Applied Soft Computing 143 (2023) 110397

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

One evolutionary algorithm deceives humans and ten convolutional neural networks trained on ImageNet at image recognition

Ali Osman Topal *, Raluca Chitic, Franck Leprévost

University of Luxembourg, House of Numbers, 6, Avenue de la Fonte, L-4364 Esch-sur-Alzette, Grand Duchy of Luxembourg

ARTICLE INFO

Article history:
Received 25 February 2022
Received in revised form 29 March 2023
Accepted 1 May 2023
Available online 11 May 2023

Keywords:
Adversarial attacks
Black-box attacks
Convolutional neural network
Evolutionary algorithm
Image classification

ABSTRACT

Convolutional neural networks (CNNs) are widely used in computer vision, but can be deceived by carefully crafted adversarial images. In this paper, we propose an evolutionary algorithm (EA) based adversarial attack against CNNs trained on ImageNet. Our EA-based attack aims to generate adversarial images that not only achieve a high confidence probability of being classified into the target category (at least 75%), but also appear indistinguishable to the human eye in a black-box setting. These constraints are implemented to simulate a realistic adversarial attack scenario. Our attack has been thoroughly evaluated on 10 CNNs in various attack scenarios, including high-confidence targeted, good-enough targeted, and untargeted. Furthermore, we have compared our attack favorably against other well-known white-box and black-box attacks. The experimental results revealed that the proposed EA-based attack is superior or on par with its competitors in terms of the success rate and the visual quality of the adversarial images produced.

1. Introduction

Convolutional neural networks (CNNs) trained on large sets of examples have recently become the dominant tool for computer vision tasks [1–5]. CNN is a type of deep learning algorithm that is composed of various layers: convolution layer, pooling layer, and fully connected layer. The convolution layer processes the input data using a convolution operation, which enables the network to identify different features and their arrangement in the input data. The pooling layer down-samples the output of the convolution layers by reducing the spatial resolution while increasing the abstraction of the learned features. The fully connected layers allow the network to create a complex decision boundary for the input data by connecting all the neurons from one layer to the next, which enables non-linear decision-making [6].

The innovation of CNNs lies in their ability to extract features from input data through a training process called back-propagation. Under this process, early convolutional layers extract simple features, such as edges, whereas later layers extract more complex semantic ones [7]. This technique empowers CNNs to detect specific features anywhere in the input images and classify the image in a given category.

However, CNNs, like other machine learning models, can be vulnerable to adversarial attacks, which are attacks specifically designed to trick the model into making incorrect predictions. Adversarial attacks on CNNs can take the form of adversarial images, which are images that have been manipulated in a way that is imperceptible to a human but that cause the CNN to make an incorrect prediction. Adversarial attacks can be a serious problem if the CNN is used in a safety-critical application, such as self-driving cars or medical diagnosis [8–11].

Such attacks are classified as *white box* or *black box* attacks, depending on the amount of information available to the attacker. In the former case, an attacker has access to the CNN's architecture and parameters, whereas no prior knowledge of any kind is assumed in the latter case. Depending on their objectives, these attacks can be either *targeted* or *untargeted*. Starting from an image $\mathcal{A}$ classified by a CNN in a category c_a, an image $\mathcal{D}$ is adversarial for the *target scenario* if the CNN classifies it in a predefined target category $c_t \neq c_a$. $\mathcal{D}$ is adversarial for the *untargeted scenario* if the CNN classifies it in a category $c \neq c_a$, without requiring anything for c beyond being distinct from c_a. However, in both scenarios, $\mathcal{D}$ is expected to remain visually close to $\mathcal{A}$ for a human eye.

Over the past few years, the prevalent use of CNNs has led to a growing interest in their robustness/stability. Christian et al. [11] were the first to discover that applying a specific, barely perceptible perturbation to an image can cause the network to misclassify the image. After this observation, many researchers focused on generating adversarial attacks to challenge the robustness/stability of CNNs. Papernot et al. [12] proposed an algorithm based on forward derivative computation. They constructed negative saliency maps that identify the features of the input that

* Corresponding author.
E-mail address: aliosman.topal@uni.lu (A.O. Topal).

https://doi.org/10.1016/j.asoc.2023.110397

Fig. 1. The images in the first row represent original images and in the second row the corresponding adversarial images and their respective class labels that are created by (a) One-Pixel attack [13], (b) Few-Pixels attack [12], (c) Fooling Transfer Net (FTN) [14], (d) Scratch that! [15], and (e) our EA-based attack [10,16,17].

most affect the classification of the output. Thus, by changing only a few pixels (about 4%) of an input image, they produced adversarial images. But the perturbations were obvious to a human eye (see Fig. 1(a)). Later, Su et al. [13] generated a targeted black-box attack using the differential evolution algorithm that perturbed a single pixel of the image, resulting in the misclassification of the target CNN. However, the confidence level of the adversarial images was too low (23%) for images of size 224 × 224 and the perturbation was obvious to the human eye (see Fig. 1(b)). Wu [14] then raised the bar and developed a universal mapping to map inputs to adversarial images. These images can deceive CNNs to classify them all into a target class, and they also have strong transferability. However, Wu's attack has similar problems to [12,13] in terms of noticeable perturbations on the adversarial images (see Fig. 1(c)). Similar to [13], Malhar et al. [15] developed a black-box attack in a constrained threat model using differential evolution algorithms. Unlike the one-pixel and few-pixels attacks, Malhar's attacks can modify a small portion of pixels in the form of scratches on an image. Even though they managed to fool the CNNs, the perturbations were still noticeable to a human eye (see Fig. 1(d)).

Given a CNN $\mathcal{C}$ trained on ImageNet, this paper describes an evolutionary algorithm $\text{EA}^{\text{target},\mathcal{C}}$, that amounts to a generic, (mainly) targeted, black-box EA-based attack, with the additional strong requirement that the obtained adversarial images should not only be very close, but should actually be indistinguishable from the original to a human eye.

In our previous studies [10,16,17], we showed that our EA-based attack can generate adversarial images for the targeted scenario against a particular CNN, VGG16 trained on CIFAR-10 to sort images of size 32 × 32 into 10 categories. The adversarial image pictured in Fig. 1(e) was obtained that way, where the added noise remained "undercover".

The present study substantially scales up this challenge. Indeed, $\text{EA}^{\text{target},\mathcal{C}}$ attacks 10 stable, diverse, state-of-the-art CNNs trained on the ImageNet dataset, namely, DenseNet121, DenseNet169, DenseNet201, MobileNet, NASNetMobile, ResNet50, ResNet101, ResNet152, VGG16, and VGG19. The algorithm now deals with images of size 244 × 244, hence 49 times larger than those of the CIFAR-10 case. Furthermore, for the *target scenario*, we require $\text{EA}^{\text{target},\mathcal{C}}$ to produce adversarial images that these CNNs label as belonging to the target category with a minimum confidence of 0.75 (which we refer to as a 0.75-*strong adversarial image* in this paper).

We also evaluated our attack by comparing it to six well-known attacks, including four white-box attacks (FGSM, BIM, PGD Inf, and PGD L2) and two black-box attacks (SimBA and AdvGAN). The attacks are compared in targeted and untargeted scenarios.

Our contributions are summarized as follows.

- Our work provides a new perspective on adversarial attacks in black-box settings and shows the potential of evolutionary algorithms in this field.
- We show that our attack can successfully generate adversarial images with high confidence and high visual quality against 10 state-of-the-art CNNs trained on the ImageNet dataset.
- The effectiveness of six prominent adversarial attacks is thoroughly evaluated using 10 CNNs trained on the ImageNet dataset. Such a level of evaluation was not provided before, even not in the attack's original papers.
- We show that our EA-based attack compares favorably with these six attacks.

The remainder of this paper is organized as follows. Section 2 describes the criteria an image should fulfill to be considered adversarial for a CNN. Section 3 summarizes the main conceptual features of our evolutionary algorithm $\text{EA}^{\text{target},\mathcal{C}}$. Substantial experiments are performed to fine-tune the parameters of the EA: the population size, crossover range, and termination condition. Section 4 specifies our evaluation methodology and the results are summarized in Section 5. Section 6 provides the comparison of our attack with six well-known attacks. After the concluding Section 7, that essentially wraps up the results of this study, two Annexes with additional data (ancestor images, classification by the CNNs with label values, detailed success rates, and samples of adversarial images) complete the paper. Annex A deals with ancestor images, and Annex B with adversarial images. The code leading to these results is available under the link: "https://github.com/aliotopal/EAbasedAttack.git"

2. Adversarial image requisites

Let $\mathcal{C}_1, \ldots, \mathcal{C}_x$ be a series of different CNNs trained on the ImageNet [18] dataset to label images into 1000 categories $c_1, \ldots, c_{1000}$. To express the *target* scenario on these x CNNs, one first considers an *ancestor* image $\mathcal{A}_a$, that a human and all $\mathcal{C}_1, \ldots, \mathcal{C}_x$ label as belonging to the same ancestor category c_a. Concretely, if $\mathbf{o}^{\mathcal{C}}_{\mathcal{I}} = (\mathbf{o}^{\mathcal{C}}_{\mathcal{I}}[1], \ldots, \mathbf{o}^{\mathcal{C}}_{\mathcal{I}}[1000])$ denotes the output vector produced by CNN $\mathcal{C}$ (trained on ImageNet) when fed with image $\mathcal{I}$, one defines a as :

$$a = \arg\max_{1\le j\le 1000}\left(\mathbf{o}^{\mathcal{C}_1}_{\mathcal{A}_a}\right) = \cdots = \arg\max_{1\le j\le 1000}\left(\mathbf{o}^{\mathcal{C}_x}_{\mathcal{A}_a}\right). \quad (1)$$

Then, one chooses a common target category $t \neq a$ (hence $c_t \neq c_a$) among the categories of ImageNet. Finally, one derives from $\mathcal{A}_a$ a series of x images $\mathcal{D}^1_{a,t}(\mathcal{A}_a), \ldots, \mathcal{D}^x_{a,t}(\mathcal{A}_a)$, where each $\mathcal{D}^k_{a,t}(\mathcal{A}_a)$ is *adversarial* for $\mathcal{C}_k$ in the following sense.

On the one hand, one requires that $\mathcal{C}_k$ classifies $\mathcal{D}^k_{a,t}(\mathcal{A}_a)$ as belonging to c_t; namely, Eq. (2) holds:

$$t = \arg\max_{1\le j\le 1000}\left(\mathbf{o}^{\mathcal{C}_k}_{\mathcal{D}^k_{a,t}(\mathcal{A}_a)}\right). \quad (2)$$

Although this condition is often considered sufficient, we additionally require that the c_t-label values of the output vectors of the considered CNNs satisfy:

$$\boldsymbol{o}^{\mathcal{C}_k}_{\mathcal{D}^k_{a,t}(\mathcal{A}_a)}[t] \geq \tau, \tag{3}$$

for τ a fixed constant threshold value $\in [0, 1]$, common to all CNNs. The closer τ is to 1, the higher the confidence with which $\mathcal{C}_k$ classifies $\mathcal{D}^k_{a,t}(\mathcal{A}_a)$ in the target category c_t. Indeed, since the second best label value is necessarily $< 1 - \tau$, the margin of confidence between the best and the second best label values is assessed by the difference $\tau - (1-\tau) = 2\tau - 1$ (a choice of $\tau > 0.5$ clearly ensures that this quantity is > 0).

On the other hand, the adversarial $\mathcal{D}^k_{a,t}(\mathcal{A}_a)$ should remain so close to the ancestor $\mathcal{A}_a$ that a human would not be able to distinguish between both images. The visual difference between the two images is assessed thanks to a convenient metric d. In practice, for d, we shall take the L_2-distance [19,20] (Sections 3.2, 5; note that other distances are conceivable, such as SSIM [21]).

Eqs. (2) and (3) lead us to define the concepts of a *good enough adversarial image* and of a *τ-strong adversarial image*, respectively. A *good enough adversarial image* is an adversarial image that the CNN classifies as belonging to the target category c_t, without any requirement on the c_t-label value beyond being strictly dominant among all label values. A *τ-strong adversarial image* is an adversarial image that the CNN not only classifies as belonging to the target category c_t, but for which its c_t-label value $\geq \tau$ (for τ is a sufficiently convincing threshold value).

Hence, the *target attack* attempts to create *good enough adversarial images* or *τ-strong adversarial images* $\mathcal{D}^k_{a,t}(\mathcal{A}_a)$ such that the distance $d(\mathcal{D}^k_{a,t}(\mathcal{A}_a), \mathcal{A}_a)$ remains small.

In practice, we set $\tau = 0.75$ in our experiments (Sections 3.2, 5). Indeed, this choice ensures a high confidence in the classification of the CNNs, since it provides a margin of confidence $\geq$0.50 with respect to the next best label value (necessarily $\leq$0.25).

Let us complete this section by stating the *untargeted attack*. Starting from $\mathcal{A}_a$, this attack aims to create an image $\mathcal{D}(\mathcal{A}_a)$, which is still close enough to $\mathcal{A}_a$ as measured by $d(\mathcal{D}(\mathcal{A}_a), \mathcal{A}_a)$, that the CNN classifies in a category $c \neq c_a$. The *untargeted attack* does not require anything from c, as long as it differs from c_a.

In particular, an *untargeted attack* is usually easier to perform than a *targeted attack*.

3. Design of EA$^{\textbf{target},\mathcal{C}}$ and choice of the population size

The purpose of this section is twofold. First, we present the key features of our evolutionary algorithm EA$^{\text{target},\mathcal{C}}$, whose aim is to construct adversarial images for the *target scenario* against a CNN $\mathcal{C}$ trained on a dataset, and we briefly highlight how these features differ from classical EAs [22]. Second, we describe a series of tests performed on our EA using different population sizes. The outcome leads to the selection of a convenient population size for the challenges addressed in Sections 4 and 5.

3.1. Key features of EA$^{target,\mathcal{C}}$

Classical EAs [22] begin with an initial population of N randomly generated individuals. The initial population of our algorithm, EA$^{\text{target},\mathcal{C}}$, already differs in this regard because it is composed of N identical individuals. Thanks to an appropriate fitness function, classical EAs proceed to the search for better individuals, generation after generation, through reproduction, cross-overs, and mutations. This is also the case for EA$^{\text{target},\mathcal{C}}$ with specifically designed mutations and cross-overs. The details of these adjustments and the results of comparison tests between (a variant of) EA$^{\text{target},\mathcal{C}}$ and classical EAs are given in [10]. The pseudocode of EA$^{\text{target},\mathcal{C}}$ is presented in Algorithm 1, and its main components are described below. Throughout the algorithm (and in the paper), we assume that N is an even integer.

Algorithm 1 EA$^{\text{target},\mathcal{C}}$ with population size = N

```
1:  BEGIN
2:      t = 0
3:      INITIALISE population P(t = 0) = A_a × N;
4:      EVALUATE P(t = 0);
5:      while isNotTerminated() do
6:          SELECT:
7:              P_e(t) = P(t).selectElites(N_e = 10); / elites
8:              P_w(t) = P(t).selectWorsts(N/2); / "didn't make it"
9:              Pm(t) = P(t) − (P_e(t) ∪ P_w(t)); /middle-class
10:         RECOMBINE / MUTATE:
11:             P_keep(t) = (P_m(t) ∪ P_w(t)).randomSelect(N/2 − N_e) ∪ P_e(t);
12:             mutate(P_keep(t) ∪ P_m(t));
13:             P_c(t) = reproduction(P_keep(t), P_m(t));
14:         EVALUATE:
15:             P(t + 1) = evaluate(P_e(t), P_c(t));
16:             t = t + 1
17:     end
18: END
```

Goal of EA$^{\textbf{target},\mathcal{C}}$.— The EA is instantiated on the following challenge for $\mathcal{C}$ a given CNN trained on ImageNet. Given a couple (c_a, c_t) of ancestor and target categories, and given an ancestor image $\mathcal{A}_a$ that $\mathcal{C}$ classifies as belonging to c_a, the purpose of the EA is to construct an adversarial image for the *target scenario* described in Section 2.

Termination condition.— The termination condition depends on whether one focuses on *good enough adversarial images* or on *τ-strong adversarial images*.

In the former case, the algorithm stops if it creates a *good enough adversarial image* in less than a fixed number X of generations (for X, the maximum number of generations to be defined). In this case, the attack is considered to be *successful*, and the first corresponding *good enough adversarial image* is stored.

In the latter case, the algorithm stops if it creates a *τ-strong adversarial image* in less than X generations. Similarly, in this case, the attack is considered *successful*, and the first corresponding *τ-strong adversarial image* is stored.

The algorithm stops after X generations anyhow, even if it does not succeed in producing a *good enough adversarial image* in the former case, or a *τ-strong adversarial image* in the latter case. Accordingly, the attack is considered *unsuccessful*.

Population initialization.— The population size N is fixed and remains the same, generation after generation. The initial population is set to N copies of the ancestor $\mathcal{A}_a$.

Fitness function and evaluation.— Each individual image *ind* of the population of a given generation is evaluated using a fitness function *fit*(*ind*) (line 2–4 and 14–15 in Algorithm 1). This fitness function, representing the objective of the evolution of *ind* towards an image classified by $\mathcal{C}$ as belonging to the target category c_t, is defined as:

$$fit(ind) = \boldsymbol{o}^{\mathcal{C}}_{ind}[t]. \tag{4}$$

The higher an individual's fitness value, the fitter this individual.

Selection, Mutation, and Cross-over.— The different steps of the evolution from a population $P(t)$ of a given generation to the next one $P(t+1)$, namely selection, mutations, and cross-overs, as summarized below, are similar to those detailed in [23].

The individuals composing population $P(t)$ of a given generation are evaluated using the fitness function. Its N population members are sorted into three groups, depending on their respective scores.

–*Selection:* The elite P_e is composed of the $N_e = 10$ individuals with the best fitness values. These individuals are moved unchanged to the next generation (meaning that their members do not undergo mutations or cross-overs). The lower class P_w represents half of the total population. It is composed of the $(N/2)$ individuals with the worst fitness values. The remaining $N/2 - N_e$ individuals constitute the middle class P_m (line 6–9 in Algorithm 1). These classes are then prepared for the evolution step. The lower class P_w is discarded and replaced with a newly created "keep" group of $N/2$ individuals. The "keep" group P_{keep} is composed of the P_e, and of $(N/2) - N_e$ random individuals from the P_w and P_m classes from $P(t)$ (line 11 in Algorithm 1)

–*Mutation:* Mutation is applied to the "keep" and "middle class" group to increase the exploration capability of the algorithm (line 12 in Algorithm 1). Pixel mutations are performed by randomly choosing (with a power law) the number of pixels to mutate, their location, and the channel (R, G, or B) to be modified. The corresponding value of this channel is modified by $+1$ or -1. The choice of such small variations is intended to minimize the modifications applied to the running image. Indeed, we want to reduce the ultimate L_2-distance between the adversarial and the ancestor images so that a human eye would be unlikely to notice any difference between the two.

–*Crossover:* Crossovers are essentially obtained by swapping a rectangular area at a uniformly random location between two individuals. The size of the rectangle, called crossover size in our case, is specified by its height and width in form of integers chosen uniformly at random in the range [1, 30] which is called crossover range in this paper. Crossovers are performed on a single channel, chosen randomly.

Finally, the individuals from the elite, together with these crossed-over individuals, form the population $P(t+1)$ of the new generation.

These steps (evaluation, selection, mutation, cross-overs) are repeated generation for generation until the termination condition is satisfied.

–*The time complexity of the algorithm:* The time complexity of evolutionary algorithms is commonly measured through the number of generations or evaluations of the fitness function. Factors such as the population size and the intricacy of the fitness function can also impact the complexity. The time complexity of these algorithms can vary, potentially being exponential, polynomial, or even constant, depending on the specific problem and implementation [24]. Our algorithm has a time complexity of O(ngm), where n is the size of the input (i.e. $224 \times 224 \times 3$ for ImageNet images), g is the number of generations, and m is the population size. The size of the input image greatly impacts the computation time of our algorithm.

For future references (Sections 4 and 5), note that even if the algorithm focuses on achieving τ*-strong adversarial images*, on the way one may store intermediate relevant information regarding the occurrences where the algorithm already produces *good enough adversarial images*, or even *adversarial images for the untargeted attack*.

3.2. Population and crossover sizes: tests design

In this subsection, we design a series of tests for $\text{EA}^{\text{target},\mathcal{C}}$ with different population and crossover sizes.

The choice of population and crossover sizes can have a significant impact on the performance of an evolutionary algorithm. In general, a larger population size can lead to a greater diversity of solutions, which can help to prevent the algorithm from getting stuck in a local optimum. However, a larger population size also requires more computation time and memory, and so there is often a trade-off between population size and computational efficiency. A larger crossover size typically results in greater exploration of the search space, which can help the algorithm to escape from local optima. However, a larger crossover size also increases the chance of introducing undesirable genetic material into the population, which can lead to a decrease in the overall quality of the population [25]. The optimal population and crossover sizes may be different for different problem, so, it requires some trial and error to find the best values for population and crossover sizes.

Indeed, in [10,17], we constructed an EA (a variant of $\text{EA}^{\text{target},\mathcal{C}}$) that successfully fooled VGG-16 trained on the CIFAR-10 dataset. Starting with ancestor images of size 32×32, we found that $N = 160$ and the crossover range [1, 10] provide the best trade-off between the effective construction of adversarial images on the one hand, and the computational time and number of generations required to do so on the other.

The situation differs here because we attack CNNs trained on the ImageNet dataset. The ancestor images are now of size 224×224 (usually; they are sometimes even larger before being processed to fit the CNNs' constraints). Said otherwise, starting with ancestor images of ImageNet size, $\text{EA}^{\text{target},\mathcal{C}}$ must deal with a search space that is 49 times 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 requires adjusting the population size and crossover size of the EA accordingly.

To find the optimal population size N_{opt} and crossover range for the threshold value $\tau = 0.75$ and the maximum number of generations $X = 10{,}000$ (the choices of τ and X are consistent with the experiments performed in Section 5), we first run $\text{EA}^{\text{target},\mathcal{C}}$ with $N = 40, 80, 120$, and 160 for $\mathcal{C}$ =VGG-16 trained on the ImageNet dataset for a series of combinations (c_a, c_t) with fixed crossover range [0, 30] and find N_{opt}. Then we performed the same experiment with different crossover ranges: [0, 10], [0, 20], [0, 30], [0, 56], and [0, 112] with N_{opt}.

More precisely, we randomly choose 5 pairs (c_{a_k}, c_{t_k}) $(1 \leq k \leq 5)$ of ancestor and target categories, and pick an ancestor image $\mathcal{A}_{a_k}$ in c_{a_k} (see Table 1). 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 in seconds, average number of generations, average time/generation, and average L_2-distance between the ancestor image and the adversarial images. Then we repeat the same experiments for each crossover range. Computations were performed on nodes with Nvidia Tesla V100 GPGPUs of the IRIS HPC Cluster at the University of Luxembourg [26].

3.3. Population size tuning: results and interpretation

Table 2 summarizes the results of the tests of Section 3.2. Experiments show that all runs successfully created 0.75-strong adversarial images in less than 6000 generations. Therefore, to be on the safe side in the experiments performed in this paper, we set the maximum number of generations to 10,000 (see Section 5).

Table 3 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.

Table 1
For $1 \leq k \leq 5$, the ancestor image $\mathcal{A}_{a_k}$, taken from the ImageNet test set, classified by VGG-16 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}.

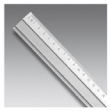
k	1	2	3	4	5
(c_{a_k}, a_k)	(hippo, 344)	(red wine, 966)	(frying pan, 567)	(armadillo, 363)	(ruler, 769)
$\mathcal{A}_{a_k}$					
c_{a_k}-label value	0.6900	0.5948	0.9999	1.0000	0.9696
(c_{t_k}, t_k)	(gibbon, 368)	(banjo, 420)	(printer, 742)	(saluki, 176)	(junco,13)

Table 2
Performance comparison of $\text{EA}^{\text{target},\mathcal{C}}$ for $\mathcal{C}$ = VGG-16 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 1 over 10 independent runs for each population size.

N	$\text{avgTime}^{0.75}_{\text{VGG16}}$	$\text{avgGens}^{0.75}_{\text{VGG16}}$	Time/gen	$\text{avg}^{0.75}_{\text{VGG16}}L_2$
40	840 s	2957	0.283	3220
80	1747 s	2551	0.686	3091
120	2434 s	2355	1.039	3029
160	3095 s	2256	1.382	2983

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.

With the machines and libraries specified in Section 5 (and used for all experiments in this study), the algorithm with $N = 40$ completes a generation in 0.283 s on average, almost five times (exactly 4.88) faster than with $N = 160$, and terminates within 840 s on average; hence, it is more than 3.68 times faster than with $N = 160$. This speed gain (per generation and altogether) significantly compensates for the 31% increase in the number of generations required with $N = 40$ as compared with $N = 160$.

3.4. Crossover size tuning: results and interpretation

Table 4 summarizes the results of the tests of Section 3.2. Experiments show that all runs successfully created 0.75-strong adversarial images in less than 6000 generations.

The findings demonstrate that the effectiveness of $\text{EA}^{\text{target},\mathcal{C}}$ is enhanced when the crossover range is set to [0,30]. However, it should be noted that the impact of the crossover range on $\text{EA}^{\text{target},\mathcal{C}}$'s performance is minimal, as the number of generations, average time (avgTime), and L_2-distances are relatively similar.

Conclusion.– A population size of $N = 40$ and crossover range of [0, 30] provide an appropriate choice for $\text{EA}^{\text{target},\mathcal{C}}$ against $\mathcal{C}$ = VGG-16 trained on ImageNet. We more generally extrapolate these choices of $N = 40$ and crossover range [0, 30] for $\text{EA}^{\text{target},\mathcal{C}}$ against any CNN trained on ImageNet. Therefore, we use $N = 40$ and crossover range [0, 30] throughout the rest of this paper.

4. One EA versus 10 CNNs: Methodology

The generic methodology used in 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 5, which concretely evaluates the efficiency of $\text{EA}^{\text{target},\mathcal{C}}$ at generating 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*) within 10,000 generations.

Specifically, Section 4.1 lists the 10 CNNs trained on ImageNet that we intend to challenge with our EA, and provides the rationale that led to their choice. Section 4.2 explains how we obtained the (ancestor, target) category pairs, and the ancestor images. Section 4.3 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.

4.1. Network domain

We challenge $\text{EA}^{\text{target},\mathcal{C}}$ against the following 10 CNNs trained on ImageNet: $\mathcal{C}_1$ = DenseNet121 [3], $\mathcal{C}_2$ = DenseNet169 [3], $\mathcal{C}_3$ = DenseNet201 [3], $\mathcal{C}_4$ = MobileNet [27], $\mathcal{C}_5$ = NASNetMobile [28], $\mathcal{C}_6$ = ResNet50 [5], $\mathcal{C}_7$ = ResNet101 [5], $\mathcal{C}_8$ = ResNet152 [5], $\mathcal{C}_9$ = VGG16 [4], and $\mathcal{C}_{10}$ = VGG19 [4].

These 10 CNNs were chosen for the following reasons: First, due to implementation considerations, we only considered CNNs that have an ImageNet pre-trained version already available in Keras [29]. To date, 26 CNNs satisfy this criterion. They are listed in Table 5.

Out of them, 15 handle images of size 224 × 224, while 11 handle images of larger sizes, varying from 240 × 240 to 600 × 600.

From this list, we only considered 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 criterion 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 composed of a group of 10 CNNs with different characteristics, and of a group of 4 CNNs that are variants of those 10. The study is therefore limited to the group of 10 stable CNNs, which, on the one hand, handle images of equal sizes (224 × 224) and, on the other hand, provide the maximal diversity in terms of characteristics and features, as illustrated in Table 6. In particular, the 3rd column provides 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 a CNN's classification satisfies the Top-1

Table 3
0.75-strong adversarial images created by $EA^{target,\mathcal{C}}$ for $\mathcal{C}$ = VGG-16 trained on ImageNet, with different population sizes of $N = 40, 80, 120$, and 160.

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

Table 4
Performance comparison of $EA^{target,\mathcal{C}}$ for $\mathcal{C}$ = VGG-16 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 crossover ranges. The results are the average of the 5 pairs of Table 1 over 10 independent runs for each crossover range.

Range	$\text{avgTime}^{0.75}_{\text{VGG16}}$	$\text{avgGens}^{0.75}_{\text{VGG16}}$	Time/gen	$\text{avg}^{0.75}_{\text{VGG16}}L_2$
10	889 s	2970	0.300 s	3221
20	895 s	3038	0.296 s	3454
30	**840** s	**2957**	**0.283** s	3220
56	947 s	3017	0.319 s	3231
112	928 s	2963	0.307 s	**3211**

(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 VGG-16 ($\mathcal{C}_9$), VGG-19 ($\mathcal{C}_{10}$), and MobileNet ($\mathcal{C}_4$) have the worst performance.

4.2. Image domain

We randomly take 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 randomly take 10 distinct images $\mathcal{A}_q^1, \ldots, \mathcal{A}_q^{10}$ 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 7 specifies the ancestor categories and the target categories obtained that way. In Appendix A, Fig. A.5 shows the 100 selected ancestor images, and Table A.11 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).

4.3. Experiments and indicators

For a threshold value τ and a bound X of the number of generations, to be specified in the concrete experiments performed in Section 5, we run $EA^{target,\mathcal{C}}$ 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 X generations, τ-*strong adversarial images* $\mathcal{D}_k(\mathcal{A}_q^p) = EA^{target,\mathcal{C}_k}(\mathcal{A}_q^p)$ 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 target (but also untargeted) attacks against each CNN. Potential biases are due, for instance, 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, $SR^{\tau}_{\mathcal{C}}$, $SR^{\text{ge}}_{\mathcal{C}}$, and $SR^{\text{untarg}}_{\mathcal{C}}$ for a CNN $\mathcal{C}$, with the former two dealing with the target attack and the latter with the untargeted attack.

The τ-Success Rate $SR^{\tau}_{\mathcal{C}}$ is the percentage of runs of $EA^{target,\mathcal{C}}$ that successfully created at least one τ-*strong adversarial image* within X generations. The *good enough* Success Rate $SR^{\text{ge}}_{\mathcal{C}}$ is the percentage of runs of $EA^{target,\mathcal{C}}$ that successfully created at least one *good enough adversarial image* within X generations, while the EA was aiming at constructing τ-*strong adversarial images*. Finally, the untargeted Success Rate $SR^{\text{untarg}}_{\mathcal{C}}$ is the percentage of runs of $EA^{target,\mathcal{C}}$ that successfully created at least one *adversarial image for the untargeted attack* within X generations, while the EA was aiming at constructing τ-*strong adversarial images*. In this latter case, however, one only considers runs performed on ancestor images that are classified in the ancestor category by the CNN (although rarely, it indeed happens that a CNN does not classify some ancestor images in the correct category despite being chosen from the validation set (see Table A.11)). As already indicated at the end of Section 3.1, one collects some relevant information regarding the production of *good enough adversarial images* or *adversarial images for an untargeted attack* on the way toward the creation of τ-*strong adversarial images*. Note that the fitness function defined by Eq. (4) was not designed to focus on an untargeted attack. Therefore, the outcome for the untargeted attack can be perceived as a by-product of the targeted attack. The inequalities $SR^{\tau}_{\mathcal{C}} \leq SR^{\text{ge}}_{\mathcal{C}} \leq SR^{\text{untarg}}_{\mathcal{C}}$ generally hold (the first inequality does hold systematically, and the second one usually holds).

With notations consistent with Section 3.2, for each CNN we measure the average number of generations ($\text{avgGens}^{all}_{\mathcal{C}}$) and the average time ($\text{avgTime}^{all}_{\mathcal{C}}$, 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 X generations (leading to $\text{avgGens}^{\tau}_{\mathcal{C}}$, $\text{avgTime}^{\tau}_{\mathcal{C}}$), or successfully create at least one *good enough adversarial image* within X generations (leading to $\text{avgGens}^{ge}_{\mathcal{C}}$, $\text{avgTime}^{ge}_{\mathcal{C}}$). *Mutatis mutandis*, we also consider the consistently defined quantities $\text{avgGens}^{untarg}_{\mathcal{C}}$ and $\text{avgTime}^{untarg}_{\mathcal{C}}$ for successful untargeted attacks.

For each CNN, we also report average L_2-distances that assess the visual quality of adversarial images obtained by successful attacks. For the target scenario, on the one hand, $\text{avg}^{\tau}_{\mathcal{C}}L_2$ 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, $\text{avg}^{ge}_{\mathcal{C}}L_2$ 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

Table 5
List of all CNNs with ImageNet pre-trained versions available in Keras.

CNN	Image size	CNN	Image size	CNN	Image size
DenseNet121 ($\mathcal{C}_1$)	224 × 224	MobileNetV2 (see $\mathcal{C}_4$)	224 × 224	Xception	299 × 299
DenseNet169 ($\mathcal{C}_2$)	224 × 224	ResNet50V2 (see $\mathcal{C}_6$)	224 × 224	InceptionV3	299 × 299
DenseNet201 ($\mathcal{C}_3$)	224 × 224	ResNet101V2 (see $\mathcal{C}_7$)	224 × 224	InceptionResNetV2	299 × 299
MobileNet ($\mathcal{C}_4$)	224 × 224	ResNet152V2 (see $\mathcal{C}_8$)	224 × 224	NASNetLarge	331 × 331
NASNetMobile ($\mathcal{C}_5$)	224 × 224			EfficientNetB0	224 × 224
ResNet50 ($\mathcal{C}_6$)	224 × 224			EfficientNetB1	240 × 240
ResNet101 ($\mathcal{C}_7$)	224 × 224			EfficientNetB2	260 × 260
ResNet152 ($\mathcal{C}_8$)	224 × 224			EfficientNetB3	300 × 300
VGG-16 ($\mathcal{C}_9$)	224 × 224			EfficientNetB4	380 × 380
VGG19 ($\mathcal{C}_{10}$)	224 × 224			EfficientNetB5	456 × 456
				EfficientNetB6	528 × 528
				EfficientNetB7	600 × 600

Table 6
The 10 CNNs trained on ImageNet, their number of parameters (in millions) and their Top-1 and Top-5 accuracy.

$\mathcal{C}_k$	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

$\text{avg}_{\mathcal{C}}^{untarg} L_2$ as the average of the L_2-distance between the ancestor image and the first adversarial image that is no longer classified as belonging to the ancestor category.

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

5. One EA versus 10 CNNs: Results

The methodology described in Section 4 is applied with parameter values $\tau = 0.75$ and $X = 10{,}000$. For the experiments reported in the present section (and in Section 3), the algorithm $\text{EA}^{\text{target},\mathcal{C}}$ was implemented using Python 3.7 [30] with NumPy 1.17 [31], TensorFlow 2.1 [32], Keras 2.2 [29], and Scikit 0.24 [33] libraries. Computations were performed on nodes with Nvidia Tesla V100 GPGPUs of the IRIS HPC Cluster at the University of Luxembourg [26].

Section 5.1 summarizes the outcomes of these experiments, and their interpretation is provided in Section 5.2.

The notations used in this section are consistent with those used in Section 4, especially regarding the indicators defined in Section 4.3, given of course for $\tau = 0.75$ and $X = 10{,}000$ (or for increasing values of the maximal number of generations up to 10,000 (see Table B.12 in Appendix B)).

5.1. Experimental results

Table 8 gives the respective performance of our EA for each CNN, for the chosen parameters $\tau = 0.75$ and $X = 10{,}000$. 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 $\text{EA}^{\text{target},\mathcal{C}}$.

Table 9 gives the success rates of our attack against each CNN, and is sorted according to decreasing values of $SR_{\mathcal{C}}^{0.75}$. Table B.12 in Appendix B details the progression of the success rates of $\text{EA}^{\text{target},\mathcal{C}}$ as the maximum number X of generations increases from 1000 to 10,000.

Finally, Fig. 2 shows the convergence characteristics of $\text{EA}^{\text{target},\mathcal{C}}$ for each CNN. In a sense, each curve shows how fast $\text{EA}^{\text{target},\mathcal{C}}$ improves 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 the 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 use the example of MobileNet $= \mathcal{C}_4$ to explain how Fig. 2 should be understood while taking into account the values of $\text{avgGens}_{\mathcal{C}}^{all}$ of Table 8, which 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 $\text{EA}^{\text{target},\mathcal{C}_4}$ give 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 required 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.

5.2. Interpretation

Let us analyze 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 $\text{EA}^{\text{target},\mathcal{C}}$.— With average success rates ≥ 92.8 (regardless of the success rate considered, $SR_{\mathcal{C}}^{0.75}$, $SR_{\mathcal{C}}^{ge}$, or $SR_{\mathcal{C}}^{\text{untarg}}$, see Table 9), the experiments clearly prove that $\text{EA}^{\text{target},\mathcal{C}}$ is highly efficient against all 10 challenged CNNs, at least when $X = 10{,}000$.

Table B.12 (Appendix B) completes the study by showing that $\text{EA}^{\text{target},\mathcal{C}}$ is already very efficient for lower values of the maximum number X of generations taken for the termination condition. For instance, the algorithm achieves average success rates (all CNNs considered, whichever the success rate) $\geq 76\%$ already for $X = 5000$.

Obviously, the success rate of $\text{EA}^{\text{target},\mathcal{C}}$ varies with $\mathcal{C}$. The algorithm $\text{EA}^{\text{target},\mathcal{C}}$ with $X = 10{,}000$ (see Table 9) proves particularly efficient against VGG-16 and MobileNet, with success rates $\geq 99\%$, and is less efficient against NASNetMobile and DenseNet-201, with success rates $\geq 80\%$. The situation is slightly different if one restricts X to $X = 5,000$. In this case (Table B.12, Appendix B), MobileNet remains the most vulnerable (97%), but is followed by ResNet-50 (82%) this time, whereas the most resistant CNN is still NASNetMobile (58%), but followed by VGG-19 (71%) this time, where the percentages given in brackets 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"

Table 7
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}	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 8
Performance comparison of $\text{EA}^{\text{target},\mathcal{C}}$ against each CNN, for $\tau = 0.75$ and $X = 10,000$. 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	$\text{avgGens}_{\mathcal{C}}^{all}$	$\text{avgGens}_{\mathcal{C}}^{0.75}$	$\text{avgGens}_{\mathcal{C}}^{ge}$	$\text{avgGens}_{\mathcal{C}}^{untarg}$	$\text{avgTime}_{\mathcal{C}}^{0.75}$	$\text{avgTime}_{\mathcal{C}}^{ge}$	$\text{avgTime}_{\mathcal{C}}^{untarg}$	$\text{avg}_{\mathcal{C}}^{0.75}L_2$	$\text{avg}_{\mathcal{C}}^{ge}L_2$	$\text{avg}_{\mathcal{C}}^{untarg}L_2$
$\mathcal{C}_4$	MobileNet	2201	2122	1662	1503	562	440	398	2461	2225	2079
$\mathcal{C}_7$	ResNet-101	3428	3154	2586	2550	1285	842	659	3002	2716	2377
$\mathcal{C}_2$	DenseNet-169	3786	3172	2434	2329	1198	919	879	2601	2295	2179
$\mathcal{C}_3$	DenseNet-201	4232	3293	2736	2410	1348	1119	984	2962	2580	2433
$\mathcal{C}_8$	ResNet-152	4054	3466	2985	2385	1246	1073	930	3128	2882	2607
$\mathcal{C}_1$	DenseNet-121	3999	3477	2459	2081	1192	841	712	2801	2450	2214
$\mathcal{C}_6$	ResNet-50	3794	3535	2839	2050	1452	1166	979	3233	2891	2577
$\mathcal{C}_9$	VGG-16	3954	3893	2999	2006	1254	965	644	3892	3429	2715
$\mathcal{C}_5$	NASNetMobile	5148	3935	3231	2495	1426	1170	902	3214	2882	2485
$\mathcal{C}_{10}$	VGG-19	4244	4126	3188	2019	1370	1060	675	4024	3548	2699
	Average	3884	3417	2712	2183	1233	960	776	3132	2790	2436

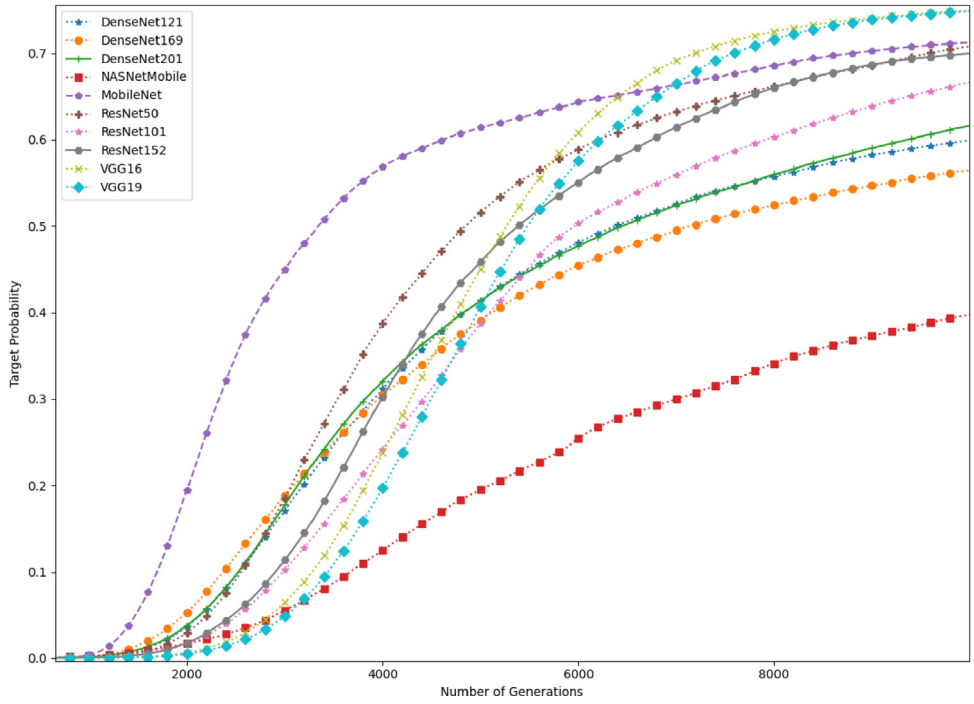


Fig. 2. Convergence characteristics of $\text{EA}^{\text{target},\mathcal{C}}$ for each CNN. Each of the 10 curves is the result of the average runs of $\text{EA}^{\text{target},\mathcal{C}}$ over 100 attacks performed against $\mathcal{C} = \mathcal{C}_k$ for $1 \leq k \leq 10$.

MobileNet and NASNetMobile both have 4M parameters (Table 6), 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 higher accuracy (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 8 (completed by Table B.12 in Appendix B) shows that these success rates are achieved between 2122 generations on average for the fastest CNN to fool, and 4126 generations for the most resistant CNN, and an overall average of 3417 generations for a CNN from 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 set goal.

In terms of the average computational time (on the hardware specified at the beginning of this section), roughly 13 min are

Table 9
Success rates of $EA^{target,C}$ for each CNN, for $\tau = 0.75$ and $X = 10,000$.

CNNs		$SR_C^{0.75}$	SR_C^{ge}	SR_C^{untarg}
C_9	VGG-16	99	100	100
C_4	MobileNet	99	99	99
C_{10}	VGG-19	98	100	100
C_6	ResNet-50	96	99	99
C_8	ResNet-152	96	99	99
C_1	DenseNet-121	92	95	95
C_7	ResNet-101	91	98	98
C_2	DenseNet-169	91	96	97
C_3	DenseNet-201	86	96	96
C_5	NASNetMobile	80	86	87
Average		92.8	96.8	97.0

necessary to create an *adversarial image for the untargeted scenario*, 16 min for a *good enough adversarial image*, and 20 min for a 0.75-*strong adversarial image*.

As shown in Fig. 2, the learning curve of $EA^{target,C}$ differs substantially from one CNN to another. The fastest learning curve (in the short- to mid-term) is achieved for MobileNet (C_4), and the slowest (in the mid- to long term) is achieved for NasNetMobile (C_5). Although the learning curves start very modestly for VGG16 and VGG19 (C_9, C_{10}) since the EA's learning curves for these two are the slowest (hence even slower than for C_5) until circa the 3000th-generation, their slopes sharply improve afterwords, and outperform the others from the $\simeq$ 7000th-generation onward.

Visual quality of the adversarial images.— In terms of the L_2-distance between adversarial and ancestor images, if one takes the value 2436 as a reference point, obtained as the average L_2-distance between such images for the *untargeted scenario* (Table 8), 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 $EA^{target,C}$ to create adversarial images for any of the tested CNNs are unnoticeable to a human eye (at least according to those of the authors of this paper). For instance, Fig. 3 compares an ancestor image and the obtained adversarial images (modulo resizing) for the most difficult CNNs, namely VGG-19 and NasNetMobile (C_{10}, C_5), and the easiest CNNs, namely MobileNet and ResNet-101 (C_4, C_7), as assessed by the values of $avgGens_C^{0.75}$ in Table 8. More precisely, the image on the left of Fig. 3 is $\mathcal{A}_8^{10}$, the ancestor image in the llama category c_{355} pictured in Fig. A.5 in Appendix A. Performing $EA^{target,C}$ on this ancestor image for the (llama, agama) ancestor–target pair for each of these CNNs with $\tau = 0.75$ and $X = 10{,}000$ leads to the 3 groups of adversarial images pictured on Fig. 3. 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 image. The rest of the 0.75-*strong*, *good-enough*, and *untargeted* adversarial images produced by our attack are shown in Appendix B, Fig. B.6, Fig. B.7, and Fig. B.8 respectively.

6. Comparison of our EA-based attack with six well-known attacks

In order to demonstrate the effectiveness of $EA^{target,C}$, we compare it with 6 well-known attacks: FGSM, BIM, PGD Inf, PGD L2, SimBA, and AdvGAN.

6.1. Summary of the attacks

–***Fast Gradient Sign Method (FGSM)*** [34], a white-box attack, is a one-step algorithm that utilizes the gradient of the loss function J(X,y) with respect to input X to determine the direction in which the original input X should be modified. In its untargeted version, the adversarial image is

$$X^{adv} = X + \epsilon sign(\Delta_X J(X, c_a)), \tag{5}$$

while in its targeted version it is

$$X^{adv} = X - \epsilon sign(\Delta_X J(X, c_t)). \tag{6}$$

where ϵ is the perturbation size which is calculated with L_{inf} norm and Δ is the gradient function. We set $eps\_step = 0.01$ and $\epsilon = 8/255$.

–***Basic Iterative Method (BIM)*** [35], a white-box attack, is an iterative version of FGSM, since X_{adv} is initialized with X and is gradually updated for a given number of steps N, as follows:

$$X_{\ell+1}^{adv} = Clip_\epsilon\{X_\ell^{adv} + \alpha sign(\Delta_{\mathcal{A}}(J_C(X_\ell^{adv}, c_a)))\} \tag{7}$$

in its untargeted version and

$$X_{\ell+1}^{adv} = Clip_\epsilon\{X_\ell^{adv} - \alpha sign(\Delta_{\mathcal{A}}(J_C(X_\ell^{adv}, c_t)))\}, \tag{8}$$

in its targeted version, where α is the step size at each iteration and ϵ is the maximum perturbation magnitude of $X^{adv} = X_N^{adv}$. We use the $eps\_step = 0.01$, $max\_iter = int(eps \times 256 \times 1.25)$, and $\epsilon = 2/255$.

–***Projected Gradient Descent Infinite (PGD Inf)*** [36], a white-box attack, is similar to the *BIM* attack, with the difference that the image at the first attack iteration is not initialized with X, but rather with a random point situated within an L_p-ball around X. The distance between X and X^{adv} is measured using $L_{\inf}$ norm. We set $norm = \inf$, $eps\_step = 0.01$, $batch\_size = 1$, and the maximum perturbation magnitude $\epsilon = 8/255$.

–***Projected Gradient Descent** L2 **(PGD L2)*** [36], a white-box attack, is similar to PGD Inf, with the difference that $L_{\inf}$ is replaced with *L2*. We set $norm = 2$, $eps\_step = 0.1$, $batch\_size = 1$, and $\epsilon = 2$.

–***Simple Black-box Attack (SimBA)*** [37], a black-box/white-box attack, is a simple and efficient algorithm that randomly samples a vector from a predefined orthonormal basis and either adds or subtracts it to the target image. It can be used for both targeted and untargeted attacks. We set the overshoot parameter epsilon to 0.2, batch_size to 1, and the maximum number of generations to 10 000 for both targeted and untargeted attacks and use the attack in the black-box setting.

–***Adversarial GAN attack (AdvGAN)*** [38], a black-box/semi-whitebox attack, uses generative adversarial network (GAN) to generate adversarial images. AdvGAN consists of three main parts: a generator, a discriminator, and the target neural network. The generator in AdvGAN is trained to produce perturbation which will result in adversarial images when it is added to the original images. The discriminator's goal is to be sure that the generated adversarial image is indistinguishable from the original image. We set the L_{inf} perturbation bound to 0.01 and use the attack in the black-box setting.

6.2. Tests design

FGSM, BIM, PGD Inf, PGD L2, and SimBA are imported from the Adversarial Robustness Toolbox (ART) [39] which is a Python library that includes several attack methods. AdvGAN is implemented from its original paper [38] for the CNNs trained with the ImageNet dataset and verified with similar implementations [40, 41]. Default settings of FGSM, BIM, PGD Inf, PGD L2, and SimBA in the Adversarial Robustness Toolbox (ART) [39] are used in

Fig. 3. Visual comparison of ancestor and adversarial images obtained by $EA^{target,\mathcal{C}}$ for $\mathcal{C}$ = VGG-19, NASNetMobile, MobileNet, and ResNet-101 ($\mathcal{C} = \mathcal{C}_{10}, \mathcal{C}_5, \mathcal{C}_4, \mathcal{C}_7$), for the (llama, agama) ancestor–target pair and the ancestor image $\mathcal{A}_8^{10}$ taken from Fig. A.5 in Appendix A.

our experiments. AdvGAN parameters are taken from its original paper.

Due to the varied techniques and constraints utilized in these attacks, determining metrics for comparing their performances can prove challenging. For instance, measuring the number of generations is an appropriate metric for evolutionary algorithm-based attacks like EA and SimBA, but it is not relevant for gradient-based attacks like FGSM, PGD, and BIM. So, to assess the effectiveness of these methods, we focused specifically on generic metrics such as success rate and visual quality as evaluation criteria.

The attacks are performed using the same ancestor/target pairs against the 10 CNNs described in Sections 5.1 and 5.2 for targeted and untargeted scenarios. So each attack attempts to generate around 1000 adversarial images in both targeted and untargeted attacks.

6.3. Results

We summarize here the qualitative and quantitative results of our experiments.

Success rate comparison : Table 10 summarizes the attacks' parameters and compares the attacks' performance in terms of success rates for both targeted and untargeted scenarios. SimBA and FGSM perform poorly in generating adversarial images in the targeted scenario. Although SimBA's epsilon value is doubled from its default setting, it does not succeed in the targeted scenario within 10000 generations. As mentioned in [37], SimBA performs better for small-size images as in Cifar10 or MNIST. When it comes to higher-resolution images, SimBA needs more generations (>10 000) to converge into the target category. Nevertheless, SimBA managed to move the majority of the ancestor images from their true class to some other classes with a success rate of 84.9%. As with SimBA, FGSM performs better for the untargeted scenario with a success rate of 77.1%, while it fails for the targeted scenario.

PGD Inf and PGD L2 successfully generate adversarial images in both scenarios with a success rate of over 85%. PGD Inf has the best success rate among all attacks for the untargeted scenario with 97.1%. BIM is the iterative version of FGSM. In our experiments, we set the L_{inf} bound for BIM at 2/255, which is four times lower than what we used for FGSM. Despite this challenging constraint, BIM still achieved a similar success rate to FGSM in the untargeted scenario, and even outperformed FGSM in the targeted scenario.

In regards to AdvGAN, we set the L_{inf} value to 2.25/255 as in its original paper. This resulted in a success rate of over 50% for both scenarios. Through our experimentation, we observed that increasing the L_{inf} value significantly improves the success rate of AdvGAN. In addition to L_{inf}, the choice of the generator and discriminator architecture has a substantial impact on the performance of AdvGAN.

As seen in Table 10, our EA attack demonstrates exceptional performance, with a success rate of 96.8%, outperforming the results of all other white- and black-box attacks in the targeted scenario. Although our attack is not specifically designed for untargeted scenarios, it still ranks as the second most effective among other attacks.

Qualitative results: For qualitative evaluation of our attack Fig. 4 presents the visual comparison of ancestor and adversarial images obtained by the attacks against VGG16 along with the label and label values. The perturbations added by FGSM can be easily discerned. The subtle contrast change and noise pattern added to the original image by SimBA and BIM, respectively, can

Table 10
Performance comparison of attacks in terms of success rate for the untargeted and targeted scenarios.

Attacks		FGSM		PGD Inf		PGD L2		BIM		SimBA		AdvGAN		EA	
Settings:		$L_{inf} = 8/255$		$L_{inf} = 8/255$		$L_2 = 2$		$L_{inf} = 2/255$		max_gen = 10 000 eps = 0.2		$L_{inf} = 2.25/255$		max_gen = 10 000	
		Untargeted	Targeted	Untargeted	Targeted	Untargeted	Targeted	Untargeted	Targeted	Untargeted	Targeted	Untargeted	Targeted	Untargeted	Targeted
C1	DenseNet-121	82	4	**99**	**98**	98	73	57	36	91	0	67	64	95	95
C2	DenseNet-169	67	1	**98**	**97**	95	80	57	42	87	0	72	72	97	96
C3	DenseNet-201	66	2	**98**	**96**	95	75	49	34	83	0	54	52	96	**96**
C4	MobileNet	78	1	**99**	94	97	96	79	54	86	1	60	58	**99**	**99**
C5	NASNetMobile	52	1	95	52	**96**	67	52	14	74	0	57	51	87	86
C6	ResNet-50	85	1	**99**	100	98	99	93	88	88	0	64	61	99	99
C7	ResNet-101	84	0	**98**	100	97	98	90	80	88	0	58	56	**98**	98
C8	ResNet-152	77	4	95	99	89	99	85	80	78	0	53	51	**99**	99
C9	VGG-16	92	7	97	100	94	86	78	61	89	0	56	39	**100**	100
C10	VGG-19	88	3	93	100	91	86	71	54	85	0	39	25	**100**	100
Overall average		77.1	2.4	**97.1**	93.6	95	85.9	71.1	54.3	84.9	0.1	58	52.9	97	**96.8**

only be detected by an attentive observer who has previously viewed the original image. The remaining attacks generated such a disturbance in the original images that they are undetectable to the human eye.

7. Conclusion

In this paper, we presented an evolutionary algorithm (EA) as a powerful black-box attack for CNNs trained on ImageNet for image classification tasks. We evaluated the performance of our EA-based attack against four commonly used white-box attacks (FGSM, PGD Inf, PGD L2, and BIM) and two other black-box attacks (SimBA and AdvGAN) on 10 stable CNNs with diverse architectures. This paper not only describes an adversarial attack based on evolutionary algorithms but also provides a comprehensive evaluation of popular adversarial attacks on generating high-resolution (224×224) adversarial images. Our results show that the EA-based attack outperforms or is on par with these methods in both targeted and untargeted attacks. Our findings suggest that our EA-based attack is the most successful attack for the selected CNNs. Despite these positive outcomes, our EA-based attack can still be improved. Specifically, the algorithm demonstrated a longer convergence time for larger images in reaching the desired category. In contrast to traditional evolutionary algorithms, the mutation process holds a vital role in our algorithm. Enhancing this process could enhance its capability to explore and achieve faster convergence toward the target category.

CRediT authorship contribution statement

Ali Osman Topal: Conceptualization, Methodology, Software, Validation, Writing – review & editing, Visualization. **Raluca Chitic:** Validation, Methodology. **Franck Leprévost:** Conceptualization, Methodology, Investigation, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A. Ancestor images

See Table A.11 and Fig. A.5.

Appendix B. Adversarial images

See Table B.12 and Figs. B.6–B.8.

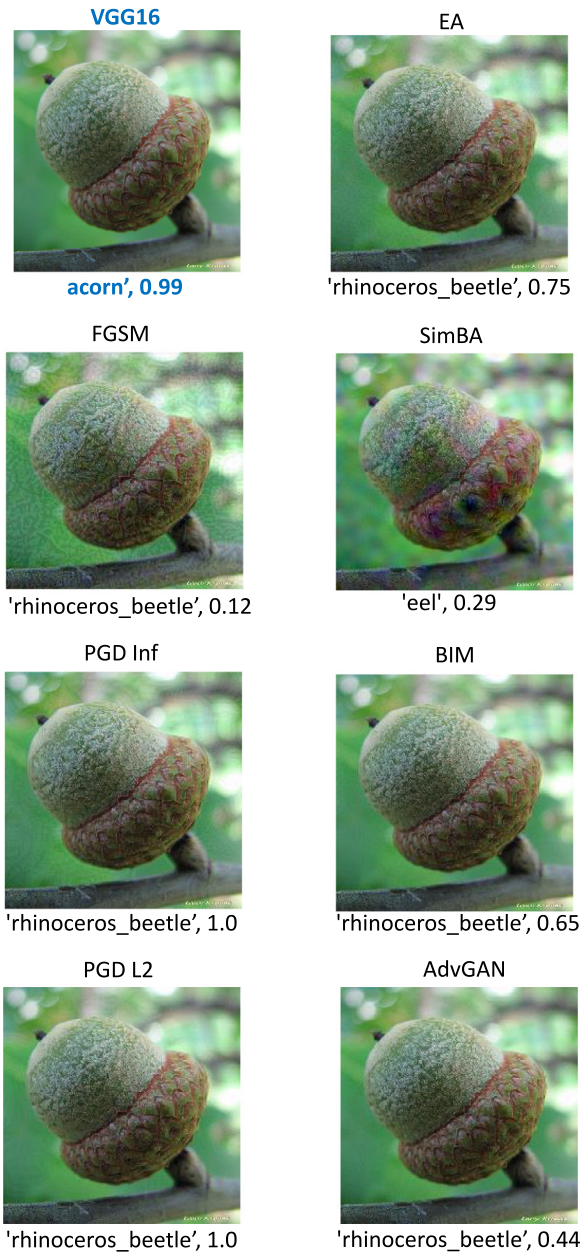


Fig. 4. A visual comparison of ancestor and adversarial images generated by FGSM, PGD Inf, PGD L2, BIM, SimBA, AdvGAN, and EA, against the VGG16 model is presented. The first image represents the original, unaltered image, while the following images depict the adversarial images. Each adversarial image is labeled with the name of the attack at the top, and the label and label values of the VGG16 at the bottom.

Fig. A.5. The 100 ancestor images $\mathcal{A}_q^p$ used in the experiments. $\mathcal{A}_q^p$ pictured in the pth column and qth 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 qth row.

Table A.11
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 Fig. A.5. A label value in red indicates that the category c_{a_q} is not the dominant one.

CNNs	p	abacus	acorn	baseball	broom	brown bear	canoe	hippopotamus	llama	maraca	mountain bike
$\mathcal{C}_1$ DenseNet-121	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
	5	1.000	0.999	1.000	0.998	0.955	0.811	1.000	1.000	0.145	0.986
	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
	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

(*continued on next page*)

Table A.11 (*continued*).

CNNs	p	abacus	acorn	baseball	broom	brown bear	canoe	hippopotamus	llama	maraca	mountain bike
$\mathcal{C}_2$ DenseNet-169	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**
$\mathcal{C}_3$ DenseNet-201	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**
$\mathcal{C}_6$ ResNet-50	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
$\mathcal{C}_7$ ResNet-101	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
	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
$\mathcal{C}_8$ ResNet-152	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
$\mathcal{C}_9$ VGG-16	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

(*continued on next page*)

Table A.11 (*continued*).

CNNs	p	abacus	acorn	baseball	broom	brown bear	canoe	hippopotamus	llama	maraca	mountain bike
	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
$\mathcal{C}_{10}$ VGG-19	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

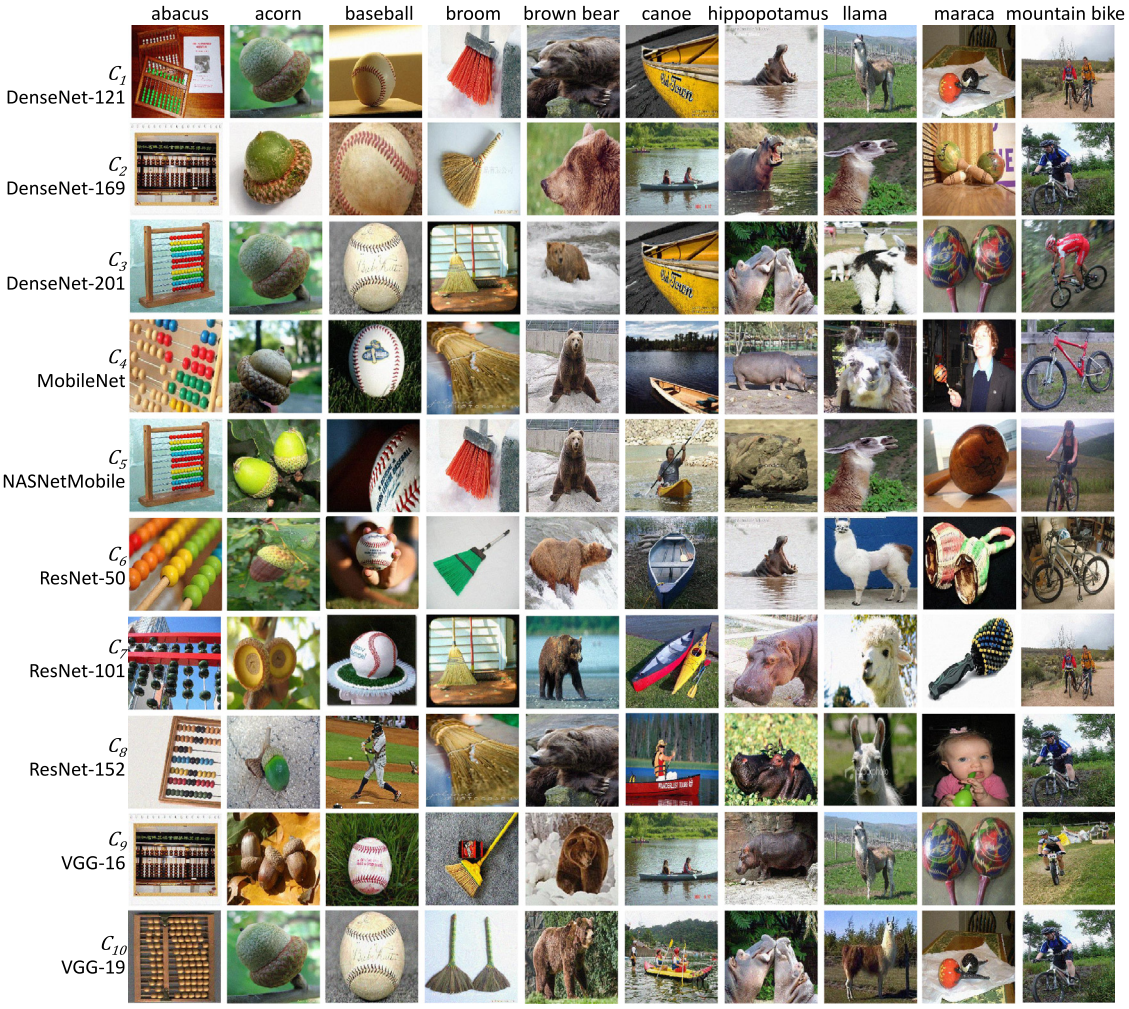


Fig. B.6. Samples of 0.75-*strong adversarial images* generated by $\text{EA}^{\text{target},\mathcal{C}_k}$ for $1 \leq k \leq 10$.

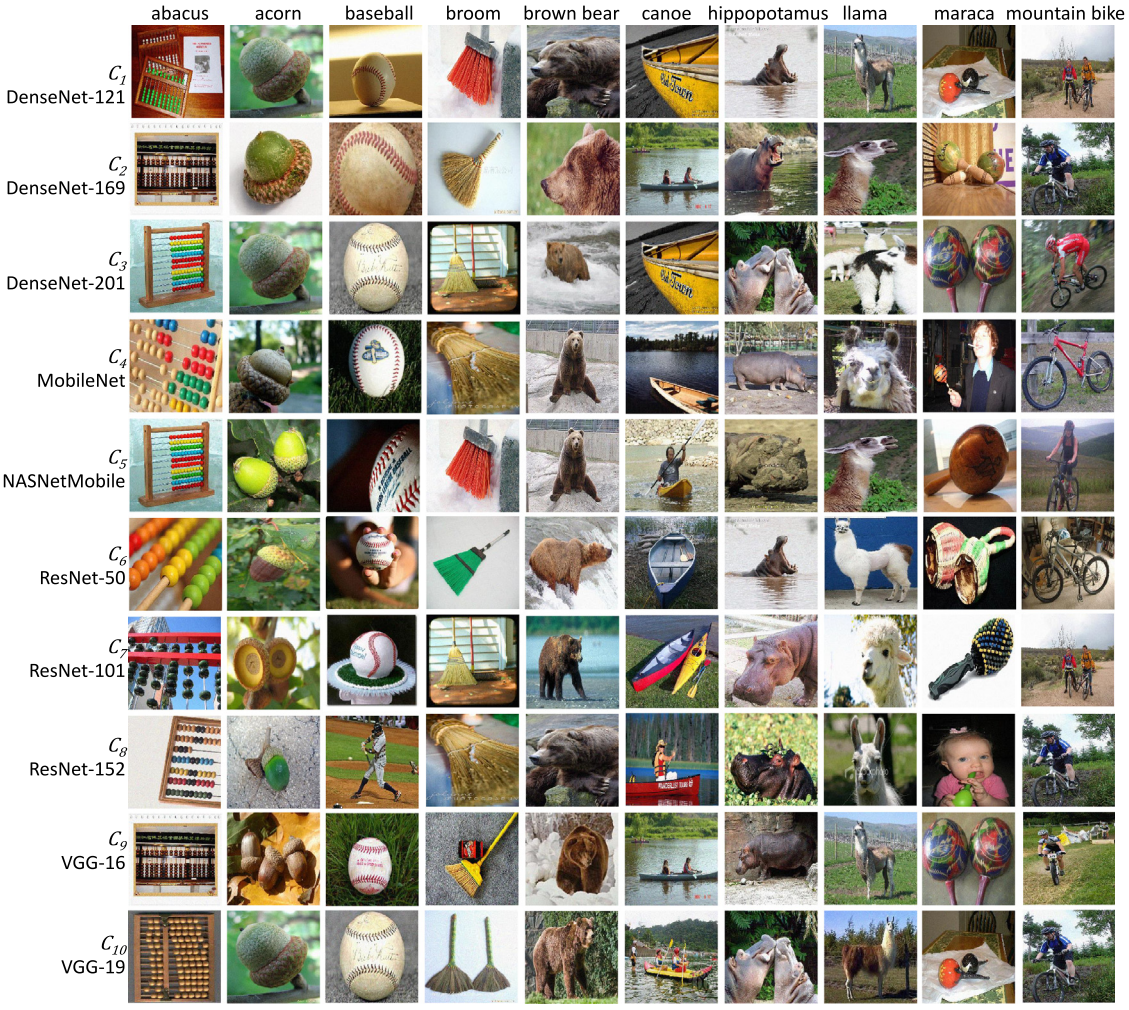


Fig. B.7. Samples of *good enough adversarial images* generated by $\text{EA}^{\text{target},C_k}$ for $1 \leq k \leq 10$.

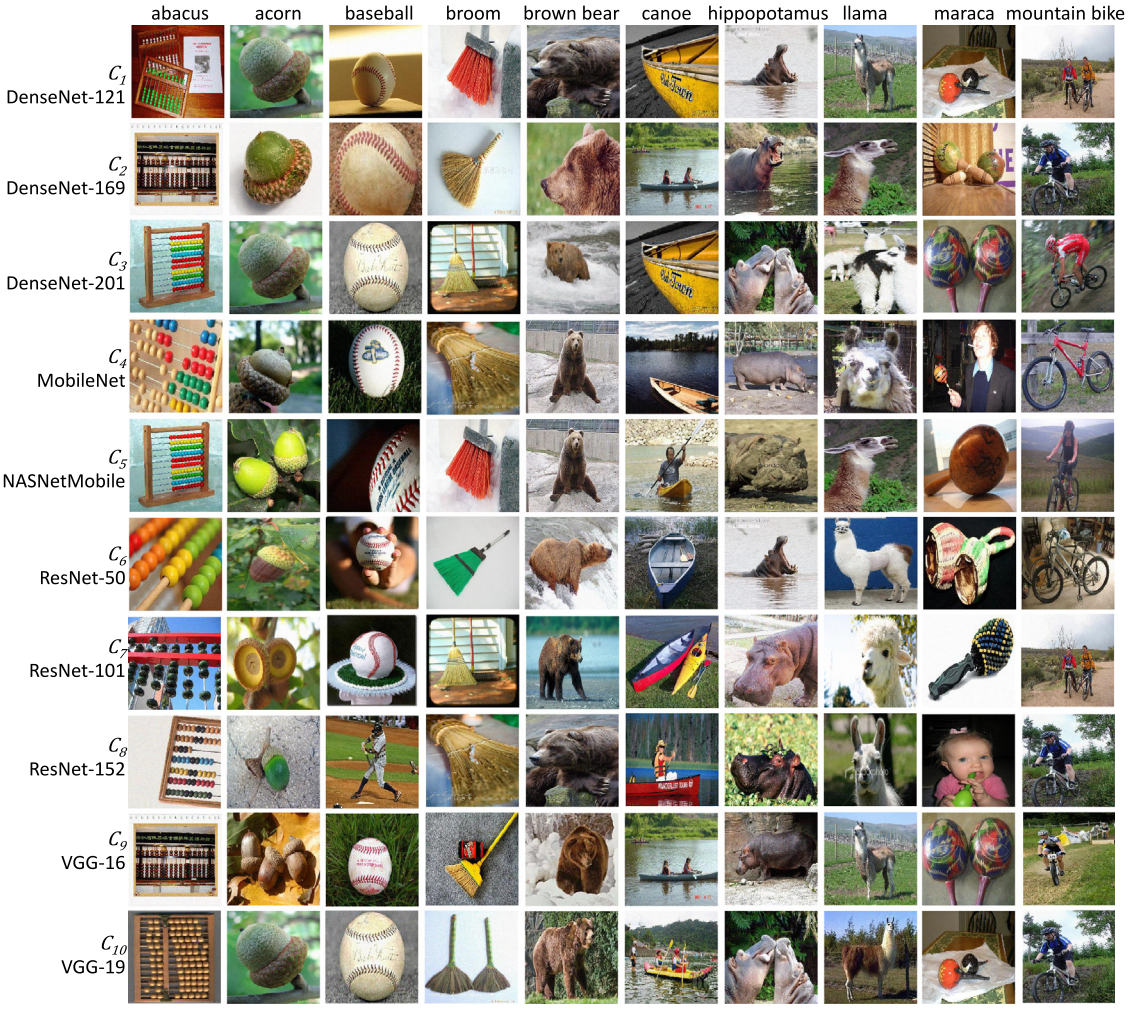


Fig. B.8. Samples of *untargeted adversarial images* generated by $\text{EA}^{\text{target},C_k}$ for $1 \leq k \leq 10$.

Table B.12
Success rates of $EA^{target,C}$ for increasing values of X while $\tau = 0.75$ for the experiments designed in Section 4 and performed in Section 5.

X	Success rates	$\mathcal{C}_1$-DenseNet-121	$\mathcal{C}_2$-DenseNet-169	$\mathcal{C}_3$-DenseNet-201	$\mathcal{C}_4$-MobileNet	$\mathcal{C}_5$-NASNetMobile	$\mathcal{C}_6$-ResNet-50	$\mathcal{C}_7$-ResNet-101	$\mathcal{C}_8$-ResNet-152	$\mathcal{C}_9$-VGG-16	$\mathcal{C}_{10}$-VGG-19	**Average**
1000	$SR_C^{0.75}$	5	1	2	10	4	2	1	2	2	2	**3.1**
	SR_C^{ge}	10	9	7	23	7	12	4	7	4	2	**8.5**
	SR_C^{untarg}	27	19	18	34	26	31	21	23	23	27	**25.0**
2000	$SR_C^{0.75}$	19	27	19	57	15	26	26	19	15	12	**23.5**
	SR_C^{ge}	45	48	34	71	28	39	35	37	27	29	**39.3**
	SR_C^{untarg}	53	51	41	76	46	56	49	48	64	62	**54.7**
3000	$SR_C^{0.75}$	47	53	40	85	33	50	43	46	35	31	**46.3**
	SR_C^{ge}	70	75	69	89	47	68	62	59	57	51	**64.7**
	SR_C^{untarg}	74	76	73	89	55	82	70	72	84	78	**75.2**
4000	$SR_C^{0.75}$	66	71	64	90	46	73	61	61	58	50	**64.0**
	SR_C^{ge}	83	82	82	97	61	87	80	83	77	72	**80.4**
	SR_C^{untarg}	84	83	85	97	72	90	83	86	89	88	**85.7**
5000	$SR_C^{0.75}$	74	77	74	97	58	82	75	79	75	71	**76.2**
	SR_C^{ge}	87	90	84	97	73	92	88	91	91	88	**88.1**
	SR_C^{untarg}	88	90	87	97	77	93	89	93	93	96	**90.1**
6000	$SR_C^{0.75}$	82	81	80	98	66	89	83	87	86	82	**83.4**
	SR_C^{ge}	92	91	89	98	76	94	92	92	95	94	**91.3**
	SR_C^{untarg}	92	91	91	98	80	96	93	94	97	97	**92.8**
7000	$SR_C^{0.75}$	84	86	84	98	72	93	86	90	93	89	**87.5**
	SR_C^{ge}	93	92	92	98	81	96	92	95	98	97	**93.4**
	SR_C^{untarg}	93	92	93	98	83	97	93	96	99	97	**94.0**
8000	$SR_C^{0.75}$	88	88	85	98	73	94	89	92	97	94	**89.8**
	SR_C^{ge}	95	95	93	99	81	97	92	97	99	99	**94.7**
	SR_C^{untarg}	95	95	94	99	86	98	93	97	99	99	**95.4**
9000	$SR_C^{0.75}$	90	90	86	98	78	95	91	94	98	97	**91.7**
	SR_C^{ge}	95	96	95	99	85	98	96	98	100	100	**96.2**
	SR_C^{untarg}	95	97	95	99	87	98	96	98	100	100	**96.5**
10 000	$SR_C^{0.75}$	92	91	86	99	80	96	91	96	99	98	**92.8**
	SR_C^{ge}	95	96	96	99	86	99	98	99	100	100	**96.8**
	SR_C^{untarg}	95	97	96	99	87	99	98	99	100	100	**97.0**

References

[1] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training data-efficient image transformers & distillation through attention, in: International Conference on Machine Learning, PMLR, 2021, pp. 10347–10357.

[2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 2818–2826, URL https://ieeexplore.ieee.org/document/7780677.

[3] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

[4] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.

[5] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[6] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[7] Z.J. Wang, R. Turko, O. Shaikh, H. Park, N. Das, F. Hohman, M. Kahng, D.H.P. Chau, CNN explainer: Learning convolutional neural networks with interactive visualization, IEEE Trans. Vis. Comput. Graphics 27 (2) (2020) 1396–1406.

[8] S. Thys, W. Van Ranst, T. Goedemé, Fooling automated surveillance cameras: adversarial patches to attack person detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019.

[9] A. Fawzi, H. Fawzi, O. Fawzi, Adversarial vulnerability for any classifier, Adv. Neural Inf. Process. Syst. 31 (2018).

[10] R. Chitic, A.O. Topal, F. Leprévost, Evolutionary algorithm-based images, humanly indistinguishable and adversarial against convolutional neural networks: efficiency and filter robustness, IEEE Access 9 (2021) 160758–160778.

[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, 2013, arXiv preprint arXiv:1312.6199.

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

[13] J. Su, D.V. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks, IEEE Trans. Evol. Comput. 23 (5) (2019) 828–841.

[14] J. Wu, Generating adversarial examples in the harsh conditions, 2020, CoRR abs/1908.11332. URL https://arxiv.org/abs/1908.11332.

[15] M. Jere, L. Rossi, B. Hitaj, G. Ciocarlie, G. Boracchi, F. Koushanfar, Scratch that! An evolution-based adversarial attack against neural networks, 2019, CoRR abs/1912.02316. URL https://arxiv.org/abs/1912.02316.

[16] R. Chitic, F. Leprévost, N. Bernard, Evolutionary algorithms deceive humans and machines at image classification: an extended proof of concept on two scenarios, J. Inf. Telecommun. (2020) 1–23.

[17] R. Chitic, N. Bernard, F. Leprévost, 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), Springer, Heidelberg, 2020, pp. 467–480.

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

[19] S.A. Fezza, Y. Bakhti, W. Hamidouche, O. Déforges, Perceptual evaluation of adversarial attacks for CNN-based image classification, in: 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX), IEEE, 2019, pp. 1–6.

[20] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 39–57.

[21] Z. Wang, A. Bovik, H. Sheikh, E. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (2004) URL https://ieeexplore.ieee.org/document/128439.

[22] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer, 2003, URL https://www.springer.com/gp/book/9783642072857.

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

[24] B. Doerr, Runtime analysis of evolutionary algorithms via symmetry arguments, Inform. Process. Lett. 166 (2021) 106064.

[25] S. Forrest, M. Mitchell, What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation, Mach. Learn. 13 (2) (1993) 285–319.

[26] S. Varrette, P. Bouvry, H. Cartiaux, F. Georgatos, Management of an academic HPC cluster: The UL experience, in: Proceedings of the 2014 International Conference on High Performance Computing & Simulation (HPCS 2014), IEEE, 2014, pp. 959–967.

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

[28] B. Zoph, V. Vasudevan, J. Shlens, Q.V. Le, Learning transferable architectures for scalable image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8697–8710.

[29] F. Chollet, et al., Keras, 2015, https://keras.io.

[30] G. Van Rossum, F.L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009.

[31] T.E. Oliphant, A Guide to NumPy, Trelgol Publishing USA, 2006.

[32] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, URL https://www.tensorflow.org/. Software available from tensorflow.org.

[33] S. Van der Walt, J.L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J.D. Warner, N. Yager, E. Gouillart, T. Yu, the scikit-image contributors, Scikit-image: image processing in python, PeerJ 2 (2014) e453, http://dx.doi.org/10.7717/peerj.453.

[34] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, 2015, CoRR abs/1810.00069. arXiv:1412.6572. URL http://arxiv.org/abs/1412.6572.
[35] A. Kurakin, I.J. Goodfellow, S. Bengio, Adversarial examples in the physical world, 2016, CoRR abs/1607.02533. arXiv:1607.02533. URL http://arxiv.org/abs/1607.02533.
[36] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, 2019, CoRR arXiv:1706.06083. URL http://arxiv.org/abs/1706.06083.
[37] C. Guo, J. Gardner, Y. You, A.G. Wilson, K. Weinberger, Simple black-box adversarial attacks, in: International Conference on Machine Learning, PMLR, 2019, pp. 2484–2493.
[38] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, D. Song, Generating adversarial examples with adversarial networks, 2018, arXiv preprint arXiv:1801.02610.
[39] M. Nicolae, M. Sinn, T.N. Minh, A. Rawat, M. Wistuba, V. Zantedeschi, I.M. Molloy, B. Edwards, Adversarial robustness toolbox v1.0.0, 2018, CoRR abs/1807.01069. URL http://arxiv.org/abs/1807.01069.
[40] C. Targonski, Tensorflow implementation of generating adversarial examples with adversarial networks, 2019, URL https://github.com/ctargon/AdvGAN-tf/.
[41] N. Jain, Keras implementation of AdvGAN, 2019, URL https://github.com/niharikajainn/adv_gan_keras.