Side-Channel Gray-Box Attack for DNNs

Yun Xiang, Yongchao Xu, Yingjie Li, Wen Ma, Qi Xuan, *Member, IEEE*, and Yi Liu, *Member, IEEE*

***Abstract*—Deep neural networks are becoming increasingly popular. However, they are also vulnerable to adversarial attacks. The existing attack methods include white-box attack and black-box attack. The white-box attack assumes full model knowledge while the black-box one assumes none. In this brief, we propose a novel attack method between these two. Specifically, we have made the following contributions: (1) we propose the gray-box attack, which utilizes the side-channel attack to predict the model structure based on a pre-trained classifier and (2) we validate our method on real-world experiments. The experimental results show that our gray-box attack can significantly outperform the existing techniques.**

***Index Terms*—Deep neural network, side-channel attack, adversarial attack.**

I. Introduction

DEEP neural networks (DNNs) are widely used in many areas, such as image classification, object detection, and natural language processing [1]–[6] etc. DNNs can achieve unprecedented performance and even surpass human counterparts. However, the security problems beneath those technologies are becoming more serious. To validate and defend against those threats, we present a side-channel based adversarial attack in this brief.

DNNs can be vulnerable. By applying an imperceptible perturbation, DNNs can easily be misled and hence, misclassify. This causes great security concerns for modern artificial intelligence (AI). The existing adversarial attacks include white-box attack [7], [8] and black-box attack [9]–[11], respectively. White-box attack requires the complete knowledge of the target model, including parameter values, network structure, and training dataset, etc. Unlike white-box attack, black-box attack assumes that the target model is unknown. Naturally, white-box attack is more effective and easier to implement. However, in real-world applications, most AI devices are considered as black-box. In general, black-box attack is less effective but more feasible.

Side-channel attack (SCA) [12] is a widely used tool to derive internal knowledge via hardware. SCA uses side-channel information leakage, such as time consumption, power consumption, and electromagnetic radiation, etc. However, this method cannot reveal some of the critical parameters, such as neuron weights, loss functions, and training sets, etc. Thus, we take advantage of the partial structure knowledge and design an SCA-based adversarial attack strategy.

In general, this brief presents a gray-box attack based adversarial examples generation method, where adversarial examples refer to distorted samples which can undermine the performances of target model [13]. It is more practical than white-box and superior to the black-box model. First, we employ the SCA-based attack to derive the basic network structure of the target model. Then, the derived network structure is used to train the substitute model. Finally, we use the trained substitute parameters to create adversarial examples, which can mislead the target model.

The rest of this brief is organized as follows. Section II introduces the related works. Section III describes the SCA-based method, Jacobian-based dataset augmentation, and the fast gradient sign method (FGSM). Section IV discusses the details of our method. Section V presents the experimental results. Section VI concludes the work.

II. Related Work

The related works can be categorized into three parts, which are adversarial attacks, SCA, and DNN frameworks, respectively.

A. Adversarial Attacks

The security problem of DNNs is important [14]. Szegedy *et al.* [13] demonstrate the concept of adversarial perturbations for DNNs. By applying an imperceptible perturbation, The DNNs may misclassify images. Goodfellow *et al.* [15] propose the FGSM algorithm, which is based on the assumption that linear behavior in high-dimensional space causes neural networks susceptible to adversarial perturbation. Moosavi-Dezfooli *et al.* [8] propose the DeepFool algorithm to compute perturbations by assuming decision boundaries to be linear. Papernot *et al.* [7] propose a Jacobian-based saliency map attack algorithm. Unlike the existing methods, they generate adversarial samples by constructing a mapping from input perturbations to output variations. However, adversarial saliency maps require significant amount of computing time.

Manuscript received May 20, 2020; revised June 26, 2020; accepted July 14, 2020. Date of publication July 27, 2020; date of current version December 21, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61873241, Grant 61973273, and Grant 61572439, and in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR19F030001. This brief was recommended by Associate Editor L. A. Camunas-Mesa. *(Corresponding author: Yi Liu.)*

Yun Xiang, Wen Ma, and Qi Xuan are with the Institute of Cyberspace Security, Zhejiang University of Technology, Hangzhou 310023, China, and also with the College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China.

Yongchao Xu, Yingjie Li, and Yi Liu are with the Institute of Process Equipment and Control Engineering, Zhejiang University of Technology, Hangzhou 310023, China (e-mail: yliuzju@zjut.edu.cn).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2020.3012005

Many black-box attacks that require no internal knowledge are also proposed. Cisse *et al.* [10] propose Houdini, the first technique to compromise any gradient-based machine learning algorithms by generating adversarial examples tailored for the task loss. Su *et al.* [9] utilize differential evolution algorithms to generate adversarial examples. Only one pixel in the image is changed. Papernot *et al.* [11] perform real-world black-box attack. They employ Jacobian-based dataset augmentation algorithm to train the substitute model, which has decision boundaries similar to those of the target model.

B. Side-Channel Attacks

SCA is an effective tool in the cryptography field. It mainly includes timing attack [16], power-analysis attack [17], and electromagnetic attack [18], etc. Kasper *et al.* [19] use homologous offset to collect the energy information of radio frequency identification devices. Barenghi *et al.* [20] implement a low-voltage attack of advanced encryption standard and asymmetric encryption algorithms in Linux systems. Hojjati *et al.* [21] collect and reproduce the actions of the 3D printer and other devices located nearby using the accelerometer, magnetometer, and other devices in smart phones.

C. DNN Frameworks

There are several popular and widely used DNN frameworks. For example, Alexnet [22] is considered as the forerunner of deep learning; VGGNet [23] uses smaller convolution kernel; and ResNet [24] proposes skip-connection, etc. We validate our technique on mainstream frameworks.

III. Preliminaries

In this section we introduce the basic underlying theories for our technique.

A. Gray-Box Model

Leaking the structure information, even without specific parameter values, can be dangerous for DNNs. The power-based SCA technique can be used to derive the internal structure of DNNs. It contains more information than black-box attack, but less than white-box one. Specifically, the power consumption for the convolution layer $P_{conv}(.)$, pooling layer $P_{pl}(.)$, fully-connected layer $P_{fc}(.)$, and activation function layer $P_{ac}(.)$ can be defined using the following equations.

$$P_{conv}(C, L, W, S, N, F) = \frac{p_m LWNCF^2}{S^2} + \frac{p_a LWNCF^2}{S^2}, \quad (1)$$

$$P_{pl}(C, L, W, S, F) = \frac{p_c CLWF^2}{S^2}, \quad (2)$$

$$P_{ac}(C, L, W) = p_{ac}\alpha CLW, \quad (3)$$

$$P_{fc}(X, Y) = p_m XY + p_a XY, \quad (4)$$

where p_m, p_a, p_c, and p_{ac} are the power consumed by multiplication, addition, comparison, and activation function, respectively; $C \times L \times W$ is the input size; $F \times F$ is the filter kernel size; S is the filter stride; N is the number of filters; X is the number of neurons at the first fully-connected layer; and Y is the number of output neurons.

It is observed that different DNN models consume varying power due to their different structures. Therefore, the model structure can be identified based on power traces. This can be implemented using appropriate machine learning algorithms. Note that the power consumption varies for different platforms. However, in this brief we assume that the attacker can have access to the targeting hardware device, which is true for many applications. Therefore, the power profile can be created for each type of device.

B. Black-Box Attack

For black-box attack, we can train a substitute model with a decision boundary similar to that of the target model. Then we utilize the substitute model to create adversarial examples. In that way, the target model can be misled. To perform the black-box attack, a small number of representative images are collected first. Then we select the substitute model structure with the highest attack success rate. Finally, the target model is used to obtain the prediction label and train the substitute model.

Jacobian-based dataset augmentation method is typically used to expand the training set. It is implemented as follows.

$$S_{\rho+1} = \{x + \lambda \cdot sign(J_F[O(x)] : x \in S_\rho\} \cup S_\rho, \quad (5)$$

where x is the original image, S_ρ is the current training set, $S_{\rho+1}$ is the new training set, $J_F[.]$ is the jacobian matrix of the substitute model, $O(x)$ is the prediction label of the training example, $sign(.)$ is the sign function, and $\lambda \in (0,1]$ is a tuning parameter.

C. Fast Gradient Sign Method

Adversarial examples include the original image and the corresponding perturbations. It can be defined as $x+\eta$, where η is the perturbation. The FGSM model [15] is a widely used adversarial examples generating method. It derives an optimal max-norm constrained perturbation using the following equation.

$$\eta = \varepsilon \cdot sign(\nabla_x J(\theta, x, y)), \quad (6)$$

where θ is the parameter of model, y is the target label of the adversarial example, $J(\theta, x, y)$ is the cost function, and $\varepsilon \in (0,1]$ is a scalar.

IV. Gray-Box Based Adversarial Examples Generation

In this section, we introduce our side-channel based gray-box attack method. First, we introduce the underlying assumption of the gray-box attack, which is shown in Figure 1. Then we describe the proposed algorithm in detail.

A. Network Structure Derivation

To perform the white-box attack, the whole internal AI information, including network structure, model parameters, training methods, and training data etc., is required. However, in real-world applications, AI devices are mostly black-box devices. Therefore, we rely upon the SCA to derive the

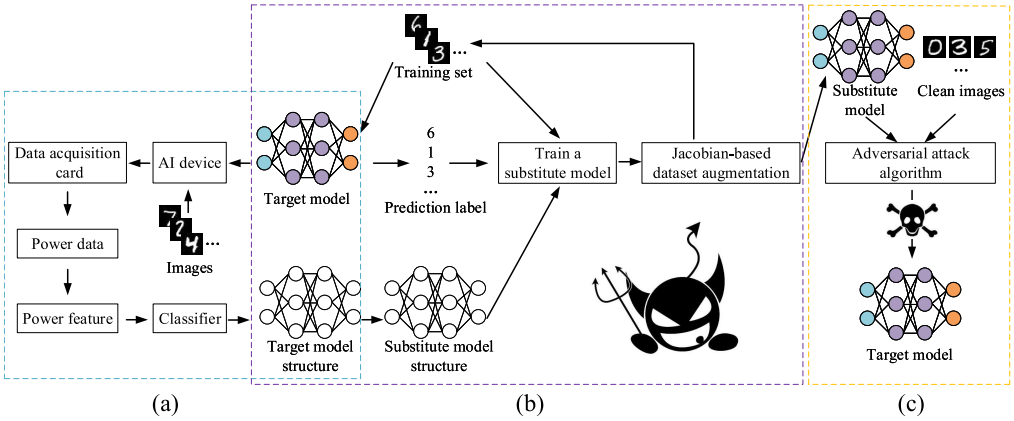

Fig. 1. The flowchart of gray-box attack algorithm. (a) Obtain the network structure of the target model. (b) Train a substitute model. (c) Create adversarial examples.

Algorithm 1 Estimation of the Target Model Network Structure

Input:

voltage $U = [u_1, u_2, ..., u_n]$, current $I = [i_1, i_2, ..., i_n]$, the trained classifier D.

Output:

the label y of network structure.

1: Calculate the power data: $P = UI = [p_1, p_2, ..., p_n]$.

2: Calculate the power features:
$p_{mean} = \frac{1}{n}\sum_{i=1}^{n} p_i$,
$p_{median} = sort(P)[n/2]$,
$p_{std} = \sqrt{\frac{\sum_{i=1}^{n}(p_i - p_{mean})^2}{n}}$.

3: Feed the power features into the trained classifier D to obtain a prediction label y.

4: **return** y.

Algorithm 2 Substitute Model Training

Input:

the target model O, initial training set S_0, substitute model structure F, parameter λ, the number of training substitute model epochs e.

Output:

the parameters of substitute model θ_F.

for $\rho \in 0, 1, 2, ..., e-1$ **do**

Use the target model O to get the label $O(x)$, $x \in S_\rho$;

Train the substitute model on the training set S_ρ and get the substitute model parameter θ_F;

Obtain a new training set $S_{\rho+1}$: $S_{\rho+1} = \{x + \lambda \cdot sign(J_F[O(x)]) : x \in S_\rho\} \cup S_\rho$.

end for

return θ_F

network structures and thus, convert the black-box attack to the gray-box one.

In practice, AI developers usually design their models based on the existing, pre-trained, and publicly available architectures. Then the models are fine-tuned based on their own training sets. Therefore, by carefully analyzing the device power features, it is possible to reveal the actual model architecture. Thus, we use the side-channel based method to derive the architecture of the models. Note that the parameters and weights information are still assumed unknown.

Without loss of generality, we run the public models on the Raspberry Pi platform and collect the power traces. Then we calculate the features such as mean, median, and standard deviation, etc. Thus, we derive a power dataset of various models. This is later labeled and used as the training set. During the testing phase, the target model is also run on the Raspberry Pi platform. Then we use the trained classifier to predict the detailed architecture of the running model. In that way, we derive the specific network architecture for the gray-box attack. The specific process is shown in Algorithm 1. For each model, there are one hundred input power traces. n equals 24. Note that n is not the number of inputs. It means that we create one input sample for every 24 measurements.

B. Substitute Model Training

Using SCA, we can derive the network structure of the target model. However, to perform the precise white-box attack, we also need the parameter values, which cannot be derived through SCA. Therefore, to generate the adversarial examples, a substitute model is required. To train the substitute model, we assume the same network structure as the target model. Thus, the performance of the substitute model can be significantly improved compared with the black-box attack.

During the process of generating the substitute model, the specific parameters, i.e., weights and biases, are derived. First we apply the Jacobian-based dataset augmentation algorithm [11]. It can reduce the number of model access by orders of magnitude. Then we train the substitute model with the help of the target device. The pseudo code of the training process is shown in Algorithm 2. Based on the application of the target model, we first create the training set by randomly selecting 150 samples from the dataset. Note that our system is based on the black-box attack model, which assumes no prior knowledge about the training set of the target model. Therefore, this training set is not required to be identical to that of the target model. Then the target device is used to generate labels for the training set. The training set samples

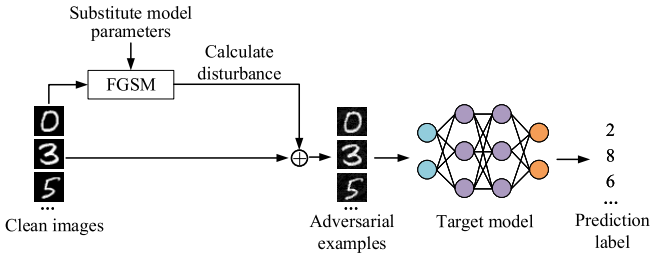

Fig. 2. The flowchart of creating adversarial examples using FGSM.

are fed into the target device in sequence and the corresponding labels can be derived. Thus, by combining the training set and the prediction labels, we can train a substitute model with the identical network architecture of the target model. Based on equation (5), we can derive a new and expanded dataset, which is used to train the substitute model.

C. Adversarial Examples Creating

After the substitute model is derived, we proceed to generate the adversarial examples. Without loss of generality, we use the FGSM model [15]. The flowchart of generating adversarial examples is shown in Figure 2. The added perturbation can be calculated as follows.

$$\eta_g = \varepsilon \cdot sign(\nabla_x J(\theta_g, x, y)), \tag{7}$$

where ε is the input parameter which controls the magnitude of disturbance, η_g is the perturbation of substitute model, θ_g is the parameter of the substitute model, and $J(\theta_g, x, y)$ is the cost function. Thus, by generating the adversarial examples, we complete our gray-box attack.

In general, our gray-box attack uses the parameters of the substitute model to create adversarial examples and calculate the perturbation based on equation (7). Since the gray-box attack utilizes more internal information than the black-box one, the decision boundary of the trained gray-box substitute model is closer to the target model. Therefore, it is more effective than the black-box attack and meanwhile more realistic than the white-box attack.

V. Experimental Results

In this section we present the experimental results which validate our technique.

A. Experimental Setup

We design our experiments based on the MNIST_fashion dataset. During the experiment, we utilize five popular networks, which are InceptionV3, Alexnet, MobilenetV1, MobilenetV2, and Resnet, respectively. Those networks are the attack targets. They are pre-optimized and the classification accuracies are all above 95%. The details of these models are shown in Table 1. The dataset size is 19,050; the learning algorithm is Adam; the number of epochs is 50; the learning rate is 0.001; and the batch size is 50. We evaluate the performance on a machine with an NVIDIA RTX 2060 GPU.

TABLE I
Network Framework Structure

InceptionV3	Alexnet	MobilnetV1	MobilenetV2	Resnet
C(3,1,32)	C(11,1,64)	C(3,1,64)	C(3,2,32)	C(3,1,16)
Inecp(16)	MP(3,2)	DWC(3,1,64)	Bottleneck(1,1,16)	Res(3,1,16)×8
Inecp(16)	C(5,1,192)	C(1,1,64)	Bottleneck(2,6,16)	Res(3,2,16)
MP(2,2)	MP(3,2)	DWC(3,2,64)	Bottleneck(1,6,24)	Res(3,1,32)×8
Inecp(16)	C(3,1,384)	C(1,1,128)	Bottleneck(2,6,32)	Res(3,2,32)
Inecp(16)	C(3,1,256)	DWC(3,1,128)	Bottleneck(1,6,32)	Res(3,1,64)×8
Inecp(16)	C(3,1,256)	C(1,1,128)	Bottleneck(1,6,32)	Res(3,2,64)
MP(2,2)	MP(3,2)	DWC(3,2,128)	AP(2,2)	Softmax(10)
Softmax(10)	FC(1024)	C(1,1,256)	Softmax(10)	
	FC(1024)	FC(512)		
	Softmax(10)	FC(512)		
		Softmax(10)		

C() - convolution(kernel size, stride, neurons)
MP() - max pooling(kernel size, stride)
AP() - average pooling()
FC() - fully-connected(neurons)
Res() - Residual(kernel size, stride, neurons)
DWC() - depth-wise convolution(kernel size, stride, neurons)
Bottleneck() - (stride, Expansion ratio, neurons)
Incep(x) - concatenate C(1,1,x), C(3,1,x), C(5,1,x) and MP(2,2)

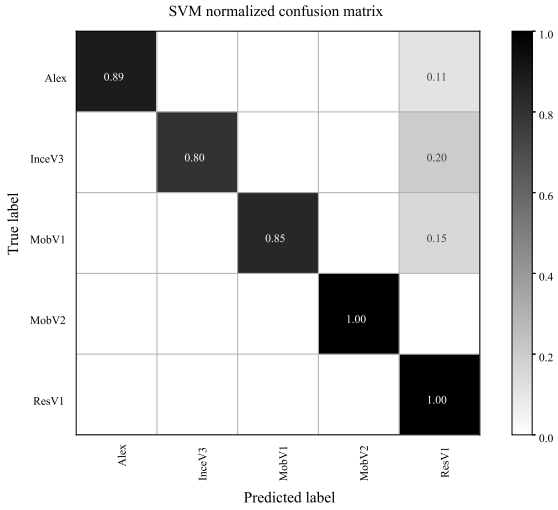

Fig. 3. Classification results of five networks by SVM.

B. Side-Channel Based Attack

To reveal the internal structure of the networks, we set up an experiment platform including a Raspberry Pi device and a data acquisition card. The five testing networks are run on the platform separately. Same group of pictures are fed into the networks. The average, median, and variance of the power consumption are calculated. Then we use the support vector machine (SVM) method with the radial basis function kernel to classify the specific networks. As shown in Figure 3, the average classification accuracy is 90.8%. The variations of power traces between different network architectures are significant. We have also tried other classification methods, such as random forest, k-nearest neighbors, and decision tree. The classification accuracies are all above 90%. Note that we are describing a general method that can be easily modified to work across many platforms. For example, we have performed similar experiments on Jetson Nano and RK3399 pro platforms. The classification accuracies reach 94.7% and 91.8%, respectively. There are many more architectures in the real-world applications. However, the main purpose of

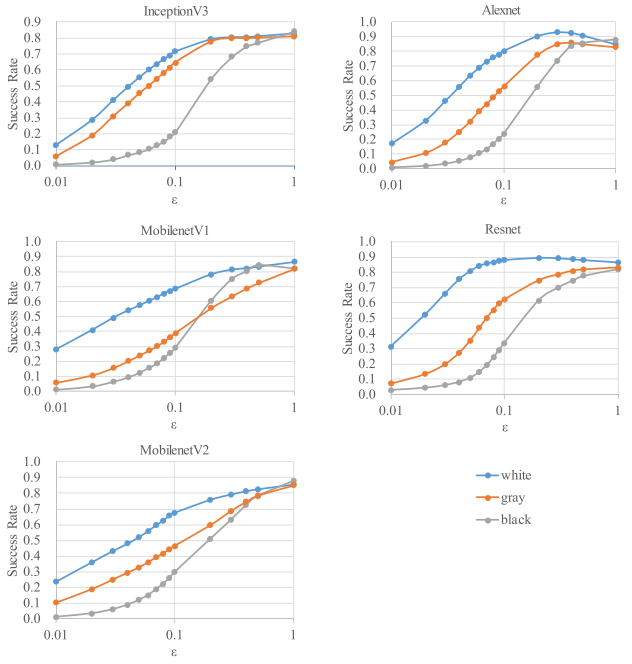

Fig. 4. The attack success rate with varying attack intensities.

this brief is to validate that the combination of side-channel attacks and adversarial attacks is feasible and can have significant impacts on the classification performance. Moreover, it is easy to expand our technique to other DNNs. Therefore, we consider the current experimenting architectures sufficient.

C. Attack Comparison

The gray-box attack uses FGSM to generate the adversarial examples. The input is the MNIST_fashion testing set and the output is the adversarial examples. The image distortion is almost imperceptible when we set the perturbation coefficient to be 0.1. To evaluate the method, we implement the corresponding white-box and black-box attacks. The white-box attack uses FGSM with identical network structures and parameters, while the black-box attack assumes a substitute network containing two convolutional layers and three fully-connected layers. Figure 4 shows the experimental results, where ε indicates attack strength and y-coordinate indicates the attack success rate, i.e., the probability of sample misclassification. In general, the gray-box attack significantly outperforms the black-box ones.

Note that exhaustive search is generally unfeasible. In real-world applications, there can be more possible architectures. For each candidate architecture, the algorithm requires to run the whole process of adversarial and side-channel attacks, which can be quite time and resource consuming. Moreover, certain more complicated attacks, e.g., the memory and bus communication attacks, rely upon the exact derivation of the architectures.

VI. Conclusion

This brief addresses the topic of attack methods for DNNs. Especially, based on the SCA strategy, we propose a simple and efficient gray-box attack method. The proposed method is more practical than white-box attack and more effective than black-box attack. For the future work, we shall consider more diverse DNN architectures.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] Y. Liu, C. Yang, Z. Gao, and Y. Yao, "Ensemble deep kernel learning with application to quality prediction in industrial polymerization processes," *Chemometr. Intell. Lab. Syst.*, vol. 174, pp. 15–21, Mar. 2018.
[3] Y. Liu, K. Liu, J. Yang, and Y. Yao, "Spatial-neighborhood manifold learning for nondestructive testing of defects in polymer composites," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4639–4649, Jul. 2020.
[4] Q. Xuan, Z. Chen, Y. Liu, H. Huang, G. Bao, and D. Zhang, "Multiview generative adversarial network and its application in pearl classification," *IEEE Trans. Ind. Electron.*, vol. 66, no. 10, pp. 8244–8252, Oct. 2019.
[5] K. Liu, Y. Li, J. Yang, Y. Liu, and Y. Yao, "Generative principal component thermography for enhanced defect detection and analysis," *IEEE Trans. Instrum. Meas.*, early access, May 6, 2020, doi: 10.1109/TIM.2020.2992873.
[6] M. Badar, M. Haris, and A. Fatima, "Application of deep learning for retinal image analysis: A review," *Comput. Sci. Rev.*, vol. 35, Feb. 2020, Art. no. 100203.
[7] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Security Privacy (EuroS&P)*, 2016, pp. 372–387.
[8] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2574–2582.
[9] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.
[10] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models," 2017. [Online]. Available: arXiv:1707.05373
[11] N. Papernot, P. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Security*, New York, NY, USA, Apr. 2017, pp. 506–519.
[12] Y. Zhou and D. Feng, "Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing," in *Proc. IACR Cryptol. ePrint Archive*, 2005, p. 388.
[13] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013. [Online]. Available: arXiv:1312.6199
[14] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.
[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014. [Online]. Available: arXiv:1412.6572
[16] P. C. Kocher, "Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems," in *Proc. Annu. Int. Cryptol. Conf.*, 1996, pp. 104–113.
[17] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, "Power-analysis attack on an ASIC AES implementation," in *Proc. Int. Conf. Inf. Technol. Coding Comput. (ITCC)*, vol. 2, 2004, pp. 546–552.
[18] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2001, pp. 251–261.
[19] T. Kasper, D. Oswald, and C. Paar, "Em side-channel attacks on commercial contactless smartcards using low-cost equipment," in *Proc. Int. Workshop Inf. Security Appl.*, Aug. 2009, pp. 79–93.
[20] A. Barenghi, G. M. Bertoni, L. Breveglieri, and G. Pelosi, "A fault induction technique based on voltage underfeeding with application to attacks against AES and RSA," *J. Syst. Softw.*, vol. 86, no. 7, pp. 1864–1878, 2013.
[21] A. Hojjati *et al.*, "Leave your phone at the door: Side channels that reveal factory floor secrets," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 883–894.
[22] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1097–1105.
[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: arXiv:1409.1556
[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.