International Journal of Innovative
Computing, Information and Control
Volume **16**, Number **1**, February **2020**

ICIC International ©2020 ISSN 1349-4198
pp. 405–414

GENERATION OF ADVERSARIAL EXAMPLES USING ADAPTIVE DIFFERENTIAL EVOLUTION

Jun-ichi Kushida, Akira Hara and Tetsuyuki Takahama

Graduate School of Information Sciences
Hiroshima City University
3-4-1 Ozukahigashi, Asaminami-ku, Hiroshima-shi 731-3194, Japan
{ kushida; ahara; takahama }@hiroshima-cu.ac.jp

Received May 2019; revised September 2019

Abstract. *Deep neural networks (DNNs) were shown to be vulnerable to adversarial examples which contain small perturbations. Attacks based on adversarial examples can be classified as either white-box or black-box attacks. In the white box attacks the adversary has complete knowledge about the model being attacked. Black-box attacks are perfromed without any internal model information. As one of the black-box attacks on computer vision, a method of generating adversarial examples using differential evolution (DE) has been reported. DE is as a stochastic direct search method using population, and enables to generate adversarial examples without having access to any information about the network parameter values or their gradients. In this paper, we generate adversarial examples using JADE, which is a variant of adaptive DE. JADE employs a control parameter adaptation mechanism and exhibits high accuracy and rapid convergence in various optimization problems. The effectiveness of the generation of adversarial examples using JADE is examined and discussed by experiments of adversarial attacks against state-of-the-art DNNs.*

Keywords: Adversarial examples, Differential evolution, JADE

1. **Introduction.** Recently, deep neural networks (DNNs) have shown outstanding performance on a wide range of domains like computer vision and natural language processing [1, 2, 3]. However, several studies have demonstrated that in image classification domain, deep neural classification models are easily fooled by adversarial examples [4, 5, 6]. Adversarial examples are inputs to a predictive machine learning model that are designed to cause poor performance [7]. In computer vision, adversarial examples are generated by adding a small amount of calculated noise to input images. Such modification can cause the classifier to label the modified image as a completely different class.

The adversarial attacks can be broadly classified into either white-box or black-box attacks. A white-box attack assumes the attacker has full knowledge and access to the machine learning model, including its parameter values, architecture, training method. However, such assumptions are clearly unrealistic. Instead, a black-box attack assumes the attacker only has access to the inputs and outputs of the model, and knows nothing about the underlying architecture or weights. Several black-box attacks that require no internal knowledge about the target systems such as gradients, have also been proposed [8, 9, 10].

Su et al. [11] have proposed a black-box attack against image classifiers named one-pixel attack. The one-pixel attack uses differential evolution (DE) to find out which pixel is to be changed and how. DE [12] is a simple yet effective evolutionary algorithm

DOI: 10.24507/ijicic.16.01.405

to solve continuous function optimization problems. In the one-pixel attack, adversarial perturbation for image is encoded to candidate solution of DE. A population of candidate solutions evolve by the application of the mutation and crossover. In [11], the experiments showed that DE is actually possible to deceive image classifiers by changing a single pixel. However, the adopted strategy is classical strategy: DE/rand/1 with no crossover.

Meanwhile, various improved DE methods have been proposed. Among them, JADE [13] is a well-known effective DE variant which employs a control parameter adaptation mechanism. JADE shows a superior performance comparing with classic DE in various benchmark functions. Therefore, in this paper, we apply JADE for search of adversarial perturbation and aim to speed up the generation of adversarial examples. The effectiveness of applying JADE for adversarial attack is evaluated by experiments against state-of-the-art DNN models. This paper is organized as follows. In the next section, we briefly describe algorithm of differential evolution and JADE. Section 3 introduces the adversarial attack using adaptive DE. Section 4 gives the experimental results of adversarial attack using JADE and standard DE against three DNNs. Finally, conclusions and future works are summarized in Section 5.

2. Differential Evolution and JADE.

2.1. Differential evolution.

DE is one of the variants of evolutionary algorithms that use a population. An individual of DE is represented by vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_D)$. There are some variants of DE that have been proposed. The variants are denoted as DE/*base*/*num*/*cross*, where "*base*" denotes the manner of constructing the mutant vector, "*num*" denotes the number of difference vectors, and "*cross*" indicates crossover method. The pseudo-code of DE/rand/1/- is presented in Algorithm 1, where $G_{\max}$ is the maximum number of generations. In the initialization phase, NP individuals $P = \{\boldsymbol{x}_i, i = 1, 2, \ldots, NP\}$ are randomly generated in a given search space. Each individual contains D genes as decision variables.

Algorithm 1: DE/rand/1/-

```
Set scaling factor F and crossover rate CR;
Set the generation number g = 0;
Initialize a population P = {x_1, ..., x_NP} randomly;
for g = 1 to G_max do
    for i = 1 to NP do
        (x_r1, x_r2, x_r3) = randomly selected from P s.t. r1 ≠ r2 ≠ r3 ≠ i;
        v_i = x_r1 + F(x_r2 − x_r3);
        u_i = trial vector generated from x_i and v_i by a crossover;
        if f(u_i) ≥ f(x_i) then
            x_i^new = u_i;
        else
            x_i^new = x_i;
    P = {x_i^new, i = 1, 2, ..., NP};
```

At each generation, DE creates a mutant vector $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ for each individual $\boldsymbol{x}_i$ (called a target vector) in the current population. Some well-known mutation operations are listed as follows.

"rand/1":

$$\boldsymbol{v}_i = \boldsymbol{x}_{r_1} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}) \tag{1}$$

"best/1":

$$\boldsymbol{v}_i = \boldsymbol{x}_{best} + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}) \tag{2}$$

"current-to-best/1":

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F(\boldsymbol{x}_{best} - \boldsymbol{x}_i) + F(\boldsymbol{x}_{r_2} - \boldsymbol{x}_{r_3}) \tag{3}$$

In the above equations, the indices r_1, r_2 and r_3 are distinct integers uniformly chosen from the set $\{1, 2, \ldots, NP\} \setminus \{i\}$. $\boldsymbol{x}_{best}$ is the best individual in the current population. The parameter F is called the scaling factor, which amplifies the difference vectors.

After mutation operation, DE performs crossover operator between target vector and mutant vector, and generates a trial vector $\boldsymbol{u}_i = (u_{i1}, u_{i2}, \ldots, u_{iD})$. There are two mostly used crossover operators used in DE: binomial crossover (bin) and exponential crossover (exp). In this paper, we use binomial crossover as follows:

$$u_{ij} = \begin{cases} v_{ij} & rand_j(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{ij} & \text{otherwise} \end{cases} \tag{4}$$

where $rand_j(0,1)$ is a uniform random number in $(0,1)$ for jth dimension, $j_{rand} \in (1, D)$ is an integer randomly chosen from 1 to D. CR is the crossover rate within the range $(0,1)$ and presents the probability of generating genes for a trial vector $\boldsymbol{u_i}$ from a mutant vector $\boldsymbol{v_i}$. If the jth element u_{ij} of the trial vector $\boldsymbol{u}_i$ is infeasible (i.e., out of the boundary $[L_j, U_j]$), it is reset as follows:

$$u_{ij} = \begin{cases} 2L_j - x_{ij} & (u_j < L_j) \\ 2U_j - x_{ij} & (u_j > U_j) \end{cases} \tag{5}$$

The selection operator is performed to select a better one from the target vector $\boldsymbol{x}_i$ and its corresponding trial vector $\boldsymbol{u}_i$ according to their fitness values $f(\cdot)$. For example, if we have a minimization problem, the selected vector is given by

$$\boldsymbol{x}_i^{new} = \begin{cases} \boldsymbol{u}_i & \text{if } f(\boldsymbol{u}_i) \leq f(\boldsymbol{x}_i) \\ \boldsymbol{x}_i & \text{otherwise} \end{cases} \tag{6}$$

and $\boldsymbol{x}_i^{new}$ is used as a target vector in the next generation. The algorithm is terminated when the maximum number of function evaluations is reached.

2.2. **JADE.** In general, the performance of DE is significantly influenced by the control parameter settings. The optimal control parameter settings depend on not only the characteristics of the objective function but the state of the search progress. Therefore, several adaptive variants of DE have been proposed, and JADE [13] has excellent performance among them. JADE modifies classic DE in three aspects: a new mutation strategy (current-to-$p$best/1), an external archive, and adaptive control of the F, CR parameter values.

The mutation strategy used in JADE is current-to-$p$best/1. It is a variant of the current-to-best/1 strategy where the greediness is adjustable using a parameter p.

$$\boldsymbol{v}_i = \boldsymbol{x}_i + F_i(\boldsymbol{x}_{pbest} - \boldsymbol{x}_i) + F_i(\boldsymbol{x}_{r_1} - \boldsymbol{x}_{r_2}) \tag{7}$$

where $\boldsymbol{x}_{pbest}$ is a vector chosen randomly from the top $p\%$ individuals. F_i is the mutation factor that is associated with $\boldsymbol{x}_i$.

In order to maintain diversity, JADE uses an optional, external archive. Target vectors $\boldsymbol{x}_i$ which were worse than the trial vectors $\boldsymbol{u}_i$ are preserved in archive A. When the archive is used, $\boldsymbol{x}_{r_2}$ in Equation (7) is selected from $P \cup A$. The size of the archive is set

to the same as that of the population (i.e., $|A| = |P|$). Whenever the size of the archive exceeds $|A|$, randomly selected elements are deleted for the newly inserted elements.

Each individual $\boldsymbol{x}_i$ is associated with its own CR_i and F_i and generates trial vectors according to these values. These parameters are set probabilistically at the beginning of each generation using adaptive control parameters μ_{CR}, μ_F according to the following equations:

$$CR_i = \text{randn}_i(\mu_{CR}, 0.1) \tag{8}$$

$$F_i = \text{randc}_i(\mu_F, 0.1) \tag{9}$$

where $\text{randn}_i(\mu_{CR}, 0.1)$ denotes a value generated by a normal distribution of mean μ_{CR} and standard deviation 0.1. $\text{randc}_i(\mu_F, 0.1)$ denotes a value generated by a Cauchy distribution with location parameter μ_F and scale parameter 0.1. In case a value for CR_i outside of $[0, 1]$ is generated, it is replaced by the limit value (0 or 1) closest to the generated value. When $F_i > 1$, F_i is truncated to 1, and when $F_i \leq 0$, Equation (9) is repeatedly applied to try to generate a valid value. At the beginning of the search, μ_{CR} and μ_F are both initialized to 0.5, and adapted during the search as follows.

In selection operation, CR_i and F_i values that succeed in generating a trial vector $\boldsymbol{u}_i$ which is better than the target vector $\boldsymbol{x}_i$ are recorded as S_{CR}, S_F. At the end of the generation, μ_{CR}, μ_F are updated as:

$$\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot \text{mean}_A(S_{CR}) \tag{10}$$

$$\mu_F = (1 - c) \cdot \mu_F + c \cdot \text{mean}_L(S_F) \tag{11}$$

where, c is a positive constant between 0 and 1. $\text{mean}_A(\cdot)$ is an arithmetic mean, and $\text{mean}_L(\cdot)$ is a Lehmer mean which is computed as:

$$\text{mean}_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{12}$$

3. Adversarial Attack Using Adaptive DE.

3.1. Problem description.

Generating adversarial examples in an image classification task can be formalized as an optimization problem with constraints. We assume an input image can be represented by an n-dimensional vector where each scalar element represents one pixel. Let f be the target image classifier which receives n-dimensional inputs and $\boldsymbol{x} = (x_1, \ldots, x_n)$ be the original natural image classified with predicted label t according to f. The probability of $\boldsymbol{x}$ belonging to the class t is $f_t(\boldsymbol{x})$. A vector $e(\boldsymbol{x}) = (e_1, \ldots, e_n)$ represents a specific additive perturbation with respect to a specific natural image $\boldsymbol{x}$. It alters the label of $\boldsymbol{x}$ from t to the target class t_{adv} where $t \neq t_{adv}$ with the modification strength less than maximum modification limitation L. The goal of adversarial attacks is to find the optimized solution $e(\boldsymbol{x})^*$ for Equation (13) or Equation (14). In the case of targeted attack, the target class t_{adv} is designated beforehand, and it can be defined as the following optimization problem:

$$\begin{aligned} \underset{e(\boldsymbol{x})^*}{\text{maximize}} \quad & f_{t_{adv}}(\boldsymbol{x} + e(\boldsymbol{x})) \\ \text{subject to} \quad & ||e(\boldsymbol{x})|| \leq L \end{aligned} \tag{13}$$

Regarding non-targeted attack, the objective function can be defined as the minimization of the soft label for the outputted class $f_t(x)$:

$$\begin{aligned} \underset{e(\boldsymbol{x})^*}{\text{minimize}} \quad & f_t(\boldsymbol{x} + e(\boldsymbol{x})) \\ \text{subject to} \quad & ||e(\boldsymbol{x})|| \leq L \end{aligned} \tag{14}$$

In the few-pixel attack using DE [11], the constraint changes to $||e(\boldsymbol{x})||_0 \leq d$ where d is a small number of dimensions ($d = 1$ for the one-pixel attack). That is, the number of nonzero elements of $e(\boldsymbol{x})$ (i.e., the number of pixels changed) must be less than or equal to d.

3.2. **Applying adaptive DE to finding adversarial perturbation.** To perform one-pixel modification, it is necessary to specify the coordinates to be changed and its RGB values. Figure 1 shows an example of one-pixel modification. In this image, one pixel surrounded by a red circle has changed yellow.

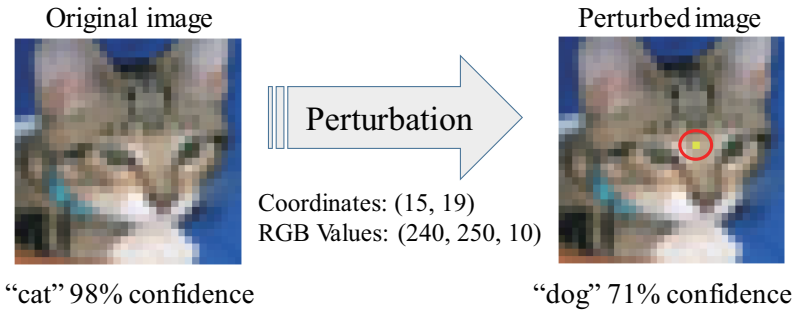


FIGURE 1. An example of one-pixel perturbation

In this paper, we use an adaptive DE variant named JADE to perform adversary attack. To this end, it is necessary to encode the perturbation into an array. Accordingly, the one-pixel modification $\boldsymbol{X}$ is defined as a 5-component tuple:

$$\boldsymbol{X} = (x, y, r, g, b) \tag{15}$$

where x, y are the coordinates of the pixel, and r, g, b are the red, green, and blue values from 0 to 255. The range of x, y are determined by the size of input images. In addition, multiple perturbations can simply be a concatenation of these tuples.

In JADE, these perturbations are treated as individuals which evolve by the genetic operators; therefore, initial perturbations are generated randomly in the search space. In this paper, we use the same coding as the original paper [11]. Assuming a d-pixel attack, each individual is represented as real-valued vector of size $D = d \times 5$ as shown in Figure 2. In standard DE variants, the length of individual vector does not change even if mutation and crossover are executed. Naturally, the solution generated by JADE always satisfies the constraint $||e(\boldsymbol{x})||_0 \leq d$. The elements of the vector will be followed back into integers only when computing image perturbations.

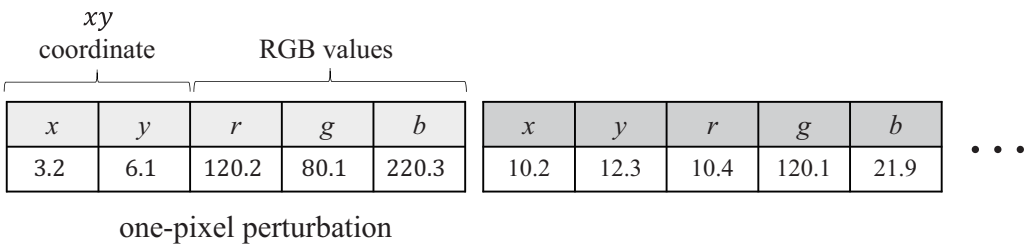


FIGURE 2. Perturbation coding to apply JADE

The fitness function design depends on the type of adversarial attack. The focus of this paper is the non-targeted attack where the purpose is to perturb the original image originally classified correctly by the classifier to cause misclassification. Due to this, when

evaluating individuals, the perturbed image by the individual is input to the classifier, and the value of the probabilistic label (confidence) of the correct class is taken as the fitness. The goal of the non-targeted attack using JADE is to minimize this fitness value.

4. Experiments.

4.1. **Setup.** In this section, in order to confirm that JADE is useful in generation of adversarial examples, we conduct adversarial attack using JADE and standard DEs and compare their performances. We trained three types of common networks: LeNet [14], ResNet [15], and DenseNet [16] as target image classifiers on cifar-10 dataset [17]. The cifar-10 dataset consists of 60000 32×32 colour images split into 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Table 1 shows the number of parameters of the networks and accuracy for cifar-10.

TABLE 1. The number of parameters of the networks and accuracy for cifar-10

name	#parameters	accuracy
LeNet	62006	0.7488
ResNet	470218	0.9231
DenseNet	850606	0.9467

For each model, 200 images are randomly selected from correctly classified test images, and non-targeted one and three pixel attacks using JADE and DE are performed. The strategies of DE are DE/rand/1/bin and DE/best/1/bin with $F = 0.5$ and $CR = 0.9$. In JADE and both DEs, the population size NP is set to 100. The parameter values of JADE are selected as $c = 0.1$ and $p = 5\%$, as suggested in [13]. The maximum number of generation $G_{\max}$ is set to 200.

As a result of each attack, we calculate attack success rate and difference of confidence c_{diff}. The success rate is defined as the percentage of adversarial images that were classified by the target DNN as a class different from true class (i.e., the probability of misclassification). c_{diff} is the difference between the true class probability of the original image and the probability of it in the adversarial image. This measure indicates how much perturbations have decreased the probability of the true class.

4.2. **Results.** Table 2 and Table 3 show success rate and c_{diff} in each adversarial attack respectively. For all models, JADE achieves almost the highest success rate and c_{diff}. It can be seen that the success rate of the attack improves as the number of pixels increases, and also the difference in performance between JADE and both standard DEs becomes greater. In other words, the larger the search space, the more effectively JADE's parameter adaptation works. Besides, the model with more parameters has a higher robustness against adversarial attacks.

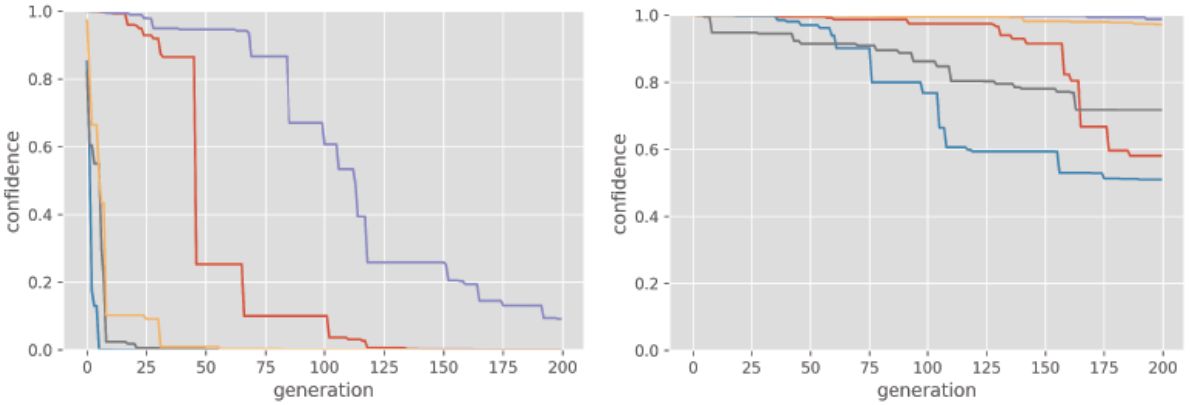
Next, the change of fitness values of JADE in DenseNet is shown in Figure 3. When the attack is successful, the fitness decreases from the early stage of the search, and it

TABLE 2. Success rates of conducting one-pixel attack and three-pixel attack on each model

	LeNet		ResNet		DenseNet	
	#pixels = 1	#pixels = 3	#pixels = 1	#pixels = 3	#pixels = 1	#pixels = 3
JADE	**53.5**%	**91.0**%	**32.5**%	**77.5**%	**28.0**%	**69.5**%
DE/rand/1/bin	49.0%	88.0%	26.0%	73.0%	25.5%	64.5%
DE/best/1/bin	53.0%	79.0%	32.0%	63.0%	27.5%	56.0%

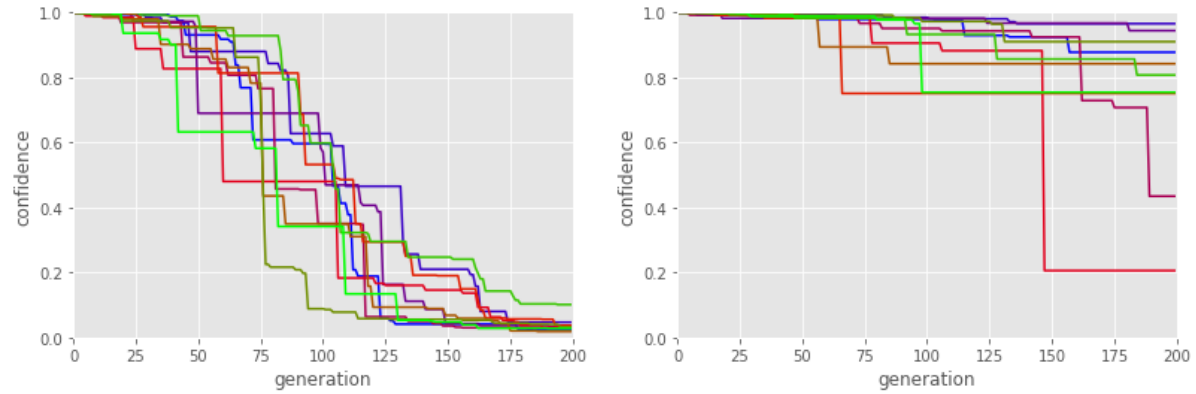
TABLE 3. c_{diff} of conducting one-pixel attack and three-pixel attack on each model

	LeNet		ResNet		DenseNet	
	#pixels = 1	#pixels = 3	#pixels = 1	#pixels = 3	#pixels = 1	#pixels = 3
JADE	0.430	**0.765**	**0.308**	**0.760**	**0.253**	**0.677**
DE/rand/1/bin	0.392	0.734	0.258	0.717	0.234	0.627
DE/best/1/bin	**0.431**	0.636	0.305	0.613	0.250	0.535



(a) Successful attack (b) Unsuccessful attack

FIGURE 3. The change of fitness values of 5 random images in three-pixel attack using JADE against DenseNet

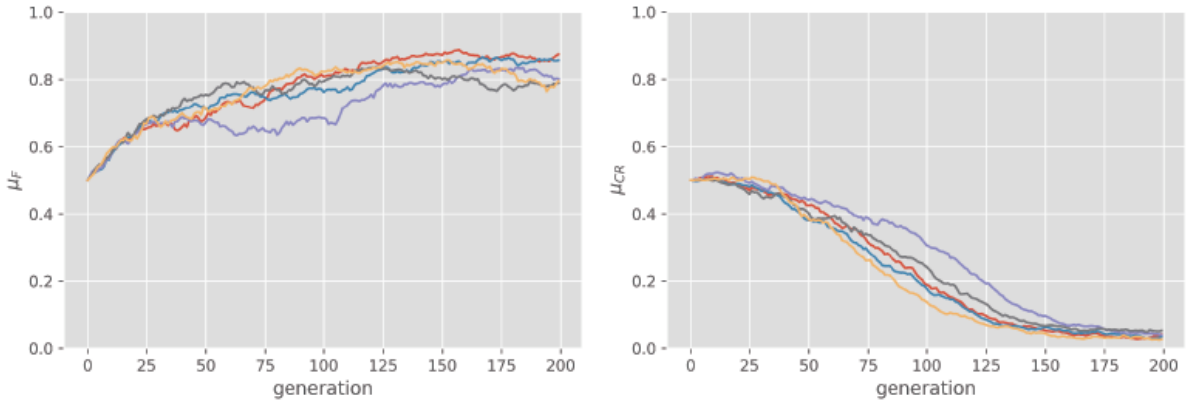


(a) JADE (b) DE/rand/1/bin

FIGURE 4. The change of fitness values of 10 independent trials for a certain one image in three-pixel attack against DenseNet

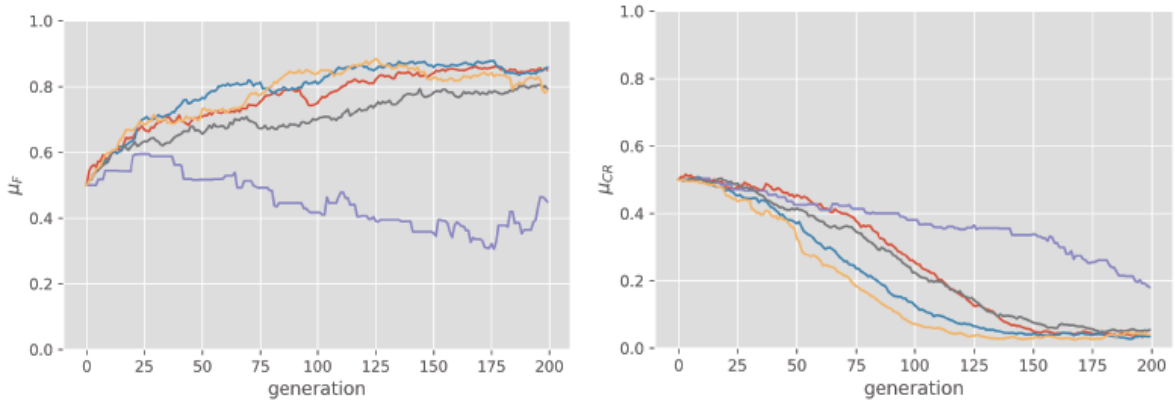
is finally close to 0. Meanwhile, in the case of failure, the decreasing speed of fitness is slow. Also, for some images, the fitness hardly decreases and it seems that population gets trapped in a local optimum. Furthermore, in order to compare the convergence speed of JADE and DE/rand/1/bin, the results of 10 independent trials for a certain one image in three-pixel attack against DenseNet are shown in Figure 4. The results clearly show that the convergence speed of JADE is faster and more stable than DE/rand/1/bin.

Figure 5 and Figure 6 show the change of μ_F and μ_{CR} values in three-pixel attack against DenseNet. Figure 5 shows the parameters for successful attack, and Figure 6 shows the case for failure. Regardless of the success or failure of the attack, the parameter adaptation



(a) μ_F (b) μ_{CR}

FIGURE 5. The change of μ_F and μ_{CR} values of 5 random success images in three-pixel attack using JADE against DenseNet



(a) μ_F (b) μ_{CR}

FIGURE 6. The change of μ_F and μ_{CR} values of 5 random failure images using JADE against DenseNet

of JADE showed almost the same result in all models. μ_F tends to increase progressively, and μ_{CR} conversely tends to decrease. Since a low CR is suitable, it is thought that the dependency between variables is small in the search for perturbations. Low values of CR produce exploratory moves parallel to a small number of search space axes [18], which means that JADE independently searched for perturbation positions and colors on the image.

Figure 7 shows sample images of one and three-pixel attack using JADE. These images are the case when the attack is successful, and the output labels are different from the true class label of the input respectively. However, since these are generated by non-targeted attack, it cannot be designated in advance to which class the perturbation image is misclassified.

5. **Conclusion.** Most machine learning models have been shown susceptible to adversarial examples resulting from adding small perturbations to inputs. In image classification task, adversarial attack with DE enables to generate adversarial examples by perturbing only few pixels. In this study, we used adaptive DE, called JADE, for generating adversarial examples and tried to speed up the search of perturbations. In order to confirm

FIGURE 7. Sample images of one and three-pixel attack using JADE

the difference in the performance of generation of adversarial examples using JADE and DE, we performed one and three-pixel attack on three different DNN models on cifar-10 dataset. From the experimental results we confirmed that the attack success rate of JADE outperformed two standard DEs (DE/rand/1/bin and DE/best/1/bin) in all models. In addition, it was confirmed that μ_F and μ_{CR} of JADE showed similar behavior regardless of the success or failure of the attack. Since μ_{CR} becomes smaller and μ_F becomes larger, a global exploratory search is considered to be effective to find adversarial perturbation.

In the future, we will intend to conduct fitness landscape analysis in the search of adversarial example, and then verify the effectiveness of the adversarial attack using other evolutionary algorithms such as particle swarm optimization or real-coded genetic algorithm.

REFERENCES

[1] A. Krizhevsky, I. Sutskever and G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems*, pp.1097-1105, 2012.

[2] C. Liu, F. Liu, L. Wang, L. Ma and Z.-M. Lu, Segmentation of nerve on ultrasound images using deep adversarial network, *International Journal of Innovative Computing, Information and Control*, vol.14, no.1, pp.53-64, 2018.

[3] I. M. Kamal, H. Bae, S. Sim, H. Kim, D. Kim, Y. Choi and H. Yun, Forecasting high-dimensional multivariate regression of Baltic Dry Index (BDI) using Deep Neural Networks (DNN), *ICIC Express Letters*, vol.13, no.5, pp.427-434, 2019.

[4] S. Moosavi-Dezfooli, A. Fawzi and P. Frossard, Deepfool: A simple and accurate method to fool deep neural networks, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.2574-2582, 2016.

[5] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik and A. Swami, The limitations of deep learning in adversarial settings, *The 1st IEEE European Symposium on Security and Privacy (EuroS&P)*, Saarbrucken, Germany, pp.372-387, 2016.

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

[7] I. J. Goodfellow, J. Shlens and C. Szegedy, Explaining and harnessing adversarial examples, *arXiv Preprint*, arXiv:1412.6572, 2014.

[8] H. Dang, Y. Huang and E. Chang, Evading classifiers by morphing in the dark, *Proc. of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp.119-133, 2017.

[9] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik and A. Swami, Practical black-box attacks against machine learning, *Proc. of the 2017 ACM on Asia Conference on Computer and Communications Security*, pp.506-519, 2017.

[10] N. Narodytska and S. P. Kasiviswanathan, Simple black-box adversarial perturbations for deep networks, *arXiv Preprint*, arXiv:1612.06299, 2016.

[11] J. Su, D. V. Vargas and K. Sakurai, One pixel attack for fooling deep neural networks, *CoRR*, vol.abs/1710.08864, 2017.

[12] R. Storn and K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, vol.11, no.4, pp.341-359, 1997.
[13] J. Zhang and A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, *IEEE Trans. Evolutionary Computation*, vol.13, no.5, pp.945-958, 2009.
[14] Y. LeCun et al., *Lenet-5, Convolutional Neural Networks*, http://yann.lecun.com/exdb/lenet, 2015.
[15] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.770-778, 2016.
[16] G. Huang, Z. Liu and K. Q. Weinberger, Densely connected convolutional networks, *CoRR*, vol.abs/1608.06993, 2016.
[17] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features from Tiny Images*, Technical Report, 2009.
[18] J. Montgomery, Crossover and the different faces of differential evolution searches, *IEEE Congress on Evolutionary Computation*, pp.1-8, 2010.