Testing Convolutional Neural Network using Adversarial Attacks on Potential Critical Pixels

Bo-Cheng Lin
National Chiao Tung University
ds934118.cs07g@nctu.edu.tw

Hwai-Jung Hsu
Feng Chia University
hjhsu@mail.fcu.edu.tw

Shih-Kun Huang
National Chiao Tung University
skhuang@cs.nctu.edu.tw

***Abstract*—Convolutional neural networks (CNNs) are known to be vulnerable to adversarial attacks. Well-crafted perturbations to the inputs can mislead a state-of-the-art CNN to make wrong decisions. Therefore, there is a pressing need for the development of methods that can test or detect the vulnerability of CNNs. In this study, we propose an adversarial attack method, called Dual Iterative Fusion (DIF) with potential critical pixels, for CNN testing to reveal the vulnerability of CNNs. DIF modifies as few as 5 pixels out of 32x32 images in this study and achieves faster, less noticeable, and more targeted attacks to a CNN. Testing CNNs with DIF, we observed that some classes are more vulnerable than the others within many classical CNNs for image classification. In other words, some classes are susceptible to misclassification due to adversarial attacks. For example, in VGG19 trained with CIFAR10 data set, the vulnerable class is "Cat". The successfully-targeted attack rate of class "Cat" in VGG19 is obviously higher than the others, 57.01% versus 25%. In the ResNet18, the vulnerable class is "Plane", with a successfully-targeted attack rate of 37.08% while the other classes are lower than 12%. These classes should be considered as vulnerabilities in the CNNs, and are pinpointed by generating test images using DIF. The issues can be mitigated through retraining the CNNs with the adversarial images generated by DIF, and the misclassification rate of the vulnerable classes declines at most from 61.67% to 6.37% after the retraining.**

***Index Terms*—AI Testing, Testing AI software, AI software quality validation**

I. Introduction

In recent years, Convolutional Neural Network (CNN) is becoming the core technique of Computer Vision (CV) related applications, such as autonomous self-driving vehicles and medical diagnosis. CNN performs well in recognizing various kinds of objects within images, i.e. to identify the object in an image into some specific item class, also known as image classification. However, CNNs are shown to be vulnerable to adversarial attack [7] [18]. The attacker may manipulate images from original images with adversarial perturbations which are negligible or harmless to human eyes. The adversarial perturbations can lead a CNN to misclassify input images, thereby rendering wrong decision making. For example, when the CNNs within a autonomous self-driving vehicle are under such an attack, the system may consider a pedestrian as nothing or a wall as a path. As a result, disasters can happen and lives are in danger due to these attacks.

Adversarial perturbations can be viewed as attacks upon bugs within a CNN, and developing a method for testing CNN to detect these bugs can be essential. The pixels within an image causing misclassification by a CNN after being perturbed are called critical pixels [28]. Manipulating critical pixels can be considered as generating inputs leading to faulty outputs of a CNN, i.e. producing test cases for the CNN. In this study, the most potential critical pixels within an input image to a CNN are located by analyzing the gradient of each pixel within the image. On the basis of the most potential critical pixels, an innovative attack method called Dual Iterative Fusion (DIF) for effective CNN testing is proposed.

DIF is derived from Basic Iterative Method (BIM) [6] and Least-Likely Class Method (LL) [6], where the two methods attack different Critical Pixels simultaneously with Momentum optimization [2]. DIF can designate the target class while generating adversarial perturbations, and can thus perform precise testing. From our experiments, DIF reveals that some classes are more vulnerable than the others within several classical CNN structures for image classification like VGG19 [4], ResNet18 [8], and GoogLeNet [29], etc., and can be considered as bugs of the structures.

The remaining parts of the paper are organized as follows. In Section II, the details of adversarial attacks are presented. In Section III, our methods including how to locate the most potential Critical Pixels within input images and the detail of DIF are introduced. In Section IV, the experiments for evaluating our methods are demonstrated. Finally, we discuss the interesting findings in this study and conclude our work in Section V and VI.

II. Background and Related Work

In this section, we report background knowledge and related work about adversarial attacks.

A. Adversarial Attacks

Given a neural network $N(x) : x \in \mathcal{X} \rightarrow y \in \mathcal{Y}$, which $\mathcal{X}$ is a set of input images and $\mathcal{Y}$ is a set of labels. $N(x)$ outputs a label y_{true} as the correct prediction for an input image x, i.e., y_{true} is the grounded-truth of x. The goal of adversarial attack is to elaborately construct images that can be misclassified by the neural network. To be specific, an attack aims to construct an adversarial example modified from x, denoted as x^{adv}, such that output label $N(x^{adv}) = y_{adv} \neq y_{true} = N(x)$, and x^{adv} looks similar to x by human eyes.

978-1-7281-7303-0/20/$31.00 ©2020 IEEE
DOI 10.1109/COMPSAC48688.2020.000-3

B. Attack Methods

The first work of finding adversarial examples in neural networks is published by Szegedy et al. [7]. Since then, various methods have been proposed for constructing adversarial images, such as FGSM [1], BIM [6], LL [6], DeepFool [3], C&W [20], JSMA [18] and universal adversarial perturbations [25]. Our proposed DIF is closer to BIM and LL.

Goodfellow et al. [1] proposed a reasonable hypothesis for the primary cause of adversarial perturbations: linear nature of neural networks, and proposed FGSM for fast generating adversarial examples. BIM is revised from FGSM that performs FGSM repeatedly In contrast, DeepFool calculates the closest direction to decision boundary in linear approximation for constructing adversarial perturbations. C&W provides a new method for targeted-attack by adding a norm limit in its loss function. Moosavi-Dezfooli et al. presents the first algorithm that can generate a universal perturbation for input images by designing a perturbation that pushes every input image out of the decision boundary. Various proposed methods in the above studies are briefly summarized as follows.

1) Fast Gradient Sign Method: Fast Gradient Sign Method (FGSM) is a one-step gradient-based approach. FGSM generates adversarial images by maximizing the loss function $J(X, \mathcal{Y}_{true})$, where J is often the cross-entropy loss. Formally, FGSM is described as

$$X^{adv} = X + \epsilon sign(\nabla_X J(X, \mathcal{Y}_{true}))$$

2) Basic Iterative Method: Basic Iterative Method (BIM) [6] is a simple extended version of Fast Gradient Sign Method(FGSM) [1]. BIM applies perturbations on the original image in multiple times. BIM is represented as

$$X^{adv}{}_0 = X$$

$$X^{adv}_{N+1} = Clip_{X_p,\epsilon}(X^{adv}_N + \alpha sign(\nabla_X J(X^{adv}_N, \mathcal{Y}_{true})))$$

Which the $Clip_{X,\epsilon}$ above is a function performs pixel-clipping to image X, and the value of the clipped pixels must be in L_∞ $\epsilon - neighborhood$.

BIM is good at keeping adversarial image away from the grounded truth, but the new label of adversarial image is not assignable.

3) Iterative Least-Likely Class Method: Iterative Least-Likely Class Method (LLC) [6] can generate adversarial images with the desired label we choose. Both FGSM and BIM is to maximize the loss function between input X and grounded-truth label $\mathcal{Y}_{true}$, while LL method is to minimize the loss function between input X and target label $\mathcal{Y}_{target}$. LLC can be written in

$$X^{adv}{}_0 = X$$

$$X^{adv}_{N+1} = Clip_{X_p,\epsilon}(X^{adv}_N - \alpha sign(\nabla_X J(X^{adv}_N, \mathcal{Y}_{LL})))$$

C. Testing Neural Networks

In field of traditional software testing, plenty methods of testing are already proposed such as fuzzing. Fuzzing is a testing technique that involves providing random input or unexpected input to a program [11]. Recently, several test coverage criteria and methods have been proposed to guide test case generation in the field of neural network. DeepXplore [19] is the first work proposed the concept of neuron coverage and the framework of whitebox testing for neural networks; TensorFuzz [13] provides the first coverage-guided fuzzing framework for neural networks; DLFuzz [22] provides a differential fuzzing framework for exposing incorrect behavior of neural networks; DeepGauge [21] proposed a lots of fine-grained test coverage criteria, then DeepHunter [15] uses these criteria and consider various mutation strategies to build a coverage-guided fuzzer for neural networks; Sun et al. [27] proposed another coverage criteria; DeepTest [26] and DeepRoad [23] provide the tool for testing autonomous self-driving cars based on neural networks; And DeepSearch [24] proposed a efficient blackbox fuzzing method for testing neural networks.

III. Methodology

A. Notations

Before the methodology is introduced, the notations used in this paper are first defined as following:

- X - The input image which is typically a 3-Dimensional tensor (width * height * color-depth). The values of the pixels in X are ranged between [0, 255] in this paper.
- $\mathcal{Y}_{true}$ - The grounded-truth label $\mathcal{Y}$ for the input image X.
- $\mathcal{Y}_{target}$ - The target label which attackers choose for adversarial attacking.
- J(X, $\mathcal{Y}$) - Loss function of the neural network model, given the input image X and label $\mathcal{Y}$ which can be either a true label or a target label.

B. Locating the Potential Critical Pixels

Narodytska et al. defined that the pixels within an image causing misclassification by a CNN after being perturbed as critical pixels [28]. Manipulating critical pixels to a certain value leads a CNN delivering faulty outputs, and reveals the defect within the CNN.

The input of a CNN-based neural network for object recognition is usually an image formed as an 1-dimensional array, , and each pixel in the image is considered as a data unit of the array. While gradient descent is applied for updating the parameters of a neural network, the gradients of the pixels of the input image can also be calculated. Since gradients indicate the directions reduce/enlarge the loss of the network, modification to a pixel with larger gradient may bring a greater impact to the final result of object recognition.

In this study, the pixels within an input image of a neural network which have the top-k largest gradient value are

Algorithm 1 Dual Iterative Fusion Method

Input:

DataSet ← Image dateset for generating adversarial images
Model ← The model for attacking
CP_Num$_{ITC}$ ← Number of the Critical Pixels to be attacked by ITC
CP_Num$_{BIM}$ ← Number of the Critical Pixels to be attacked by BIM
α_{ITC} ← The α value of ITC
α_{BIM} ← The α value of BIM
Iter_Num$_{ITC}$ ← The number of iteration of ITC
Iter_Num$_{BIM}$ ← The number of iteration of BIM
$\mathcal{Y}_{target}$ ← The target label of adversarial images.

Misclassification := 0
Successfully_Targeted := 0
for X, $\mathcal{Y}_{true}$ ∈ DataSet **do**
    X_grad = Compute_Gradient(X, $\mathcal{Y}_{target}$)
    CP_ITC = Get_Critical_Pixels(CP_Num$_{ITC}$, $\mathcal{Y}_{target}$, X, X_grad), which is Equation III-C2
    X'= ITC(CP_ITC, α_{ITC}, ϵ, Iter_Num$_{ITC}$, $\mathcal{Y}_{target}$, X)
    X_grad = Compute_Gradient(X, $\mathcal{Y}_{true}$)
    CP_BIM = Get_Critical_Pixels(CP_Num$_{BIM}$, $\mathcal{Y}_{true}$, X'), which is Equation III-C1
    X^{adv} = BIM(CP_BIM, α_{BIM}, ϵ, Iter_Num$_{BIM}$, $\mathcal{Y}_{true}$, X')
    Prediction = Model(X^{adv})
    if Prediction $\neq \mathcal{Y}_{true}$ **then**
        misclassification ++
    end if
    if Prediction $= \mathcal{Y}_{target}$ **then**
        Successfully_Targeted ++
    end if
end for
procedure GET_CRITICAL_PIXELS(CP_Number, $\mathcal{Y}$, X, X_grad)
    X_grad = Absolute_Value(X_grad)
    CP = Top_K(X_grad, CP_Number)
    return CP
end procedure

defined as the top-k potential critical pixels. The gradient of $J(X, \mathcal{Y}_{label})$ with respect to X can be defined as

$$G = \nabla_X J(X, \mathcal{Y}_{label})$$

The gradient value of a pixel of X is defined as

$$G_p = \nabla_{X_p} J(X_p, \mathcal{Y}_{label}) \quad (1)$$

which parameter p is the point of X. Then, the Critical Pixel is defined as

$$G_p = \arg\max_p \nabla_{X_p} J(X_p, \mathcal{Y}_{label}) \quad (2)$$

TABLE I
THE RELATION BETWEEN NUMBERS OF POTENTIAL CRITICAL PIXELS OF ITC AND BIM ATTACK. COMBINING ITC WITH BIM IMPROVES BOTH MISCLASSIFICATION RATE (MIS RATE) AND SUCCESSFULLY-TARGETED RATE (S-T RATE).

Critical Pixels	Mis Rate	S-T Rate
1 ITC + 4 BIM	78.58%	13.32%
4 ITC + 1 BIM	60.35%	17.90%
3 ITC + 2 BIM	69.33%	17.21%
2 ITC + 3 BIM	59.10%	12.55%
5 ITC	41.42%	16.82%
5 BIM	80.64%	12.70%

Thus, the top-k potential critical pixels in X are represented as

$$G_{p1,p2,...,p_K} = \arg\max_{p=\{p_1,p_2,...,p_K\}} \nabla_{X_p} J(X_p, \mathcal{Y}_{label}) \quad (3)$$

C. Dual Iterative Fusion Method with Potential Critical Pixels

In this study, Basic Iterative (BIM) [6] and Iterative Target Class (ITC) which is a revised version of Least Likely Class (LLC) [6] by enabling LLC to choose the target class for misclassification are integrated together as Dual Iterative Fusion (DIF) for CNN testing.

As Algorithm 1 shows, BIM and ITC are applied among the top-k potential critical pixels separately. The individual number of potential critical pixels being manipulated by BIM and ITC influences the performance of DIF. Table I demonstrate the performance of DIF with different allocation of potential critical pixels to BIM and ITC. The misclassification rate (Mis rate) shows the ratio of the perturbed images which are misclassified by the CNN under testing, and the successfully-targeted rate (S-T rate) shows the ratio of the perturbed images which are misclassified as the designated target class.

Overall, comparing to adopting BIM and ITC separately, DIF is improved in the following way:

(1) Momentum [2] is adopted to enhance the efficiency of both methods. With momentum, the less iteration for constructing an adversarial image is achieved.
(2) DIF generates fewer perturbations by attacking only the potential critical pixels when the other methods such as FGSM [1], BIM [6] and DeepFool [3] modify entire input images, and thus are more deceptive than the existent methods.
(3) . The perturbations crafted by ITC can't escape from the ground truth, and thus perform poorly in targeted-attack. DIF properly mix ITC and BIM to get the adversarial away from the ground truth, and achieves better targeted-attack than applying only ITC or BIM.

1) Basic Iterative Method with Potential Critical Pixels: To apply the original BIM [6] onto potential critical pixels, we revise BIM as following:

$$X_p^{adv}{}_0 = X_p, \quad v_p = 0, \quad Momentum = 0.9$$

$$v_p = (Momentum * v_p) + \alpha sign(\nabla_{X_p} J(X_p^{adv}{}_N, \mathcal{Y}_{true}))$$

TABLE II
COMPARISON OF THE MODELS UNDER EVALUATION

Model	Test Acc.	Mis Rate	S-T Rate
VGG19	93.66	64.56%	11.66%
ResNet18	95.40	39.18%	6.70%
EfficientNetB0	89.05	23.21%	3.84%
GoogleNet	94.37	39.67%	6.48%

$$X_p^{adv}{}_{N+1} = Clip_{X_p,\epsilon}(X_p^{adv}{}_N + v_p)$$

2) Iterative Target Class Method with Critical Pixels: Comparing to LLC [6], ITC is able to choose the target class freely, and the loss function replaced by $\mathcal{Y}_{target}$. Formally, LLC is revised as:

$$X_p^{adv}{}_0 = X_p, \quad v_p = 0, \quad Momentum = 0.9$$

$$v_p = (Momentum * v_p) + \alpha sign(\nabla_{X_p} J(X_p^{adv}{}_N, \mathcal{Y}_{target}))$$

$$X_p^{adv}{}_{N+1} = Clip_{X_p,\epsilon}(X_p^{adv}{}_N - v)$$

To make an adversarial image which is classified as $\mathcal{Y}_{target}$, this method minimizes the loss between adversarial image and $\mathcal{Y}_{target}$. Therefore, the intermediate adversarial image is closer to the target label in the iterative procedure.

The whole workflow of DIF is detailed in Algorithm 1.

IV. EVALUATION

To evaluate the effectiveness of DIF, three classical CNNs, VGG19 [4], ResNet18 [8], and GoogLeNet [29], and EfficientNetB0 [16], a new light-weighted model proposed by Google reaching outstanding performance with very low parameter size, are chosen for experiments.

In Sec. 4.1, the experiment setting is specified. The procedure and result of attack is presented in Sec. 4.2. Then we retrain the models with adversarial images and the result is discussed in Sec. IV-C.

All our experiments is developed by PyTorch 1.3.1, run on the Manjaro Linux with kernel 5.2 on a 6-core 2.8GHz Intel Core i5-8400 CPU, 16GB of RAM and a NVIDIA GTX 970 GPU.

A. Experiment Settings

1) Models and Data set: All the models are trained on CIFAR-10, which is a popular data set for image classification, for 120 epochs. The accuracy of these models is shown in Table II.

CIFAR-10 is composed of 60,000 images with 10 labels, 6,000 images for each label. In our experiments, 50,000 out of the 60,000 images within CIFAR-10, 5,000 for each label, are used as the training set, and the other 10,000 images, 1,000 for each label, are used as the testing set.

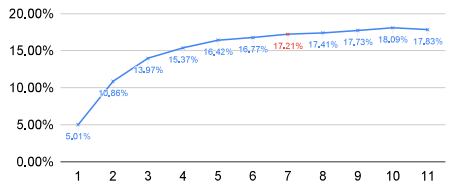

Fig. 1. How iteration time affects the successful rate of attacking.

2) Hyper Parameters: The parameters of attacking are listed below:

- The iteration is 7. Figure 1 shows how iteration time affects the successful rate of attacking. And iteration for 7 times is the balance point between computation time and performance.
- The number of critical points is 5, 3 for ITC, 2 for BIM. The reason of setting critical pixels as 5 is that some labels are more vulnerable than others. We can find the vulnerable labels through adjusting the intensity of attack. A vulnerable label remains high successfully-targeted rate even though the attack is set as low intensity.
- The α is 4.
- The ϵ is 10.
- The Momentum is 0.9.

B. CNN Testing through DIF Attack

Among the training set, 1,000 images are randomly drawn from each label for generating adversarial images for retraining, and the total of 10,000 images in testing set is used for measuring misclassification rate and successfully-targeted rate. The misclassification rate and successfully-targeted rate are gauged by testing set after retraining the model with adversarial images generated from training set.

The results are shown in Table II. DIF works better on VGG19 and ResNet18, and we also found that the models with high accuracy in testing set can also be more vulnerable. In other words, the accuracy in testing set is not related to the resistance to adversarial attack.

Through the attacking, as Table III shows, we observed that among all the models, some classes are easier to be targeted than the others. For example, after DIF attack, the S-T Rate of class "Cat" in VGG19 is 57.01% which are much higher than the S-T rate of any other classes. We can also observe that class "Plane" has relatively high S-T rate among all the models, and the classes like "Automobile", "Bird", and "Cat" has relatively high S-T rate in only some models. The classes which are easy to be targeted are marked in bold and italic in Table III.

We believe that the defects are brought by the biases buried within the training data set, and testing the CNN with DIF can reveal the biases before applying the trained model in practice. DIF reveals the defects of a CNN structure that some labels are not trained well with the structure like "Cat" in VGG19 and "Plane" in ResNet18, i.e. the model does not truly learn the features of these labels, and just finds a way

TABLE III
THE SUCCESSFULLY-TARGETED RATE OF EACH LABEL.

Model	Plane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
VGG19	*12.67%*	***24.89%***	*13.13%*	***57.01%***	0.70%	3.98%	0.36%	2.51%	0.34%	0.99%
ResNet18	***37.08%***	5.34%	*11.01%*	*8.96%*	0.74%	1.19%	0.99%	0.78%	0.37%	0.50%
EfficientNetB0	***10.38%***	3.84%	*8.83%*	4.95%	1.39%	2.55%	1.49%	2.65%	0.92%	1.44%
GoogLeNet	***20.79%***	*5.49%*	***19.86%***	*9.43%*	0.58%	*6.51%*	0.34%	0.65%	0.15%	1.03%

TABLE IV
THE SUCCESSFULLY-TARGETED RATE OF EACH LABEL AFTER RETRAINING THE MODEL.

Model	Plane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
VGG19	9.01%	6.49%	12.36%	7.49%	1.90%	6.11%	1.15%	2.31%	0.56%	1.75%
ResNet18	6.43%	1.56%	5.12%	2.31%	0.44%	2.26%	0.58%	0.97%	0.49%	0.62%
EfficientNetB0	3.22%	1.13%	3.56%	2.62%	0.91%	1.89%	0.91%	1.42%	0.67%	0.93%
GoogLeNet	2.65%	1.26%	5.90%	5.01%	5.82%	1.93%	0.36%	0.80%	0.65%	0.71%

TABLE V
THE RESULT OF FGSM ATTACK BEFORE AND AFTER RETRAINING WITH ADVERSARIAL IMAGES GENERATED BY DIF ATTACK

Model	Original	Retrained
VGG19	42.43%	39.33%
ResNet18	38.32%	33.54%
EfficientNetB0	68.19%	45.34%
GoogLeNet	77.01%	56.17%

to correctly classify these labels in the training and testing set. The classification can be problematic because the CNN structure can be more vulnerable than the others while be trained with the same data set.

C. Mitigating the Defects through Retraining

Retraining a CNN with adversarial images is one of the effective ways to resist adversarial attack [5] [12], i.e. to fix the vulnerability caused by the defect of the CNN structure. The seven chosen models are retrained with the original training set together with the adversarial images generated from the training set. After the retraining, we examine whether the defects are fixed by attacking the retrained model with the same parameters mentioned in Sec IV-C..

The comparisons of retrained models and original models are shown in Table IV. Overall, the retrained models show more resistance and robustness to adversarial attack in the same intensity. The S-T rates of the classes with hidden defects obviously drop while S-T rate of the other classes slightly rise. Retraining the models with adversarial images ease the biases within the original training data sets with some minor costs.

Table V shows that the misclassification rate caused by FGSM before and after retraining. These models are retrained with the adversarial images generated by DIF attack. The results show that retraining models with adversarial images generated by DIF attack also improves resistance and robustness to the other kinds of adversarial attacks.

V. DISCUSSION

A. Why some labels are easier to be targeted?

First, one of the possible reasons is the inner bias of the data set. Some examples in the data set are more complex than the others. Therefore, the CNN might not learn the real features of the label with the limited examples.

Another reason causing the phenomenon could be the preferences among various CNN architectures. Our results indicate that CNNs showing different preferences to the labels. For example, The Successfully Targeted rate of label "Cat" of VGG19 is unusually higher and so is the label "plane" in ResNet18. While comparing the impacts in adversarial attack made by DIF among various models with Missification rate, the same preferences remain but become implicit. The results remain the same after the experiments are re-executed with different random selection in training and testing set. Thus, we may conclude that applying DIF for CNN testing can really locate the vulnerable labels for a specific CNN structure.

B. Why are the manipulation of potential critical pixels conspicuous in some cases?

While manipulating the potential critical pixels with DIF, the hue of the pixels can be changed. Even if the perturbation is minuscule, the manipulation can still be conspicuous to human eyes that the attack can no longer be deceptive even only limited numbers of pixels are manipulated. This problem might be solved if we limit the search of potential critical pixels that only the pixels can be manipulated without significant change of the hue of the pixel are chosen. In this way, we may guarantee the deception of the adversarial examples generated by DIF.

VI. Concluding Remarks

In this paper, Dual Iterative Fusion (DIF), an innovative adversarial attack methodology which focuses on attacking the potential critical pixels of the input image is introduced for CNN testing. By inducing CNNs not only to misclassify the input images but also to recognize the input images as the designated classes with limited modification, e.g. 5 pixels within 32x32 images in this paper. DIF is efficient, deceptive, and can be helpful in locating the vulnerabilities of a CNN.

With DIF, we found that some labels are easier being targeted for misclassification than the others within various classical object recognition CNNs. This phenomenon can be considered as bugs within the CNNs, and can possibly be resulted from the hidden biases buried within the training data. These bugs makes the CNNs vulnerable since the attackers can manipulate the CNNs to recognize an object as some specific class they want. By retraining the networks with the adversarial images generated by DIF, the vulnerabilities are obviously diminished. Our results show that the retrained models become more robust that it resist not only attack from DIF but also some other classical adversarial attack methods like FGSM.

References

[1] Ian J. Goodfellow, Jonathon Shlens and Christian Szegedy, "Explaining and Har-nessing Adversarial Examples," arXiv: 1412.6572v3 [stat.ML].

[2] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu and Jianguo Li, "Boosting Adversarial Attacks with Momentum, " IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2018.

[3] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi and Pascal Frossard, "DeepFool: a simple and accurate method to fool deep neural networks," arXiv:1511.04599v3[cs.LG].

[4] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556v6 [cs.CV].

[5] Bo Li, Yevgeniy Vorobeychik and Xinyun Chen, "A General Retraining Framework for ScalableAdversarial Classification," arXiv:1604.02606v2 [cs.GT].

[6] Alexey Kurakin, Ian Goodfellow and Samy Bengio, "Adversarial examples in the physical world," Workshop track - The International Conference on Learning Representations (ICLR) 2017.

[7] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow and Rob Fergus, "Intriguing properties of neural networks," arXiv:1312.6199v4 [cs.CV].

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep Residual Learning for Image Recognition," arXiv:1512.03385v1 [cs.CV].

[9] Jiawei Su, Danilo Vasconcellos Vargas and Sakurai Kouichi, "One pixel attack for fooling deep neural networks," IEEE Transactions on Evolutionary Computation, Vol.23 , Issue.5 , pp. 828–841, 2019.

[10] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu and Kaiming He, "Aggregated Residual Transformations for Deep Neural Networks," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2017.

[11] Valentin J.M. Man'es, HyungSeok Han, Choongwoo Han, Sang Kil Cha, Manuel Egele,Edward J. Schwartz and Maverick Woo, "The Art, Science, and Engineering of Fuzzing: A Survey," arXiv:1812.00140v4 [cs.CR].

[12] Xinyun Chen, Bo Li and Yevgeniy Vorobeychik, "Evaluation of Defensive Methods for DNNs against Multiple Adversarial Evasion Models," The International Conference on Learning Representations (ICLR) 2017.

[13] Augustus Odena and Ian Goodfellow, "TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing," arXiv:1807.10875 [stat.ML].

[14] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi and Cho-Jui Hsieh, "ZOO: Zeroth Order Optimization based Black-box Attacks to Deep Neural Networks without Training Substitute Models," ACM Workshop on Artificial Intelligence and Security (AISEC) 2017.

[15] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Hongxu Chen, Minhui Xue, Bo Li, Yang Liu, Jianjun Zhao, Jianxiong Yin and Simon See, "DeepHunter: Hunting Deep Neural Network Defects via Coverage-Guided Fuzzing," arXiv:1809.01266v3 [cs.SE].

[16] Mingxing Tan and Quoc V. Le,"EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," International Conference on Machine Learning (ICML) 2019.

[17] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill and Rob Ashmore, "DeepConcolic: Testing and Debugging Deep Neural Networks," International Conference on Software Engineering (ICSE) 2019.

[18] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik and Ananthram Swami, "The Limitations of Deep Learning in Adversarial Settings," IEEE European Symposium on Security & Privacy (IEEE Euro S&P), IEEE 2016.

[19] Kexin Pei, Yinzhi Cao, Junfeng Yang and Suman Jana, "DeepXplore: Automated Whitebox Testing of Deep Learning Systems," Symposium on Operating Systems Principles (SOSP) 2017.

[20] Nicholas Carlini and David Wagner,"Towards Evaluating the Robustness of Neural Networks," arXiv:1608.04644v2 [cs.CR].

[21] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao and Yadong Wang, "DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems," IEEE/ACM International Conference on Automated Software Engineering (ASE) 2018.

[22] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen and Jiaguang Sun, "DLFuzz: Differential Fuzzing Testing of Deep Learning Systems," The ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) 2018.

[23] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu and Sarfraz Khurshid "DeepRoad: GAN-based Metamorphic Autonomous Driving System Testing," IEEE/ACM International Conference on Automated Software Engineering (ASE) 2018.

[24] Fuyuan Zhang, Sankalan Pal Chowdhury and Maria Christakis, "DeepSearch: Simple and Effective Blackbox Fuzzing of Deep Neural Networks," arXiv:1910.06296 [cs.LG].

[25] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi and Pascal Frossard, "Universal adversarial perturbations," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2017.

[26] Yuchi Tian, Kexin Pei, Suman Jana and Baishakhi Ray "DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars," International Conference on Software Engineering (ICSE) 2018.

[27] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill and Rob Ashmore, "Testing Deep Neural Networks," arXiv:1803.04792v4 [cs.LG].

[28] Nina Narodytska and Shiva Prasad Kasiviswanathan, "Simple black-box adversarial perturbations for deep networks," arxiv:1612.06299 2016 [cs.LG]

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich, "Going Deeper with Convolutions," arXiv:1409.4842 [cs.CV].

[30] Jie Hu, Li Shen, Samuel Albanie, Gang Sun and Enhua Wu, "Squeeze-and-Excitation Networks," IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2018.

[31] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan and Jiashi Feng, "Dual Path Networks," Neural Information Processing Systems 30 (NIPS 2017).