Multi-objective differential evolution in the generation of adversarial examples

Bartlett, Antony; Liem, Cynthia C.S.; Panichella, Annibale

DOI
10.1016/j.scico.2024.103169

Publication date
2024

Document Version
Final published version

Published in
Science of Computer Programming

Science of Computer Programming 238 (2024) 103169

Contents lists available at ScienceDirect

Science of Computer Programming

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

Original Software Publication

Multi-objective differential evolution in the generation of adversarial examples

Antony Bartlett [*], Cynthia C.S. Liem, Annibale Panichella

Delft University of Technology (TU Delft), Mekelweg 5, 2628 CD, Delft, Netherlands

ARTICLE INFO

Keywords:
Differential evolution
Adversarial examples
Deep learning search-based software engineering
Software testing

ABSTRACT

Adversarial examples remain a critical concern for the robustness of deep learning models, showcasing vulnerabilities to subtle input manipulations. While earlier research focused on generating such examples using white-box strategies, later research focused on gradient-based black-box strategies, as models' internals often are not accessible to external attackers. This paper extends our prior work by exploring a gradient-free search-based algorithm for adversarial example generation, with particular emphasis on differential evolution (DE). Building on top of the classic DE operators, we propose five variants of gradient-free algorithms: a single-objective approach (GA_{DE}), two multi-objective variations ($\text{NSGA-II}_{\text{DE}}$ and $\text{MOEA/D}_{\text{DE}}$), and two many-objective strategies ($\text{NSGA-III}_{\text{DE}}$ and $\text{AGE-MOEA}_{\text{DE}}$). Our study on five canonical image classification models shows that whilst GA_{DE} variant remains the fastest approach, $\text{NSGA-II}_{\text{DE}}$ consistently produces more minimal adversarial attacks (i.e., with fewer image perturbations). Moreover, we found that applying a post-process minimization to our adversarial images, would further reduce the number of changes and overall delta variation (image noise).

Code metadata

Code metadata description	
Current code version	*v2.0*
Permanent link to code/repository used for this code version	*https://github.com/ScienceofComputerProgramming/SCICO-D-23-00377* [1]
Permanent link to Reproducible Capsule	*https://doi.org/10.5281/zenodo.10250866 (See README.md)*
Legal Code License	*GNU*
Code versioning system used	*GIT*
Software code languages, tools, and services used	*Python 3.10*
Compilation requirements, operating environments and dependencies	*Pip, pipenv, docker*
Link to developer documentation/manual	*https://doi.org/10.5281/zenodo.10250866 (See README.md)*
Support email for questions	*a.j.bartlett@tudelft.nl*

* Corresponding author.
E-mail addresses: a.j.bartlett@tudelft.nl (A. Bartlett), c.c.s.liem@tudelft.nl (C.C.S. Liem), a.panichella@tudelft.nl (A. Panichella).

https://doi.org/10.1016/j.scico.2024.103169
Received 2 December 2023; Received in revised form 30 June 2024; Accepted 4 July 2024

Available online 8 July 2024

1. Introduction

Modern applications increasingly rely on machine learning models for automated decision-making based on observed data patterns.

Despite early challenges in processing high-dimensional but semantically low-level multimedia data (such as raw pixel values in images) [2], the computational surge and increasing popularity of deep learning has enabled the direct learning of associations between input data and output labels. Ever since so-called deep convolutional neural networks outperformed hand-crafted methods in the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [3], deep learning models have become mainstream in computer vision, but also in many other applied machine learning domains, such as natural language processing and music information retrieval.

Deep learning models have been lauded for yielding high-accuracy predictions and thus, have become attractive candidates for integration in real-life systems that may be safety-critical (e.g. vision components in self-driving cars). At the same time, they have been criticized for making intolerable and sometimes incomprehensible prediction errors, jeopardizing safety. As has been shown in the machine learning world, they are e.g. inherently vulnerable to so-called adversarial attacks, in which perceptually small changes to input data can cause very different, erroneous model predictions [4,5].

Adversarial examples have been extensively investigated in the literature, where the idea is to introduce subtle changes in the data (e.g., changing the pixels in a target image) that do not change the ground truth but make a Deep Learning model predict incorrect output. Existing approaches to adversarial example generation can be classified into white-box and black-box methods. White-box approaches [6–11] require access to the model under test (i.e. the model architecture, neuron weight values, and gradients). Black-box strategies [12–16], instead, only require access to model inputs and outputs. These approaches are considered more realistic as it reflects what external attackers can obtain [16], e.g., in the case of remote API access.

In this domain, Lin et al. [16] introduced the *Black-box Momentum Iterative Fast Gradient Sign Method* (BMI-FGSM) to approximate the gradient, based on a few data points. These points are obtained by mutating existing images (seeds) with *evolutionary algorithm* (EA), and *differential evolution* (DE) in particular. Their results showed that BMI-FGSM outperforms white-box and black-box approaches previously proposed in the literature.

In spite of these undisputed results, we observe that BMI-FGSM requires thousands of iterations to successfully generate adversarial attacks, despite the usage of gradient-based methods. In our previous work [17], we compared the state-of-the-art BMI-FGSM to our single and multi-objective black-box attack methods, across five image recognition models. This experiment was run across a small set of images (50), to which we discovered the following. Our single-objective approach performed well in quickly creating adversarials, whilst the multi-objective approach would take slightly longer but create adversarial examples with fewer perturbations.

Through this comparative analysis, we established that employing pure black-box attacks using differential evolution was considerably faster than previous adversarial example generation. Further to this, we were able to generate adversarial examples with significantly fewer perturbations (pixel changes).

Therefore, this paper maintains its focus on black-box strategies targeting deep neural network (DNN) models for image recognition. We investigate the usage of DE alone—i.e., without employing gradient-based methods—as the core technique to generate adversarial attacks in a black-box fashion. In particular, we utilize five customized operators for DE that introduce perturbations in the target images. Notably, we focus on DE as meta-heuristics due to their reported superiority over gradient-based approaches, as observed in recent studies [18].

This work extends the previous by adding a second multi-objective variant *MOEA/D*[1] [19] (MOEA/D$_{\text{DE}}$), along with two many-objective variants, *NSGA-III*[2] [20] (NSGA-III$_{\text{DE}}$) and *AGE-MOEA*[3] [21] (AGE-MOEA$_{\text{DE}}$). These additions, particularly the many-objective variants utilized as part of our multi-objective solution, expand the scope of our investigation.

Additionally, we expand our tests, sampling an extra 100 images from the 2012 ImageNet validation dataset. This culminates in a total of 150 images, with that being the original 50 images and the new 100. We have continued sampling from this particular dataset, as it is the validation data released in conjunction with the data, on which the models under test have been trained.

Subsequently, our exploration delves into the application of post-processing techniques on generated adversarial examples. We employed a randomized minimization algorithm designed to gradually revert the changes applied to the original images while preserving the adversarial capabilities in fooling the targeted DNNs. The goal of this algorithm is to generate a set of less noticeable changes, whilst maintaining a flipped prediction.

We thus focus on the following two research questions:

RQ1 : *How do various black-box evolutionary algorithms perform in generating adversarial examples?*
RQ2 : *How effective is post-processing at improving the quality of our adversarial examples, by minimizing pixel color delta variation?*

2. Related work

White-box testing [6–11] has been extensively researched, but needs internal model access, which is not always realistic in practice. Therefore, we will adopt a black-box testing approach instead, which purely focuses on modifying a system's input (in our

[1] https://pymoo.org/algorithms/moo/moead.html.
[2] https://pymoo.org/algorithms/moo/nsga3.html.
[3] https://pymoo.org/algorithms/moo/age.html.

case, an image) to trigger undesired changes in the system's output (in our case, the object classification for the input image). We will employ evolutionary strategies for this; beyond images, these have e.g. been proposed on credit scoring models [22] and speech audio [23,24].

In literature, various black-box attacks on image recognition DNNs have been proposed [12,13,16]. Nguyen et al. [12] generate random images that are noise to humans, but are misclassified as actual objects by a DNN. In our case, we will seek more adversarial examples, where input is kept as close to the original as possible, and thus human-recognizable.

Zhou et al. [13] proposed a hybrid black-box approach that combines EAs with the bisection method. The images are mutated by injecting full black or white pixels. Instead, Chan and Cheng [25] introduced a black-box approach that adds Gaussian noise to large portions of the images. Besides, their work targets object detection models rather than image recognition. In contrast, our paper investigates the adversarial example generation in a multi-objective variant where both (1) the model misclassification and (2) the number of changed pixels are taken into account.

Several works explicitly focused on minimizing perturbations, such that fewer modifications to an image would already lead to different system output. One example of this is the work by Suzuki et al. [14], which proposes a Discrete Cosine Transform-based method for modifying images. While such perturbations parametrically are small, they still will affect many pixels at once. A similar consideration holds for the work by Sun et al. [15], focusing on minimum visibility of the modification from a perceptual perspective, but not explicitly constraining the number of pixels to modify.

On the other end of the extreme, one may search for attacks that modify as few pixels as possible (and as such will naturally not stand out, when compared to the total amount of pixels in an image). For example, Su et al. [26] propose a single-pixel adversarial attack using DE, executed against the classical CIFAR-10[4] [27] and ImageNet object classification datasets. Comparing the results on these two datasets, a high success rate is reported for CIFAR-10, but this success rate is much lower for ImageNet, where single-pixel attacks mainly succeed in situations where the model's original classification of the image was already quite low. This may have to do with the difference in search space; the test images in CIFAR-10 are much smaller ($32 \times 32 = 1024$ pixels) than those in ImageNet ($224 \times 224 = 50,176$ pixels).

A stronger, yet compact attack is proposed by Lin et al. [16], who combined DE [28] and the Fast Gradient Sign Method [29] for black-box adversarial sample generation. Executing a single-objective attack called `Black-box Momentum Iterative Fast Gradient Sign Method (BMI-FGSM)`, to generate an efficient and effective perturbation that is similar to the benign input. Their approach utilizes double-step size and candidate reuse whilst approximating the gradient direction. An initial gradient sign population is generated using DE. The input is then gradually modified using gradient sign approximation until an adversarial example is created that is visibly the same as the original input, but now classified as something different. Lin et al. [16] showed that `BMI-FGSM` successfully generates adversarial examples for large models, outperforming other state-of-the-art white-box and black-box approaches.

While Lin et al. [16] showed that black-box approaches based on Evolutionary Algorithms (EAs) can be very competitive with their white-box alternatives, existing approaches have various drawbacks. First, `BMI-FGSM` requires a large number of iterations (in the order of thousands) and population size (hundreds of individuals). In other words, attackers need to query the model under attack many times, increasing the chances of detection. Second, `BMI-FGSM` combines multiple techniques, making its implementation less trivial and introducing more hyper-parameters to tune. Finally, the generated attacks are not minimal, i.e., the prediction flip is achievable but requires changing all pixels in the original image (or seed).

Deepfool by Moosavi-Dezfooli et al. [29] sits in a grey area, as it discusses gradient approximation in their approach, but does not specifically define a black or white-box approach. Moreover, they do not show many examples of complex image perturbations and focus solely on comparisons between itself and the Fast Gradient Sign Method (`FGSM`) [5] on the `MNIST` and `CIFAR-10` datasets. Similar to our approach, they attempt to create a minimal adversarial example. However, their approach is effectively a gradient descent algorithm with an adaptive step size. Additionally, when validated against the `ILSVRC2012` (the dataset used in this paper), they report an increasingly higher error rate.

With these approaches in mind, we have decided on a black-box approach, which does not require an understanding of the model's inner workings. We simply need to understand the input required and craft it accordingly whilst monitoring the output for discrete changes.

In this paper, we extend our previous work [17], wherein we showed a black-box approach, based purely on DE, could produce adversarial examples with minimal perturbations applied, whilst taking considerably less time than existing methods. We introduce three more multi-objective approaches to compare their effectiveness in generating minimal adversarial examples. Furthermore, we introduce a post-processing approach that reduces the noticeability of our adversarial examples by adjusting the applied changes back towards their original image RGB values.

3. Approach

Without loss of generality, an image classifier f is a mathematical function/model $f : I \longrightarrow L \times \mathbb{R}^n$, which takes as input an image $i \in I$ and returns a label $l \in L$ and a confidence vector $conf \in \mathbb{R}^n$, which contains the probabilities associated with all labels $l \in L$ in descending order. Here, i is an image of size 224×224 pixels, where each pixel in the image is represented by an RGB integer

[4] https://www.tensorflow.org/datasets/catalog/cifar10.

value in the range 0 to 256. The first element $conf_1$ is the probability associated with the most likely (predicted) label l, while the remaining entries in $conf_2 \dots conf_n$ are related to the other possible labels $\in L$.

We can now reformulate adversarial attack generation as a search problem:

Definition 1. Let $f : I \longrightarrow L \times \mathbb{R}^n$ be a trained model that takes as input an image $i \in I$ and returns a predicted label $l \in L$ and a confidence vector $conf \in \mathbb{R}^n$. Let $m : I \longrightarrow I$ be a transformation function that mutates (i.e. applies changes to) an image $i \in I$. The problem is finding a mutated image $m(i)$ such that $f(m(i)) \neq f(i)$, with the constraints that both i and $m(i)$ share the same correct label (same ground truth/oracle).

Attack generation strategies can be *targeted* or *un-targeted*. The former aims to flip the prediction to a specific label or classification outcome, while the latter aims to lead the model toward producing any incorrect outcome.

In this paper, we focus on un-targeted attack generation: for demonstrating the vulnerability of a machine learning model, it is sufficient to generate mutated images $m(i)$ that flip the predicted output to any other label than the ground truth label. Since a classification model returns both the label and the corresponding confidence level, we can use the latter to guide the search toward the flipped prediction. More precisely, given a classification model $f : I \longrightarrow L \times \mathbb{R}^n$, a seed image i, and its mutated variant $m(i)$, we optimize the following objective:

$$\min O_1 = \begin{cases} f(m(i))_1^{conf} - f(m(i))_2^{conf} & \text{if } f(i)_1^l = f(m(i))_1^l \\ -f(m(i))_1^{conf} & \text{if } f(i)_1^l \neq f(m(i))_1^l \end{cases} \tag{1}$$

In other words, this objective aims to reduce the confidence for the most likely prediction/label ($f(m(i))_1^{conf}$), while increasing the confidence for the second-most-likely prediction ($f(m(i))_2^{conf}$). Therefore, the overall goal is to reduce the difference between the top-2 labels until the model f flips the prediction to a different label (condition $f(m(i))_1^{label} \neq f(m(i))_1^{label}$). In general, Equation (1) takes values in [-1,1]. A zero value indicates that the models assign equal confidence scores to the top-2 labels. A negative value indicates that the model f flips the prediction to a different label, whose confidence level corresponds to the absolute value of Equation (1).

We can expand this to a multi-objective problem where both "fooling" the model and reducing the number of perturbations (at the pixel level) are equally important:

Problem Definition 1. The problem is finding a mutated image $m(i)$ such that $f(m(i)) \neq f(i)$ and that minimizes the distance $d(i, m(i))$, with the constraints that both i and $m(i)$ share the same correct label (same ground truth).

Beyond flipping the prediction outcome by optimizing for O_1, we now also need an additional objective to guide the search towards minimizing the difference between the original image i and its mutated counterpart $m(i)$.

Therefore, our second objective counts the number of pixels that differ between the seed image i and the mutated image $m(i)$:

$$\min\ O_2 = \pi(m(i[a,b]) \neq i[a,b]) \tag{2}$$

$$= |\{i[a,b] \in i : i[a,b] \neq m(i[a,b])\}| \tag{3}$$

where $i[a,b]$ and $m(i[a,b])$ denote the pixel values in row a and column b for the two images i and $m(i)$, respectively.

These two objectives are *conflicting*. A simple solution for O_1 may include changing all pixels in the original figure i so that the object is no longer recognizable for the model f. However, such a solution would not be optimal for O_2. Vice versa, a new image with zero alteration would be optimal for O_2 but not flip the prediction as sought by O_1.

Given our objectives' conflicting nature, it is impossible to find one single solution that optimizes them all. In other words, the problem is inherently multi-objective, and the goal is to find the set of optimal trade-offs between O_1 and O_2.

In our context, a solution (image perturbation) x is said to *dominate* another solution y (denoted with $x <_p y$) if and only if (1) x is better than y w.r.t. one of the objective (e.g., O_1) and it is not worse than y for the other objectives. Among all possible solutions, we are interested in finding a set of solutions x^* that are not dominated by any other solution:

$$O(x^*) <_p O(y)\ \ \forall y \neq x^* \in \Omega(i) \tag{4}$$

where $\Omega(i)$ denotes the set of all possible perturbations that can be applied to the seed image i. All the solutions that are not dominated by any other solutions are said to form a *Pareto optimal set*. The corresponding objective vectors (containing the values of two objectives O_1 and O_2) are said to form a *Pareto front* (or frontier).

3.1. Single and multi-objective differential evolution

To find adversarial attacks, we apply our differential evolution (DE) approach in two ways. Firstly, we apply GA_{DE}, a traditional single-objective variant (to optimize O_1). Secondly, we apply multi-objective variants (NSGA-II$_{DE}$, NSGA-III$_{DE}$, MOEA/D$_{DE}$, AGE-MOEA$_{DE}$) to optimize O_1 and O_2. All our variants are based on their respective traditional *environmental selection* operators, which are detailed below.

All variants iteratively evolve a pool of N randomly generated adversarial attacks, called *population*. In each iteration, N offspring attacks are generated from the population using variation operators. Then, the population for the next iteration is obtained by combining the previous population and the offspring attack, forming a pool Q of $2 \times N$ attacks and selecting the N top individuals. The *environmental selection* of these individuals is specific to each algorithm and represents the main difference between the single- and multi-objective problems.

- In GA_{DE} [30] (single-objective), the *environmental selection* is applied by selecting the best N individuals among the parent and the offspring solutions/attacks according to the main objective O_1. This mechanism is *elitist* since the best attacks can survive across the generations until new better solutions are found.
- In NSGA-II$_{\text{DE}}$, the *environmental selection* is performed by applying the *fast non-dominated sorting algorithm NSGA-II* [31], which ranks the solutions in Q into sub-dominated fronts based on the dominance relation. For solutions within the same front, the selection is further made considering the *crowding distance*, which aims to promote solutions in the less-crowded areas of the front.
- MOEA/D$_{\text{DE}}$ [19] is a decomposition-based evolutionary algorithm [19] that breaks down multi- or many-objective problems into several single-objective sub-problems through sum scalarization. It initializes a set of predetermined search directions (or reference vectors) evenly spread across objective space. These directions are obtained by normalizing the objective scores of the individual in the population and applying systematic approaches (e.g., [32]) to create uniformly distributed directions, each with different weights. During the search, MOEA/D$_{\text{DE}}$ promotes solutions that are closer to those reference directions, optimizing for both optimality and diversity.
- NSGA-III$_{\text{DE}}$ [20] is a many-objective extension to NSGA-II. Like its predecessor, NSGA-III$_{\text{DE}}$ uses the non-dominated sorting algorithm. Next, it utilizes the *reference points* defined in the starting parameters, to find a solution with the smallest perpendicular distance to the reference directions in the normalized objective space. The reference points are generated using the Das and Dennis systematic approach [32], which are uniformly distributed in the objective space, thus promoting diversity in the generated solutions.
- AGE-MOEA$_{\text{DE}}$ [21], the algorithm executes similarly to *NSGA-II*, but with a modified *crowding distance* formula. Non-dominated fronts are sorted using the non-dominated sorting procedure (like in NSGA-II), whereby the first front is utilized to normalize the object scores and estimate Pareto front geometry. This geometry (or curvature p) is estimated using heuristics and used to measure both the diversity and proximity of the generated solutions using the p-norm.

In the following, we detail (1) the encoding schema, (2) how we initialize the initial population, and (3) the variation operator.

3.1.1. Encoding schema

As mentioned before, an adversarial attack is produced by altering a seed image i. Instead of representing/encoding an adversarial attack as a completely new image, we only encode the changes to be applied, also called the *mask*. In particular, given the seed image i, we encode a solution/attack as a list of pixels to change: $X = [x_1, \ldots, x_k]$. Each entry x_j in X is a tuple $[a, b, value_j]$, where a and b determine the position of the pixel to change (i.e., a is the row index and b is the column index), while $value_j$ indicates the new pixel value in RGB notation.

3.1.2. Initialization

We first begin our initialization by generating an initial pool of adversarial attacks. To this aim, we create N attacks by creating an empty mask $X = []$ and adding some changes using the *add* operator, one of three alternative variation operators described below.

3.1.3. Variation operator

Given a parent attack X, we design three types of operators that *add, delete*, or *change* entries in X. Each operator is applied with probability 1/3.

The **add** operator randomly inserts one entry in X with probability $P = 1$; a second entry is added with probability $P = 0.5$; the third one with probability $P = 0.25$; and so on until no other element is added. To add a new element/entry x in X, this operator randomly selects one pixel from the original seed image i with position row_j and col_j and draws three random (noise) values $\delta(\mu, \lambda)$ from a Gaussian distribution with mean μ and standard deviation λ, that will be applied to the respective R, G and B channels. Hence, the new entry x will be equal to $[row_j, col_j, value_j + \delta(\mu, \lambda)]$.

The **delete** operator simply deletes one entry/tuple from the mask X. However, this operator is applied only if X contains at least two entries. This operator plays a critical role in our multi-objective formulation as it allows to remove spurious pixel changes that do not contribute to changing the prediction results of the model f under test.

The **change** operator selects an entry from the mask X and modifies the pixels values using the traditional *differential operator*. Given a parent mask to mutate X and a donor mask Y, a new solution X' is formed by using the following formula:

$$X'[a,b] = \begin{cases} R_1[a,b] + F \cdot \left(R_2[a,b] - Y[a,b]\right) & \text{if } r < CR \\ X[a,b] & otherwise \end{cases} \tag{5}$$

where $r \in [0,1]$ is a randomly generated number; R_1 and R_2 are other solutions within the population. There are various variants of the differential operator that differ in how R_1, R_2, and the donor solution Y are selected. In this paper, we use the standard

Algorithm 1: Post-processing matrix mutation.

```
Input:
$Y$ = original image from ImageNet ILSVRC2012 validation dataset
$X$ = change matrix from RQ1
$Y^*$ = $Y$ with change matrix applied
Output: $X^* \subset X$
1  $X^* \leftarrow X$
2  $max\_iterations = 100,000$
3  $stop\_counter = 30$
4  for max_iterations do
5      $x_j$ = randomly selected item from matrix $X^*$
6      while prediction($Y$) $\neq$ prediction($Y^*$) do
7          original_pixel = $Y[x_j][a,b]$
8          current_pixel = $X^*[x_j]$
9          diff = $Y[x_j][value_j]/100 * 20$                ▷ Take 20% of the original pixels color value
10         $Y^*[x_j][value_j] = Y^*[x_j][value_j] +$ diff
11         if prediction($Y$) == prediction($Y^*$) then
12             $stop\_counter - 1$
13             if $stop\_counter == 0$ then
14                 end                                      ▷ Stop the algorithm
15             $Y^*[x_j]$ = current_pixel                   ▷ Resets pixel to before we started modifying it
16             end while
17         if $Y^*[x_j][value_j] >=$ original_pixel$[value_j]$ then
18             $Y^*[x_j]$ = original_pixel
19             $X^*.remove(x_j)$
20             end while
21         if prediction($Y$) $\neq$ prediction($Y^*$) then
22             $stop\_counter = 30$                         ▷ Reset stop_counter
```

DE/rand/1 variant, where rand indicates that the donor Y is randomly selected, while 1 indicates there is only one donor solution. Finally, the other solutions R_1 and R_2 are always randomly selected from the population.

In Equation (5), $F \in [0,2]$ is called *scaling factor* and has a typical value in the range 0.5 - 1.0. It establishes the amplification of the new solution X' from the original solution X based on the differential values of the donor solution Y. Hence, F balances both exploration and exploitation. Finally, $CR \in [0,1]$ is the *crossover rate* and determines how many pixels in X will be changed.

If the pixel $X[a,b]$ differs from the original seed solution, Equation (5) may remove this change if $R_1[a,b]$, $R_2[a,b]$, and $Y[a,b]$ are identical to the original seed image. To prevent this case, we apply a small tweak in our context compared to the traditional differential operator. We set the pixel $R_1[a,b] = [0,0,0]$ (in RGB notation) if $R_1[a,b]$ is identical to the pixel of the initial image/seed. The same is done for $R_2[a,b]$ and $Y[a,b]$. Notice that this tweak is applied only if $X[a,b]$ differs from the original seed's pixel in row a and column b.

3.2. Post processing

To post-process the generated adversarial attacks, we take a previously generated mask X and modify the contents, whilst maintaining a flipped prediction. Each element (x_j) in X is a tuple $[a, b, value_j]$, where a and b represent the location of the pixel in the original image, and $value_j$ signifies the color applied to that pixel.

Our process involves selecting a pixel from X (x_j) and gradually mutating $value_j$ towards the original pixels values, whilst ensuring that our prediction remains flipped. If, during this color mutation process, $value_j$ reverts to its original value, x_j is removed from X, thus reducing the number of changes we apply to the adversarial example. The color mutation is applied in a loop, increasing 20% at a time. If, at any point during the loop, the prediction becomes un-flipped, we then revert $value_j$ to the previous working value in the loop and seek a new $x_j \in X$ to minimize.

To ensure a suitable time for post-processing to further minimize our adversarial examples, we run a loop with an arbitrarily chosen $100,000$ iterations. The loop will continue until it reaches the total number of iterations. We also apply a stop_counter variable. This has an initial value of 30, which gradually decreases on failed pixel mutations and is reset upon successful pixel mutations. This process is detailed in Algorithm 1.

4. Study design

To answer RQ1, we run each of our five approaches against five different, well-known, deep-learning computer vision models from the Keras python library. We chose VGG-16 and VGG-19 which are both classified as "very deep" convolutional neural networks for large-scale image recognition[5] [33], that got a canonical status due to their strong performance in the ImageNet benchmark

[5] https://keras.io/api/applications/vgg/.

Algorithm 2: Dataset Acquisition.

```
Input:
$\Omega$ = A set of 1000 randomly selected images from ImageNet ILSVRC2012 validation dataset
$f$ = the model under test
Output: $\Omega^* \subset \Omega$
1  $Classes = \emptyset$
   $\Omega^* = \emptyset$
   for $\omega \in \Omega$ do
2      while $|\Omega^*| < 150$ do
3          $i \leftarrow convert(\omega, 224 \times 224)$;
4          if $0.8 \leq f(i)_1^{conf} \leq 0.9$ then
5              if $f(i)_1^{label} \notin Classes$ then
6                  if $f(i)_1^{label} \equiv GroundTruth(i)$ then
7                      $\Omega^* = \Omega^* \cup \{i\}$;
8                      $Classes = Classes \cup \{f(i)_1^{label}\}$;
9  224x224
```

challenges.[6] VGG-16 was one of the best-performing models in the 2014 ILSVRC challenge and achieved 92.7% top-5 test accuracy on the ImageNet dataset. VGG-16 and VGG-19 both consist of 3x3 convolutional layers stacked on top of each other in increasing depth, with VGG-16 having 16 convolutional layers, and VGG-19 being 'deeper' with 19 convolutional layers. Furthermore, ResNet50, ResNet101 and ResNet152 all are based on deep residual learning for image recognition[7] [34]. As input, the models take images of size $224 \times 224 \times 3$ pixels, with L (the collection of possible target labels) encompassing 1000 different object classes.

In RQ1, we ran each image, model, and approach combination a total of 10 times. This resulted in $\approx 37,500$ test runs in total, the majority of which would generally result in the creation of an adversarial example. To now answer RQ2, we sample from this set of test runs, randomly selecting an adversarial example from each of the image, model, and approach combinations. It should be noted that post-processing was not executed against all combinations due to an adversarial not always being found during RQ1. With our adversarials selected, we execute post-processing 10 times against each adversarial image to create an aggregated set of results.

In our experiments, we use the ImageNet pre-trained weights released by the original authors after training on the ILSVRC2012 training set, as released through Keras [35]. Note that despite these details, we consider all models as a black box, given the fact that our approach (and the baseline) does not need access to the model internals.

4.1. Dataset

For our experiments, we sample 150 images from the ImageNet ILSVRC2012 validation dataset [3]. As such, this data contains verified ground truth labels, that were obtained in similar fashion to the labels on which the models under test were trained. Thus, in choosing to keep sampling from the ILSVRC2012 validation dataset, we are certain that this data should not be out-of-distribution for the model, while at the same time can assume the models have not yet seen our testing data during training time.

To ensure unbiased pre-formatting, we perform the same steps to resize images during dataset acquisition as when running our main experiment. By utilizing the Pillow[8] Python library, we resize using bi-linear interpolation. This ensures samples are resized as close to the original as possible.

As for the prediction confidence, if the correct ground-truth label is predicted with confidence between 0.8 and 0.9, we consider the image to be a valid image for our experiments: as the ground-truth label is recognized with high confidence, we can be confident it is visually distinguishable and not ambiguous w.r.t. other classes. Concerning this, an image with too high confidence (e.g., greater than 0.9) may be too obviously one particular image class, and flipping may be difficult as a consequence. By retaining only one image per object class, we also ensure a reasonably diverse set of images.

Algorithm 2 helps to outline this information. We first draw an initial pool of images for the input by selecting 1000 random images from the ImageNet validation data folder. After pre-processing the images to have a 224×224 input size, (line 3 in Algorithm 2) we then choose whether to drop or retain the image, based on the prediction confidence by the VGG-19 model (condition in line 4) and class uniqueness (condition in line 5).

4.2. Implementation and parameter settings

We have implemented the various DE-based approaches in Pymoo v0.6.0 [36], using Python 3.10 and Keras 2.12.2. Pymoo [36] is an open-source framework that allows us to easily adapt the original search algorithms to our DE approach. Runs are evaluated inside a Docker container on an AMD EPYC 7713 64-Core Processor running at 2.6 GHz and with 256 available CPUs. We had 3

[6] https://image-net.org/challenges/LSVRC/index.php.

[7] https://keras.io/api/applications/resnet.

[8] https://pypi.org/project/Pillow/.

Nvidia A40 GPUs each with 48 GB GDDR6 running CUDA version 11.6 available to us. The Dockerfile in our implementation can be rebuilt on any system, easily modifying the CUDA container for a different system.

4.2.1. Parameter settings

We set our approaches, both single- and multi-objective DE, to evolve a population size of 20 over a maximum of 600 generations. In the case of `MOEA/D`DE and `NSGA-III`DE, we were also required to supply reference directions. For `MOEA/D`DE, we used Das and Dennis's systematic approach [32] for generating well-spaced reference points, as done in the original paper [20]. For `MOEA/D`DE, we use uniform sampling on the unit simplex. For both algorithms, we set the number of partitions ('`n_partitions`') equal to 20, i.e., identical to the population size. For the parameter settings, we have chosen the same values as suggested in the literature [37,31]. Further to this, when an adversarial example is found, we allow the test to run for a further 20 generations before killing it and returning the results. This allows the test to continue the search for a better adversarial example, if one may exist.

We use the variation operators as described in Section 3 with a crossover rate $CR = 0.9$ and scaling factor $F = 0.8$, which are the recommended values in the literature [37]. For all algorithms, solutions/attacks are selected for reproduction using the binary-tournament selection [31]. For GADE, the binary selection is based on the single-objective value to optimize (i.e., O_1). Instead, in the multi-objective approaches, the selection relies on dominance to decide which solution wins each tournament round. Finally, we opted for a relatively small population size $p = 20$ (smaller than $p = 100$ used in other studies [16,37]) as suggested in the literature from problems with expensive objective computation [38].

4.3. Evaluation criteria

In our previous work [17], we concluded that the single-objective approach GADE was able to create adversarial examples quicker than our multi-objective `NSGA-II`DE. However, `NSGA-II`DE would create adversarial examples with fewer perturbations. To answer our first research question, we analyze our multi-objective approaches to see how they compare against each other. As mentioned earlier in the paper, we utilized `AGE-MOEA`DE and `NSGA-III`DE as multi-objective problems.

We execute GADE, `NSGA-II`DE, `NSGA-III`DE, `MOEA/D`DE and `AGE-MOEA`DE against each image in our dataset. For every image, we run each algorithm 10 times, to account for their random nature. As a consequence, with 150 images, the end result is a total of 37,500 test runs (7500 for each method). For each of the executions, a new random seed was generated and stored for future replications, together with the results of the generated attacks.

For the main evaluation, we consider two performance metrics: (1) the success rate, indicating the percentage out of the 10 runs for which our methods were capable of causing a change in prediction output, and (2) how many pixels needed modification (i.e., how many tuples are in mask X) in the best solution. For the comparison, we considered the best solution/attack. With GADE, this is the attack with the smallest number of perturbations (changed pixels). Our multi-objective solutions, however, provide a set of Pareto-optimal solutions. For our analysis, among all solutions/attacks that lead to flipping the prediction (i.e., those with negative values for the first objective O_1), we have chosen the one with the lowest number of changed pixels (second objective O_2).

To further visualize the effectiveness of our approaches, we also consider the run-time, along with the number of generations required for generating an adversarial example. As stated in the Parameter Settings 4.2.1, we allow a total of 600 generations for each test run. On top of this, we also record the run-time. This is a simple calculation, taking the time between the start and end of a single test run.

To assess the significance of the differences among our approaches, we use the Friedman test [39], following the guidelines [40–42] for comparing meta-heuristics across a collection of independent benchmark problems (images in our context). We employed the Friedman test —a non-parametric counterpart to ANOVA— because it is suitable for unreplicated block designs and it does not require data to adhere strictly to a normal distribution. For this test, we use the confidence level $\alpha = 0.95$.

While the Friedman test reveals whether some approaches (DE variants) statistically differ, it does not pinpoint the specific combinations (DE variants) for which such a significant difference holds. To gain a deeper understanding of which approach(es) outperform, we utilized the Nemenyi [43] post-hoc test. This test assesses treatment disparities by determining the average rank of each DE variant across multiple problems [43] (images in our case). The significance between two DE variants (e.g., `NSGA-III`DE and `MOEA/D`DE) is established if their respective average ranks differ by at least the provided *critical distance* (CD) [43,41].

To evaluate the effectiveness of our post-processing approach, we observe two performance metrics. (1) the overall color difference applied by the original adversarial example compared to the post-process adversarial and (2) the number of elements in the mask. For the comparison, we obtain the color difference by utilizing the `OpenCV`[9] Python library. The *absdiff* method from `OpenCV` takes two images as its input, to output the overall RGB channel difference between the two images. This gives us two values, *start_diff*, which is the difference between the original image and the adversarial example. As well as *end_diff*, which is the difference between the original image and the post-process adversarial. Finally, we compare the size of the original adversarial examples mask (*start_mask*) to the post-process adversarial mask (*end_mask*).

5. Empirical evaluation

In this section, we discuss the results of our research questions separately.

[9] https://github.com/opencv/opencv-python.

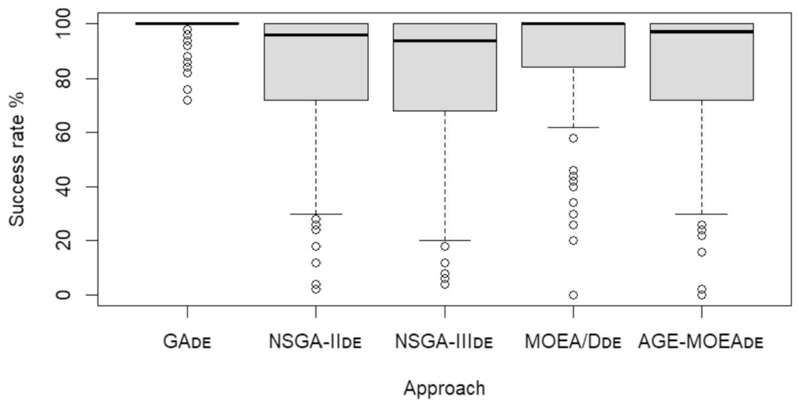

Fig. 1. Success rate of each approach for all 150 images over all 5 models.

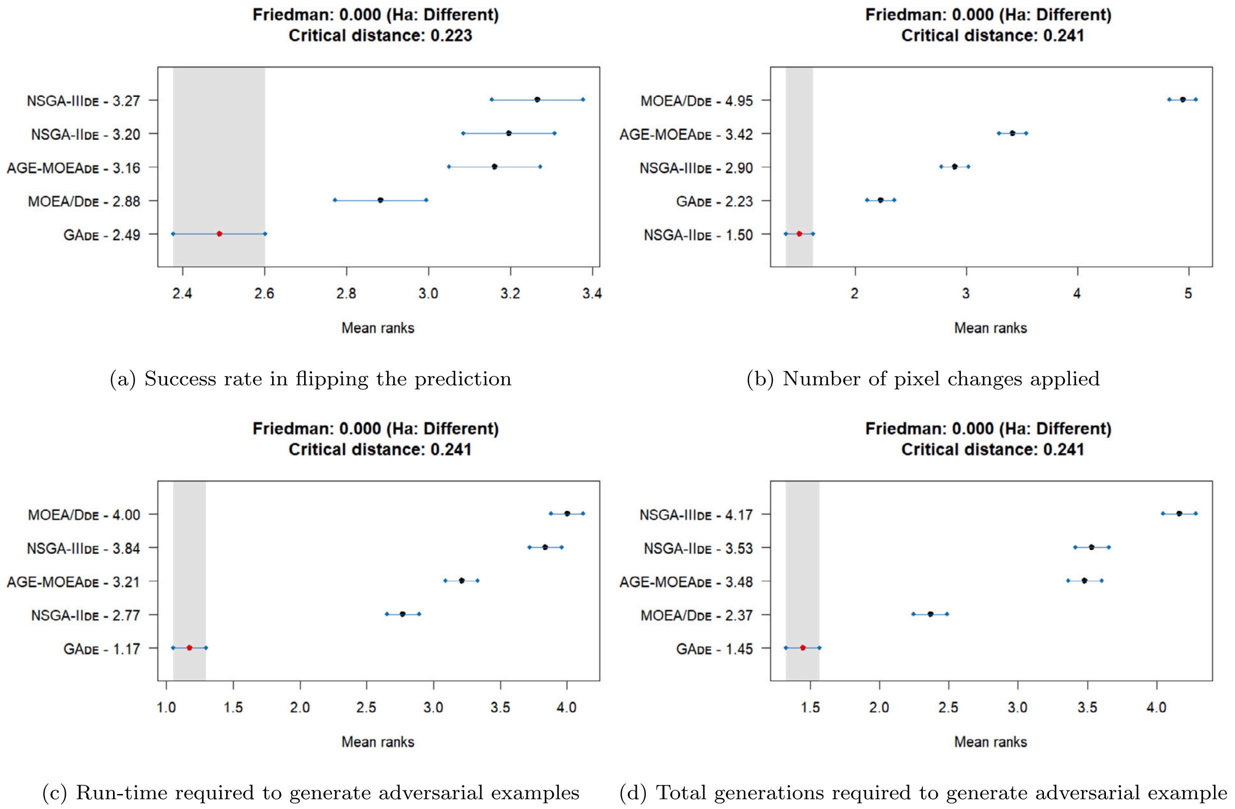

Fig. 2. Results of the Friedman test and of the Nemenyi post-hoc procedure.

5.1. *Results for RQ1*

Fig. 1 compares all five approaches with regard to their ability to flip a model's prediction. The box plot represents the success rate for the total number of executions. Where each image, model, and approach was executed 10 times. As expected from our previous work, we see that GADE is extremely effective at flipping the prediction of an image.

To better understand the results of our experiment, we turn now to Fig. 2, which displays the results of the Friedman test and of the Nemenyi post-hoc procedure. These statistical tests confirm the results shown in Fig. 1, as we see statistically significant superiority of GADE in flipping an image's prediction, i.e., generating a successful adversarial attack. The Friedman test reveals a significant p-value < 0.05, while Nemenyi test suggests that the single-objective DE variant is statistically better than all other meta-heuristics in the comparison.

Interestingly, we observe that MOEA/DDE is a strong contender when flipping a model prediction. This is evident in Fig. 1 wherein the median success rate for MOEA/DDE and GADE are identical (100% success rate), with the former showing only a lower first and second

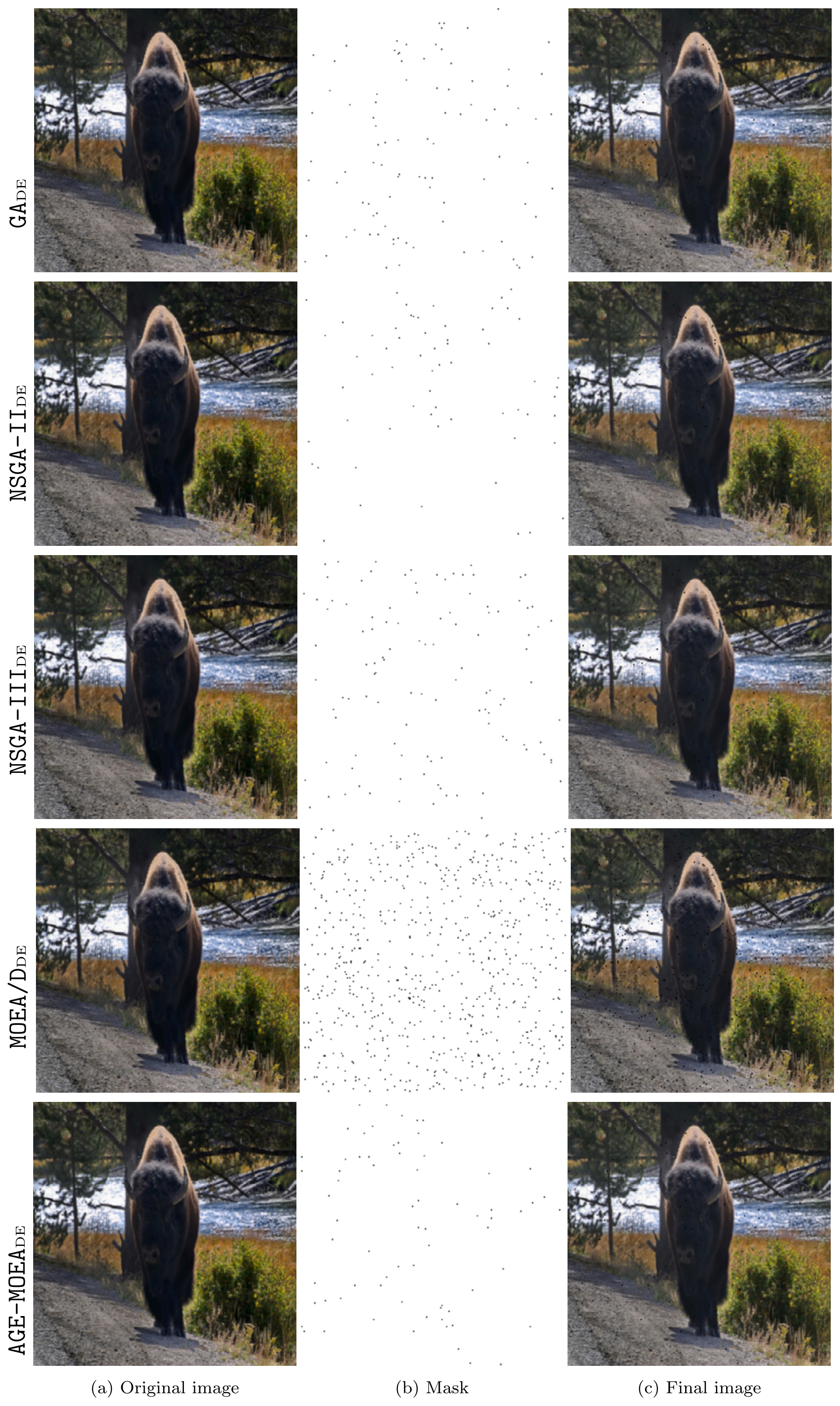

Fig. 3. Adversarial example generation results for Image 'ILSVRC2012_val_00030959.JPEG'.

quartile. However, whilst we observe that MOEA/D$_{DE}$ performs well during prediction flips, we also note that it is less successful in other aspects of our test.

For example, Fig. 2b displays MOEA/D$_{DE}$ as significantly worse than all other approaches w.r.t. the number of pixel changes applied. This is also evident when looking at Fig. 3, in which we visibly see the number of changes applied by the various approaches.

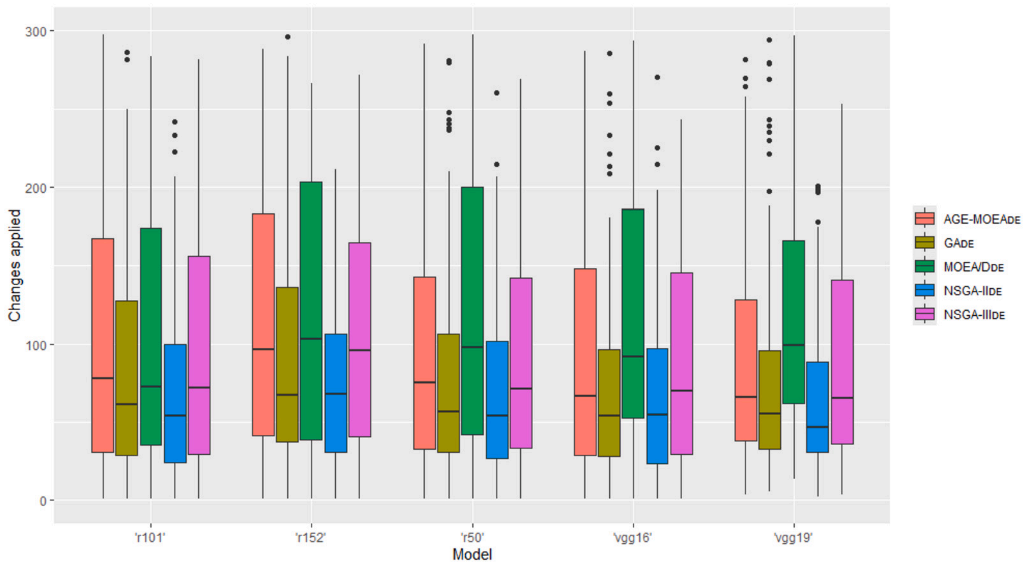

Fig. 4. Changes applied by each approach to each model.

Furthermore, we also observe a statistically better performance of NSGA-II_{DE} w.r.t. its ability to apply minimal (i.e., fewer) pixel changes.

When we consider the run-time along with the number of generations (our maximum being 600), we notice that GADE performs considerably well. It is also worth noticing that MOEA/DDE requires markedly fewer generations compared to the other multi-objective approaches. However, it has a high run-time associated with this. As a result of each iteration of MOEA/DDE being more expensive than for other DE variants due to the computational cost of its internal environmental selection strategy.

In order to comprehend the impact of these outcomes, we will now conduct an in-depth look at the VGG-19 specific results. We have chosen VGG-19, due to it having been the baseline for image selection, as mentioned earlier in Section 4.1.

To better understand the type of attacks generated by our multi-objective approaches, Fig. 3 shows the results of a randomly chosen image in our dataset ('ILSVRC2012_val_00030959.JPEG'). This image was successfully flipped by all five approaches and was originally ground-truthed as 'bison'(165), with a confidence value of 0.8631084. Here, the confidence value represents the model's prediction confidence, where 1.0 is the highest obtainable value (i.e., the model is 100% certain about the prediction).

With Fig. 3, we are able to witness the transition from the original image, to the final image with the mask applied. As previously demonstrated in Fig. 2b, we can observe that NSGA-IIDE generated the smallest mask size, with MOEA/DDE having a considerably larger mask. This is increasingly apparent in the final image for each approach, where the changes applied by MOEA/DDE are more pronounced.

Finally, given that our goal is minimal adversarial example generation, we observe the overall applied changes by each approach. Fig. 4 displays each model tested, along with the changes applied by each approach over the 150 test images. This box plot affirms our previous observations, in which NSGA-IIDE generated the least number of changes in an adversarial example, followed by GADE. MOEA/DDE generated the largest number of changes, but with quite a broad distribution. While AGE-MOEADE and NSGA-IIIDE are our middle ground for applied changes.

5.2. Results for RQ2

When investigating the masks generated by our approaches, it is visible in Fig. 3 that they tend to work towards a dark-colored pixel $RGB(0,0,0)$. In some cases, this can lead to less subtle adversarial examples. Therefore, we applied post-processing to our adversarial examples to observe whether this technique could prove useful in the minimal attack generation process.

Table 1 displays the results of our post-processing against the 'Welsh springer spaniel' image (ILSVRC2012_val_00022868). Here we witness the *start* and *end* specific values of color difference and mask size. These values for our original adversarial example, along with our post-process adversarial, allow us to observe the reduction obtained by running post-processing. Lastly, we note the post-processing time required in reaching this further minimized adversarial example, along with the number of repetitions taken.

What we witness from these results, is that post-processing can drastically reduce the amount of change in our adversarial examples. More importantly, for MOEA/DDE, which, as mentioned previously generated an extremely large number of changes, post-processing yields a considerably smaller number of changes, rivaling that of other approaches.

NSGA-IIDE continues to be the best-performing model w.r.t. minimal perturbation; however, this is likely due to the size of the original mask. A smaller mask results in a smaller search space for the post-processing approach, leading to fewer repetitions and quicker post-processing. It is interesting, that despite the smaller mask size of NSGA-IIDE, we are still able to observe a reduction of nearly half the original changes. For example, by obtaining the overall reduction percentage between *start_diff* and *end_diff*:

Table 1
Original adversarial example (*start*) and post-process adversarial (*end*) execution data.

	ILSVRC2012_val_00022868					
Approach	start diff	end diff	start mask	end mask	run-time	reps
GADE	7438	1086	28	3	34	85
NSGA-IIDE	4719	549	16	2	18	47
NSGA-IIIDE	4875	1268	20	5	19	48
AGE-MOEADE	12884	2692	48	9	44	102
MOEA/DDE	1550187	415	6098	3	4893	7739

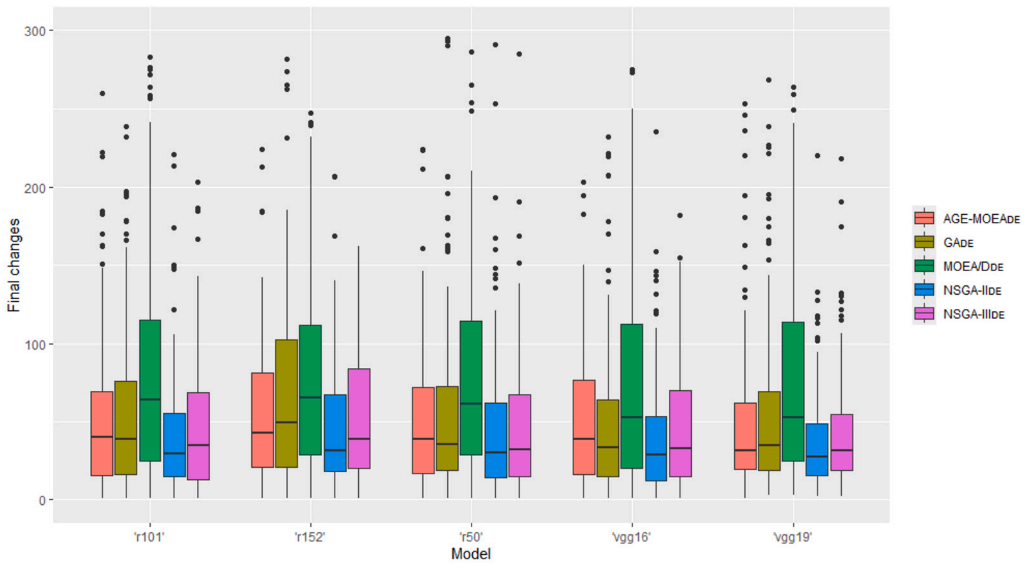

Fig. 5. Changes applied after post-processing.

$$\text{reduction} = \frac{\text{end\_diff} - \text{start\_diff}}{\text{start\_diff}} \times 100 \tag{6}$$

We ascertain a reduction of 42.86% over the color our original adversarial example applied versus the post-process adversarial. By utilizing the same Equation (6), but for *start/end_mask*, we produce a mask reduction of 40%.

Further evidence can be observed through Fig. 5, which displays the median mask size after post-processing was applied. Here, we take the median values for the 10 runs of each image, model, and approach combination, where an original adversarial was successfully generated.

To complement this, we also take note of Fig. 6. Here, we observe the median run-time for the same examples as Fig. 5. As displayed in Table 1, run-time is heavily correlated to the original mask size.

Finally, we observe in Fig. 7 the overall mask reduction, together with the original and final adversarial examples. We display here the changes from the MOEA/DDE approach against the 'ILSVRC2012_val_00022868' (Welsh springer spaniel) in the ResNet101 model.

6. Threats to validity

Several threats to validity can be identified for our study. In the next subsections, we discuss the threats and how we addressed them.

6.1. Construct validity

While we assume our adversarial examples have perceptually visible changes, but these changes are small enough not to change the object of focus (i.e., the ground truth) for an image, we do not formally validate this in human experiments. Thus, it is possible that found adversarial examples may be degraded to the extent that a change in ground truth label would be needed and justifiable. However, in our current experiments, we already see many prediction flips happening when changing less than 200 (so less than 0.3%) out of 50,176 pixels, making it unlikely that the object of focus would be completely obscured by the mutations. Next to this, we validated the model's initial ground-truth label against the data supplied by the ImageNet website. Despite this, the model may have an erroneous initial prediction, and even if it would be correct in comparison to the ground truth, plausible labels beyond the indicated ground truth may exist if the visual scene is complex or semantically ambiguous [44]. Thus, while we frame our technique

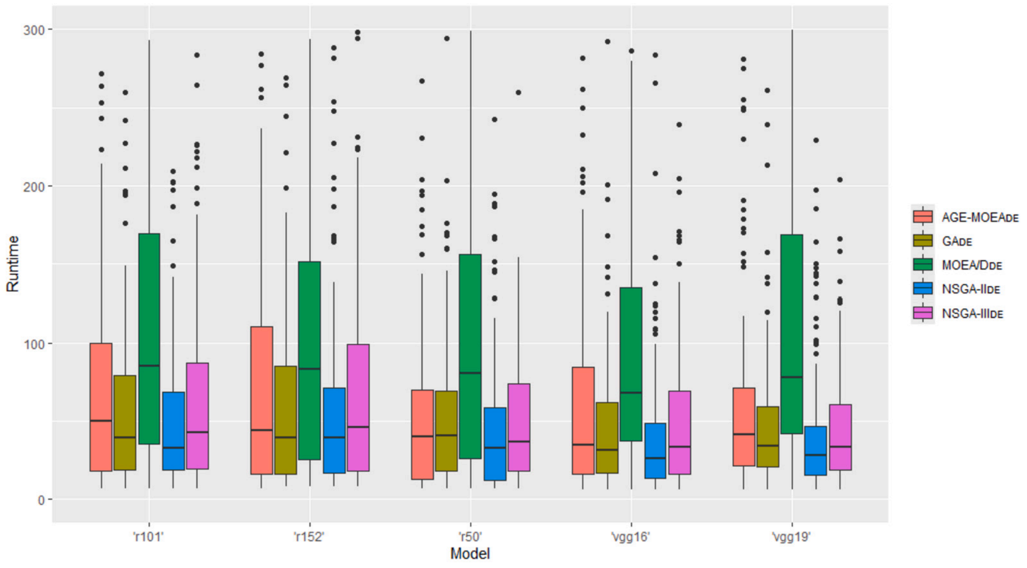

Fig. 6. Median post-processing run-time.

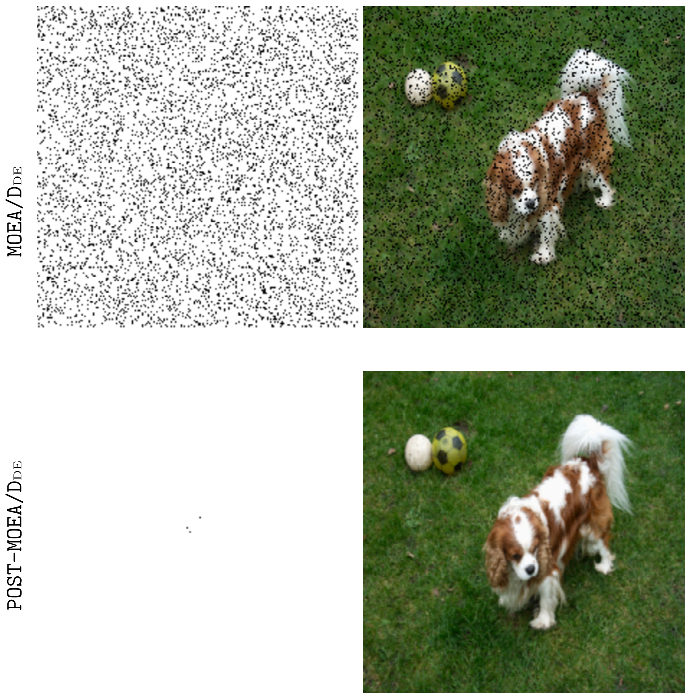

Fig. 7. Post-processing reduction for Image ‘ILSVRC2012_val_00022868’.

as one creating adversarial attacks, it cannot be guaranteed that we necessarily push the model towards being wrong. Still, the aspect about adversarial attacks that keeps holding is that subtle changes lead to different prediction outcomes; as such, a flip in output prediction is still an indicator of the model not giving robust predictions.

6.2. *Internal validity*

While our image selection procedure yielded a random draw of images from 150 unique classes, the ILSVRC2012 classes semantically are not uniformly distributed (e.g. having multiple classes with sub-species of dogs). Future sampling strategies could seek to do more explicitly mitigate this.

Furthermore, as noted in Section 4.1, our image selection remained limited to the ImageNet ILSVRC2012 validation dataset. Although this may seem like a limiting choice, we could assume the models under test had not seen the images during training.

Coupled with this, the images closely aligned to the training data, in both acquisition and ground-truth labeling procedure. As such, they should not be out-of-distribution for the models under test, while it is ambiguous whether this may be the case for other, externally obtained images of the learned object categories.

We do recognize that in practical applications, the intention should very likely be that models should robustly be capable of recognizing a broad variety of images exemplifying the object categories they were trained to recognize. As such, for future work, it will be interesting to possibly include such images obtained from other datasets.

However, this will then also require further explicitization of requirements on a model and subsequent performance assertions. If `VGG-19` e.g. should be more than a prediction program trained to recognize exemplars of 1000 object categories that are similar to what was in ImageNet ILSVRC2012, how much more would it be? Should it be considered the same artefact if retrained with more diverse data? Should it be capable of generalizing to object categories that are similar to the 1000 categories it was trained on? But if so, how would this generalization be defined? As we pointed out in earlier work [45], questions like these tend to be under-specified in machine learning contexts, while more explicit articulation will be essential for developing more robust models together with stronger testing mechanisms.

6.3. External validity

Currently, our approach was only tested against (the state-of-the-art) `BMI-FGSM` [17]. It will be worthwhile to also test it against further attack approaches, such as the one-pixel attack [26]. Furthermore, beyond our current set of DNN models, more canonical models exist that can be studied, such as Inception-v3 [46].

6.4. Conclusion validity

In some runs, our multi-objective approaches fail to find an adversarial example; further optimizations with regard to population and generation size may be required.

7. Conclusion and future work

In this study, we have introduced three new approaches based on pure differential evolution, along with a post-processing technique. We anticipate that this study will contribute to the efficient generation of concise adversarial examples, facilitating their use in the subsequent training of Deep Neural Networks (DNNs).

The approaches, namely `MOEA/D`$_{\text{DE}}$, `AGE-MOEA`$_{\text{DE}}$, and `NSGA-III`$_{\text{DE}}$, each execute customized differential operators for generating adversarial examples. Our post-processing approach utilizes random search in an image change space to optimize for a less noticeable adversarial example.

Our empirical study with multiple DNN models for image recognition showed that (1) a single-objective approach is the most efficient way of generating adversarial examples. (2) `NSGA-II`$_{\text{DE}}$ is the optimal solution for generating adversarial examples with minimal perturbations. (3) By applying a post-processing approach, we can reduce the overall amount of changes and delta variation for our adversarial examples. This process was effective on all approaches, proving surprisingly effective on `MOEA/D`$_{\text{DE}}$ where the initial amount of changes was considerably higher than others.

In our future work, it will be worthwhile to explicitly look at the potential difficulty of images due to semantic ambiguity, which already can show in the initial classification confidence of a model [44]. Furthermore, we intend to extend our comparison to DNN models with different input/output criteria.

Moreover, in this study, we utilized many-objective approaches as multi-objective algorithms. In future work, it would be interesting to utilize our many-objective methods with a third objective. As we have seen in this work, we typically reduce our pixel color *(RGB)* value to *black (0,0,0)*. By optimizing the color delta variation (image noise), we could produce better minimal adversarial examples without the need for post-processing.

CRediT authorship contribution statement

Antony Bartlett: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Cynthia C.S. Liem:** Writing – review & editing, Supervision. **Annibale Panichella:** Writing – review & editing, Supervision.

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.

References

[1] B. Antony, sbft23-journal-code-submission, in: sbft23' special edition (2.0.0). Zenodo, https://doi.org/10.5281/zenodo.10250866.

[2] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain, Content-based image retrieval at the end of the early years, IEEE Trans. Pattern Anal. Mach. Intell. 22 (12) (2000) 1349–1380.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.

[4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: Proceedings of the International Conference on Learning Representations (ICLR), 2014.

[5] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: Proceedings of the International Conference on Learning Representations (ICLR), 2015.

[6] Y. Tian, K. Pei, S. Jana, B. Ray, Deeptest: automated testing of deep-neural-network-driven autonomous cars, in: Proceedings of the 40th International Conference on Software Engineering, ICSE'18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 303–314.

[7] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, Y. Wang, DeepMutation: mutation testing of deep learning systems, in: Proceedings of the 29th International Symposium on Software Reliability Engineering (ISSRE), 2018.

[8] K. Pei, Y. Cao, J. Yang, S. Jana, Deepxplore: automated whitebox testing of deep learning systems, Commun. ACM 62 (11) (2019) 137–145, https://doi.org/10.1145/3361566.

[9] J. Kim, R. Feldt, S. Yoo, Guiding deep learning system testing using surprise adequacy, in: Proceedings of the 41st International Conference on Software Engineering, ICSE'19, IEEE Press, 2019, pp. 1039–1049.

[10] J. Guo, Y. Zhao, H. Song, Y. Jiang, Coverage guided differential adversarial testing of deep learning systems, IEEE Trans. Netw. Sci. Eng. 8 (2) (2021) 933–942, https://doi.org/10.1109/TNSE.2020.2997359.

[11] P. Zhang, B. Ren, H. Dong, Q. Dai, Cagfuzz: coverage-guided adversarial generative fuzzing testing for image-based deep learning systems, IEEE Trans. Softw. Eng. (2021) 1, https://doi.org/10.1109/TSE.2021.3124006.

[12] A. Nguyen, J. Yosinski, J. Clune, 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), 2015.

[13] Y. Zhou, Y. Tan, Q. Zhang, X. Kuang, Y. Han, J. Hu, An evolutionary-based black-box attack to deep neural network classifiers, https://doi.org/10.1007/s11036-019-01499-x, 2021.

[14] T. Suzuki, S. Takeshita, S. Ono, Adversarial example generation using evolutionary multi-objective optimization, in: 2019 IEEE Congress on Evolutionary Computation (CEC), 2019, pp. 2136–2144.

[15] W. Sun, J. Jin, W. Lin, Minimum noticeable difference based adversarial privacy preserving image generation, https://arxiv.org/abs/2206.08638, 2022.

[16] J. Lin, L. Xu, Y. Liu, X. Zhang, Black-box adversarial sample generation based on differential evolution, 2020.

[17] A. Bartlett, C.C.S. Liem, A. Panichella, On the strengths of pure evolutionary algorithms in generating adversarial examples, in: 2023 IEEE/ACM International Workshop on Search-Based and Fuzz Testing (SBFT), 2023, pp. 1–8.

[18] T. Salimans, J. Ho, X. Chen, S. Sidor, I. Sutskever, Evolution strategies as a scalable alternative to reinforcement learning, arXiv preprint, arXiv:1703.03864, 2017.

[19] Q. Zhang, H. Li, Moea/d: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.

[20] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints, IEEE Trans. Evol. Comput. 18 (4) (2014) 577–601, https://doi.org/10.1109/TEVC.2013.2281535.

[21] A. Panichella, An adaptive evolutionary algorithm based on non-Euclidean geometry for many-objective optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 595–603.

[22] S. Ghamizi, M. Cordy, M. Gubri, M. Papadakis, A. Boystov, Y. Le Traon, A. Goujon, Search-Based Adversarial Testing and Improvement of Constrained Credit Scoring Systems, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1089–1100.

[23] S. Khare, R. Aralikatte, S. Man, Adversarial black-box attacks on automatic speech recognition systems using multi-objective evolutionary optimization, in: Proceedings of INTERSPEECH, 2019.

[24] S. Ishida, S. Ono, Adjust-free adversarial example generation in speech recognition using evolutionary multi-objective optimization under black-box condition, Artif. Life Robot. 26 (2) (2021) 243–249, https://doi.org/10.1007/s10015-020-00671-x.

[25] K. Chan, B.H. Cheng, Evoattack: an evolutionary search-based adversarial attack for object detection models, in: International Symposium on Search-Based Software Engineering, Springer, 2022, pp. 83–97.

[26] J. Su, D.V. Vargas, K. Sakurai, One pixel attack for fooling deep neural networks, IEEE Trans. Evol. Comput. 23 (5) (2019) 828–841, https://doi.org/10.1109/tevc.2019.2890858.

[27] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech. rep., 2009.

[28] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359, https://doi.org/10.1023/A:1008202821328.

[29] S.-M. Moosavi-Dezfooli, A. Fawzi, P. Frossard, Deepfool: a simple and accurate method to fool deep neural networks, https://doi.org/10.48550/ARXIV.1511.04599, 2015.

[30] J.H. Holland, Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, 1992.

[31] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: nsga-ii, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[32] I. Das, J.E. Dennis, Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems, SIAM J. Optim. 8 (3) (1998), https://doi.org/10.1137/S1052623496307510 631–657.

[33] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015.

[34] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, https://doi.org/10.48550/ARXIV.1512.03385, 2015.

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

[36] J. Blank, K. Deb, Pymoo: multi-objective optimization in python, IEEE Access 8 (2020) 89497–89509.

[37] J. Montgomery, S. Chen, An analysis of the operation of differential evolution at high and low crossover rates, in: IEEE Congress on Evolutionary Computation, IEEE, 2010, pp. 1–8.

[38] T. Chugh, K. Sindhya, J. Hakanen, K. Miettinen, A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms, Soft Comput. 23 (2019), https://doi.org/10.1007/s00500-017-2965-0.

[39] W.J. Conover, Practical Nonparametric Statistics, vol. 350, John Wiley & Sons, 1999.

[40] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization, J. Heuristics 15 (2009) 617–644.

[41] A. Panichella, A systematic comparison of search-based approaches for lda hyperparameter tuning, Inf. Softw. Technol. 130 (2021) 106411.

[42] X. Devroey, A. Gambi, J.P. Galeotti, R. Just, F. Kifetew, A. Panichella, S. Panichella, Juge: an infrastructure for benchmarking Java unit test generators, Softw. Test. Verif. Reliab. 33 (3) (2023) e1838.

[43] P. Nemenyi, Distribution-free multiple comparisons, PhD Thesis, Princeton University, 1963.

[44] C.C.S. Liem, A. Panichella, Oracle issues in machine learning and where to find them, in: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, 2020.

[45] A. Panichella, C.C.S. Liem, What are we really testing in mutation testing for machine learning? A critical reflection, in: Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering - NIER Track, 2021.

[46] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.