One-Pixel Adversarial Example that Is Safe for Friendly Deep Neural Networks

Hyun Kwon[1], Yongchul Kim[2], Hyunsoo Yoon[1], and Daeseon Choi[3](✉)

[1] School of Computing, Korea Advanced Institute of Science and Technology, Daejeon, South Korea
[2] Department of Electrical Engineering, Korea Military Academy, Seoul, South Korea
[3] Department of Medical Information, Kongju National University, Gongju, South Korea
sunchoi@kongju.ac.kr

Abstract. Deep neural networks (DNNs) offer superior performance in machine learning tasks such as image recognition, speech recognition, pattern analysis, and intrusion detection. In this paper, we propose a one-pixel adversarial example that is safe for friendly deep neural networks. By modifying only one pixel, our proposed method generates a one-pixel-safe adversarial example that can be misclassified by an enemy classifier and correctly classified by a friendly classifier. To verify the performance of the proposed method, we used the CIFAR-10 dataset, ResNet model classifiers, and the Tensorflow library in our experiments. Results show that the proposed method modified only one pixel to achieve success rates of 13.5% and 26.0% in targeted and untargeted attacks, respectively. The success rate is slightly lower than that of the conventional one-pixel method, which has success rates of 15% and 33.5% in targeted and untargeted attacks, respectively; however, this method protects 100% of the friendly classifiers. In addition, if the proposed method modifies five pixels, this method can achieve success rates of 20.5% and 52.0% in targeted and untargeted attacks, respectively.

Keywords: Deep neural network (DNN) · Adversarial example · One-pixel attack · Differential evolution (DE)

1 Introduction

Deep neural networks (DNNs) [12] have been widely used for image recognition, speech recognition, pattern analysis, and intrusion detection. However, adversarial examples [15] are a serious threat to DNNs. An adversarial example is a distorted sample that adds a small amount of noise to the original sample; this can lead to misclassification of the DNN. Adversarial examples and their effects have been extensively studied.

In recent years, one-pixel adversarial attacks [14] have emerged as a threat to DNNs. Unlike conventional adversarial attacks, this method causes DNN misclassification by modifying a single pixel and is useful for applications such as

B. B. Kang and J. Jang (Eds.): WISA 2018, LNCS 11402, pp. 42–54, 2019.
https://doi.org/10.1007/978-3-030-17982-3_4

stickers. However, this method has not been considered for situations such as military scenarios, where enemy forces and friendly forces are mixed.

The adversarial example of friendly forces can be useful in situations such as military engagements. Because battlegrounds are shared by enemy forces and friendly forces, friend-safe adversarial examples [9] that can be misclassified by enemy classifiers and correctly classified by friendly classifiers could prove to be an invaluable tool. For example, it may be necessary to modify the road signs on a battlefield to deceive only the enemy's self-driven vehicles.

Thus, we propose an advanced one-pixel adversarial example that preserves the recognition of friendly classifiers. By modifying only one pixel, the proposed method can generate a one-pixel-safe adversarial example that can be misclassified by enemy classifiers and correctly classified by friendly classifiers. This paper makes the following contributions:

- First, we propose a one-pixel-safe adversarial example by modifying a single pixel in a DNN and systematically organize the frameworks of the proposed scheme.
- The proposed scheme has two configurations: targeted and untargeted attacks. The proposed method can generate the one-pixel-safe adversarial example in for both configurations.
- We used the CIFAR-10 dataset to validate the performance and analyze the success rates for targeted attack and untargeted attacks. In addition, we analyzed the performance of this method by modifying groups of three and five pixels.

The remainder of this paper is structured as follows: Sect. 2 reviews related works. Our proposed adversarial example attack is presented in Sect. 3. The experiments and evaluation are shown in Sect. 4, and a discussion of the proposed system is presented in Sect. 5. Finally, conclusions are given in Sect. 6.

2 Related Works

The study of adversarial examples was introduced by Szegedy et al. [15] in 2014. The main goal of using an adversarial example is to induce the DNN into making a mistake by adding a small amount of noise to the original image such that humans cannot tell the difference between the original and the distorted image.

The basic method for generating adversarial examples is described in Sect. 2.1. Adversarial examples can be categorized in three ways: recognition of an adversarial example, information on the target model information, and method for generation, as described in Sects. 2.2–2.4.

2.1 Adversarial Example Generation

The basic architecture that generates an adversarial example comprises two elements: a target model and a transformer. The transformer takes the original

sample x and original class y as input data. Next, the transformer creates a transformed example $x^* = x + w$, with noise value w added to the original sample x as output; the transformed example x^* is given as input data to the target model. The target model then provides the transformer with the class probability results for the transformed example. Following this, the transformer updates the noise values w in the transformed example $x^* = x + w$ so that the other class probabilities are higher than the original class probabilities, while minimizing the distortion distances between x^* and x.

2.2 Categorization by Recognition of Adversarial Example

According to the class that the target model recognizes from the adversarial examples, we can divide these examples into two subcategories: targeted and untargeted. A targeted adversarial example is an adversarial example that causes the target model to recognize the adversarial image as a particular intended class. This can be expressed mathematically as follows:

Given a target model and original sample $x \in X$, the problem is an optimization problem that generates a targeted adversarial example x^*:

$$x^* : \underset{x^*}{\operatorname{argmin}}\ L(x, x^*)\ \text{ s.t. }\ f(x^*) = y^*$$

where $L(\cdot)$ is the distance measure between the original sample x and the transformed example x^*, and y^* is the particular intended class. $f(\cdot)$ is an operation function that provides class results for the input values of the target model.

An untargeted adversarial example is an adversarial example that causes the target model to recognize the adversarial image as a class other than the original class. It can be expressed mathematically as follows:

Given a target model and original sample $x \in X$, the problem is an optimization problem that generates an untargeted adversarial example x^*:

$$x^* : \underset{x^*}{\operatorname{argmin}}\ L(x, x^*)\ \text{ s.t. }\ f(x^*) \neq y$$

where $y \in Y$ is the original class.

Untargeted adversarial examples have the advantages of less distortion from the original image and shorter learning time when compared with targeted adversarial examples. However, targeted adversarial examples are more sophisticated attacks as they are misclassified as target classes chosen by the attacker.

2.3 Categorization by Information on Target Model

Depending on the amount of target information required, attacks that generate adversarial examples can also be divided into two types: white box attacks and black box attacks. However, black box attacks only require an input response; no additional information about the target is needed. In this paper, the proposed method is a white box attack that knows both the enemy and friendly classifiers.

2.4 Categorization by Method for Adversarial Example Generation

There are four typical attacks that generate adversarial examples. The first is the fast-gradient sign method (FGSM) [4], which can find x^* through L_∞:

$$x^* = x + \epsilon \cdot \text{sign}(\bigtriangledown loss_{F,t}(x))$$

where F is an object function and t is a target class. In every FGSM iteration, the gradient is updated by ϵ from the original x, and x^* is found through optimization. This method is simple and demonstrates good performance.

The second attack is iterative FGSM (I-FGSM) [8], which is an updated version of the FGSM. Instead of changing the amount ϵ in each step, the smaller amount of α is updated and eventually clipped by the same ϵ:

$${x_i}^* = {x_{i-1}}^* - \text{clip}_\epsilon(\alpha \cdot \text{sign}(\bigtriangledown loss_{F,t}({x_{i-1}}^*)))$$

The I-FGSM provides better performance than the FGSM.

The third type of attack is the Deepfool method [10], which is an untargeted attack and uses the L_2 distance measure. This method generates an adversarial example that is more efficient than the FGSM and as close as possible to the original image. To generate an adversarial example, this method constructs of a neural network and looks for x^* using linearization approximation. However, because the neural network is not completely linear, we must find the adversarial example through multiple iterations; i.e., it is a more complicated process than that of the FGSM.

The fourth attack method is the Carlini attack [2], which is the latest attack method and delivers better performance than the FGSM and I-FGSM. This method can achieve a 100% success rate, even against a distillation structure [11], which was recently introduced in the literature. The key principle of this method involves the use of a different objective function:

$$D(x, x^*) + c \cdot f(x^*)$$

Instead of using the conventional objective function $D(x, x^*)$, this method proposes a means to find an appropriate binary c value. In addition, it suggests a method to control the attack success rate even with some increased distortion by reflecting the confidence value as follows:

$$f(x^*) = \max(Z(x^*)_t - \max\{Z(x^*)_t : i \neq t\}, -k)$$

where $Z(\cdot)$ represents the pre-softmax classification result vector and t is a target class.

The four previously mentioned methods add a small amount of noise to the entire original sample, causing misclassification. In a recent study, Su et al. [14] proposed a one-pixel attack that causes misclassification by modifying a single pixel. In one-pixel attacks, differential evolution [3,13] is used as the optimizer. The advantage of one-pixel attacks is that they do not affect the rest of the pixels in the original sample. For example, a traffic sign could be attacked by modifying one pixel in the sign when it is deployed. In this paper, the proposed method is constructed by applying a similar one-pixel attack.

3 Proposed Method

To generate a one-pixel-safe adversarial example, we propose a network architecture that consists of a transformer, a friendly discriminator D_{friend}, and an enemy discriminator D_{enemy}, as shown in Fig. 1.

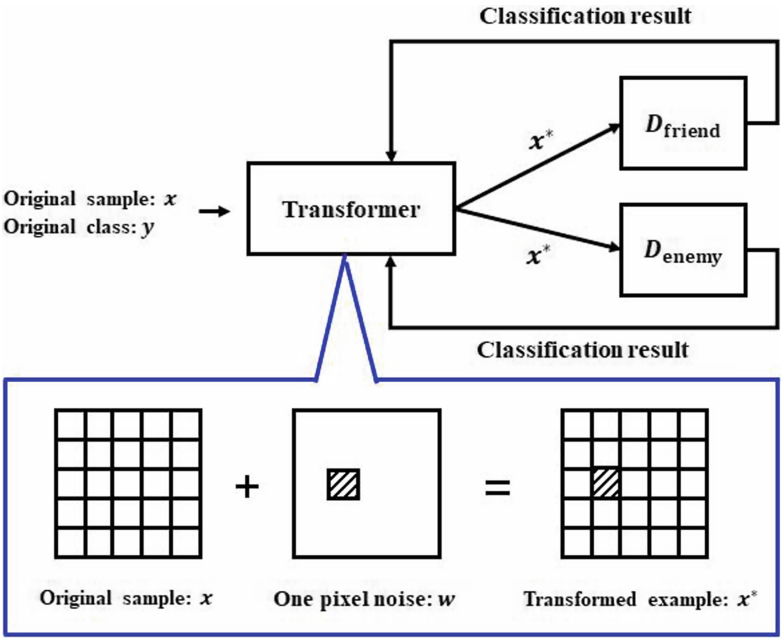


Fig. 1. Proposed architecture.

The transformer takes the original sample $x \in X$ and the original class $y \in Y$ as input and converts the original sample to the transformed example x^*. Furthermore, D_{friend} and D_{enemy} are pre-trained classifiers and are not changed during transformation. They take x^* as input and provide their classification result (i.e., confidence) to the transformer.

The goal of this architecture is to add one pixel of noise to the original sample so that the transformed example x^* is misclassified by D_{enemy} and correctly classified by D_{friend}. There are two configurations in which the transformed example x^* is incorrectly classified by D_{enemy}: the targeted adversarial and untargeted adversarial examples. In mathematical expressions, the operation functions of D_{enemy} and D_{friend} are denoted as $f^{\text{enemy}}(x)$ and $f^{\text{friend}}(x)$, respectively. Given the pre-trained D_{friend} and D_{enemy} and the original input $x \in X$, we have an optimization problem that generates the targeted adversarial example x^*:

$$x^* : \underset{x^*}{\text{argmin}}\ L(x, x^*)\ \text{ s.t. }\ f^{\text{friend}}(x^*) = y \text{ and } f^{\text{enemy}}(x^*) = y^*,$$

where $L(\cdot)$ is the chosen measure of the distance between the original sample x and transformed example x^*, and $y^* \in Y$ is the target class chosen by the attacker. An untargeted adversarial example x^* is similarly generated:

$$x^* : \underset{x^*}{\operatorname{argmin}} L(x, x^*) \text{ s.t. } f^{\text{friend}}(x^*) = y \text{ and } f^{\text{enemy}}(x^*) \neq y.$$

This procedure consists of pre-training D_{friend} and D_{enemy} and creating a transformation that generates a one-pixel-safe adversarial example, x^*. First, D_{friend} and D_{enemy} are trained to classify the original sample x.

$$f^{\text{friend}}(x) = y \in Y \text{ and } f^{\text{enemy}}(x) = y \in Y.$$

In our experiments, D_{friend} and D_{enemy} were trained to classify the original samples using CIFAR-10 with more than 92% accuracy. Second, the transformer accepts the original sample and original class as input and produces the transformed example x^*. For this study, we modified the transformer architecture given in [9,14], and defined x^* as

$$x^* = x + w,$$

where $x = (x_1, ..., x_n)$ is the original sample with n n-dimensional inputs and $w = (w_1, ..., w_n)$ is noise with n-dimension.

The classification results of x^* by D_{friend} and D_{enemy} are returned to the transformer. The transformer then calculates the total objection function, $f^{\text{T}}(x^*)$, and generates a one-pixel-safe adversarial example x^* by iteratively maximizing $f^{\text{T}}(x^*)$. $f^{\text{T}}(\cdot)$ is defined as

$$\underset{x^*}{\text{maximize}}\ f^{\text{T}}(x^*) = f_y^{\text{friend}}(x^*) + f_{y^*}^{\text{enemy}}(x^*) \text{ subject to } \|w\|_0 \leq d, \quad (1)$$

where $f_y^{\text{friend}}(x^*)$ and $f_{y^*}^{\text{enemy}}(x^*)$ are the objection functions of D_{friend} and D_{enemy}, and d is 1 as the one-pixel attack. In d dimension, all but one of the remaining pixels in the original sample are zero.

To satisfy the Eq. (1), the one-pixel attack uses differential evolution (DE) [3,13]. DE is a population-based optimization algorithm used to solve optimization problems for multiple models. During iteration, this method generates candidate solutions (children) based on the current solution (parent). By comparing the children with their parent, one of two solutions is selected; specifically, this method aims to find the solution with the highest fitness value. The DE equation [3,13,14] is as follows:

$$x_i(g+1) = x_a(g) + F(x_b(g) + x_c(g)),$$

$$a \neq b \neq c,$$

where x_i is the element of the candidate solution; a, b, and c are arbitrary values; F is a scale parameter set; and g is the current generation index. When a candidate solution (child) is generated, the solution with the highest fitness values will survive by comparing the candidate solution (child) with the parent

or current solution (parent). The above process is continued until the iteration is over. Using the DE method, our proposed method generates x^* by maximizing f^{T}.

To satisfy $f^{\mathrm{friend}}(x^*) = y$, $f_y^{\mathrm{friend}}(x^*)$ should be maximized as

$$\underset{x^*}{\text{maximize}}\ f_y^{\mathrm{friend}}(x^*),$$

where y is the original class.

However, f^{enemy} presents two cases (targeted and untargeted adversarial examples).

To satisfy $f^{\mathrm{enemy}}(x^*) = y^*$, in targeted adversarial examples, $f_{y^*}^{\mathrm{enemy}}(x^*)$ should be maximized as

$$\underset{x^*}{\text{maximize}}\ f_{y^*}^{\mathrm{enemy}}(x^*),$$

where y^* is the targeted class.

To satisfy $f^{\mathrm{enemy}}(x^*) \neq y$ in an untargeted adversarial example, $f_{y^*}^{\mathrm{enemy}}(x^*)$ should be maximized as

$$\underset{x^*}{\text{maximize}}\ f_{y^*}^{\mathrm{enemy}}(x^*) \ = \underset{x^*}{\text{maximize}}\ \{-f_y^{\mathrm{enemy}}(x^*)\},$$

where y is the original class. The procedure for generating a one-pixel-safe adversarial example is detailed in Algorithm 1.

Algorithm 1. one-pixel-safe adversarial example generation in a transformer.

Input: original sample x, one-pixel noise w, original class y, targeted class y^*, iterations r, dimension d.

Targeted adversarial example generation:

$d \leftarrow 1$
$x^* \leftarrow 0$
for r step **do**
  $x^* \leftarrow x + w$
  Update w by maximizing $f_y^{\mathrm{friend}}(x^*) + f_{y^*}^{\mathrm{enemy}}(x^*)$ subject to $\|w\|_0 \leq d$
end for
return x^*

Untargeted adversarial example generation:

$d \leftarrow 1$
$x^* \leftarrow 0$
for r step **do**
  $x^* \leftarrow x + w$
  Update w by maximizing $f_y^{\mathrm{friend}}(x^*) - f_y^{\mathrm{enemy}}(x^*)$ subject to $\|w\|_0 \leq d$
end for
return x^*

4 Experiment and Evaluation

Our experiments showed that the proposed scheme can generate a one-pixel-safe adversarial example that is incorrectly classified by enemy classifiers and

correctly classified by friendly classifiers. We used the Tensorflow [1] library (a widely used open source library) for machine learning on a Xeon E5-2609 1.7-GHz server.

4.1 Experimental Method

In this experiment, we used the CIFAR-10 dataset [7] (planes, cars, birds, cats, deer, dogs, frogs, horses, boats, and trucks). The CIFAR-10 dataset consists of 50,000 training data and 10,000 test data. The experimental method consisted of (1) pre-training D_{friend} and D_{enemy} and (2) transforming the one-pixel-safe adversarial example.

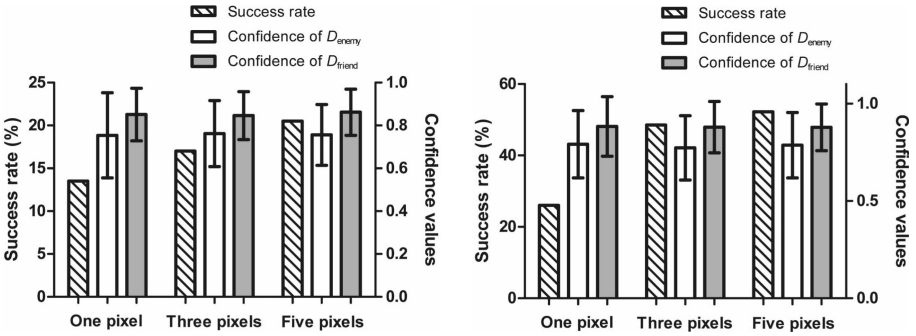
First, during pre-training, D_{friend} and D_{enemy} were common Resnet networks [5]. Their configuration and training parameters are shown in Tables 2 and 3 of the appendix. 50,000 training data were used to train D_{friend} and D_{enemy}. During

Table 1. A one-pixel-safe adversarial example for each target class that was misclassified by D_{enemy} for each original sample: plane "0," cars "1," birds "2," cats "3," deer "4," dogs "5," frogs "6," horses "7," boats "8," and trucks "9."

Original	Targeted classes misclassified by D_{enemy}									
	"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"
										
										
										
										

testing, D_{friend} and D_{enemy} correctly classified the original samples with 92.15% and 92.31% accuracy, respectively.

Next, DE optimization was applied to generate the one-pixel-safe adversarial example [3,13]. The population size was 400, the scale parameter set was 0.5, the iteration was 100, and perturbation was 1. The initial population used the uniform distribution $U(1, 32)$ to generate the x–y coordinates, and the RGB values followed an average of 128 and a standard deviation of 127 for the Gaussian distribution. For a given number of iterations, the transformer updated the output x^* and provided it to D_{friend} and D_{enemy}, from which it receives feedback. At the end of the iterations, the transformation result x^* was evaluated in terms of the accuracy of D_{friend}, which was the attack success rate. In detail, the accuracy of D_{friend} is the coincidence rate between the original class and the output class of D_{friend}; the attack success rate is the rate at which D_{enemy} incorrectly classifies x^*. The attack success rate has two configurations: targeted and untargeted. The targeted attack success rate is the coincidence rate between the targeted class and the class output by D_{enemy}; the untargeted attack success rate is the rate of inconsistency between the original class and the output class of D_{enemy}.



(a) Targeted attack (b) Untargeted attack

Fig. 2. The success rate and confidence of D_{enemy} and D_{friend} for 200 random adversarial examples generated by modifying one-, three-, and five-pixel attacks.

4.2 Experimental Results

The evaluation of one-pixel-safe adversarial examples is divided into targeted and untargeted adversarial examples. In addition, we analyzed the success rate and confidence by additionally modifying groups of three and five pixels.

Targeted Attack. Table 1 shows one-pixel-safe adversarial examples that were incorrectly classified as the targeted class by D_{enemy} and correctly classified as the original class by D_{friend} for each original sample. Furthermore, Table 1 shows that the one-pixel-safe adversarial example is similar to the original sample; specifically, there is a difference of only one pixel.

Figure 2(a) shows the success rate and confidence of D_{enemy} and D_{friend} for one-, three-, and five-pixel targeted attacks. In the one-pixel attack, the success rate was 13.5% where the target attack success rate and friend accuracy were 100%. The confidences of the one-pixel attack also showed that the values of 0.754 and 0.851 for D_{enemy} and D_{friend} were the highest fitness values. Thus, we know that it is more difficult to deceive the enemy classifier owing to the high confidence in the friend classifier. Figure 2(a) also shows that the success rate increased along with the number of changeable pixels. In the five-pixel attack, the success rate was 20.5%.

Untargeted Attack. Figure 2(b) shows the success rate and confidence of D_{enemy} and D_{friend} for one-, three-, and five-pixel untargeted attacks. In the one-pixel attack, the success rate was 26.0% when the untargeted attack success rate and the friend accuracy were 100%. Because untargeted attacks are easier to optimize than targeted attacks, the success rate of untargeted attacks was 12.5% higher. Similar to targeted attacks, Fig. 2(b) shows that the success rate increased along with the number of changeable pixels. In the five-pixel attack, the success rate was 52.0%.

5 Discussion

Attack Method Consideration. Our proposed method added one-pixel noise with strange colors to the original sample; however, human perception was maintained at 100%. In addition, one out of 3,072 pixels of CIFAR-10 is about 0.03% part, meaning one-pixel noise was very low.

We considered targeted or untargeted attacks depending on priority differences between the success rate and the goals of the target model misrecognition. If the success rate was more important, the attacker chose an untargeted attack. If an attacker wished to change the misclassification into a target class of their choice, they used a targeted attack.

The assumption of the proposed method was a white box attack that has identified the enemy and friendly classifiers. Since DE optimization searches for one pixel with the highest confidence value, the proposed method must know the classification results of D_{enemy} and D_{friend} to derive the input values.

Application. The one-pixel-safe adversarial example can be applied to practical applications (such as stickers). For example, once a traffic left sign has been deployed, a hybrid adversarial left sign can be generated by replacing one pixel that was generated in advance. Thus, a one-pixel-safe adversarial left sign can be misclassified as a right sign by an enemy vehicle and can be correctly classified as a left sign by a friendly vehicle.

Limitation. The proposed method has a lower success rate than the conventional one-pixel attack method. In the ResNet model [5], the conventional one-pixel attack achieved a higher success rate than the proposed method with 15.0% and 33.5% success rates for targeted and untargeted attacks, respectively. Since the proposed method has a higher recognition condition for a friendly classifier, it is difficult to confirm that a modification of one pixel satisfies a one-pixel-safe adversarial example. To increase the success rate of the proposed method, we must also increase the number of pixels that can be modified.

6 Conclusion

In this paper, we proposed a novel one-pixel-safe adversarial example by modifying a single pixel. The proposed method generated a one-pixel-safe adversarial example by adding one pixel of noise to the original sample, thereby causing misclassification for enemy classifiers and maintaining correct recognition for friendly classifiers. Experimental results on CIFAR-10 data confirmed that the proposed method showed a success rate of 13.5% and 26.0% for targeted and untargeted attacks, respectively. Although these rates are slightly lower than those of the conventional one-pixel method (15% and 33.5%), our proposed method protects 100% of friendly classifiers. When the proposed method was applied to five pixels, the success rates were 20.5% and 52.0% for targeted and untargeted attacks, respectively. This method can also be used in applications such as stickers.

In future work, we will expand the proposed method to new datasets, such as the ImageNet and Voice. In addition, future work will involve the implementation of new algorithms to improve upon the results achieved in these experiments. Finally, challenges related to countermeasures against our proposed method must be addressed.

Acknowledgement. This work was supported by National Research Foundation (NRF) of Korea grants funded by the Korean government (MSIT) (2016R1A4A1011761 and 2017R1A2B4006026) and an Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korean government (MSIT) (No. 2016-0-00173).

Appendix

Table 2. D_{friend} and D_{enemy} model of 34-layer ResNet [5]

Layer type	Model shape
#1 layer Convolution+ReLU	[7, 7, 64]
Max pooling	[3, 3]
#2 layer Convolution+ReLU	[1, 1, 64]
#2 layer Convolution+ReLU	[3, 3, 64]
#2 layer Convolution+ReLU	[1, 1, 256]
#2 layer Repeat	3 times
#3 layer Convolution+ReLU	[3, 3, 128]
#3 layer Convolution+ReLU	[3, 3, 128]
#3 layer Repeat	4 times
#4 layer Convolution+ReLU	[3, 3, 256]
#4 layer Convolution+ReLU	[3, 3, 256]
#4 layer Repeat	6 times
#5 layer Convolution+ReLU	[3, 3, 512]
#5 layer Convolution+ReLU	[3, 3, 512]
#5 layer Repeat	3 times
Fully connected+ReLU	[1000]
Softmax	[10]

Table 3. D_{friend} and D_{enemy} model parameters.

Parameter	Values
Learning rate of SGD [6]	0.1
Momentum	0.9
Delay rate	1 (decay 0.0001)
Iteration	50,000
Batch size	128
Epochs	200

References

1. Abadi, M., et al.: TensorFlow: a system for large-scale machine learning. OSDI. **16**, 265–283 (2016)
2. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57. IEEE (2017)
3. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011)
4. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
6. Ketkar, N.: Stochastic gradient descent. In: Ketkar, N. (ed.) Deep Learning with Python, pp. 111–130. Apress, Berkeley (2017). https://doi.org/10.1007/978-1-4842-2766-4_8
7. Krizhevsky, A., Nair, V., Hinton, G.: The CIFAR-10 dataset (2014). http://www.cs.toronto.edu/kriz/cifar.html
8. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. In: ICLR Workshop (2017)
9. Kwon, H., Kim, Y., Park, K.W., Yoon, H., Choi, D.: Friend-safe evasion attack: an adversarial example that is correctly recognized by a friendly classifier. Comput. Secur. **78**, 380–397 (2018)
10. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: DeepFool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2574–2582 (2016)
11. Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A.: Distillation as a defense to adversarial perturbations against deep neural networks. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 582–597. IEEE (2016)
12. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
13. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)
14. Su, J., Vargas, D.V., Kouichi, S.: One pixel attack for fooling deep neural networks. arXiv preprint arXiv:1710.08864 (2017)
15. Szegedy, C., et al.: Intriguing properties of neural networks. In: International Conference on Learning Representations (2014). http://arxiv.org/abs/1312.6199