# Evolutionary algorithms deceive humans and machines at image classification: An extended proof of concept on two scenarios

Raluca Chitic[1 [0000−0003−1113−2343]], Franck Leprévost[1 [0000−0001−8808−2730]], and Nicolas Bernard[2 [0000−0003−3857−9591]]

[1] University of Luxembourg
House of Numbers, 6, avenue de la Fonte, L-4364 Esch-sur-Alzette
G-D of Luxembourg
Raluca.Chitic@uni.lu & Franck.Leprevost@uni.lu
[2] La Fraze
1288, chemin de la Fraze, F-88380 Arches, France
Nicolas.Bernard@lafraze.com

**Abstract.** The range of applications of Neural Networks encompasses image classification. However, Neural Networks are vulnerable to attacks, and may misclassify adversarial images, leading to potentially disastrous consequences. Pursuing some of our previous work, we provide an extended proof of concept of a black-box, targeted, non-parametric attack using evolutionary algorithms to fool both Neural Networks and humans at the task of image classification. Our feasibility study is performed on VGG-16 trained on CIFAR-10. For any category $c_{\mathcal{A}}$ of CIFAR-10, one chooses an image $\mathcal{A}$ classified by VGG-16 as belonging to $c_{\mathcal{A}}$. From there, two scenarios are addressed. In the first scenario, a target category $c_t \neq c_{\mathcal{A}}$ is fixed a priori. We construct an evolutionary algorithm that evolves $\mathcal{A}$ to a modified image that VGG-16 classifies as belonging to $c_t$. In the second scenario, we construct another evolutionary algorithm that evolves $\mathcal{A}$ to a modified image that VGG-16 is unable to classify. In both scenarios, the obtained adversarial images remain so close to the original one that a human would likely classify them as still belonging to $c_{\mathcal{A}}$.

## 1 Introduction

Fast and accurate image classification has become an important topic in a series of industrial (automation, robots, self-driving cars, etc.), security and military (monitoring, face recognition, etc.) domains, just to mention a few [12]. Neural Network (NN) approaches to image classification have outperformed traditional image processing techniques, becoming today's most effective tool for these tasks [29]. Concretely, a NN is trained on a large set of examples. During this process, the NN is given both the input image and the output category it is expected to associate with the input image. Based on this accumulated "knowledge", the NN can then classify new images into categories with high confidence.

However, the process leading to the classification by a NN still seems to differ significantly from the way humans perform the same task. It is therefore tempting to exploit this difference and cause the NN to make classification errors. Indeed, trying to fool NNs has become an intensive research topic [19, 18, 11], since the vulnerabilities of NNs used for critical applications can have disastrous consequences, and lead to harmful decisions. It is hence important to better understand these weaknesses.

The present work is a contribution to this understanding. It provides an effective method to construct adversarial images without prior knowledge of the features of the NN. The proof of concept of this method is achieved by a test against a concrete NN. More concretely, our work further substantiates our research program [3] to construct Evolutionary Algorithms (EA) which create images to deceive machines and humans at image classification. The present article, extending significantly [4], describes the construction of two such EAs (that actually differ only by their fitness function, and are variants of those used in [2]) in the following context. Trained on an image dataset, a given NN sorts images into $\ell$ categories. For any category $c_{\mathcal{A}}$, one chooses an image $\mathcal{A}$ classified by the NN as belonging to $c_{\mathcal{A}}$. From there, two scenarios (see [3] for more) are considered. In the first "target" scenario, a target category $c_t \neq c_{\mathcal{A}}$ is fixed a priori. We construct the evolutionary algorithm $\mathrm{EA}_d^{\mathrm{target}}$, that evolves $\mathcal{A}$ to a modified image that the NN classifies as belonging to $c_t$. In the second "flat" scenario, we construct the evolutionary algorithm $\mathrm{EA}_d^{\mathrm{flat}}$ that evolves $\mathcal{A}$ to a modified image that the NN is unable to classify with certainty to any of the $\ell$ categories. In both scenarios, the EAs are used with two different *similarity measures* $d$ ($L_2$ and SSIM), that assess differently the proximity between the created adversarial images and the image $\mathcal{A}$. In each case, a human would likely classify the obtained adversarial images as still belonging to $c_{\mathcal{A}}$.

The strategy being set, experiments are performed with the neural network VGG-16 [25] trained on the image dataset CIFAR-10 [14] to sort images into $\ell = 10$ categories. The outcome is that the algorithms $\mathrm{EA}_d^{\mathrm{target}}$ and $\mathrm{EA}_d^{\mathrm{flat}}$ do create adversarial images that, VGG-16 misclassifies in the former case, and that VGG-16 cannot classify with certainty in any category in the latter case. The evolved images remain similar to the original ones. Our results give substantial reasons to believe that humans would not do the same misclassification as VGG-16, and would hence label the modified images as belonging to the original category, both for the "target" scenario and for the "flat" scenario.

The remainder of this article is organised as follows. Section 2 summarizes the necessary background about NNs and EAs, and positions our evolutionary algorithm approach as a a black-box, targeted, non-parametric attack. Section 3 details the two scenarios considered, and the strategy to address them. In particular, it details the generic fitness functions used in $\mathrm{EA}_d^{\mathrm{target}}$, and in $\mathrm{EA}_d^{\mathrm{flat}}$. Section 4 provides the parameters used by $\mathrm{EA}_d^{\mathrm{target}}$ and $\mathrm{EA}_d^{\mathrm{flat}}$ to fool VGG-16

trained on CIFAR-10. Section 5 (respectively section 6) gives the results outputted by $\mathrm{EA}_d^{\mathrm{target}}$ in the "target" scenario (respectively by $\mathrm{EA}_d^{\mathrm{flat}}$ in the "flat" scenario). More complete results are provided in the Appendix (section A), including concrete ancestor and evolved images for both scenarios. Conclusions about the present work, and the announcement of further directions are given in section 7.

## 2 Neural Networks, Evolutionary Algorithms and typologies of attacks

### 2.1 Neural Networks and Evolutionary Algorithms in a nutshell

We give here the very minimum background about Neural Networks (NNs) and Evolutionary Algorithms (EAs) needed to position our contribution (section 3) in the context of some related work (2.2).

*A Neural Network* can be considered here as a black box (see [10] for a detailed description of NNs in the context of Deep Learning), which takes an image $p$ as input and produces the corresponding output, consisting of a vector $\boldsymbol{o}_p$ of length $\ell$. Each position $1 \leq i \leq \ell$ in the output vector represents a category $c_i$ of objects defined in a dataset. Since the vector components are probabilities, one has $0 \leq \boldsymbol{o}_p[i] \leq 1$, and $\sum_{i=1}^{\ell} \boldsymbol{o}_p[i] = 1$. A NN is first trained on images of the dataset. Once trained, the NN produces a label for an image $p$ by extracting the category for which the probability is highest, e.g. $c_i$ if $i = \arg\max_{1 \leq j \leq \ell}(\boldsymbol{o}_p[j])$.

*Evolutionary Algorithms* search for solutions to optimisation problems by imitating natural evolution (see [26, sec. 3] for a quick introduction. For a general introduction to evolutionary algorithms, one can consult [13] and [35]). Whereas NNs are trained on a set of examples, EAs are heuristic and search for solutions in the entire image space. Each generation of the algorithm presents a new, generally improved population $g_{i+1}$, compared to the previous one $g_i$. An EA mainly consists of a loop in which 1) an initial population of individuals is created or chosen; 2) each member of the population is evaluated through a fitness function; 3) a new population is created from the existing one, taking into account each individual's fitness.

### 2.2 Context and typologies of attacks

Consider a sample $x$, which is classified correctly by $M$, a machine learning model ($MLM$), as $M(x) = c_{true}$ (for instance the classification of an image by a NN). To fool this MLM, one can create an adversarial sample $x'$ (this concept was introduced by Szegedy *et al.* [28]) as the result of a tiny (and unnoticed, ideally) perturbation of the original sample $x$, such that $M$ misclassifies it, i.e. $M(x') \neq c_{true}$. Let us classify these attacks here according to two criteria: white-box vs black-box attacks and targeted vs non-targeted attacks.

White-box and black-box attacks depend on the amount of information about the MLM available to the adversaries. In white-box attacks, the adversaries have full access to the MLM. For instance, if the MLM is a NN, they know the training algorithm, the training set and the parameters of the NN. The adversaries analyse the model constraints and identify the most vulnerable feature space, altering the input accordingly [21, 28]. In black-box attacks, the adversaries have no knowledge about the MLM (the NN in our context) and they instead analyse the vulnerabilities by using information about the past input/output pairs. More specifically, the adversaries attack a model by inputting a series of adversarial samples and observing corresponding outputs.

Previous attempts [23, 7] to deceive NNs dealt primarily with untargeted attacks. In these, the task is to take some input $x$, which is correctly labeled as $c_{true}$ by a NN, and to modify it towards an $x'$ with distance $d(x, x') < e$, which is labeled as a different class than $c_{true}$. For example, a small change imperceptible to humans misleads the network into classifying an image of a tabby cat as guacamole [24]. A targeted attack, while usually harder to perform, is able to produce adversarial samples for a specific target class. Given classes $c_a$ and $c_b$, with an input $x$ classified as belonging to $c_a$, a targeted attack produces a similar $x'$, which the NN classifies as belonging to $c_b$.

### 2.3 Evolutionary Algorithms leading to black-box, targeted, non parametric attacks

Within the category of black-box attacks, which are attempted in this paper, there are multiple methods to fool a NN. Most of these approaches, however, are still reliant on the model and its parameters [30, 22]. Bypassing them might be achieved merely by training a NN to recognise the product of these methods. By contrast, EAs are extremely flexible and adaptive. Our contribution positions EAs as an alternative approach that has the advantage of being non-parametric (see [34], [16] for other non-parametric models for adversarial attacks). Our non-parametric model presents no bias, unless of course such a bias is programmed in the EA.

For EAs, among the most important aspects are the definition of their target and surrounding landscape, both done through the fitness function. The fitness function also reveals the main difference between our paper and the previous work on adversarial attacks using evolutionary algorithms. While Su *et al.* [26] use an EA to produce misclassifications by only changing one pixel per input image, our method does not limit the number of pixels which can be modified. Limiting the number of modified pixels leads to higher perturbation values for a few pixels, which become very different from their surroundings. However, our focus is on making the modifications less visible by humans, and hence the goal is to limit the overall value change, rather than the pixel count.

The following section describes our black-box, targeted, non parametric attack. More precisely, it provides the construction of our evolutionary algorithms, that create adversarial images which aim to fool NNs, and human beings.

## 3 Scenarios, Strategy and Implementation

Our algorithm is an EA that aims to deceive both NNs and human beings. Clearly, the design of the EA depends on the scenario governing this dual deception. The two scenarios addressed here (see [3] for others) start the same way. One is initially given an image, the "ancestor" $\mathcal{A}$, labelled by the NN 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 NN 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 NN is unable to classify with certainty to any specific category, in the sense that the NN 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 $c_{\mathcal{A}}$.

$\mathrm{EA}_d^{\mathrm{target}}$ refers to the evolutionary algorithm of the first scenario, and $\mathrm{EA}_d^{\mathrm{flat}}$ to the evolutionary algorithm of the second scenario, where $d$ is one of the two similarity measures of section 3.2.

### 3.1 Common features between $\mathbf{EA}_{\boldsymbol{d}}^{\mathbf{target}}$ and $\mathbf{EA}_{\boldsymbol{d}}^{\mathbf{flat}}$

$\mathrm{EA}_d^{\mathrm{target}}$ and $\mathrm{EA}_d^{\mathrm{flat}}$ are descendants — in spirit if not in code — of the algorithm used in [2]. While their fitness functions differ, and hence so does the evaluation step, the population initialization and the evolution steps are similar for $\mathrm{EA}_d^{\mathrm{target}}$ and for $\mathrm{EA}_d^{\mathrm{flat}}$.

*Population initialisation.* 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 $\mathrm{EA}_d^{\mathrm{target}}$ and of $\mathrm{EA}_d^{\mathrm{flat}}$ 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 -1 or 1, in order to maintain a high similarity between the images.
  - 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.

As in [2], an equivalence is made between the population of the EA and a batch of the NN so that the NN 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:
$$L_2\left(i, i'\right) = \sum_{p_j} \left| i[p_j] - i'[p_j] \right|^2, \tag{1}$$
where $p_j$ is the pixel of the image in $j^{\text{th}}$ 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 [33]) 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 $N \times N$:
$$SSIM_W\left(W_x, W_y\right) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \tag{2}$$
The quantities $\mu_x$ and $\mu_y$ are the mean pixel intensities of $W_x$ and $W_y$, $\sigma_x^2$ and $\sigma_y^2$ the variance of intensities of $W_x$ and $W_y$, and $\sigma_{xy}$ their covariance. The purpose of $c_1$ and $c_2$ is to ensure that the denominator remains far enough from 0 when both $\mu_x^2$ and $\mu_y^2$ and/or both $\sigma_x^2$ and $\sigma_y^2$ 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:
$$N_W = \left(h - 2\left\lfloor \frac{N-1}{2} \right\rfloor\right) \times \left(w - 2\left\lfloor \frac{N-1}{2} \right\rfloor\right) \times c. \tag{3}$$
The SSIM value for two images $i$ and $i'$ is the mean average of the values obtained for the $N_W$ window pairs $(i_k,\ i'_k)$:
$$SSIM(i, i') = \frac{1}{N_W} \sum_{k=1}^{N_W} SSIM_W\left(i_k, i'_k\right). \tag{4}$$
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.* [26], 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 $\mathbf{EA}_{\boldsymbol{d}}^{\mathbf{target}}$

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

$$fit_{L_2}^{\text{target}}(ind, g_i) = A_{L_2}^{\text{target}}(g_i)\boldsymbol{o}_{ind}[c_t] - B_{L_2}^{\text{target}}(g_i)L_2(ind, \mathcal{A}), \tag{5}$$

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 NN 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:

$$fit_{SSIM}^{\text{target}}(ind, g_i) = A_{SSIM}^{\text{target}}(g_i)\boldsymbol{o}_{ind}[c_t] + B_{SSIM}^{\text{target}}(g_i)SSIM(ind, \mathcal{A}). \tag{6}$$

### 3.4 The fitness function of $\mathbf{EA}_{\boldsymbol{d}}^{\mathbf{flat}}$

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 $D_{\text{flat}}$ of the "flatness" of a classification is defined by the equation (7), where flat is a vector of $\ell$ values, all set to $\text{flat}[k] = \frac{1}{\ell}$.

$$D_{\text{flat}}(ind) = \sum_{k=1}^{\ell} \Big( \left(o_{ind}[k] - flat[k]\right) \log_{10}\left(o_{ind}[k]\right) \Big)^2 \geq 0. \tag{7}$$

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:

$$fit^{\text{flat}}_{L_2}(ind, g_i) = -A^{\text{flat}}_{L_2}(g_i)D_{\text{flat}}(ind) - B^{\text{flat}}_{L_2}(g_i)L_2(ind, \mathcal{A}), \quad (8)$$

and for $d = SSIM$ by:

$$fit^{\text{flat}}_{SSIM}(ind, g_i) = -A^{\text{flat}}_{SSIM}(g_i)D_{\text{flat}}(ind) + B^{\text{flat}}_{SSIM}(g_i)SSIM(ind, \mathcal{A}). \quad (9)$$

# 4 Dataset, Neural Network Architecture and Parameters of the two EAs

## 4.1 VGG-16 trained on CIFAR-10

The feasibility study regarding the two scenarios is tested against one concrete example: CIFAR-10 and VGG-16. The dataset CIFAR-10 [14] contains $50,000$ training images and $10,000$ test images of size 32x32x3. CIFAR-10 sorts $\ell = 10$ categories (see Table 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.

**Table 1.** CIFAR-10.– For $1 \leq i \leq 10$, the $2^{\text{nd}}$ row specifies the category $c_i$ of CIFAR-10. The $3^{\text{rd}}$ 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 10, 11, or on the first row of Figure 12 in the Appendix.

| $i$ | 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 |

VGG-16 [25] is a convolutional neural network (CNN) that passes input images through 16 layers to produce a classification output. As shown in Figure 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 $3 \times 3$ and a stride of 1. Meanwhile, pooling is applied on regions of size $2 \times 2$, with no overlap.


**Fig. 1.** Architecture of VGG-16. 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 VGG-16 was initially designed for the ImageNet dataset [6], a series of adjustments were necessary for its use with the CIFAR-10 dataset [9]. The CNN used here was therefore an adapted VGG-16 architecture obtained through the steps described in [17]. Specifically, VGG-16'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 VGG-16 pre-trained on CIFAR-10 with a validation accuracy of 93.56% [9]. The same pre-trained model was used throughout the evolutionary algorithms $\mathrm{EA}_d^{\mathrm{target}}$ and $\mathrm{EA}_d^{\mathrm{flat}}$.

### 4.2 EA Parameters

The implementation of $\mathrm{EA}_d^{\mathrm{target}}$ and of $\mathrm{EA}_d^{\mathrm{flat}}$ was done from scratch, using Python 3.7 with the NumPy [20] library. Keras [5] was used to load and run the VGG-16 [25] model, and Scikit-image [32] to compute SSIM (using Scikit-image's default options). We ran our experiments on nodes with Nvidia Tesla V100 GPGPUs of the Iris HPC cluster at the University of Luxembourg [31].

Both EAs run with a population of size 160, the ancestor images being chosen from the CIFAR-10 [14] 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 1's last row, and pictured in the appendix section A.

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 al-

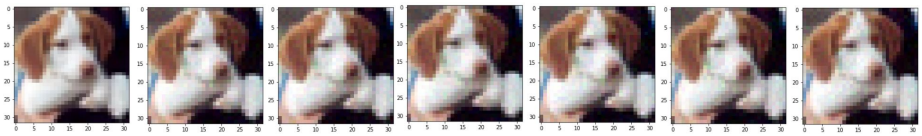

**Fig. 2.** "Target" scenario with the *dog→horse* combination.– Comparison of the original (on the left) with 3 evolved pictures created by $EA_{L_2}^{\text{target}}$ (3 next from the left), and 3 created by $EA_{SSIM}^{\text{target}}$ (3 last) at different stages, classified by VGG-16 as the target category "horse" with probabilities $6.08 \times 10^{-6}$, 0.5, 0.90, and 0.95.

together (10 source categories and 9 different 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.

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

To illustrate, Figure 2 shows an original image in the category "dog" and evolved images classified by VGG-16 as the target category "horse" 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 3 shows the graph of the source "dog" and target "horse" class probabilities obtained along the evolution referred to in Figure 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.

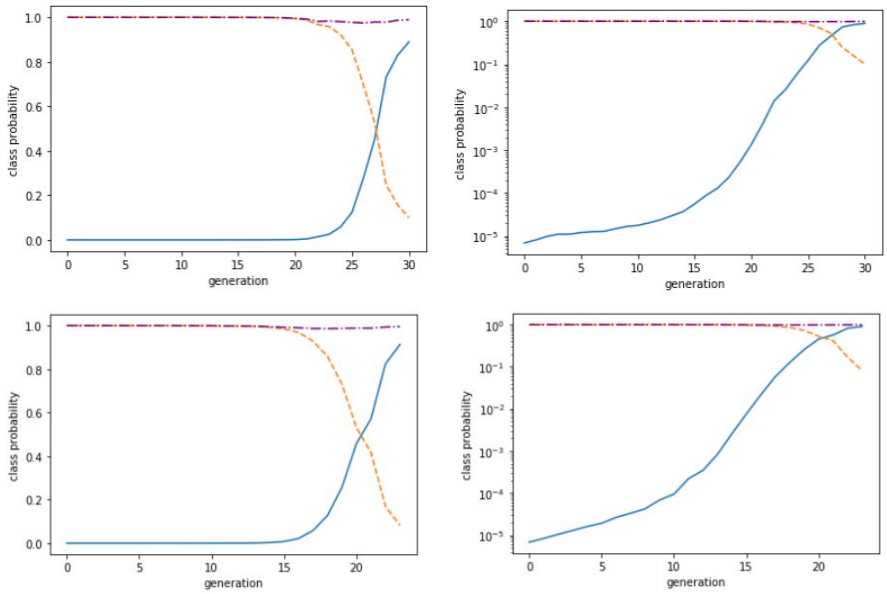


**Fig. 3.** "Target" scenario with the *dog→horse* combination of Figure 2.– Linear (left) and logarithmic (right) scale plots of an evolution of source (dashed line) and target (solid line) class probabilities using the $L_2$ norm (top) or using SSIM (bottom), with respect to the generation number. 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 shows examples of the four possible (*source, target*) combinations created by $EA_d^{\text{target}}$, 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 2.

**Table 2.** Number of generations required by $\text{EA}_d^{\text{target}}$ to evolve the four *animals* and *objects* combinations of images of Figure 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 |

**Fig. 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 $\mathrm{EA}_d^{\mathrm{target}}$ for $d = L_2$ (2nd image) and $d = SSIM$ (3rd image). At the top, *animal*→*animal* and *animal*→*object* combinations: From left to right, VGG-16 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.

### 5.1 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 section (in Figure 10 with $L_2$ and Figure 11 with SSIM). The average computation time per generation is $0.03 \pm 0.01$s for $L_2$ and $0.15 \pm 0.02$s 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 3 presents the mean number of generations for the 4 combination types for the 90 evolutions of Figure 10 and of Figure 11 in Appendix.

**Table 3.** Mean value and standard deviation for the number of generations required by $\mathrm{EA}_d^{\mathrm{target}}$ 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 10 and Figure 11 in Appendix.

| | 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 $\mathrm{EA}_{L_2}^{\mathrm{target}}$ and by $\mathrm{EA}_{SSIM}^{\mathrm{target}}$, neither appears in general to be clearly better than the other (see Figures 2 and 4 for example, as well as Figures 10 and 11 in Appendix) 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 $\mathrm{EA}_{d}^{\mathrm{target}}$ (for $d = L_2$ and $d = SSIM$) achieved the objective set in the "target" scenario. Adversarial descendant images were constructed, which VGG-16 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. Section 5.2 addresses this issue specifically.

### 5.2 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 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 NN, since they are only vulnerable to targeted noise [8]). 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 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 6 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 [15] $KL$ 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 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

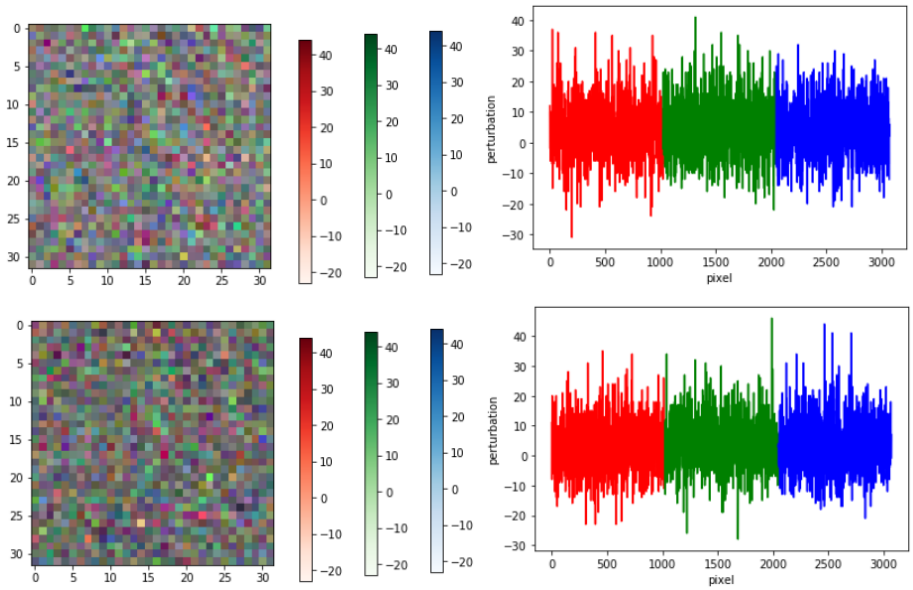


**Fig. 5.** "Target" scenario.– Difference between ancestor and descendant images obtained by $\mathrm{EA}_d^{\mathrm{target}}$ ($d = L_2$ in the top row, $d = SSIM$ in the bottom row) for the *truck*→*frog* combination of Figure 4. The image on the left gives an idea of the spatial distribution of the 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.

patterns are indeed very similar.

However, the SSIM results are more symmetrical than the $L_2$-norm ones (see Figure 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 6 for an example) indicate that $\mathrm{EA}_{L_2}^{\mathrm{target}}$ tends to leave more pixels unchanged than $\mathrm{EA}_{SSIM}^{\mathrm{target}}$.

## 6 Running $\mathrm{EA}_d^{\mathrm{flat}}$: Examples, Results and Discussion

Figure 7 shows an ancestor image in the "dog" category (category $c_6$ of Table 1), and descendant evolved images reached by $EA_d^{\mathrm{flat}}$ 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 VGG-16 for these evolved images are given in row $c_6$

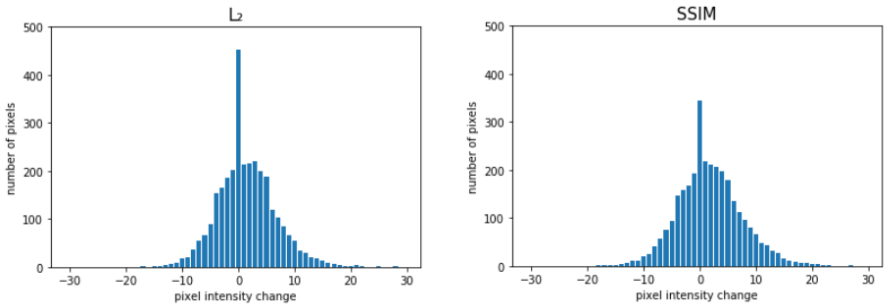


**Fig. 6.** "Target" scenario for the *dog*→*horse* evolution.– Changes in pixel intensity of the dog ancestor shown in Figure 2, with the $L_2$-norm (left) and SSIM (right). To reduce the possible impact of random fluctuations on the results, the figures are averaged on six runs. In both cases, the 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.

of Table 4 for $L_2$ and of Table 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 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 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 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 $c_4$).

### 6.1 Results

Depending on the chosen source category, reaching almost flatness required between 142 and 552 generations for the 10 altogether evolutions, as specified in the Appendix section (Table 9). The average time required per generation was $0.04\pm0.01$s with $L_2$ and $0.17\pm0.02$s with SSIM. For the "flattening" process, the "horse" category took the fewest number of generations, and the "deer" category

**Table 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 $i^{\text{th}}$ position on the $2^{\text{nd}}$ row of Figure 12. For $1 \leq j \leq 10$, the value given on the $j^{\text{th}}$ column is the label value for the category $c_j$ output by VGG-16 for this adversarial image. The column $D_{\text{flat}}$ gives the value of the function $D_{\text{flat}}$ for the descendant "flat" image obtained by $EA_{L_2}^{\text{flat}}$. The columns $\Delta^+$ and $\Delta^-$ indicate the maximal deviation exceeding 0.100 from above or from below in the row.

| | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $c_7$ | $c_8$ | $c_9$ | $c_{10}$ | $D_{\text{flat}}$ | $\Delta^+$ | $\Delta^-$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $c_1$ | 0.102 | 0.115 | 0.102 | 0.083 | 0.111 | 0.113 | 0.092 | 0.092 | 0.096 | 0.093 | $1 \times 10^{-3}$ | 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 | $4 \times 10^{-4}$ | 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 | $4 \times 10^{-5}$ | 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 | $7 \times 10^{-4}$ | 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 | $1 \times 10^{-6}$ | 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 | $5 \times 10^{-4}$ | 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 | $8 \times 10^{-6}$ | 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 | $2 \times 10^{-4}$ | 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 | $3 \times 10^{-4}$ | 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 | $2 \times 10^{-6}$ | 0.001 | 0.001 |

**Table 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 $EA_{SSIM}^{\text{flat}}$ and pictured on the $i^{\text{th}}$ position on the $3^{\text{rd}}$ row of Figure 12. For $1 \leq j \leq 10$, the value given on the $j^{\text{th}}$ column is the label value for the category $c_j$ output by VGG-16 for this adversarial image. The column $D_{\text{flat}}$ gives the value of the function $D_{\text{flat}}$ for the descendant "flat" image obtained by $EA_{SSIM}^{\text{flat}}$. The columns $\Delta^+$ and $\Delta^-$ indicate the maximal deviation exceeding 0.100 from above or from below in the row.

| | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $c_5$ | $c_6$ | $c_7$ | $c_8$ | $c_9$ | $c_{10}$ | $D_{\text{flat}}$ | $\Delta^+$ | $\Delta^-$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $c_1$ | 0.106 | 0.114 | 0.105 | 0.082 | 0.105 | 0.113 | 0.090 | 0.091 | 0.101 | 0.093 | $1 \times 10^{-3}$ | 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 | $9 \times 10^{-4}$ | 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 | $1 \times 10^{-3}$ | 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 | $1 \times 10^{-3}$ | 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 | $1 \times 10^{-3}$ | 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 | $1 \times 10^{-3}$ | 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 | $1 \times 10^{-3}$ | 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 | $1 \times 10^{-3}$ | 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 | $1 \times 10^{-3}$ | 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 | $1 \times 10^{-3}$ | 0.014 | 0.016 |

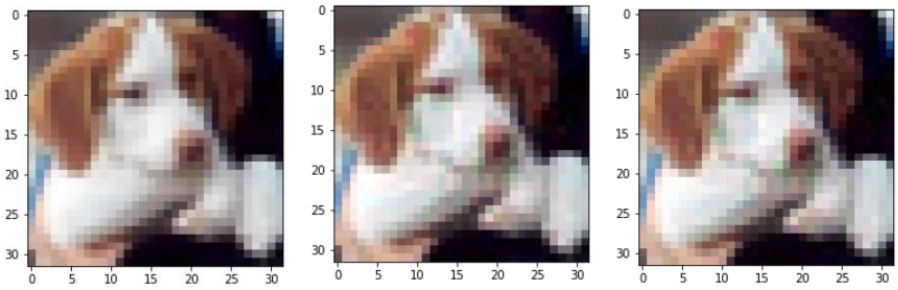

**Fig. 7.** "Flat" scenario with the *dog* original image.– Comparison of the original (on the left) with 2 evolved pictures created by $EA_{L_2}^{\text{flat}}$ and $EA_{SSIM}^{\text{flat}}$. The number of generations required by the two evolutions was 238 and 233.

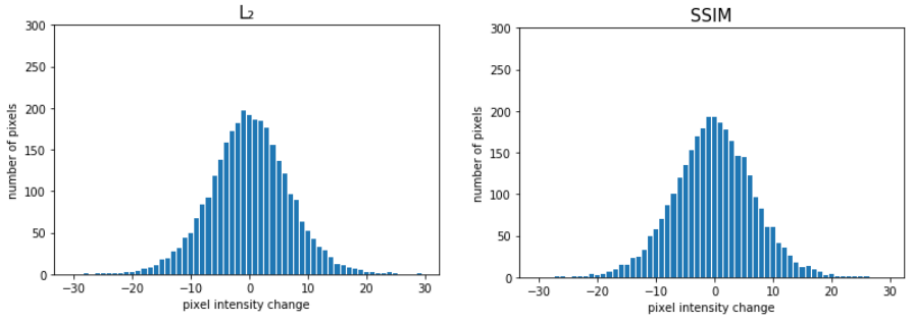


**Fig. 8.** "Flat" scenario for the *dog* original image.– Changes in pixel intensity of the dog ancestor shown in Figure 7, with the $L_2$-norm (left) and SSIM (right). To reduce the possible impact of random fluctuations on the results, the figures are averaged on six runs. The Kullback-Leibler divergence computed between the $L_2$-norm and SSIM probability densities is 0.004; Reversing this order also leads to 0.004.

the largest number of generations, at least with the ancestor pictures taken in these categories.

When one compares the adversarial images created with $EA_{L_2}^{\text{flat}}$ and those created with $EA_{SSIM}^{\text{flat}}$, neither appears better than the other (see Figure 7 for an example of the flattening of an ancestor in the "dog" category, and Figure 12 in Appendix for examples for all categories) for the human eye. Both EAs produced adversarial images remaining very close to the ancestor image.

Tables 4 and 5 display the label values of VGG-16 for the 10 "flattened" images of Figure 12 created by $EA_d^{\text{flat}}$, with $d = L_2$ and SSIM. The amplitudes $\Delta^+$ and $\Delta^-$ 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 $\pm 0.001$), they can achieve far larger values ($+0.015$ and $-0.017$ for $L_2$; $+0.025$

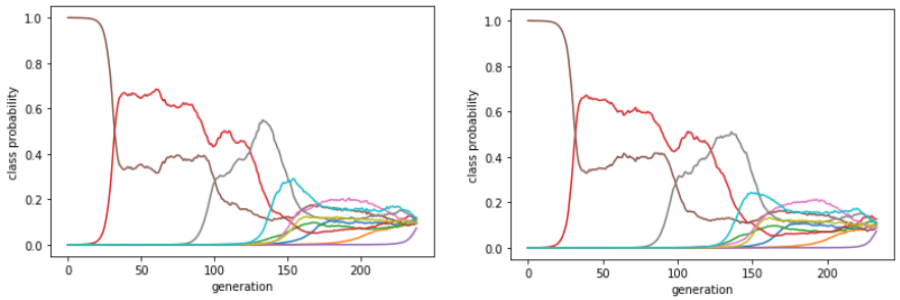


**Fig. 9.** "Flat" scenario with the *dog* original image of Figure 7.– Linear scale plots of an evolution of the 10 class probabilities using the $L_2$ norm (left) or using SSIM (right), with respect to the generation number.

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 $\Delta^+$ and $\Delta^-$ indeed makes our point. We nevertheless come back to this aspect in the conclusion part.

### 6.2 Visualising 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 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 8 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 9 for the "flat" scenario with the *dog* ancestor pictured in Figure 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.

## 7 Conclusions and Future Work

Pursuing the research program announced in [3], this work substantially complements [4] 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 neural network VGG-16 [25], trained on the dataset CIFAR-10 [14] 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.

While we consider that our point is fully made by $EA_d^{\mathrm{target}}$ in the context of the "target" scenario, at least for VGG-16 and images from CIFAR-10, we intend to perform a more in-depth study of the "flat" scenario, let alone because this latter scenario seems harder to fulfill than the former one. We would like to further explore the balance between the three following components: the size of the $\Delta^+$ and $\Delta^-$ values given in Tables 4 and 5, the proximity between the evolved pictures and the ancestor pictures, and the size of the pictures. This study, that may lead to different penalty values $A_d^{\mathrm{flat}}(g_i)$ and $B_d^{\mathrm{flat}}(g_i)$, will not be limited to pictures of CIFAR-10 size, but will consider larger pictures as well. Said otherwise, such a study may provide an indication about the maximum amplitudes of the $\Delta^+$ and $\Delta^-$ values around $\frac{1}{\ell}$ that one can not diminish without compromising the proximity of the descendant pictures with the ancestor pictures. It would be interesting to get a heuristic bound on this amplitude, not only with respect to $\frac{1}{\ell}$, but also in terms of the size $n \times n$ of the pictures considered, with other $(n, \ell)$ values than the $(32, 10)$ case of VGG-16 trained on CIFAR-10 considered in this paper.

For both scenarios, our current confidence that the similarity aspect between the original and adversarial images is indeed satisfied is limited in the following sense. While all three authors of this paper classified the modified images as belonging to their original categories, three people represent a small sample. Therefore, part of an on-going work is to conduct a statistically significant study to check whether our evolutionary algorithms do what we expect from

them, even if the three of us believe that it does.

Although our algorithms successfully managed to deceive the neural network, while producing adversarial images similar to the original, a closer comparison of the original and modified images reveals the noisiness of the evolved images. This noisiness is noticeable here as CIFAR-10 images have a low resolution. We observed in this paper, that the noise, although appearing to be random and evenly distributed across the image, is targeted. This leads to three additional research directions, that we plan to explore.

The first direction aims at explaining why these small perturbations of the input are able to produce misclassifications with high confidence. In particular, results for the "target" scenario do not indicate that $\mathrm{EA}_d^{\mathrm{target}}$ creates a shape pattern to fool VGG-16, but rather acts on texture. Note that this hypothesis is supported by the phenomenon observed on the evolved images created by $\mathrm{EA}_d^{\mathrm{flat}}$ in the "flat" scenario as well (with the order of the successive "rising categories" being close to the ancestor category from a texture point of view).

The second direction aims at producing adversarial images, that are almost entirely indistinguishable from the original for a human eye. Towards this goal, several steps could be taken, such as optimizing the EAs with different mutations or using other datasets, notably with larger images. As a complement to this, one of the main advantages of our evolutionary algorithms is the ability to treat the neural network as a black box, with no required knowledge of its architecture or parameters. We hence intend to extend the present study to more CNNs (Inception [27], etc.) trained on larger datasets (CIFAR-100 [14], ImageNet [6], etc.) with images of larger resolution.

The third direction aims at confronting the resistance to filters (e.g. denoising filters like the median filter) of the adversarial images constructed by our EAs compared to adversarial images obtained by other methods, for instance those of [26].

Once these studies performed, we will be armed to conduct a thorough comparison between our evolutionary algorithm approach and other adversarial image generation approaches listed for instance in the survey [1].

## References

1. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: A survey. CoRR **abs/1801.00553** (2018), `http://arxiv.org/abs/1801.00553`
2. Bernard, N., Leprévost, F.: Evolutionary algorithms for convolutional neural network visualisation. In: High Performance Computing – 5th Latin American Conference, CARLA 2018 (Bucaramanga, Colombia, Sep 23-28, 2018). Communications

in Computer and Information Science, vol. 979, pp. 18–32. Springer, Heidelberg (2018)
3. 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)
4. 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)
5. Chollet, F.: Keras. GitHub code repository (2015-2018), `https://github.com/fchollet/keras`
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: The imagenet image database (2009), `http://image-net.org`
7. Fawzi, A., Fawzi, H., Fawzi, O.: Adversarial vulnerability for any classifier. CoRR **abs/1802.08686** (2018), `http://arxiv.org/abs/1802.08686`
8. 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`
9. Geifman, Y.: cifar-vgg (2018), `https://github.com/geifmany/cifar-vgg`
10. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, Massachusetts (2016), `http://www.deeplearningbook.org`
11. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2014), arXiv preprint arXiv:1412.6572
12. Javidi, B.: Image recognition and classification: algorithms, systems, and applications. Taylor & Francis Group (2002)
13. Jong, K.A.D.: Evolutionary Computation: A Unified Approach. A Bradford book, MIT Press, Cambridge, Massachusetts (2006)
14. Krizhevsky, A., et al.: The CIFAR datasets (2009), `https://www.cs.toronto.edu/~kriz/cifar.html`
15. Kullback, S., Leibler, R.: On information and sufficiency. The Annals of Mathematical Statistics **22**, 79–86 (1951)
16. Li, X., Chen, Y., He, Y., Xue, H.: Advknn: Adversarial attacks on $k$-nearest neighbor classifiers with approximate gradients. CoRR **abs/1906.06591** (2019), `http://arxiv.org/abs/1906.06591`
17. Liu, S., Deng, W.: Very deep convolutional neural network based image classification using small training sample size. IAPR (2015)
18. Moosavi Dezfooli, S.M., Alhussein, F., Pascal, F.: Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2574–2582. IEEE (2016)
19. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 427–436 (2015)
20. Oliphant, T.E.: A guide to NumPy. Trelgol Publishing USA (2006)
21. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP). pp. 582–597. IEEE (2016)

22. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. pp. 506–519. ACM (2017). https://doi.org/10.1145/3052973.3053009, http://doi.acm.org/10.1145/3052973.3053009
23. Shafahi, A., Huang, W.R., Studer, C., Feizi, S., Goldstein, T.: Are adversarial examples inevitable? CoRR **abs/1809.02104** (2018), http://arxiv.org/abs/1809.02104
24. Shamir, A., Safran, I., Ronen, E., Dunkelman, O.: A simple explanation for the existence of adversarial examples with small hamming distance. CoRR **abs/1901.10861** (2019), http://arxiv.org/abs/1901.10861
25. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014), http://arxiv.org/abs/1409.1556
26. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. CoRR **abs/1710.08864** (2017)
27. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. arXiv **1409.4842** (2014), https://arxiv.org/pdf/1409.4842.pdf
28. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (2013), arXiv:1312.6199
29. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the 2014 IEEE conference on computer vision and pattern recognition (CVPR). pp. 1701–1708. IEEE (2014)
30. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction apis. In: Proceedings of the 25th USENIX Security Symposium Austin, TX, USA August 10–12, 2016. pp. 601–618. Usenix (2016)
31. Varrette, S., Bouvry, P., Cartiaux, H., Georgatos, F.: Management of an academic HPC cluster: The UL experience. In: Proceedings of the 2014 International Conference on High Performance Computing & Simulation (HPCS 2014). pp. 959–967. IEEE, Bologna, Italy (July 2014), see also https://hpc.uni.lu.
32. 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
33. 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)
34. Yang, Y., Rashtchian, C., Wang, Y., Chaudhuri, K.: Adversarial examples for non-parametric methods: Attacks, defenses and large sample limits. CoRR **abs/1906.03310** (2019), http://arxiv.org/abs/1906.03310
35. Yu, X., Gen, M.: Introduction to Evolutionary Algorithms. Springer, Heidelberg (2010)

# A Appendix

## A.1 "Target" scenario

**Table 6.** "Target" scenario.– For $i \neq j$, the element at the intersection of the $i^{\text{th}}$ row and $j^{\text{th}}$ 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 $i^{\text{th}}$ diagonal position in Figure 10 (and Figure 11). 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 7.** "Target" scenario.– The pair of integers at the intersection of the $i^{\text{th}}$ row and $j^{\text{th}}$ 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 10 with $L_2$ (left-hand side of the pair) and in Figure 11 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) | |

## A.2 "Flat" scenario

**Table 8.** "Flat" scenario.– For $1 \leq i \leq 10$, the element in $i^{\text{th}}$ position in the $2^{\text{nd}}$ 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 $c_i \rightarrow$ "flat" evolution of the ancestor $\mathcal{A}_i$ on $i^{\text{th}}$ position on the first row in Figure 12. 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.** "Flat" scenario.– The pair of integers on the $2^{\text{nd}}$ row represents the number of generations necessary to create the adversarial image in the evolution $c_i \rightarrow c_j$, as specified in Figure 12 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) |

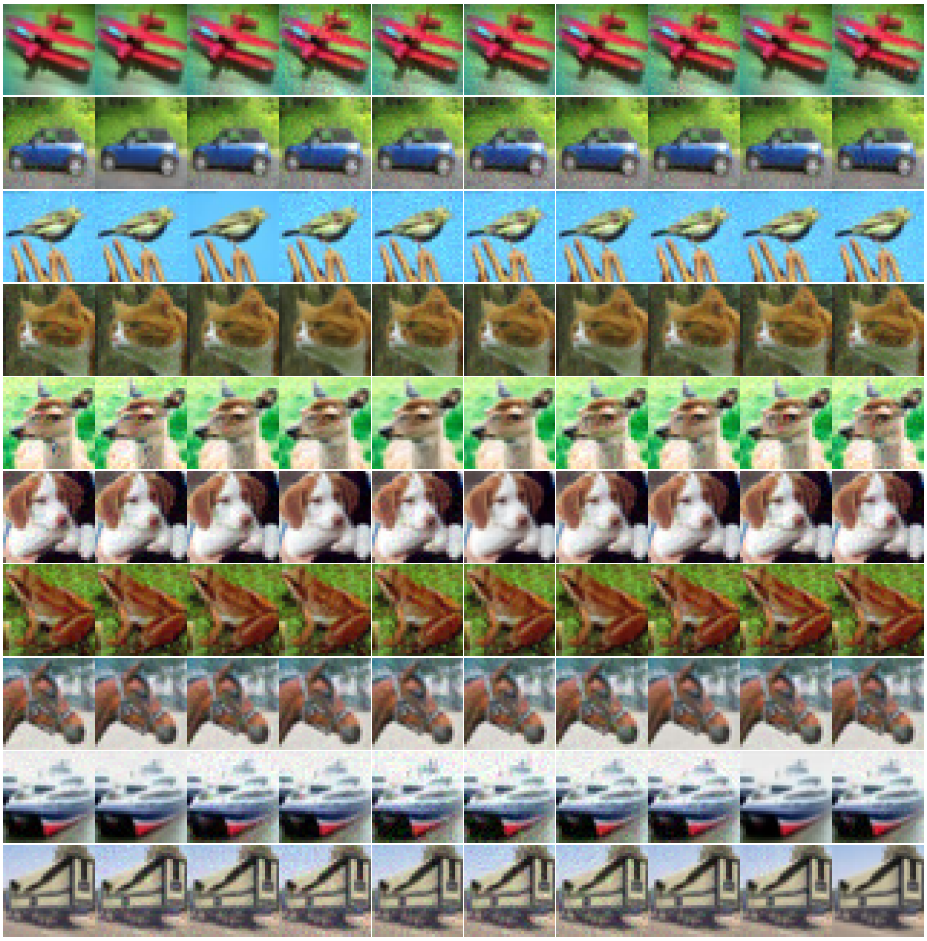

**Fig. 10.** "Target" scenario, case $L_2$.– Pictures on the diagonal are the ancestors $\mathcal{A}_i$ belonging to the category $c_{\mathcal{A}_i} = c_i$, for $1 \leq i \leq 10$. On each row $1 \leq i \leq 10$, the picture on the $j^{\text{th}}$ column, with $j \neq i$, is the descendant picture $\mathcal{D}_{ij}$, obtained by applying $\text{EA}_{L_2}^{\text{target}}$ to $\mathcal{A}_i$, that VGG-16 classifies as belonging to $c_j$.

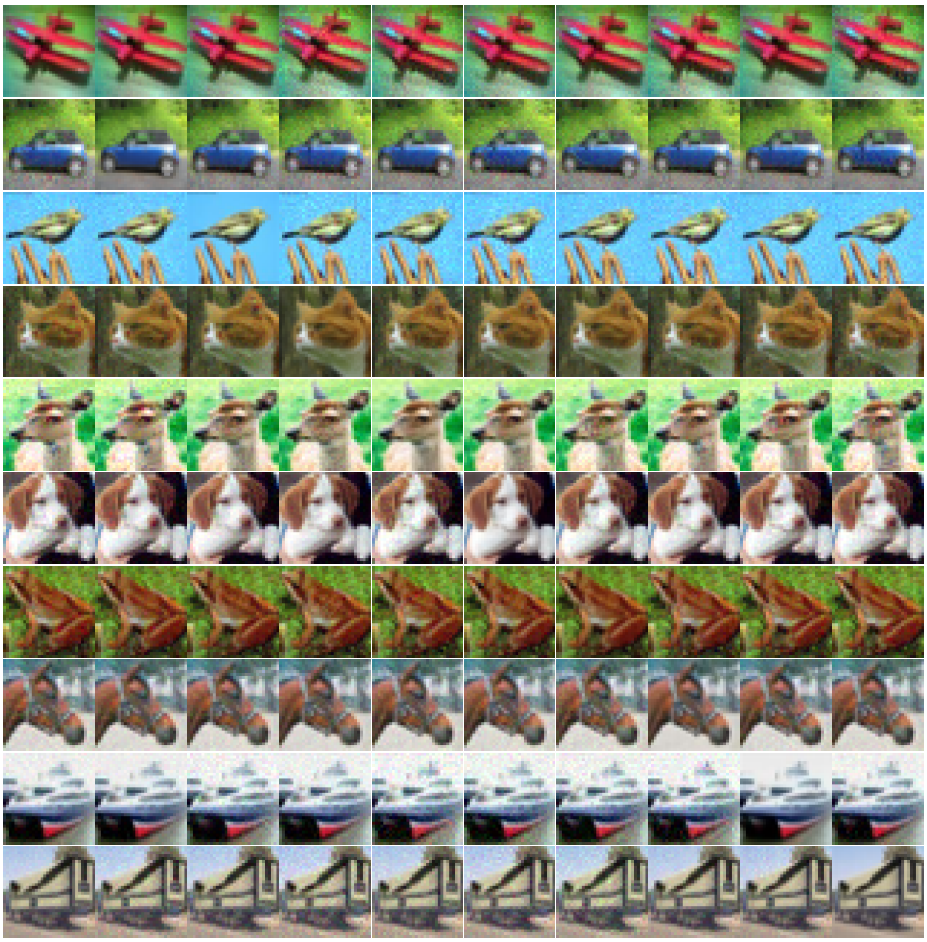

**Fig. 11.** "Target" scenario, case $SSIM$.– Pictures on the diagonal are the ancestors $\mathcal{A}_i$ belonging to the category $c_{\mathcal{A}_i} = c_i$, for $1 \leq i \leq 10$. On each row $1 \leq i \leq 10$, the picture on the $j^{\text{th}}$ column, with $j \neq i$, is the descendant picture $\mathcal{D}_{ij}$, obtained by applying $\text{EA}_{SSIM}^{\text{target}}$ to $\mathcal{A}_i$, that VGG-16 classifies as belonging to $c_j$.

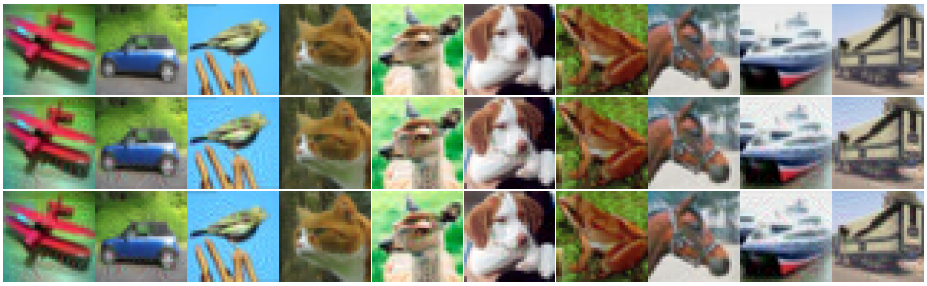

**Fig. 12.** "Flat" scenario.– Pictures on the 1st row are the ancestors $\mathcal{A}_i$ belonging to the category $c_{\mathcal{A}_i} = c_i$, for $1 \leq i \leq 10$ (they are the same as those on the diagonals of Figures 10 and 11). For $1 \leq i \leq 10$, the picture in $i^{\text{th}}$ position on the 2nd row is the adversarial descendant picture obtained by applying $\text{EA}_{L2}^{\text{flat}}$ to $\mathcal{A}_i$, and that VGG-16 is unable to classify with certainty. Mutatis mutandis 3rd row with $\text{EA}_{SSIM}^{\text{flat}}$.