FUNN: Flexible Unsupervised Neural Network

David Vigouroux Sylvain Picard

IRT Saint Exupery, Toulouse, France
david.vigouroux@irt-saintexupery.com
sylvain.picard@irt-saintexupery.com

Résumé

Les réseaux neuronaux profonds ont démontré une grande précision dans les tâches de vision par ordinateur. Cependant, ils se sont avérés non robustes face aux exemples adversariaux. Une petite perturbation dans l'image peut totalement modifier le résultat d'une classification. Ces dernières années, plusieurs moyens de défense ont été proposés pour tenter de résoudre ce problème dans le cadre de tâches de classification supervisées, sans arriver à des résultats satisfaisants. Dans cet article nous proposons une méthode permettant d'obtenir des features robustes contre les attaques adversariales dans le cadre de tâches d'apprentissage non supervisées. Notre méthode se distingue des solutions existantes par l'apprentissage direct des features robustes sans qu'il soit nécessaire de projeter les exemples adversariaux dans l'espace de distribution des images d'origine. Un premier auto-encodeur, l'attaquant, est chargé de perturber l'image d'entrée afin de tromper un second auto-encodeur, le defenseur, qui lui est chargé de régénérer l'image d'origine. L'attaquant essaie de trouver l'image la moins perturbée sous la contrainte que l'erreur dans la sortie du defenseur soit au moins égale à un seuil. Grâce à cette formation, l'encodeur du defenseur sera robuste contre les attaques adversariales et pourra être utilisé dans différentes tâches comme la classification. En utilisant une architecture de réseau de l'état de l'art, nous démontrons la robustesse des fonctionnalités obtenues grâce à cette méthode dans les tâches de classification.

Abstract

Deep neural networks have shown high accuracy in computer vision tasks. However, they are known to be weak against adversarial examples. A small perturbation in the image can change the classification dramatically. In recent years, several defences methods have been proposed to solve the issue in the context of supervised classification tasks. We propose a method to find robust features against adversarial attacks in the context of unsupervised learning. Our method differs from existing solutions by directly learning the robust features without projecting the adversarial examples in the original distribution space.A first auto-encoder, called attacker, perturbs the input image in order to fool a second auto-encoder, called defender, which tries to regenerate the original image.The goal of the attacker is to perturb images as little as possible while guaranteeing that the reconstructed images will be at a given distance from the original images.After such training, we extract from the defender an encoder that should be robust against adversarial attacks. Using state-of-art network architectures, we demonstrate the robustness of the features obtained by this method in classification tasks.

1 Introduction

Neural networks and especially convolutional neural networks have shown impressive results in many different tasks in computer vision such as object detection, image recognition and segmentation. A downside of these techniques is their lack of robustness [3, 6, 9, 22] : imperceptible perturbations of the input (image, sound, ...) can disturb the networks and drastically change the output. Such perturbed inputs are called *"adversarial examples"* [6] , and it is possible to design algorithms that can generate such examples. Those algorithms are known as *"adversarial attacks"* [6]. We commonly distinguish two types of attacks:

— **white box** attacks, when the network architecture and the weights are known;
— **black box** attacks, when they are not.

This work focus on non-targeted white-box attacks computed using different attack methods.

The field of adversarial generation is an active research field. Various methods have been developed in order to produce adversarial example [25]. In this work, we consider five of the most common attack methods from the literature: the Fast Gradient Sign Method (FGSM) [6], Iterative FGSM [10], Single Pixel attack and LocalSearch [15] and Deepfool [14]. Those methods

are *gradient-based* or *score-based* approaches that try to find minimal perturbations that will change the model prediction.

Different methods have been proposed in order to deal with these adversarial attacks. Some approaches try to project the adversarial examples into the original distribution [12, 18, 19] while others propose to add adversarial examples into the training dataset [6, 14, 22], similarly to data augmentation. Finally, recent methods propose to transform the input data to make it less sensitive to perturbations [8]. All these methods are attack-specific, and thus not efficient against new or different attacks. Recently, new methods have been proposed to learn the attacking and defending concept concurrently in order to be independent of the attacking method. Some of those methods use Generative Adversarial Networks (GANs) to generate adversarial examples as in [11, 18] or Auto-Encoders as in [1, 4, 20].

In this paper, we attempt to learn an encoder with robust features in an unsupervised manner. For this, we train concurrently an *attacker* and a *defender*. The attacker generates perturbed images from the original image in order to fool the defender. The defender try to reconstruct the original image from the perturbed one in order to produce robust features.

2 Related Work

2.1 Attack strategies

Several attack strategies have been proposed, which can be split in two categories: **black-box** and **white-box** attacks. White-box attacks have access to the weights and gradients of the classifier while, black-box strategies only have access to the predictions of the network.In this work, we concentrate on two kind of white-box attacks: gradient-based and score-based. This section describes the method we consider.

Fast Gradient Sign Method (FGSM) [6] Given an image X and its corresponding label Y, the FGSM attack sets the perturbation δ to:

$$\delta(X, Y) = \epsilon \cdot sign(\nabla_X L(X, Y)) \tag{1}$$

where ϵ is a small number and ∇ is the gradient of the loss function L with respect to the input X. For each pixel, FGSM slightly increases or decreases its value depending on the sign of the gradient with respect to the pixel.

DeepFool [14] The DeepFool method performs an iterative attack using a linear approximation in order to find the shortest distance between the original input and the decision boundary of adversarial examples. If f is a binary differentiable classifier, an iterative method is used to approximate the perturbation. The minimal perturbation is computed as:

$$\begin{aligned} \arg\min_{\eta_i} \quad & \|\eta_i\|_2 \\ s.t. \quad & f(x_i) + \nabla f(x_i)^T \eta_i = 0 \end{aligned} \tag{2}$$

η_i is the noise level and x_i is the perturbed image at iteration i, f is the binary classifier.

The method can be extended to multi-class classifiers by finding the closest hyperplanes. DeepFool provides less perturbation and reduces intensity compared to FGSM.

Single Pixel [21] Single Pixel is an iterative methods that tries to create an adversarial example by setting the value of a **single** pixel in the image to the minimum or maximum value of any pixel in the image. The method randomly selects a set of candidate pixel to modify and then iterate over them until an adversarial example is generated. The method can fail if none of the candidate pixel can be modified to generate an adversarial example.

Local Search [15] This local search procedure tries to find a pixel (or a group of pixels) that is critical for the classifier robustness and then modify its value to generate adversarial images. The Local Search algorithm finds pixel locations to modify and then applies a fixed transformation function to the selected pixels in order to generate an adversarial image.

2.2 Defence strategies

Several studies have assumed that the lack of robustness comes from the fact that adversarial examples are outside of the dataset's distribution, near the border of decision as shown in [5, 24]. Multiple defence methods have been proposed to increase the robustness of deep neural networks against adversarial attacks. This section describes the most common strategies.

Adversarial training Adversarial training is an intuitive method that consists in augmenting the training dataset with adversarial examples [23]. While this method is efficient to increase robustness against adversarial examples similar to the ones added to the training dataset, it is attack-specific and thus poorly generalize to adversarial examples generated differently from the ones in the training dataset.

Defensive distillation Defensive distillation [16] is a method that train the classifier in two steps using a variation of the distillation [7] method. This creates a smoother network, thus reducing the amplitude of gradients around input points, and consequently increasing the robustness

against adversarial examples. However, this method has proven to be inefficient against recent black-box attacks [2].

Adversarial detectors Another method of defence is to detect adversarial examples [13] and exclude them. For each attack method considered, a classifier (detector) is trained to tell whether an input is normal or adversarial. The detector is directly trained on both normal and adversarial examples. This method shows good performance when the training and testing attack examples are generated from the same process and the perturbations are large enough, but does not generalize well across different attack parameters and attack generation processes.

MagNet MagNet [12] is a method based on adversarial detector strategy. It trains a reformer network (which is an auto-encoder or a collection of auto-encoders) to move adversarial examples closer to the manifold of legitimate, or natural, examples. It is an effective strategy against grey-box attacks where the attacker is aware of the network architecture and defences, but does not know its weights.

3 Proposed Method

3.1 Motivation

The previously described defence methods provide some intuition on what can make a network robust to adversarial examples. Our strategy differs from these by performing adversarial training with data generated by a separated network during training. Unlike adversarial detectors strategies, our generating network is only used during training, and not when doing inference. This paper focus only on unsupervised learning strategies, which is why we use auto-encoders. An auto-encoder is a deep neural network composed by an encoder, responsible for extracting deep features from input images, and a decoder, responsible for reconstructing images from the extracted features. During training, the network is optimized in order to generate images similar to input images by minimizing the distance between the original and generated images.

We propose a defence strategy to increase robustness against white box attacks using Euclidean norm (L_2) as training objective. We then demonstrate its efficiency against attacks that use L_2 or infinity norm (L_∞). The approach is to train two auto-encoders, an **attacker** and a **defender**. The attacker generates adversarial examples by perturbing the original image while the defender tries to reconstruct the original image from adversarial examples given by the attacker. The two models are trained concurrently and adapt themselves during training until convergence much like how Generative Adversarial Networks (GAN) are trained. At the end of the training, we expect that the encoder of the defender will be able to extract deep features which are robust against the perturbations generated by the attacker. We expect this training method to generate deep features representing a larger distribution of input images than the original one, and thus increase the robustness of the defending auto-encoder.

The attacker model is similar to a style transfer model, it must reconstruct the input image while adding noise in order to induce errors in the defender output. Thus the attacker has two adversarial training objectives:

— generate an image close to the input image;
— add perturbation to the generated image in order to disrupt the defender with a minimum noise level β.

The objective of the defender is to produce an image as close as possible to the original image (input of the attacker) using the image generated by the attacker.

The distance between the original and reconstructed image, L_α, is the loss function of the problem, and can be any distance. In our experiences, we choose the Euclidean distance. The corresponding optimization problem is described by equation (3–5). This optimization problem represents a non zero-sum game.

$$\min_{\theta_{att}} \quad E_{x\sim\chi}\left(L_\alpha\left(Att\left(x,\theta_{att}\right),x\right)\right) \tag{3}$$

$$\min_{\theta_{def}} \quad E_{x\sim\chi}(L_\alpha(x, Def(Att(x,\theta_{att}),\theta_{def}))) \tag{4}$$

$$E_{x\sim\chi}(L_\alpha(x, Def(Att(x,\theta_{att}),\theta_{def}))) \geq \beta \tag{5}$$

where:

— χ is the ensemble of distribution examples,
— θ_{att} is the weight of the *attacker* auto-encoder,
— θ_{def} is the weight of the *defender* auto-encoder,
— $L_\alpha : \chi,\chi \rightarrow R$ is the loss function,
— $Att(x,\theta_{att}) : \chi \rightarrow \chi$ is a function with parameters θ_{att},
— $Def(x,\theta_{def}) : \chi \rightarrow \chi$ is a function with parameters θ_{def}.

Objective (3) is to minimize the distance between the original image and the perturbed one (attacker). Objective (4) is to minimize the distance between the original image and the reconstructed image (defender). Constraint (5) enforces a given minimum distance between the original image and the reconstructed one.

To solve this under constraint optimization problem, we choose to relax Constraint (5) by penalizing it in the objective function:

$$\min_{\theta_{att}} E_{x\sim\chi}(L_\alpha(Att(x,\theta_{att}),x)) + \gamma * \max(E_{x\sim\chi}(L_\alpha(x, Def(Att(x,\theta_{att}),\theta_{def}))) - \beta, 0.0)$$

where γ is an hyper-parameter that controls noise generation and must be large enough to satisfy Constraint (5).

3.2 Model and training procedure

The attacker is implemented as an auto-encoder where an uniform random vector is concatenated to the encoder's features and then passed through the generator. Adding an uniform random vector allows the attacker to generate noise without modifying the real features. The defender is implemented as a standard auto-encoder.

We use a two-stage approach, with an *initialization stage* that trains the encoder of attacker and the decoder of the defender, and an *optimization stage* that deals with the real optimization problem.

During the initialization stage, both auto-encoders are trained to generate realistic images, without constraints (Equation (3, 4)). This first phase facilitates the training procedure and guarantees that the optimization stage goes in the correct direction.

During the optimization stage, the weights of the attacker's encoder and the defender's generator are not updated anymore. This choice is justified since:

— The purpose of the attacker is to produce adversarial examples from given non-perturbed examples. After the initialization stage, the encoder of the attacker is already able to compress input images into their features and does not need to be trained anymore. During the optimization stage the attacker can be seen as an adversarial example generator taking as input previously learned features concatenated with random noise.

— The goal of the defender is to be robust against adversarial attacks. After the initialization stage, the generator of the defender is able to reproduce correct images from good features. We want the features of adversarial examples to be identical to the features of non-perturbed examples. By fixing the generator weights, we want the defender to extract features from perturbed images that are very close to the features of non-perturbed images.

3.3 Testing procedure

Since auto-encoders are architectures built for unsupervised learning, no classifiers are used during training. After training, in order to be able to use adversarial attacks and estimate the robustness of the network, we train a classifier using the features produced by the encoder of the defender on images coming from the original dataset. The robustness of this classifier is evaluated against the following attacks: DeepFool (L_2-norm and L_∞-norm), Single Pixel, LocalSearch and FGSM.

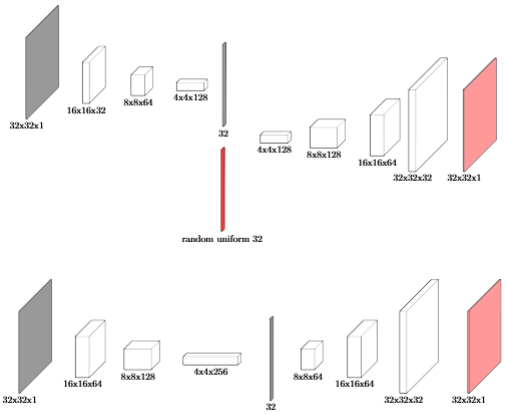

Figure 1 – Network architectures.

We evaluate robustness using two criterion:

— the *attack success rate*, i.e., the percentage of times the attack was able to fool the classifier;

— the *noise level* of adversarial examples defined as the L_2-norm between the original image and the generated adversarial example.

4 MNIST Experiments

4.1 Configuration

We evaluate our method on the MNIST dataset with the networks shown in Figure 1.

The neural networks use convolutionnal layers with kernel size 3x3, strides 2x2 and ReLU activation functions except for the last layer. The last layers have $(1+\tanh)/2$ as activation function. We use a batch size of 128 and Adam optimizers with learning rates described in Table 1.

	Initialisation	Optimisation
Defender	5.10^{-4}	10^{-5}
Attacker	10^{-3}	10^{-5}

Table 1 – Learning rates for the Adam optimizer.

We use $\gamma = 5.0$, $\beta = 0.01$ and the L_2 Euclidean norm as the loss function. We run the initialization stage for 9 epochs and the optimization stage for 31 epochs.

The classifier is trained during 20 epochs using an Adam optimizer with a learning rate of 10^{-3}, and a softmax-cross-entropy loss. The classifier uses the defender's encoder (whose weights are fixed) and a hidden dense layer of 64 units with ReLU activation functions.

The network architectures and hyper parameters (γ and β) have not been optimized for this task. However, several learning rates have been tested to achieve the

presented results: a bad choice of learning rate can create a non-robust network.

4.2 Results

Adversarial attacks tools Tools have been implemented to facilitate adversarial attacks implementation such as Cleverhans [**?**] or Foolbox [17]. Those tools are open-source frameworks that propose state-of-art algorithms used to generate adversarial examples on any model.

In order to compare our method against a traditional network, we trained a classic auto-encoder with an architecture similar to the defender architecture. We then train a classifier as previously described on the defender's encoder network. After 20 epochs, this classifier achieves an accuracy close to 97% on the MNIST test set. We then attack this classifier with several methods on the whole test set using Foolbox [17].

The evaluation is only done on the first 1000 images of the dataset due to computation time. Results are reported in Table 2

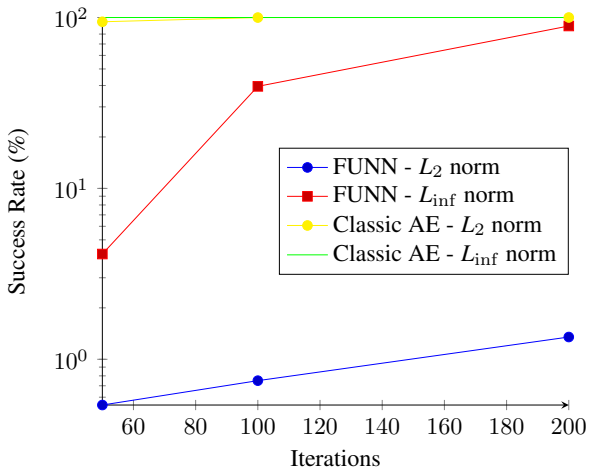

Figure 2 – DeepFool attacks

4.3 Analysis

DeepFool with L_2-norm In Table 2, we see that only 0.5% of attack succeed when DeepFool is used with a maximum of 50 iterations, generating a mean noise level around 0.0075. This is expected as, during learning, the attacker tries to find noise level greater than $\beta = 0.01$ (in fact, very close to β as explained above) which can fool the defender. If the defender manages to defend against this level of noise, we expect it to be robust against levels of noise lower than this threshold. The classifier seems to be robust against L_2 DeepFool attacks with at most 50 iterations. When we increase the DeepFool maximum number of iterations, the success rate of attacks slowly increases. With 200 iterations, 1.35% of attack succeeds, and with 1000 iterations, the percentage of successful attacks starts to becomes significant (15.25%), but the noise level increases significantly at the same time to reach 0.73 which is far from the threshold value of 0.01 used in training. One may think that increasing the threshold β during training would increase the robustness of our method, but using the MNIST dataset, it is impossible because doing so, the attacker would generate images that are almost entirely black. Indeed, the average L_2 distance between an MNIST image and a black image is around 0.07, it is thus not possible to increase the threshold that much.

DeepFool with L_∞-norm With the default maximum number of steps for DeepFool (50), the classifier seems to be robust (4.13% of success rate) for the L_∞ norm. However, the classifier quickly fails as the number of steps increase, with a 89.20% success rate of attack for 200 steps.Usually, such number of iteration steps are not tested in the literature, but this shows that even if some networks seem to be robust against few iterations of DeepFool, larger number of iterations often manage to fool most of these networks. However, even with 50 steps, the noise level is already quite high (1.13), which is why it does not necessarily mean that the network is not robust against this attack.

Single Pixel and LocalSearch Even if the attack success rate is relatively low (4.72%), this level of success rate is higher than what we may expect since the L_2 distances between input images and adversarial examples generated by single pixel are quite small. For our methods, this attack looks more powerful than DeepFool which is designed precisely to minimize the L_2-norm.

Gradient Methods FGSM and Gradient have a high attack success rate, near 100% for Gradient Sign and Gradient. This is expected since the defender is not trained against the high levels of noise generated by all these methods.

MagNet Comparaison MagNet [12] and our method show similar performances. However, MagNet uses an additional network in order to detect adversarial examples upstream to the main network. This detector network is also used during inference, inducing more computation and thus slowing down the inference process. Our method use two auto encoders during the training process, but only the encoder of the defender is used during inference, coupled with, e.g., a small classifier for computer vision tasks such as classification. The global architecture of the final network is thus much smaller than the one used during

Attack Name	Classic Auto Encoder Success Rate	Classic Auto Encoder Noise Level	MagNet Success Rate	Our Method Success Rate	Our Method Noise Level
DeepFool (max-iter=50 – L_2)	94,41%	1,25	—	**0.54%**	**0.0075**
DeepFool (max-iter=50 – L_∞)	100%	1,81	**0.6%**	4.13%	**1.13**
FGSM (ϵ=0.005)	0.41%	0.07	3.2%	**1.61%**	**0.04**
Iterative (ϵ=0.005)	1.04%	0.08	4.8%	**1.25%**	**0.04**
Single Pixel	9.17%	1.0	—	**4.72%**	**1.0**
LocalSearch	44.41%	5.22	—	**27.51%**	**9.71**

Table 2 – Attack success rates and noise levels for various attacks.

training, and the computation time required for inference is reduced without loss of robustness.

5 Conclusion

Without adding extra computational complexity, such as image filtering or adversarial image detection, we demonstrated that a classical network can be robust against L_2 attacks by using an appropriate unsupervised learning procedure. This procedure shows promising results for learning robust unsupervised networks. However, even if it was expected due to the defence strategy, the trained network is not robust against all types of attacks, in particular it is weak against gradient attacks. Our future work will focus on creating a more general defence learning procedure, by using different distance methods. We also plan to test this method on more complex datasets, such as CIFAR-10 and ImageNet, with different network architectures.

References

[1] Baluja, Shumeet and Ian Fischer: *Learning to Attack: Adversarial Transformation Networks*. page 9.

[2] Carlini, Nicholas and David Wagner: *Defensive Distillation is Not Robust to Adversarial Examples*. arXiv:1607.04311 [cs], July 2016. http://arxiv.org/abs/1607.04311, visited on 2019-05-24, arXiv: 1607.04311.

[3] Chakraborty, Anirban, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay: *Adversarial Attacks and Defences: A Survey*. arXiv:1810.00069 [cs, stat], September 2018. http://arxiv.org/abs/1810.00069, visited on 2019-06-06, arXiv: 1810.00069.

[4] Folz, Joachim, Sebastian Palacio, Joern Hees, Damian Borth, and Andreas Dengel: *Adversarial Defense based on Structure-to-Signal Autoencoders*. arXiv:1803.07994 [cs, stat], March 2018. http://arxiv.org/abs/1803.07994, visited on 2019-05-24, arXiv: 1803.07994.

[5] Gilmer, Justin, Luke Metz, Fartash Faghri, Samuel S. Schoenholz, Maithra Raghu, Martin Wattenberg, and Ian Goodfellow: *Adversarial Spheres*. http://arxiv.org/abs/1801.02774, visited on 2019-05-24, arXiv: 1801.02774, January 2018.

[6] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy: *Explaining and Harnessing Adversarial Examples*. arXiv:1412.6572 [cs, stat], December 2014. http://arxiv.org/abs/1412.6572, visited on 2019-05-24, arXiv: 1412.6572.

[7] Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean: *Distilling the Knowledge in a Neural Network*. arXiv:1503.02531 [cs, stat], March 2015. http://arxiv.org/abs/1503.02531, visited on 2019-05-24, arXiv: 1503.02531.

[8] Kabilan, Vishaal Munusamy, Brandon Morris, and Anh Nguyen: *VectorDefense: Vectorization as a Defense to Adversarial Examples*. arXiv:1804.08529 [cs], April 2018. http://arxiv.org/abs/1804.08529, visited on 2019-05-24, arXiv: 1804.08529.

[9] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio: *Adversarial examples in the physical world*. arXiv:1607.02533 [cs, stat], July 2016. http://arxiv.org/abs/1607.02533, visited on 2019-05-24, arXiv: 1607.02533.

[10] Kurakin, Alexey, Ian Goodfellow, and Samy Bengio: *Adversarial Machine Learning at Scale*. arXiv:1611.01236 [cs, stat], November 2016. http://arxiv.org/abs/1611.01236, visited on 2019-05-24, arXiv: 1611.01236.

[11] Lee, Hyeungill, Sungyeob Han, and Jungwoo Lee: *Generative Adversarial Trainer: Defense to Adversarial Perturbations with GAN*. arXiv:1705.03387 [cs, stat], May 2017. http://arxiv.org/abs/1705.03387, visited on 2019-05-24, arXiv: 1705.03387.

[12] Meng, Dongyu and Hao Chen: *MagNet: A Two-Pronged Defense against Adversarial Examples*. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* -

CCS '17, pages 135–147, Dallas, Texas, USA, 2017. ACM Press, ISBN 978-1-4503-4946-8. http://dl.acm.org/citation.cfm?doid=3133956.3134057, visited on 2019-05-24.

[13] Metzen, Jan Hendrik, Tim Genewein, Volker Fischer, and Bastian Bischoff: *On Detecting Adversarial Perturbations*. arXiv:1702.04267 [cs, stat], February 2017. http://arxiv.org/abs/1702.04267, visited on 2019-05-24, arXiv: 1702.04267.

[14] Moosavi-Dezfooli, Seyed Mohsen, Alhussein Fawzi, and Pascal Frossard: *DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks*. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, Las Vegas, NV, USA, June 2016. IEEE, ISBN 978-1-4673-8851-1. http://ieeexplore.ieee.org/document/7780651/, visited on 2019-05-24.

[15] Narodytska, Nina and Shiva Prasad Kasiviswanathan: *Simple Black-Box Adversarial Perturbations for Deep Networks*. arXiv:1612.06299 [cs, stat], December 2016. http://arxiv.org/abs/1612.06299, visited on 2019-05-24, arXiv: 1612.06299.

[16] Papernot, N., P. McDaniel, X. Wu, S. Jha, and A. Swami: *Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks*. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597, May 2016.

[17] Rauber, Jonas, Wieland Brendel, and Matthias Bethge: *Foolbox: A Python toolbox to benchmark the robustness of machine learning models*. arXiv:1707.04131 [cs, stat], July 2017. http://arxiv.org/abs/1707.04131, visited on 2019-05-24, arXiv: 1707.04131.

[18] Samangouei, Pouya, Maya Kabkab, and Rama Chellappa: *Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models*. arXiv:1805.06605 [cs, stat], May 2018. http://arxiv.org/abs/1805.06605, visited on 2019-05-24, arXiv: 1805.06605.

[19] Santhanam, Gokula Krishnan and Paulina Grnarova: *Defending Against Adversarial Attacks by Leveraging an Entire GAN*. arXiv:1805.10652 [cs, stat], May 2018. http://arxiv.org/abs/1805.10652, visited on 2019-05-24, arXiv: 1805.10652.

[20] Srinivasan, Vignesh, Arturo Marban, Klaus Robert Müller, Wojciech Samek, and Shinichi Nakajima: *Counterstrike: Defending Deep Learning Architectures Against Adversarial Samples by Langevin Dynamics with Supervised Denoising Autoencoder*. arXiv:1805.12017 [cs, stat], May 2018. http://arxiv.org/abs/1805.12017, visited on 2019-05-24, arXiv: 1805.12017.

[21] Su, J., D. V. Vargas, and K. Sakurai: *One Pixel Attack for Fooling Deep Neural Networks*. IEEE Transactions on Evolutionary Computation, pages 1–1, 2019, ISSN 1089-778X.

[22] Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus: *Intriguing properties of neural networks*. arXiv:1312.6199 [cs], December 2013. http://arxiv.org/abs/1312.6199, visited on 2019-05-24, arXiv: 1312.6199.

[23] Tramèr, Florian, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel: *Ensemble Adversarial Training: Attacks and Defenses*. arXiv:1705.07204 [cs, stat], May 2017. http://arxiv.org/abs/1705.07204, visited on 2019-05-24, arXiv: 1705.07204.

[24] Tramèr, Florian, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel: *The Space of Transferable Adversarial Examples*. arXiv:1704.03453 [cs, stat], April 2017. http://arxiv.org/abs/1704.03453, visited on 2019-05-24, arXiv: 1704.03453.

[25] Yuan, X., P. He, Q. Zhu, and X. Li: *Adversarial Examples: Attacks and Defenses for Deep Learning*. IEEE Transactions on Neural Networks and Learning Systems, pages 1–20, 2019, ISSN 2162-237X.