МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА
Факультет Вычслительной Математики и Кибернетика
Кафедра Информационной Безопасности

Ли Хуаюй

«Собрание аннотаций по научным статьям»

Научный руководитель: канд. физ.-мат. наук
с.н.с лаборатории ОИТ Намиот Д.Е

Москва, 2021

Contents

1 Intriguing properties of neural networks

1.1 Link

This article [1] was found and cited at the link: `http://arxiv.org/abs/1312.6199`.

1.2 Introduction

This paper introduces the concept of adversarial samples for the first time and presents a gradient method for generating adversarial samples based on the Box-constrained L-BFGS White-Box attack. The so-called adversarial sample is the introduction of a small perturbation on the original sample, which can make the model misclassify the situation, which is also the idea inspired by the adversarial generative network. Also, this paper argues that the semantic information in deep neural networks is based on the whole network and not on the neurons of a particular layer.

Note 1.1. *In the 3-rd chapter, the property is proved (experiments are performed on the MINST dataset): in the higher levels of the neural network, it is the whole space that contains semantic information, not a particular unit.*

In the 4-th section, the property is described: blind spots in neural networks - finding Adversarial Examples. and the idea of finding adversarial samples is provided: formulaic description, experimental results, conclusions.

Note 1.2. *(Why do adversarial samples arise in neural networks?) The authors give the following hypothesis(In the 4-th section):*

1. *It is assumed that is possible for the output unit to assign nonsignificant (and, presumably, non-epsilon) probabilities to regions of the input space that contain no training examples in their vicinity.*

2. *The adversarial examples represent low-probability (high-dimensional) "pockets" in the manifold, which are hard to efficiently find by simply randomly sampling the input around a given example.*

1.3 Key idea

According to the authors, finding the adversarial samples is a stepwise optimization process divided into two aspects:

1. We need to make sure that the added perturbations are as small as possible so that they are imperceptible to the naked eye.

2. On the other hand, we need to make sure that the model misclassifies the adversarial samples.

Based on these two points, the authors give the following objective function:

$$\begin{aligned}&\text{Minimize } ||r||_2 \text{ subject to :}\\&1.\ f(x+r)=l\\&2.\ x+r\in[0,1]^m\end{aligned}$$

Where,

1. x is the original image.
2. r is the added perturbation
3. f is the classifier.
4. l is the target category (which is different from the correct category for x).

We want r to be as small as possible, while making the adversarial sample $x+r$ be misclassified under a specified category l. We also need the generated $x+r$ to have a value between $[0,1]$ (to ensure it is a legitimate image). But finding a suitable solution to this problem is not easy, and in the paper the authors give a modified solution: finding the optimal perturbation r from the perspective of the loss function:

$$\text{Minimized } c|r| + \text{loss}_f(x+r,l) \text{ which subjects to } x+r\in[0,1]^m$$

This problem can be solved by the Box-Constrained L-BFGS algorithm.

1.4 Datasets

The Datasets used in this article are as follows:

1. MNIST dataset
   —A simple fully connected network consisting of one or more hidden layers and softmax classifiers — called "FC".
   —A classifier trained on an autoencoder — called "AE".

2. ImageNet dataset
   —AlexNet Network.

3. 10M image samples from Youtube
   —Unsupervised training network with 1 billion learnable parameters — called "QuocNet".

1.5 Result

The experimental results are detailed in the original article.(In the subsection 4.2)

1.6 Conclusion

Based on the experimental results in the paper, the following conclusions can be roughly drawn:

1. For all the Network Frameworks mentioned in the paper (AlexNet, QuocN...), adversarial samples can be generated by using methods(In the subsection 4.1).

2. Adversarial samples have the ability to generalize across models: a large portion of the adversarial samples generated on model A are also valid on model B (which has the same structure as model A with different hyperparameters).

3. D_1, D_2 are different subsets of dataset D, each trained with a different model, and the adversarial samples generated on D_1 are also valid on D_1.

2 Explaining and harnessing adversarial examples

2.1 Link

This article [2] was found and cited at the link: `http://arxiv.org/abs/1412.6572`.

2.2 Introduction

Currently, machine learning has been widely used in various fields of daily life, but Szegedy et al. first discovered in 2014 that current machine learning models, including neural networks, are vulnerable to adversarial examples. The so-called adversarial examples, i.e., attackers generate adversarial samples by slightly perturbing normal samples, and achieve the purpose of misleading the classifier while ensuring that the attack does not affect the recognition of human eyes.

This article mainly proposes a linear hypothesis that is different from the previous papers to explain the existence of adversarial examples. At the same time, the paper proposes a simple method for generating adversarial examples called FGSM, and then uses the adversarial examples generated by the attack method for adversarial training. In general, this article mainly describes three aspects of adversarial samples:

1. Existence
2. Attack methods
3. Defense methods

2.3 Key idea

1. The reason why neural networks are so vulnerable to adversarial sample attacks is because of the linear nature of the network.
2. The article also presents the earliest FGSM adversarial sample generation method.
3. The authors also propose an alternative perturbation: a very small rotation of the image that increases the value of the loss function.
4. By adding a certain number of adversarial samples (randomly generated) to the training samples, a certain regularization effect on the model can be achieved.
5. Adversarial training also improves the robustness of the model to adversarially perturbed samples, and the authors also make the point that perturbing the input samples in adversarial training is more effective than perturbing the hidden layer features, and the experiments show a weak regularization effect of the training model obtained from hidden layer perturbation. (Some ideas in the section 3,4,6)

2.4 Datasets

The Datasets used in this article are as follows:

1. MNIST dataset

2.5 Result

For the adversarial sample experiments: The authors experimented with Softmax and Maxout neural networks on the MNIST dataset and the parameters $\epsilon = 0.25$ and $\epsilon = 0.1$, respectively, and the error rates for the adversarial samples were as follows:

- $\epsilon = 0.25$: 99.9% and 89.4%.

- $\epsilon = 0.25$: 99.9% and 89.4%.
- $\epsilon = 0.1$: Maxout 87.15%.(In the section 4)

and the confidence level of the model for both error samples is extremely high.
For the adversarial training experiments:

- The error rate of Maxout network for the adversarial sample before adversarial training was 89.4%.
- After adversarial training, the error rate decreases to 17.9%.
- Also, the error rate of the adversarial samples generated by the model after adversarial training is effectively reduced in the original model.(In the section 5)

2.6 Conclusion

1. The adversarial samples can be interpreted as a high-dimensional product, and they are more due to the high-dimensional linear features of DNNs.
2. The generalizability of the adversarial samples across multiple models of the same task can be interpreted as the high similarity of different models in terms of weights.
3. The perturbation direction is more important compared to the particular point in the sample space.
4. Confrontation training helps regularization.
5. The authors propose that the fully nonlinear model RBF network can resist the adversarial samples to some extent (making their misclassification confidence relatively low). (More informations are in the section 10)

3 Adversarial examples in the physical world

3.1 Link

This article [3] was found and cited at the link: `http://arxiv.org/abs/1607.02533`.

3.2 Introduction

This article was published by Goodfellow and others. It is a classic paper in the field of adversarial examples. This paper differs from others in that it focuses on the input of adversarial samples to the convolutional neural network Inceptionv-Net3 through sensors such as cameras, which is equivalent to an actual attack in the physical world. The paper also proposes BIM(BASIC ITERATIVE METHOD):

$$X_0^{adv} = X, X_{N+1}^{adv} = \mathrm{Clip}_{X,\epsilon}\left\{X_N^{adv} + \alpha \mathrm{sign}(\nabla_X J(X_N^{adv}, y_{true}))\right\},$$

and ILCM(ITERATIVE LEAST- LIKELY CLASS METHOD):

$$X_0^{adv} = X, X_{N+1}^{adv} = \mathrm{Clip}_{X,\epsilon}\left\{X_N^{adv} - \alpha \mathrm{sign}(\nabla_X J(X_N^{adv}, y_{LL}))\right\},$$

adversarial sample generation methods and compares the results with the previously proposed FGSM algorithm [2] on the ImageNet dataset. This paper proposes new attack methods that introduce metrics for the effectiveness of adversarial attacks, with data collected through actual physical cameras. Then a white-box attack was performed in the real world, and finally a black-box attack was performed.

3.3 Key idea

1. BIM method
2. ILCM method
3. Attacks in the physical world are achieved by collecting printed clean samples and counter samples through the phone camera. When the denominator is 1, clean samples are correctly classified and adversarial samples are incorrectly classified, and when the numerator is 1, clean samples are correctly classified and adversarial samples are incorrectly classified, and after transformation (taken after printing) the adversarial samples are correctly classified. (detailed in the subsection 3.1)

3.4 Datasets

The Datasets used in this article are as follows:

1. ImageNet dataset

3.5 Result

1. The adversarial images generated by the "Fast" method are more robust than those generated by the iterative method because the iterative method generates more minute perturbations that are ignored by the photographic transformation.

2. In some cases, the adversarial corruption rate is higher in the "Preiltered case" than in the "Average case".(In the subsection 3.2 and 3.3)

3. Since black-box scenarios are a more realistic model for many security threats. The authors demonstrate that physical adversarial samples deceive a model of a different architecture through a black box attack. The experiments are demonstrated through an open source image classification mobile app.(In the subsection 3.4)

4. The "fast method" is the most robust and the "iterativa least-likely" is the least robust under the photo transformation.

5. Contrast and luminance changes do not have a significant effect on the counter samples.

6. Blur, noise and JPEG encoding have higher corruption rates than contrast and luminance changes.(In the subsection 3.5)

3.6 Conclusion

The authors' team used images taken from a cell phone camera as input to the Inception v3 image classification neural network. The authors show that in such a setup, most of the adversarial images produced using the original network are misclassified even when fed into the classifier via the camera, a finding that demonstrates the possibility of adversarial samples for machine learning systems in the physical world.

4 On Detecting Adversarial Perturbations

4.1 Link

This article [4] was found and cited at the link: `https://arxiv.org/abs/1702.04267`.

4.2 Introduction

The detection network approach will directly predict whether a given sample is an adversarial sample by means of a neural network, i.e., the identification of an adversarial sample is directly transformed into a binary classification problem trained in an end-to-end manner. The detector network approach extends the original neural network (classifier) with a detector sub-network whose task is to discriminate whether the sample is from real data or not. The output of the detector is a scalar of range that represents the probability that the data belongs to the adversarial sample. The design of the detector is related to the specific dataset and the architecture used is generally a convolutional neural network.

4.3 Key idea

To train this detector, the classifier is first trained on the original training Dataset, and when the classifier is trained, a corresponding adversarial sample is generated for each sample in the training Dataset using methods such as FGSM or DeepFool.

This results in one original Dataset and one adversarial sample Dataset of the same size, and then the detector is trained. The training data consists of half of the real sample data and half of the generated adversarial samples, with the labels of the samples replaced with the corresponding sources, i.e., the real samples or the adversarial samples. For detector training, the weights of the original classification network are fixed and the detector is trained using cross-entropy as the loss function.

4.4 Datasets

The Datasets used in this article are as follows:

1. ImageNet dataset
2. CIFAR10 dataset

4.5 Result

Experiments show that when the FGSM method is used to generate adversarial samples, setting , the detection network can detect 90% of the adversarial samples, and setting , the detection network can detect 97% of the adversarial samples; when the DeepFool method is used to generate adversarial samples, the detection network can detect 82% of the adversarial samples. When the attacker knows the gradients of both the classification network and the detection network, and the FGSM method is used for dynamic adversarial training, setting , the detection network can detect 89% of the adversarial samples.(In the section 4)

4.6 Conclusion

1. Using the detector sub-network attached to the main classification network, adversarial examples can be detected.
2. Although this does not directly allow the correct classification of adversarial examples, it can mitigate adversarial attacks on machine learning systems by resorting to backup solutions.
3. The gradient propagated back through the detector may be used as a source of regularization of the classifier against adversarial examples.
4. Developing methods for training detectors explicitly such that they can detect many different kinds of attacks reliably at the same time would be essential for safety- and security-related applications.

5 Synthesizing Robust Adversarial Examples

5.1 Link

This article [5] was found and cited at the link: `https://arxiv.org/abs/1707.07397`.

5.2 Introduction

After analyzing the previous related work on adversarial sample generation, the authors Athalye et al. made a more in-depth study in the direction of physical environment transformation. Questions are raised on the generation and effectiveness of adversarial samples in 2D, 3D and real-world environments, respectively. This work proposes the framework of Expectation OverTransformation (EOT) algorithm. Simulating specific environments (including 2D and 3D), the authors also propose an adversarial sample generation algorithm based on this framework, which generates samples with good robustness in various perspectives and given distributions of environmental conditions. Real-world adversarial objects are also generated based on this framework with the support of 3D printing technology to demonstrate the existence of 3D adversarial objects.

5.3 Key idea

The adversarial samples obtained by the above method are not immune to various visual transformations. Therefore, based on the above method, EOT introduces the transformation distribution T. For any transformation function t, the input of the classifier is changed from the original adversarial sample x' to $t(x')$. In practice, various transformations such as rotation, translation, noise addition, etc. can be represented. Once the EOT is parameterized, the distribution T is determined and the EOT framework can optimize the samples under the distribution T to obtain the adversarial samples.(In the subsection 2.1)

5.4 Datasets

The Datasets used in this article are as follows:

1. ImageNet dataset

5.5 Result

In the experimental session, the authors used different transform distributions for different experimental environments. In the 2D environment, the authors used transform distributions including scaling, rotation, illumination, Gaussian noise, and other transformations, and the expectation of the transformations was calculated by randomly sampling 1000 transformations from these transform distributions.

In the 3D environment, the authors modeled the transformations by 3D rendering, considering camera distance, lateral translation, object rotation, and solid color background, and randomly sampled 100 transformations to represent the distribution, Figure 16.16 shows the 3D adversarial sample example. In the 3D physical environment, the authors used 3D model printing to fabricate an adversarial sample model, considering various angles and backgrounds for the classification of the adversarial samples.(Detailed in the subsection 3.2 and 3.3)

5.6 Conclusion

The authors prove the existence of robust 3D adversarial objects and present the first algorithm for synthesizing adversarial samples over a selected transform distribution.

The authors synthesize 2D adversarial images that are robust to noise, distortions and affine transformations.

Finally, the authors apply the algorithm to complex 3D objects, using 3d printing to fabricate the first physical adversarial object.

6 Deepfool: a simple and accurate method to fool deep neural networks

6.1 Link

This article [6] was found and cited at the link: `https://garxiv.org/abs/1511.04599`.

6.2 Introduction

This paper is a classic paper in the direction of Adversary Attack. The algorithm is called DeepFool and its goal is to find the minimum perturbation to achieve the goal of generating an adversarial sample.

6.3 Key idea

The section introducing the algorithms is divided into binary(in the section 2) and multiclass classifiers(in the section 3), each of which is extended from the linear to the non-linear case respectively. A simple and accurate method is proposed to compute the robustness of comparative misclassifiers to adversarial perturbations.

6.4 Datasets

1. CIFAR-10
2. MNIST

6.5 Result

The authors compare the robustness of three algorithms, EOT, FGSM and deepfool. See section 4.

6.6 Conclusion

1. In this paper, we propose a new algorithm DeepFool.
2. This paper is based on the iterative linearization of classifiers to generate small perturbations and produce very effective attacks.

3. this paper provides extensive experimental evidence on three datasets and eight classifiers, demonstrating the superiority of the method over existing methods for computing adversarial perturbations, and the effectiveness of the method.

7 Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples

7.1 Link

This article [7] was found and cited at the link: `https://arxiv.org/abs/1802.00420`.

7.2 Introduction

Some existing defence models based on counter-sample attacks appear to be able to resist attacks based on optimisation strategies, but in reality, this is an illusion; in an iterative attack, we encounter a phenomenon similar to gradient masking, which we call "obfuscated gradients". Iterative training based on obfuscated gradients will fail to obtain an optimal loss, thus giving the illusion that the attack has failed. The phenomenon of obfuscated gradients can be divided into the following three types, and the solutions are as follows:

1. Shattered Gradients: Back Pass Differentiable Approximation (BPDA)
2. Stochastic Gradients: Expectation Over Transformation
3. Vanishing/Exploding Gradients:Reparameterisation or Optimisation in the space beyond the gradient explosion/vanishing

.

7.3 Key idea

There are three common approaches to combating obfuscated gradients:

1. Shattered gradients are non-existent or incorrect gradients caused intentionally (through non-minimisable manipulation) or unintentionally (through numerical instability) by the defender.
2. Random gradients are caused by random defences.
3. Vanishing/exploding gradients are found in very deep neural network tests.

7.4 Datasets

1. CIFAR-10
2. MNIST
3. ImageNet

7.5 Result

7.6 Conclusion

8 Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks

8.1 Link

This article [8] was found and cited at the link: `https://arxiv.org/abs/1704.01155`.

8.2 Introduction

Feature compression reduces the search space available to the adversary by combining samples corresponding to many different feature vectors in the original space into a single sample. By comparing the DNN model's prediction of the original input with the prediction of the compressed input, feature compression can detect adversarial samples very well. The method is not demanding on computational resources, can be complementary to other defences and can be used in conjunction with a joint detection framework.

8.3 Key idea

1. Reduce the colour bit depth of each pixel.
2. Use spatial smoothing to reduce the difference between individual pixels.

8.4 Datasets

1. CIFAR-10
2. MNIST
3. ImageNet

8.5 Result

See Pages 10, 11, 12

8.6 Conclusion

By predicting the correct labels for non-adaptive adversarial samples, feature compression can significantly enhance the robustness of the model while retaining the accuracy of legitimate inputs, allowing for accurate detection of static adversarial samples.

9 Efficient Defenses Against Adversarial Attacks

9.1 Link

This article [9] was found and cited at the link: `https://arxiv.org/abs/1707.06728`.

9.2 Introduction

This thesis proposes an effective defence method based on real-world observations that is easy to integrate into the model and performs better than the best defence strategies available to date. The main strategy is to enhance the structure of the deep neural network so that the predictions are more stable and less likely to be confused by the adversarial samples. The paper also compares with other defence methods in the context of different attack strategies, black-box and white-box attacks. Moreover, the proposed method of enhancing the structure of the neural network in this paper does not impose a training burden and negative optimisation of the training effect on the initial deep neural network model.

9.3 Key idea

A double-defence method is proposed that can accomplish the configuration with less impact on the standard training cost of the original model. A series of experiments are conducted to apply various counter-attack methods to various defence methods, including the proposed defence method, and to explore black-box and white-box attacks and the portability of the attacks. The effectiveness of the various methods is also evaluated, proving that the effectiveness of the defence method is imperfect when judged by accuracy alone. The new defence approach proposed in this paper starts with two perspectives:

1. Bounded ReLU: Here t is set according to the range of inputs. Traditional ReLU activation functions, which are constrained by the size of t and the weights learned between layers, can therefore enhance the stability of the network.

2. Gaussian data augmentation: The idea of this defence method is to augment the generalisation ability of the model so that the model can classify the original data and the perturbed data in the same way.

9.4 Datasets

1. CIFAR-10

2. MNIST

9.5 Result

See Pages 13, 14(This work proposes two such strategies, used alone or in combination, that can improve the robustness of deep models.)

9.6 Conclusion

This work proposes two such strategies, used alone or in combination, that can improve the robustness of deep models.

10 Detecting Adversarial Samples from Artifacts

10.1 Link

This article [10] was found and cited at the link: `https://arxiv.org/abs/1703.00410`.

10.2 Introduction

Neural networks are more robust to random noise than many traditional machine learning algorithms, but recent research has shown that they are much more vulnerable to adversarial samples. An adversarial sample is a sample that is misclassified by a classifier by adding a specific noise interference to a clean sample (basically by increasing the loss function of the classifier). The emergence of adversarial samples has created a wave of research in the field of machine learning security, and in previous work Szegedy and Goodfellow have discussed the causes of adversarial samples, each in their own way. In summary, however, the intuition is that there is a distance between the adversarial sample and the distribution of the real data (the original statement is that the adversarial sample exists slightly outside the flow region away from the real data). The authors accordingly propose two methods for analysing the distribution of the adversarial sample:

1. Kernel density estimation (KDE): this is computed in the feature space of the last layer of hidden neurons. This method can be used to detect and analyse points that are far from the data stream shape.

2. Bayesian uncertainty estimation: the parameters of the uncertainty estimation can be obtained by dropout, this method can be used to detect points located at low confidence levels and can detect points that are not detected by KDE.

10.3 Key idea

The authors argue that this can be explained by the idea of data streamlining. Many training data, like pictures, which actually exist in low-dimensional streamlined regions in a high-dimensional space. The anti-perturbation does not change the true labels (latent labels) of the original data, it simply moves the data out of the data stream shape. The authors' hypothesis is based on this, i.e. that the counteracting samples are outside the data manifold.

10.4 Datasets

1. CIFAR-10

2. MNIST

3. SVHN

10.5 Result

See page 8.

10.6 Conclusion

Experiments show that adversarial samples crafted to fool DNNs can be effectively detected using two new features: kernel density estimation in the last hidden layer subspace, and Bayesian neural network uncertainty estimation.

11 Adversarial and Clean Data Are Not Twins

11.1 Link

This article [11] was found and cited at the link: `https://arxiv.org/abs/1704.04960`.

11.2 Introduction

11.3 Key idea

11.4 Datasets

11.5 Result

11.6 Conclusion

12 On the Geometry of Adversarial Examples

12.1 Link

This article [12] was found and cited at the link: `https://arxiv.org/abs/1811.00525`.

12.2 Introduction

12.3 Key idea

12.4 Datasets

12.5 Result

12.6 Conclusion

13 Generative Adversarial Networks

13.1 Link

This article [10] was found and cited at the link: `https://arxiv.org/abs/1703.00410`.

13.2 Introduction

13.3 Key idea

13.4 Datasets

13.5 Result

13.6 Conclusion

14 Generative Adversarial Networks

14.1 Link

This article [13] was found and cited at the link: `https://arxiv.org/abs/1406.2661`.

14.2 Introduction

14.3 Key idea

14.4 Datasets

14.5 Result

14.6 Conclusion

15 Generative Adversarial Networks

15.1 Link

This article [13] was found and cited at the link: `https://arxiv.org/abs/1406.2661`.

15.2 Introduction

15.3 Key idea

15.4 Datasets

15.5 Result

15.6 Conclusion

16 Generative Adversarial Networks

16.1 Link

This article [13] was found and cited at the link: `https://arxiv.org/abs/1406.2661`.

16.2 Introduction

16.3 Key idea

16.4 Datasets

16.5 Result

16.6 Conclusion

17 Generative Adversarial Networks

17.1 Link

This article [13] was found and cited at the link: `https://arxiv.org/abs/1406.2661`.

17.2 Introduction

17.3 Key idea

17.4 Datasets

17.5 Result

17.6 Conclusion

18 Generative Adversarial Networks

18.1 Link

This article [13] was found and cited at the link: `https://arxiv.org/abs/1406.2661`.

18.2 Introduction

18.3 Key idea

18.4 Datasets

18.5 Result

18.6 Conclusion

19 Generative Adversarial Networks

19.1 Link

This article [13] was found and cited at the link: `https://arxiv.org/abs/1406.2661`.

19.2 Introduction

19.3 Key idea

19.4 Datasets

19.5 Result

19.6 Conclusion

20 Generative Adversarial Networks

20.1 Link

This article [13] was found and cited at the link: `https://arxiv.org/abs/1406.2661`.

20.2 Introduction

20.3 Key idea

20.4 Datasets

20.5 Result

20.6 Conclusion

21 Generative Adversarial Networks

21.1 Link

This article [13] was found and cited at the link: `https://arxiv.org/abs/1406.2661`.

21.2 Introduction

21.3 Key idea

21.4 Datasets

21.5 Result

21.6 Conclusion

22 A List Of Articles

References

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

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

[3] A. Kurakin, I. Goodfellow, S. Bengio, *et al.*, "Adversarial examples in the physical world," 2016.

[4] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[5] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *International conference on machine learning*, pp. 284–293, PMLR, 2018.

[6] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "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.

[7] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International conference on machine learning*, pp. 274–283, PMLR, 2018.

[8] W. Xu, D. Evans, and Y. Qi, "Feature squeezing: Detecting adversarial examples in deep neural networks," *arXiv preprint arXiv:1704.01155*, 2017.

[9] V. Zantedeschi, M.-I. Nicolae, and A. Rawat, "Efficient defenses against adversarial attacks," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 39–49, 2017.

[10] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *arXiv preprint arXiv:1703.00410*, 2017.

[11] Z. Gong, W. Wang, and W.-S. Ku, "Adversarial and clean data are not twins," *arXiv preprint arXiv:1704.04960*, 2017.

[12] M. Khoury and D. Hadfield-Menell, “On the geometry of adversarial examples,” *arXiv preprint arXiv:1811.00525*, 2018.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.