On the Strengths of Pure Evolutionary Algorithms in Generating Adversarial Examples

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

DOI
10.1109/SBFT59156.2023.00012

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 IEEE/ACM International Workshop on Search-Based and Fuzz Testing (SBFT)

Citation (APA)

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.
Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.
On the Strengths of Pure Evolutionary Algorithms in Generating Adversarial Examples

Antony Bartlett  
Delft University of Technology  
Delft, the Netherlands  
a.j.bartlett@tudelft.nl

Cynthia C. S. Liem  
Delft University of Technology  
Delft, the Netherlands  
c.c.s.liem@tudelft.nl

Annibale Panichella  
Delft University of Technology  
Delft, the Netherlands  
a.panichella@tudelft.nl

Abstract—Deep learning (DL) models are known to be highly accurate, yet vulnerable to adversarial examples. While earlier research focused on generating adversarial examples using white-box strategies, later research focused on black-box strategies, as models often are not accessible to external attackers. Prior studies showed that black-box approaches based on approximate gradient descent algorithms combined with meta-heuristic search (i.e., the BMI-FGSM algorithm) outperform previously proposed white- and black-box strategies. In this paper, we propose a novel black-box approach purely based on differential evolution (DE), i.e., without using any gradient approximation method. In particular, we propose two variants of a customized DE with customized variation operators: (1) a single-objective (Pixel-SOO) variant generating attacks that fool DL models, and (2) a multi-objective variant (Pixel-MOO) that also minimizes the number of changes in generated attacks. Our preliminary study on five canonical image classification models shows that Pixel-SOO and Pixel-MOO are more effective than the state-of-the-art BMI-FGSM in generating adversarial attacks. Furthermore, Pixel-SOO is faster than Pixel-MOO, while the latter produces subtler attacks than its single-objective variant.

Index Terms—differential evolution, adversarial examples, deep learning, search-based software engineering

I. INTRODUCTION

Many systems we encounter in daily life include machine learning components that make automated decisions or inferences based on observed data patterns. Ever since so-called deep convolutional neural networks outperformed handcrafted methods in the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [26], 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 [11], [34].

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 DL model predict incorrect output. Existing approaches to adversarial example generation can be classified into white-box and black-box methods. White-box approaches [12], [16], [20], [25], [35], [37] require access to the model under test (i.e. the model architecture, neuron weight values, and gradients). Black-box strategies [19], [23], [31], [32], [39], instead, only require access to model inputs and outputs. These approaches are considered more realistic as it reflects what external attackers can obtain [19], e.g., in the case of remote API access.

Therefore, in this paper, we focus on black-box strategies and target deep neural network (DNN) models for image recognition. In this field, the state-of-the-art approach by Lin et al. [19] uses 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.

Despite 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 this paper, 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 introduce three customized operators for DE that introduce perturbations (pixel changes) in the target images. We focus on DE as meta-heuristics since they can outperform gradient-based approaches (e.g., reinforcement learning) as reported in recent studies [27].

We introduce two variants of DE: (1) a single-objective variant (Pixel-SOO) that steers for pixel-based input changes to cause an output prediction to change gradually; (2) a multi-objective variant (Pixel-MOO) that additionally seeks to minimize the number of pixel modifications made to the original input image. Then, we conduct a preliminary study focused on answering the following research questions:

**RQ1**: How do Pixel-SOO and Pixel-MOO perform compared to the state-of-the-art BMI-FGSM in generating adversarial attacks?
RQ2: What are the strengths and weaknesses of Pixel-SOO and Pixel-MOO?

The results of our study with five DNN models show that the proposed DE-based approaches are more effective in generating adversarial attacks than BMI-FGSM despite not using any gradient-based methods. In particular, BMI-FGSM struggles to generate adversarial examples for all DNNs under test; besides, when successful, it required more generations than Pixel-MOO and Pixel-SOO. Instead, our approaches are always able to generate attacks within the same search budget used for BMI-FGSM. Finally, we discover that Pixel-MOO and Pixel-SOO provides two different trade-offs. The former generates attacks with fewer and more subtle changes than the latter, but requires more generations to converge.

II. RELATED WORK

As discussed in our Introduction, white-box testing [12], [16], [20], [25], [35], [37] 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 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 [10] and speech audio [14], [15].

In literature, various black-box attacks on image recognition DNNs have been proposed [19], [23], [39]. Nguyen et al. [23] 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. [39] propose 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 [4] 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. [32], 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. [31], 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. [30] propose a single-pixel adversarial attack using DE, executed against the classical CIFAR-10 [17] 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 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. [19], who combined DE [29] and the Fast Gradient Sign Method [22] 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. [19] 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. [19] showed that black-box approaches based on 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).

In this paper, we introduce a simpler approach purely based on DE. Furthermore, we introduce both a single- and multi-objective variant of our approach. The former focuses on flipping the model prediction, while the latter considers an additional objective that aims to directly minimize the number of modified pixels. As our results will show, our approach requires a much lower number of model queries and introduces fewer image changes compared to BMI-FGSM.

III. APPROACH

Without loss of generality, an image classifier $f$ is a mathematical function/model $f : I \rightarrow L \times \mathbb{R}^n$, which takes as input an image $i \in I$ and returns a label $l \in L$ and a confidence vector $\text{conf} \in \mathbb{R}^n$, which contains the probabilities associated with all labels $l \in L$ in descending order. The first element $\text{conf}_1$ is the probability associated with

1https://www.tensorflow.org/datasets/catalog/cifar10
the most likely (predicted) label $l$, while the remaining entries in $\text{conf}_2, \ldots, \text{conf}_n$ are related to the other possible labels $\in L$. We can now reformulate adversarial attack generation as a search problem:

Definition 1: Let $f : I \rightarrow L \times \mathbb{R}$ be a trained model that takes as input an image $i \in I$ and returns a predicted label $l \in L$. Let $m : I \rightarrow I$ 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).

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 $f : I \rightarrow L \times \mathbb{R}$, a seed image $i$, and its mutated variant $m(i)$, we optimize the following objective:

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

In other words, this objective aims to reduce the confidence for the most likely prediction/label ($f(m(i))_1^{\text{conf}}$), while increasing the confidence for the second-most-likely prediction ($f(m(i))_2^{\text{conf}}$). Therefore, the overall goal is to reduce the difference between the top-2 labels until the model flips the prediction to a different label (condition $f(m(i))_1^{\text{label}} \neq f(m(i))_2^{\text{label}}$). 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:

Definition 2: 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 $m(i)$. To this end, our second objective counts the number of pixels that differ between the seed image $i$ and the mutated image $m(i)$:

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

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 consist of changing all pixels in the original figure $i$ such 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 the conflicting nature of our objectives, it is not possible to find one single solution that simultaneously optimizes them all. In other words, the problem is inherently multi-objective where the goal becomes to find the set of optimal trade-offs between $O_1$ and $O_2$. In particular, we aim to find Pareto efficient trade-offs based on the concepts of dominance and Pareto optimality.

A. Single and Multi-objective Differential Evolution

To find adversarial attacks, we rely on differential evolution (DE) only. Hence, compared to BMI-FGSM, we do not use any algorithm to approximate the gradient. We consider two different variants of DE: Pixel-SOO, a traditional single-objective variant (to optimize $O_1$) and Pixel-MOO, a multi-objective variant based on the non-dominated sorting algorithm (NSDE) [2] (to optimize $O_1$ and $O_2$).

Both 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 selection is performed using an environmental selection and represents the main difference between Pixel-SOO and Pixel-MOO. In Pixel-SOO, 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 Pixel-MOO, the environmental selection is performed using the fast non-dominated sorting algorithm [8], which ranks the solutions in $Q$ into sub-dominated fronts based on the dominance relation.

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

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 in NSDE 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.
2) Initialization: The first step to initializing NSDE involves generating an initial pool of adversarial attacks. To this aim, we create \( N \) attacks by creating an empty mask \( X = \cdot \) and adding some changes using the add operator, one of three alternative variation operators described below.

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 \( \sigma = 1 \); a second entry is added with probability \( \sigma = 0.5 \); the third one with probability \( \sigma = 0.25 \); 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 \( \text{row}_j \) and \( \text{col}_j \) and draws three random (noise) values \( \delta(\mu, \lambda) \) from a Gaussian distribution with mean \( \mu \) and standard deviation \( \lambda \), that will be applied to the respective R, G, and B channels. Hence, the new entry \( x \) will be equal to \( [\text{row}_j, \text{col}_j, \text{value}_j + \delta(\mu, \lambda)] \).

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 changes the pixel values using the traditional differential operator. Given a parent image to mutate \( X \) and a donor solution \( Y \), a new solution \( X' \) is formed by using the following formula:

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

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 \( X_1, X_2 \), and the donor solution \( Y \) are selected. In this paper, we use the standard 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 4, \( F \in [0, 2] \) is called scaling factor and establishes how far the new solution \( X' \) is from the original solution \( X \) based on the differentials 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.

We apply a small tweak in our context compared to the traditional differential operator. If the pixel \( X[a, b] \) differs from the original seed solution, Equation 4 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 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 \( b \).

B. Additional Remarks

BMI-FGSM by Lin et al. [19] and our approach share the main goal of flipping the label prediction. However, there are critical differences that are worth highlighting. The first difference regards the main objective or fitness function. The main (single) fitness function used by BMI-FGSM aims to "suppress the probability of the ground-truth label" [19] until another false label is predicted. Our \( O_1 \) explicitly maximizes the confidence level for the false label, even after the prediction has been flipped (second case in Equation 1).

The most critical difference is within the variation operator. BMI-FGSM combines differential operators and the iterative fast gradient sign method. Instead, Pixel-MOO and Pixel-DOO rely on the differential operators but introduce two novel ways to generate offspring: (1) the add and (2) delete operators introduced in Section III-A. The latter allows deleting pixel changes that do not contribute to optimizing \( O_1 \) (for Pixel-SOO) or that are not Pareto efficient w.r.t. \( O_1 \) and \( O_2 \) (for Pixel-MOO). Finally, Pixel-MOO explores multi-objective optimization where the mask size is considered as an explicit objective with the goal of generating minimal adversarial attacks. Instead, BMI-FGSM targets only the prediction outcome and does not constrain the number of pixels that can be altered to reach a prediction flip.

IV. EMPIRICAL EVALUATION

To answer our RQs, we run our approach with 5 different, well-known deep computer vision models from the Keras library. We chose VGG16 and VGG19 which are both classified as "very deep" convolutional neural networks for large-scale image recognition [28], that got a canonical status due to their strong performance in the ImageNet benchmark challenges [3]. VGG16 was one of the best-performing models in the 2014 ILSVRC challenge and achieves 92.7% top-5 test accuracy on the ImageNet dataset. VGG16 and VGG19 both consist of 3x3 convolutional layers stacked on top of each other increasing the depth, with VGG16 having 16 convolutional layers, and VGG19 being ‘deeper’ with 19 convolutional layers. Furthermore, ResNet50, ResNet101 and ResNet152 all are based on deep residual learning for image recognition [13]. 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 [5]. 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.

A. Dataset

For our experiments, we sample 50 images from the ImageNet ILSVRC2012 Validation data set [26]. We chose the validation set, due to a lack of ground-truth data availability for the test set. First, we draw an initial pool of images for

2https://keras.io/api/applications/vgg/ 
3https://image-net.org/challenges/LSVRC/index.php  
4https://keras.io/api/applications/resnet
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, we then choose whether to drop or retain the image based on the prediction confidence by the VGG19 model and class uniqueness. 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; at the same time, for too high confidence, it may be too obviously one particular image class, and flipping may as a consequence be hard. By retaining only one image per object class, we also ensure a reasonably diverse set of images.

B. Implementation and Parameter settings

We have implemented the different DE-based approaches in PyMoo v0.5.0 [3], using Python 3.10 and Keras 2.2.2. PyMoo [3] is an open-source framework that allows us to easily adapt the simple DE and the NSDE for generating adversarial attacks. 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 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. Our implementation is available on Zenodo: https://doi.org/10.5281/zenodo.7741267

Due to issues with running the code from BMI-FGSM, we were not able to execute it using CUDA [11] (GPU). This meant we could only run the baseline on CPU, which has likely resulted in a slower execution time compared to our multi-objective results which were executed on GPU.

Parameters setting. We set both the multi-, single-objective DE, and BMI-FGSM to evolve a population size of 20 over a maximum of 400 generations. When a test image has been wrongly predicted, we kill the test and allow it to run for five more generations, so better fronts may still be found. For the parameter settings, we have chosen the same values as suggested in the literature [8], [21].

We use the variation operators as described in Section III with a crossover rate \( CR = 0.9 \) and scaling factor \( F = 0.8 \), which are the recommended values in the literature [21]. For both algorithms, solutions/attacks are selected for reproduction using the binary-tournament selection [8]. For Pixel-SSO, the binary selection is based on the single-objective value to optimize (i.e., \( O_1 \)). Instead, in Pixel-MOO, 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 [19], [21]) as suggested in the literature from problems with expensive objective computation [6].

C. Study Design

To answer our first research question, we compare our multi-objective approach Pixel-MOO to our single-objective approach Pixel-SSO (only optimizing for \( O_1 \)); next to this, we compare our single-objective approach to the BMI-FGSM method by Lin et al. [19]. In this, we try to stay as close to the implementation by the authors as possible; as a consequence, beyond the parameters setting population size and generations, we do not modify the authors’ codebase. The BMI-FGSM codebase only supports VGG16, ResNet50 and ResNet101; here, we started our experiments with VGG16. We execute BMI-FGSM, Pixel-SSO and Pixel-MOO against each image in our dataset to generate adversarial examples. For each image, we run each algorithm 10 times, to account for their random nature. As a consequence, with 50 images, the end result is a total of 1500 test runs (500 for each method). For each of the Pixel-SSO and Pixel-MOO executions, a new random seed was generated and stored for future replications, together with the results of the generated attacks. For BMI-FGSM a slight modification was made to the codebase to output the current prediction data. This data is stored for each run, along with the adversarial image.

For evaluation, we consider two performance metrics: (1) the success rate, indicating the percentage out of the 10 runs for which Pixel-SSO and Pixel-MOO 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 in the final population (last generation) of Pixel-SSO. Instead, Pixel-MOO provides a set of non-dominated solutions (front) rather than one single solution. 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 compare BMI-FGSM and Pixel-MOO, we analyze the success rate in flipping the prediction and the median number of changes required to flip the model’s prediction over the 10 runs. We also apply statistical analysis to further assess whether the observed differences are significant or not. We use Fisher’s exact test [9] for the success rate, considering the results of each run (for each algorithm) as a binary/dichotomy outcome (i.e., the prediction was flipped or not). To assess the significance of the differences among Pixel-SSO and Pixel-MOO w.r.t. the number of altered pixels, we use the Wilcoxon rank sum test [7]. For both statistical tests, we use a confidence level \( \alpha = 0.95 \). Furthermore, we complement the test for significance with the Vargha-Delaney statistic (\( A_{12} \)) to measure the effect size of the results [36].

D. Results

Results on VGG16. A full breakdown of the VGG16 results can be seen in Figure 1, which depicts the success rate of BMI-FGSM, Pixel-SSO and Pixel-MOO in creating adversarial examples for the 50 images in our experiment. As we can observe, BMI-FGSM was rarely able to flip the predictions for VGG16 (median, second, and third quartiles
To better understand the type of attacks generated by Pixel-MOO and the baselines BMI-FGSM, Fig. 2 shows the results of the first image in our dataset ‘ILSVRC2012_val_00000323.JPEG’. This is one of the few images for which BMI-FGSM could successfully generate an adversarial attack. The original image was ground-truthed and classified as a ‘grey fox’ (67). When initially running the image through BMI-FGSM, the classification was shown as ‘garbage truck, dustcart’ (280). This is an incorrect classification and may be due to the pre-processing BMI-FGSM applies. Instead, both Pixel-SOO and Pixel-MOO were able to generate adversarial attacks for all the images in at least 1 of the ten individual runs. Indeed, there is no “−” entry in Table I for these two algorithms. W.r.t. the number of introduced changes, we can observe that both algorithms required to change fewer pixels than the state-of-the-art BMI-FGSM. Pixel-MOO changed on average 67 pixels while Pixel-MOO generated successful attacks by changing even fewer pixels (54 pixels on average).

To better understand the type of attacks generated by Pixel-MOO and the baselines BMI-FGSM, Fig. 2 shows the results of the first image in our dataset ‘ILSVRC2012_val_00000323.JPEG’. This is one of the few images for which BMI-FGSM could successfully generate an adversarial attack. The original image was ground-truthed and classified as a ‘grey fox’ (67). When initially running the image through BMI-FGSM, the classification was shown as ‘garbage truck, dustcart’ (280). This is an incorrect classification and may be due to the pre-processing BMI-FGSM applies. Instead, both Pixel-SOO and Pixel-MOO were able to generate adversarial attacks for all the images in at least 1 of the ten individual runs. Indeed, there is no “−” entry in Table I for these two algorithms. W.r.t. the number of introduced changes, we can observe that both algorithms required to change fewer pixels than the state-of-the-art BMI-FGSM. Pixel-MOO changed on average 67 pixels while Pixel-MOO generated successful attacks by changing even fewer pixels (54 pixels on average).

To better understand the type of attacks generated by Pixel-MOO and the baselines BMI-FGSM, Fig. 2 shows the results of the first image in our dataset ‘ILSVRC2012_val_00000323.JPEG’. This is one of the few images for which BMI-FGSM could successfully generate an adversarial attack. The original image was ground-truthed and classified as a ‘grey fox’ (67). When initially running the image through BMI-FGSM, the classification was shown as ‘garbage truck, dustcart’ (280). This is an incorrect classification and may be due to the pre-processing BMI-FGSM applies. Instead, both Pixel-SOO and Pixel-MOO were able to generate adversarial attacks for all the images in at least 1 of the ten individual runs. Indeed, there is no “−” entry in Table I for these two algorithms. W.r.t. the number of introduced changes, we can observe that both algorithms required to change fewer pixels than the state-of-the-art BMI-FGSM. Pixel-MOO changed on average 67 pixels while Pixel-MOO generated successful attacks by changing even fewer pixels (54 pixels on average).
flip to 'Model T' (272), while Pixel-SOO successfully flipped it from the correct class to a 'coyote' (58). Fig. 2 depicts (1) the original image, (2) the 'perturbation mask', and (3) the resulting adversarial attacks.

We can observe that, for BMI-FGSM, the median amount of changed pixels is 4325 and for Pixel-SOO is 25, which is a remarkable difference. From Fig. 2, we can barely see any modification in the change matrix, and this is because BMI-FGSM subtly changes many pixels. Our method, however, stands out for the smallest amount of perturbations which involves making larger color differences to the pixel. However, as we can see by the matrix and the final image, the changes applied are still very subtle.

On top of having considerably fewer perturbations and a better success rate, we observe from Table I that the number of generations required by Pixel-SOO and Pixel-MOO is also much lower than the generations needed by BMI-FGSM for converging. BMI-FGSM took on average 240 generations to reach a prediction flip, while our Pixel-SOO method took only 10, once again being considerably quicker.

**Results on other models.** The original BMI-FGSM implementation (which we have reused) was not compatible with the other models except for the ResNet models. However, in our experiment, BMI-FGSM took a considerable amount of time to complete just a single run (often over 7 hours) and often could not generate any adversarial example within the search budget. In the following, we therefore only report the results for the two DE-based approaches proposed in this paper, namely Pixel-SOO and Pixel-MOO.

Figure 3 depicts boxplots for the success rate over the different runs of Pixel-MOO and Pixel-SOO for the five DNN models in our study. As we can observe, Pixel-SOO is successful almost 100% of the time, with some outliers only for ResNet101. Pixel-MOO still achieves a 100% success rate for more than 50% of the images.

These differences are also confirmed by Fisher’s exact test, of which the results are reported in Table II. More specifically, Pixel-MOO and Pixel-SOO are statistically equivalent in terms of success rate for the large majority of the images (success rate over ten runs) and across all models. For around 20% of images, the single-objective variants statistically outperform the multi-objective variant. For ResNet101 in particular, Pixel-SOO frequently has a higher success rate compared to its multi-objective counterpart. As Pixel-MOO searches for trade-offs between prediction flip (O1) whilst preserving as many original pixel values as possible (O2), we hypothesize it may be slower to reach optima in comparison to Pixel-SOO, and may perform better when a larger search budget (i.e., more generations) would be used.

Finally, we compare the attacks generated by Pixel-MOO and Pixel-SOO w.r.t. the number of pixel perturbations injected in the seed images. Table III reports the results of the Wilcoxon rank sum test and the $A_{12}$ statistics. The results indicate that the adversarial attacks generated by Pixel-MOO contain fewer pixel alterations than those produced by Pixel-SOO, and the results are statistically significant in more than 60% of the comparison, with an effect size being large in the large majority of the significant cases.

Therefore, we can draw some general conclusion: Pixel-MOO and Pixel-SOO provide two different trade-offs concerning the speed and magnitude of the image perturbation. Pixel-SOO is faster by producing attacks with less subtle changes. Instead, Pixel-MOO produced more subtle attacks but requires more time to converge.

**V. THREATS TO VALIDITY**

**Internal validity.** While our image selection procedure yielded a random draw of images from 50 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 more explicitly mitigate for this.

**External validity.** Currently, our approach was only tested against (the state-of-the-art) BMI-FGSM. It will be worthwhile to also test it against further attack approaches, such as the one-pixel attack [30]. Furthermore, beyond our
current set of DNN models, more canonical models exist that can be studied, such as Inception-v3 [33].

**Conclusion validity.** In some runs, Pixel-MOO fails to find an adversarial example; further optimizations w.r.t. population and generation size may be needed.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have introduced two approaches based on pure differential evolution, namely Pixel-MOO and Pixel-SOO, with customized differential operators for generating adversarial example. Our empirical study with multiple DNN models for image recognition showed that: (1) both approaches outperform the state-of-the-art BNN-FGSM in terms of success rate and by applying fewer (yet subtle) pixel changes, and (2) Pixel-SOO and Pixel-MOO provide two different trade-offs in terms of convergence speed (Pixel-SOO is faster) and subtler changes (Pixel-MOO applies fewer pixel changes).

In our future work, it will be worthwhile to more explicitly look at the potential difficulty of images due to semantic ambiguity, which already can show in the initial classification confidence of a model [18]. Furthermore, we intend to extend our comparison to multiple deep learning models, and different search algorithms (e.g., MOEA/D [38] and AGE-MOEA [24]) or combine the strengths of Pixel-MOO and Pixel-SOO with hybrid approaches.

**REFERENCES**

[14] Ishida, S., Ono, S.: Adjust-free adversarial example generation in speech recognition using evolutionary multi-objective optimization under black-box condition (2021)