INPUT TRANSFORMATION IN DEFENSE AGAINST ADVERSARIAL ATTACKS

HONGBIN CHEN and YONG MAO
GROUP 16

1. INTRODUCTION

We introduce a new defense way to attenuate the effect of some attack ways to a neural network, which is vulnerable to FGSM or some other adversarial attack methods.

The most core part of the way that we are proposing is to make a special transform based on a key stream to the input and to interfere with the targeted gradient trained by the neural network and attacked by FGSM. Here were taking white box attacking as example, but it also has an excellent effect to defend a black box attack. And the core part of the transform is related to some fields in cryptography. We borrowed some ideas from cryptography, i.e. a key stream that only defenders know, and we can ensure that different key streams will result in a different input to the network, so we only need to keep the key streams in secret and then the network will be in a relatively secure state.

An advantage of our method to protect the neural network is that we expand the advantage of defenders against attackers in the aspect of the amount of the information. As we all know an attacker should get more information to implement a more threatening and effective attack, once we increase the amount of the information to defenders, which is kept in secret, and then the relative information that the attackers hold is much smaller than before.

Another advantage of the method we propose is safe. The transform we apply to the input is highly based on the secret key so that its highly data independent, which will create a hard-to-learn state for attackers from the input data. And non-differentiability of the security transformation makes the proposed approach end-to-end non-differentiable and provides the additional difficulties for the attacker.

2. BACKGROUND AND RELATED WORK

2.1 General Ways to Generate Adversarial Samples

Many efforts have explored generation of adversarial samples. The FGSM method [1] identifies directions in the input space that reduce the most the probability of correct classification as adversarial directions. Given a target model h, an input image x, its label y, the loss function $L(h(x), y^{true})$, FGSM creates adversarial samples by adding perturbation to input images.

In addition to FGSM adversarial directions, random adversarial directions can also be identified, for example with random Gaussian noise added to pixels [2]. Sabour et al. [3] generate adversarial samples by manipulating the deep representation of images to make them look like other specific images to neural networks. Ensemble Adversarial Training [4] augment the training data with perturbations from other models. Ensemble Adversarial Training decouples the generation of adversarial samples and the model being trained to counter adversarial attacks because adversarial perturbations can be transferred from one NN to another due to the transferability of adversarial attacks. Other efforts use GANs [5, 6] to generate adversarial training examples. Xiao et al. [7] generate adversarial examples with adversarial networks, where GANs generate the perturbations. Kos et al. [8] utilize variational autoencoder (VAE) and VAE-GAN for generation of adversarial training examples. Baluja et al. [9] uses two approaches for generation of adversarial samples: perturbation in residual blocks in residual networks; adversarial autoencoding using regularizers.

In addition, more researches on black-box have been done these years. Therere roughly three mainstream types. Score-based attack methods rely on the predicted scores (e.g. class probabilities or logits) to estimate the underlying gradients. Typical methods in this category include black-box variants of JSMA [18] and of the CW attack [19, 20], as well as generator networks that predict adversarial [21]. Decision-based attack was proposed by [22]. It starts from a large adversarial perturbation and then iteratively reduces the perturbation while ensuring that the adversarial example stays in the adversarial region. Transfer-based attack aims to train a substitute network using the information queried from the underlying target model [23]. This method uses inputs synthetically crafted by the local substitute model and labeled by the target oracle. It has been shown that when the substitute model was well-trained, its adversarial examples can also mislead the target model. The previous work of transfer-based attack [23] used Jacobian based data augmentation, so the number of query doubles with the increase of iterations. Although it used reservoir sampling method which is one of the random select methods to reduce the queries, the exponential growth in the first few iterations has led to a significant increase in the number of queries.

2.2 General Ways to Defend Against Adversarial Attacks

Current defense approaches against adversarial attacks can largely e classified into two types. The first type are methods that make the model architecture more robust. The second type are methods that change the adversarial samples to make them less adversarial.

The most common method of the first type is adversarial training by Kurakin et al. [10], where the dataset is augmented with adversarial samples, often of the same kind as the attack, and the classifier is retrained. Adversarial training augments the dataset by filling out low probability gaps with additional data points and then retraining the classifier to find a better boundary. FGSM [1] method uses adversarial training. Virtual Adversarial Training (VAT) by Miyato et al. [11] identifies directions that can deviate the inferred output distribution the most and smoothes the output distribution in these anisotropic directions. Defensive distillation [12] by Papernot et al. extracts knowledge for improving a networks own resilience to adversarial attacks. It is based on the distillation training procedure [13], which trains a NN using knowledge from a different NN. Defensive distillation adapts distillation to use the knowledge to improve its own resilience, instead of transferring it to another NN.

Second type methods identify adversarial samples and move them towards the data distribution or the natural data manifold. Defense-GAN by Samangouei et al. [14] uses a GAN with a gen-

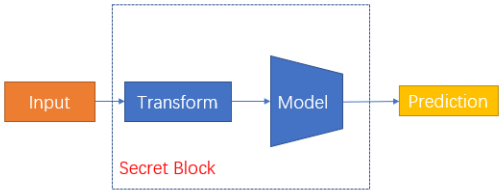


Fig. 1. The scenario of gray-box attacks: The attackers have no information about input transformation and limited knowledge about model body, but defenders have information advantage over attackers on input transformations.

erator to project adversarial points onto the manifold of natural images. Defense-GAN uses a classifier to which GAN input or GAN output or both GAN input and GAN output data points are used. Magnet by Meng et al. [15] also moves adversarial samples towards the manifold of natural images by utilizing an autoencoder or a collection of autoencoders. Jha et al. [16] focus on identification of adversarial samples. PixelDefend by Song et al. [17] proposes generative models to move adversarial images towards the distribution seen in the data. PixelDefend identifies adversarial samples with statistical methods (p-value) and finds more probable samples by generating similar images and looking for highest probability images within a distance from a given image.

3. MOTIVATION

Recent works to defend adversarial attacks rarely focus on the information advantage of the view of defenders, which is a great treasure to use. So we are trying in some aspects to make a special transform to the input which has a tight relationship with the advantages held by defenders. In addition, we are trying to expand this superiority by generating a secret key stream, a defense strategy using a cryptographic formulation. we consider a party consisting of: encoder E(.) and classifier (decoder) D(.) that share a common secret, i.e., secret key that is used in one of the blocks. This party is a defender that plays against the attacker, who does not know the secret key stream. The key stream will be generated by some certain ways such as time seed random or some others. In other papers which also focus on input transform, algorithms in cryptography is almost not mentioned, and an encryption is making a defense more secure and reliable. Therefore, we have a high expectation on the way we find out to defend against adversarial attacks.

4. PROBLEM FORMULATION

We study defenses against gray-box adversarial attacks and black-box adversarial attacks. More details are shown in figure 1, we define the secret block of a classic neural network model, which includes input transformation and model body. In this scenario, attackers know nothing about input transformation box, but the model body may be exposed to attackers, thus they can launch gray-box attacks with some white-box attack algorithm, such FGSM. Besides, attackers also may launch totally black-box attacks with a small set of train data.

Our goal is to defend adversarial attacks in that scenario by applying input transformation on origin model. That is for each input x, it should be transformed to $x^{'}$ by $x^{'} = f(x)$, $f(.)$ is the transformation function, then the model is trained and used with transformed inputs $\{x^{'}, \ldots\}$. This input transformation should strengthen the model's robustness and defenders can have information advantages over attackers relying on the input transformation.

5. PROPOSED METHODS

Our methods in defense against adversarial attacks is input transformation. To find a proper transformation is the most significant work, the transformation should have the next principles:

—Information advantage of the defender over attacker.
—Data-independence: the transformation itself is independent of input data.
—Non-linear: change the distribution of origin inputs.
—Maintain most information of input data.

5.1 An Overview of Generating Adversarial Examples

Before introducing the proposed adversarial defense method, we give an overview of generating adversarial examples. Let X_n denote the n-th image in a dataset containing N images, and let y_n^{label} denote the corresponding ground-truth label. We use to denote the network parameters, and $L(X_n; y_n^{label}; \theta)$ to denote the loss. For the adversarial example generation, the goal is to maximize the loss $L(X_n + r_n; y_n^{label}; \theta)$ for each image X_n, under the constraint that the generated adversarial example $X_n^{adv} = X_n + r_n$ should look visually similar to the original image X_n, i.e., $||r_n|| \leq \epsilon$, and the corresponding predicted label $y_n^{adv} \neq y_n^{label}$.

In our experiment, we use two methods to generate adversarial examples. One is FGSM [1] and the other is black-box method [23]. In order for generality and reliability of results, we use the cleverhans library to generate adversarial examples.

5.2 An Overview of Proposed Method

Our goal is to find a kind of input transformation that makes a model more robust and gives defenders information advantages over attackers, so that the models have higher accuracy on adversarial examples. We mainly work on images classifiers whose input is RGB channels or gray scale, the values of input are all integers between 0 - 255. Based on that, we propose a input transformation method that changes the distribution of input data non-linearly to defense against adversarial examples. The rough idea is: defender holds a **key** and he generates a key stream using the **key**, the key stream has same dimension as an input data; then each input data does the bitwise XOR with the key stream to generate corresponding transformed input data. We just apply this input transformation on models, there is no re-training or fine-tuning needed.

5.3 Key Stream Generator

Key stream generator, which is a concept in cryptography, is used to generate a key stream in one-time-pad methods. A key stream generator can generate a highly non-linear key stream using a **key**.

In our method, key stream generator is used to generate a key stream more easily. In our method, a key stream actually can be considered as a key of input transformation. However, it is difficult to find a so long highly non-linear key stream directly, so we propose to use key stream generator and then we can generate a key stream using just a number - **key**.

The implementations of key stream generator are various, for it is a successful field of cryptography, there are many theories to design a well-performed key stream generator. In our experiment, we use a naive idea to implement a simple key stream generator. We use **key**

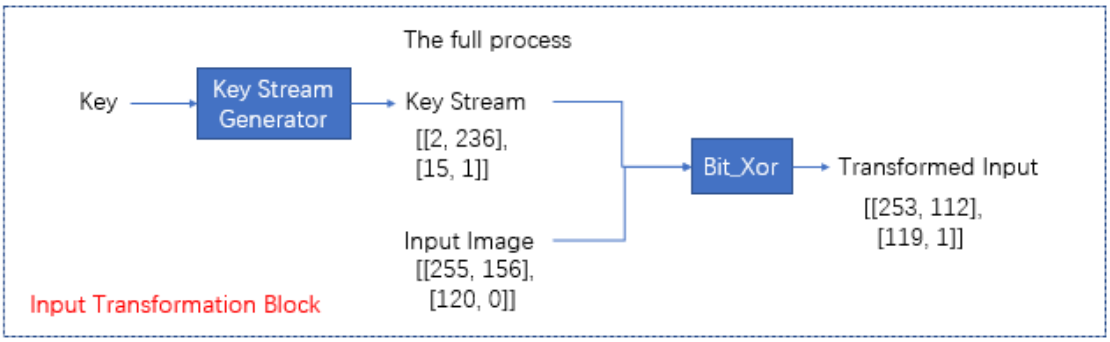


Fig. 2. Illustration of input transformation process: A number **key** is passed to key stream generator and then the key stream is generated. The dimension of key stream and input image are both 2×2, and their elements are all between 0 - 255. Each input image does bitwise XOR operation with key stream to generate corresponding transformed input.

as seed of random, that is after we set the random seed of numpy as **key**, successively calling *randomInt* function to generate a proper key stream.

5.4 Input Transformation Process

The process of proposed input transformation is as shown in figure 2, first, we pass the **key** to key stream generator and get a corresponding key stream K, K has same dimension as input X; then for each input X, we get transformed input $X^{'}$ by doing bitwise XOR between K and X, that is $X^{'} = bit_{x}or(X, K)$. Finally the transformed input $X^{'}$ will be fed into model body as input data.

It should be paid more attention that though our motivation of the transformation is from one-time-pad of cryptography, the input transformation is indeed different from encoding and decoding. We just transform input as masking, so the process only has encoding part but no decoding part and decoding is not necessary. A **key** generates a very different key stream, which stands for a totally different transformation. The attacker is hard to reconstruct the transformation unless he knows both **key** and the implementation of key stream generator which is impossible. Besides, the input transformation consumes little computation resources for it mostly does xor operations.

6. EXPERIMENTS

6.1 Experiment Setup

Dataset: MNIST with 60000 training data and 10000 testing data.
Basic Model: Three-layer CNN classifier.
Error Reduction: Run three times independently for each experiment and calculate their average accuracy as the final result.

6.2 FGSM Attack Scenario

In FGSM attack scenario, target model and defense model shown in figure 3 have same model structure, but the input of defense model needs to be transformed. We generate adversarial examples from target model and attack defense model. We test the accuracy of adversarial examples on defense model.

The result of FGSM attack scenario is shown in figure 4, the orange line is the result of our method while the blue line is the original model without any defense. We can see our method has an obvious defense effect from figure 4. When **eps** is 0, which means no adversarial attacks, we see that our method has a little inferiority (about 1%) compared to original model, maybe its because therere too many zeros of pixels in the images of MNIST dataset and our input transformation changes the zero-background into non-zero-background, resulting in little reduction of accuracy on clean images. However, when **eps** grows larger, our method has an obvious improvement on accuracy of adversarial examples. When **eps** is so large, the accuracy on defense model is also very low, but in this case, the perturbation on images is too large that it has no great reference value, which means the attacks are unpractical.

6.3 Black-box Attack Scenario

Under black-box attack scenario, the attacker has limited access to labeled data and he trains a substitute model by augmenting dataset, as shown in figure 3. Adversarial examples are generated from substitute model using FGSM and used to attack defense model. We test the accuracy of adversarial examples on defense model.

The result of black-box attack scenario is shown in figure 5, the orange line is the result of our method while the blue line is the original model without any defense. We can find that black-box attack is quite weaker than FGSM attack. When **eps** is small, attacks have little effect on model, no matter whether the model has defense methods. Our method has a little Inferiority compared to original model when **eps** is small, the reason may be just as stated in Section 6.2. When **eps** grows larger, we can see an improvement of about 15% - 25% on accuracy. As a result, our method also has potential to defend against black-box attacks. We think the interpretation may be that input transformation changes the distribution of input data largely, so the substitute model trained by black-box attack methods with origin data will be very different from defense model, thus the adversarial examples generated from substitute model seems weaker to attack defense model.

7. CONCLUSION

In this project, we propose a new kind of input transformation in defense against adversarial attacks. We think our contribution is trying to solve the security problem in neural network by referring to traditional security theory, such as cryptography. We try to apply the encoding method of one-time-pad in cryptography on input transformation. After experiments, we find this kind of input transformation does have defense effect against some white-box attacks and black-box attacks. What's more, in our proposed method, a **key** stands for a totally input transformation, these various transforma-

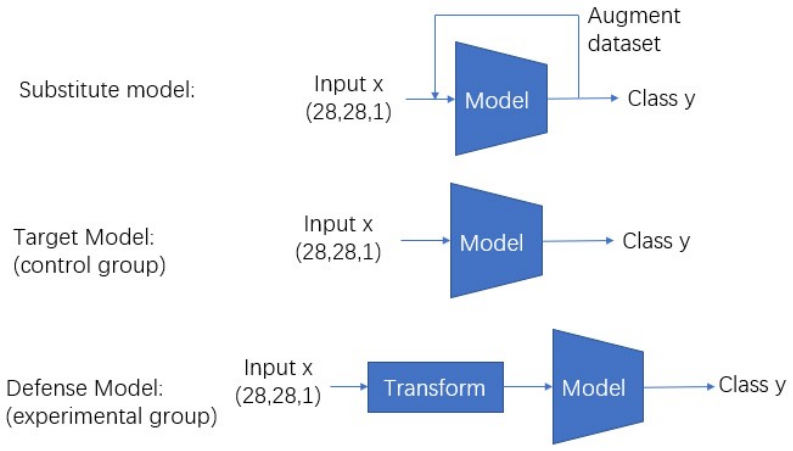
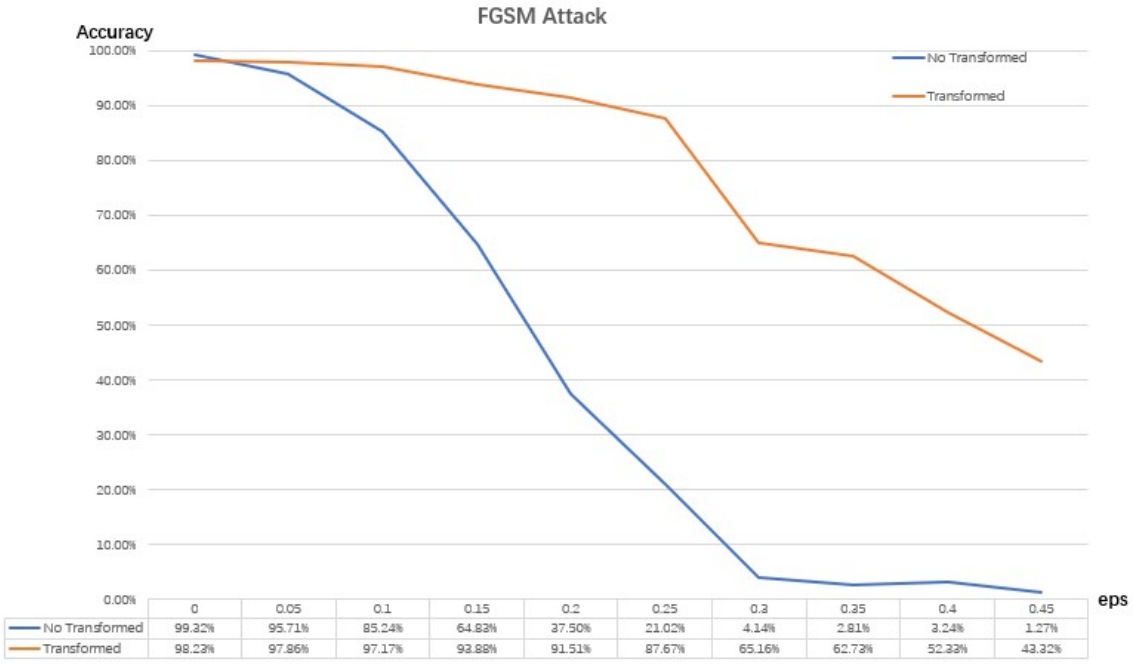


Fig. 3. Block diagram detailing the experiment setups in Section 6.2 and Section 6.3. The model structures in target model and defense model are same. In Section 6.2, we generate adversarial examples from target model and test the accuracy of adversarial examples on defense model. In Section 6.3, we construct a substitute model with black-box attack method, generate adversarial examples from substitute model using FGSM and test the accuracy of adversarial examples on defense model.



	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45
No Transformed	99.32%	95.71%	85.24%	64.83%	37.50%	21.02%	4.14%	2.81%	3.24%	1.27%
Transformed	98.23%	97.86%	97.17%	93.88%	91.51%	87.67%	65.16%	62.73%	52.33%	43.32%

Fig. 4. Top-1 classification accuracy under FGSM attack scenario. The **eps** suggests the power of an attack, a larger epsilon means a stronger attack.

tions make attackers hard to reconstruct the corresponding input transformation and difficult to attack the model.

REFERENCES

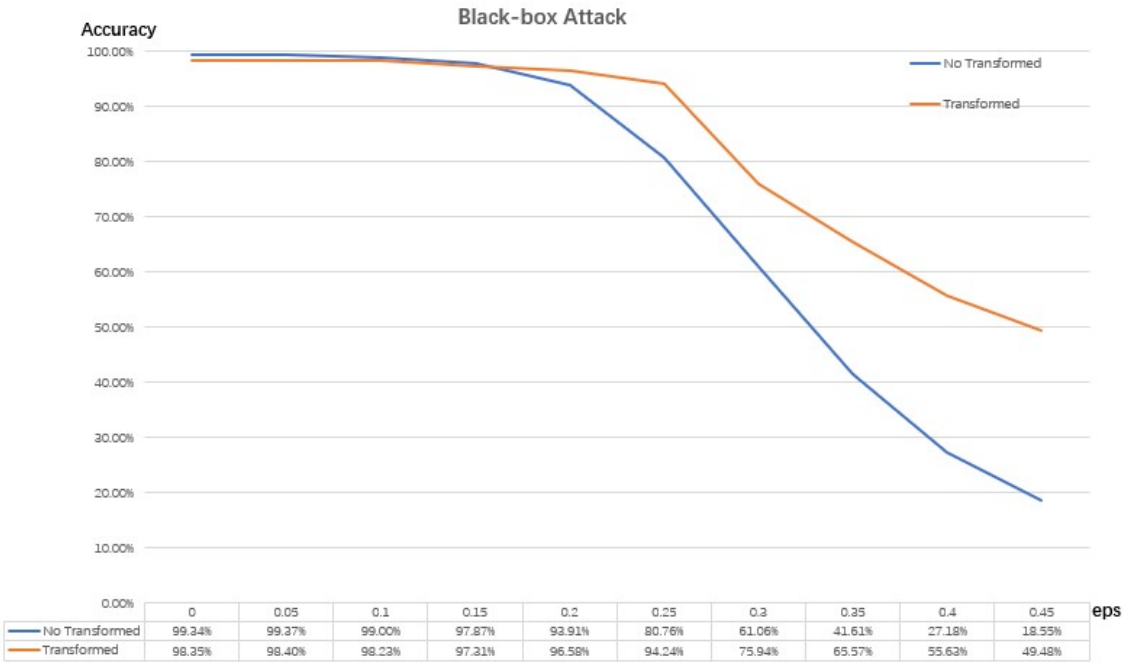
[1] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples (2014). arXiv preprint arXiv:1412.6572.

[2] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. CoRR, abs/1312.6199, 2013.

[3] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. Adversarial manipulation of deep representations. arXiv preprint arXiv:1511.05122, 2015

[4] Florian Tramer, Alexey Kurakin, Nicolas Papernot, Ian 'Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. arXiv preprint arXiv:1705.07204, 2017.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Gen-



	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45
No Transformed	99.34%	99.37%	99.00%	97.87%	93.91%	80.76%	61.06%	41.61%	27.18%	18.55%
Transformed	98.35%	98.40%	98.23%	97.31%	96.58%	94.24%	75.94%	65.57%	55.63%	49.48%

Fig. 5. Top-1 classification accuracy under black-box attack scenario. The **eps** suggests the power of an attack, a larger epsilon means a stronger attack.

erative adversarial nets. In Advances in neural information processing systems, pages 26722680, 2014.

[6] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

[7] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. CoRR, abs/1801.02610, 2018.

[8] Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models. In 2018 IEEE Security and Privacy Workshops (SPW), pages 3642. IEEE, 2018.

[9] Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. CoRR, abs/1703.09387, 2017.

[10] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236, 2016.

[11] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. arXiv preprint arXiv:1704.03976, 2017.

[12] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In 2016 IEEE Symposium on Security and Privacy (SP), pages 582597. IEEE, 2016.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. 2014.

[14] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. CoRR, abs/1805.06605, 2018.

[15] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 135147. ACM, 2017.

[16] Susmit Jha, Uyeong Jang, Somesh Jha, and Brian Jalaian. Detecting adversarial examples using data manifolds. In MILCOM 2018. IEEE Military Communications Conference (MILCOM), pages 425430. IEEE, 2018.

[17] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. CoRR, abs/1710.10766, 2017.

[18] N. Narodytska and S. P. Kasiviswanathan, Simple Black-Box Adversarial Perturbations for Deep Networks, arXiv preprint arXiv:1612.06299, 2016.

[19] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, Zoo: Zeroth Order Optimization Based Black-Box Attacks to Deep Neural Networks without Training Substitute Models, in Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, 2017, pp. 1526.

[20] C.-C. Tu, P. Ting, P.-Y. Chen, S. Liu, H. Zhang, J. Yi, C.-J. Hsieh, and S.-M. Cheng, AutoZOOM: Autoencoder-based Zeroth Order Optimization Method for Attacking Black-box Neural Networks, arXiv preprint arXiv:1805.11770, 2018.

[21] J. Hayes and G. Danezis, Machine Learning as An Adversarial Service: Learning Black-Box Adversarial Examples, arXiv preprint arXiv:1708.05207, 2017.

[22] W. Brendel, J. Rauber, and M. Bethge, Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models, arXiv preprint arXiv:1712.04248, 2017.

[23] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, Practical Black-Box Attacks against Machine Learning, in Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 2017, pp. 506519.