Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 140 (2018) 152–161

Procedia Computer Science

www.elsevier.com/locate/procedia

Complex Adaptive Systems Conference with Theme: Cyber Physical Systems and Deep Learning, CAS 2018, 5 November – 7 November 2018, Chicago, Illinois, USA

Adversarial Attacks and Defenses Against Deep Neural Networks: A Survey

Mesut Ozdag

University of Central Florida, 4000 Central Florida Blvd, Orlando, FL, 32816 USA

Abstract

Deep learning has achieved great successes in various types of applications over recent years. On the other hand, it has been found that deep neural networks (DNNs) can be easily fooled by adversarial input samples. This vulnerability raises major concerns in security-sensitive environments. Therefore, research in attacking and defending DNNs with adversarial examples has drawn great attention. The goal of this paper is to review the types of adversarial attacks and defenses, describe the state-of-the-art methods for each group, and compare their results. In addition, we present some of the top-scored competition submissions for Neural Information Processing Systems (NIPS) in 2017, their solution models, and demonstrate their results. This adversary competition was organized by Google Brain for research scientists to come up with novel solutions that generate adversarial examples and also defend against them. Its contribution is significant on this era of machine learning and DNNs.

Keywords: deep learning; deep neural network; adversarial examples; security

1. Introduction

In recent research studies, machine learning models including DNNs perform very well in pattern-recognition tasks, such as image classification problems (i.e. ImageNet). Furthermore, human-level performance has been achieved by DNNs (i.e. DeepFace). However, it has been noticed that it is possible to lead any machine learning system to misclassify images with high confidence by adding some imperceptible perturbation on them. This modification is almost indistinguishable to a human observer. Therefore, adversarial images pose a threat due to the fact that machine learning systems could be exposed to an attack performed by these images. Naturally, security and privacy in the machine learning applications have seen a surge in interest for the last few years. The vulnerability of deep learning

10.1016/j.procs.2018.10.315

systems to adversarial samples is a key source of concern in many applications, such as ATMs [1, 2], surveillance systems [2, 3, 4, 5], face recognition tasks [3], speech recognition problems [6], Apple Siri [7], Amazon Alexa [8], Microsoft Cortana [9], and autonomous driving [10, 11, 12]. In addition, the adversarial robustness of neural networks has been studied in order to improve resistance of these networks to a wide range of adversarial attacks [13, 14]. Therefore, these methods introduce concrete security for training and attacking neural networks that would be protected against any adversary. They present a great insight into the guarantees they provide. It is also demonstrated that these methods are resistant to a wide range of attacks as opposed to defending only against some certain attacks.

2. Adversarial Attacks

The typical purpose of an adversarial attack is to add a natural perturbation (i.e. fast gradient sign, fog, and sunlight etc.) on an image so that the target model misclassifies the sample, but it is still correctly classified by the human eye. Three main types of studies in adversarial attacks have been mainstream in this area:

- ***Non-targeted adversarial attacks*** are developed to fool a machine learning classifier by modifying source image. The neural network does not return a certain class as opposed to targeted attacks. The output can be a random class excluding the original one.
- ***Targeted adversarial attacks*** are designed to misclassify an image as a specified target class by modifying source image. The output of this neural network is only one certain class. Impersonation can be an example for this type of attacks because an adversarial image can disguise a face as an admin user [3].
- ***Defenses against adversarial attacks*** are aimed to build such a robust classifier so that it correctly identifies adversarial images.

Recent research studies for adversary generation introduce some new methods. In this review paper, the most well-known and state-of-the-art attacks and defenses are presented:

2.1. Fast Gradient Sign Method (FGSM)

One practical way of generating adversarial examples are suggested by their linear behavior [15]. It is observed that the models which are easy to perturb are the ones with linearity. Therefore, adversarial training is processed in a fast and simple way with this method. The goal of FGSM is to fool the classification (i.e. GoogLeNEt) of the image by adding an unnoticeably small vector. The elements of the vector are obtained by taking the sign of the gradient of a loss function associated with that feature vector.

$$\eta = \varepsilon * \text{sign}(\nabla_x J(\theta, x, y)) . \quad (1)$$

Let $J(\theta, x, y)$ be the cost function for training a neural network, η the perturbation, θ the parameters, x the input, and y be the targets to the model. The cost function can be obtained as linear which causes an optimal perturbation as in (1). It is demonstrated that this method reliably misclassifies the model inputs.

An example of adversary can be a sample input data which is modified intentionally in order to cause a machine learning system to incorrectly classify it. A gradient descent is an optimization function which can be used to find the local minimum for a differentiable function. First off, we describe current adversarial samples for linear structures. The individual input feature has limited precision in many cases, such as the digital images with 8 bits per pixel can dismiss all the information below *1/255* of the dynamic range. Since there is limited precision for any feature, it is illogical for the classifier to return a different class for an input x than to an adversarial input $x' = x + \eta$ where η is the perturbation of each element less than the precision of features. Technically, for the cases which result from well-separated outputs, the classifier is expected to give the same output to x and x' as long as $||\eta||_\infty < \varepsilon$ where ε is small enough so that it can be dismissed by the sensor or data storage apparatus related to the case.

Let a weight vector be w, an adversarial sample x, and consider the dot product between them:

$$w^T x' = w^T x + w^T \eta . \quad (2)$$

In this case, the activation increases linearly by $w^T\eta$ due to the adversarial perturbation. This demonstrates that a basic linear model can provide adversarial samples if there is enough dimension of its inputs.

Next, we present the main idea behind FGSM which is to add some weak noise on every step of optimization that moves towards the expected class or away from the correct one. To keep the attack subtle, the amplitude (intensity of pixel channel) of noise has to be limited. An example can be shenanigans that are investigated by a human observer. This case is very similar to the optimization problem in which we optimize the noise for the sake of error. This error can be easily calculated, and the gradient can be computed. Another but more practical case can be where we attack a complete black box model. In this case, a picture would be taken by the model and only a class would be returned. The same procedure follows: The noise is generated and then it is added to the image and sent to the classifier. This process is repeated until the neural network incorrectly classifies it. Next, no matter what the limit of amplitude of the noise is, it never returns the true class. Lastly, the weakest possible noise which returns the same result is investigated. A simple binary search overcomes this task. As a result, when we consider different cross-sections of the image space, FGSM moves the direction towards the link between the misclassified result and correct class. The first and foremost interesting observation is that when the estimated class changes by following gradient, it can be confidently said that the attack achieves good results. This approach is very simple and effective. It has the strength of fooling pretty much any machine learning algorithm in cases when there is no defense against them.

A variation of FGSM has also been studied recently [16]. It is demonstrated that one-step attack is easy to transfer, whereas it is also easy to defend. Therefore, adversarial images are created by applying momentum to FGSM in a more iterative approach:

$$g_{t+1} = \mu\, g_t + (\nabla_x J_\theta(x'_t, l) / \|\nabla_x J_\theta(x'_t, l)\|) ,$$
$$x'_{t+1} = x'_t + \varepsilon * \mathrm{sign}(g_{t+1}) . \quad (3)$$

The gradient and adversarial samples are calculated in the given equation (3) according to the FGSM with momentum. The effectiveness is boosted by this method and it took the first place in the competition for targeted and non-targeted adversarial attacks held by NIPS in 2017.

2.2. Projected Gradient Descent (PGD)

Multi-step variant of FGSM, which is actually PGD on the negative loss function, is a more powerful adversary than FGSM itself [14, 17]:

$$x^{t+1} = \pi_{x+S}\,(x^t + \varepsilon * \mathrm{sign}(\nabla_x J(\theta, x, y))) . \quad (4)$$

FGSM does not increase robustness. More specifically, the network might overfit to the adversarial examples generated by FGSM. This is called label leaking. These networks do not show any robustness against PGD adversaries. Therefore, PGD is the strongest attack using the local first-order information about the network [14].

Moreover, FGSM is a one-step approach and there exists some shortcomings for this method. However, PGD is very useful because it works very well for the large scale constrained optimization. Generally, we restart the PGD technique from many points in the l_∞ balls around some data points from a given evaluation sets to explore a large part of the loss landscape. In the loss function of model parameters, the value of all the local maxima is similar. These values are the same for both type of networks; normal networks and adversarially trained networks. Robustness against the PGD adversary gives robustness against all the first-order adversaries that means it is the attack that would depend upon the first-order information only. When we use gradients of the loss functions in the case of adversary with regard to different input values, it is concluded that this might not find any significantly better local maxima than PGD.

For the results demonstrated in Table.1, a network consisted of a convolutional layer with 32 filters, one 2 x 2 max-pooling, another convolutional layer with 64 filters followed by another 2 x 2 max-pooling, and lastly one fully connected layer of size 1024 is used (Source: Madry 2017). It is observed that PGD is the best attack. As for Table.2, ResNet and its ten times wider variant are used to train the network. Again, PGD is observed as the most effective attack.

Table 1. Performance results of the adversarially trained model against each adversary for ε = 0.3 using MNIST. (A: the network itself - white-box attack).

Method	Steps	Restarts	Source	Accuracy
Natural	-	-	-	98.8%
FGSM	-	-	A	95.6%
PGD	40	1	A	93.2%
PGD	100	1	A	91.8%
PGD	40	20	A	90.4%
PGD	100	20	A	**89.3%**
Targeted	40	1	A	92.7%
CW	40	1	A	94.0%
CW+	40	1	A	93.9%

Table 2. Performance results of the adversarially trained model against each adversary for ε = 8 using CIFAR10. (A: the network itself - white-box attack).

Method	Steps	Source	Accuracy
Natural	-	-	87.3%
FGSM	-	A	56.1%
PGD	7	A	50.0%
PGD	20	A	**45.8%**
PGD	30	A	46.8%

If a network is trained to be strong against PGD adversaries, it can also be secure against a wide range of different attacks as well. With the help of first-order techniques, a large majority of optimization problems in machine learning is solved. Most common methods for training deep learning models are the variants of Stochastic Gradient Descent (SGD). SGD provides the optimal parameters of neurons in each and every layer.

This attack has the property of not having a direct access to the target network. It is also called zero-order attacks which have no direct access to the classifier for the adversary. It would only be able to compute the selected examples without the gradient feedback.

All in all, the inner optimization problem can be solved by the application of PGD in a successful manner. To train an adversarial robust network, the outer optimization problem of the saddle point formulation has to be solved. In other words, model parameters have to be found in order to minimize the adversarial loss, the value of the inner maximization problem.

2.3. Basic Iterative Method (BIM)

BIM is an iterative version of FGSM. That is, adversarial noise is applied many times iteratively instead of applying it only once [17]. This method is convenient since extra control is taken over the attack. For instance, the loop can be terminated when there is misclassification. BIM was demonstrated to be more effective than the FGSM attack on ImageNet dataset [17]. In addition, adversarial training can increase the robustness of neural networks for one-step attack (e.g., FGSM) but might not work under iterative attacks (e.g., BIM). Furthermore, pixel values are clipped to disregard big changes on each pixel in every iteration:

$$\mathrm{Clip}_{x,\varsigma}(x') = \min(255, x+\varsigma, \max(0, x-\varsigma, x')), \tag{5}$$

where $Clip_{x,\varsigma}(x')$ is clipping value in every iteration limited by ς. Multiple iterations are used to create the adversarial images:

$$x_0 = x,$$
$$x_{n+1} = \mathrm{Clip}_{x,\varsigma}(x_n + \varepsilon * \mathrm{sign}(\nabla_x J(x_n, y))). \quad (6)$$

For the BIM method, FGSM is applied multiple times with the help of small step size. For the process of adversarial training, the core idea is to inject the adversarial examples into the training set, continually creating new adversarial samples at each step of training. Initially, small models were used for developing adversarial training; however, there was no batch normalization during that process. Batch normalization is found very important after scaling the adversarial training to ImageNet.

Generally, a loss function is used in order to obtain independent control of the number and relative weight of adversarial examples in every batch. In the experiment, it is found that adversarial training uses different types of one-step methods. It is concluded that the adversarial training using any type of one-step method increases robustness to all types of one-step adversarial examples. There is a difference between accuracy on clean and adversarial examples that would vary depending on the combination of methods used for the training and evaluation.

By adding or removing some regularization methods (dropout, label smoothing and weight decay), it is observed that clean or adversarial examples can be affected in different ways. For instance, adding a regularizer to adversarial training may increase accuracy while it decreases the accuracy on clean images.

Most of the results show that the adversarial training should be applied in two scenarios. The first scenario is the case when a model is overfitting and a regularizer is a requirement. The second scenario is the case when there is a security concern against adversarial examples. This case suggests that adversarial training is the method that gives the most security of any known defending model at the cost of losing only a little amount of accuracy.

2.4. Carlini-Wagner (CW)

Carlini-Wagner introduces an approach that is able to overcome defensive distillation. It presents three powerful attacks against this recent promising defense algorithm in [18]: L_2 attack, L_0 attack, and L_∞ attack [13]. These three attacks demonstrate 100% success rate in finding adversarial samples on defensively distilled and undistilled networks.

Defensive distillation was used to improve the robustness of an arbitrary neural network model against adversarial samples [18]. However, CW proposes L_2, L_0, and L_∞ algorithms that generate adversarial samples causing misclassification with the same label; therefore, they beat defensive distillation with 100% probability. It is processed in four steps:

- Train a neural network,
- Calculate the softmax in a smoother way and compute the soft training labels by applying the network to each case in the training data set,
- Then train the distilled network with soft training labels,
- Lastly, the distilled network is tested on the new input to classify.

One of the main contributions of the work CW presents is that L_0 attack is the first approach that results in some specified misclassification on ImageNet. Secondly, after all these three attacks are applied to defensive distillation, it is proved that there is little difference in security between undistilled and distilled networks.

L_0 attack measures the distance based on the number of pixels that has been changed while the L_2 attack is used to measure the distance using the root-mean square between x and x'; where x is an input and x' is a new input found that is similar to x but classified as a targeted t. On the other hand, L_∞ is used to measure the maximum changes to any of the coordinates.

It is observed that L_2 attack has low distortion while L_0 attack is non-differentiable and bad-suited for gradient descent. In addition, L_∞ is not fully differentiable and it is also seen that gradient descent does not give good results for it.

2.5. Jacobian-based Saliency Map Approach (JSMA)

JSMA [19] iteratively perturbs features of input data that has large adversarial saliency scores. This score describes the adversary meaning by taking a sample away from its source class towards a certain target class. All other methods use output variations to find related input perturbations for adversarial samples, whereas JSMA constructs a mapping from input perturbations to output variations. This is called forward derivative, a matrix defined as the Jacobian of the function learned by the DNN:

$$\nabla J(X) = \partial J(X) / \partial X = [\partial J_j(X) / \partial x_i]_{i \in 1..M,\, j \in 1..N} \tag{7}$$

In the given formulation (7), *X* is the sample and *J(X)* is the classified result by the network defined in [19]; where *M* is the input dimension or the number of neurons on input layer and *N* is the output dimension or the number of neurons on output layer. This is actually the Jacobian of the function regarding to what is learned by training the neural network. The smallest possible perturbation is crafted in order to fool the neural network. Then two adversarial saliency maps are introduced to manipulate pixels/features in each iteration. 97.10% success rate is achieved to create an adversarial sample that can be misclassified with perturbing only 4.02% of the input features per image.

Generally, saliency map is known as visualization tool to explore adversarial examples. This map symbolizes which input features should perturb to affect the changes in the network. As Papernot et al. describes, it is a repetitive method for targeted misclassification. In this classification method, the derivative of DNN is exploited and it tries to find the adversarial perturbation that allows to misclassify the models into the required goals. Let us take the input *y* and a neural network *M* and the output of the class *i* is noted by the *M(y)* for class *i*. Let us also specify the required class *p*. *M(y)* for *p* would be increased while probabilities *M(y)* for class *i* of all other classes that are not *p* will decrease until the maximum of *M(y)* is obtained. After defining the adversarial saliency map for the input feature, the pair of features is considered to maximize the saliency map between both of them and perturbs each feature by some constant offset. This process is repeated until the target misclassification is reached.

3. NIPS 2017 Adversarial Learning Competition Results

Defenses and attacks for classification task in this competition are applied to ImageNet dataset. Therefore, participants are expected to develop robust ImageNet classifiers and generate adversarial examples on them.

The evaluation is made on two datasets which are ImageNet-compatible: DEV dataset and TEST dataset. DEV dataset is used for experiments and development throughout the competition time and it contains 1000 images. TEST dataset is kept secret, and it is released and used after the competition in order to score final submissions. For the final scoring, attack receives one point when an adversarial image is misclassified by defense:

$$\text{score}_{\text{attack}} = \sum_{\text{defense} \in D} \sum_{k=1}^{N} [\text{defense}(\text{attack}(\text{Image}_k)) \neq \text{TrueLabel}_k] \tag{8}$$

The given equation 8 represents the final scoring formula where *D* is all the defense sets, *N* is the number of images, $Image_k$ is the *k-th* image in the dataset, and $TrueLabel_k$ is the target label for image *k*.

The tables 3, 4, and 5 present the scores and run time statistics of the NIPS 2017 adversary competition submissions that won the first 10 places [20] in all the three tracks. Each row of these files corresponds to one submission. Columns have following meaning:

- Kaggle Team ID - either Kaggle Team ID or ID of the baseline.
- Team Name - human readable team name.
- Score - raw score of the submission.
- Normalized Score - normalized (to be between 0 and 1) score of the submission.
- Mean - average evaluation time of 100 images.

Table 3. NIPS 2017 Non-targeted Attack Competition Results

Kaggle Team ID	Team Name	Score	Normalized Score	Mean
833047	yinpeng	410363	0.78164381	423.22
823044	sangxia	407849	0.776855238	421.18
828273	zhangyuc	406363	0.774024762	496.74
829274	takiba	403715	0.768980952	76.9
881706	toshik	396689	0.755598095	448.72
908175	hx173149	366194	0.697512381	358.74
887539	adversarial	363501	0.692382857	377.94
818996	scottclowe	340492	0.64855619	499
826874	alekseynp	336724	0.641379048	449.16
806510	anlthms	333988	0.636167619	496.94

Table 4. NIPS 2017 Targeted Attack Competition Results

Kaggle Team ID	Team Name	Score	Normalized Score	Mean
823879	yinpeng	211161.0	0.402211428571	388.96
824554	sangxia	193606.0	0.368773333333	409.16
806865	yaozhoa	193215.0	0.368028571429	493.18
806502	anlthms	191390.0	0.364552380952	493.02
826875	alekseynp	182666.0	0.347935238095	486.94
830141	malzantot	169718.0	0.323272380952	500.08
784320	tarobxl	167335.0	0.318733333333	438.34
875646	shijing	166774.0	0.317664761905	326.36
917607	toshik	164319.0	0.312988571429	426.94
809250	voilin	149033.0	0.283872380952	420.7

Table 5. NIPS 2017 Defense Competition Results

Kaggle Team ID	Team Name	Score	Normalized Score	Mean
820405	liaofz	691044.0	0.953164137931	50.2443976724
858196	cihangxie	669555.0	0.923524137931	121.826296892
806513	anlthms	663259.0	0.91484	95.2925591185
779225	erkowa	661172.0	0.91196137931	86.4433576823
867887	acganesh	660180.0	0.910593103448	127.386777269
828434	tarobxl	658035.0	0.907634482759	103.463414634
806867	yaozhao	653566.0	0.901470344828	81.6122322645
817194	hx173149	651490.0	0.898606896552	46.2123313111
816739	rwightman	651419.0	0.898508965517	133.200940943
771244	confirm	649370.0	0.895682758621	88.2916924601

3.1. A Momentum-based Iterative FGSM (MI-FGSM) for Non-targeted Attack: Team yinpeng

The introduced method in this work [16] won the first places in both non-targeted and targeted attack tracks. In this section, the algorithm for non-targeted attack is presented.

This method is created by adding a momentum term to BIM in order to have more transferable adversarial examples. While one-step methods (i.e. FGSM) can underfit the model, BIM can overfit them; which are mostly

untransferable across different models. In order to overcome these problems, a momentum term is integrated into BIM to have more balanced update directions and get rid of poor local optima. In conclusion, having stronger and more transferable attack is the main advantage coming with this approach.

$$g^{t+1} = \mu\, g^t + (\nabla_x J(f(x^t_{adv}), y_{true}) / \|\nabla_x J(f(x^t_{adv}), y_{true})\|_1) ,$$
$$x^{t+1}_{adv} = Clip_{[0,1]}(x^t_{adv} + \alpha.sign(g^{t+1})) , \quad (9)$$

$$f(x) = \sum_{k=1}^{K} w^k f^k(x)$$

In the given formulation 9, $g^0 = 0$, $x^0_{adv} = x$, $\alpha = \varepsilon/T$, and let T be the number of iterations, g^t the gradients of the t number of iterations, μ the decay factor, and x^t_{adv} the perturbed adversarial example. In non-targeted attack, this algorithm is used for ensemble models; $f^k(x)$ is the k-th model, w^k is the ensemble weight. The models that are attacked are as follows:

- Normally trained models: Inception v3 [21], Inception v4 [22], Inception ResNet v2 [22], and ResNet v2-101 [23].
- An adversarially trained model: Inception $v3_{adv}$ [17].
- Ensemble adversarially trained models: Inc-$v3_{ens3}$, Inc-$v3_{ens4}$, and IncRes-$v2_{ens}$ [24].

3.2. A Momentum-based Iterative FGSM (MI-FGSM) for Targeted Attack: Team yinpeng

The formulation of momentum iterative targeted attack is slightly different from the one used for non-targeted attacks:

$$g^{t+1} = \mu.g^t + (\nabla_x J(f(x^t_{adv}), y_{target}) / std(\nabla_x J(f(x^t_{adv}), y_{target}))) ,$$
$$x^{t+1}_{adv} = Clip_{[0,1]}(x^t_{adv} - \alpha.Clip_{[-2,2]}(round(g^{t+1}))) . \quad (10)$$

In the equations 10, *std(.)* refers to the standard deviation and *round(.)* is rounding to the nearest integer. Transferability is not considered in targeted attacks. Same models for non-targeted track are used for targeted attacks in this work.

3.3. An Iterative FGSM-based Approach for Non-targeted Attack: Team Sangxia

The presented method in this work won the second places in both non-targeted and targeted attack tracks. In this section, the algorithm for non-targeted attack is introduced.

The solution by this team is inspired by the method which uses FGSM with Random Perturbation (RAND+FGSM) [24]. The optimal perturbation is detected under a first-order approximation after a random perturbation is added to an input. The reason why the random perturbation is used is because it skips the non-smooth vicinity of the data point. In [24], RAND+FGSM has a high success rate in white-box setting. However, the transferability of perturbations has low rate. Therefore, a geometric transformation is applied in order to increase transferability.

FGSM is a one-shot algorithm which means it only applies a single gradient computation. One-shot attacks provide resiliency for adversarially trained models. However, those models are vulnerable to the attacks which iteratively maximize their loss function. Iterative FGSM (I-FGSM) iteratively applies FGSM k times with the size $\varepsilon'=\varepsilon/k$. It computes multiple gradient updates and lowers error rates more than FGSM. In conclusion, the presented approach in this project outperforms other one-shot methods on all the trained models.

Table 6. The models and their success rate for non-targeted attack by team Sangxia.

Classifier	Success Rate
Inception v3	96.74%
ResNet 50	92.78%

Inception ResNet v2	92.32%
Inception v4	91.69%
EnsAdv Inception ResNet v2	87.36%
Adv Inception v3	83.73%
Ens-3-Adv Inception v3	62.76%
Ens-4-Adv Inception v3	58.11%

3.4. An Iterative FGSM-based Approach for Targeted Attack: Team Sangxia

The targeted attack has a similar approach with the non-targeted attack described in the previous subsection. The main differences are as follows:

- The loss function is minimized instead of maximizing.
- Image augmentation is not used for targeted-attacks due to its lowering the success rate.

3.5. High-level Guided Denoiser (HGD) for Defense: Team liaofz

This method won the first place in the defense track of NIPS 2017 competition. The purpose of this work [25] is to remove the adversarial noise by training a DNN-based denoiser.

20,000 images from ImageNet composed of 20 images for each class are used in order to train the denoiser. FGSM and Iterative FGSM (I-FGSM) are applied to a variety of models, such as Pre-trained Inception v3, Inception ResNet v2, and ResNet50 v2.

A denoising U-Net (DUNET) is used in order to remove adversarial perturbation with the following formulation:

$$dx' = D_w(x^{adv}),$$
$$x' = x^{adv} - dx'. \quad (11)$$

In the given equation 11, the clean image is obtained by subtracting the perturbation from the noisy image where D_w is a denoiser network, dx' is estimated perturbation, and x^{adv} is reconstructed noise-free image.

3.6. A Randomization-based Method for Defense: Team cihangxie

This model won the second place in the defense track with a score of 92.4% among 107 defense submissions. It introduces a defense method against adversarial samples that applies one random resizing layer and one random padding layer afterwards to the first part of classification network models. It provides with great benefits: First, it does not require an extra fine-tuning or training steps. Second, it uses few extra steps to do its computations. And third, it has compatibility with other adversarial defense algorithms. All in all, it has effective robustness against adversarial attacks.

The main motivation of this work is the occurrence of over-fitted adversarial noises due to iterative attacks; therefore, they are less transferable. It is claimed that some certain structure of adversarial noises can be destroyed by random transformations, such as resizing and padding. This is because a random transformation is applied to each test image during the generation of adversarial perturbation by the attacker unaware of this transformation.

Fig. 1. The pipeline of the defense model proposed by team chiangxie. From left to right: input image x, resized image x', padded image x'', and a deep network used for classification. (1) represents a random resizing layer and (2) is a random padding layer.

4. Conclusion

Adversarial images comprise a menace to security and privacy since machine learning systems can be exposed to some type of attacks performed by those images. In this paper, we analyzed and compared current novel methods for generating adversarial examples and defending against them. In addition, we demonstrated some top-scored solutions in all three tracks of the NIPS 2017 Adversarial Learning competition. This competition made significant contributions on machine learning security.

References

[1] C. Middlehurst. (2015) "China unveils world's first facial recognition atm." http://www.telegraph.co.uk/news/worldnews/asia/china/11643314/China-unveils-worlds-first-facial-recognition-ATM.html.

[2] A. Harvey. CV Dazzle. (2010) "Camouflage from face detection." Master's thesis, New York University. Available at: http://cvdazzle.com.

[3] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. (2016) "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. ACM, 2016, pp. 1528-1540.

[4] NEC. Face recognition. \\ http://www.nec.com/en/global/solutions/biometrics/technologies/face recognition.html.

[5] NEURO Technology. SentiVeillance SDK. \\ http://www.neurotechnology.com/sentiveillance.html

[6] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, et al. (2016) "Deep speech 2: End-to-end speech recognition in english and mandarin." in International Conference on Machine Learning, pages 173-182.

[7] iOS - Siri - Apple, https://www.apple.com/ios/siri/.

[8] Alexa, https://developer.amazon.com/alexa.

[9] Cortana - Your Intelligent Virtual and Personal Assistant - Microsoft, https://www.microsoft.com/en-us/windows/cortana.

[10] A. Kurakin, I. Goodfellow, and S. Bengio. (2016) "Adversarial examples in the physical world." arXiv preprint arXiv:1607.02533.

[11] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. (2017) "Robust physical-world attacks on deep learning models.", arXiv preprint arXiv:1707.08945, vol. 1.

[12] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille. (2017) "Adversarial examples for semantic segmentation and object detection." in International Conference on Computer Vision. IEEE.

[13] Nicholas Carlini and David Wagner. (2016) "Towards evaluating the robustness of neural networks." arXiv preprint arXiv:1608.04644.

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. (2017) "Towards deep learning models resistant to adversarial attacks." arXiv preprint arXiv:1706.06083. The complete code, along with the description of the challenge, is available at https://github.com/MadryLab/mnist_challenge and https://github.com/MadryLab/cifar10_challenge.

[15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. (2014) "Explaining and harnessing adversarial examples." CoRR, abs/1412.6572. URL http://arxiv.org/abs/1412.6572.

[16] Y. Dong, F. Liao, T. Pang, H. Su, X. Hu, J. Li, and J. Zhu, (2017) "Boosting adversarial attacks with momentum", arXiv preprint arXiv:1710.06081.

[17] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. (2016) "Adversarial machine learning at scale." arXiv preprint arXiv:1611.01236.

[18] N. Papernot, P/ McDaniel, X. Wu, S. Jha, and A. Swami. (2016) "Distillation as a defense to adversarial perturbations against deep neural networks." IEEE Symposium on Security and Privacy.

[19] Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., and Swami, A. (2016) "The limitations of deep learning in adversarial settings." in Proceedings of the 1st IEEE European Symposium on Security and Privacy, pp. 372-387, 2016b.

[20] https://www.kaggle.com/google-brain/nips17-adversarial-learning-final-results.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. (2016) "Rethinking the inception architecture for computer vision." in CVPR.

[22] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. (2017) "Inception-v4, inception-resnet and the impact of residual connections on learning." in AAAI.

[23] K. He, X. Zhang, S. Ren, and J. Sun. (2016) "Identity mappings in deep residual networks." in ECCV.

[24] Florian Tram'r, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick D. McDaniel. (2017) "Ensemble adversarial training: Attacks and defenses." arXiv preprint arXiv:1705.07204.

[25] F. Liao, M. Liang, Y. Dong, T. Pang, J. Zhu, and X. Hu. (2017) "Defense against adversarial attacks using high-level representation guided denoiser." arXiv preprint arXiv:1712.02976.