ADVFilter: Adversarial Example Generated by Perturbing Optical Path

Lili Zhang[1]* and Xiaodong Wang[1]

College of Computer, National University of Defense Technology, Changsha, China
{zhanglili18,xdwang}@nudt.edu.cn

Abstract. Deep Neural Networks (DNNs) have achieved great success in many applications, and they are taking over more and more systems in the real world. As a result, the security of DNN system has attracted great attention from the community. In typical scenes, the input images of DNN are collected through the camera. In this paper, we propose a new type of security threat, which attacks a DNN classifier by perturbing the optical path of the camera input through a specially designed filter. It involves many challenges to generate such a filter. First, the filter should be input-free. Second, the filter should be simple enough for manufacturing. We propose a framework to generate such filters, called ADVFilter. ADVFilter models the optical path perturbation by thin plate spline, and optimizes for the minimal distortion of the input images. ADVFilter can generate adversarial pattern for a specific class. This adversarial pattern is universal for the class, which means that it can mislead the DNN model on all input images of the class with high probability. We demonstrate our idea on MNIST dataset, and the results show that ADVFilter can achieve up to 90% success rate with only 16 corresponding points. To the best of our knowledge, this is the first work to propose such security threat for DNN models.

Keywords: adversarial example · deep neural networks · security threat · physical attack.

1 Introduction

Deep Neural Networks (DNNs)[1] have achieved tremendous success in many real world applications. DNN models are taking over the control of more and more systems, which are traditionally considered to be operated only by humans, such as automatic pilot system. Therefore, the security [2] of DNN is directly related to the safety of the physical world, including the safety of human life and property. The community have conducted extensive research on DNN security [2].

An important direction of DNN security is adversarial example attack [3], [4]. A deliberate perturbation on the input data can mislead the DNN model to incorrect predictions. For computer vision applications, the DNN model usually

* Corresponding author

uses the camera to obtain the input image. Surprisingly, most cameras are physically accessible to malicious users, so that it is difficult to detect abnormality even when installing or refitting a filter on the camera. Traditional security research only focuses on the electronic part of the DNN system, while this paper points out that the optical part can also trigger adversarial attacks.

Compared with traditional adversarial example attack, adversarial filter should conquer more difficulties. First, traditional adversarial example usually handles a specific input image, while adversarial filter is input independent. Clearly, we cannot preset the input image of the camera, so that adversarial filter should mislead the DNN model on the whole data distribution of a certain class. Second, traditional adversarial example can apply pixel-level modification, while adversarial filter should be simple enough for manufacturing. Under the view of adversarial example, it means that adversarial filter has much fewer dimensions to operate than the pixel-level adversarial example. Consequently, the difficulty of generating adversarial filter is much higher.

To conquer such difficulties, we propose a novel framework to generate adversarial filters, called ADVFilter. ADVFilter models the optical path perturbation by thin plate spline (TPS). We use as few corresponding points as possible in TPS method to control the comlexity of the result, and optimizes for the minimal distortion of the input images. For a given DNN model, ADVFilter can generate adversarial pattern for a specific class. This pattern acts as a virtual filter to perturb all input images of the DNN model. The adversarial pattern is universal for the selected class, which means that it can mislead the DNN model on all input images of the class with high probability. We demonstrate our idea on MNIST dataset, and the results show that ADVFilter can achieve up to 90% success rate with only 16 corresponding points. To the best of our knowledge, this is the first work to propose such security threat for DNN models.

To summarize, we list our contributions as follows:

1. We propose the security threat for DNN model by perturbing the optical path, and demonstrate that this threat can lead to serious consequences.

2. We propose novel framework to generate adversarial filters, named ADVFilter. ADVFilter can generate universal adversarial filters, which attacks a DNN model in an input-independent way. Moreover, ADVFilter adopts as few corresponding points as possible to simplify the result.

3. We conduct experiments to verify the idea by MNIST dataset. The results show that ADVFilter can achieve up to 90% success rate with only 16 corresponding points, which constitutes a space with only 64 dimensions.

The rest of this paper is organized as follows. We introduce the related works in Section 2. We overview the design of ADVFilter in Section 3. Next, we present the details of ADVFilter in Section 4 and show the experiment results in Section 5. Finally, we conclude this work in Section 6.

2 Related Work

The research on the security of neural networks has attracted significant efforts[3]–[7], Early work focused on generating adversarial examples in different ways to explore the border of DNN security. Another research direction is to attack black-box model in an efficient way. This section mainly discusses the research closely related to our work.

Some papers [8]–[12] propose to generate universal adversarial examples that are image-agnostic. The adversarial patterns generated by these methods can be applied to different images. These patterns can convert a given image into an adversarial example. These methods only consider the color space of the pixels, so that they cannot solve the problem of adversarial filters. Compared with these works, ADVFilter only adopts several corresponding points to generate universal adversarial pattern. The dimension of optimization space is much smaller than that of traditional methods, so our problem is correspondingly more difficult.

Some researchers explore image transformation to generate adversarial examples [13]–[15]. These methods generate adversarial examples by rotations and translations. They only obtain the adversarial ability on single target image, which cannot be generalized to our scenario. In contrast, ADVFilter generates a perturb pattern that can be applied to any image of a certain class. The problem of ADVFilter is more general and therefore more difficult to solve.

Some studies break the boundary between physics and cyber space, so as to generate adversarial examples in the physical world. Early attempt simply prints adversarial examples and fool DNN classifier through camera input [16]. Study [17] also show that deliberately constructed glasses can mislead face recognition systems. Besides, printable adversarial patches can control the model prediction to a predefined target class [18]. Moreover, some studies [19]–[22] explore robust adversarial examples in a predefined distribution. They show the ability to fool DNN model under a continuous viewpoints in three dimensional physical world. Adv t-shirt [23] generates a t-shirt with special designed pattern that can evade person detectors. This work adopts TPS to model the non-rigid surface of the t-shirt. Adv t-shirt shows the potential of TPS-based technique to generate physical adversarial examples. Compared with this study, ADVFilter deals more general cases that should fit all potential inputs of the target model.

In a word, perturbing optical path is a new type of DNN attack, which cannot be realized by simply expanding the existing technology.

3 ADVFilter Overview

This section introduces the design of the ADVFilter, and shows the architecture of the framework.

We show the architecture of ADVFilter in Figure 1. In the figure, the part surrounded by dotted lines is a standard DNN prediction pipeline. The model collects images through an input camera, and outputs the prediction results. In order to facilitate the algorithm description, we only discuss the case of white

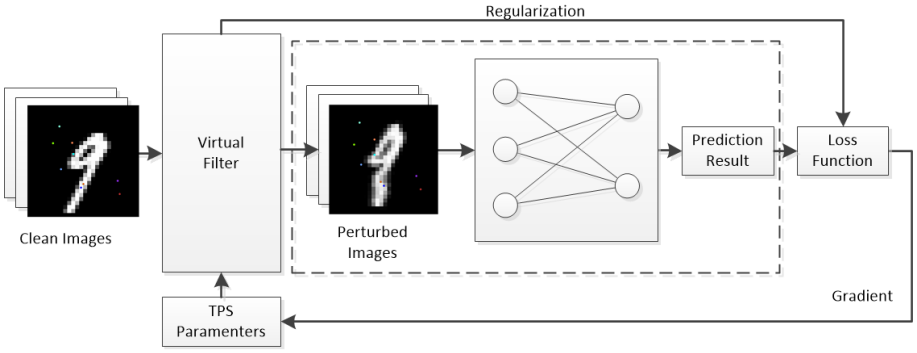

Fig. 1. The archetecture of ADVFilter

box attack in this section. We will discuss how to extend ADVFilter to black box attack in the following section. In order to evaluate the effect of a malicious filter on the camera, we design the virtual filter module, which is the core of the whole system. After adding the virtual filter, all input images of the DNN model will be universally perturbed by the virtual filter. In the figure, the clean images are the images of the physical world, and the perturbed images are the images perturbed by the virtual filter. Under this configuration, the DNN model can only get the perturbed images as the input.

We model the effect of the adversarial filter by TPS. TPS is a physical analogy involving the bending of a thin sheet of metal, which is widely used as the non-rigid transformation model in image alignment and shape matching. It achieves several advantages to adopt TPS for adversarial filter generation. First, TPS produces smooth surface, which is consistent with the characteristics of optical path change. Second, TPS adopts several discrete corresponding points to control the entire transformation over the whole image, which facilities the manufacturing of the adversarial filter. Third, TPS has closed form solutions for both warping and parameter estimation, which can be integrated to current deep learning frameworks. The mathematical details of the virtual filter module are discussed in the next section.

In the view of the learning architecture, the goal of generating adversarial filter is to get an appropriate set of parameters for the virtual filter model, i.e. the parameters of the TPS transformation. The goal of the optimization is two folded. On one hand, the result should mislead the target DNN model, so that the loss function contains the prediction result of the DNN model. On the other hand, the overall distortion of the image should be as low as possible. We add several regularization terms to the loss function to limit the deformation of the images. Finally, the parameters of TPS are optimized by gradient information. The final loss function of ADVFilter consists of three parts:

$$Loss = \lambda_1 L_{\text{CrossEntropy}} + \lambda_2 L_{\text{radius}} + \lambda_3 L_{\text{distortion}} \tag{1}$$

In the loss function, $L_{\text{CrossEntropy}}$ is standard cross entropy loss to control the output label, expressed as:

$$L_{\text{CrossEntropy}} = -\sum y_i \log(p_i) \tag{2}$$

where y_i denotes each element of the one-hot vector of target class, and p_i denotes each element of the predicted probability vector of the target model. The cross entropy of a batch is the mean of each cross entropy loss for each data in the batch.

Besides, L_{radius} and $L_{distortion}$ are two regularization terms to control the visibility of the adversarial filter, λ_1, λ_2 and λ_3 are hyper parameters to adjust the attack ability and distortion. These regularization terms are closely related to TPS implementation. We leave the detail design of this part to the following section.

4 ADVFilter Design

In this section, we formally define the problem of generating adversarial filters and show the mathematical details of ADVFilter design. We also adopt white-box attack model for clarity and discuss the black-box case in the end of this section.

We first introduce the formal definition of adversarial examples. Let set X and set Y denote the possible input and output of the target model, respectively. Given a particular input $x \in X$ and a target class $y \in Y$, we can obtain the model prediction $P(y \mid x)$, as well as the gradient $\nabla_x P(y \mid x)$. We use a function M to denote the predict result with maximum probability. For example, $M(x) = y$ means that label y is the most likely result under input x. Traditional adversarial example is to find an input x' within the ε-ball of the initial input x, i.e. $\|x' - x\| < \varepsilon$ that is classified as another class y_t, i.e. $M(x') = y_t$ and $y_t \neq y$.

The definition of adversarial filter is on the whole input set rather than a single image. An adversarial filter is defined as a transformation F with parameter θ on the whole input set X. For any input $x \in X$, the result of the transformation $F(x;\theta)$ is misclassified by the target model, i.e. $M(F(x;\theta)) = y_t$ and $y_t \neq y$. Hence, generating adversarial filter is to find an appropriate transformation $F(x;\theta)$ and optimize parameter θ.

ADVFilter adopts TPS as the transformation function. In two dimensional case, the TPS fits a mapping function between two corresponding point-sets $\{p_i\}$ and $\{p'_i\}$ one by one. The result is to minimize the following energy function:

$$E(F) = \sum_{i=1}^{K} \|p'_i - F(p_i)\|^2 \mid \tag{3}$$

where K is the number of corresponding points. We do not consider the smoothness variant of standard TPS, so that the mapping results accurately coincide with the target corresponding points. Given the corresponding point-sets $\{p_i\}$ and $\{p'_i\}$, the TPS has closed-form solution for the optimal mapping.

Clearly, the parameter space for standard TPS is $\theta = \{p_i; p'_i\}$. To simplify the expression in deep learning framework, we convert the TPS parameters into an equivalent form. We preserve the source corresponding point-sets $\{p_i\}$ and rewrite the target corresponding point-sets $\{p'_i\}$ in a relative form. That is,

$$p'_i = p_i + r_i R(\alpha_i) \tag{4}$$

where r_i denotes the distance between the two corresponding points, $R(\alpha) = \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \end{bmatrix}$ denotes the relative angle between the two corresponding points. The parameter space of the problem is $\theta = \{p_i, r_i, \alpha_i\}$. After the transformation, the radius regularization term of the loss function is as follow:

$$L_{\text{radius}} = \left(\sum r_i^2\right)^{1/2} \tag{5}$$

Moreover, we also set an upper bound value r_{max} for all radius values and clip the maximum absolute value of each r_i to r_{max}.

The distortion regularization term is defined as the distance between the original image and the transformed image as follow:

$$L_{\text{distortion}} = \left[\sum \left(x_i - F(x_i; \theta)\right)^2\right]^{1/2} \tag{6}$$

Note that all TPS transformations share the same parameter θ, which is the most significant difference between this work and existing researches.

The parameter space of the optimization is $\theta = \{p_i, r_i, \alpha_i\}$, where p_i denotes the locations of the source corresponding points, r_i and α_i denote the distance and relative angles between the corresponding points. We adopt Adam optimizer with learning rate 0.1 to search the optimal solution.

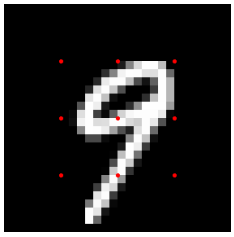

Fig. 2. Initial distribution of the corresponding points

The initialization of corresponding point $\{p_i\}$ is in grid form. Let $n = k^2$, $k \in N$ denote the total number of corresponding points. We set n to be a square number for simplicity. Suppose the size of the input image is $h \times w$, then the distribution of all k^2 points is a two-dimensional grid with an interval of $\frac{h}{k+1} \times \frac{w}{k+1}$, as shown in Figure 2. This design ensures that the TPS transformation can capture the vulnerability of any position of the input image.

For black-box attack case, ADVFilter can leverage off-the-shelf gradient estimation scheme, such as zero order optimization [7] or natural evolution strategies [24]. The cost of these algorithms depends heavily on the dimension number of the entire search space. Note that the search space of TPS is very low. For example, if we adopt 16 corresponding points, the dimension of the whole search space is only 16*4=64, which is far less than the dimensions of pixel color space. As a result, the overhead of gradient estimation is correspondingly low.

5 Experiments

We conduct several experiments to verify the correctness and the efficiency of ADVFilter.

5.1 Experiment setup

We conduct all the experiments on a laptop computer with Intel Core i7-8550U and 16G RAM. The software platform for deep learning is Tensorflow 1.15.4.

The target dataset is the MNIST database, which is a data collection of handwritten digits. In the process of generating the adversarial pattern, we record all the local optimal solutions. The criteria include two aspects: the success rate of the attacking and the average l2 distance between the clean images and the adversarial images in the tested batch. Based on this vector criterion, we record all results with lower average l2 distance or higher success rate. To facilitate comparison, we fix the average l2 distance to a predefined value and compare the success rate in differate settings.

For all the following tests, the batch size is 100, which means we randomly sample 100 images from the dataset. In each iteration, we clip the distances between all corresponding points to 0.05. That is, we set r_{max}=0.05 to control the distortion of the adversarial images.

5.2 Correctness test

This section demonstrates how adversarial filter attacks the DNN model. We conduct an untargeted attack with all images labeled “9” in the dataset. The number of corresponding points is 9, which are arranged in 3*3 grid initially.

Figure 3 shows the experiment results. The first row of the images are the clean images, which are the input of the the adversarail filter. The second row of the images are the adversarial images, which are output of the adversarail filter. The colored dots are the corresponding points of the TPS conversion. Clearly, the same color indicates the corresponding relationship of points. After several iterations, all correspongding points are far away from their initial grid position, and moves to the sensitive position to change the features of the target images.

Note that all the adversarail images follow the same conversion pattern, which is the major difference between our work with existing ones. Comparing the clean images with the adversarail images, we found that most most areas of the

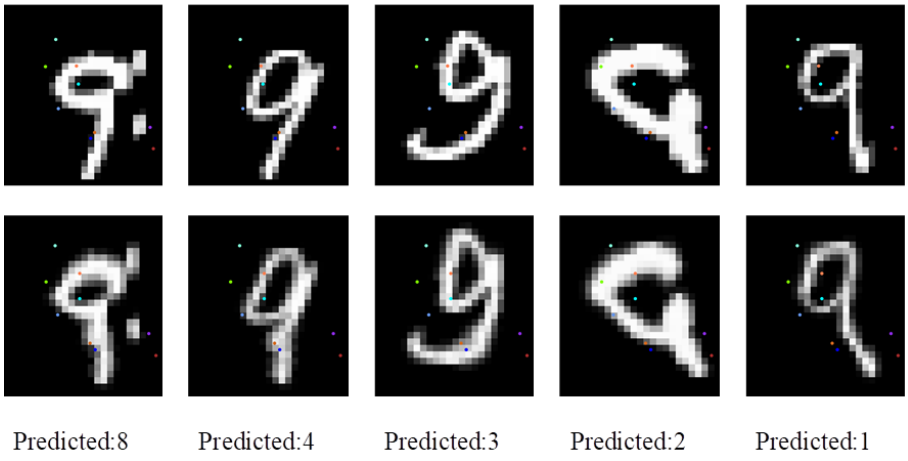

Fig. 3. The clean images and the adversarial images in MNIST dataset

images had only a small amount of modification, and only the lower right part is deflected sharply to the right. Through the visual judgment of human eyes, it is difficult to distinguish between the clean images and adversarail images. Therefore, we will still label these adversaril images as "9". In contrast, these adversarail images are all misclassified by the DNN classifier.

5.3 Comparison with SOTA universal adversarial examples

This section compare our ADVFilter with existing universal adversarial examples [8]–[11]. We perform the experiment with the same configuration as in Section 5.2. That is, we use the same DNN model, the same dataset, and the same source image set. Under this configuration, we generate pixel-level universal adversarial example (PUAE), as shown in Figure 4. The perturbation of all adversarial examples follows the same pattern. After adding the perturbation, we clip the color value of each pixel to the legal range.

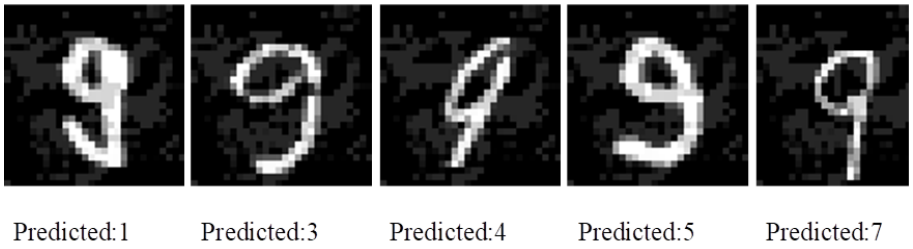

Fig. 4. The adversarial examples generated by perturbing color space

It is difficult to make a quantitative comparison between PUAE and ADVFilter, because there are inherent difference in the techniqual roadmap. We first discuss the similarities between PUAE and ADVFilter. First, both PUAE and ADVFilter generate a universal adversarial pattern, which is independent of specific input image. Second, they all achieved a relatively high attack success rate, i.e. 90% in our experiment. Finally, the adversarial examples generated by both methods are not easily perceptible to humans, while the invisibility are defined in completely different ways. Next, we discuss the differences between PUAE and ADVFilter. First of all, the dimension of freedom for manipulating color space is much larger than that of TPS. As a result, the problem of generating an adversarial filter is much more difficult. Second, The result of ADVFilter only change the position distribution of pixels, hence the distribution of the color space is similar to the clean image. In contrast, PUAE significantly changes the color distribution of the image. Last, in terms of physical manufacturing, advfilter does not need pixel level alignment, thus to accept higher manufacturing errors. In contrary, PUAE requairs pixel-level alignment accuracy, which is more challenging.

5.4 The number of corresponding points

This section discusses the impact of the number of corresponding points on the success rate. We conduct an untargeted attack with all images labeled "9" in the dataset. We repeat the experiments with different number of corresponding points and record the success rate under different settings, as shown in Figure 5.

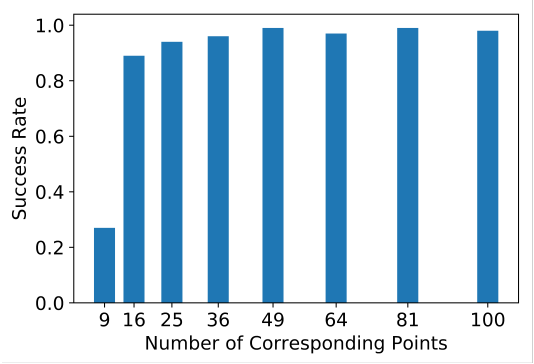

Fig. 5. The success rate under various numbers of corresponding points

In order to evenly distribute all corresponding points in the image, we set the number of corresponding points to be square number. The test range covers from three square to ten square, i.e. from 9 to 100. We conclude from the figure that the success rate will increase approximately monotonically with the increase of

the number of the corresponding points in the first half. However, there is an upper limit of this increase trend. After reaching a specific value, i.e. 36 in this test, the success rate will remains stable.

This experiment shows that universal adversarial examples can be generated in a very low dimensional subspace. For example, in this experiment, we use only 16 corresponding points to achieve the success rate of about 90%. Even if we ignore the restriction between the dimensions of the corresponding points, this means that our estimated dimension is higher than the actual one. The total dimension of this subspace is only 16*4=64.

6 Conclusion and Future Work

In this paper, we propose a new type of threat for DNN system, which is to generate adversarial example by perturbing optical path. One possible way to apply this idea is to add a specially designed filter to the input camera of the target DNN model. We define the problem of generating such an adversarial filter, and propose a framework, named ADVFilter. ADVFilter models the optical path perturbation by thin plate spline, and optimizes for the minimal distortion of the input images. The generated adversarail filter can mislead the DNN model on all input images of the class with high probability, which shows that the threat can lead to serious consequences. The future work is to physically produce a filter and test the attack effect in the physical world.

Acknowledgements We are particularly grateful to Inwan Yoo who implements TPS in Tensorflow and shares the code on https://github.com/iwyoo/tf_ThinPlateSpline.

References

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 2012, 2012, pp. 1097–1105.
2. X. Wang, J. Li, X. Kuang, Y. Tan, and J. Li, "The security of machine learning in an adversarial setting: A survey," J. Parallel Distrib. Comput., vol. 130, pp. 12–23, Aug. 2019, doi: 10.1016/j.jpdc.2019.03.003.
3. C. Szegedy et al., "Intriguing properties of neural networks," presented at the ICLR, 2014. Accessed: Aug. 22, 2019. [Online]. Available: http://arxiv.org/abs/1312.6199
4. I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," presented at the ICLR, 2015. Accessed: Aug. 22, 2019. [Online]. Available: http://arxiv.org/abs/1412.6572
5. 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 (AISec), 2017, pp. 15–26. doi: 10.1145/3128572.3140448.
6. A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box Adversarial Attacks with Limited Queries and Information," Apr. 2018. Accessed: Aug. 18, 2019. [Online]. Available: http://arxiv.org/abs/1804.08598

7. C.-C. Tu et al., "AutoZOOM: Autoencoder-based Zeroth Order Optimization Method for Attacking Black-box Neural Networks," 2019. Accessed: Aug. 18, 2019. [Online]. Available: http://arxiv.org/abs/1805.11770
8. S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal Adversarial Perturbations," 2017, p. 9.
9. Y. Li, S. Bai, C. Xie, Z. Liao, X. Shen, and A. Yuille, "Regional Homogeneity: Towards Learning Transferable Universal Adversarial Perturbations Against Defenses," in Computer Vision – ECCV 2020, Cham, 2020, vol. 12356, pp. 795–813.
10. C. Zhang, P. Benz, T. Imtiaz, and I. S. Kweon, "Understanding Adversarial Examples From the Mutual Influence of Images and Perturbations," 2020, p. 10.
11. S. Baluja and I. Fischer, "Learning to Attack: Adversarial Transformation Networks," in The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI) 2018, 2018, p. 9.
12. M. Li, Y. Yang, K. Wei, X. Yang, and H. Huang, "Learning Universal Adversarial Perturbation by Adversarial Example," in Proceedings of the AAAI Conference on Artificial Intelligence, 2022, vol. 36, pp. 1350–1358. doi: 10.1609/aaai.v36i2.20023.
13. L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the Landscape of Spatial Robustness," Sep. 2019. Accessed: Apr. 28, 2022. [Online]. Available: http://arxiv.org/abs/1712.02779
14. R. Alaifari, G. S. Alberti, and T. Gauksson, "ADef: an Iterative Algorithm to Construct Adversarial Deformations," Jan. 2019. Accessed: Apr. 28, 2022. [Online]. Available: http://arxiv.org/abs/1804.07729
15. C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially Transformed Adversarial Examples," Jan. 2018. Accessed: Apr. 28, 2022. [Online]. Available: http://arxiv.org/abs/1801.02612
16. A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," ArXiv160702533 Cs Stat, Jul. 2016, Accessed: Aug. 22, 2019. [Online]. Available: http://arxiv.org/abs/1607.02533
17. M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition," in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16, Vienna, Austria, 2016, pp. 1528–1540.
18. T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial Patch," ArXiv171209665 Cs, Dec. 2017, Accessed: Aug. 22, 2019. [Online]. Available: http://arxiv.org/abs/1712.09665
19. A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing Robust Adversarial Examples," 2018. Accessed: Jul. 28, 2019. [Online]. Available: http://arxiv.org/abs/1707.07397
20. A. Athalye, N. Carlini, and D. Wagner, "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples," Feb. 2018. Accessed: Aug. 18, 2019. [Online]. Available: http://arxiv.org/abs/1802.00420
21. K. Eykholt et al., "Robust Physical-World Attacks on Deep Learning Visual Classification," in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018, pp. 1625–1634. doi: 10.1109/CVPR.2018.00175.
22. D. Wang et al., "FCA: Learning a 3D Full-Coverage Vehicle Camouflage for Multi-View Physical Adversarial Attack," in Proceedings of the AAAI Conference on Artificial Intelligence, 2022, vol. 36, pp. 2414–2422. doi: 10.1609/aaai.v36i2.20141
23. K. Xu et al., "Adversarial T-Shirt! Evading Person Detectors in a Physical World," in Computer Vision – ECCV 2020, Cham, 2020, vol. 12350, pp. 665–681.

24. D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber, "Natural Evolution Strategies," J. Mach. Learn. Res. 15, pp. 949–980, 2014, doi: 10.1109/CEC.2008.4631255.