(\mathcal{D})$ is stuck at some local optimum < 0.95. The symbols $\ddagger y$ and $\dagger z$ measure the quality of these local optimum. $\ddagger y$ (respectively $\dagger z$) counts the number of runs among the x unsuccessful ones for which the local optimum for the descendant $\mathcal{D}$ (respectively $F_5(\mathcal{D})$) stayed very low (between 10^{-3} and 10^{-6}).



	plane	car	bird	cat	deer	dog	frog	horse	ship	truck	Row Average
plane ($\mathcal{A}_1$)		169	435	2455	7000 (*10, ‡0, †10)	866	512	1330	174	1562	1611
car ($\mathcal{A}_2$)	1600		562	1704	1170	1911	832	1640	655	1010	1231
bird ($\mathcal{A}_3$)	1320	7000 (*10, ‡0, †10)		1508	1197	2201	5132 (*7, ‡0, †7)	3440	1111	1808	2746
cat ($\mathcal{A}_4$)	1468	1320	459		559	359	266	839	1016	1466	861
deer ($\mathcal{A}_5$)	2931	4797 (*1, ‡1, †0)	723	810		431	1098	1217	2266	2924	1910
dog ($\mathcal{A}_6$)	1582	1275	723	189	2171		775	573	1440	1365	1121
frog ($\mathcal{A}_7$)	1761	1574	576	1074	1262	1037		1863	1775	1695	1401
horse ($\mathcal{A}_8$)	768	503	475	450	435	814	753		7000 (*10, ‡0, †10)	391	1287
ship ($\mathcal{A}_9$)	475	262	1333	740	2333	1226	1186	3279		811	1293
truck ($\mathcal{A}_{10}$)	225	1011	638	1224	706	1391	1051	436	503		798
Column Average	1347	1990	658	1128	1870	1137	1289	1624	1771	1448	

Figure 6.7: $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$'s performance on all possible ancestor/target combinations. The rows give the ancestor categories c_a (and the specific ancestor $\mathcal{A}_a$ in c_a), the columns indicate the target class c_t, and the cell values give the average number of generations required by $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ to terminate, computed on 10 independent runs.

Out of the 900 performed runs, 38 did not terminate successfully, and 3 out of the 90 possible ancestor/target scenarios were not achieved, namely the pairs $(plane(\mathcal{A}_1), deer)$, $(bird(\mathcal{A}_3), car)$, $(horse(\mathcal{A}_8), ship)$. Therefore, the experiments show a success rate of $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ of 96, 66%, and a probability that the algorithm terminates successfully for a given run of 95, 77%.

Comparing Figure 6.7 to Figure 6.3 when all 10 runs terminate successfully for both $\text{EA}_{L_2}^{\text{target,VGG-16}}$ and $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ for a $(ancestor(\mathcal{A}_a), target)$ pair (83 cases altogether), the latter algorithm requires usually more generations than the former in average (with 3 notable exceptions, namely the $(ship(\mathcal{A}_9), deer)$, the $(ship(\mathcal{A}_9), dog)$ and the $(truck(\mathcal{A}_{10}), cat)$ pairs for which it needs 10%, 18% and 13% less generations). The fact that, for the 80 remaining pairs, $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ requires between 1.12 and 3.87 (depending on the pair considered) times more generations than $\text{EA}_{L_2}^{\text{target,VGG-16}}$ to terminate successfully is not surprising since there are 3 and no longer 2 criteria to fulfill.

For all 87 combinations $(ancestor(\mathcal{A}_a), target)$ for which $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ terminated successfully in at least one of the 10 independent runs, Figure 9.14 (Appendix 9.4.2) displays the first adversarial image $\mathcal{D}_{a,t}^{F_5}(\mathcal{A}_a)$ obtained by $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ (with $\mathcal{D}_{a,t}^{F_5}(\mathcal{A}_a) = \mathcal{A}_a$ repeated on the diagonal for the sake of consistency and comparison), and Table 9.18 (Appendix 9.4.2) gives the corresponding label values.

6.5.2 Running $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ with 50 ancestors per category

For the sake of completeness, we performed the same experiments as in Subsection 6.3.2 with the same 500 ancestor images (50 ancestor images per ancestor category), but this time with $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ instead of $\text{EA}_{L_2}^{\text{target,VGG-16}}$. Figure 6.8 shows the outcome. Out of 4500 attacks, 543 were unsuccessful, hence the success rate of $\text{EA}_{L_2,F_5}^{\text{target,VGG-16}}$ is 88%, and requires between

1250 and 2404 generations in average.

	plane	car	bird	cat	deer	dog	frog	horse	ship	truck	Row Average
plane		3423 (*17, ‡1, †16)	1038 (*4, ‡0, †4)	1914 (*8, ‡0, †8)	2348 (*12, ‡0, †12)	2438 (*15, ‡0, †15)	1920 (*6, ‡0, †6)	2816 (*15, ‡0, †15)	2190 (*8, ‡1, †8)	3546 (*22, ‡1, †21)	2404
car	1829 (*1, ‡0, †1)		1552 (*1, ‡1, †0)	2168 (*2, ‡2, †0)	1964 (*3, ‡1, †2)	1786 (*2, ‡0, †2)	1898 (*2, ‡2, †0)	1187	2814 (*7, ‡5, †2)	1066 (*2, ‡1, †1)	1807
bird	1505 (*2, ‡0, †2)	3280 (*11, ‡2, †8)		2112 (*10, ‡0, †10)	1821 (*8, ‡0, †8)	1491 (*5, ‡1, †4)	1514 (*7, ‡0, †7)	2144 (*7, ‡0, †7)	2246 (*10, ‡0, †10)	3254 (*15, ‡0, †15)	2152
cat	1406 (*1, ‡0, †1)	3132 (*15, ‡2, †12)	894 (*3, ‡0, †3)		2260 (*12, ‡0, †12)	855 (*4, ‡0, †4)	1842 (*10, ‡0, †10)	1745 (*7, ‡0, †7)	2820 (*13, ‡0, †13)	3876 (*21, ‡0, †21)	2092
deer	1367 (*3, ‡1, †3)	3680 (*18, ‡12, †8)	535 (*1, ‡0, †1)	870 (*2, ‡0, †2)		723	1362 (*5, ‡0, †5)	1189 (*4, ‡0, †4)	1485 (*2, ‡0, †2)	2524 (*8, ‡3, †7)	1526
dog	1842 (*5, ‡0, †5)	2911 (*12, ‡0, †12)	1027 (*4, ‡0, †4)	603 (*2, ‡0, †2)	1856 (*8, ‡0, †8)		2120 (*11, ‡0, †11)	1781 (*7, ‡0, †7)	2419 (*10, ‡0, †10)	3028 (*14, ‡0, †14)	1954
frog	1481	3712 (*16, ‡8, †8)	613	734	895 (*2, ‡0, †2)	904		1961 (*3, ‡1, †2)	1775 (*4, ‡0, †4)	2583 (*10, ‡1, †9)	1629
horse	1419 (*1, ‡0, †1)	2687 (*5, ‡0, †5)	779	1215 (*2, ‡0, †2)	956 (*3, ‡0, †3)	997 (*2, ‡0, †2)	1866 (*7, ‡0, †7)		1783 (*2, ‡0, †2)	2959 (*10, ‡1, †9)	1629
ship	1218 (*2, ‡0, †2)	2724 (*9, ‡0, †9)	1222 (*1, ‡0, †1)	1494 (*2, ‡0, †2)	2649 (*13, ‡0, †13)	1710 (*2, ‡1, †1)	2431 (*12, ‡0, †12)	2056 (*4, ‡2, †2)		2490 (*11, ‡0, †11)	1999
truck	1355 (*2, ‡0, †2)	1180 (*4, ‡0, †4)	978	1199 (*2, ‡1, †1)	1584 (*3, ‡0, †3)	1337 (*1, ‡1, †0)	1302 (*2, ‡0, †2)	1157 (*1, ‡0, †1)	1157 (*1, ‡0, †1)		1250
Column Average	1491	2970	960	1368	1815	1360	1806	1782	2077	2814	

Figure 6.8: 500 attacked images with 50 samples per ancestor class. Rows correspond to source classes, columns correspond to target classes, and cell values correspond to the average number of generations needed by $\mathrm{EA}^{\mathrm{target,VGG\text{-}16}}_{L_2,F_5}$ to terminate.

Comparing Figure 6.3 with Figure 6.7 and Figure 6.4 with Figure 6.8 shows that $\mathrm{EA}^{\mathrm{target,VGG\text{-}16}}_{L_2,F_5}$ usually requires more generations than $\mathrm{EA}^{\mathrm{target,VGG\text{-}16}}_{L_2}$ to construct adversarial images, what is to expect since $\mathrm{EA}^{\mathrm{target,VGG\text{-}16}}_{L_2,F_5}$ must satisfy not two, but three conditions.

6.5.3 Robustness of $\mathcal{D}^{F_5}_{a,t}(\mathcal{A}_a)$ against VGG16$\circ F_k$ for all filters

Using again the images of Figure 9.14 (Appendix 9.4.2) obtained as described in Subsection 6.5.1, the ancestor $\mathcal{A}_a$ and the corresponding adversarial images $\mathcal{D}^{F_5}_{a,t}(\mathcal{A}_a)$ are then tested against all five filters of subsection 6.4.1. Figure 6.9 shows the outcome of this process for the dog ancestor $\mathcal{A}_6$ and the adversarial images $\mathcal{D}^{F_5}_{6,t}(\mathcal{A}_6)$.

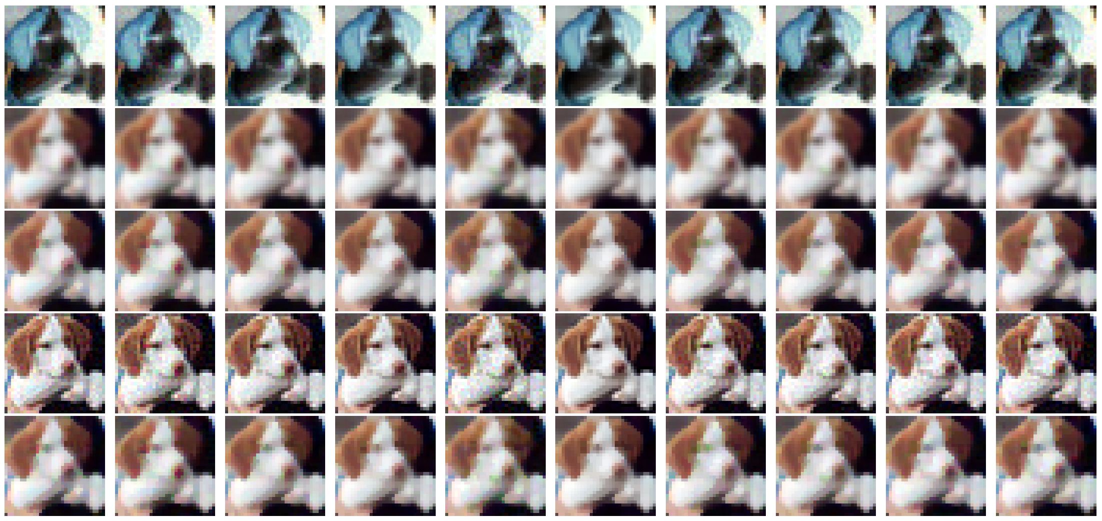


Figure 6.9: Impact of filters on the ancestor $\mathcal{A}_6$ and adversarial images $\mathcal{D}^{F_5}_{6,t}(\mathcal{A}_6)$. The k^{th} row represents $F(\mathcal{D}^{F_5}_{6,t}(\mathcal{A}_6))$ in t^{th} position (with $\mathcal{D}^{F_5}_{6,6}(\mathcal{A}_6) = \mathcal{A}_6$), where $F = F_k$ for $1 \leq k \leq 5$.

These filtered images are given to VGG16 for classification (see Appendix 9.4.2, Table 9.18 for F_5, and Table 9.19 for F_1, F_2, F_3 and F_4, with $\mathcal{D}^{F_5}_{a,a}(\mathcal{A}_a) = \mathcal{A}_a$ to ease the notations).

Mutatis mutandis, one obtains Table 6.3 in a similar way as Table 6.2. Note that the upper bounds of the indicators are impacted by the fact that 3 combinations $(c_a(\mathcal{A}_a), c_t)$ were not achieved. Indeed, one has $0 \leq \rho^{F_5}_k(\mathcal{A}_a) \leq 9$ for $a = 1, 3, 8$, and $0 \leq \rho^{F_5}_k(\mathcal{A}_a) \leq 10$ otherwise. One writes $\delta^{F_5}_k(\mathcal{A}_a) = 1$ if the filtered ancestor and all filtered adversarial images are classified back to the ancestor category whenever possible. Consistently, one has $0 \leq \tau^{F_5}_k(\mathcal{A}_a), \widetilde{\tau}^{F_5}_k(\mathcal{A}_a) \leq 8$ for $a = 1, 3, 8$, and $0 \leq \tau^{F_5}_k(\mathcal{A}_a), \widetilde{\tau}^{F_5}_k(\mathcal{A}_a) \leq 9$ otherwise. As a consequence, one has $0 \leq \Sigma^{F_5}_k \leq 97$, $0 \leq \Delta^{F_5}_k, \mathcal{M}^{F_5}_k \leq 10$, and $0 \leq \mathcal{T}^{F_5}_k, \widetilde{\mathcal{T}}^{F_5}_k \leq 87$.

Table 6.3 clearly shows that the produced images are not only adversarial for F_5, but also for F_3 and F_4 to a large extent for the *target scenario* (88% and 84% respectively), and for the *untargeted scenario* (89% and 88% respectively) as well. Additionally, 56% of these images are efficient against F_1 for the *untargeted scenario*, while this percentage drops to 23% with F_2.

This study shows that the $\text{EA}^{\text{target,VGG-16}}_{L_2,F_5}$ attack, designed to be robust against F_5, is also robust to some significant extent against all individual filters considered for the *untargeted* scenario, the Gaussian filter (F_2) being the most efficient at removing the adversarial character of the constructed images.

6.6 Summary of the outcomes

This work successfully addresses the four issues raised in the chapter's introduction. First, an extensive experimental study further showed the intrinsic efficiency of our algorithm $\text{EA}^{\text{target,VGG-16}}_{L_2}$ at constructing adversarial images for the *target scenario* performed against VGG16 with images from Cifar-10. We then challenged the adversarial images obtained against a series of filters, and

Table 6.3: Indicator values assessing the robustness of adversarial images $\mathcal{D}_{a,t}^{F_5}(\mathcal{A}_a)$ against filters. For each ancestor $\mathcal{A}_a$, computation of $(\rho_k^{F_5}(\mathcal{A}_a), \delta_k^{F_5}(\mathcal{A}_a), \mu_k^{F_5}(\mathcal{A}_a))$ in the 1st row, and of $(\tau_k^{F_5}(\mathcal{A}_a), \widetilde{\tau}_k^{F_5}(\mathcal{A}_a))$ in the 2nd row. The last two rows give the sums $\sum_a$ of these quantities for all possible a.

k / $\mathcal{A}_a$	1	2	3	4	5
$\mathcal{A}_1$	(9,1,0) (0,0)	(0,0,1) (2,6)	(1,0,0) (8,8)	(0,0,1) (8,7)	(1,0,0) (8,8)
$\mathcal{A}_2$	(1,0,0) (1,9)	(2,0,0) (4,8)	(1,0,0) (9,9)	(4,0,0) (6,6)	(1,0,0) (9,9)
$\mathcal{A}_3$	(6,0,0) (2,3)	(9,1,0) (0,0)	(6,0,0) (3,3)	(1,0,0) (8,8)	(1,0,0) (8,8)
$\mathcal{A}_4$	(5,0,0) (2,5)	(9,0,0) (1,1)	(1,0,0) (8,9)	(1,0,0) (5,9)	(1,0,0) (9,9)
$\mathcal{A}_5$	(0,0,1) (3,8)	(10,1,0) (0,0)	(1,0,0) (8,9)	(1,0,0) (9,9)	(1,0,0) (9,9)
$\mathcal{A}_6$	(0,0,1) (3,6)	(0,0,1) (1,0)	(0,0,1) (7,6)	(1,0,0) (9,9)	(1,0,0) (9,9)
$\mathcal{A}_7$	(6,0,0) (2,4)	(8,0,0) (2,2)	(1,0,0) (9,9)	(5,0,0) (5,5)	(1,0,0) (9,9)
$\mathcal{A}_8$	(0,0,1) (1,8)	(6,0,0) (2,3)	(1,0,0) (8,8)	(1,0,0) (7,8)	(1,0,0) (8,8)
$\mathcal{A}_9$	(6,0,0) (1,4)	(1,0,1) (2,2)	(2,0,0) (8,8)	(3,0,0) (7,7)	(1,0,0) (9,9)
$\mathcal{A}_{10}$	(0,0,1) (1,2)	(0,0,1) (2,1)	(1,0,0) (9,9)	(1,0,0) (9,9)	(1,0,0) (9,9)
$(\Sigma_k^{F_5}, \Delta_k^{F_5}, \mathcal{M}_k^{F_5})$	(33,1,4)	(45,2,4)	(15,0,1)	(18,0,1)	(10,0,0)
$(\mathcal{T}_k^{F_5}, \widetilde{\mathcal{T}}_k^{F_5})$	(16,49)	(16,23)	(77,78)	(73,77)	(87,87)

finally designed a variant $\text{EA}_{L_2,F}^{\text{target,VGG-16}}$ of the EA, designed specifically to fool VGG16 and VGG16 composed with a filter F, and demonstrated the efficiency of the produced adversarial images not only against the specific filter chosen, but also against other filters as well.

Chapter 7

Comparative Analysis of the EA and BIM Adversarial Attacks

The work presented in this chapter is extracted from [14].

7.1 Introduction

This chapter focuses on understanding the underlying manner in which the EA-based attack deceives the CNNs. A thorough analysis is performed through various perspectives and experiments with the adversarial images and their noise. Additionally, the entire study is simultaneously performed on another successful, but opposing attack, which allows for their comparison.
This study aims at gaining an insight into the functioning of adversarial attacks by analyzing the adversarial images on the one hand, and the reactions of CNNs when exposed to adversarial images on the other hand. These analyses and comparisons are performed from different perspectives: behaviour while looking at smaller regions, noise frequency, transferability and changes in image texture, penultimate layers. The reasons for considering these perspectives are as follows. The first question we attempt to answer is whether adversarial attacks exploit the CNNs' bias towards texture [25]. This issue is related to the frequency of the noise, in the sense that changes of image texture are reflected by the input of high frequency noise [53]. This issue is also related to what happens at smaller image regions, since texture modifications should also be noticed at these levels. The transferability issue measures how far the adversarial noise is specific to the attacked CNN, or to the training data. Finally, studying the behaviour of the penultimate layers of the addressed CNNs provides a close look at the direction of the adversarial noise with respect to each object category.

This insight is addressed *via* a thorough experimental study. We selected 10 CNNs that are very diverse in terms of architecture, number of layers, etc. These CNNs are trained on the ImageNet dataset to sort images of size 224×224 into 1000 categories. We then intentionally chose two attacks that are on opposing edges of the attacks' classification. More precisely, here we consider the gradient-based BIM [36] and the score-based $\text{EA}^{\text{target},\mathcal{C}}$ [57, 5, 11, 13], both having high success rates against CNNs trained on ImageNet [31, 57].

We run these two algorithms to fool the 10 CNNs, with the additional very demanding requirement that, in order for an image to be considered adversarial, its c_t-label value should exceed

0.999. Starting with 10 random pairs of ancestor and target categories (c_a, c_t), and 10 random ancestor images in each c_a, hence 100 ancestor images altogether, out of the 1000 performed runs per attack, the two attacks succeeded for 84 common ancestors, leading to 2 distinct groups (one for each attack) of 437 adversarial images coming from these 84 convenient ancestors. The 2×437 adversarial images and the 10 CNNs are then analyzed and compared from the above-mentioned perspectives. Each of these perspectives is addressed in a dedicated subsection, that contains the specific obtained outcomes.

The study is organized as follows.

Section 7.2 explains the criteria leading to the selection of the 10 CNNs, of the ancestor and target categories, as well as the choice of the ancestor images in each category. We recall the design of our algorithm $\text{EA}^{\text{target},\mathcal{C}}$ and of BIM, and explain how we obtained the 0.999-strong adversarial images used in our experiments.

In Section 7.3, we analyze whether the adversarial noise introduced by the EA and by BIM has an adversarial impact at regions of smaller size. We also explore whether this local noise alone is sufficient to mislead the CNN, either individually or globally but in a shuffled way.

In Section 7.4, we provide a visualization of the noise that the EA and BIM add to an ancestor image to produce an adversarial image. In particular, we identify the frequencies of the noise introduced by the EA and BIM, and, among them, those that are key to the adversarial nature of the images created by each of the two attacks.

Section 7.5 explores the potential transferability of the adversarial images from one CNN to another. The issue is to clarify whether adversarial images are specific to their targeted CNN, or whether they contain rather general features that are perceivable by others. Since ImageNet-trained CNNs are biased towards texture [25], we examine whether texture is changed by the EA and by BIM, and whether CNNs with differing amounts of texture bias agree on which image modifications have the largest adversarial impact.

The transferability issue is pursued in Section 7.6. We explore whether the adversarial noise at regions of smaller sizes is less CNN-specific, hence more transferable, than at full scale. This issue is addressed in two ways. First, we check whether and how a modification of the adversarial noise intensity affects the c_a and the c_t-label values predicted by a CNN when fed with a different CNN's adversarial image, and the influence of shuffling in this process. Secondly, we keep the adversarial noise as it is (meaning without changing its intensity), and we check whether adversarial images are more likely to transfer when they are shuffled.

Finally, we delve inside the CNNs in Section 7.7. We study the changes that adversarial images produce in the activation of the CNNs penultimate layers.

The concluding Section 7.8 wraps up our results. This study is completed by Appendix 9.5, which displays the ancestors, the convenient ancestors, and some 0.999-strong adversarial images obtained by the EA and by BIM. The Appendix also contains a series of tables and graphs supporting our findings.

7.2 Adversarial images created by BIM and by EA$^{\text{target},\mathcal{C}}$

This section first lists both the 10 CNNs and the (ancestor, target) category pairs on which the targeted attacks are performed (Subsection 7.2.1). Since this paper's focus is on performing experiments with the adversarial images, rather than evaluating the attacks' functioning or performances, we only give a brief overview of the two algorithms used here, namely EA$^{\text{target},\mathcal{C}}$ and BIM (Subsection 7.2.2). Lastly, we specify the parameters used by EA$^{\text{target},\mathcal{C}}$ and BIM to construct the adversarial images used in the remainder of this paper (Subsection 7.2.3).

7.2.1 Selected CNNs, ancestor and target categories

We challenge a significant series of well-known CNNs, that cover a large part of the existing deep learning approaches to object recognition. For practical reasons and for comparison purposes, we require the availability of their pre-trained versions in the PyTorch [48] library, and that they handle images of similar size. These criteria led us to select the following 10 CNNs, trained on ImageNet, and handling images of size 224 × 224: $\mathcal{C}_1$ = DenseNet121 [29], $\mathcal{C}_2$ = DenseNet169 [29], $\mathcal{C}_3$ = DenseNet201 [29], $\mathcal{C}_4$ = MobileNet [28], $\mathcal{C}_5$ = MNASNet [56], $\mathcal{C}_6$ = ResNet50 [27], $\mathcal{C}_7$ = ResNet101 [27], $\mathcal{C}_8$ = ResNet152 [27], $\mathcal{C}_9$ = VGG16 [52], $\mathcal{C}_{10}$ = VGG19 [52] (Two additional CNNs BagNet17 [8] and ResNet50-SIN [25] are considered in Section 7.5 for reasons explained thereof).

Among the 1000 categories of ImageNet, we randomly pick ten ancestor $a_1, \cdots, a_{10}$ and ten target categories $t_1, \cdots, t_{10}$. These are given in Table 7.1. For each (ancestor, target) pair (c_{a_q}, c_{t_q}) (with $1 \leq q \leq 10$), we randomly select 10 ancestor images $\mathcal{A}_q^p$ (with $1 \leq p \leq 10$), resized to 224×224 using bilinear interpolation if necessary. These 100 ancestor images, pictured in Figure 9.15 in Appendix 9.5.1, are labeled by the 10 CNNs as a_q in 97% cases, with negligible c_{t_q}-label values (approximately between $9e-11$ and $2e-3$). Two different algorithms are used to perform targeted attacks on all 10 CNNs, all 10 (c_{a_q}, c_{t_q}) $(1 \leq q \leq 10)$ pairs and all 10 $\mathcal{A}_q^p$ $(1 \leq p \leq 10)$.

Table 7.1: For $1 \leq q \leq 10$, the 2nd row gives the ancestor category c_{a_q} and its index number a_q among the categories of ImageNet (Mutatis mutandis for the target categories, 3rd row).

q	1	2	3	4	5	6	7	8	9	10
c_{a_q} a_q	abacus 398	acorn 988	baseball 429	broom 462	brown bear 294	canoe 472	hippopotamus 344	llama 355	maraca 641	mountain bike 671
c_{t_q} t_q	bannister 421	rhinoceros beetle 306	ladle 618	dingo 273	pirate 724	Saluki 176	trifle 927	agama 42	conch 112	strainer 828

7.2.2 Design of BIM

Given a trained CNN $\mathcal{C}$, this section summarizes the key features of BIM [36]. The algorithm's purpose is to evolve an ancestor image $\mathcal{A}$ into a τ-strong adversarial image (for some convenient value of τ) that deceives $\mathcal{C}$ at image classification.

As opposed to the EA, BIM is a white-box attack, since it requires the knowledge of the CNN's parameters and architecture. The algorithm does not stop when a particular c_t-label value has been reached, but rather once a given number N of steps has been performed. More concretely, BIM can be seen as an iterative extension of the FGSM [26] attack. It creates a sequence of images (X_ℓ^{adv}), where the initial value is set to the ancestor $\mathcal{A}$, namely $X_0^{adv} = \mathcal{A}$, and the next images are defined step-wise by the induction formula:

$$X^{adv}_{\ell+1} = Clip_{\epsilon}\{X^{adv}_{\ell} - \alpha sign(\Delta_{\mathcal{A}}(J_{\mathcal{C}}(X^{adv}_{\ell}, c_t)))\}, \tag{7.1}$$

where $J_{\mathcal{C}}$ is the CNN's loss function, $\Delta_{\mathcal{A}}$ is the gradient acting on that loss function, α is a constant that determines the perturbation magnitude at each step, and $Clip_{\epsilon}$ is the function that maintains the obtained image within $[\mathcal{A} - \epsilon, \mathcal{A} + \epsilon]$, where ϵ is a constant that defines the overall perturbation magnitude. Once the number N of steps is specified, BIM's output is the image X^{adv}_{N}. This image is then given to $\mathcal{C}$ in order to get its c_t-label value, and its classification.

A major difference between BIM and the EA is that with BIM, the c_t-label values are measured *a posteriori*, while with the EA the τ-threshold is fixed *a priori*.

7.2.3 Creation of 0.999-strong adversarial images by EA$^{\textbf{target},\mathcal{C}}$ and by BIM

For both algorithms, we set $\delta = 2/255$ and $\epsilon = 8/255$. Specifically for the EA-based attack, we set $A = 1$, $B = 0$, and a population size of 40. For $\mathcal{C} = \mathcal{C}_k$, we write $atk = \text{EA}^{\text{target},\mathcal{C}}$ or BIM, and use $\mathcal{D}^{atk}_{k}(\mathcal{A}^{p}_{q})$ to denote a 0.999-strong adversarial image obtained by the corresponding algorithm for the target scenario performed on the (ancestor, target) category pair (c_{a_q}, c_{t_q}) against $\mathcal{C}_k$ with ancestor image $\mathcal{A}^{p}_{q}$. The τ threshold value was set to 0.999 mainly due to BIM's behaviour, as explained below.

With a number N of steps equal to 5, all BIM runs led to images satisfying equation (2.3). Out of the 1000 images obtained that way, 549 turned out to be 0.999-strong adversarial. It is precisely because so many BIM adversarials had such a high c_t-label value that we set $\tau = 0.999$ for EA$^{\text{target},\mathcal{C}_k}$ as well, in order to obtain adversarial images that are comparable to those created by BIM. We also fixed the second stopping condition for EA$^{\text{target},\mathcal{C}_k}$, namely the maximal number of generations, to $G = 103,000$. This very large value was necessary in order to allow the EA to create τ-strong adversarial images for a τ as high as 0.999. The EA successfully created 0.999-strong adversarial images in 716 cases. Note that our point is not to compare the performance of the algorithms, but to study the adversarial images they obtain.

In order to reduce any potential bias when comparing the adversarial images, we only considered the combinations of ancestor images $\mathcal{A}^{p}_{q}$ and CNNs for which both the EA and BIM successfully created 0.999-strong adversarial images for the corresponding (c_{a_q}, c_{t_q}) pairs. This notion defines "convenient ancestors" and "convenient combinations".

In Appendix 9.5.1, Figure 9.16 lists the 84 convenient ancestors. Table 9.20 shows that there are 437 convenient combinations (Note that all 10 CNNs belong to at least one such combination). Figures 9.17 and 9.18 provide examples of the obtained adversarial images for some convenient ancestors.

All experiments of the subsequent sections are therefore performed on the 84 convenient ancestors and on the 2×437 corresponding adversarial images.

7.3 Local effect of the adversarial noise on the target CNN

Here we analyze whether the adversarial noise introduced by the EA and by BIM also has an adversarial effect at regions of smaller size, and whether this local effect alone would be sufficient to mislead the CNNs, either individually (subsection 7.3.1) or globally but in a ”patchwork” way (subsection 7.3.2).

7.3.1 Is each individual patch adversarial?

To examine the adversarial effect of local image areas, we replace non-overlapping 16×16, 32×32, 56×56, and 112×112 patches of the ancestors with patches taken from the same location in their adversarial versions (this process is performed for BIM and for the EA separately), one patch at a time, starting from the top-left corner. Said otherwise, each step leads to a new hybrid image I, that coincides with the ancestor image $\mathcal{A}$ everywhere except for one patch, taken at the same emplacement from the adversarial $\mathcal{D}_k^{atk}(\mathcal{A})$. At each step the hybrid image I is sent to $\mathcal{C}_k$, to extract the c_a and c_t-label values, $o_I^{\mathcal{C}_k}[a]$ and $o_I^{\mathcal{C}_k}[t]$. Figure 7.1 shows an example of the plots of these successive c_a and c_t-label values, step-by-step, for the ancestor image $\mathcal{A}_5^4$, the CNN $\mathcal{C}_6$, and the adversarial images obtained by the EA and by BIM. The behaviour illustrated in this example is representative of what happens for all ancestors and CNNs (see Figure 9.19 in Appendix 9.5.2).

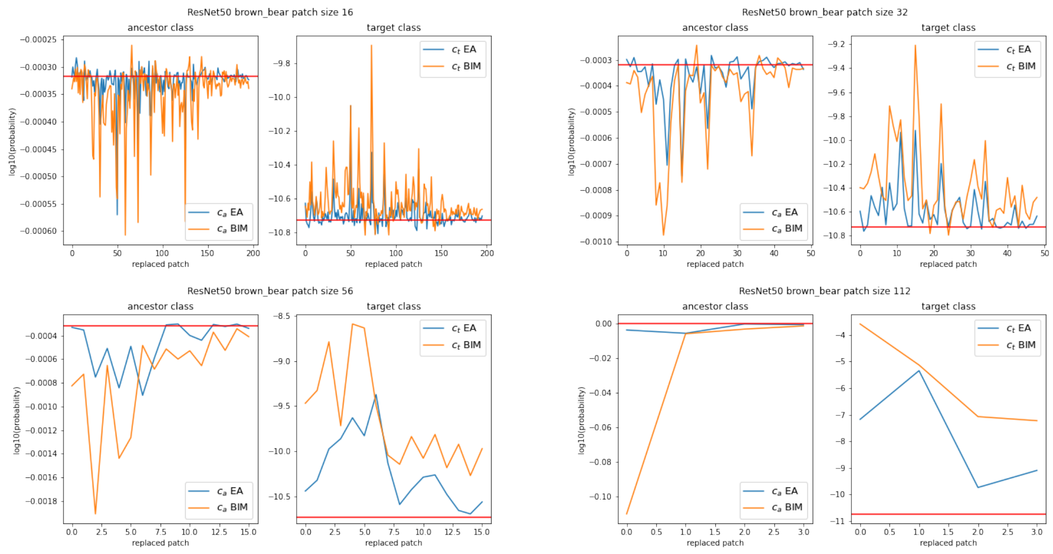


Figure 7.1: Single patch replacement for $\mathcal{A}_5^4$ and $\mathcal{C} = \mathcal{C}_6$. The 4 pairs of graphs correspond to patches of size 16×16, 32×32, 56×56 and 112×112, respectively. Each pair represents the step-wise plot of $log(o_I^{\mathcal{C}}[a])$ (left graph) and of $log(o_I^{\mathcal{C}}[t])$ (right graph) for the EA (blue curve) and BIM (orange curve). The red horizontal line recalls the c_a-label value (left graph) or the c_t-label value (right graph) of $\mathcal{A}_5^4$ with no replaced patch.

For all values of s and with both attacks, almost all patches individually increase the c_t-label value and decrease the c_a-label value. The fact that the peaks often coincide between the EA and BIM proves that modifying the ancestor in some image areas, rather than others, can make a large difference. However, BIM’s effect is usually larger than the EA’s. Also note that no single patch is sufficient to fool the CNNs, in the sense that it would create a hybrid image with a dominating c_t-label value.

7.3.2 Is the global random aggregation of local adversarial effect sufficient to fool the CNNs?

Firstly, replacing all patches simultaneously and at the correct location is by definition enough for a targeted misclassification, since its completion leads to the adversarial image. Secondly, most of the patches taken individually have a local adversarial impact, but none is enough individually to achieve a targeted attack.

The issue addressed here is whether the global aggregation of the local adversarial effect is strong enough, independently on the location of the patches, to create the global adversarial effect we are aiming at.

We proceed as follows. Given an Image I, and an integer s so that patches of size $s \times s$ create a partition of I, $sh(I, s)$ is a shuffled image deduced from I by randomly swapping all its patches. With these notations, $sh(\mathcal{D}_k^{atk}(\mathcal{A}_q^p), s)$ (with $atk = BIM$ or EA) is sent to the $\mathcal{C}_k$ CNN. One obtains the c_a and the c_t-label values, as well as the dominant category (that may differ from c_a, c_t). The values of s used in our tests are $16, 32, 56, 112$, leading to partitions of the 224×224 images into 196, 49, 16 and 4 patches respectively.

Table 7.2 gives the outcome of these tests. For each value s, each cell is composed of a triplet of numbers. The left one corresponds to the tests with the ancestor images, the middle one to the tests with images obtained by the EA, and the right one to the tests with images obtained by BIM. Each number is the percentage of images $sh(\mathcal{A}_q^p, s)$ or of images $sh(\mathcal{D}_k^{atk}(\mathcal{A}_q^p), s)$, taken for all ancestor images $\mathcal{A}_q^p$, all (ancestor, target) category pairs, and all $\mathcal{C}_1, \cdots, \mathcal{C}_{10}$, that are classified in category c, where c is the ancestor category c_a, the target category c_t, or any other class. To allow comparisons, the randomly-selected swapping order of the patches is only performed once per value of s. For each s, this uniquely defined sequence is applied in the same way to create the $sh(\mathcal{A}_q^p, s)$, $sh(\mathcal{D}_k^{EA}(\mathcal{A}_q^p), s)$, and $sh(\mathcal{D}_k^{BIM}(\mathcal{A}_q^p), s)$ shuffled images.

s	Number of patches	$c = c_a$	$c \notin \{c_a, c_t\}$	$c = c_t$
16	196	0.4, 0.1, 0.1	99.6, 99.9, 99.9	0.0, 0.0, 0.0
32	49	18.0, 9.2, 5.3	82.0, 90.8, 94.4	0.0, 0.0, 0.3
56	16	67.6, 39.3, 15.8	32.4, 60.3, 70.1	0.0, 0.4, 14.1
112	4	88.4, 62.3, 22.3	11.6, 33.2, 35.9	0.0, 4.5, 41.8

Table 7.2: Percentages of shuffled images $sh(\mathcal{A}_q^p, s)$ (1st percentage), $sh(D_k^{EA}(\mathcal{A}_q^p), s)$ (2nd percentage), and $sh(D_k^{BIM}(\mathcal{A}_q^p), s)$ (3rd percentage) for which the predicted class is c.

Contrary to what happens with $s = 32, 56$ and 112, the proportion of shuffled ancestors $sh(\mathcal{A}_q^p, s)$ classified as c_a is negligible for $s = 16$. Therefore, $s = 16$ seems to lead to patches that are too small for a 224×224 image to allow a meaningful comparison between the ancestor and the adversarials, and is consequently disregarded in the remainder of this subsection. At all other values of s, the classification of the shuffled adversarial image as a class different from c_a (c_t or other) is more common with BIM than with EA. With $s = 112$, it is noticeable that as many as 41.8% of BIM shuffled adversarials still produce targeted misclassifications. Enlarging s from 56 to 112 dramatically increases the proportion of shuffled adversarials classified as c_t with BIM (with a modest such increase with the EA), and as c_a with the EA (with a modest such increase with BIM). Moreover, the shuffled EA adversarials behave similarly to the shuffled ancestors, whose c_a probability increases considerably as the size of the patches gets larger and the original

c_a object becomes clearer (despite its shuffled aspect).

7.3.3 Summary of the outcomes

Both the EA and BIM attacks have an adversarial local effect, even at patch sizes as small as 16×16, but they generally require the image to be at full scale in order to be adversarial in the targeted sense. Still, a difference between the attacks is that, as the patch size increases (without reaching full scale, and while being subject to a shuffling process) and the c_a shape consequently becomes more obvious (even despite the shuffling), the EA's noise has a lower adversarial effect, while BIM's c_t-meaningful noise actually accumulates and has a higher global adversarial effect.

7.4 Adversarial noise visualization and frequency analysis

This section first attempts to provide a visualization of the noise that the EA and BIM add to an ancestor image to produce an adversarial image (Subsection 7.4.1). We then look more thoroughly at the frequencies of the noise introduced by the EA and BIM (Subsection 7.4.2). Finally, we look for the frequencies that are key to the adversarial nature of an image created by the EA$^{\text{target},\mathcal{C}}$ and by BIM (Subsection 7.4.3).

7.4.1 Adversarial noise visualization

The visualization of the noise that EA$^{\text{target},\mathcal{C}_k}$ and BIM add to $\mathcal{A}_q^p$ to create the 0.999-strong adversarial images $\mathcal{D}_k^{EA}(\mathcal{A}_q^p)$ and $\mathcal{D}_k^{BIM}(\mathcal{A}_q^p)$ is performed in two steps. Firstly, the difference $\mathcal{D}_k^{atk}(\mathcal{A}_q^p) - \mathcal{A}_q^p$, between each adversarial image and its ancestor, is computed for each of the RGB channels. Secondly, one displays the histogram of the adversarial noise. This leads to a measurement of the magnitude of each pixel modification. An example, typical of the general behaviour whatever the channel, is illustrated in Figure 7.2, showing the noise (the fact that the displayed dominating colors of the noise representation on Figure 7.2 are green, yellow and purple stems from the 'viridis' setting in Python's matplotlib library, which could be changed at will. Still, a scale gives the amplitude of the noise per pixel in the range $[-\epsilon, \epsilon] = [-0.03, 0.03]$, and hence justifies the position of the observed colors) and histogram of the perturbations added to the red channel of $\mathcal{A}_5^4$ to fool $\mathcal{C}_6$ with the EA and with BIM.

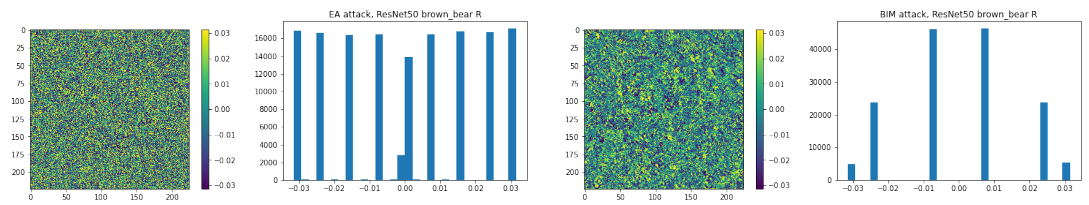


Figure 7.2: Display of the noise and histogram of the perturbations added by the EA (left pair) and by BIM (right pair) to the red channel of $\mathcal{A}_5^4$ to fool $\mathcal{C}_6$.

Recall that both attacks perform pixel perturbations with a maximum perturbation magnitude of $\epsilon = 0.03$ (see subsection 7.2.3). However, while with BIM the smaller magnitudes dominate the histogram, the adversarial noise is closer to a uniform distribution with the EA. Another difference is that, whereas with BIM all pixels are modified, a considerable amount of pixels

(9.3% on average) are not modified at all with the EA. Overall, there is a larger variety of noise magnitudes with the EA than with BIM, which can also be noticed visually in the image display of the noise.

7.4.2 Assessment of the frequencies present in the adversarial noise

The adversarial perturbations $\mathcal{D}_k^{atk}(\mathcal{A}_q^p) - \mathcal{A}_q^p$ having been assessed (subsection 7.4.1) for each RGB channel, we proceed to the analysis of the frequencies present in the adversarial noise per channel. Concretely, the Discrete Fourier Transform (DFT) is used to obtain the 2D-magnitude spectra of the adversarial perturbations. One computes two quantities, magn (diff) $= |DFT(\mathcal{D}_k^{atk}(\mathcal{A}_q^p) - \mathcal{A}_q^p)|$, and diff (magn) $= |DFT(\mathcal{D}_k^{atk}(\mathcal{A}_q^p))| - |DFT(\mathcal{A}_q^p)|$. Figure 7.3 displays a typical example of the general outcome regarding the adversarial noise in the red channel added by the EA or by BIM. For each image, the low frequencies are represented in the centre, the high frequencies in the corners, and the vertical bar (on the right) maps the frequency magnitudes to the colours shown in the image.

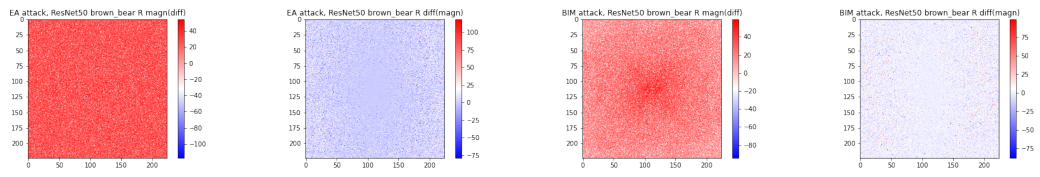


Figure 7.3: For $atk = EA$ (left pair) and $atk = BIM$ (right pair), representation of $|DFT(\mathcal{D}_6^{atk}(\mathcal{A}_5^4) - \mathcal{A}_5^4)|$ (magn (diff), 1st image) and $|DFT(\mathcal{D}_6^{atk}(\mathcal{A}_5^4))| - |DFT(\mathcal{A}_5^4)|$ (diff (magn), 2nd image) for the red channel.

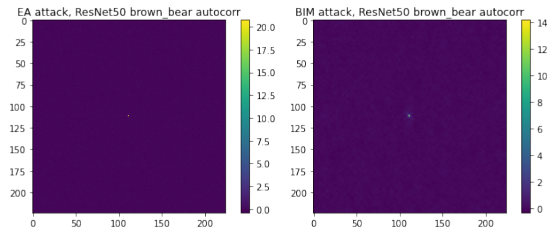


Figure 7.4: For $atk = EA$ (left) and $atk = BIM$ (right), autocorrelation of $\mathcal{D}_6^{atk}(\mathcal{A}_5^4) - \mathcal{A}_5^4$ for the red channel.

A clear difference between the EA and BIM is visible from the magn (diff) visualizations. With the EA, the high magnitudes do not appear to be concentrated in any part of the spectrum (with the exception of occasional high magnitudes in the centre), indicating the white noise nature of the added perturbations. Supporting evidence for this white noise nature for the EA comes from the $2D$ autocorrelation of the noise. Figure 7.4 shows that the $2D$ autocorrelation for both attacks have a peak at lag 0, which is to be expected. It turns out that this is the only peak when one considers the EA, which is no longer the case when one considers BIM. Unfortunately, this is hard to see on Figure 7.4, since the central peak takes very high values, hence the other peaks fade away in comparison. With BIM, the magn (diff) visualizations display considerably higher magnitudes for the low frequencies, indicating that BIM primarily makes use of low-frequency noise to create adversarial images.

In the case of diff (magn), both the EA and BIM exhibit larger magnitudes for the high frequencies than for the low frequencies. This can be interpreted as a larger effect of the adversarial noise on the high frequencies than on the low frequencies. Natural images from ImageNet have significantly more low-frequency than high-frequency information [65]. Therefore, even a quasi-uniform noise (such as the EA's) has a proportionally larger effect on the components that are numerically less present than on the more numerous ones.

7.4.3 Band-stop filtering shuffled and unshuffled images: which frequencies make an image adversarial?

So far, the results of this study divulge the quantity of all frequency components present in the adversarial perturbations, but their relevance to the attack effectiveness is still unknown. To address this issue, we band-stop filter the adversarial images $\mathcal{D}_k^{atk}(\mathcal{A}_q^p)$ to eliminate various frequency ranges, and we check the effect produced on the CNN predictions. In order to evaluate the proportion of low vs. high frequencies of the noise introduced by the two attacks, the process is repeated with the shuffled adversarials $sh(\mathcal{D}_k^{atk}(\mathcal{A}_q^p), s)$ for $s = 32, 56$ and 112.

We first obtain the DFT of all shuffled or unshuffled ancestor and adversarial images, followed by filtering with band-stop filters of 10 different frequency ranges $F_{bst,rc}$, where the range centre rc goes from 15 to 115 units per pixel, with steps of 10, and the bandwidth bw is fixed to 30 units per pixel. For example, the last band-stop filter $F_{bst,115}$ removes frequencies in the range of $(115 - 15, 115 + 15)$ units per pixel. The band-stopped images are passed through the Inverse DFT (IDFT) and sent to the CNN, which results in 10 pairs of (c_a, c_t)-label values for each image, be it an ancestor or an adversarial. Figure 7.5 presents some results that are typical of the general behaviour (also see Appendix 9.5.3 Figures 9.20 and 9.21 for the EA and Figures 9.22 and 9.23 for BIM).

For both the EA and BIM, the c_t probability tends to increase as rc becomes larger. This means that lower frequencies have a larger impact on the adversarial classification than higher frequencies. As shown on the left column of each pair of graphs, it is the low frequencies that matter for the correct classification of the ancestor, as well. Although with both attacks the c_t probability tends to increase at higher values of rc, with BIM it is dominant at considerably smaller values of rc, whereas the EA adversarials are usually still classified as c_a. Hence, the EA adversarials require almost the full spectrum of the perturbations to fool the CNNs, while the lower part of the spectrum is sufficient for the BIM adversarials. This result matches those of magn (diff) in Figure 7.3, where the EA and BIM were found to introduce white and predominantly low-frequency noise, respectively.

As for the shuffled images, it is clear that their low-frequency features are affected by the shuffling process and, as a result, the c_t probability cannot increase to the extent it does in the unshuffled images. With BIM and $s = 112$, at high rcs the band-stop graphs show a slower increase of the c_t probability than when the images are not shuffled. This means that a large part of the BIM adversarial image's low-frequency noise is meaningful only for the unshuffled image. When this low-frequency noise changes location through the shuffling process, one needs to gather noise across a broader bandwidth in order to significantly increase the c_t probability of the shuffled adversarial.

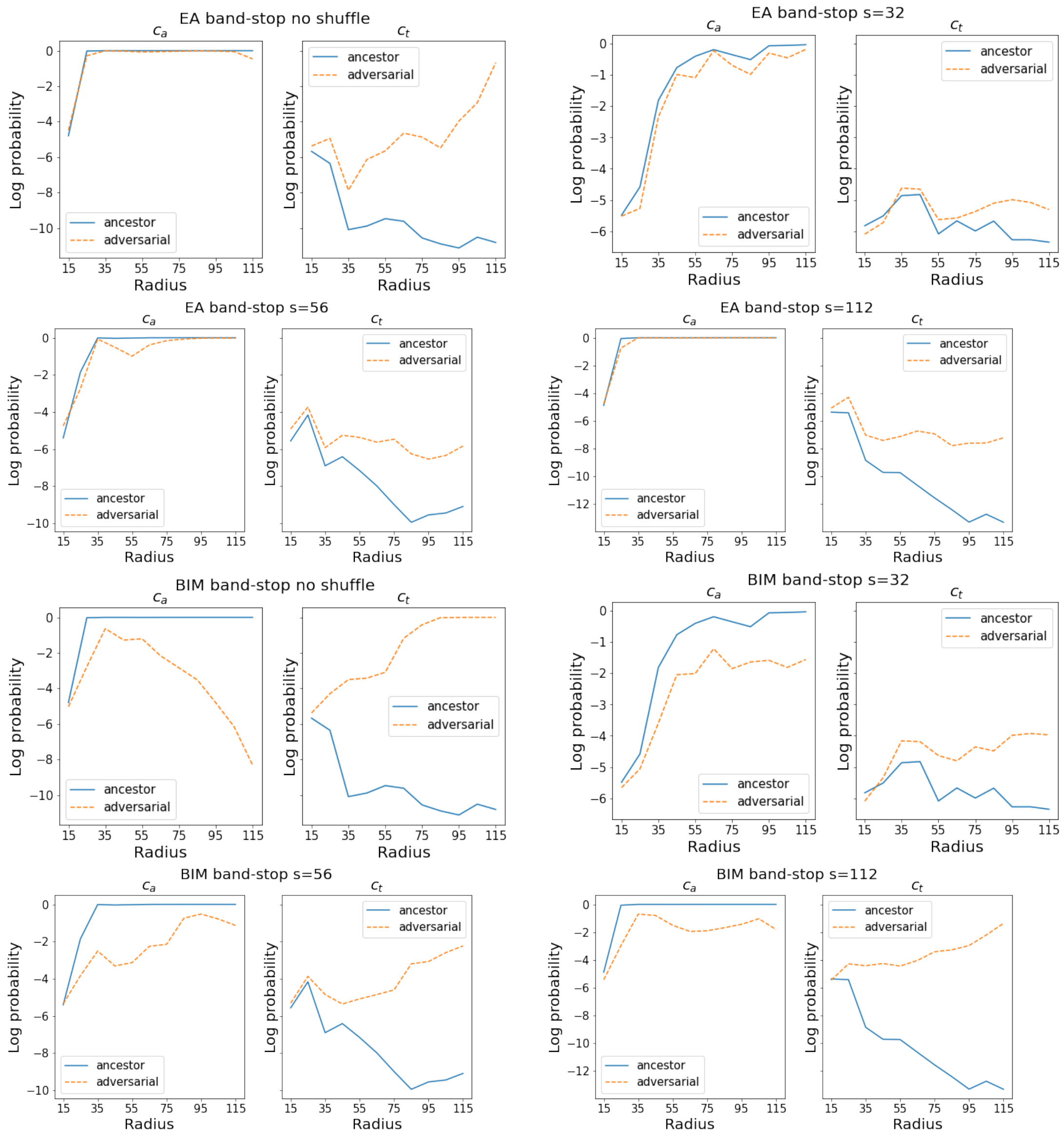


Figure 7.5: For atk = EA (1st and 2nd row) and $atk = BIM$ (3rd and 4th row), the following images are fed to $\mathcal{C}_6$: $\mathcal{A}_5^4$ and $\mathcal{D}_6^{atk}(\mathcal{A}_5^4)$ (1st pair), $sh(\mathcal{A}_5^4, 32)$ and $sh(\mathcal{D}_6^{atk}(\mathcal{A}_5^4), 32)$ (2nd pair), $sh(\mathcal{A}_5^4, 56)$ and $sh(\mathcal{D}_6^{atk}(\mathcal{A}_5^4), 56)$ (3rd pair), $sh(\mathcal{A}_5^4, 112)$ and $sh(\mathcal{D}_6^{atk}(\mathcal{A}_5^4), 112)$ (3rd pair). In each pair of graphs, the left graph displays the c_a-label values given by $\mathcal{C}_6$ as the images are band-stop filtered with bandwidths centred on different rc values, and the right graph displays the c_t-label values, *mutatis mutandis*.

Even if the BIM adversarials require a larger bandwidth in order to be adversarial when shuffled, they still reach this goal. By contrast, the shuffled EA adversarials have band-stop graphs that closely resemble the shuffled ancestors' graph. Only BIM's remaining low and middle frequencies are meaningful enough to c_t and still manage to increase the c_t probability.

7.4.4 Summary of the outcomes

The histogram of the adversarial noise introduced by BIM follows a bell shape (hence smaller magnitudes dominate) while it is closer to a uniform distribution with the EA (hence with a larger variety of noise magnitudes in this case). In addition, BIM modifies all pixels, while the EA leaves many (circa 14000 out of $224 \times 224 \times 3$, hence 9.3% on average) unchanged.

In terms of the frequency of the adversarial noise, the EA introduces white noise (meaning that all possible frequencies occur with equal magnitude), while BIM introduces predominantly low-frequency noise. Although for both attacks the lower frequencies have the highest adversarial impact, the low and middle frequencies are considerably more effective with BIM than with EA.

7.5 Transferability and texture bias

This section checks whether adversarial images are specific to their targeted CNN or whether they contain rather general features that are perceivable by other CNNs (Subsection 7.5.1). Since ImageNet-trained CNNs are biased towards texture [25], it is natural to ask whether adversarial attacks take advantage of this property. More precisely, we examine whether texture is changed by the EA and by BIM, and whether this could be the common "feature" perceived by all CNNs (Subsection 7.5.2). Thanks to heatmaps, we evaluate whether CNNs with differing amounts of texture bias agree on which image modifications have the largest adversarial impact and whether texture bias plays any role in transferability (Subsection 7.5.3).

7.5.1 Transferability of adversarial images between the 10 CNNs

For each attack $atk \in \{EA, BIM\}$, we check the transferability of the adversarial images as follows. Starting from an ancestor image $\mathcal{A}_q^p$, we input the $\mathcal{D}_k^{atk}(\mathcal{A}_q^p)$ image, which is adversarial against $\mathcal{C}_k$, to a different $\mathcal{C}_i$ (hence $i \neq k$). We then extract the probability of the dominating category, the c_t probability and the c_a probability given by $\mathcal{C}_i$ for that image.

Then, we check whether the predicted class is precisely c_t (targeted transferability), or if it is any other class different from both c_a and c_t. Out of all possible CNN pairs, our experiments showed that none of the adversarial images created by the EA for one CNN are classified by another as c_t, while this phenomenon occurs for 5.4% of the adversarial images created by BIM. As for classification in a category $c \neq c_a, c_t$, the percentages are 5.5% and 3.2% for the EA and BIM, respectively.

7.5.2 How does CNNs' texture bias influence transferability?

Knowing that CNNs trained on ImageNet are biased towards texture, we assume that a high probability for a particular class given by such a CNN expresses the fact that the input image contains more of that class's texture. Our goal is to check whether this occurs for adversarial images as well.

Table 7.3: Images, adversarial for the CNNs of the rows, are fed to the CNNs of the columns to get their c_a and c_t label values. Each cell of the Table gives a pair of numbers, the left one corresponding to $atk = EA$, and the right one to $atk = BIM$. Each number is the average difference in the c_a (Table a) and c_t (Table b) label values between the adversarial $\mathcal{D}_{T_t}^{atk}(\mathcal{A}_q^p)$ and the ancestor $\mathcal{A}_q^p$.

	T_1	T_2	T_3
T_1		-0.03, -0.05	0.01, 0.02
T_2	-0.15, -0.22		1.9e-3,-0.01
T_3	-0.15, -0.15	-0.05, -0.13	

(a) c_a

	T_1	T_2	T_3
T_1		5.6e-5, 2.0e-5	1.0e-5, 4.1e-5
T_2	2.9e-4, 1.1e-3		5.0e-5, 1.2e-4
T_3	2.4e-4,6.8e-5	5.3e-5, 5.7e-4	

(b) c_t

We restrict our study to adversarial images obtained by the EA and by BIM for the following three CNNs, that have a similar architecture, and that have been proven [30] to gradually have less texture bias and less reliance on their texture-encoding neurons : T_1 = BagNet17 [8], T_2 = ResNet50 and T_3 = ResNet50-SIN [25]. The experiments amount to checking the transferability of the adversarial images between these three CNNs. The fact that the statement about the graduation is fully proven only for these three justifies that we limit our study to them, since no such hierarchy is known for other CNNs in general.

Even in this case of three CNNs with a similar architecture, experiments show that targeted transferability between the three CNNs is 0%, whichever the attack. As a consequence, checking whether c_t becomes dominant for another CNN is pointless. We rather calculate the difference produced in a CNN's predictions of the c_t and c_a probabilities between the ancestor and another CNN's adversarial image. The average results over all images are presented in Table 7.3.

When transferring from T_2 = ResNet50 to T_1 = BagNet17, experiments show that the c_a-label value decreases, while the c_t-label value increases, with the former being larger in magnitude than the latter. If the assumption formulated in the first paragraph holds, this phenomenon implies that the attacks change image texture. Still, the similarly low transferability from T_1 = BagNet17 to T_2 = ResNet50 proves that texture change is not sufficient to generate adversarial images. The texture change observed in T_2 = ResNet50 adversarials might simply be a side-effect of the perturbations created by the EA and BIM.

Nevertheless, Table 7.3 (a) reveals that texture bias seems to play a role in transferability. It shows that the more texture-biased the CNN you are transferring the adversarial images to, the larger the decrease of its c_a-label values. Indeed, this c_a decrease is larger when transferring from T_3 = ResNet50-SIN to T_2 = ResNet50 and from T_2 = ResNet50 to T_1 = BagNet17 than vice-versa.

7.5.3 How does texture change relate to adversarial impact on the CNNs?

In this subsection, BagNet17 is used to visualize, thanks to heatmaps, whether texture change correlates to the adversarial impact of the obtained images for the 10 CNNs $\mathcal{C}_1, \cdots, \mathcal{C}_{10}$.

Although we have seen that both attacks affect BagNet17's c_a probability on average, here we attempt to find the image areas in which these changes are most prominent, and to compare the locations in the $\mathcal{C}_k$ adversarials that have the largest impact on BagNet17 and on $\mathcal{C}_k$.

To do so, we proceed in a similar way as in Subsection 7.3.1, with the difference that we allow overlaps. We replace all overlapping 17×17 patches of the ancestor $\mathcal{A}_q^p$ with patches from the same location in $\mathcal{D}_k^{atk}(\mathcal{A}_q^p)$, one single patch at a time, and we extract and store the c_a and c_t probabilities given by $\mathcal{C}_k$ of the obtained hybrid image I at each such step. Contrary to the situation in Subsection 7.3.1, note that there are as many patches as pixels in the present case. Simultaneously, these patches are also fed to BagNet17 (leading to 50176 predictions for each adversarial image) to also extract the c_a and c_t-label values of these patches. The stored c_a and c_t label values (and combinations of them) can be displayed in a square box of size 224×224 (hence of size equal to the size of the handled images), resulting in a heatmap.

More precisely, given an ancestor image $\mathcal{A}_q^p$, all hybrid adversarial images obtained as above via the EA lead to 5 heatmaps, and all those obtained by BIM lead to 5 heatmaps as well. For both attacks, the first four heatmaps are obtained thanks to BagNet17, and the fifth is obtained thanks to $\mathcal{C}_k$ for comparison purposes. Each heatmap assesses the 10% largest variations in the following sense.

We have a first sequence $(c_a(P(\mathcal{D}(\mathcal{A}))))_P$ of c_a-label values coming from the evaluation by BagNet17 of the patches of the adversarial images, and a second similar sequence $(c_a(P(\mathcal{A})))_P$ of c_a-label values coming from the patches of the ancestor images. Both sequences are naturally indexed by the successive same patch locations P. We then consider the sequence, also indexed by the patches, made of the differences $c_a(P(\mathcal{D}(\mathcal{A}))) - c_a(P(\mathcal{A}))$. The selection of the locations of the smallest 10% out of this sequence of differences leads to the first heatmap. One proceeds similarly for the second heatmap, by selecting the location of the largest 10% among the values of $c_t(P(\mathcal{D}(\mathcal{A}))) - c_t(P(\mathcal{A}))$ (with obvious notations). The process is similar for the third and fourth heatmaps, where one considers the location of the largest 10% of the values of $c_a(P(\mathcal{D}(\mathcal{A}))) - c_t(P(\mathcal{D}(\mathcal{A})))$ for the third heatmap, and of the values of $c_t(P(\mathcal{D}(\mathcal{A}))) - c_a(P(\mathcal{D}(\mathcal{A})))$ for the fourth heatmap.

Finally, the fifth heatmap is obtained by considering the largest 10% among the values $c_t(I_{P(\mathcal{D}(\mathcal{A}))}) - c_t(\mathcal{A})$, where the two members of the difference are the c_t-label values given by the CNN $\mathcal{C}_k$ for a full image, the right one for the ancestor image, and the left one for the hybrid image obtained as explained above.

Figure 7.6 shows the outcome of this process for $\mathcal{C}_6$= ResNet50 and ancestor $\mathcal{A}_{10}^8$ (see Figure 9.24 in Appendix 9.5.4 for other examples). Figure 7.6 (a) shows the adversarial images $\mathcal{D}_6^{EA}(\mathcal{A}_{10}^8)$ (top) and $\mathcal{D}_6^{BIM}(\mathcal{A}_{10}^8)$ (bottom). Figure 7.6 (b) to (e) are the four heatmaps obtained thanks to BagNet17 (in the order stated above), and Figure 7.6 (f) is the heatmap obtained thanks to $\mathcal{C}_6$ (the top row corresponds to the EA, and the bottom row to BIM). The 10% largest variations are represented by the yellow points in each heatmap.

With both attacks, actually stronger with BIM than with the EA, modifying the images in and around the object locations is the most effective at increasing $\mathcal{C}_k$'s c_t probability, as shown in Figure 7.6 (f).

For both attacks, the locations of c_a texture decrease coincide with the locations of most adversarial impact for $\mathcal{C}_k$ (Figure 7.6 (b) and (f)), while the c_t texture increase is slightly more disorganised, being distributed across more image areas (Figure 7.6 (c)). Still, even though the c_a texture decreases, it remains dominant in the areas where the c_a shape is also present (Figure

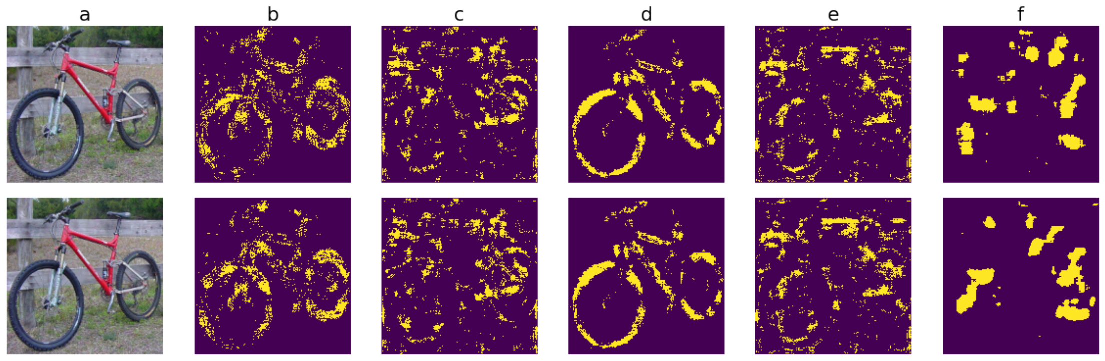


Figure 7.6: Heatmaps obtained with the ancestor $\mathcal{A} = \mathcal{A}_{10}^{8}$ and the adversarial image $\mathcal{D}(\mathcal{A}) = \mathcal{D}_{6}^{atk}(\mathcal{A}_{10}^{8})$ pictured in (a), where atk = EA in the 1^{st} row and atk = BIM in the 2^{nd} row.

7.6 (d)), without being replaced by the c_t texture, which only dominates in other, less c_a object-related areas (Figure 7.6 (e)). The c_a texture and shape coupling encourages the classification of the image into c_a, which may explain why the adversarial images are not transferable.

7.5.4 Summary of the outcomes

Both attacks' adversarial images are generally not transferable in the targeted sense. Although some c_a texture is distorted by the attacks, the c_t texture is not significantly increased (while the opposite is true for the targeted CNNs' c_a and c_t probabilities) and this increase is nevertheless not correlated with an adversarial impact on the CNNs. However, we find that EA's and BIM's adversarial images transfer more to CNNs which have higher texture bias.

7.6 Transferability of the adversarial noise at smaller image regions

On the one hand, the very low transferability rate observed in Section 7.5 shows that most obtained adversarial images are specific to the CNNs they fool. On the other hand, the size of the covered region increases linearly with successive CNN layers [41]. Moreover, the similarity between the features captured by different CNNs is higher in earlier layers than in later layers [37, 44]. Roughly said, the earlier layers tend to capture information of a general nature, common to all CNNs, while the features captured by the later layers diverge from one CNN to another.

The question addressed in this section goes in the direction of a potential stratification of the adversarial noise's impact according to the successive layers of the CNNs. Put differently, this amounts to clarifying whether it is possible to sieve the adversarial noise, so that one would identify the part of the noise (if any) that has an adversarial impact for all CNNs up to some layers, and the part of the noise whose adversarial impact becomes CNN-specific from some layer on. This is a difficult challenge since the adversarial noise is modified all the way along until a convenient adversarial image is created. In particular, the "initial" noise, created at some early point of the process and potentially adversarial for the first layers of different CNNs, is likely to be modified as well during this process, and to lose its initial "quasi-universal" adversarial characteristic, potentially to the benefit of a new adversarial noise. Note *en passant* that a careful study in this direction may contribute to "reverse engineer" a CNN, namely to reconstruct its architecture (up to a point). This direction is only indicated here, and is not explored in full details at this stage.

More modestly, and more specifically, in this Section we ask whether the adversarial noise for regions of smaller sizes is less CNN-specific, hence more transferable, than at full scale, namely 224×224 in the present case, where we know that, in general, it is not transferable.

This issue is addressed in two ways. First, we check whether and how a modification of the adversarial noise intensity affects the c_a and the c_t-label values of an image, adversarial for a given CNN, when exposed to a different CNN, and the influence of shuffling in this process (Subsection 7.6.1). Secondly, we keep the adversarial noise as it is, and we check whether adversarial images are more likely to transfer when they are shuffled (Subsection 7.6.2).

7.6.1 Generic versus specific direction of the adversarial noise

One is given a convenient ancestor image $\mathcal{A}_q^p$, a CNN $\mathcal{C}_k$, and the adversarial images $\mathcal{D}_k^{EA}(\mathcal{A}_q^p)$ and $\mathcal{D}_k^{BIM}(\mathcal{A}_q^p)$ obtained by both attacks.

We perform a first series of experiments, that consists in changing the adversarial noise magnitude of these adversarial images by a factor f in the $0\% - 300\%$ range, and in submitting the corresponding modified f-boosted adversarial images to different $\mathcal{C}_i$'s to check whether they fool them.

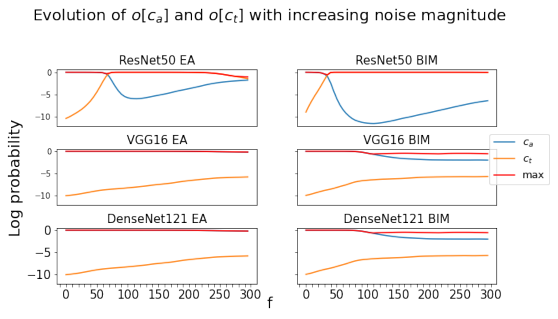
Figure 7.7 (a) shows what happens for the particular case of $\mathcal{A}_5^4$, $k = 6$, and the $\mathcal{C}_i$'s equal to $\mathcal{C}_1, \mathcal{C}_6$, and $\mathcal{C}_9$ (the f-boosted adversarial image is sent back to $\mathcal{C}_6$ as well), representative of the general behaviour. In particular, it shows that the direction of the created noise for the EA adversarials is highly specific to the targeted CNN since the images cannot be made transferable by any change in magnitude. *A contrario*, the noise of BIM's adversarials has a more general direction, since amplifying its magnitude eventually leads to untargeted misclassifications by other CNNs.

A second series of experiments is performed in a similar way as above, with the difference that this time, it is on the shuffled adversarial images $sh(\mathcal{D}_k^{EA}(\mathcal{A}_q^p), s)$ and $sh(\mathcal{D}_k^{BIM}(\mathcal{A}_q^p), s)$ for $s = 32$, 56 or 112.

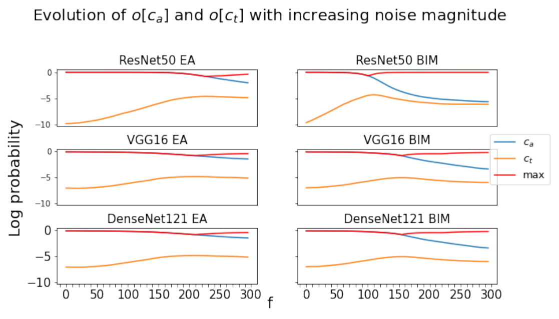
Figures 7.7 (b), (c), (d) show the typical outcome of this experiment. It reveals another difference between the adversarial noise obtained by the two attacks, namely that when s is enlarged from 32 to 56 and to 112, BIM images have a higher adversarial effect on other CNNs, whereas EA images only have a higher adversarial effect when s is increased from 32 to 56. As the size of the shuffled boxes increases to $s = 112$ and reveals the ancestor object more clearly, the EA adversarials actually have a lower fooling effect on other CNNs.

Moreover, in contrast to Figure 7.7 (a) where the considered region is at sull scale, i.e. coincides with the full image size, Figures 7.7 (b), (c), and (d) show that the noise direction is more general at the local level, and that an amplification of the noise magnitude is able to lead the adversarial images outside of other CNNs' c_a bounds, even with the EA.

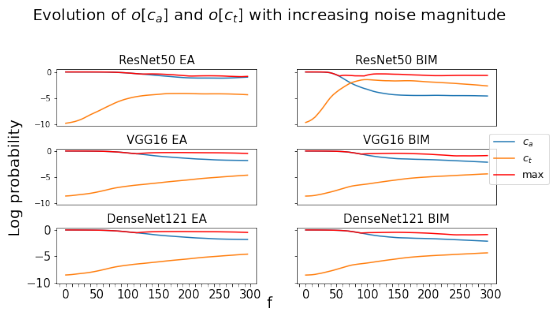
To make sure that the observed effects were not simply due to shuffling, but were also due to the adversarial noise, we repeated the experiment of Figure 7.7 with random normal noise. As



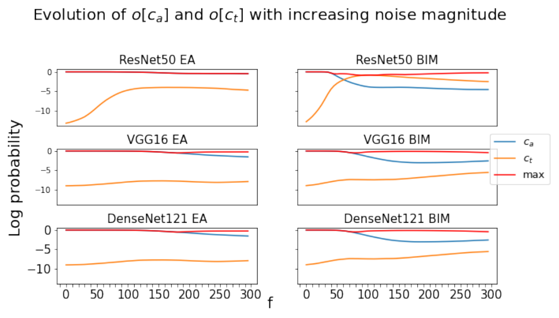
(a) no shuffle



(b) $s = 32$



(c) $s = 56$



(d) $s = 112$

Figure 7.7: Evolution of $log(o[a])$, $log(o[t])$, and $log(max(o))$ for $\mathcal{D}_6^{atk}(\mathcal{A}_5^4)$ (a), and for $sh(\mathcal{D}_6^{atk}(\mathcal{A}_5^4), s)$ for $s = 32$ (b), $s = 56$ (c) and $s = 112$ (d) when fed to $\mathcal{C}_6$, $\mathcal{C}_9$ and $\mathcal{C}_1$ (1^{st}, 2^{nd} and 3^{rd} row of each set of graphs, respectively), when the noise is impacted by a factor $f \in [0\%, 300\%]$.

expected, it turned out that, with random noise, the c_a-label value always remained dominant and the c_t-label value barely increased as f varied from 0% to 300% (see Figure 9.25 in Appendix 9.5.5). The close to zero impact of random noise on unshuffled images was already known [22]. These experiments confirm that this also holds for shuffled images. Therefore, the observed effects were indeed due to the adversarial noise's transferability at local level. Nevertheless, although the adversarial noise is general enough to affect other CNNs' c_a-label values, its effect on c_t-label values is never as strong as for the targeted CNN.

7.6.2 Effects of shuffling on adversarial images' transferability

Here, we do not change the intensity of the noise. In other words, $f = 100\%$. The point is no longer to visualize the graph of the evolution of the c_t-label values of shuffled adversarials, but to focus on their actual values for the "real" noise (at $f = 100\%$). The issue is to check if the adversarial images are more likely to transfer when they are shuffled.

We proceed as follows. We input the unshuffled ancestor $\mathcal{A}_q^p$ and the unshuffled adversarial $\mathcal{D}_k^{atk}(\mathcal{A}_q^p)$ to all $\mathcal{C}_i$'s for $i \neq k$ (hence all CNNs but the targeted one). We extract the c_a and c_t-label values for each i, namely $c_a^i(\mathcal{A}_q^p)$, $c_a^i(\mathcal{D}_k^{atk}(\mathcal{A}_q^p))$, $c_t^i(\mathcal{A}_q^p)$ and $c_t^i(\mathcal{D}_k^{atk}(\mathcal{A}_q^p))$. We compute the difference of the c_a-label values between the two images for each i, and, similarly, the difference of the c_t-label values, to get

$$\Delta_a^{k,i}(\mathcal{A}_q^p) = c_a^i(\mathcal{D}_k^{atk}(\mathcal{A}_q^p)) - c_a^i(\mathcal{A}_q^p), \quad \Delta_t^{k,i}(\mathcal{A}_q^p) = c_t^i(\mathcal{D}_k^{atk}(\mathcal{A}_q^p)) - c_t^i(\mathcal{A}_q^p).$$

For $s = 32, 56$ and 112, this process is repeated with the shuffled ancestor $sh(\mathcal{A}_q^p, s)$ and the shuffled adversarial $sh(\mathcal{D}_k^{atk}(\mathcal{A}_q^p), s)$, and we get the differences:

$$\Delta_a^{k,i}(sh(\mathcal{A}_q^p, s)) = c_a^i(sh(\mathcal{D}_k^{atk}(\mathcal{A}_q^p), s)) - c_a^i(sh(\mathcal{A}_q^p, s)),$$

and

$$\Delta_t^{k,i}(sh(\mathcal{D}_k^{atk}(\mathcal{A}_q^p), s)) = c_t^i(sh(\mathcal{D}_k^{atk}(\mathcal{A}_q^p), s)) - c_t^i(sh(\mathcal{A}_q^p), s).$$

These $\Delta^{k,i}$ assess the impact of the adversarial noise both when unshuffled and when shuffled.

Regarding the c_a-label values, both differences are ≤ 0 (the c_a-label value of the ancestor dominates the c_a-label value of the adversarial, shuffled or not). We take the absolute value of both quantities (k and i are fixed). Finally, we compute the percentage over all k, all $i \neq k$, of all convenient ancestors $\mathcal{A}_q^p$ for which

$$|\Delta_a^{k,i}(sh(\mathcal{A}_q^p, s))| \geq |\Delta_a^{k,i}(\mathcal{A}_q^p)|.$$

Regarding the c_t-label values, both differences are ≥ 0 (this is obvious for the unshuffled images, but it also turns out to be the case for the shuffled one). In this case, there is therefore no need to take the absolute values. We compute the percentage over all k, all $i \neq k$, of all convenient ancestors $\mathcal{A}_q^p$ for which

$$\Delta_t^{k,i}(sh(\mathcal{A}_q^p, s)) \geq \Delta_t^{k,i}(\mathcal{A}_q^p).$$

Table 7.4 presents the outcome of these computations for each value of s, and for the adversarials obtained by both attacks.

Note that we do not simply present the c_t-label values of shuffled adversarial images, because then the measured impact could have two sources: either the adversarial noise or the fact that the c_a shape is distorted by shuffling, leaving room for the c_t-label value to increase. Since our goal is to only measure the former source, we compare the c_t-label values of shuffled adversarials with that of shuffled ancestors.

Table 7.4: For the c_a-label value (2nd row) and the c_t-label value (3rd row), percentage of cases where the adversarial noise has a stronger impact when shuffled than unshuffled. In each cell, the 1st percentage corresponds to $atk = EA$, and the 2nd to $atk = BIM$.

	$s = 32$	$s = 56$	$s = 112$
$\lvert\Delta_a^{k,i}(sh(\mathcal{A}_q^p, s))\rvert \geq \lvert\Delta_a^{k,i}(\mathcal{A}_q^p)\rvert$	52.02, 45.41	66.94, 64.29	57.58, 54.67
$\Delta_t^{k,i}(sh(\mathcal{A}_q^p, s)) \geq \Delta_t^{k,i}(\mathcal{A}_q^p)$	52.69, 49.79	65.09, 58.37	48.24, 43.11

Whenever the percentage is larger than 50%, the adversarial images have on average a stronger adversarial effect (for the untargeted scenario if one considers $\Delta_a^{k,i}$, and for the target scenario if one considers $\Delta_t^{k,i}$) when shuffled than when they are not. The adversarial effect is then perceived more by other CNNs for regions of the corresponding same size than at full scale.

For all values of s, the 1st percentage is larger than the 2nd one. This means that distorting the shape of the ancestor object (done by shuffling) helps the EA more than BIM towards fooling other CNNs than the targeted $\mathcal{C}_k$. This occurs although computation shows that shuffled BIM adversarials are typically classified with a larger c_t-label value than shuffled EA adversarials.

The percentages achieved with $s = 56$ are not only the largest as compared to those with $s = 32$ or 112, but they also exceed 50% by far. Said otherwise, a region size of 56×56 achieves some optimum here. An interpretation could be that a region of that size is small enough to distort the c_a - related information more, while also being large enough to enable the adversarial pixel perturbations to join forces and create adversarial features with a larger impact on different CNNs than the targeted one.

7.6.3 Summary of the outcomes

The direction of the created adversarial noise for the EA adversarials is very specific to the targeted CNN. No change of magnitude in the adversarial noise make them more transferable. The situation differs to some extent with the noise of BIM's adversarials. This latter noise has a more general direction, since its amplification leads to untargeted misclassifications by other CNNs. When images are shuffled, and the noise is intensified, BIM's adversarials have a higher adversarial effect on other CNNs as s grows. This is also the case with the EA's adversarials as s grows from 32 to 56, but not anymore when s grows from 56 to 112.

A second outcome is that the EA and BIM adversarial images get closer to being transferable in a targeted sense when shuffled with $s = 56$ than when unshuffled (at their full scale), and that $s = 56$ is optimal in this regard as compared to $s = 32$ or 112. In the untargeted sense, this happens at regions of sizes 56×56, and 112×112 (for both attacks the corresponding percentages exceed 50%).

7.7 Penultimate layer activations with adversarial images

In this section, we intend to closely examine (in Subsection 7.7.2) the changes that adversarial images produce in the activation of the CNNs' penultimate layers (for reasons explained in Subsection 7.7.1). In the work that led to this paper, we performed a similar study on the activation changes of the CNNs' convolutional layers. However, to the difference of what happens with the penultimate layers, the results obtained with adversarial noise were identical to those obtained with random noise. Hence, visualizing the intermediate layer activations requires a more in-depth method than the one employed here, and we restrict the current paper to the study of the penultimate classification layers.

It is important to notice that we do not pay attention here to the black-box or white-box nature of the attack. We use the adversarial images independently on how they are obtained. Indeed, we assume full access to the architectures of the CNNs. This full access to the CNNs' architectures goes without saying when one considers BIM, since it is a prerequisite for this attack. But it is worthwhile explicitly stating for the EA, since the EA attack excludes any *a priori* knowledge of the CNNs' architectures.

Still, the study of the way layers are activated by the adversarial images may reveal differences of behavior according to the methods used to construct them. It is not excluded that patterns of layer activations differ according to the white-box or black-box nature of the attack that created the adversarial images sent to the CNNs. Should it be the case, this difference of patterns according to the nature of the attack may lead to attack detection measures or even to protection measures. The study of this issue is not undertaken in this paper.

7.7.1 Relevance of analyzing the activation of c_t- and of c_a-related units

The features extracted by the convolutional CNN layers pass through the next group of layers of the CNN, namely the classification layers. We focus here on the penultimate classification layer, *i.e.* the layer just before the last one that gives the output classification vector.

When a CNN $\mathcal{C}$ is exposed to an adversarial image $\mathcal{D}(\mathcal{A})$, the perturbation of the features propagates and modifies the activation of the classification layers, which in turn leads to an output vector $o^{\mathcal{C}}_{\mathcal{D}(\mathcal{A})}$ (drastically different from the output vector $o^{\mathcal{C}}_{\mathcal{A}}$ for the ancestor) in which the probability corresponding to the target class c_t is dominant. To achieve this result, it is certain that previous classification layers are modified in a meaningful manner, with higher activations of the units that are relevant to c_t.

However, it is not obvious how the changes in these classification layers take place. Since the penultimate layer has a direct connection with the final layer and the impact of changes in activation are thus traceable, we delve here into the activations of the CNNs' penultimate layers to answer two questions essentially: Do all c_t-relevant units have increased activation? Do c_a-related units have decreased activations?

The connection between the penultimate and final layers is made through a weight vector W, which, for each class in the output vector, provides the weights by which to multiply the penultimate layer's activation values. Whenever a weight that connects one penultimate layer unit with one class is positive, that particular unit of the penultimate layer is indicative of that class' presence in the image, and vice-versa for negative weights. We can thus know which penultimate layer units are c_a- or c_t-related.

7.7.2 How are the CNNs' classification layers affected by adversarial images?

For each CNN $\mathcal{C}_k$, we do the following. The aforementioned weights are extracted and, for both c_a and c_t, they are separated into positive and negative weights. Then, we compute the difference of activation values in the penultimate layer between each adversarial $\mathcal{D}^{atk}_k(\mathcal{A}^p_q)$ and its ancestor $\mathcal{A}^p_q$. Since our intention is to measure the proportion of units, relevant to a class, that are increased or decreased by the adversarial noise, we compute the average percentage of both positively- and negatively-related units – Table 7.5 for c_a and Table 7.6 for c_t (see Tables 9.21 and 9.22 in Appendix 9.5.6 for an individual outcome) – whose activation increased, stagnated or decreased. For c_a and c_t, Table 7.7 presents the average change in penultimate layer activation for both the positively- and negatively-related units.

Note *en passant* that $\mathcal{C}_9$ and $\mathcal{C}_{10}$ have a different behavior than the other CNNs as far as the values of $W_{pos}\Delta_0$ and $W_{neg}\Delta_0$ are concerned. The EA and BIM change the activations of $\mathcal{C}_9$ and $\mathcal{C}_{10}$ much less frequently than with the other CNNs. Indeed, between 50.28% and 74.85% of the activations of these two CNNs are left unchanged, and this is valid for c_a and for c_t and for both attacks. Observe that the group of units that contribute to the values taken by $W_{pos}\Delta_0$ and by $W_{neg}\Delta_0$ for c_a coincides with the group of units that contribute to the values taken by

	For c_a	$W_{pos}\Delta_{pos}$	$W_{pos}\Delta_0$	$W_{pos}\Delta_{neg}$	$W_{neg}\Delta_{pos}$	$W_{neg}\Delta_0$	$W_{neg}\Delta_{neg}$
$\mathcal{C}_1$	DenseNet121	(51.28,48.83)	(0.02,0.04)	(48.70,51.13)	(65.36,65.19)	(0.13,0.09)	(34.51,34.72)
$\mathcal{C}_2$	DenseNet169	(49.03,49.26)	(0.03,0.03)	(50.95,50.72)	(62.42,61.13)	(0.11,0.05)	(37.47,38.82)
$\mathcal{C}_3$	DenseNet201	(49.48,48.66)	(0.07,0.03)	(50.45,51.31)	(61.04,60.45)	(0.06,0.06)	(38.90,39.49)
$\mathcal{C}_4$	MobileNet	(43.73,46.59)	(0.46,0.31)	(55.81,53.10)	(62.64,65.80)	(1.24,0.71)	(36.12,33.48)
$\mathcal{C}_5$	MNASNet	(47.64,49.93)	(4.81,3.43)	(47.55,46.64)	(58.34,61.04)	(8.77,6.17)	(32.89,32.79)
$\mathcal{C}_6$	ResNet50	(45.80,45.61)	(0.02,0.00)	(54.17,54.39)	(65.86,66.11)	(0.02,0.00)	(34.13,33.89)
$\mathcal{C}_7$	ResNet101	(48.26,46.05)	(0.01,0.00)	(51.73,53.95)	(67.63,67.75)	(0.05,0.02)	(32.32,32.23)
$\mathcal{C}_8$	ResNet152	(46.84,45.56)	(0.00,0.00)	(53.16,54.44)	(67.18,66.92)	(0.01,0.00)	(32.81,33.08)
$\mathcal{C}_9$	VGG16	(23.67,19.64)	(50.63,52.98)	(25.71,27.39)	(22.08,19.15)	(72.50,74.85)	(5.43,6.01)
$\mathcal{C}_{10}$	VGG19	(23.85,20.38)	(50.28,51.30)	(25.87,28.33)	(21.44,20.46)	(73.02,73.23)	(5.54,6.31)

Table 7.5: For c_a, average percentage of both positively-related (W_{pos}, columns 2-4) and negatively-related (W_{neg}, columns 5-7) units whose activation increased (Δ_{pos}), stagnated (Δ_0) or decreased (Δ_{neg}). Each cell contains the results for EA (left) and BIM (right).

	For c_t	$W_{pos}\Delta_{pos}$	$W_{pos}\Delta_0$	$W_{pos}\Delta_{neg}$	$W_{neg}\Delta_{pos}$	$W_{neg}\Delta_0$	$W_{neg}\Delta_{neg}$
$\mathcal{C}_1$	DenseNet121	(70.06,67.06)	(0.04,0.04)	(29.90,32.90)	(50.00,50.30)	(0.12,0.10)	(49.89,49.60)
$\mathcal{C}_2$	DenseNet169	(64.88,64.26)	(0.03,0.03)	(35.09,35.72)	(49.32,48.95)	(0.11,0.05)	(50.58,51.01)
$\mathcal{C}_3$	DenseNet201	(64.72,63.61)	(0.08,0.06)	(35.21,36.32)	(48.80,48.52)	(0.05,0.03)	(51.15,51.45)
$\mathcal{C}_4$	MobileNet	(69.74,71.62)	(0.52,0.31)	(29.74,28.07)	(38.13,42.25)	(1.24,0.74)	(60.63,57.01)
$\mathcal{C}_5$	MNASNet	(64.51,66.94)	(5.60,3.50)	(29.89,29.56)	(42.75,45.32)	(8.13,6.23)	(49.12,48.44)
$\mathcal{C}_6$	ResNet50	(75.16,73.24)	(0.01,0.00)	(24.82,26.76)	(46.29,47.73)	(0.03,0.00)	(53.69,52.27)
$\mathcal{C}_7$	ResNet101	(77.68,74.77)	(0.01,0.00)	(22.31,25.23)	(48.16,48.78)	(0.05,0.02)	(51.79,51.20)
$\mathcal{C}_8$	ResNet152	(75.37,73.43)	(0.00,0.00)	(24.63,26.57)	(48.21,48.39)	(0.01,0.00)	(51.78,51.61)
$\mathcal{C}_9$	VGG16	(35.62,31.40)	(52.69,55.55)	(11.70,13.04)	(12.51,9.81)	(70.75,72.61)	(16.74,17.58)
$\mathcal{C}_{10}$	VGG19	(35.35,32.82)	(53.13,53.65)	(11.52,13.53)	(12.13,10.51)	(70.80,71.31)	(17.06,18.17)

Table 7.6: For c_t, average percentage of both positively-related (W_{pos}, columns 2-4) and negatively-related (W_{neg}, columns 5-7) units whose activation increased (Δ_{pos}), stagnated (Δ_0) or decreased (Δ_{neg}). Each cell contains the results for EA (left) and BIM (right).

		W_{pos}	W_{neg}
$\mathcal{C}_1$	DenseNet121	(-0.02±0.07, -0.05±0.09)	(0.17±0.05, 0.21±0.06)
$\mathcal{C}_2$	DenseNet169	(0.01±0.06, -0.01±0.06)	(0.13±0.03, 0.14±0.06)
$\mathcal{C}_3$	DenseNet201	(0.00±0.05, 0.00±0.06)	(0.09±0.04, 0.12±0.04)
$\mathcal{C}_4$	MobileNet	(-0.09±0.10, -0.05±0.16)	(0.18±0.06, 0.26±0.10)
$\mathcal{C}_5$	MNASNet	(-0.01±0.07, 0.02±0.09)	(0.12±0.04, 0.18±0.09)
$\mathcal{C}_6$	ResNet50	(-0.02±0.07, -0.05±0.09)	(0.17±0.05, 0.20±0.08)
$\mathcal{C}_7$	ResNet101	(0.00±0.07, -0.04±0.15)	(0.21±0.05, 0.25±0.10)
$\mathcal{C}_8$	ResNet152	(-0.02±0.07, -0.06±0.07)	(0.21±0.04, 0.22±0.04)
$\mathcal{C}_9$	VGG16	(-0.14±0.15, -0.16±0.20)	(0.20±0.05, 0.23±0.10)
$\mathcal{C}_{10}$	VGG19	(-0.09±0.08, -0.17±0.12)	(0.21±0.06, 0.20±0.07)

(a) c_a

		W_{pos}	W_{neg}
$\mathcal{C}_1$	DenseNet121	(0.27±0.08, 0.30±0.10)	(-0.06±0.06, -0.07±0.08)
$\mathcal{C}_2$	DenseNet169	(0.18±0.05, 0.20±0.09)	(-0.02±0.05, -0.03±0.05)
$\mathcal{C}_3$	DenseNet201	(0.15±0.05, 0.17±0.05)	(-0.02±0.03, -0.03±0.04)
$\mathcal{C}_4$	MobileNet	(0.27±0.06, 0.38±0.14)	(-0.16±0.08, -0.15±0.11)
$\mathcal{C}_5$	MNASNet	(0.19±0.05, 0.28±0.12)	(-0.06±0.06, -0.06±0.07)
$\mathcal{C}_6$	ResNet50	(0.30±0.11, 0.32±0.14)	(-0.04±0.04, -0.05±0.06)
$\mathcal{C}_7$	ResNet101	(0.35±0.08, 0.39±0.17)	(-0.03±0.05, -0.04± 0.07)
$\mathcal{C}_8$	ResNet152	(0.33±0.05, 0.35±0.09)	(-0.03±0.04, -0.05±0.04)
$\mathcal{C}_9$	VGG16	(0.29±0.14, 0.35±0.28)	(-0.14±0.06, -0.17±0.07)
$\mathcal{C}_{10}$	VGG19	(0.33±0.10, 0.29±0.18)	(-0.13±0.05, -0.17±0.04)

(b) c_t

Table 7.7: For c_a (a) and c_t (b), average and standard deviation of the activation change in the positively-related (W_{pos}, column 2) and negatively-related (W_{neg}, column 3) units. Each cell contains the results for EA (left) and BIM (right).

$W_{pos}\Delta_0$ and by $W_{neg}\Delta_0$ for c_t.

Overall, Tables 7.5 and 7.6 show that neither the EA, nor BIM increase the activation of all positively c_t-related penultimate layer units; the percentages where such an increase happens is similar between the two attacks, and varies between 31.40% and 77.68% throughout the different CNNs. Still, in all cases, more positively c_t-related units are increased, rather than decreased in activation. Meanwhile, for c_a, this preference for increasing, rather than decreasing the activation is present for the negatively c_a-related units.

These observations are consistent with the results of Table 7.7, which show that the average activation changes are large and positive for (W_{neg},c_a) and (W_{pos},c_t) for both attacks. Additionally, the averages and standard deviations corresponding to (W_{pos}, c_t) are higher than those corresponding to (W_{neg}, c_a), with both attacks. However, both the averages and the standard deviations are larger with BIM than with the EA.

In order to verify how the penultimate layer activations of a CNN are changed by adversarial images that are designed for other CNNs, we perform the experiments that led to Tables 7.5 and 7.6 with the change that all CNNs are fed the adversarial images of $\mathcal{C}_1$ (DenseNet121). The results (see Tables 9.21 and 9.22 in Appendix 9.5.6) are that, with both attacks, the percentages of positive and negative activation changes are approximately equal. Therefore, the pixel perturbations are not necessarily meaningful towards decreasing the c_a-label value or increasing the c_t-label value of other CNNs.

Therefore, it appears that the attacks do not significantly impact the existing positively c_a-related features. They rather create some features that relate negatively to c_a and some that increase the confidence for c_t. Also, although both attacks usually (except against $\mathcal{C}_1$ and $\mathcal{C}_2$, where the proportion is only around one third) increase the activation of around two thirds of the positively c_t-related and negatively c_a-related units, BIM does so with a larger magnitude than the EA. The latter change is in particular the most striking difference between the attacks. It could explain why the band-stop graphs in Figure 7.5 show a much larger decrease of the c_a-label value with BIM than with the EA, and why BIM adversarial images are more likely to transfer than those coming from the EA.

7.7.3 Summary of the outcomes

In terms of the penultimate layer, the most prominent changes of both attacks are the increase in activation value of the units which are positively related to c_t and of those which are negatively related to c_a. However, BIM performs the latter activation changes with a larger magnitude than the EA.

7.8 Summary of the outcomes

Through the lens of frequency, transferability, texture change, smaller image regions and penultimate layer activations, this work investigates the properties that make an image adversarial against a CNN. To this purpose, we consider 10 ImageNet-trained CNNs, and the adversarial images created by a white-box, gradient-based attack and by a black-box, evolutionary algorithm-based attack, to fool them.

This study, performed on 84 convenient ancestor images belonging to appropriate ancestor categories c_as, and two groups of 437 adversarial images (one group per attack) belonging to appropriate target categories c_ts, gives an insight into the internal functioning of the considered attacks.

The main outcomes are that the aggregation of features at smaller regions is generally not sufficient for a targeted misclassification. We also find that image texture change is likely to be a side-effect of the attacks, rather than play a crucial role, even though EA and BIM adversarials are more likely to transfer to more texture-biased CNNs. While the low-frequency noise has the highest adversarial effect for both attacks, in contrast to the EA's white noise, BIM's mostly low-frequency noise impacts the local c_a features considerably more than the EA does. This effect intensifies at larger regions.

Although in the penultimate CNN layers neither the EA, nor BIM affect the features that are positively related to c_a, BIM's gradient-based nature allows it to find noise directions that are more general across different CNNs, introducing more features that are negatively related to c_a, and that are perceivable by other CNNs as well. However, with both attacks, the c_t-related adversarial noise that targets the final CNN layers is specific to the targeted CNN when the adversarial images are at full scale. On the other hand, its adversarial impact on other CNNs increases when the considered region is reduced from full scale to 56×56.

Chapter 8

Conclusion and Perspectives

The broad topic of this thesis falls under adversarial attacks in the context of image classification by Convolutional Neural Networks (CNNs). Specifically, this work introduces a black-box attack based on an evolutionary algorithm (EA) that is designed to evolve an original image into an adversarial one against CNNs. The introduced algorithm is generic, since it can be applied on any original image to fool any CNN in any scenario tested here, such as untargeted, *flat*, *good enough* targeted, or τ-strong targeted. Moreover, its black-box nature makes the EA-based attack highly practical, since the attacker only requires access to the input and the vector of probabilities outputted by the CNN.

After describing the attack method, this thesis proves the algorithm's high success rate at attacking VGG16 trained on the Cifar10 dataset [34] in both the targeted and flat scenarios, as well as at attacking 10 different CNNs trained on the ImageNet [18] dataset. Moreover, this work proposes and experimentally evaluates two different ways to apply the EA-based attack on high-resolution images, which have much greater sizes than the images typically contained in the Cifar10 and ImageNet datasets.

To assure that the EA-based attack could not be defended against by image filters placed in front of the CNNs, we thoroughly evaluate the algorithm's robustness in the specific context of VGG16 trained on Cifar10. After discovering a set of filters which could reduce the attack's success rate, we return to the construction of the EA method and improve it such that it becomes robust to all tested filters.

Lastly, part of the goal of this thesis is to understand the inner functioning of the EA-based attack and to find the reasons for its high success rate at fooling CNNs. In an effort towards this goal, we perform several experiments with the attacked CNNs, as well as with the adversarial images and their contained noise. In studying them, we adopt perspectives such as image frequency, image transferability, behaviour at lower image regions, penultimate CNN layer activations, and image texture change. In order to place the findings into a broader context, we simultaneously perform all above-mentioned experiments on an attack with a similarly high success rate, but which differs greatly from the EA-based attack, since its white-box, gradient-based nature requires access to all parameters of the CNN to be fooled.

Although this thesis is extensive, it is not comprehensive, as it can surely be expanded in multiple ways. Starting from one of the findings of this thesis, one option is to investigate the efficiency of an attack detection algorithm based on image shuffling. Additionally, the attack on high-resolution images can be improved in efficiency, by using attention maps to find the specific image areas where the CNNs place more importance and then reduce the search space of the EA-based attack to these particular regions. Another possible direction towards a higher explainability of the EA-based attack is the small-dimension visualization of the attacked CNNs' decision boundaries. Yet another option would be to apply the EA-based attack on Spiking Neural Networks, to evaluate the degree to which they could be robust alternatives to CNNs.

Chapter 9

Appendix

9.1 Target and flat scenarios: Attack against VGG16 trained on Cifar10

9.1.1 Target scenario

Table 9.1: Target scenario.– For $i \neq j$, the element at the intersection of the i^{th} row and j^{th} column is $\Big(10^3 KL\big(p_{L_2}(c_i \to c_j)||p_{SSIM}(c_i \to c_j)\big), 10^3 KL\big(p_{SSIM}(c_i \to c_j)||p_{L_2}(c_i \to c_j)\big)\Big)$, where $KL\big(p_{L_2}(c_i \to c_j)||p_{SSIM}(c_i \to c_j)\big)$ is the Kullback-Leibler divergence computed between the L_2 and the $SSIM$ probability densities of the normalisation of the histograms representing the changes in pixel intensities through the $c_i \to c_j$ evolution of the ancestor $\mathcal{A}_i$ on i^{th} diagonal position in Figure 9.1 (and Figure 9.2). Mutatis mutandis $KL\big(p_{SSIM}(c_i \to c_j)||p_{L_2}(c_i \to c_j)\big)$.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1		(54,68)	(25,38)	(77,58)	(74,70)	(63,80)	(67,75)	(51,40)	(57,75)	(41,33)
c_2	(82,84)		(66,69)	(81,56)	(62,71)	(109,93)	(62,58)	(56,110)	(80,123)	(74,63)
c_3	(104,92)	(68,60)		(66,73)	(76,55)	(56,64)	(47,22)	(80,81)	(116,67)	(39,56)
c_4	(5,60)	(67,74)	(77,75)		(54,78)	(66,58)	(77,68)	(53,29)	(44,74)	(63,75)
c_5	(36,80)	(126,87)	(73,74)	(62,101)		(95,93)	(59,57)	(92,69)	(59,42)	(82,76)
c_6	(116,79)	(63,104)	(49,39)	(80,80)	(66,77)		(60,43)	(68,93)	(145,135)	(69,63)
c_7	(89,96)	(53,72)	(66,70)	(49,63)	(63,61)	(77,87)		(57,51)	(90,67)	(55,71)
c_8	(84,98)	(155,157)	(43,75)	(62,51)	(74,50)	(40,46)	(74,52)		(41,49)	(93,92)
c_9	(57,77)	(48,53)	(48,39)	(57,76)	(69,54)	(78,53)	(183,192)	(57,75)		(156,121)
c_{10}	(68,70)	(81,77)	(75,98)	(53,68)	(81,60)	(79,75)	(96,103)	(90,81)	(56,60)	

Table 9.2: Target scenario.– The pair of integers at the intersection of the i^{th} row and j^{th} column (for $i \neq j$) represents the number of generations necessary to create the adversarial image with in the evolution $c_i \to c_j$, as specified in Figure 9.1 with L_2 (left-hand side of the pair) and in Figure 9.2 with SSIM (right-hand side of the pair).

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
c_1		(52,55)	(5,5)	(24,23)	(83,79)	(45,44)	(41,44)	(29,31)	(51,53)	(12,13)
c_2	(59,46)		(37,40)	(27,29)	(53,61)	(39,38)	(66,65)	(38,39)	(49,46)	(29,30)
c_3	(56,48)	(60,59)		(47,49)	(58,56)	(52,47)	(84,87)	(34,31)	(77,82)	(38,37)
c_4	(36,41)	(42,47)	(34,32)		(26,24)	(26,25)	(24,21)	(11,11)	(31,34)	(36,36)
c_5	(105,98)	(91,105)	(118,124)	(30,31)		(39,34)	(24,24)	(50,48)	(55,55)	(62,61)
c_6	(54,58)	(39,42)	(45,41)	(35,36)	(11,11)		(56,53)	(31,36)	(26,23)	(41,41)
c_7	(52,54)	(56,53)	(47,43)	(30,26)	(40,38)	(51,54)		(44,42)	(59,53)	(65,62)
c_8	(34,33)	(21,20)	(21,20)	(27,26)	(18,18)	(22,19)	(26,26)		(29,30)	(26,26)
c_9	(27,28)	(33,37)	(16,17)	(72,62)	(39,36)	(96,91)	(82,60)	(43,52)		(69,95)
c_{10}	(14,16)	(54,50)	(32,31)	(67,62)	(30,34)	(55,54)	(57,45)	(27,24)	(24,26)	

9.1.2 Flat scenario

Table 9.3: Flat scenario.– For $1 \leq i \leq 10$, the element in i^{th} position in the 2^{nd} row is $\Big(10^3 KL\big(p_{L_2}(c_i)||p_{SSIM}(c_i)\big), 10^3 KL\big(p_{SSIM}(c_i)||p_{L_2}(c_i)\big)\Big)$, where $KL\big(p_{L_2}(c_i)||p_{SSIM}(c_i)\big)$ is the Kullback-Leibler divergence computed between the L_2 and the $SSIM$ probability densities of the normalisation of the histograms representing the changes in pixel intensities through the $c_i \to$ flat evolution of the ancestor $\mathcal{A}_i$ on i^{th} position on the first row in Figure 9.3. Mutatis mutandis $KL\big(p_{SSIM}(c_i)||p_{L_2}(c_i)\big)$.

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
$(112, 110)$	$(56, 51)$	$(77, 60)$	$(43, 53)$	$(86, 121)$	$(67, 67)$	$(58, 62)$	$(29, 37)$	$(57, 93)$	$(42, 46)$

Table 9.4: Flat scenario.– The pair of integers on the 2^{nd} row represents the number of generations necessary to create the adversarial image in the evolution $c_i \to c_j$, as specified in Figure 9.3 with L_2 (left-hand side) and SSIM (right-hand side).

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
(298,152)	(245,247)	(288,276)	(178,183)	(552,421)	(238,233)	(212,212)	(142,142)	(434,380)	(274,280)

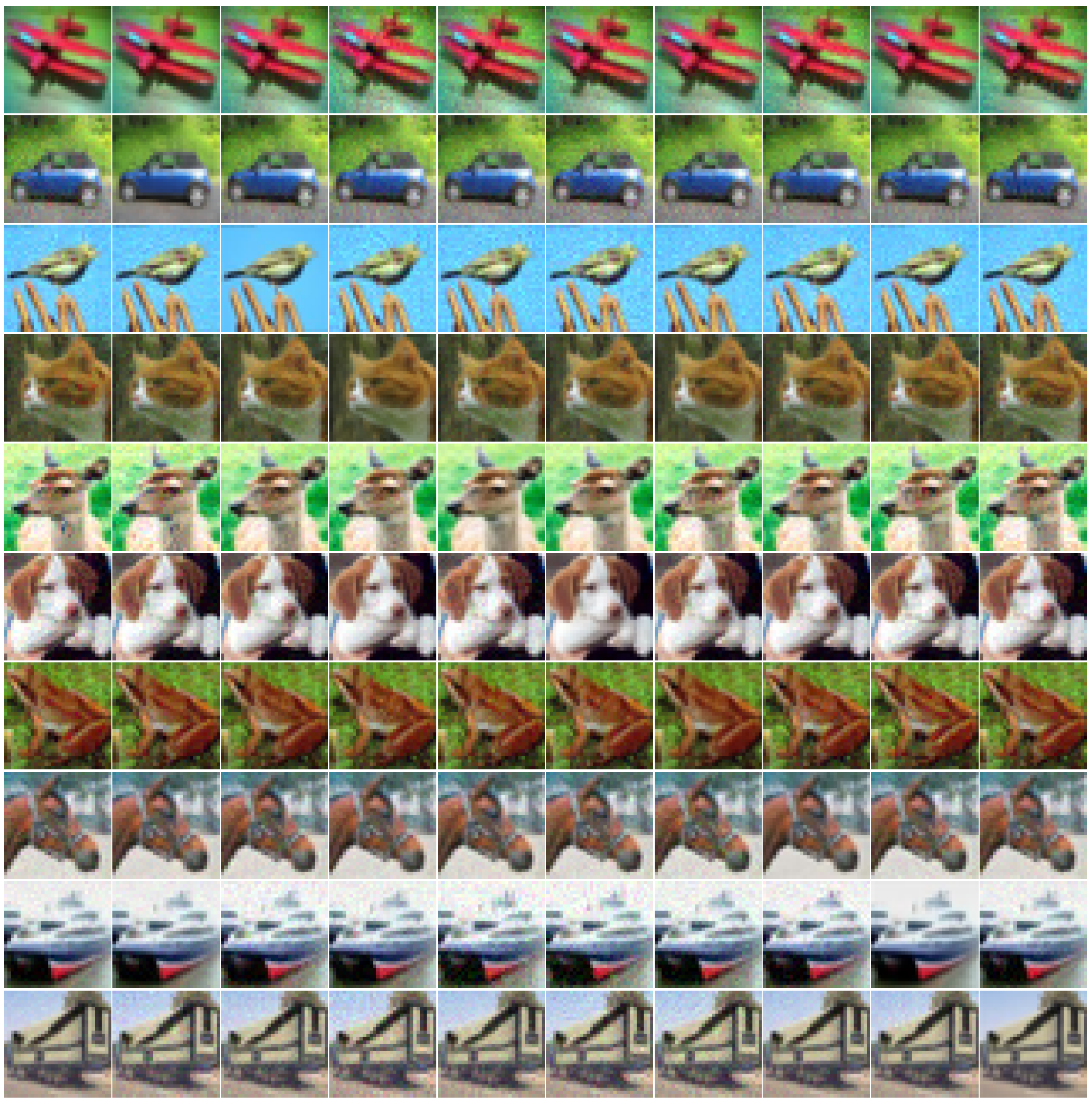


Figure 9.1: Target scenario, case L_2.– Pictures on the diagonal are the ancestors $\mathcal{A}_i$ belonging to the category $c_{\mathcal{A}_i} = c_i$, for $1 \leq i \leq 10$. On each row $1 \leq i \leq 10$, the picture on the j^{th} column, with $j \neq i$, is the descendant picture $\mathcal{D}_{ij}$, obtained by applying $\text{EA}^{\text{target}}_{L_2}$ to $\mathcal{A}_i$, that VGG16 classifies as belonging to c_j.

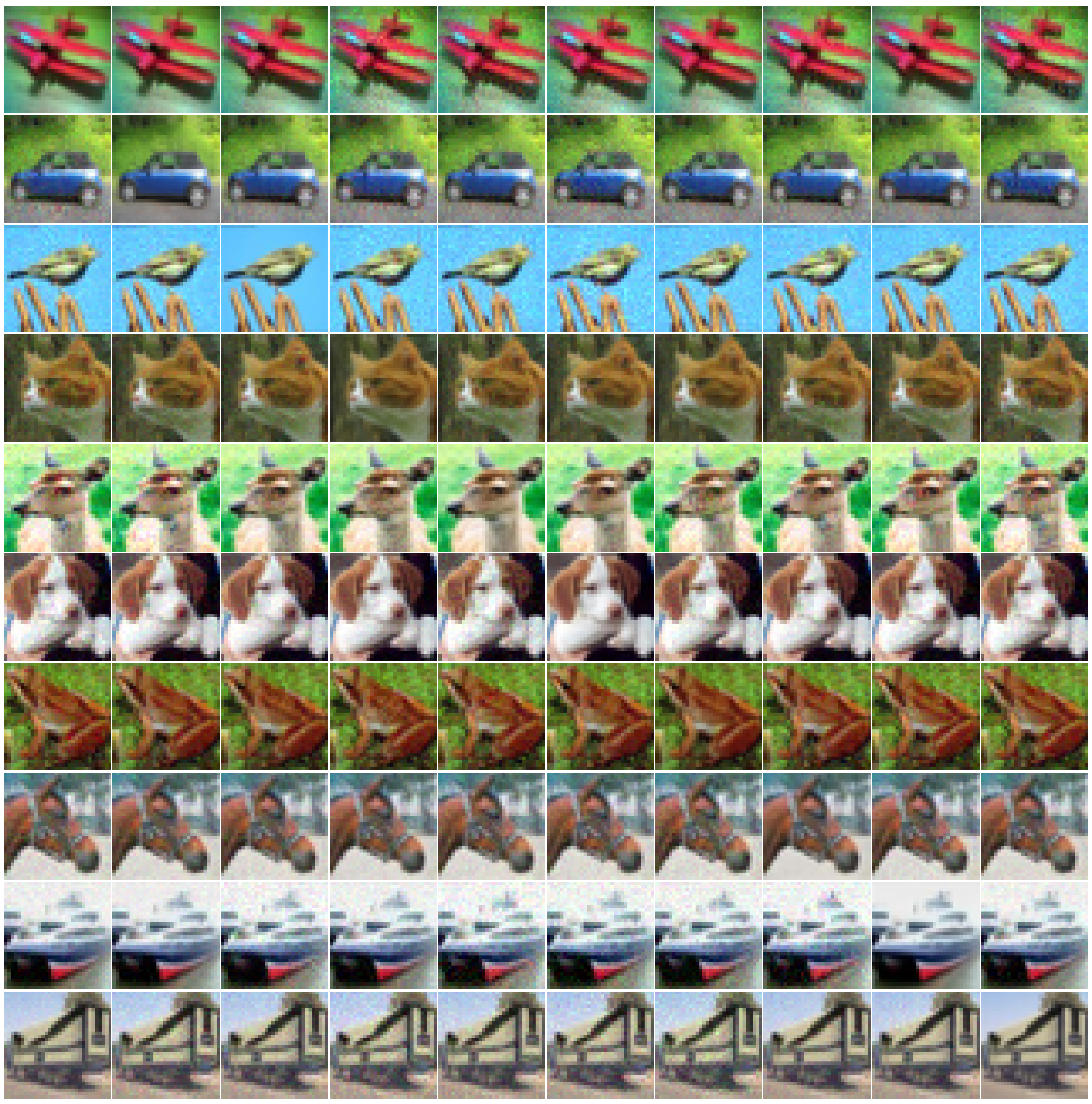


Figure 9.2: Target scenario, case $SSIM$.– Pictures on the diagonal are the ancestors $\mathcal{A}_i$ belonging to the category $c_{\mathcal{A}_i} = c_i$, for $1 \leq i \leq 10$. On each row $1 \leq i \leq 10$, the picture on the j^{th} column, with $j \neq i$, is the descendant picture $\mathcal{D}_{ij}$, obtained by applying $\text{EA}^{\text{target}}_{SSIM}$ to $\mathcal{A}_i$, that VGG16 classifies as belonging to c_j.

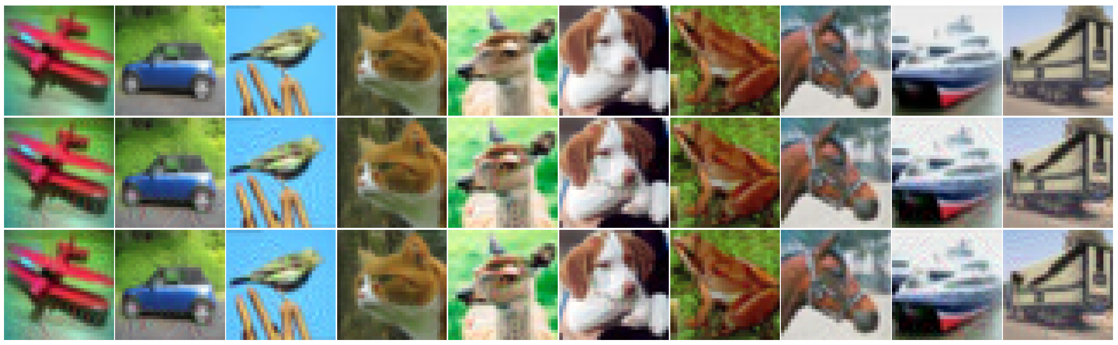


Figure 9.3: Flat scenario.– Pictures on the 1st row are the ancestors $\mathcal{A}_i$ belonging to the category $c_{\mathcal{A}_i} = c_i$, for $1 \leq i \leq 10$ (they are the same as those on the diagonals of Figures 9.1 and 9.2). For $1 \leq i \leq 10$, the picture in i^{th} position on the 2nd row is the adversarial descendant picture obtained by applying $\text{EA}^{\text{flat}}_{L2}$ to $\mathcal{A}_i$, and that VGG16 is unable to classify with certainty. Mutatis mutandis 3rd row with $\text{EA}^{\text{flat}}_{SSIM}$.

9.2 Target scenario: attack against 10 CNNs trained on ImageNet

9.2.1 Ancestor images

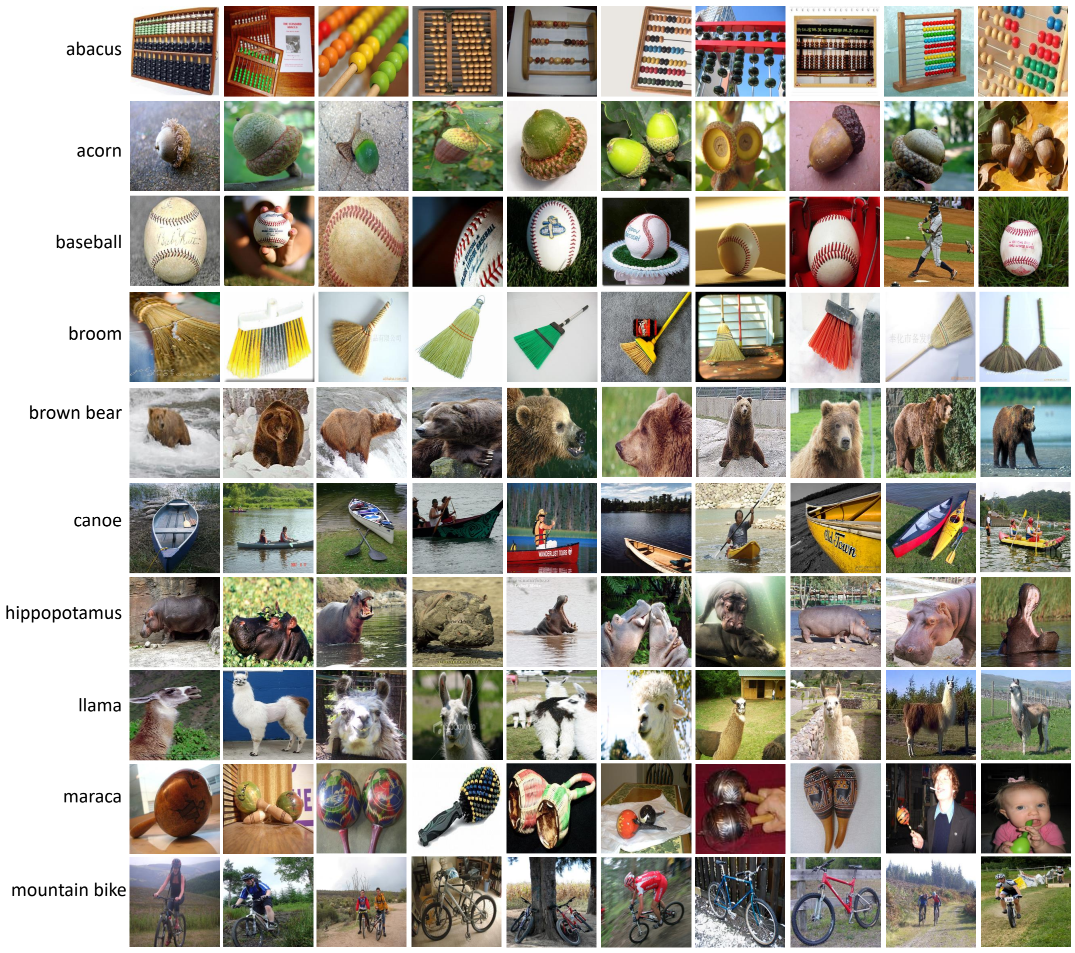


Figure 9.4: The 100 ancestor images $\mathcal{A}_q^p$ used in the experiments. $\mathcal{A}_q^p$ pictured in the p^{th} column and q^{th} row ($1 \leq q, p \leq 10$) is randomly chosen from the ImageNet validation set of the ancestor category c_{a_q} specified on the left of the q^{th} row.

CNNs	p	abacus	acorn	baseball	broom	brown bear	canoe	hippopotamus	llama	maraca	mountain bike
	1	1.000	0.981	0.997	0.999	0.953	0.992	0.999	0.997	0.607	0.942
	2	1.000	0.997	0.989	1.000	0.994	0.909	0.998	0.987	0.883	0.987
	3	0.998	0.845	1.000	1.000	0.996	0.836	0.987	0.997	1.000	0.891
	4	0.996	0.997	1.000	1.000	0.997	0.620	0.239	0.984	0.648	0.619
$\mathcal{C}_1$	5	1.000	0.999	1.000	0.998	0.955	0.811	1.000	1.000	0.145	0.986
DenseNet121	6	1.000	1.000	0.957	0.998	1.000	0.990	0.997	0.916	0.692	0.999
	7	0.998	0.999	0.999	0.973	0.977	0.525	0.985	0.974	0.756	0.940
	8	1.000	0.999	0.993	0.993	0.995	0.913	1.000	1.000	0.999	0.962
	9	1.000	0.998	0.981	1.000	0.997	0.820	0.999	1.000	0.999	0.992
	10	1.000	0.996	1.000	0.999	0.995	0.923	0.999	0.886	0.572	0.870
	1	0.998	0.978	1.000	0.999	0.997	0.997	0.997	0.999	0.952	0.873

Table 9.5 continued from previous page

CNNs	p	abacus	acorn	baseball	broom	brown bear	canoe	hippopotamus	llama	maraca	mountain bike
$\mathcal{C}_2$ DenseNet169	2	1.000	0.999	0.998	0.992	0.782	0.764	0.998	0.999	0.995	0.861
	3	1.000	0.998	0.999	1.000	0.999	0.880	1.000	0.994	1.000	0.977
	4	0.990	0.996	1.000	1.000	1.000	0.549	0.553	0.981	0.900	0.973
	5	1.000	1.000	1.000	0.994	1.000	0.915	1.000	0.994	0.530	0.997
	6	1.000	1.000	0.998	1.000	1.000	0.997	0.995	0.975	0.091	0.991
	7	1.000	1.000	1.000	1.000	0.998	0.827	0.996	1.000	0.857	0.945
	8	1.000	1.000	0.998	0.998	0.999	0.951	0.999	1.000	1.000	0.975
	9	1.000	1.000	0.943	1.000	0.999	0.905	1.000	1.000	0.993	0.964
	10	1.000	1.000	0.999	1.000	0.997	0.952	0.999	0.998	**0.258**	**0.478**
$\mathcal{C}_3$ DenseNet201	1	1.000	0.975	0.998	1.000	0.992	0.990	0.998	0.996	**0.323**	0.986
	2	1.000	1.000	0.984	1.000	0.884	0.957	0.996	0.996	0.993	0.997
	3	0.987	0.950	0.998	1.000	0.998	0.669	0.999	0.994	1.000	0.886
	4	0.886	0.994	1.000	1.000	0.998	0.822	0.870	1.000	0.878	0.947
	5	1.000	1.000	0.999	0.983	0.980	0.586	1.000	0.998	0.141	0.980
	6	1.000	1.000	0.995	1.000	1.000	0.994	0.999	0.724	0.693	0.996
	7	1.000	1.000	1.000	1.000	0.993	0.865	0.997	0.970	0.876	0.917
	8	1.000	1.000	0.993	1.000	0.874	0.978	0.990	0.999	0.997	0.993
	9	1.000	0.999	0.877	1.000	0.984	0.995	1.000	0.999	0.987	0.988
	10	0.996	1.000	0.998	0.999	0.978	0.984	0.987	0.963	**0.253**	0.983
$\mathcal{C}_4$ MobileNet	1	0.999	0.994	1.000	0.995	0.999	0.999	0.998	0.954	**0.329**	0.938
	2	0.994	0.936	0.993	0.998	0.972	0.620	0.991	0.914	0.946	0.631
	3	1.000	0.979	1.000	1.000	1.000	0.825	1.000	0.999	1.000	0.947
	4	1.000	0.999	1.000	0.998	0.999	0.826	0.758	1.000	0.762	0.966
	5	1.000	0.984	1.000	0.955	0.997	0.944	1.000	0.944	0.906	0.978
	6	1.000	1.000	1.000	0.992	1.000	0.961	0.992	0.589	0.645	0.862
	7	0.999	0.998	1.000	0.989	0.996	0.812	0.264	1.000	0.999	0.729
	8	1.000	1.000	1.000	0.632	0.997	0.952	0.997	1.000	0.809	0.998
	9	1.000	0.991	0.915	0.997	0.997	0.989	1.000	1.000	0.525	0.988
	10	1.000	1.000	1.000	1.000	0.982	0.930	1.000	0.988	0.618	0.706
$\mathcal{C}_5$ NASNet Mobile	1	0.932	0.945	0.885	0.948	0.902	0.925	0.914	0.945	0.288	0.869
	2	0.947	0.946	0.905	0.892	0.961	0.932	0.829	0.951	0.957	0.902
	3	0.903	0.884	0.858	0.978	0.948	**0.059**	0.926	0.754	0.911	0.923
	4	0.844	0.929	0.895	0.961	0.910	**0.358**	0.656	0.928	0.994	0.667
	5	0.943	0.930	0.886	0.914	0.936	0.586	0.921	0.976	**0.091**	0.972
	6	0.973	0.945	0.949	0.972	0.925	0.792	0.846	0.936	**0.040**	0.854
	7	0.983	0.897	0.842	0.944	0.906	0.869	0.893	0.941	0.803	0.781
	8	0.962	0.950	0.870	0.908	0.887	0.864	0.824	0.965	0.930	0.904
	9	0.975	0.904	0.691	0.949	0.925	0.783	0.925	0.949	0.965	0.957
	10	0.925	0.957	0.851	0.955	0.809	0.860	0.941	0.929	0.397	**0.028**
$\mathcal{C}_6$ ResNet50	1	0.937	0.795	0.998	0.841	1.000	0.998	0.999	0.999	0.801	0.986
	2	0.411	1.000	1.000	1.000	0.999	0.991	1.000	0.998	0.850	0.995
	3	1.000	0.901	1.000	1.000	1.000	0.778	1.000	1.000	1.000	0.993
	4	1.000	0.993	1.000	1.000	0.999	0.897	0.881	0.999	0.646	0.929
	5	1.000	1.000	1.000	0.969	0.996	0.945	1.000	0.381	**0.001**	0.995
	6	0.999	1.000	1.000	0.999	0.999	0.995	1.000	0.771	0.211	0.941
	7	1.000	1.000	1.000	0.988	0.996	0.743	1.000	1.000	0.993	0.892
	8	1.000	0.998	0.998	0.999	0.997	0.993	0.962	1.000	0.999	0.987
	9	1.000	1.000	0.695	1.000	0.999	0.971	1.000	1.000	0.998	0.999
	10	1.000	0.999	1.000	0.999	0.959	0.994	0.970	0.723	0.003	0.965
	1	0.998	0.982	0.999	0.995	0.985	0.999	1.000	1.000	0.984	0.969
	2	1.000	1.000	0.973	1.000	0.998	0.986	1.000	0.988	0.975	0.997
	3	1.000	0.929	1.000	1.000	1.000	0.882	1.000	1.000	1.000	0.895
	4	0.778	0.999	1.000	1.000	0.993	**0.467**	0.680	0.999	0.951	0.970
	5	1.000	1.000	1.000	0.991	0.945	0.835	1.000	0.940	**0.001**	0.990
	6	1.000	1.000	0.994	0.998	0.999	0.996	1.000	0.722	**0.002**	0.998
	7	1.000	1.000	1.000	1.000	0.981	0.961	1.000	1.000	0.753	0.756
	8	1.000	1.000	1.000	0.996	0.910	0.994	0.976	1.000	0.995	0.990

$\mathcal{C}_7$
ResNet101

Table 9.5 continued from previous page

CNNs	p	abacus	acorn	baseball	broom	brown bear	canoe	hippopotamus	llama	maraca	mountain bike
	9	1.000	1.000	0.979	1.000	0.997	0.848	1.000	1.000	0.959	0.980
	10	1.000	0.993	1.000	1.000	0.927	0.975	0.996	0.917	**0.003**	0.984
$\mathcal{C}_8$ ResNet152	1	0.994	0.998	1.000	0.996	0.997	0.987	0.999	0.999	0.954	0.991
	2	0.713	1.000	0.997	1.000	0.996	0.983	1.000	1.000	0.956	0.998
	3	1.000	0.665	1.000	1.000	1.000	**0.205**	0.999	1.000	1.000	0.969
	4	0.998	0.997	1.000	1.000	1.000	**0.347**	0.872	0.972	0.960	0.960
	5	1.000	1.000	1.000	1.000	0.999	0.841	1.000	0.927	**0.067**	0.993
	6	1.000	1.000	1.000	0.994	1.000	0.997	0.999	0.805	0.436	0.986
	7	1.000	1.000	1.000	1.000	0.967	**0.442**	0.995	1.000	0.973	0.860
	8	1.000	1.000	1.000	1.000	0.951	0.965	0.999	1.000	1.000	0.991
	9	1.000	1.000	0.857	1.000	0.978	0.979	0.992	1.000	0.949	0.999
	10	1.000	1.000	1.000	1.000	0.861	0.871	1.000	0.872	**0.161**	0.961
$\mathcal{C}_9$ VGG16	1	1.000	0.392	1.000	**0.272**	0.991	0.990	0.999	0.940	**0.112**	0.862
	2	0.952	0.997	1.000	0.918	0.472	0.918	1.000	0.968	0.683	0.979
	3	0.998	0.688	1.000	1.000	1.000	0.896	1.000	1.000	1.000	0.952
	4	0.996	0.999	1.000	0.993	0.998	0.764	0.214	0.999	**0.259**	0.740
	5	1.000	0.999	1.000	0.913	0.997	0.678	1.000	0.918	**0.090**	0.936
	6	1.000	1.000	0.674	0.972	0.999	0.883	1.000	0.828	**0.027**	0.952
	7	0.999	0.998	0.999	0.999	0.995	0.595	0.935	1.000	**0.018**	0.640
	8	0.987	0.995	1.000	0.844	0.999	0.952	0.999	1.000	0.979	0.973
	9	1.000	0.999	0.896	0.992	0.915	0.382	1.000	1.000	0.918	0.895
	10	1.000	1.000	1.000	0.998	0.964	0.981	1.000	0.998	0.745	0.614
$\mathcal{C}_{10}$ VGG19	1	1.000	0.959	1.000	**0.491**	0.981	0.547	1.000	0.977	0.507	0.909
	2	0.990	0.998	0.999	0.957	0.991	0.812	1.000	0.983	0.514	0.903
	3	1.000	0.767	1.000	0.996	1.000	0.946	1.000	1.000	1.000	0.912
	4	0.995	0.980	1.000	0.994	0.996	0.663	0.241	0.995	**0.079**	0.270
	5	1.000	0.999	1.000	0.617	0.997	**0.267**	1.000	**0.134**	**0.008**	0.934
	6	1.000	1.000	0.998	0.975	0.999	0.779	0.999	0.932	**0.064**	0.957
	7	1.000	0.999	1.000	0.999	0.999	0.586	0.995	1.000	**0.221**	0.422
	8	1.000	1.000	1.000	0.956	0.997	0.846	0.997	1.000	0.994	0.930
	9	1.000	1.000	0.575	0.991	0.988	0.441	1.000	1.000	0.660	0.752
	10	1.000	1.000	1.000	1.000	0.993	0.859	0.999	0.966	0.731	0.862

Table 9.5: For $1 \leq p \leq 10$, the ancestor category c_{a_q}-label values given by the 10 CNNs of the image $\mathcal{A}_q^p$ pictured in Figure 9.4. A label value in red indicates that the category c_{a_q} is not the dominant one.

9.2.2 Adversarial images

X	Success Rates	$\mathcal{C}_1$-DenseNet121	$\mathcal{C}_2$-DenseNet169	$\mathcal{C}_3$-DenseNet201	$\mathcal{C}_4$-MobileNet	$\mathcal{C}_5$-NASNetMobile	$\mathcal{C}_6$-ResNet50	$\mathcal{C}_7$-ResNet101	$\mathcal{C}_8$-ResNet152	$\mathcal{C}_9$-VGG16	$\mathcal{C}_{10}$-VGG19	**Average**
1000	$SR_{\mathcal{C}}^{0.75}$	5	1	2	10	4	2	1	2	2	2	**3.1**
	$SR_{\mathcal{C}}^{\text{ge}}$	10	9	7	23	7	12	4	7	4	2	**8.5**
	$SR_{\mathcal{C}}^{\text{untarg}}$	27	19	18	34	26	31	21	23	23	27	**25.0**
2000	$SR_{\mathcal{C}}^{0.75}$	19	27	19	57	15	26	26	19	15	12	**23.5**
	$SR_{\mathcal{C}}^{\text{ge}}$	45	48	34	71	28	39	35	37	27	29	**39.3**
	$SR_{\mathcal{C}}^{\text{untarg}}$	53	51	41	76	46	56	49	48	64	62	**54.7**
3000	$SR_{\mathcal{C}}^{0.75}$	47	53	40	85	33	50	43	46	35	31	**46.3**
	$SR_{\mathcal{C}}^{\text{ge}}$	70	75	69	89	47	68	62	59	57	51	**64.7**
	$SR_{\mathcal{C}}^{\text{untarg}}$	74	76	73	89	55	82	70	72	84	78	**75.2**
4000	$SR_{\mathcal{C}}^{0.75}$	66	71	64	90	46	73	61	61	58	50	**64.0**
	$SR_{\mathcal{C}}^{\text{ge}}$	83	82	82	97	61	87	80	83	77	72	**80.4**
	$SR_{\mathcal{C}}^{\text{untarg}}$	84	83	85	97	72	90	83	86	89	88	**85.7**
5000	$SR_{\mathcal{C}}^{0.75}$	74	77	74	97	58	82	75	79	75	71	**76.2**
	$SR_{\mathcal{C}}^{\text{ge}}$	87	90	84	97	73	92	88	91	91	88	**88.1**
	$SR_{\mathcal{C}}^{\text{untarg}}$	88	90	87	97	77	93	89	93	93	96	**90.1**
6000	$SR_{\mathcal{C}}^{0.75}$	82	81	80	98	66	89	83	87	86	82	**83.4**
	$SR_{\mathcal{C}}^{\text{ge}}$	92	91	89	98	76	94	92	92	95	94	**91.3**
	$SR_{\mathcal{C}}^{\text{untarg}}$	92	91	91	98	80	96	93	94	97	97	**92.8**
7000	$SR_{\mathcal{C}}^{0.75}$	84	86	84	98	72	93	86	90	93	89	**87.5**
	$SR_{\mathcal{C}}^{\text{ge}}$	93	92	92	98	81	96	92	95	98	97	**93.4**
	$SR_{\mathcal{C}}^{\text{untarg}}$	93	92	93	98	83	97	93	96	99	97	**94.0**
8000	$SR_{\mathcal{C}}^{0.75}$	88	88	85	98	73	94	89	92	97	94	**89.8**
	$SR_{\mathcal{C}}^{\text{ge}}$	95	95	93	99	81	97	92	97	99	99	**94.7**
	$SR_{\mathcal{C}}^{\text{untarg}}$	95	95	94	99	86	98	93	97	99	99	**95.4**
9000	$SR_{\mathcal{C}}^{0.75}$	90	90	86	98	78	95	91	94	98	97	**91.7**
	$SR_{\mathcal{C}}^{\text{ge}}$	95	96	95	99	85	98	96	98	100	100	**96.2**
	$SR_{\mathcal{C}}^{\text{untarg}}$	95	97	95	99	87	98	96	98	100	100	**96.5**
10000	$SR_{\mathcal{C}}^{0.75}$	92	91	86	99	80	96	91	96	99	98	**92.8**
	$SR_{\mathcal{C}}^{\text{ge}}$	95	96	96	99	86	99	98	99	100	100	**96.8**
	$SR_{\mathcal{C}}^{\text{untarg}}$	95	97	96	99	87	99	98	99	100	100	**97.0**

Table 9.6: Success rates of $\text{EA}^{\text{target},\mathcal{C}}$ for increasing values of X while $\tau = 0.75$ for the experiments designed in subsection 4.2.2 and performed in subsection 4.2.3.

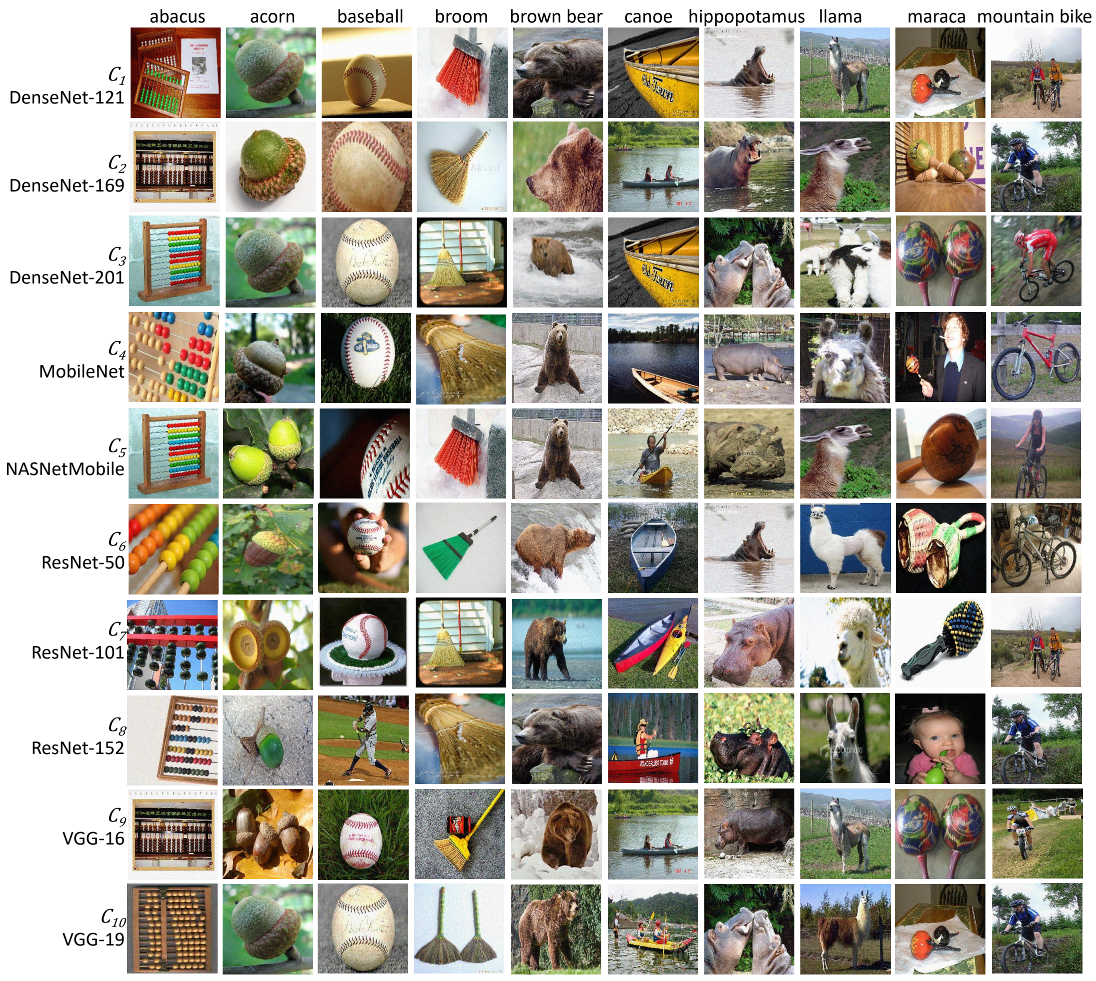


Figure 9.5: Samples of 0.75-*strong adversarial images* generated by $\text{EA}^{\text{target},\mathcal{C}_k}$ for $1 \leq k \leq 10$.

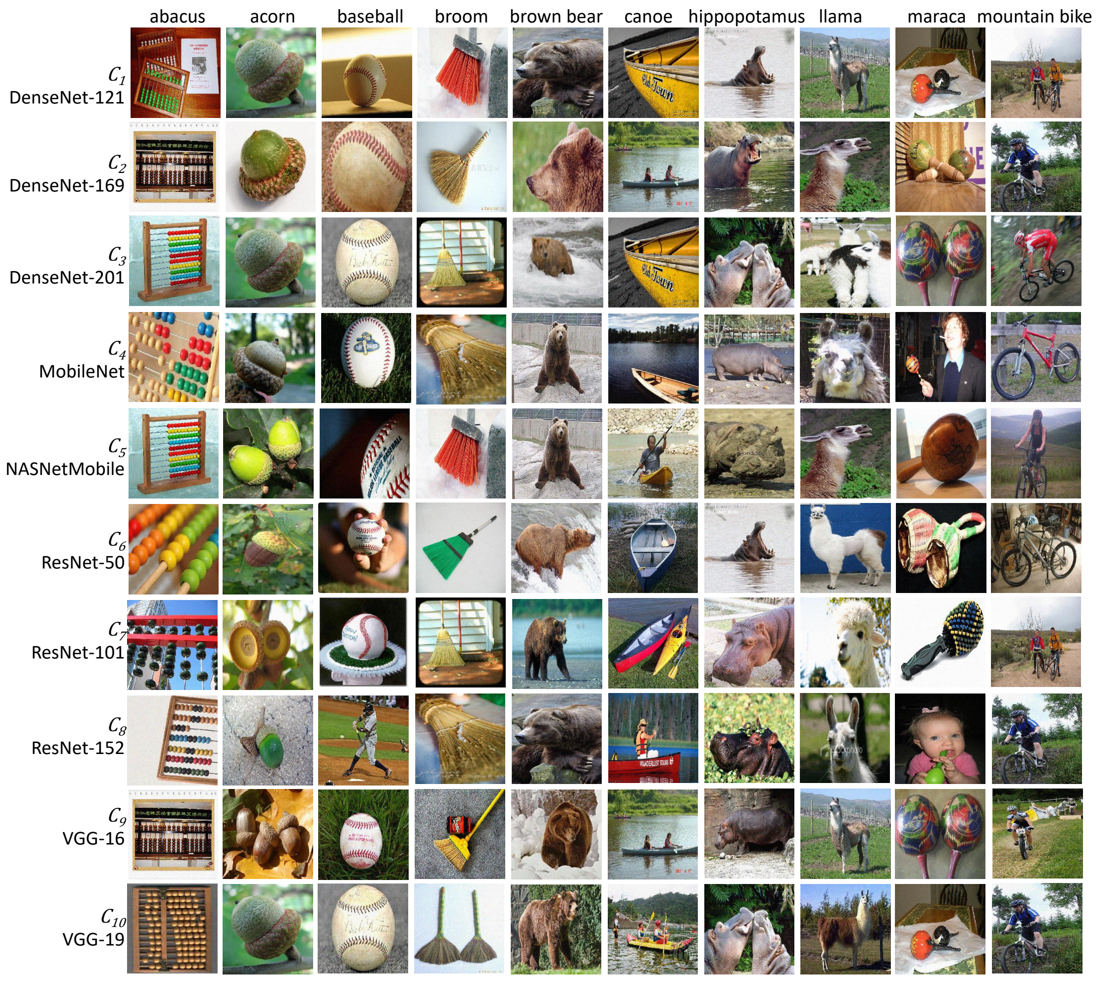


Figure 9.6: Samples of *good enough adversarial images* generated by $\text{EA}^{\text{target},\mathcal{C}_k}$ for $1 \leq k \leq 10$.

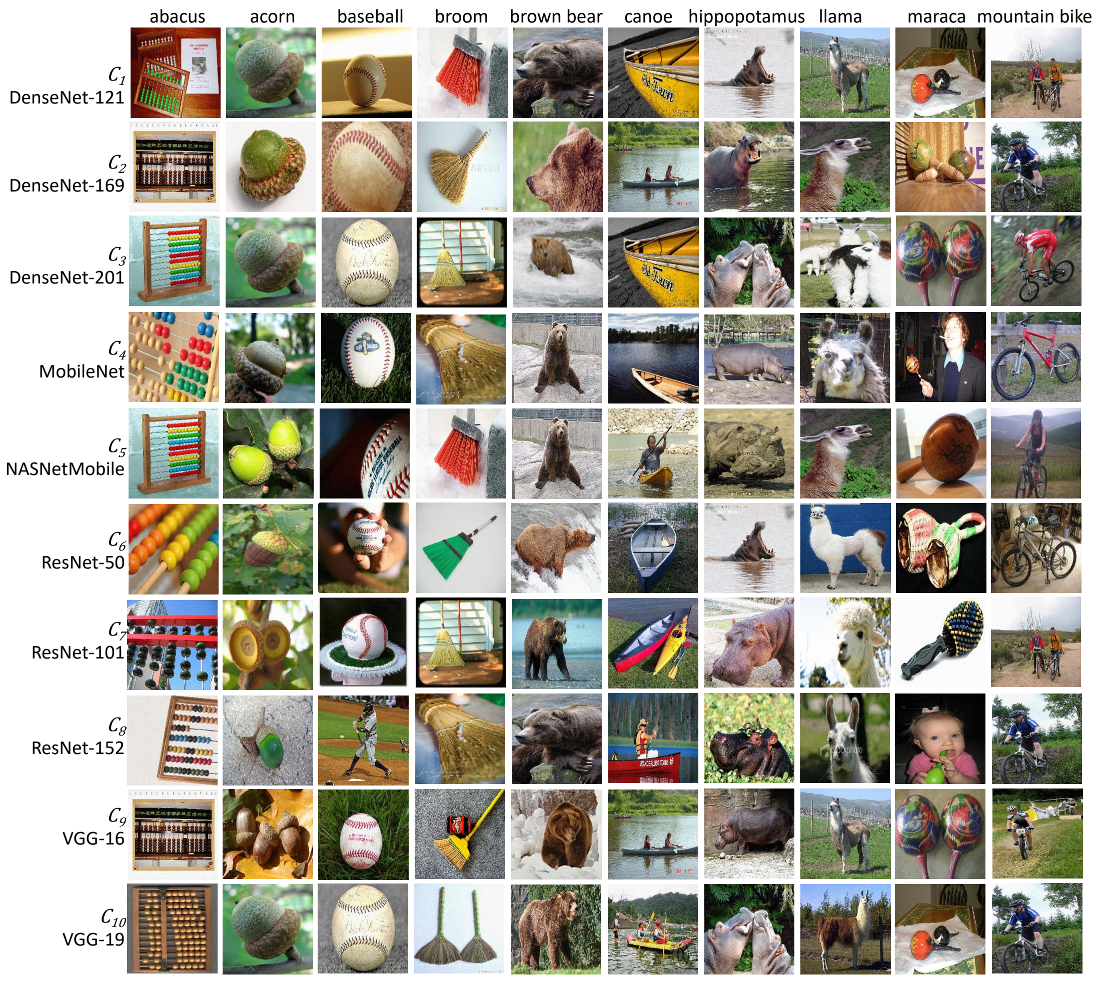
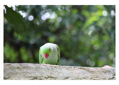
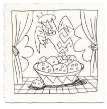


Figure 9.7: Samples of *untargeted adversarial images* generated by $\text{EA}^{\text{target},\mathcal{C}_k}$ for $1 \leq k \leq 10$.

9.3 Attack on High Resolution Images: Method and Performance

9.3.1 A

Table 9.7: For $1 \leq a \leq 10$, the image $\mathcal{A}_a^{\text{hr}}$ classified by each CNN in the category c_a (interpolation = ”lanczos”).

a	1	2	3	4	5	6	7	8	9	10
$\mathcal{A}_a^{hr}$										
$w \times h$	910×604	960×640	910×607	2462×2913	910×607	641×600	1280×800	1280×800	1954×2011	1740×1710
$\mathcal{C}_1$	cheetah 0.872	Eskimo_dog 0.691	koala 0.987	lampshade 0.512	white_stork 0.484	screen 0.659	fountain 0.223	sports_car 0.840	book_jacket 0.237	buckle 0.249
$\mathcal{C}_2$	cheetah 0.986	Eskimo_dog 0.822	koala 0.997	lampshade 0.673	Granny_Smith 0.213	screen 0.818	comic_book 0.322	sports_car 0.587	rubber_eraser 0.327	book_jacket 0.168
$\mathcal{C}_3$	cheetah 0.976	Eskimo_dog 0.737	koala 0.997	table_lamp 0.614	toucan 0.194	screen 0.724	comic_book 0.453	sports_car 0.808	handkerchief 0.194	book_jacket 0.237
$\mathcal{C}_4$	cheetah 0.816	Eskimo_dog 0.516	koala 0.999	table_lamp 0.884	flamingo 0.497	screen 0.706	totem_pole 0.161	sports_car 0.740	tray 0.297	book_jacket 0.439
$\mathcal{C}_5$	cheetah 0.923	Eskimo_dog 0.613	koala 0.902	table_lamp 0.488	spoonbill 0.209	screen 0.804	fountain 0.951	sports_car 0.711	book_jacket 0.372	manhole_cover 0.116
$\mathcal{C}_6$	cheetah 0.972	Eskimo_dog 0.704	koala 0.994	lampshade 0.507	macaw 0.433	screen 0.697	gasmask 0.280	sports_car 0.813	pillow 0.163	coffee_mug 0.175
$\mathcal{C}_7$	cheetah 0.948	Eskimo_dog 0.629	koala 0.555	lampshade 0.686	white_stork 0.224	screen 0.904	fountain 0.204	sports_car 0.470	book_jacket 0.378	buckle 0.378
$\mathcal{C}_8$	cheetah 0.899	Eskimo_dog 0.760	koala 0.979	table_lamp 0.641	toucan 0.163	screen 0.699	fountain 0.702	sports_car 0.546	envelope 0.243	matchstick 0.569
$\mathcal{C}_9$	cheetah 0.953	Eskimo_dog 0.343	koala 0.997	lampshade 0.536	toucan 0.455	screen 0.706	comic_book 0.492	sports_car 0.480	binder 0.283	coffee_mug 0.084
$\mathcal{C}_{10}$	cheetah 0.867	Eskimo_dog 0.412	koala 0.964	table_lamp 0.588	lorikeet 0.145	screen 0.665	comic_book 0.553	sports_car 0.649	lighter 0.229	prayer_rug 0.158
c_t	poncho	goblet	weimaraner	weevil	wombat	swing	altar	beagle	triceratops	hamper

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.123	0.130	0.118	0.115
Bicubic	0.064	0.061	0.073	0.057
Bilinear	0.024	0.015	0.011	0.023
Lanczos	0.124	0.123	0.132	0.121

(a) $\mathcal{C}_1$

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.092	0.084	0.075	0.093
Bicubic	0.074	0.053	0.055	0.061
Bilinear	0.058	0.048	0.058	0.050
Lanczos	0.050	0.041	0.043	0.049

(b) $\mathcal{C}_2$

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.095	0.088	0.096	0.086
Bicubic	0.005	-0.001	0.011	-0.009
Bilinear	0.024	0.022	0.039	0.015
Lanczos	0.009	0.002	0.007	-0.003

(c) $\mathcal{C}_3$

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.088	0.106	0.123	0.109
Bicubic	0.047	0.064	0.061	0.057
Bilinear	0.052	0.053	0.071	0.055
Lanczos	0.087	0.095	0.084	0.091

(d) $\mathcal{C}_4$

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.142	0.138	0.156	0.141
Bicubic	0.024	0.005	0.034	-0.013
Bilinear	0.124	0.112	0.130	0.107
Lanczos	0.017	-0.005	0.034	-0.015

(e) $\mathcal{C}_5$

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.089	0.087	0.105	0.078
Bicubic	0.056	0.046	0.055	0.048
Bilinear	0.037	0.033	0.042	0.035
Lanczos	0.113	0.110	0.122	0.102

(f) $\mathcal{C}_6$

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.058	0.071	0.081	0.058
Bicubic	0.057	0.038	0.059	0.024
Bilinear	0.087	0.082	0.104	0.076
Lanczos	0.098	0.090	0.115	0.082

(g) $\mathcal{C}_7$

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.134	0.131	0.120	0.126
Bicubic	0.057	0.047	0.057	0.043
Bilinear	0.067	0.060	0.076	0.058
Lanczos	0.061	0.050	0.067	0.053

(h) $\mathcal{C}_8$

ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.058	0.074	0.077	0.084
Bicubic	0.051	0.059	0.076	0.052
Bilinear	0.056	0.066	0.094	0.056
Lanczos	0.048	0.057	0.070	0.051

(i) $\mathcal{C}_9$

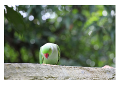
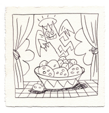
ρ \ λ	Nearest Neighbor	Bicubic	Bilinear	Lanczos
Nearest Neighbor	0.020	0.024	0.048	0.030
Bicubic	0.078	0.082	0.108	0.070
Bilinear	0.099	0.102	0.134	0.090
Lanczos	0.070	0.065	0.092	0.051

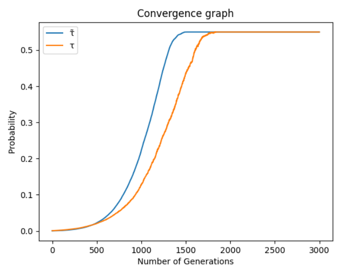
(j) $\mathcal{C}_{10}$

Figure 9.8: The heat-maps of the loss function $\mathcal{L}^{\mathcal{C}}(\mathcal{A}_a^{\text{hr}}) = \tilde{\tau}_a - \tau_a$ for each CNN.

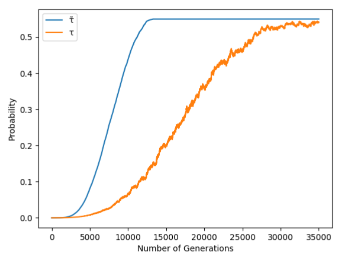
9.3.2 B

Table 9.8: Success rates (SR) of $\text{EA}^{\text{target},\mathcal{C}}$ for each CNN over 10 independent runs, for $\tau = 0.55$ and $X = 35,000$.

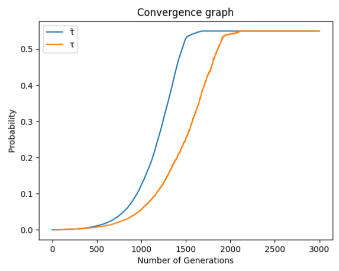
a	1	2	3	4	5	6	7	8	9	10	
$\mathcal{A}_a^{hr}$											
c_t	poncho	goblet	weimaraner	weevil	wombat	swing	altar	beagle	triceratops	hamper	**SR(%)**
$\mathcal{C}_1$	0	10	10	10	10	10	10	9	0	10	**79**
$\mathcal{C}_2$	10	10	10	10	10	10	10	9	10	10	**99**
$\mathcal{C}_3$	0	10	10	10	10	10	10	10	6	10	**86**
$\mathcal{C}_4$	10	10	10	10	10	10	10	10	5	10	**95**
$\mathcal{C}_5$	2	2	10	7	10	10	10	2	0	10	**63**
$\mathcal{C}_6$	10	10	10	10	10	10	10	10	0	10	**90**
$\mathcal{C}_7$	10	10	10	10	10	10	10	10	2	10	**92**
$\mathcal{C}_8$	9	10	10	10	10	10	10	10	8	10	**97**
$\mathcal{C}_9$	10	10	10	10	10	10	10	10	10	10	**100**
$\mathcal{C}_{10}$	10	10	10	10	10	10	10	10	9	10	**99**
Avg.	**8.9**	**9.2**	**10**	**9.7**	**10**	**10**	**10**	**9**	**7.1**	**10**	**90**



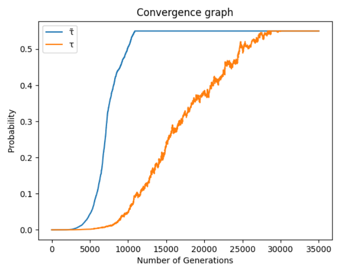
(a) $\mathcal{C}_1 - min(\mathcal{L}_{max})$:$\mathcal{A}_7$



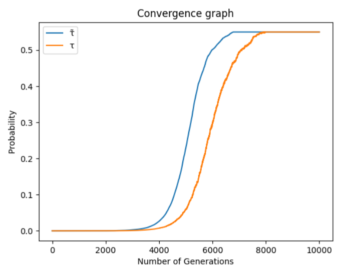
(b) $\mathcal{C}_1 - max(\mathcal{L}_{max})$:$\mathcal{A}_8$



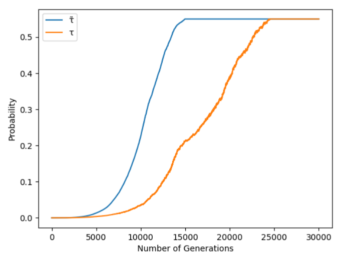
(c) $\mathcal{C}_2 - min(\mathcal{L}_{max})$:$\mathcal{A}_7$



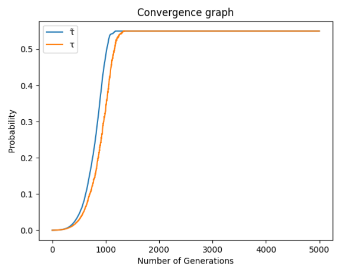
(d) $\mathcal{C}_2 - max(\mathcal{L}_{max})$:$\mathcal{A}_1$



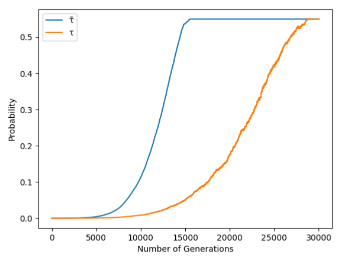
(e) $\mathcal{C}_3 - min(\mathcal{L}_{max})$:$\mathcal{A}_6$



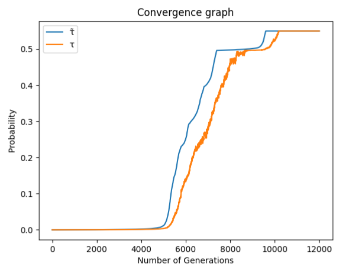
(f) $\mathcal{C}_3 - max(\mathcal{L}_{max})$:$\mathcal{A}_9$



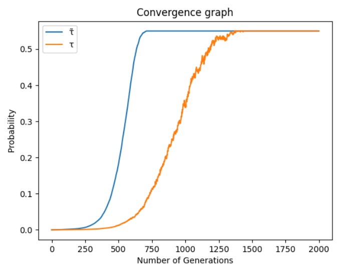
(g) $\mathcal{C}_4 - min(\mathcal{L}_{max})$:$\mathcal{A}_2$



(h) $\mathcal{C}_4 - max(\mathcal{L}_{max})$:$\mathcal{A}_9$

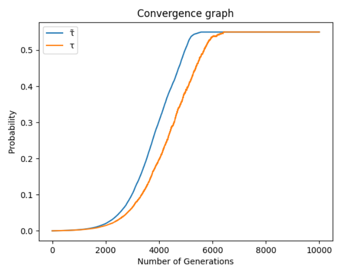


(i) $\mathcal{C}_5 - min(\mathcal{L}_{max})$:$\mathcal{A}_6$

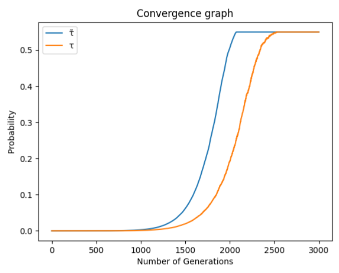


(j) $\mathcal{C}_5 - max(\mathcal{L}_{max})$:$\mathcal{A}_7$

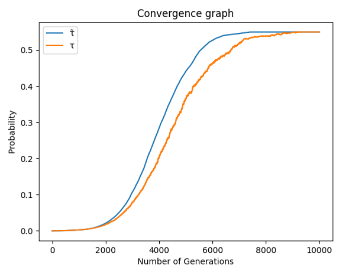
Figure 9.9: Convergence characteristics of the EA based on τ_t and $\tilde{\tau}_t$ for each CNN. Only the pairs with the smallest and largest $\mathcal{L}_{max}$ values are shown in the figures. (Group 1)



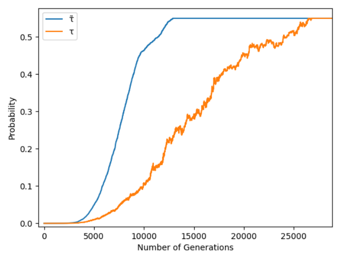
(a) $\mathcal{C}_6 - min(\mathcal{L}_{max})$:$\mathcal{A}_2$



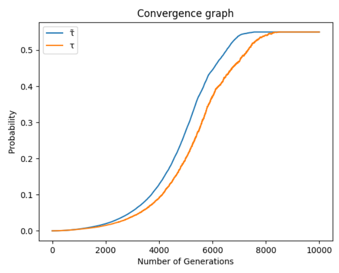
(b) $\mathcal{C}_6 - max(\mathcal{L}_{max})$:$\mathcal{A}_3$



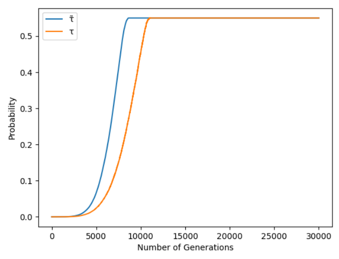
(c) $\mathcal{C}_7 - min(\mathcal{L}_{max})$:$\mathcal{A}_2$



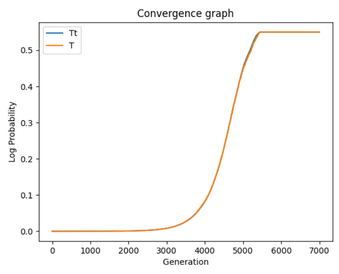
(d) $\mathcal{C}_7 - max(\mathcal{L}_{max})$:$\mathcal{A}_1$



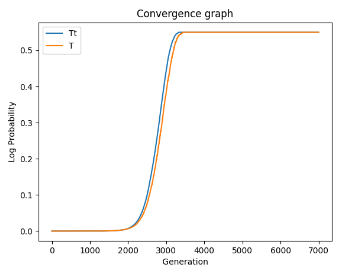
(e) $\mathcal{C}_8 - min(\mathcal{L}_{max})$:$\mathcal{A}_2$



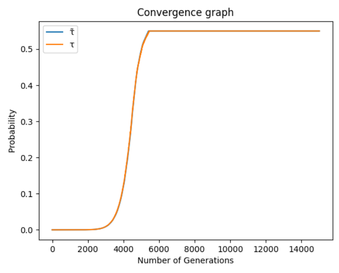
(f) $\mathcal{C}_8 - max(\mathcal{L}_{max})$:$\mathcal{A}_{10}$



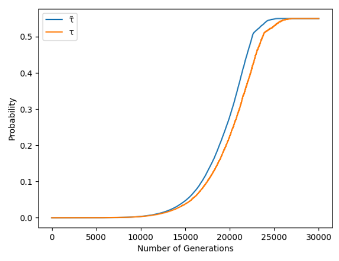
(g) $\mathcal{C}_9 - min(\mathcal{L}_{max})$:$\mathcal{A}_6$



(h) $\mathcal{C}_9 - max(\mathcal{L}_{max})$:$\mathcal{A}_4$



(i) $\mathcal{C}_{10} - min(\mathcal{L}_{max})$:$\mathcal{A}_6$



(j) $\mathcal{C}_{10} - max(\mathcal{L}_{max})$:$\mathcal{A}_9$

Figure 9.10: Convergence characteristics of the EA based on τ_t and $\tilde{\tau}_t$ for each CNN. Only the pairs with the smallest and largest $\mathcal{L}_{max}$ values are shown in the figures. (Group 2)

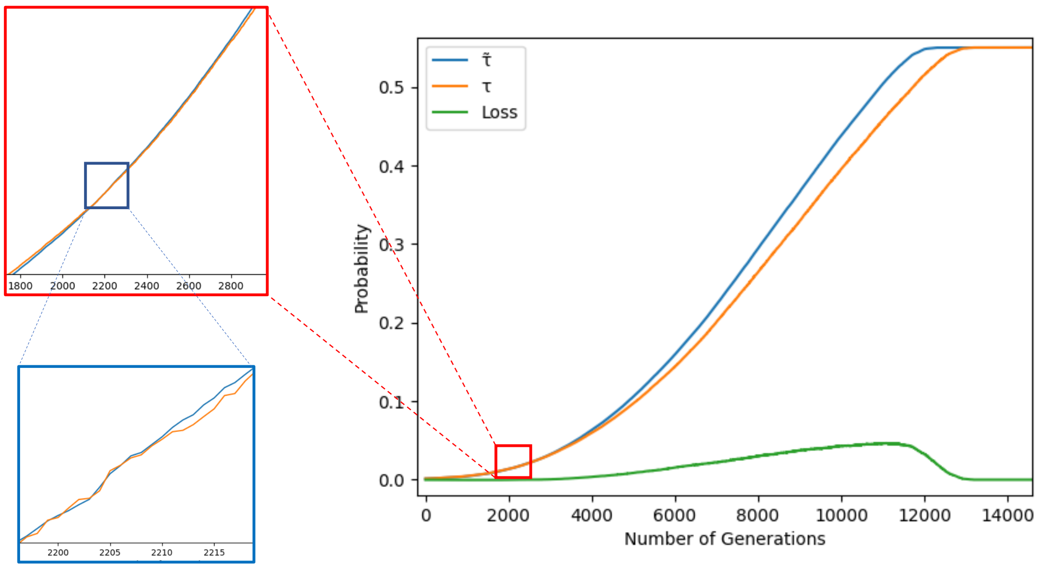


Figure 9.11: The average convergence characteristics of $\text{EA}^{\text{target},\mathcal{C}}$ for $\mathcal{C}$ = VGG16 aiming at generating a high-resolution adversarial image by directly evolving $\mathcal{A}_{10}^{hr}$. The horizontal axis of the graph is the number of generations, and the vertical axis is the target probability τ_t, $\tilde{\tau}_t$ and the loss $\mathcal{L} = \tilde{\tau}_t - \tau_t$. The zoomed-in section of the graph shows when the $\tilde{\tau}_t$ becomes bigger than the τ_t ($\approx 2209^{th} generation$). As the loss $\mathcal{L}$ curve shows, the distance between $\tilde{\tau}_t$ and τ_t increases over the generations.

9.3.3 C

Table 9.9: Direct attack results of $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$ for the easiest (c_a, c_t) pairs after 48 hours of execution of the algorithm. In the last column $c_t\_ratio = c_t\_end/c_t\_start$

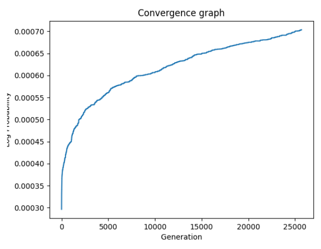
(c_a, c_t)		(toucan, wombat)					(comic_book, altar)					
		# of gen.	c_a_start	c_a_end	c_t_start	c_t_end	# of gen.	c_a_start	c_a_end	c_t_start	c_t_end	c_t_ratio
$\mathcal{C}_1$	DenseNet121						25695	0.223	0.227	3.0E-04	7.0E-04	2.4
$\mathcal{C}_2$	DenseNet169						28155	0.322	0.294	1.4E-04	3.0E-04	2.1
$\mathcal{C}_3$	DenseNet201						24983	0.453	0.467	1.4E-04	2.9E-04	2.1
$\mathcal{C}_4$	MobileNet	49082	0.497	0.394	7.77E-06	4.31E-05						5.5
$\mathcal{C}_5$	NASNetMobile						25098	0.951	0.748	8.3E-05	4.1E-04	5.0
$\mathcal{C}_6$	ResNet50						25448	0.280	0.270	2.9E-04	6.9E-04	2.3
$\mathcal{C}_7$	ResNet101						26178	0.204	0.084	9.1E-05	1.9E-04	2.1
$\mathcal{C}_8$	ResNet152						25328	0.702	0.575	2.0E-05	5.2E-05	2.6
$\mathcal{C}_9$	VGG16	46721	0.455	0.405	1.08E-05	1.90E-05						1.8
$\mathcal{C}_{10}$	VGG19	47668	0.145	0.132	5.32E-05	9.11E-05						1.7

Table 9.10: Direct attack results of $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$ for all (c_a, c_t) pairs after 100 generations (hence less than 48 hours). The results show the time spent by the main operations of $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$ in one generation.

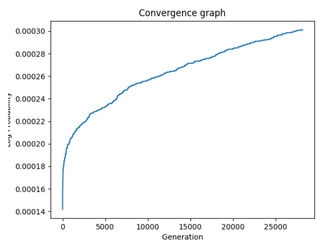
	Input Image	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_6$	$\mathcal{A}_7$	$\mathcal{A}_8$	$\mathcal{A}_9$	$\mathcal{A}_{10}$	
	Image Size (n)	910x604	960x640	910x607	2462x2913	910x607	641x600	1280x800	1280x800	1954x2011	1740x1710	%
Avg. of all CNNs	**time per gen**	**3.528**	**3.790**	**3.427**	**45.570**	**3.469**	**2.499**	**6.239**	**6.248**	**24.815**	**18.871**	
	resize	**0.384**	**0.401**	**0.371**	**3.829**	**0.373**	**0.294**	**0.635**	**0.636**	**2.209**	**1.729**	9.2
	prediction	**0.155**	**0.150**	**0.149**	**0.156**	**0.150**	**0.149**	**0.150**	**0.150**	**0.155**	**0.153**	1.3
	mutation	**2.063**	**2.215**	**1.995**	**29.922**	**2.026**	**1.410**	**3.768**	**3.778**	**16.050**	**12.118**	63.6
	crossover	**0.161**	**0.179**	**0.161**	**2.067**	**0.161**	**0.112**	**0.297**	**0.297**	**1.133**	**0.861**	4.6
	time per gen/n	**6.42E-06**	**6.17E-06**	**6.21E-06**	**6.35E-06**	**6.28E-06**	**6.50E-06**	**6.09E-06**	**6.10E-06**	**6.31E-06**	**6.34E-06**	

Table 9.11: Indirect attack results of $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$ for all (c_a, c_t) pairs after 100 generations. The results show the time spent by the main operations of $\mathrm{EA}^{\mathrm{target},\mathcal{C}}$ in one generation.

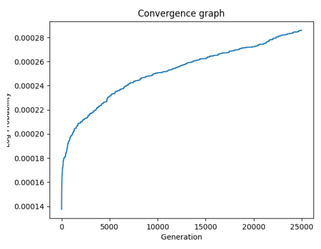
	Input Image	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_6$	$\mathcal{A}_7$	$\mathcal{A}_8$	$\mathcal{A}_9$	$\mathcal{A}_{10}$	
	Image Size (n)	910x604	960x640	910x607	2462x2913	910x607	641x600	1280x800	1280x800	1954x2011	1740x1710	%
Avg. of all CNNs	**time per gen**	0.512	0.516	0.518	0.673	0.517	0.512	0.526	0.530	0.596	0.572	
	resize	0.020	0.022	0.020	0.174	0.020	0.016	0.032	0.032	0.101	0.079	9.4
	prediction	0.154	0.155	0.155	0.155	0.156	0.154	0.155	0.155	0.154	0.154	28.3
	mutation	0.143	0.142	0.145	0.146	0.145	0.147	0.142	0.145	0.144	0.143	26.4
	crossover	0.009	0.010	0.009	0.010	0.009	0.009	0.009	0.009	0.010	0.010	1.7
	time per gen/n	**9.31E-07**	**8.39E-07**	**9.38E-07**	**9.39E-08**	**9.36E-07**	**1.33E-06**	**5.14E-07**	**5.17E-07**	**1.52E-07**	**1.92E-07**	



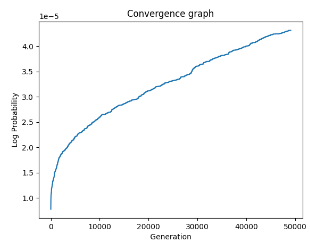
(a) $\mathcal{C}_1 - \mathcal{A}_7^{\text{hr}}$



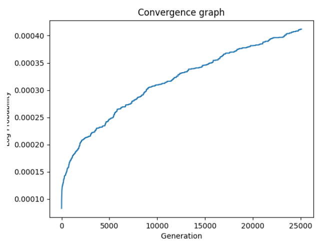
(b) $\mathcal{C}_2 - \mathcal{A}_7^{\text{hr}}$



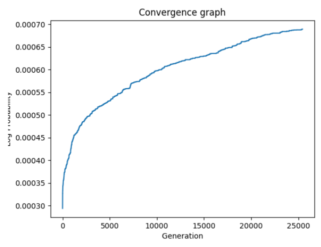
(c) $\mathcal{C}_3 - \mathcal{A}_7^{\text{hr}}$



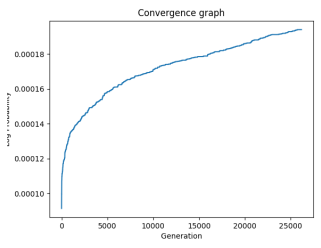
(d) $\mathcal{C}_4 - \mathcal{A}_7^{\text{hr}}$



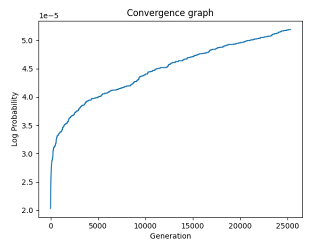
(e) $\mathcal{C}_5 - \mathcal{A}_5^{\text{hr}}$



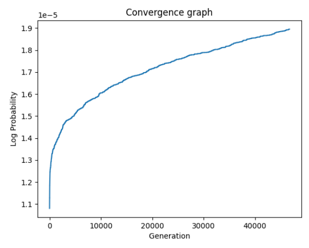
(f) $\mathcal{C}_6 - \mathcal{A}_7^{\text{hr}}$



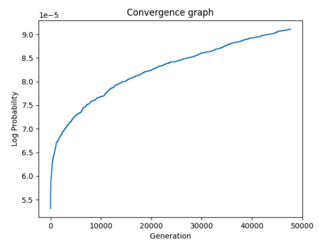
(g) $\mathcal{C}_7 - \mathcal{A}_7^{\text{hr}}$



(h) $\mathcal{C}_8 - \mathcal{A}_7^{\text{hr}}$



(i) $\mathcal{C}_9 - \mathcal{A}_5^{\text{hr}}$



(j) $\mathcal{C}_{10} - \mathcal{A}_5^{\text{hr}}$

Figure 9.12: Convergence characteristics of $\text{EA}^{\text{target},\mathcal{C}}$ aiming at generating within 48 hours a high-resolution adversarial image by directly evolving $\mathcal{A}_5^{\text{hr}}$ for the (toucan, wombat) pair and $\mathcal{C}$ = MobileNet (d), VGG16 (i), VGG19 (j), and $\mathcal{A}_7^{\text{hr}}$ for the (comic book, altar) and $\mathcal{C}$ = DenseNet121 (a), DenseNet169 (b), DenseNet201 (c), NasNetMobile (e), ResNet50 (f), ResNet101 (g), ResNet152 (h).

9.4 Robustness of Attack Against Filters

9.4.1 Without filters

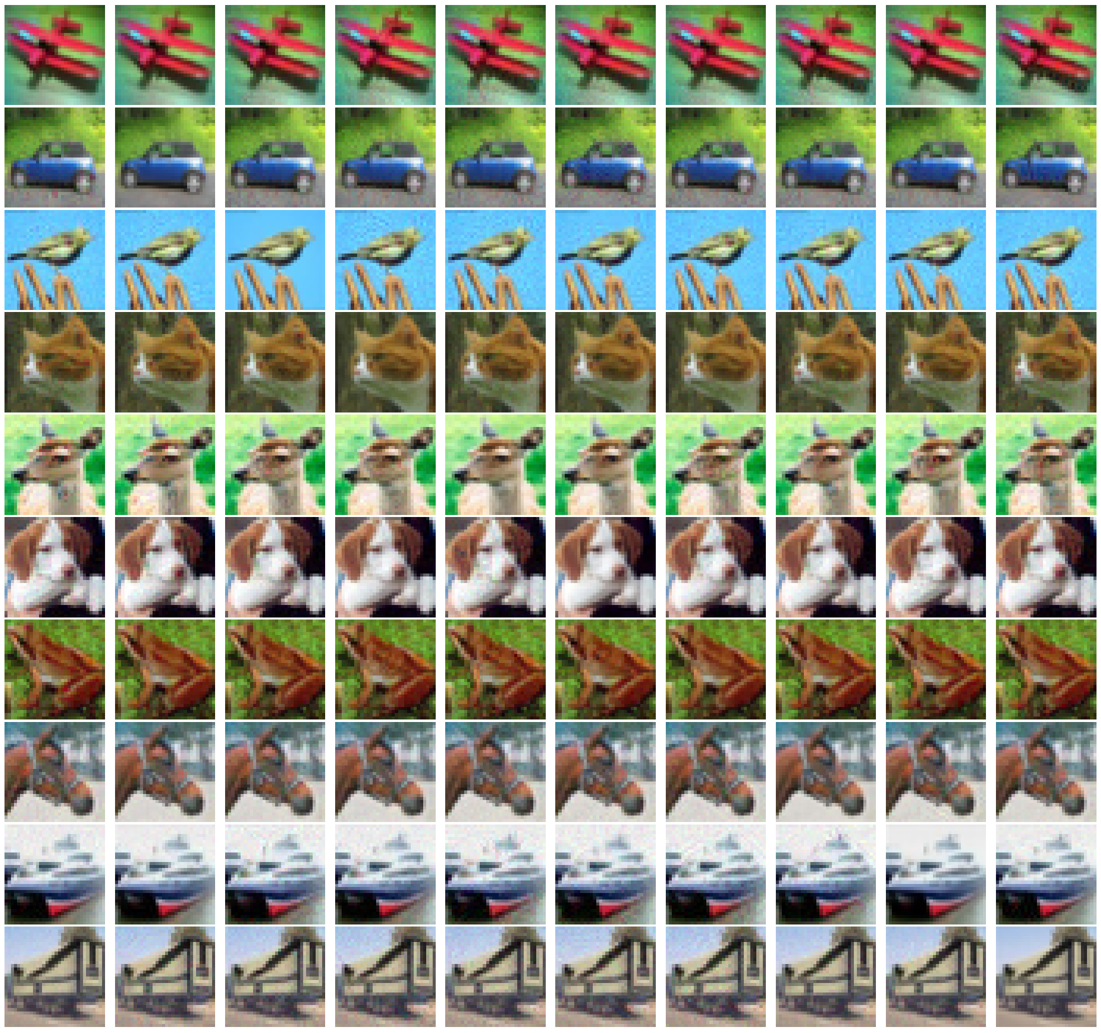


Figure 9.13: For $1 \leq a \leq 10$, the image on the diagonal of the a^{th} row is the ancestor $\mathcal{A}_a$ (recovered from Table 6.1) classified by VGG16 as belonging to the category c_a, and the picture in the t^{th} column, with $t \neq a$, is the adversarial picture $\mathcal{D}_{a,t}(\mathcal{A}_a) = \mathrm{EA}^{\text{target,VGG-16}}_{L_2}(\mathcal{A}_a, c_t)$ classified by VGG16 as belonging to c_t, obtained after the first of the 10 independent runs.

Table 9.12: For $\mathcal{C}$ = VGG16, each of the cell in $(a,t)^{\text{th}}$-position contains a pair (maximum label value, corresponding class) given by $\mathcal{C}$ for $\mathcal{D}_{a,t}(\mathcal{A}_a)$ (with $\mathcal{D}_{a,a}(\mathcal{A}_a) = \mathcal{A}_a$).

	plane c_1	car c_2	bird c_3	cat c_4	deer c_5	dog c_6	frog c_7	horse c_8	ship c_9	truck c_{10}
plane($\mathcal{A}_1$)	0.6900 plane	0.9506 car	0.9501 bird	0.9500 cat	0.9501 deer	0.9500 dog	0.9502 frog	0.9501 horse	0.9537 ship	0.9531 truck
car($\mathcal{A}_2$)	0.9519 plane	0.9999 car	0.9546 bird	0.9515 cat	0.9534 deer	0.9509 dog	0.9508 frog	0.9606 horse	0.9509 ship	0.9502 truck
bird($\mathcal{A}_3$)	0.9502 plane	0.9505 car	0.9999 bird	0.9501 cat	0.9511 deer	0.9509 dog	0.9517 frog	0.9523 horse	0.9506 ship	0.9510 truck
cat($\mathcal{A}_4$)	0.9514 plane	0.9507 car	0.9512 bird	0.9998 cat	0.9510 deer	0.9519 dog	0.9543 frog	0.9503 horse	0.9514 ship	0.9552 truck
deer($\mathcal{A}_5$)	0.9524 plane	0.9501 car	0.9507 bird	0.9514 cat	0.9999 deer	0.9545 dog	0.9520 frog	0.9501 horse	0.9560 ship	0.9510 truck
dog($\mathcal{A}_6$)	0.9516 plane	0.9502 car	0.9529 bird	0.9518 cat	0.9501 deer	0.9996 dog	0.9502 frog	0.9512 horse	0.9508 ship	0.9518 truck
frog($\mathcal{A}_7$)	0.9519 plane	0.9528 car	0.9501 bird	0.9530 cat	0.9521 deer	0.9527 dog	0.9999 frog	0.9529 horse	0.9521 ship	0.9515 truck
horse($\mathcal{A}_8$)	0.9502 plane	0.9523 car	0.9503 bird	0.9568 cat	0.9521 deer	0.9510 dog	0.9521 frog	0.9998 horse	0.9587 ship	0.9514 truck
ship($\mathcal{A}_9$)	0.9504 plane	0.9543 car	0.9581 bird	0.9506 cat	0.9500 deer	0.9516 dog	0.9517 frog	0.9505 horse	0.9996 ship	0.9504 truck
truck($\mathcal{A}_{10}$)	0.9525 plane	0.9532 car	0.9518 bird	0.9517 cat	0.9557 deer	0.9511 dog	0.9516 frog	0.9507 horse	0.9517 ship	0.9984 truck

Table 9.13: For $\mathcal{C}$ = VGG16, the cell in $(a,t)^{\text{th}}$-position gives (top part) the c_a-label value and the c_t-label value, and (bottom part) the maximum label value and corresponding class of $\mathcal{C} \circ F_1$ for $\mathcal{D}_{a,t}(\mathcal{A}_a)$ (with $\mathcal{D}_{a,a}(\mathcal{A}_a) = \mathcal{A}_a$).

	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
plane ($\mathcal{A}_1$)	0.9923	0.9870	0.9830	0.8888	0.5932	0.9845	0.9857	0.9436	0.9007	0.9806
	0.9923	1.56e-03	5.16e-04	4.17e-04	1.71e-04	9.59e-04	1.92e-03	8.26e-05	9.18e-02	2.52e-05
	0.9923	0.9870	0.9830	0.8888	0.5932	0.9845	0.9857	0.9436	0.9007	0.9806
	plane	plane	plane	plane	plane	plane	plane	plane	plane	plane
car ($\mathcal{A}_2$)	2.77e-04	0.7608	1.71e-04	5.46e-05	1.09e-04	7.36e-04	6.81e-04	1.63e-04	2.60e-05	3.29e-03
	0.1501	0.7608	1.44e-04	1.82e-03	4.02e-05	2.29e-04	1.00e-04	1.08e-05	0.9993	8.08e-03
	0.8491	0.7608	0.9958	0.9962	0.9965	0.9837	0.9969	0.9864	0.9993	0.9877
	ship	car	ship	ship	ship	ship	ship	ship	ship	ship
bird ($\mathcal{A}_3$)	0.2966	0.7862	0.9996	0.2070	0.8838	0.9665	0.8639	0.4902	0.4071	0.5798
	5.15e-02	0.1254	0.9996	8.18e-04	1.38e-04	2.49e-04	0.1120	5.17e-04	0.5643	1.63e-03
	0.4964	0.7862	0.9996	0.6009	0.8838	0.9665	0.8639	0.4902	0.5643	0.5798
	ship	bird	bird	ship	bird	bird	bird	bird	ship	bird
cat ($\mathcal{A}_4$)	0.1906	4.82e-02	4.13e-02	0.9176	0.6035	0.1058	0.5140	0.1909	1.27e-02	2.49e-02
	0.1682	2.36e-02	2.79e-04	0.9176	5.37e-03	1.72e-03	4.64e-02	6.87e-02	0.9800	1.78e-02
	0.5977	0.8971	0.9162	0.9176	0.6035	0.8079	0.5140	0.4871	0.9800	0.9054
	ship	ship	ship	cat	cat	ship	cat	frog	ship	ship
deer ($\mathcal{A}_5$)	3.90e-05	7.41e-03	9.00e-02	5.71e-04	0.3245	0.7423	2.16e-03	0.1149	7.83e-03	8.01e-04
	0.9985	4.49e-02	2.39e-03	0.9982	0.3245	1.23e-02	3.97e-02	7.53e-04	0.4939	3.41e-02
	0.9985	0.8634	0.8982	0.9982	0.5838	0.7423	0.6883	0.7741	0.4939	0.9616
	plane	plane	cat	cat	plane	deer	cat	cat	ship	cat
dog ($\mathcal{A}_6$)	3.08e-04	9.21e-03	5.23e-03	1.69e-03	1.32e-03	0.0014	7.82e-02	1.03e-02	4.41e-03	5.18e-03
	8.74e-04	1.28e-03	1.18e-04	0.9977	1.80e-04	0.0014	0.4415	3.46e-05	1.19e-02	0.5093
	0.9925	0.6375	0.9727	0.9977	0.9380	0.9983	0.4415	0.9794	0.8320	0.5093
	truck	cat	cat	cat	truck	cat	frog	cat	truck	truck
frog ($\mathcal{A}_7$)	0.5602	0.9198	9.37e-02	0.4898	0.2230	0.2092	0.9140	0.2013	9.97e-04	0.4658
	0.4272	5.41e-03	4.93e-04	0.4423	1.90e-03	5.25e-02	0.9140	1.20e-03	0.9941	1.37e-02
	0.5602	0.9198	0.7740	0.4898	0.7097	0.3961	0.9140	0.4221	0.9941	0.4658
	frog	frog	plane	frog	cat	cat	frog	cat	ship	frog
horse ($\mathcal{A}_8$)	3.06e-05	3.20e-04	1.83e-04	1.89e-04	1.46e-04	5.03e-05	1.09e-05	0.0004	2.27e-05	1.27e-04
	0.9316	2.43e-03	2.29e-02	9.12e-03	4.38e-04	0.8593	7.05e-03	0.0004	0.9470	6.98e-03
	0.9316	0.4645	0.5039	0.3962	0.6209	0.8593	0.8860	0.7479	0.9470	0.8123
	plane	plane	dog	plane	plane	dog	plane	dog	ship	plane
ship ($\mathcal{A}_9$)	0.9136	0.9445	0.1834	0.9034	7.89e-03	1.71e-03	0.9801	3.43e-02	0.9865	0.9242
	7.62e-02	4.45e-04	1.30e-03	8.70e-02	0.6532	0.9306	1.79e-03	1.04e-02	0.9865	3.55e-04
	0.9136	0.9445	0.7320	0.9034	0.6532	0.9306	0.9801	0.9311	0.9865	0.9242
	ship	ship	plane	ship	deer	dog	ship	cat	ship	ship
truck ($\mathcal{A}_{10}$)	2.35e-05	2.68e-04	3.16e-05	1.43e-04	6.35e-05	4.79e-05	6.68e-04	3.60e-04	1.38e-04	0.0001
	0.9994	4.12e-05	4.41e-04	5.57e-05	1.43e-04	4.18e-05	4.62e-05	1.65e-03	1.30e-02	0.0001
	0.9994	0.9971	0.9970	0.9941	0.9946	0.9941	0.8366	0.9947	0.9865	0.9973
	plane	plane	plane	plane	plane	plane	ship	plane	plane	plane

Table 9.14: For $\mathcal{C}$ = VGG16, the cell in $(a,t)^{\text{th}}$-position gives (top part) the c_a-label value and the c_t-label value, and (bottom part) the maximum label value and corresponding class of $\mathcal{C} \circ F_2$ for $\mathcal{D}_{a,t}(\mathcal{A}_a)$ (with $\mathcal{D}_{a,a}(\mathcal{A}_a) = \mathcal{A}_a$).

	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
plane ($\mathcal{A}_1$)	0.2591	0.2052	0.1978	0.1256	0.2017	0.1885	0.1907	0.1415	9.10e-02	0.2188
	0.2591	0.5786	0.1627	4.95e-03	5.82e-04	1.89e-02	3.08e-02	1.51e-04	0.7307	3.94e-05
	0.4463	0.5786	0.3455	0.5941	0.6175	0.4134	0.4057	0.7010	0.7307	0.5850
	car	car	ship	ship	ship	ship	car	ship	ship	ship
car ($\mathcal{A}_2$)	5.29e-03	0.9988	9.92e-02	5.78e-03	0.4091	0.1245	5.40e-02	1.37e-02	3.80e-02	0.9989
	3.23e-04	0.9988	0.7795	4.33e-02	7.96e-04	9.59e-04	0.5196	6.14e-05	3.73e-02	1.76e-04
	0.8311	0.9988	0.7795	0.7865	0.4091	0.7438	0.5196	0.9224	0.8378	0.9989
	bird	car	bird	bird	car	bird	frog	bird	bird	car
bird ($\mathcal{A}_3$)	0.9996	0.9998	0.9998	0.9997	0.9997	0.9997	0.9998	0.9996	0.9997	0.9998
	1.98e-04	5.04e-06	0.9998	1.13e-04	5.52e-06	4.45e-05	1.72e-05	2.11e-05	3.80e-05	5.91e-06
	0.9996	0.9998	0.9998	0.9997	0.9997	0.9997	0.9998	0.9996	0.9997	0.9998
	bird	bird	bird	bird	bird	bird	bird	bird	bird	bird
cat ($\mathcal{A}_4$)	0.9876	0.9959	0.9743	0.9992	0.9955	0.9691	0.9983	0.9723	0.9968	0.9917
	4.65e-06	7.52e-07	8.52e-04	0.9992	5.96e-04	3.02e-02	2.37e-04	4.62e-05	8.10e-06	9.37e-07
	0.9876	0.9959	0.9743	0.9992	0.9955	0.9691	0.9983	0.9723	0.9968	0.9917
	cat	cat	cat	cat	cat	cat	cat	cat	cat	cat
deer ($\mathcal{A}_5$)	0.9997	0.9985	0.9989	0.9988	0.9998	0.9983	0.9996	0.9992	0.9985	0.9997
	8.28e-06	1.35e-06	9.30e-04	7.09e-04	0.9998	1.49e-03	1.73e-05	1.43e-04	5.85e-06	1.02e-06
	0.9997	0.9985	0.9989	0.9988	0.9998	0.9983	0.9996	0.9992	0.9985	0.9997
	deer	deer	deer	deer	deer	deer	deer	deer	deer	deer
dog ($\mathcal{A}_6$)	2.17e-04	3.40e-03	2.52e-03	1.48e-04	3.64e-03	3.14e-04	3.71e-04	8.51e-04	1.95e-04	2.33e-04
	1.34e-05	2.26e-06	5.67e-05	0.9998	1.88e-05	3.14e-04	8.29e-06	1.63e-05	2.77e-06	2.94e-06
	0.9997	0.9965	0.9973	0.9998	0.9962	0.9996	0.9995	0.9990	0.9997	0.9997
	cat	cat	cat	cat	cat	cat	cat	cat	cat	cat
frog ($\mathcal{A}_7$)	0.9994	0.9997	0.9995	0.9977	0.9980	0.9941	0.9998	0.9982	0.9995	0.9996
	1.90e-05	8.46e-06	2.25e-04	1.21e-03	8.21e-05	4.55e-03	0.9998	4.94e-05	6.97e-05	7.29e-06
	0.9994	0.9997	0.9995	0.9977	0.9980	0.9941	0.9998	0.9982	0.9995	0.9996
	frog	frog	frog	frog	frog	frog	frog	frog	frog	frog
horse ($\mathcal{A}_8$)	0.7487	0.9692	0.8900	0.9062	0.9967	0.9568	0.9164	0.9997	0.9792	0.9758
	4.02e-04	7.37e-05	9.74e-02	8.47e-02	1.94e-03	3.46e-03	7.37e-04	0.9997	4.28e-04	2.19e-05
	0.7487	0.9692	0.8900	0.9062	0.9967	0.9568	0.9164	0.9997	0.9792	0.9758
	horse	horse	horse	horse	horse	horse	horse	horse	horse	horse
ship ($\mathcal{A}_9$)	1.73e-03	4.33e-02	5.67e-02	1.85e-02	0.8242	0.1091	8.23e-02	7.87e-03	0.3924	1.55e-03
	0.9894	0.7334	1.25e-03	3.34e-04	2.25e-05	8.34e-04	1.54e-04	7.16e-05	0.3924	4.34e-03
	0.9894	0.7334	0.8965	0.6402	0.8242	0.8392	0.4956	0.8441	0.4525	0.9214
	plane	car	plane	plane	ship	plane	plane	plane	plane	plane
truck ($\mathcal{A}_{10}$)	1.61e-03	6.03e-04	1.57e-03	1.05e-04	7.27e-04	4.96e-03	9.52e-03	3.86e-03	1.02e-03	5.62e-03
	0.9955	2.88e-04	2.79e-03	1.88e-04	3.04e-02	4.73e-04	2.58e-04	0.1820	9.07e-05	5.62e-03
	0.9955	0.9873	0.9920	0.9931	0.9670	0.9184	0.9876	0.8099	0.9974	0.9919
	plane	plane	plane	plane	plane	plane	plane	plane	plane	plane

Table 9.15: For $\mathcal{C}$ = VGG16, the cell in $(a,t)^{\text{th}}$-position gives (top part) the c_a-label value and the c_t-label value, and (bottom part) the maximum label value and corresponding class of $\mathcal{C} \circ F_3$ for $\mathcal{D}_{a,t}(\mathcal{A}_a)$ (with $\mathcal{D}_{a,a}(\mathcal{A}_a) = \mathcal{A}_a$).

	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
plane ($\mathcal{A}_1$)	0.7298	0.6388	0.3414	0.1388	0.3576	0.2870	0.3931	0.2510	0.1163	0.3362
	0.7298	6.08e-02	0.1755	8.80e-04	3.35e-04	3.20e-02	1.14e-02	2.19e-04	0.8616	2.08e-05
	0.7298	0.6388	0.4205	0.8034	0.5857	0.5863	0.5051	0.6781	0.8616	0.6479
	plane	plane	ship	ship	ship	ship	ship	ship	ship	ship
car ($\mathcal{A}_2$)	0.3643	0.9997	0.9734	0.9666	0.9858	0.8656	0.9075	0.9986	0.9607	0.9997
	1.23e-03	0.9997	1.79e-02	1.44e-03	7.33e-05	5.59e-04	5.22e-02	8.15e-06	1.97e-03	5.86e-05
	0.5191	0.9997	0.9734	0.9666	0.9858	0.8656	0.9075	0.9986	0.9607	0.9997
	bird	car	car	car	car	car	car	car	car	car
bird ($\mathcal{A}_3$)	0.9998	0.9998	0.9999	0.9998	0.9998	0.9998	0.9999	0.9998	0.9998	0.9998
	7.36e-05	3.88e-06	0.9999	5.44e-05	4.91e-06	2.73e-05	1.26e-05	1.22e-05	2.02e-05	4.76e-06
	0.9998	0.9998	0.9999	0.9998	0.9998	0.9998	0.9999	0.9998	0.9998	0.9998
	bird	bird	bird	bird	bird	bird	bird	bird	bird	bird
cat ($\mathcal{A}_4$)	0.9971	0.9977	0.9873	0.9994	0.9980	0.9969	0.9986	0.9916	0.9975	0.9963
	2.73e-06	1.95e-06	9.16e-03	0.9994	1.09e-04	2.56e-03	2.80e-04	1.33e-04	1.16e-05	1.81e-06
	0.9971	0.9977	0.9873	0.9994	0.9980	0.9969	0.9986	0.9916	0.9975	0.9963
	cat	cat	cat	cat	cat	cat	cat	cat	cat	cat
deer ($\mathcal{A}_5$)	0.9994	0.9995	0.9998	0.9996	0.9998	0.9991	0.9999	0.9995	0.9986	0.9997
	1.02e-05	7.05e-07	1.73e-05	7.95e-05	0.9998	8.14e-04	1.17e-05	2.09e-04	4.97e-06	1.04e-06
	0.9994	0.9995	0.9998	0.9996	0.9998	0.9991	0.9999	0.9995	0.9986	0.9997
	deer	deer	deer	deer	deer	deer	deer	deer	deer	deer
dog ($\mathcal{A}_6$)	0.2027	0.8235	0.4543	1.37e-02	0.1518	0.2668	1.11e-02	0.1644	5.96e-02	3.49e-02
	1.70e-05	3.99e-06	1.20e-03	0.9861	2.06e-05	0.2668	2.52e-05	4.93e-05	4.62e-06	4.38e-06
	0.7969	0.8235	0.5443	0.9861	0.8479	0.7329	0.9886	0.8352	0.9401	0.9649
	cat	dog	cat	cat	cat	cat	cat	cat	cat	cat
frog ($\mathcal{A}_7$)	0.9997	0.9998	0.9998	0.9993	0.9997	0.9994	0.9998	0.9994	0.9997	0.9997
	1.74e-05	1.30e-05	4.40e-05	4.63e-04	1.08e-05	2.41e-04	0.9998	1.72e-05	5.84e-05	6.97e-06
	0.9997	0.9998	0.9998	0.9993	0.9997	0.9994	0.9998	0.9994	0.9997	0.9997
	frog	frog	frog	frog	frog	frog	frog	frog	frog	frog
horse ($\mathcal{A}_8$)	0.9965	0.9985	0.9958	0.9988	0.9997	0.9992	0.9891	0.9998	0.9974	0.9994
	6.08e-05	1.93e-05	3.63e-03	1.24e-04	1.37e-04	5.16e-05	2.94e-05	0.9998	2.45e-05	8.11e-06
	0.9965	0.9985	0.9958	0.9988	0.9997	0.9992	0.9891	0.9998	0.9974	0.9994
	horse	horse	horse	horse	horse	horse	horse	horse	horse	horse
ship ($\mathcal{A}_9$)	0.5341	0.4759	0.8869	0.7291	0.9987	0.7917	0.4933	0.6350	0.9942	0.2402
	0.3343	0.4978	8.15e-04	3.15e-04	3.33e-06	2.92e-04	1.03e-04	5.64e-05	0.9942	1.07e-02
	0.5341	0.4978	0.8869	0.7291	0.9987	0.7917	0.4933	0.6350	0.9942	0.6106
	ship	car	ship	ship	ship	ship	ship	ship	ship	car
truck ($\mathcal{A}_{10}$)	0.9685	0.9701	0.5662	0.6816	0.7751	0.7993	0.8764	0.8361	0.6649	0.9924
	2.94e-02	2.13e-02	4.58e-03	7.19e-04	0.1100	1.29e-04	2.88e-04	8.23e-04	7.35e-03	0.9924
	0.9685	0.9701	0.5662	0.6816	0.7751	0.7993	0.8764	0.8361	0.6649	0.9924
	truck	truck	truck	truck	truck	truck	truck	truck	truck	truck

Table 9.16: For $\mathcal{C}$ = VGG16, the cell in $(a,t)^{\text{th}}$-position gives (top part) the c_a-label value and the c_t-label value, and (bottom part) the maximum label value and corresponding class of $\mathcal{C} \circ F_4$ for $\mathcal{D}_{a,t}(\mathcal{A}_a)$ (with $\mathcal{D}_{a,a}(\mathcal{A}_a) = \mathcal{A}_a$).

	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
plane($\mathcal{A}_1$)	0.4425	1.10e-02	1.62e-02	5.08e-03	6.28e-03	1.16e-02	1.27e-02	2.14e-03	4.610e-02	4.84e-03
	0.4425	0.9871	0.9689	0.9843	0.9813	0.9566	0.9610	0.9815	0.9146	0.9792
	0.5497	0.9871	0.9689	0.9843	0.9813	0.9566	0.9610	0.9815	0.9146	0.9792
	car	car	bird	cat	deer	dog	frog	horse	ship	truck
car($\mathcal{A}_2$)	9.97e-03	0.9999	0.8717	0.1439	0.3505	9.05e-02	0.2418	6.16e-02	0.7124	3.99e-02
	0.9879	0.9999	0.1196	0.8162	0.6083	0.8912	0.7558	0.9287	0.2840	0.9597
	0.9879	0.9999	0.8717	0.8162	0.6083	0.8912	0.7558	0.9287	0.7124	0.9597
	plane	car	car	cat	deer	dog	frog	horse	car	truck
bird($\mathcal{A}_3$)	1.63e-03	1.75e-03	0.9999	5.26e-03	8.46e-03	1.27e-02	4.31e-03	8.68e-03	1.49e-03	3.86e-03
	0.9710	0.9903	0.9999	0.9268	0.9705	0.9505	0.9952	0.9505	0.9916	0.9606
	0.9710	0.9903	0.9999	0.9268	0.9705	0.9505	0.9952	0.9505	0.9916	0.9606
	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
cat($\mathcal{A}_4$)	8.86e-03	4.34e-04	3.31e-02	0.9998	6.74e-03	4.28e-02	2.03e-03	7.95e-02	3.50e-03	5.92e-04
	0.8439	0.9948	0.7611	0.9998	0.9860	7.94e-02	0.9979	0.6060	0.9079	0.9833
	0.8439	0.9948	0.7611	0.9998	0.9860	0.8764	0.9979	0.6060	0.9079	0.9833
	plane	car	bird	cat	deer	frog	frog	horse	ship	truck
deer($\mathcal{A}_5$)	1.65e-04	9.22e-05	3.67e-02	2.14e-03	0.9999	2.27e-02	3.95e-04	1.00e-02	1.27e-03	1.16e-03
	0.9932	0.9970	0.9630	0.9902	0.9999	0.9771	0.9987	0.9852	0.9957	0.9951
	0.9932	0.9970	0.9630	0.9902	0.9999	0.9771	0.9987	0.9852	0.9957	0.9951
	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
dog($\mathcal{A}_6$)	3.01e-03	6.08e-05	3.49e-03	7.11e-02	8.47e-04	0.9998	6.87e-04	2.71e-03	4.09e-04	2.47e-05
	0.9524	0.9985	0.9830	0.9286	0.9943	0.9998	0.9960	0.9960	0.9974	0.9994
	0.9524	0.9985	0.9830	0.9286	0.9943	0.9998	0.9960	0.9960	0.9974	0.9994
	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
frog($\mathcal{A}_7$)	7.74e-02	1.94e-02	9.23e-02	0.2083	0.1896	0.4448	0.9999	0.5326	8.68e-02	4.41e-02
	0.9017	0.9796	0.9075	0.7900	0.8091	0.5505	0.9999	0.4461	0.9092	0.9519
	0.9017	0.9796	0.9075	0.7900	0.8091	0.5505	0.9999	0.5326	0.9092	0.9519
	plane	car	bird	cat	deer	dog	frog	frog	ship	truck
horse($\mathcal{A}_8$)	5.42e-03	5.40e-03	5.30e-03	1.65e-02	8.67e-03	1.40e-02	1.94e-04	0.9998	1.63e-02	6.68e-03
	0.9515	0.9768	0.8715	0.8458	0.9852	0.9342	0.9958	0.9998	0.9648	0.9316
	0.9515	0.9768	0.8715	0.8458	0.9852	0.9342	0.9958	0.9998	0.9648	0.9316
	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
ship($\mathcal{A}_9$)	0.2174	0.1769	3.05e-02	2.13e-02	6.81e-03	0.2438	5.26e-03	3.09e-02	0.9997	6.52e-02
	0.6631	0.8214	0.9155	0.9712	0.8909	0.6297	0.9929	0.9414	0.9997	0.9095
	0.6631	0.8214	0.9155	0.9712	0.8909	0.6297	0.9929	0.9414	0.9997	0.9095
	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
truck($\mathcal{A}_{10}$)	0.1588	1.10e-02	2.70e-02	4.66e-03	0.2818	1.23e-02	6.82e-03	0.1163	6.94e-02	0.9993
	0.8403	0.9869	0.9666	0.9878	0.6789	0.9517	0.9914	0.7095	0.9270	0.9993
	0.8403	0.9869	0.9666	0.9878	0.6789	0.9517	0.9914	0.7095	0.9270	0.9993
	plane	car	bird	cat	deer	dog	frog	horse	ship	truck

Table 9.17: For $\mathcal{C}$ = VGG16 and $F_5 = F_3 \circ F_4$, the cell in $(a,t)^{\text{th}}$-position gives (top part) the c_a-label value and the c_t-label value, and (bottom part) the maximum label value and corresponding class of $\mathcal{C} \circ F_5$ for $\mathcal{D}_{a,t}(\mathcal{A}_a)$ (with $\mathcal{D}_{a,a}(\mathcal{A}_a) = \mathcal{A}_a$).

	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
plane($\mathcal{A}_1$)	0.8817	0.8366	0.5261	0.2653	0.5060	0.5797	0.5169	0.3472	0.1666	0.6224
	0.8817	4.33e-02	0.1688	9.90e-04	4.08e-04	3.43e-02	1.12e-02	3.96e-04	0.8131	3.38e-05
	0.8817	0.8366	0.5261	0.6487	0.5060	0.5797	0.5169	0.5715	0.8131	0.6224
	plane	plane	plane	ship	plane	plane	plane	ship	ship	plane
car($\mathcal{A}_2$)	0.9637	0.9998	0.9907	0.9971	0.9980	0.9907	0.9935	0.9994	0.9961	0.9997
	2.85e-04	0.9998	7.17e-03	3.92e-04	2.70e-05	1.01e-04	3.75e-03	7.29e-06	3.66e-04	6.04e-05
	0.9637	0.9998	0.9907	0.9971	0.9980	0.9907	0.9935	0.9994	0.9961	0.9997
	car	car	car	car	car	car	car	car	car	car
bird($\mathcal{A}_3$)	0.9998	0.9998	0.9999	0.9997	0.9998	0.9998	0.9999	0.9998	0.9998	0.9998
	8.86e-05	3.74e-06	0.9999	6.25e-05	6.93e-06	3.01e-05	1.42e-05	1.25e-05	2.32e-05	4.46e-06
	0.9998	0.9998	0.9999	0.9997	0.9998	0.9998	0.9999	0.9998	0.9998	0.9998
	bird	bird	bird	bird	bird	bird	bird	bird	bird	bird
cat($\mathcal{A}_4$)	0.9990	0.9984	0.9853	0.9997	0.9987	0.9982	0.9990	0.9833	0.9983	0.9978
	3.46e-06	2.06e-06	1.13e-02	0.9997	7.19e-05	9.95e-04	2.55e-04	2.29e-04	1.57e-05	2.76e-06
	0.9990	0.9984	0.9853	0.9997	0.9987	0.9982	0.9990	0.9833	0.9983	0.9978
	cat	cat	cat	cat	cat	cat	cat	cat	cat	cat
deer($\mathcal{A}_5$)	0.9982	0.9959	0.9996	0.9995	0.9992	0.9974	0.9999	0.9988	0.9958	0.9995
	1.71e-05	1.77e-06	1.43e-05	5.01e-05	0.9992	2.49e-03	1.28e-05	6.28e-04	7.19e-06	1.58e-06
	0.9982	0.9959	0.9996	0.9995	0.9992	0.9974	0.9999	0.9988	0.9958	0.9995
	deer	deer	deer	deer	deer	deer	deer	deer	deer	deer
dog($\mathcal{A}_6$)	0.3989	0.9915	0.8812	0.1154	0.1433	0.9148	7.58e-02	0.6196	0.4921	0.1617
	2.67e-05	1.96e-06	2.37e-03	0.8843	2.35e-05	0.9148	5.47e-05	7.70e-05	6.71e-06	1.04e-05
	0.6006	0.9915	0.8812	0.8843	0.8563	0.9148	0.9238	0.6196	0.5074	0.8379
	cat	dog	dog	cat	cat	dog	cat	dog	cat	cat
frog($\mathcal{A}_7$)	0.9998	0.9998	0.9998	0.9997	0.9998	0.9996	0.9999	0.9997	0.9998	0.9997
	2.03e-05	3.11e-05	6.01e-05	1.86e-04	1.38e-05	1.14e-04	0.9999	1.07e-05	4.63e-05	1.33e-05
	0.9998	0.9998	0.9998	0.9997	0.9998	0.9996	0.9999	0.9997	0.9998	0.9997
	frog	frog	frog	frog	frog	frog	frog	frog	frog	frog
horse($\mathcal{A}_8$)	0.9924	0.9953	0.9900	0.9989	0.9998	0.9991	0.9945	0.9998	0.9963	0.9995
	8.50e-05	4.29e-05	9.55e-03	8.48e-05	7.39e-05	4.19e-05	2.76e-05	0.9998	2.31e-05	1.99e-05
	0.9924	0.9953	0.9900	0.9989	0.9998	0.9991	0.9945	0.9998	0.9963	0.9995
	horse	horse	horse	horse	horse	horse	horse	horse	horse	horse
ship($\mathcal{A}_9$)	0.7220	0.1382	0.5000	0.7736	0.9983	0.6808	0.4648	0.6148	0.9945	0.4792
	4.71e-02	0.8536	6.50e-04	3.24e-04	3.35e-06	3.04e-04	8.79e-05	3.85e-05	0.9945	9.87e-03
	0.7220	0.8536	0.5000	0.7736	0.9983	0.6808	0.4955	0.6148	0.9945	0.4792
	ship	car	ship	ship	ship	ship	car	ship	ship	ship
truck($\mathcal{A}_{10}$)	0.9894	0.9847	0.5815	0.9414	0.9378	0.9084	0.9752	0.9194	0.9244	0.9987
	9.42e-03	1.34e-02	2.45e-03	3.01e-04	2.02e-02	1.71e-04	1.62e-04	5.57e-04	7.97e-03	0.9987
	0.9894	0.9847	0.5815	0.9414	0.9378	0.9084	0.9752	0.9194	0.9244	0.9987
	truck	truck	truck	truck	truck	truck	truck	truck	truck	truck

9.4.2 With filters

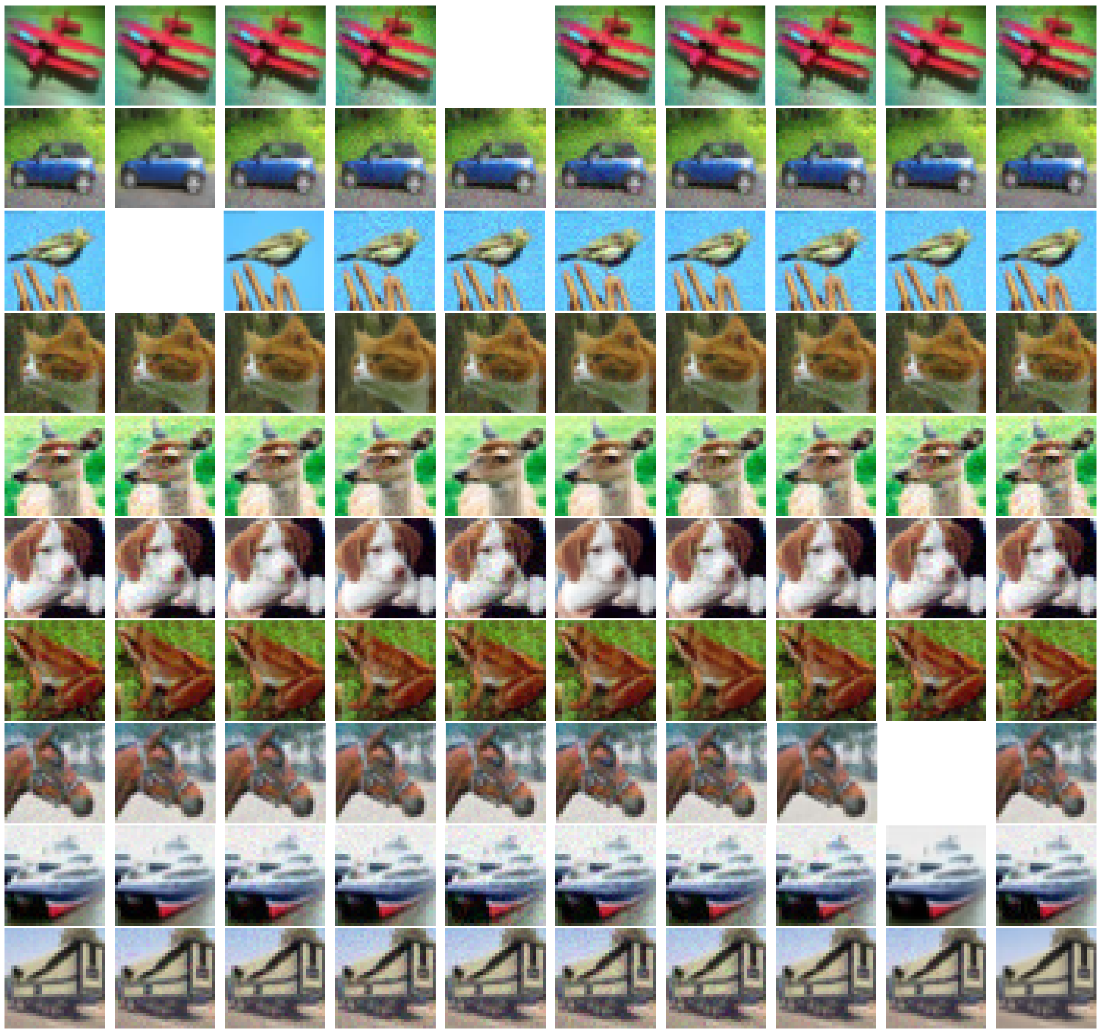


Figure 9.14: For $1 \leq a \leq 10$, the image on the diagonal at the $(a, a)^{th}$ position is the ancestor $\mathcal{A}_a$ (recovered from Table 6.1) classified by VGG16 as belonging to the category c_a. The picture in the $(a, t)^{th}$ position, with $t \neq a$, is the adversarial picture $\mathcal{D}^{F_5}_{a,t}(\mathcal{A}_a) = \mathrm{EA}^{\text{target},\text{VGG-16}}_{L_2,F_5}(\mathcal{A}_a, c_t)$ obtained after the first successful run of the algorithm. Both images $\mathcal{D}^{F_5}_{a,t}(\mathcal{A}_a)$ and $F_5(\mathcal{D}^{F_5}_{a,t}(\mathcal{A}_a))$ are classified by VGG16 as belonging to c_t with a c_t-label value ≥ 0.95. The 3 fully empty pictures correspond to the $(ancestor(\mathcal{A}_a), target)$ combinations for which the algorithm did not terminate successfully for any of the 10 runs.

Table 9.18: For $\mathcal{C}$ = VGG16, each of the two parts of the cell in $(a, t)^{\text{th}}$-position contains a pair (maximum label value, corresponding class) given by $\mathcal{C}$ (top) and by $\mathcal{C} \circ F_5$ (bottom) for $\mathcal{D}^{F_5}_{a,t}(\mathcal{A}_a)$ (with $\mathcal{D}^{F_5}_{a,a}(\mathcal{A}_a) = \mathcal{A}_a$) whenever applicable (3 cells are empty).

	plane c_1	car c_2	bird c_3	cat c_4	deer c_5	dog c_6	frog c_7	horse c_8	ship c_9	truck c_{10}
plane($\mathcal{A}_1$)	0.6900, plane	0.9871, car	0.9513, bird	0.9503, cat		0.9503, dog	0.9510, frog	0.9510, horse	0.9503, ship	0.9502, truc
	0.8817, plane	0.9503, car	0.9828, bird	0.9702, cat		0.9840, dog	0.9814, frog	0.9737, horse	0.9863, ship	0.9824, truc
car($\mathcal{A}_2$)	0.9533, plane	0.9999, car	0.9526, bird	0.9529, cat	0.9505, deer	0.9513, dog	0.9500, frog	0.9505, horse	0.9614, ship	0.9507, truc
	0.9998, plane	0.9998, car	0.9998, bird	0.9984, cat	0.9998, deer	0.9987, dog	0.9994, frog	0.9998, horse	0.9998, ship	0.9997, truc
bird($\mathcal{A}_3$)	0.9509, plane		0.9999, bird	0.9507, cat	0.9514, deer	0.9502, dog	0.9507, frog	0.9521, horse	0.9500, ship	0.9503, truc
	0.9992, plane		0.9999, bird	0.9848, cat	0.9995, deer	0.9933, dog	0.9978, frog	0.9788, horse	0.9909, ship	0.9796, truc
cat($\mathcal{A}_4$)	0.9547, plane	0.9521, car	0.9514, bird	0.9998, cat	0.9534, deer	0.9515, dog	0.9546, frog	0.9537, horse	0.9546, ship	0.9552, truc
	0.9964, plane	0.9997, car	0.9996, bird	0.9997, cat	0.9996, deer	0.9997, dog	0.9983, frog	0.9999, horse	0.9967, ship	0.9992, truc
deer($\mathcal{A}_5$)	0.9509, plane	0.9516, car	0.9509, bird	0.9506, cat	0.9999, deer	0.9528, dog	0.9549, frog	0.9501, horse	0.9517, ship	0.9503, truc
	0.9987, plane	09922, car	0.9989, bird	0.9998, cat	0.9992, deer	0.9991, dog	0.9999, frog	0.9994, horse	0.9999, ship	0.9997, truc
dog($\mathcal{A}_6$)	0.9507, plane	0.9514, car	0.9523, bird	0.9556, cat	0.9517, deer	0.9996, dog	0.9504, frog	0.9535, horse	0.9528, ship	0.9517, truc
	0.9989, plane	0.9997, car	0.9996, bird	0.9993, cat	0.9989, deer	0.9148, dog	0.9989, frog	0.9995, horse	0.9980, ship	0.9993, truc
frog($\mathcal{A}_7$)	0.9509, plane	0.9506, car	0.9545, bird	0.9502, cat	0.9555, deer	0.9554, dog	0.9999, frog	0.9520, horse	0.9510, ship	0.9539, truc
	0.9996, plane	0.9996, car	0.9995, bird	0.9998, cat	0.9999, deer	0.9999, dog	0.9999, frog	0.9999, horse	0.9997, ship	0.9994, truc
horse($\mathcal{A}_8$)	0.9503, plane	0.9541, car	0.9528, bird	0.9659, cat	0.9541, deer	0.9580, dog	0.9500, frog	0.9998, horse		0.9516, truc
	0.9998, plane	0.9997, car	0.9997, bird	0.9995, cat	0.9997, deer	0.9998, dog	0.9999, frog	0.9998, horse		0.9987, truc
ship($\mathcal{A}_9$)	0.9531, plane	0.9509, car	0.9516, bird	0.9517, cat	0.9508, deer	0.9523, dog	0.9510, frog	0.9539, horse	0.9996, ship	0.9515, truc
	0.9996, plane	0.9991, car	0.9999, bird	0.9990, cat	0.9983, deer	0.9995, dog	0.9972, frog	0.9998, horse	0.9945, ship	0.9902, truc
truck($\mathcal{A}_{10}$)	0.9507, plane	0.9503, car	0.9535, bird	0.9515, cat	0.9509, deer	0.9509, dog	0.9526, frog	0.9542, horse	0.9503, ship	0.9984, truc
	0.9958, plane	0.9993, car	0.9982, bird	0.9992, cat	0.9994, deer	0.9993, dog	0.9997, frog	0.9989, horse	0.9980, ship	0.9987, truc

Table 9.19: For $\mathcal{C}$ = VGG16, each of the 4 parts of the cell in $(a,t)^{\text{th}}$-position contains a pair (maximum label value, corresponding class) given, respectively from the top to the bottom, by $\mathcal{C} \circ F_1$, $\mathcal{C} \circ F_2$, $\mathcal{C} \circ F_3$, and $\mathcal{C} \circ F_4$ for $\mathcal{D}^{F_5}_{a,t}(\mathcal{A}_a)$ (with $\mathcal{D}^{F_5}_{a,a}(\mathcal{A}_a) = \mathcal{A}_a$) whenever applicable.

	plane c_1	car c_2	bird c_3	cat c_4	deer c_5	dog c_6	frog c_7	horse c_8	ship c_9
plane($\mathcal{A}_1$)	0.9923, plane 0.4463, car 0.7298, plane 0.5497, car	0.9590, plane 0.8199, car 0.8210, car 0.9942, car	0.9734, plane 0.3563, ship 0.9223, bird 0.9812, bird	0.5456, plane 0.5816, ship 0.6407, cat 0.9814, cat		0.6592, plane 0.5081, ship 0.8733, dog 0.9681, dog	0.9511, plane 0.3520, car 0.9058, frog 0.9612, frog	0.9203, plane 0.5766, ship 0.4090, horse 0.9661, horse	0.8721, plane 0.7817, ship 0.9859, ship 0.9092, ship
car($\mathcal{A}_2$)	0.9709, ship 0.6160, bird 0.9994, plane 0.9788, plane	0.7608, car 0.9988, car 0.9997, car 0.9999, car	0.9204, ship 0.9729, bird 0.9996, bird 0.9510, car	0.9925, ship 0.8053, cat 0.9940, cat 0.9553, cat	0.9990, ship 0.8379, bird 0.9989, deer 0.8565, car	0.5563, ship 0.8068, bird 0.9163, dog 0.4930, car	0.9989, ship 0.5687, frog 0.9972, frog 0.8633, frog	0.9984, ship 0.8987, bird 0.9986, horse 0.7029, horse	0.9996, ship 0.5995, ship 0.9993, ship 0.5694, ship
bird($\mathcal{A}_3$)	0.7736, bird 0.9993, bird 0.9634, plane 0.9256, plane		0.9996, bird 0.9998, bird 0.9999, bird 0.9999, bird	0.4883, cat 0.9989, bird 0.6508, bird 0.9623, cat	0.8968, bird 0.9997, bird 0.7539, deer 0.9641, deer	0.7456, ship 0.9994, bird 0.6483, dog 0.9335, dog	0.8491, bird 0.9998, bird 0.9180, bird 0.9925, frog	0.9912, bird 0.9993, bird 0.9183, bird 0.9805, horse	0.8356, ship 0.9996, bird 0.9361, bird 0.9838, ship
cat($\mathcal{A}_4$)	0.3250, plane 0.9388, cat 0.4563, bird 0.5571, frog	0.6128, cat 0.9380, cat 0.9514, car 0.9710, car	0.5194, ship 0.9709, cat 0.9990, bird 0.7841, frog	0.9176, cat 0.9992, cat 0.9994, cat 0.9998, cat	0.9581, frog 0.9672, cat 0.9952, deer 0.6331, deer	0.4988, cat 0.6787, dog 0.9995, dog 0.9296, frog	0.6131, cat 0.9895, cat 0.9780, frog 0.9959, frog	0.5113, cat 0.6719, cat 0.9997, horse 0.6838, frog	0.5035, ship 0.9904, cat 0.8182, ship 0.8859, ship
deer($\mathcal{A}_5$)	0.9630, plane 0.9974, deer 0.9309, plane 0.9857, plane	0.6593, car 0.8741, deer 0.7593, bird 0.9851, car	0.9989, cat 0.9817, deer 0.9804, bird 0.9812, bird	0.9990, cat 0.9823, deer 0.9994, cat 0.9775, cat	0.5838, plane 0.9998, deer 0.9998, deer 0.9999, deer	0.4605, cat 0.9953, deer 0.7601, dog 0.9623, dog	0.9763, cat 0.9991, deer 0.9991, frog 0.9993, frog	0.7703, cat 0.9978, deer 0.9783, horse 0.9761, horse	0.4648, cat 0.9979, deer 0.9995, ship 0.9898, ship
dog($\mathcal{A}_6$)	0.7988, truck 0.9994, cat 0.9467, plane 0.6362, plane	0.8132, frog 0.9994, cat 0.8896, car 0.6567, car	0.9632, cat 0.9984, cat 0.9971, bird 0.9670, bird	0.9979, cat 0.9997, cat 0.9991, cat 0.9376, cat	0.9329, frog 0.9884, cat 0.8806, deer 0.9768, deer	0.9983, cat 0.9996, cat 0.7329, cat 0.9998, dog	0.9000, frog 0.9963, cat 0.7521, frog 0.9923, frog	0.8948, cat 0.9984, cat 0.9926, horse 0.9558, horse	0.9982, truck 0.9989, cat 0.6905, cat 0.9460, ship
frog($\mathcal{A}_7$)	0.9304, plane 0.9991, frog 0.9971, plane 0.8194, plane	0.8875, frog 0.9995, frog 0.9981, car 0.9574, car	0.6184, frog 0.9906, frog 0.9851, bird 0.9367, bird	0.8088, frog 0.6322, frog 0.9997, cat 0.6233, frog	0.7472, cat 0.9745, deer 0.9998, deer 0.6790, frog	0.7688, cat 0.6311, frog 0.9996, dog 0.8118, frog	0.9140, frog 0.9998, frog 0.9998, frog 0.9999, frog	0.5688, frog 0.5651, horse 0.9999, horse 0.6397, frog	0.8974, ship 0.9976, frog 0.9982, ship 0.6303, ship
horse($\mathcal{A}_8$)	0.7407, plane 0.9532, horse 0.9977, plane 0.8848, plane	0.7159, plane 0.9753, horse 0.9911, car 0.9716, car	0.6002, plane 0.8861, bird 0.9990, bird 0.9888, bird	0.8016, plane 0.6032, cat 0.9982, cat 0.9538, cat	0.7816, bird 0.9827, horse 0.9989, deer 0.9506, deer	0.3995, bird 0.8946, horse 0.9997, dog 0.7921, bird	0.4876, plane 0.6117, cat 0.9998, frog 0.9910, frog	0.7479, dog 0.9997, horse 0.9998, horse 0.9998, horse	
ship($\mathcal{A}_9$)	0.8772, ship 0.9977, plane 0.9995, plane 0.4929, plane	0.9944, ship 0.7723, car 0.9924, car 0.7072, car	0.6944, cat 0.9636, plane 0.9997, bird 0.4667, ship	0.8931, ship 0.9109, plane 0.9631, cat 0.9631, cat	0.7483, car 0.6752, ship 0.8506, deer 0.6244, deer	0.5348, dog 0.9598, plane 0.9901, dog 0.2485, ship	0.9940, ship 0.8060, plane 0.5932, ship 0.9813, frog	0.3876, cat 0.9857, plane 0.9996, horse 0.5592, horse	0.9865, ship 0.4525, plane 0.9942, ship 0.9997, ship
truck($\mathcal{A}_{10}$)	0.9994, plane 0.9946, plane 0.9964, plane 0.8131, plane	0.5934, plane 0.9903, plane 0.9976, car 0.9432, car	0.9977, plane 0.9899, plane 0.9765, bird 0.9767, bird	0.6818, ship 0.9610, plane 0.9985, cat 0.8556, cat	0.9290, plane 0.9766, plane 0.9984, deer 0.9121, deer	0.7131, plane 0.9682, plane 0.9984, dog 0.9743, dog	0.9674, ship 0.9586, plane 0.9997, frog 0.9925, frog	0.9840, plane 0.5843, horse 0.9972, horse 0.5812, horse	0.9654, plane 0.9970, plane 0.9776, ship 0.9181, ship

9.5 Comparative Analysis of the EA and BIM Adversarial Attacks

9.5.1 Ancestor and adversarial images

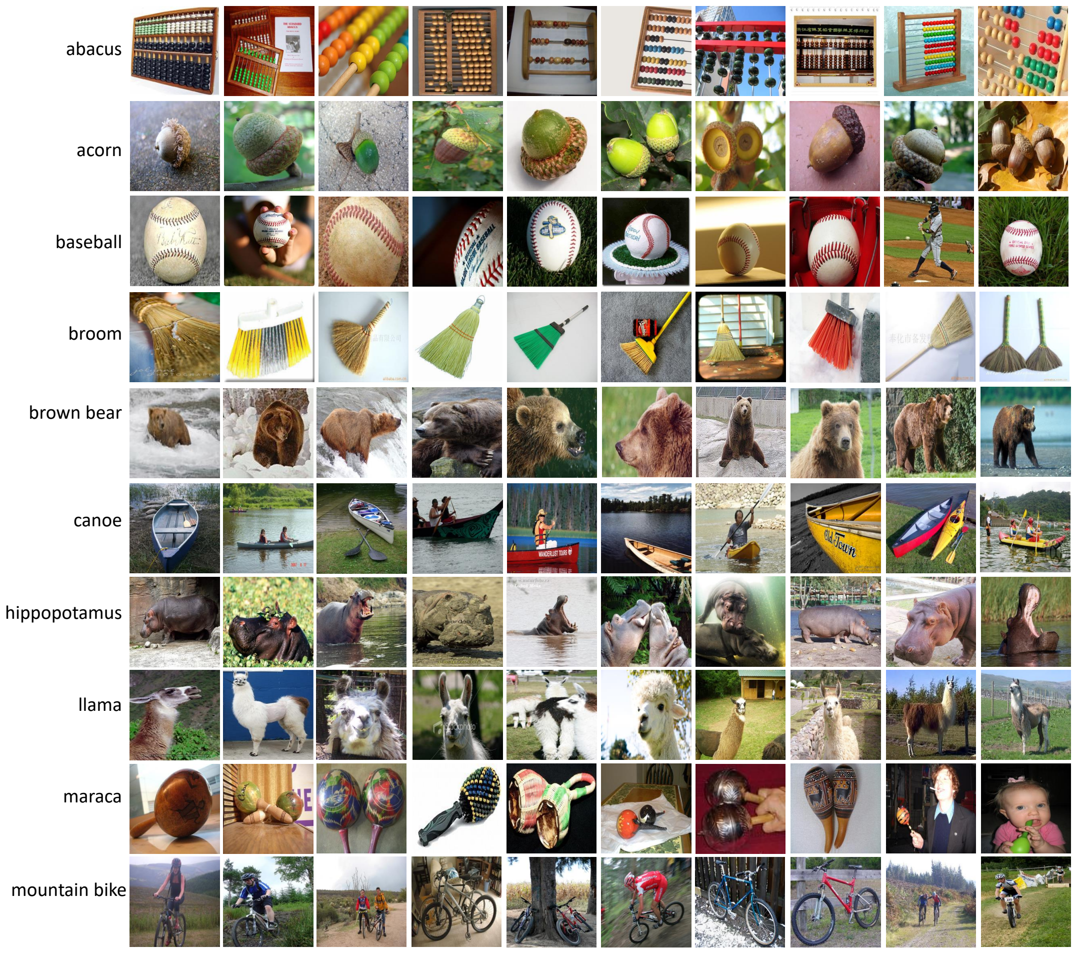


Figure 9.15: The 100 ancestor images $\mathcal{A}_q^p$ used in the experiments. $\mathcal{A}_q^p$ pictured in the q^{th} row and p^{th} column ($1 \leq p, q \leq 10$) is randomly chosen from the ImageNet validation set of the ancestor category c_{a_q} specified on the left of the q^{th} row.

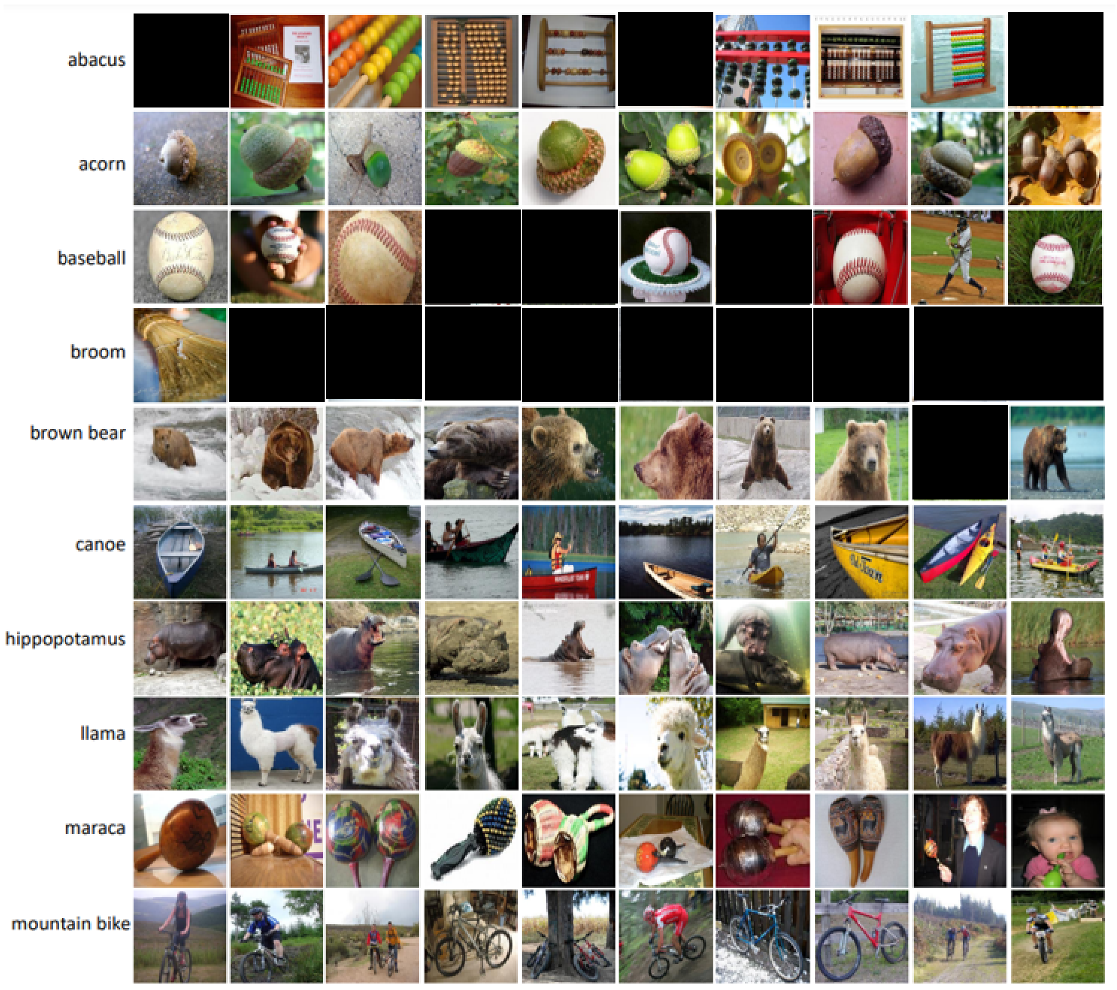


Figure 9.16: The 84 convenient ancestor images $\mathcal{A}_q^p$ used in the experiments, for which both the EA and BIM created 0.999-strong adversarial images $\mathcal{D}_k^{EA}(\mathcal{A}_q^p)$ and $\mathcal{D}_k^{BIM}(\mathcal{A}_q^p)$.

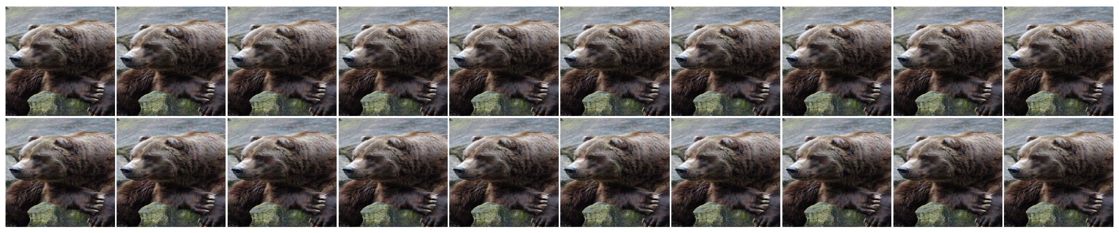


Figure 9.17: Adversarial images $\mathcal{D}_k^{atk}(\mathcal{A}_5^4)$ stemming from the $\mathcal{A}_5^4$ ancestor, obtained with the EA (top) and BIM (bottom). From left to right, the attacked CNNs are $\mathcal{C}_1 \cdots \mathcal{C}_{10}$.

	(c_{a_1}, c_{t_1})	(c_{a_2}, c_{t_2})	(c_{a_3}, c_{t_3})	(c_{a_4}, c_{t_4})	(c_{a_5}, c_{t_5})	(c_{a_6}, c_{t_6})	(c_{a_7}, c_{t_7})	(c_{a_8}, c_{t_8})	$(c_{a_9}, c_{t_9}$
$\mathcal{C}_1$	7,3,3	10,7,7	10,3,3	10,3,3	6,0,0	10,8,8	10,7,7	10,5,5	10,5,5
$\mathcal{C}_2$	7,3,3	10,8,8	8,0,0	9,4,3	6,0,0	10,4,4	9,2,2	9,5,5	9,6,5
$\mathcal{C}_3$	4,2,1	9,7,6	7,3,2	7,9,7	3,0,0	7,5,4	9,3,3	8,4,3	8,8,7
$\mathcal{C}_4$	4,1,1	7,7,5	6,1,0	8,6,5	2,1,1	9,2,2	8,5,3	8,5,3	7,8,5
$\mathcal{C}_5$	4,1,1	7,7,5	5,2,1	8,7,6	1,0,0	6,3,3	7,1,1	7,5,4	8,10,8
$\mathcal{C}_6$	5,5,3	7,8,5	4,1,0	5,5,2	2,1,1	8,7,7	7,8,6	7,6,4	7,9,6
$\mathcal{C}_7$	4,4,2	7,7,4	4,4,1	8,10,8	4,0,0	7,8,6	8,8,7	8,10,8	8,8,6
$\mathcal{C}_8$	4,4,3	8,9,7	6,4,2	5,5,3	1,0,0	7,4,4	9,8,8	7,7,5	7,7,4
$\mathcal{C}_9$	6,5,4	8,10,8	7,2,1	8,10,8	6,1,1	8,10,8	8,10,8	8,9,7	8,9,8
$\mathcal{C}_{10}$	7,5,4	8,7,7	8,2,1	8,8,7	7,1,1	8,10,8	8,9,7	8,10,8	9,5,5
Total	52, 33, 25	81, 77, 62	65, 22, 11	76, 67, 52	38, 4, 4	80, 61, 54	83, 61, 52	80, 66, 52	81, 75,

Table 9.20: For $1 \leq k, q \leq 10$, the cell at the intersection of the row $\mathcal{C}_k$ and column (c_{a_q}, c_{t_q}) is composed of a triplet α, β, γ, where α is the number of ancestors in c_{a_q} for which $\text{EA}^{\text{target},\mathcal{C}_k}$ created 0.999-strong adversarial images, β is the number of ancestors in c_{a_q} for which BIM_k created 0.999-strong adversarial images, and γ is the number of common ancestors for which both algorithms terminated successfully.

Figure 9.18: Adversarial images $\mathcal{D}_k^{atk}(\mathcal{A}_{10}^8)$ stemming from the $\mathcal{A}_{10}^8$ ancestor, obtained with the EA (top) and BIM (bottom). From left to right, the attacked CNNs are $\mathcal{C}_1 \cdots \mathcal{C}_{10}$.

9.5.2 Local effect of adversarial noise on target CNNs

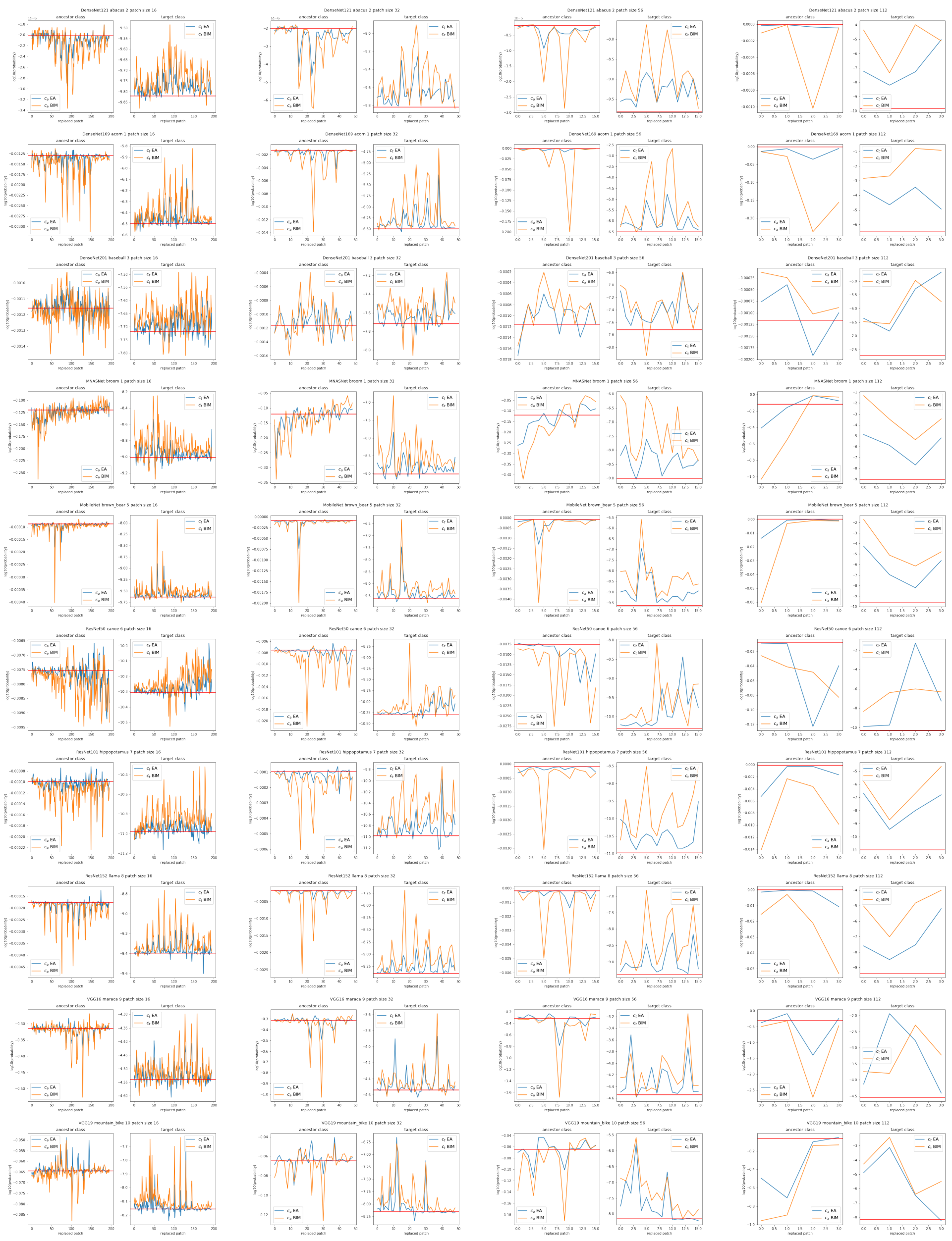


Figure 9.19: From the 1st row to the 10th row, single patch replacement for $(\mathcal{A}_1^2, \mathcal{C}_1)$, $(\mathcal{A}_2^1, \mathcal{C}_2)$, $(\mathcal{A}_3^3, \mathcal{C}_3)$, $(\mathcal{A}_4^1, \mathcal{C}_4)$, $(\mathcal{A}_5^5, \mathcal{C}_5)$, $(\mathcal{A}_6^6, \mathcal{C}_6)$, $(\mathcal{A}_7^7, \mathcal{C}_7)$, $(\mathcal{A}_8^8, \mathcal{C}_8)$, $(\mathcal{A}_9^9, \mathcal{C}_9)$, $(\mathcal{A}_{10}^{10}, \mathcal{C}_{10})$. From left to right, the 4 pairs of graphs correspond to patches of size 16×16, 32×32, 56×56 and 112×112, respectively. Each pair represents the step-wise plot of $log(o_I^{\mathcal{C}}[a])$ (left graph) and of $log(o_I^{\mathcal{C}}[t])$ (right graph) for the EA (blue curve) and BIM (orange curve). The red horizontal line recalls the c_a-label value (left graph) or the c_t-label value (right graph) of $\mathcal{A}$ with no replaced patch.

9.5.3 Adversarial noise visualization and frequency analysis

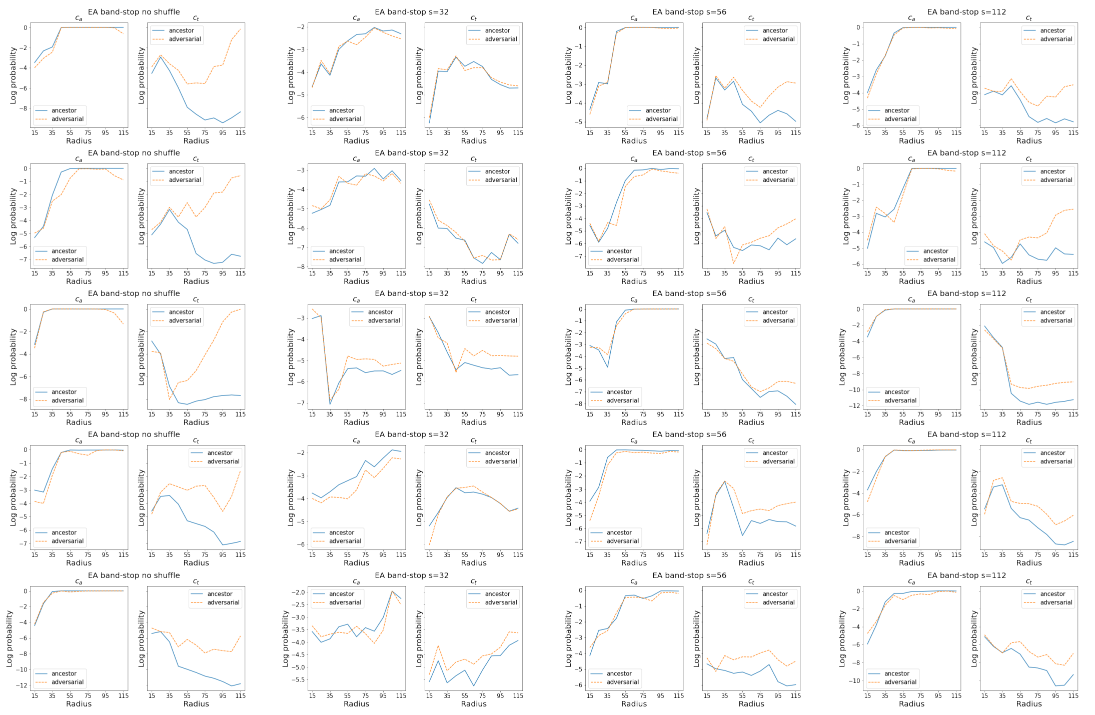


Figure 9.20: From the 1st row to the 5th row, band-stop graphs of the c_a and c_t probabilities for $(\mathcal{A}_1^2, \mathcal{C}_1)$, $(\mathcal{A}_2^1, \mathcal{C}_2)$, $(\mathcal{A}_3^3, \mathcal{C}_3)$, $(\mathcal{A}_4^1, \mathcal{C}_4)$, $(\mathcal{A}_5^5, \mathcal{C}_5)$. In each row, from left to right, the following images are fed to $\mathcal{C}_k$: $\mathcal{D}_k^{EA}(\mathcal{A}_q^p)$, $sh(\mathcal{D}_k^{EA}(\mathcal{A}_q^p), 32)$, $sh(\mathcal{D}_k^{EA}(\mathcal{A}_q^p), 56)$, and $sh(\mathcal{D}_k^{EA}(\mathcal{A}_q^p), 112)$, which are the unshuffled and shuffled versions of the adversarial image, and $\mathcal{A}_q^p$, $sh(\mathcal{A}_q^p, 32)$, $sh(\mathcal{A}_q^p, 56)$, and $sh(\mathcal{A}_q^p, 112)$, which are the unshuffled and shuffled versions of the ancestor.

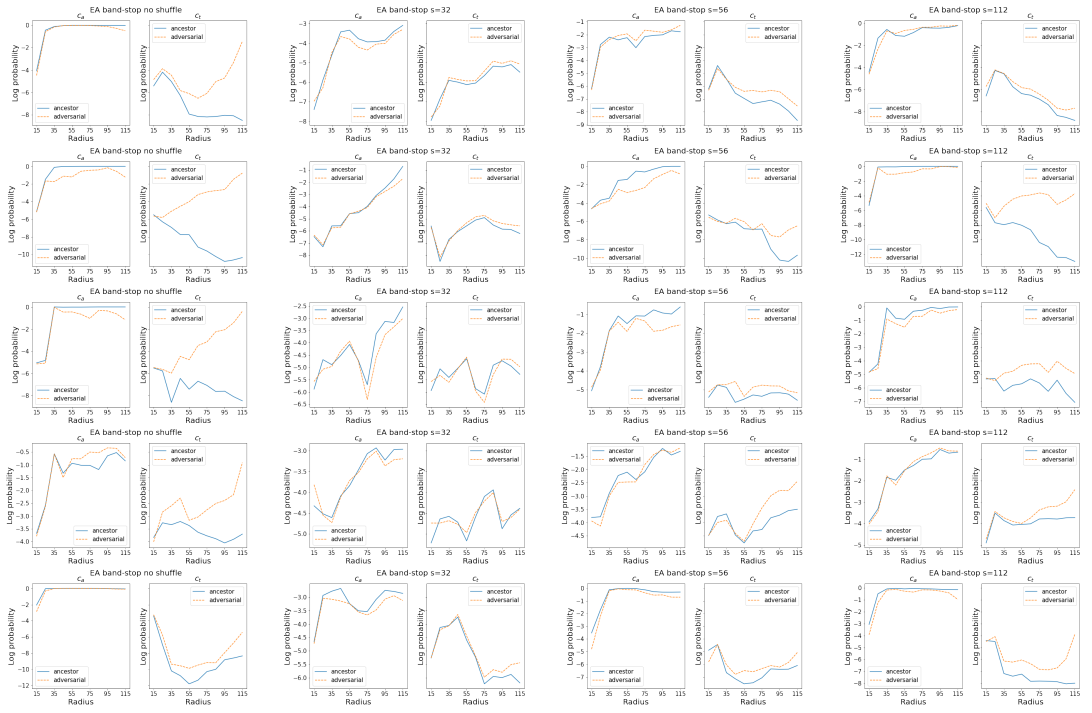


Figure 9.21: From the 1st row to the 5th row, band-stop graphs of the c_a and c_t probabilities for $(\mathcal{A}_6^6, \mathcal{C}_6)$, $(\mathcal{A}_7^7, \mathcal{C}_7)$, $(\mathcal{A}_8^8, \mathcal{C}_8)$, $(\mathcal{A}_9^9, \mathcal{C}_9)$, $(\mathcal{A}_{10}^{10}, \mathcal{C}_{10})$. In each row, from left to right, the following images are fed to C_k: $\mathcal{D}_k^{EA}(\mathcal{A}_q^p)$, $sh(\mathcal{D}_k^{EA}(\mathcal{A}_q^p), 32)$, $sh(\mathcal{D}_k^{EA}(\mathcal{A}_q^p), 56)$, and $sh(\mathcal{D}_k^{EA}(\mathcal{A}_q^p), 112)$, which are the unshuffled and shuffled versions of the adversarial image, and $\mathcal{A}_q^p$, $sh(\mathcal{A}_q^p, 32)$, $sh(\mathcal{A}_q^p, 56)$, and $sh(\mathcal{A}_q^p, 112)$, which are the unshuffled and shuffled versions of the ancestor.

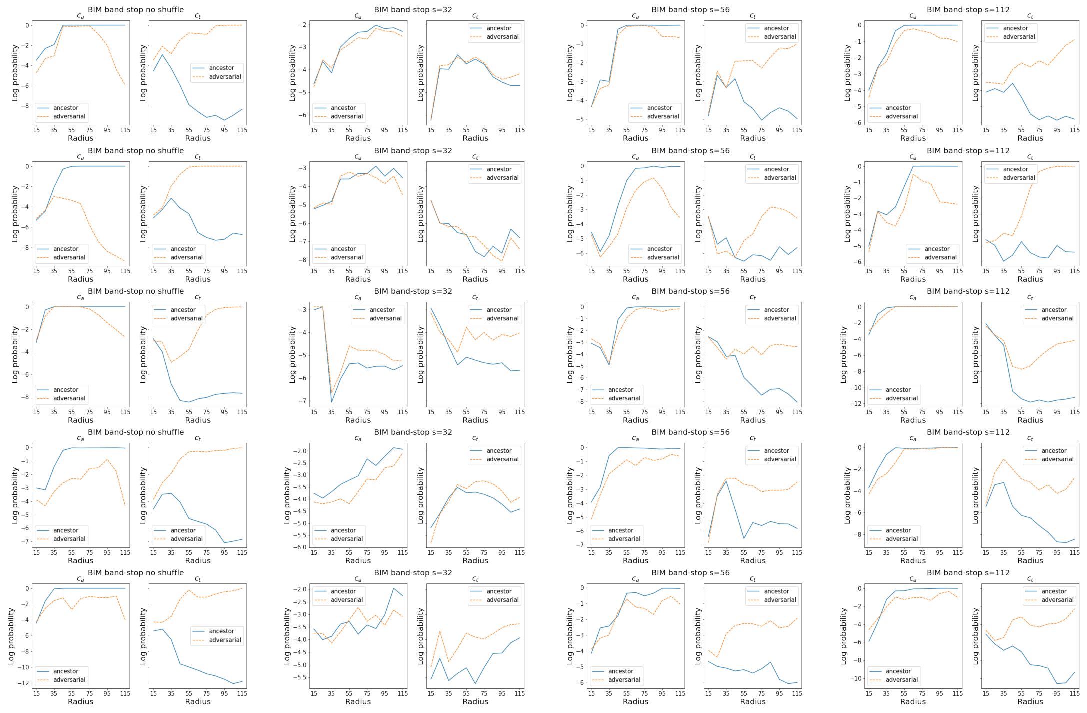


Figure 9.22: From the 1st row to the 5th row, band-stop graphs of the c_a and c_t probabilities for $(\mathcal{A}_1^2, \mathcal{C}_1)$, $(\mathcal{A}_2^1, \mathcal{C}_2)$, $(\mathcal{A}_3^3, \mathcal{C}_3)$, $(\mathcal{A}_4^1, \mathcal{C}_4)$, $(\mathcal{A}_5^5, \mathcal{C}_5)$. In each row, from left to right, the following images are fed to C_k: $\mathcal{D}_k^{BIM}(\mathcal{A}_q^p)$, $sh(\mathcal{D}_k^{BIM}(\mathcal{A}_q^p), 32)$, $sh(\mathcal{D}_k^{BIM}(\mathcal{A}_q^p), 56)$, and $sh(\mathcal{D}_k^{BIM}(\mathcal{A}_q^p), 112)$, which are the unshuffled and shuffled versions of the adversarial image, and $\mathcal{A}_q^p$, $sh(\mathcal{A}_q^p, 32)$, $sh(\mathcal{A}_q^p, 56)$, and $sh(\mathcal{A}_q^p, 112)$, which are the unshuffled and shuffled versions of the ancestor.

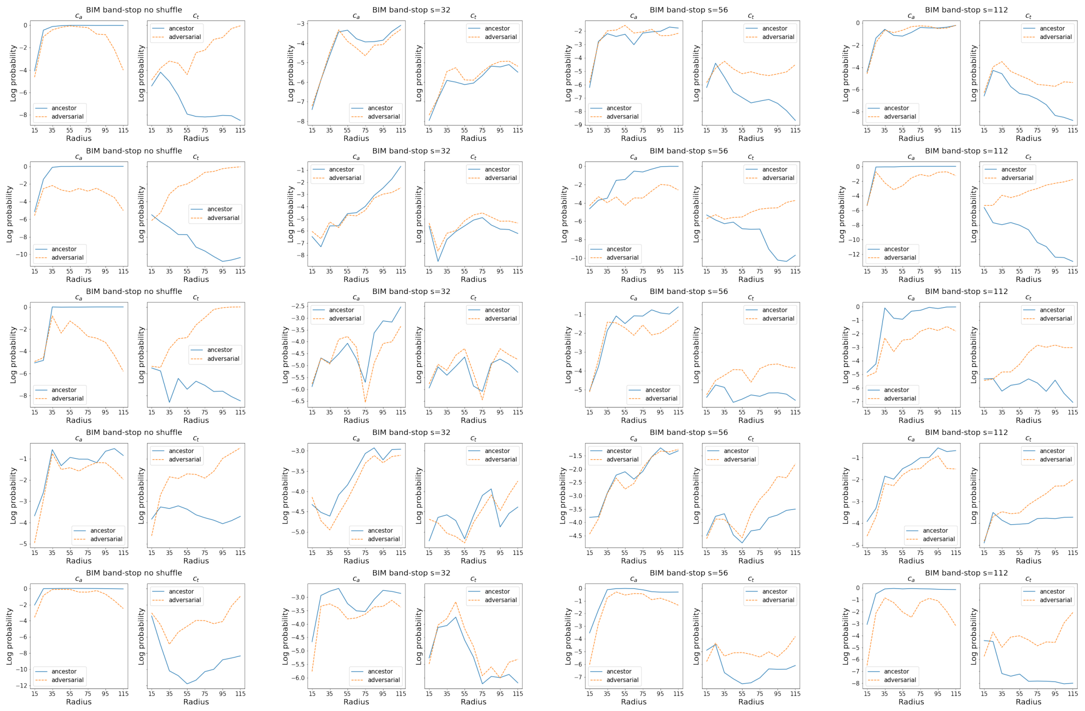
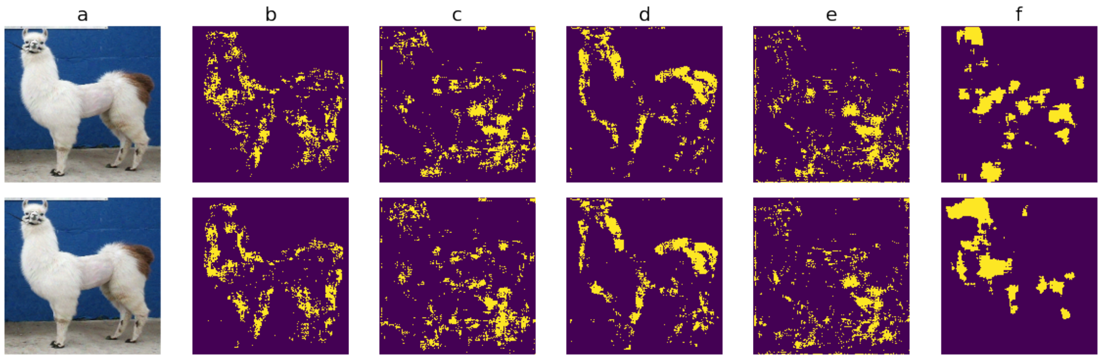
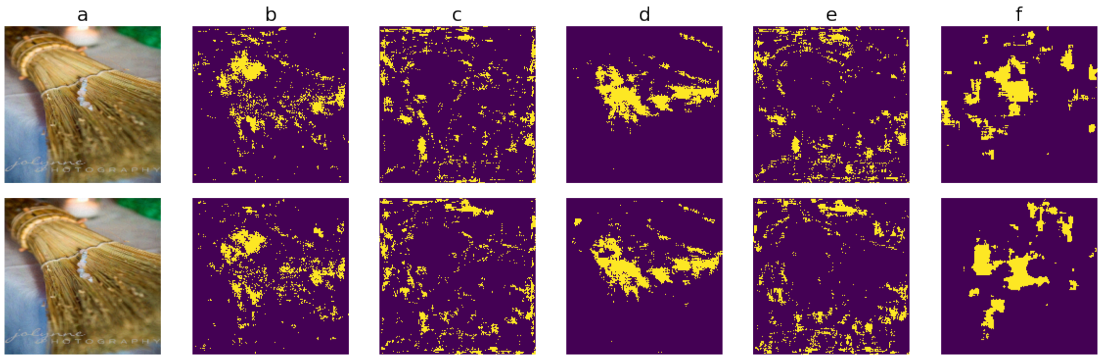


Figure 9.23: From the 1st row to the 5th row, band-stop graphs of the c_a and c_t probabilities for $(\mathcal{A}_6^6, \mathcal{C}_6)$, $(\mathcal{A}_7^7, \mathcal{C}_7)$, $(\mathcal{A}_8^8, \mathcal{C}_8)$, $(\mathcal{A}_9^9, \mathcal{C}_9)$, $(\mathcal{A}_{10}^{10}, \mathcal{C}_{10})$. In each row, from left to right, the following images are fed to $\mathcal{C}_k$: $\mathcal{D}_k^{BIM}(\mathcal{A}_q^p)$, $sh(\mathcal{D}_k^{BIM}(\mathcal{A}_q^p), 32)$, $sh(\mathcal{D}_k^{BIM}(\mathcal{A}_q^p), 56)$, and $sh(\mathcal{D}_k^{BIM}(\mathcal{A}_q^p), 112)$, which are the unshuffled and shuffled versions of the adversarial image, and $\mathcal{A}_q^p$, $sh(\mathcal{A}_q^p, 32)$, $sh(\mathcal{A}_q^p, 56)$, and $sh(\mathcal{A}_q^p, 112)$, which are the unshuffled and shuffled versions of the ancestor.

9.5.4 Transferability and texture bias



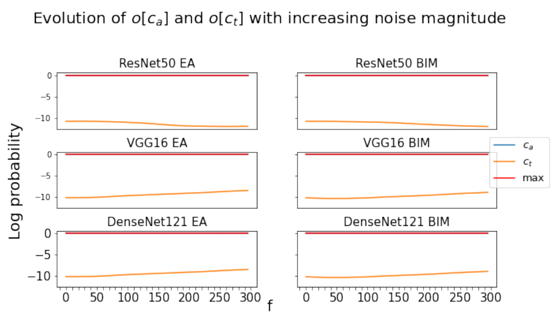
(a) $\mathcal{A}_8^2$



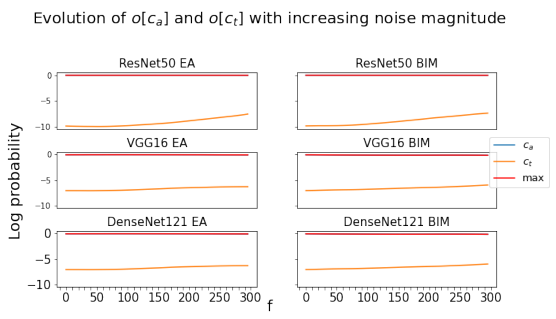
(b) $\mathcal{A}_4^1$

Figure 9.24: Heatmaps obtained with the ancestor $\mathcal{A}_8^2$ and the adversarial images $\mathcal{D}_6^{atk}(\mathcal{A}_8^2)$ (a) and with the ancestor $\mathcal{A}_4^1$ and the adversarial images $\mathcal{D}_6^{atk}(\mathcal{A}_4^1)$ (b). In each pair of rows, $atk =$ EA in the first row and $atk =$ BIM in the second.

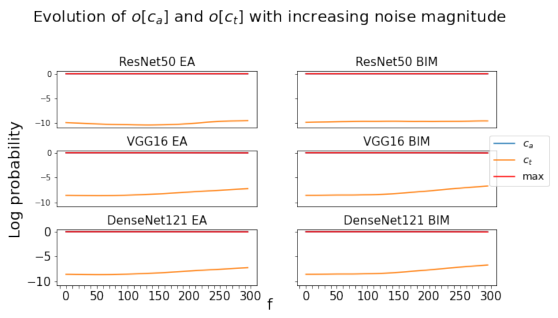
9.5.5 Effects of shuffling on the transferability of the adversarial images



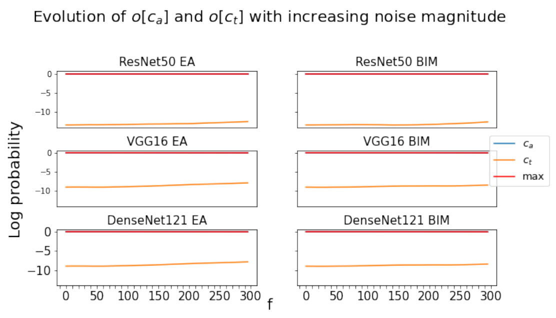
(a) no shuffle



(b) $s = 32$



(c) $s = 56$



(d) $s = 112$

Figure 9.25: Evolution of c_a and c_t for $\mathcal{A}_5^4$ (a), $sh(\mathcal{A}_5^4, 32)$ (b), $sh(\mathcal{A}_5^4, 56)$ (c) and $sh(\mathcal{A}_5^4, 112)$ (d) when fed to $\mathcal{C}_6$, $\mathcal{C}_9$ and $\mathcal{C}_1$ (1^{st}, 2^{nd} and 3^{rd} row of each set of graphs, respectively). In each set of graphs, the unshuffled or shuffled ancestor is perturbed with random normal noise created using the minimum and maximum noise magnitude of $\mathcal{D}_6^{EA}(\mathcal{A}_5^4)$ and $\mathcal{D}_6^{BIM}(\mathcal{A}_5^4)$. Along the x axis, the noise is attenuated or amplified by a factor f ($noise \times f$).

9.5.6 Layer activations

	For c_a	$W_{pos}\Delta_{pos}$	$W_{pos}\Delta_0$	$W_{pos}\Delta_{neg}$	$W_{neg}\Delta_{pos}$	$W_{neg}\Delta_0$	$W_{neg}\Delta_{neg}$
$\mathcal{C}_2$	DenseNet169	(48.59,48.24)	(0.22,0.09)	(51.19,51.67)	(53.70,53.52)	(0.47,0.26)	(45.82,46.22)
$\mathcal{C}_3$	DenseNet201	(50.97,49.21)	(0.25,0.16)	(48.79,50.63)	(54.32,54.86)	(0.61,0.24)	(45.07,44.90)
$\mathcal{C}_4$	MobileNet	(45.43,45.08)	(1.36,0.91)	(53.21,54.02)	(47.53,52.09)	(4.22,3.23)	(48.24,44.68)
$\mathcal{C}_5$	MNASNet	(42.80,43.82)	(11.93,10.19)	(45.27,45.99)	(40.21,43.33)	(19.96,17.97)	(39.83,38.70)
$\mathcal{C}_6$	ResNet50	(44.81,41.87)	(0.12,0.08)	(55.07,58.05)	(52.35,53.65)	(0.24,0.11)	(47.41,46.24)
$\mathcal{C}_7$	ResNet101	(48.00,47.81)	(0.06,0.02)	(51.94,52.16)	(53.37,56.61)	(0.38,0.23)	(46.26,43.16)
$\mathcal{C}_8$	ResNet152	(48.00,45.75)	(0.08,0.08)	(51.92,54.17)	(51.71,54.14)	(0.33,0.26)	(47.95,45.60)
$\mathcal{C}_9$	VGG16	(14.19,15.07)	(65.24,63.32)	(20.56,21.62)	(5.55,7.34)	(89.73,87.81)	(4.72,4.86)
$\mathcal{C}_{10}$	VGG19	(13.04,13.03)	(65.92,64.43)	(21.04,22.54)	(4.63,5.94)	(91.32,89.81)	(4.05,4.25)

Table 9.21: For c_a, average percentage of both positively-related (W_{pos}, columns 2-4) and negatively-related (W_{neg}, columns 5-7) units whose activation increased (Δ_{pos}), stagnated (Δ_0) or decreased (Δ_{neg}). In each row, the respective CNN is only fed with $\mathcal{C}_1$'s adversarial images $\mathcal{D}_1^{atk}(\mathcal{A}_q^p)$. Each cell contains the results for EA and BIM.

	For c_t	$W_{pos}\Delta_{pos}$	$W_{pos}\Delta_0$	$W_{pos}\Delta_{neg}$	$W_{neg}\Delta_{pos}$	$W_{neg}\Delta_0$	$W_{neg}\Delta_{neg}$
$\mathcal{C}_2$	DenseNet169	(53.72,54.18)	(0.32,0.17)	(45.96,45.65)	(49.59,48.76)	(0.41,0.20)	(50.00,51.04)
$\mathcal{C}_3$	DenseNet201	(55.22,54.95)	(0.44,0.20)	(44.34,44.85)	(51.00,50.17)	(0.44,0.22)	(48.56,49.61)
$\mathcal{C}_4$	MobileNet	(48.80,51.84)	(2.85,2.09)	(48.34,46.07)	(44.28,45.60)	(2.87,2.16)	(52.85,52.24)
$\mathcal{C}_5$	MNASNet	(43.38,45.68)	(15.02,13.30)	(41.60,41.01)	(39.79,41.72)	(17.19,15.14)	(43.02,43.14)
$\mathcal{C}_6$	ResNet50	(51.49,52.34)	(0.17,0.07)	(48.34,47.59)	(48.07,46.79)	(0.20,0.12)	(51.73,53.09)
$\mathcal{C}_7$	ResNet101	(54.85,56.81)	(0.29,0.14)	(44.86,43.04)	(48.71,50.76)	(0.22,0.14)	(51.07,49.10)
$\mathcal{C}_8$	ResNet152	(52.01,53.81)	(0.22,0.16)	(47.77,46.03)	(48.94,48.79)	(0.25,0.20)	(50.82,51.01)
$\mathcal{C}_9$	VGG16	(10.74,12.41)	(78.09,75.98)	(11.18,11.61)	(8.37,9.50)	(79.49,77.73)	(12.14,12.77)
$\mathcal{C}_{10}$	VGG19	(8.58,9.78)	(79.76,78.03)	(11.66,12.19)	(8.13,8.52)	(80.41,79.06)	(11.46,12.42)

Table 9.22: For c_t, average percentage of both positively-related (W_{pos}, columns 2-4) and negatively-related (W_{neg}, columns 5-7) units whose activation increased (Δ_{pos}), stagnated (Δ_0) or decreased (Δ_{neg}). In each row, the respective CNN is only fed with $\mathcal{C}_1$'s adversarial images $\mathcal{D}_1^{atk}(\mathcal{A}_q^p)$. Each cell contains the results for EA and BIM.

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., et al.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org

[2] Agrafiotis, D.: Chapter 9 - video error concealment. In: Theodoridis, S., Chellappa, R. (eds.) Academic Press Library in signal Processing, Academic Press Library in Signal Processing, vol. 5, pp. 295–321. Elsevier (2014). https://doi.org/https://doi.org/10.1016/B978-0-12-420149-1.00009-0, https://www.sciencedirect.com/science/article/pii/B9780124201491000090

[3] Andriushchenko, M., Croce, F., Flammarion, N., Hein, M.: Square attack: a query-efficient black-box adversarial attack via random search. In: European Conference on Computer Vision. pp. 484–501. Springer (2020)

[4] Archana, J.N., Aishwarya, P.: A review on the image sharpening algorithms using unsharp masking. IJESC **6** (2016), https://www.researchgate.net/publication/305985620_A_Review_on_the_Image_Sharpening_Algorithms_Using_Unsharp_Masking

[5] Bernard, N., Leprévost, F.: Evolutionary algorithms for convolutional neural network visualisation. In: High Performance Computing – 5th Latin American Conference, 2018 (Bucaramanga, Colombia, Sep 23-28, 2018). Communications in Computer and Information Science, vol. 979, pp. 18–32. Springer, Heidelberg (2018)

[6] Bernard, N., Leprévost, F.: How evolutionary algorithms and information hiding deceive machines and humans for image recognition: A research program. In: Proceedings of the OLA'2019 International Conference on Optimization and Learning (Bangkok, Thailand, Jan 29-31, 2019). pp. 12–15. Springer, Heidelberg (2019)

[7] Blier, L.: A brief report of the heuritech deep learning meetup#5 (2016), https://heuritech.wordpress.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/

[8] Brendel, W., Bethge, M.: Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet. CoRR **abs/1904.00760** (2019), http://arxiv.org/abs/1904.00760

[9] Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: Adversarial Attacks and Defences: A Survey. CoRR **abs/1810.00069** (2018), http://arxiv.org/abs/1810.00069

[10] Chen, P.Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.J.: ZOO. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. ACM (nov 2017), https://doi.org/10.1145%2F3128572.3140448

[11] Chitic, R., Bernard, N., Leprévost, F.: A proof of concept to deceive humans and machines at image classification with evolutionary algorithms. In: Intelligent Information and Database Systems, 12th Asian Conference, ACIIDS 2020 (Phuket, Thailand, March 23-26, 2020). pp. 467–480. Springer, Heidelberg (2020)

[12] Chitic, R., Deridder, N., Bernard, N., Leprévost, F.: Robustness of adversarial images against filters. In: Communications in Computer and Information Science, International Conference on Optimization and Learning (OLA) 2021. vol. 1443. Springer (2021)

[13] Chitic, R., Leprévost, F., Bernard, N.: Evolutionary algorithms deceive humans and machines at image classification: an extended proof of concept on two scenarios. Journal of Information and Telecommunication pp. 1–23 (2020)

[14] Chitic, R., Leprévost, F., Topal, A.O.: Empirical Perturbation Analysis of Two Adversarial Attacks: Black-box versus White-box. Submitted (2022)

[15] Chitic, R., Topal, A.O., Leprévost, F.: Evolutionary Algorithm-based images, humanly indistinguishable and adversarial against Convolutional Neural Networks: efficiency and filter robustness. IEEE Access **9** (2021), `https://ieeexplore.ieee.org/document/9627925`

[16] Chollet, F.: Keras. GitHub code repository (2015-2018), `https://github.com/fchollet/keras`

[17] Demush, R.: A brief history of computer vision (and convolutional neural networks) (2019), `https://hackernoon.com/a-brief-history-of-computer-vision-and-convolutional-neural-networks-8fe8aacc79f3`

[18] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: The imagenet image database (2009), `http://image-net.org`

[19] Duchon, C.E.: Lanczos filtering in one and two dimensions. Journal of Applied Meteorology and Climatology **18**(8), 1016–1022 (1979)

[20] Education, I.C.: Convolutional neural networks (2020), `https://www.ibm.com/cloud/learn/convolutional-neural-networks`

[21] Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer (2003), `https://www.springer.com/gp/book/9783642072857`

[22] Fawzi, A., Moosavi-Dezfooli, S., Frossard, P.: Robustness of classifiers: from adversarial to random noise. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016 (Barcelona, Spain, Dec 5-10, 2016). pp. 1624–1632 (2016), `http://papers.nips.cc/paper/6331-robustness-of-classifiers-from-adversarial-to-random-noise`

[23] Frieden, B.R.: A new restoring algorithm for the preferential enhancement of edge gradients. Journal of the Optical Society of America **66** (1976), `10.1117/12.954697`

[24] Geifman, Y.: cifar-vgg (2018), `https://github.com/geifmany/cifar-vgg`

[25] Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F., Brendel, W.: ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. CoRR **abs/1811.12231** (2018), `http://arxiv.org/abs/1811.12231`

[26] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and Harnessing Adversarial Examples. CoRR **abs/1810.00069** (2015), `http://arxiv.org/abs/1412.6572`

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

[28] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)

[29] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)

[30] Islam, M.A., Kowal, M., Esser, P., Jia, S., Ommer, B., Derpanis, K.G., Bruce, N.D.B.: Shape or texture: Understanding discriminative features in cnns. CoRR **abs/2101.11604** (2021), `https://arxiv.org/abs/2101.11604`

[31] Jason Jung, N.A., Hassan, G.M.: Analysing Adversarial Examples for Deep Learning . SciTePress (2021), `https://www.scitepress.org/Papers/2021/103137/103137.pdf`

[32] Jere, M., Hitaj, B., Ciocarlie, G.F., Koushanfar, F.: Scratch that! an evolution-based adversarial attack against neural networks. CoRR **abs/1912.02316** (2019), http://arxiv.org/abs/1912.02316

[33] Keys, R.: Cubic convolution interpolation for digital image processing. IEEE transactions on acoustics, speech, and signal processing **29**(6), 1153–1160 (1981)

[34] Krizhevsky, A., Nair, V., Hinton, G.: The CIFAR datasets (2009), https://www.cs.toronto.edu/~kriz/cifar.html

[35] Kullback, S., Leibler, R.: On information and sufficiency. The Annals of Mathematical Statistics **22**, 79–86 (1951)

[36] Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. CoRR **abs/1607.02533** (2016), http://arxiv.org/abs/1607.02533

[37] Lenc, K., Vedaldi, A.: Understanding image representations by measuring their equivariance and equivalence. CoRR **abs/1411.5908** (2014), http://arxiv.org/abs/1411.5908

[38] Leprévost, F., Topal, A.O., Avdusinovic, E., Raluca, C.: A Strategy creating High Resolution Adversarial Images against Convolutional Neural Networks, and a Feasibility Study on 10 CNNs. Submitted (2022)

[39] Lim, J.S.: Two-Dimensional Signal and Image Processing. Prentice Hall (1989)

[40] Liu, S., Deng, W.: Very deep convolutional neural network based image classification using small training sample size. IAPR (2015)

[41] Luo, W., Li, Y., Urtasun, R., Zemel, R.S.: Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. CoRR **abs/1701.04128** (2017), http://arxiv.org/abs/1701.04128

[42] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. CoRR **abs/1706.06083** (2019), http://arxiv.org/abs/1706.06083

[43] Mallawaarachchi, V.: Introduction to genetic algorithms — including example code (2017), https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3#:~:text=A%20genetic%20algorithm%20is%20a,offspring%20of%20the%20next%20generation.

[44] Morcosa, A.S., Raghu, M., Bengio, S.: Insights on representational similarity in neural networks with canonical correlation. CoRR **abs/1806.05759** (2018), https://arxiv.org/abs/1806.05759

[45] Oliphant, T.E.: A guide to NumPy. Trelgol Publishing USA (2006)

[46] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 372–387. IEEE (2016), https://ieeexplore.ieee.org/document/7467366

[47] Parsania, P.S., Virparia, P.V.: A comparative analysis of image interpolation algorithms. International Journal of Advanced Research in Computer and Communication Engineering **5**(1), 29–34 (2016)

[48] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703 (2019)

[49] Patel, V., Mistree, K.: A review on different image interpolation techniques for image enhancement. International Journal of Emerging Technology and Advanced Engineering **3**(12), 129–133 (2013)

[50] Pereira, A., Thomas, C.: Challenges of machine learning applied to safety-critical cyber-physical systems. MDPI Machine Learning and Knowledge Extraction (2020), https://www.mdpi.com/2504-4990/2/4/31

[51] Petra Vidnerová, R.N.: Vulnerability of classifiers to evolutionary generated adversarial examples. Neural Networks **127**, 168–181 (2020), https://www.sciencedirect.com/science/article/abs/pii/S0893608020301350

[52] Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. CoRR **abs/1312.6034** (2013), http://arxiv.org/abs/1312.6034

[53] Sinha, S., Garg, A., Larochelle, H.: Curriculum By Texture. CoRR **abs/2003.01367** (2020), https://arxiv.org/abs/2003.01367

[54] SpeedyGraphito: Mes 400 Coups. Panoramart (2020)

[55] Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation **23**(5), 828–841 (2019), https://ieeexplore.ieee.org/document/8601309

[56] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEECVF Conference on Computer Vision and Pattern Recognition. pp. 2820–2828 (2019)

[57] Topal, A.O., Chitic, R., Leprévost, F.: One evolutionary algorithm deceives humans and ten convolutional neural networks trained on ImageNet at image recognition. Submitted (2022)

[58] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. CoRR **abs/2012.12877** (2021), https://arxiv.org/abs/2012.12877

[59] Tutorialspoint: Genetic algorithms - introduction, https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_introduction.htm

[60] Van Rossum, G., Drake, F.L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA (2009)

[61] van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E., Yu, T., the scikit-image contributors: scikit-image: image processing in Python. PeerJ **2**, e453 (2014). https://doi.org/10.7717/peerj.453, https://doi.org/10.7717/peerj.453

[62] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing **13**(4), 600–612. (2004)

[63] Wu, J.: Generating adversarial examples in the harsh conditions. CoRR **abs/1908.11332** (2020), https://arxiv.org/abs/1908.11332

[64] Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J., Jain, A.K.: Adversarial attacks and defenses in images, graphs and text: A review. CoRR **abs/1909.08072** (2019), http://arxiv.org/abs/1909.08072

[65] Yin, D., Lopes, R.G., Shlens, J., Cubuk, E.D., Gilmer, J.: A Fourier Perspective on Model Robustness in Computer Vision. CoRR **abs/1906.08988** (2019), http://arxiv.org/abs/1906.08988

[66] Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8697–8710 (2018)