Evaluation of GAN Architectures for Adversarial Robustness of Convolution Classifier

Weimin Zhao, Sanaa Alwidian and Qusay H. Mahmoud

Department of Electrical, Computer, and Software Engineering
Ontario Tech University, 2000 Simcoe St., ON, Oshawa, L1G 0C5, Canada

Abstract

Deep learning models are vulnerable to adversarial attacks, which could generate adversarial perturbation to make deep learning classifiers fail in classification tasks. In this paper, we reviewed a variety of defensive methods against adversarial attacks and proposed our solution. We present a modification of Generative Adversarial Network (GAN) architecture to train a classifier for the purpose of defending against adversarial attacks. We trained multiple deep learning classifiers with different generator formulations to compare and evaluate their robustness against adversarial attacks. We show how the GAN architecture could contribute to the adversarial machine learning problem and how the capacity of the generator affects the training performance. Our results show that GAN architectures can improve the adversarial robustness of a deep learning classifier with more efficiency training time. The CIFAR 10 classifier accuracy remains around 45% under 8/255 L infinity norm adversarial distortion.

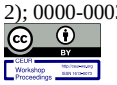
Keywords

Machine learning, deep learning, adversarial attacks, adversarial samples, adversarial training, generative adversarial networks (GAN)

1. Introduction

In recent years, there has been an increasing attention to the security of deep learning networks regarding the area of adversarial machine learning. Deep learning models are prone to the threats caused by well-known attack algorithms that could generate malicious data samples (e.g., fast gradient sign method [1] and projected gradient descent [2]). These attacks are known as Adversarial Attacks, and the data samples generated from the attack algorithms are known as Adversarial Samples. Recent research has proved that these attack algorithms are highly effective in generating small perturbations for any given sample of data that could substantially affect the output of any deep learning model. The famous example of the algorithms includes the Fast Gradient Sign Method (FGSM) [1] and the Projected Gradient Descent (PGD) [2], which exploit the loss gradient of the deep learning model and use it to generate adversarial samples. The other algorithms such as Carlini and Wagner attack [3] and evolutionary based algorithms [4] generate adversarial samples under different constraints.

There are several proposals for mitigating the problems of adversarial samples. *Adversarial training* is one of the well-established defense methods that augment the training data with the attack algorithms to make the classifiers generalize on adversarial samples [1, 2]. Adversarial training is effective in defending the attack algorithm and provide a baseline robustness to defend other similar or weaker

The AAAI-23 Workshop on Artificial Intelligence Safety (SafeAI 2023), February 13–14, 2023, Washington DC, USA
EMAIL: weimin.zhao@ontariotechu.net (A. 1);
sanaa.alwidian@ontariotechu.ca (A. 2);
qusay.mahmoud@ontariotechu.ca (A. 3)
ORCID: 0000-0002-6664-5632 (A. 1); 0000-0002-9339-1308 (A. 2); 0000-0003-0472-5757 (A. 3)


attack algorithms [2]. The other algorithms include defensive distillation [5] and data augmentation [6]. However, adversarial training remains one of the top effective methods of defending adversarial attacks.

The limitations of adversarial training include the challenge of generalization, the clean and robustness trade-offs, and the high training complexity. Usually, a classifier needs to be trained against a multi-iteration adversarial attack to achieve a good robustness against strong adversarial attacks [2]. Hence, in each training loop, some extra iterations of gradient descent need to be computed to obtain the worst-case adversarial noise for data augmentation. This process increases the training time exponentially with a more complex dataset and larger data number.

Wang et al. [7] proposed a training framework by replacing the attack algorithm with a generative network in an extension of adversarial training. As a result, the framework formulates a GAN architecture, where the generator acts as an attacker, and the discriminator acts as a classifier. This method removed the need for an attack algorithm during the training and reduced the multi-step backpropagation complexity compared to the PGD-adversarial training. In this paper, we consider extending the GANs defensive training architecture from Wang et al. [7] to evaluate multiple generators and classifiers. The contributions of this paper are:

1. The design and development of a GAN architecture to improve the adversarial robustness of a deep learning classifier on the classification against the gradient-based white-box attack algorithms.
2. A dual generator framework architecture to improve the effectiveness of generating adversarial samples during the training process.
3. The implementation of different formulations of generators and evaluation of the training performance and the robustness of the classifiers that train against the different formulations. Furthermore, we evaluated if the generative network could capture the strong adversarial noise perturbation direction and perform on par with the strong adversarial attack in adversarial training.

The rest of the paper is organized as follows: Section 2 discusses the related work regarding adversarial attacks and defenses and introduces the defensive methods related to GANs. Section 3 introduces the methodologies and architectures of our frameworks. Section 4 presents our experimental setups and results and provides discussion regarding the results. Finally, Section 5 concludes the paper and provides a discussion about the future work.

2. Related Work

In this section, we discuss recent works related to *adversarial attacks*, *adversarial training* and *GANs'* contributions to the adversarial machine learning community. The paper focuses on reviewing the gradient-based adversarial attacks since they have a closer relationship with GANs training process.

2.1. Gradient-Based Adversarial Attacks

A simpler formulation of gradient-based attack algorithm is Fast Gradient Sign methods (FGSM) [1], which has the following formulation:

$$X' = X + \varepsilon \cdot sign(\nabla x\, L(X, y)), \quad (1)$$

where X is the original sample data and $\nabla_x L(X, y)$ represents a one-step gradient calculation regarding the classifier loss L. This loss gradient is usually computed from a backpropagation of a deep learning classifier. The algorithm takes the sign direction of the gradient and adds it to the original sample with a scaler $\mathcal{E}$ to generate the adversarial samples. The scaler $\mathcal{E}$ constrains the maximum L infinity norm of the perturbation vector, which also is known as the *attack strength*.

Projected Gradient descent (PGD) attack [2] was an improved version of the gradient-based attack that iteratively computed the gradients of the classifier. The base formula is written as follows:

$$X_{t+1} = \prod_{x+s}(X' + \varepsilon\, sign(\nabla x\, L(x, y))), \quad (2)$$

where the gradient sign $sign(\nabla x\ L(x,\ y))$ is calculated multiple times and iteratively adds on the original input X. X' in this formula represents the perturbed input from the last step of the calculation, and X_{t+1} is the output of the current iterative. We think these gradient-based algorithms had a similar property to GANs formulation, that the generator of GANs could also be used to capture the loss gradient of a classifier.

2.2. Adversarial Training

Adversarial training refers to the training schema that incorporates an adversarial attack algorithm to augment the training data [1, 2]. In general, any attack algorithms could be used for data augmentation. However, the most common ones are gradient-based attacks since they are efficient for implementation, and the classifier's loss gradient could be directly accessed during the training process. Gradient-based adversarial training was introduced by Goodfellow et al. [1] after the discovery of FGSM. The general formulation of adversarial training could be expressed as follows:

$$\min \sum \max L(f(x_i + \delta), y_i), \quad (3)$$

where the classifier is optimized to minimizes the cross-entropy loss L on adversarial sample $x+\delta$, and the adversarial sample $x+\delta$ maximizes the loss of the classifier. Later, Madry et al. [2] suggested using multi-step gradient PGD attacks to improve the performance of adversarial training to defend against more precise first-order attacks. The PGD adversarial training achieved state-of-the-art accuracy against the strong PGD attacks and was used as a baseline adversarial training method. However, limitations were found with a more complex dataset. Zhang et al. [8] suggested a trade-off between robustness and accuracy. The recent developed classifier had a limited parameter and capacity to generalize on both clean and adversarial data. Furthermore, the training time increases significantly with the implementation of the multi-iteration gradient attack within every training iteration [2].

2.3. GANs

GANs are popular deep learning techniques for data synthesis. Recently, some high-quality synthetic images could be generated by using some state-of-the-art GANs [9]. The formulation of GANs incorporates a *generator network* and a *discriminator network* to form a min-max game. The discriminator could be optimized to differentiate the generated data and real data. The gradient is backpropagated from the discriminator to improve the generator performance. Normally, the goal of GANs is to optimize the generator to improve data synthesis. Xiao et al. [10] used a generator network to capture the loss gradient regarding the input images of a classifier network. The generator can be also used as an effective adversarial sample generator after training.

On the other side, multiple GANs frameworks were proposed to defend the adversarial attack. Shen et al. [11] and Samangouei et al. [12] proposed a GANs to cleanse the adversarial perturbation. These GANs optimize a generator to transfer the adversarial images to harmless samples and reduce the effect of adversarial perturbation. The other defensive GANs [13, 14, 15, 16] also utilized the feature-to-feature transfer generative model to denoise the adversarial samples. Liang et al. [17] implemented a defensive GANs architecture to learn the perturbation features and provide a robust classification result. Liu et al. [18] proposed a GANs adversarial training schema to improve adversarial robustness of a classifier. Wang et al. [7] also suggested that GANs can improve the adversarial robustness of the deep learning classifier model. This framework [7] has an alternative optimization goal that focuses on the performance of the classifier instead of the generator. The formulation of the optimization is expressed as follows:

$$min_D max_G \sum L\,(D(X_i), y_i) + \sum L\big(D\big(X_i + \epsilon G(X_i)\big), y_i\big), \quad (4)$$

The key difference between this framework and adversarial training is that it replaced the attack algorithm as a generative network to synthesize the adversarial samples. The generator outputs a vector of perturbation *G(X)*, and the perturbation *G(X)* adds to the original sample *X* with a scaler ϵ to construct the final adversarial samples. The scaler ϵ is a variable to constrain the L infinity norm of the perturbation vector from the generator. The optimization goals were to minimize the classification loss *L(D(Xi),yi)* of discriminator *D* regarding both original sample *X* and perturbed sample *Xi+ϵG(Xi)*, and generator *G* is optimized to maximize the loss of *D*. The results showed that this framework could improve the robustness of the classifier (i.e., discriminator) and the classifier could generalize on the generated samples easier than traditional adversarial attack samples. Our work makes inspirations from this work. We propose multiple modifications to this framework and aim to improve the stability and overall performance of the training.

In this paper, we applied GAN architecture to adversarial training formulation. Different from the previous works, we implemented four frameworks with four different generators' formulations. The four formulations estimate the adversarial noises as four function

transformations from different inputs. The training performance of the GAN frameworks involved different generator formulations was compared and analyzed.

3. Methodology

The general formulation of our network is a modification to equation (4) from Wang et al. [7]. It can be described as following:

$$min_D max_G \sum L\left(D(X_i), y_i\right) + \sum L\big(D\big(x_i + \epsilon G(\boldsymbol{I})\big), y_i\big), \quad (5)$$

with a slight difference in considering a set of different inputs for the generator. The different inputs are represented by I in the formula and it affects the formulation of the generator for adversarial noise generation. The following describes the four different formulations we considered for the generator:

$$N = G1(x), \quad (6)$$

$$N = G2(x, z), \quad (7)$$

$$N = G3\big(sign(\nabla)\big), \quad (8)$$

$$N = G4(sign(\nabla), x), \quad (9)$$

The four types of generator formulation are represented as:

1. *G1*: formulates adversarial noise N as function transformation of only target image x,
2. *G2*: formulates adversarial noise N as function transformation of the target image x and a latent noise z,
3. *G3*: formulates adversarial noise N as function transformation of sign of the loss gradient sign(∇) of the current classifier state regarding the target input x,
4. and *G4*: formulates adversarial noise N as function transformation of the target image x and the sign of the loss gradient sign(∇) of the current classifier state regarding the target input x.

The loss gradient of the classifier is calculated each time before the input is fed into the generator for the formulations (8) and (9). For simplicity, we will refer to the formula (8) and (9) as G(∇) and G(∇, x) in the subsequent sections.

The optimization goal is to minimize the discriminator (classifier) loss on both clean and synthesized data from the generator and maximize the synthesis image's loss regarding the discriminator's classification output. The generator in our framework is responsible for constructing perturbation vectors instead of images. The perturbation vectors are added to the original images to produce adversarial samples that feed into the classifier for training. The perturbation from generator output is soft-clipped by "tanh" activation and constrained by a scaler value ϵ that is defined manually. This scaler value is the control variable that defines the maximum L infinity norm of the perturbation. The details of the value settings will be discussed in the experiment section. We also refer to this scaler as the strength of the perturbation or attack strength in the later section.

The GAN architecture also involves dual generators during the training. Both generators provide perturbation solutions for the classifier to train. Im et al. [19] suggested that involving multiple generator and discriminator pairs could improve the stability of training, and the generator could generate more variety of distributed samples. The pairs architecture should act as a regularization to the GANs that reduces the overfitting effect of the network. The architecture of our framework is illustrated in Figure 1.

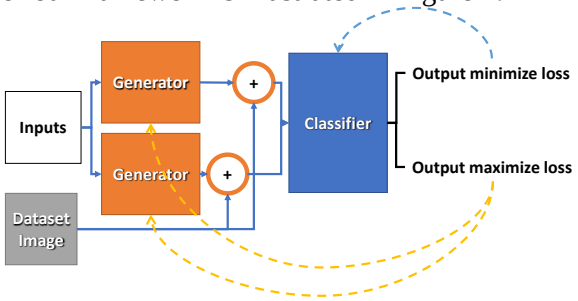


Figure 1 Framework architecture. The dotted lines represent gradient propagation.

Additionally, we used a standard categorical cross-entropy loss as the loss function for the discriminator. As for the generator, the optimization should maximize the cross-entropy loss of the discriminator. Hence, the loss function is defined as a negative cross-entropy loss in contrast to the discriminator loss.

4. Experimental Results

We considered the Canadian Institute For Advanced Research (CIFAR) 10 dataset for the experiments. We chose the CIFAR 10 dataset as it is a more challenging problem to optimize a robust classifier for colorful images and scalability of CIFAR 10 is within our scope of the project.

In our experiment, we used a generator similar to the cycle GAN generator [20]. We consider this type of generator because it is useful for image and texture transformation. The architecture of the generator is illustrated in Figure 2. A shallow

VGG-like architecture was implemented as the discriminator. Each layer used batch normalization and ReLU activation. The details of the filter parameters of the models are shown in Table 1. We implemented multiple generator filter settings and one discriminator filter setting for our experiments. The "×1" generator filter setting is more efficient to train and was used in generator formulations comparison. However, from some extensive experiments, we found that the "×2" filter setting slightly improve the overall performance, so we implemented this setting for the further experiments.

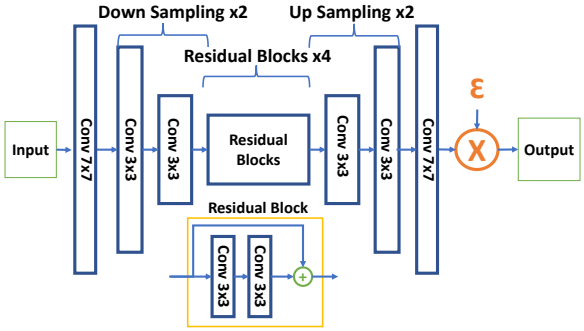


Figure 2 The generator architecture. The Ɛ variable constrains the maximum L infinity norm of the output vector.

Table 1

The filter settings of the generator and discriminator.

Output size (generator)	Generator Filters x1	Generator Filters x2	Output size (discriminator)	Discriminator Filters
32×32	7×7, 32, stride 1	7×7, 32, stride 1	32×32	[3×3, 32, stride 1] ×2
16×16	5×5, 64, stride 2	5×5, 128, stride 2	16×16	Max pool, stride 2
8×8	5×5, 128, stride 2	5×5, 256, stride 2	16×16	[3×3, 64, stride 1] ×2
8×8	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}$ ×4	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}$ ×4	8×8	Max pool, stride 2
16×16	5×5, 64, stride 2	5×5, 128, stride 2	8×8	[3×3, 128, stride 1] ×3
32×32	5×5, 32, stride 2	5×5, 32, stride 2	4×4	Max pool, stride 2
32×32	7×7, 3, stride 1	7×7, 3, stride 1	4×4	[3×3, 256, stride 1] ×3
			2×2	Max pool, stride 2
			1024	Flatten
			10	10-d fc

During training, we added dropouts as regularization layers to each layer of the discriminator. For every shallow VGG model, the dropout value is 0.3 for the first two layers and 0.4 for the rest of the layers except the output layer.

The optimizer implemented for all the experiments training is the Adam optimizer with a 0.0002 learning rate, 0.5 $Beta_1$, 0.9 $Beta_2$ for generators and 0.0001 learning rate, 0.5 $Beta_1$, and 0.9 $Beta_2$ for discriminators. The learning rate for the generators is higher because, during the training dynamic, we expect the generator to capture the loss gradient of the discriminator's current state. Hence, we reduced the learning process of the discriminator to make the learning process easier for generators.

We used a scaler (ϵ) of 16/255 as perturbation strength to constrain the generator perturbation vectors.

All the images' pixels were normalized within the range of [0, 1] before training and testing. Additional data augmentations were used, including random horizontal flips and random shifts within the range of 0.1 fractions of the original image size.

A baseline models was also trained without GANs framework. We used a conventional training schema to train the baseline models. The structure of baseline model is consistent with the shallow VGG model.

4.1. Generator Formulations

This section lists and compares the training results from different formulations. In this section, we used "×1" generator filter number for experiments, shown in Table 1. The same filter parameters were used for all formulations from section 3. The baseline model was also tested for comparison. To evaluate the robustness of the models, we implemented FGSM and PGD as two standard testing attack algorithms. It is suggested that only the robustness evaluated under the strong multi-iteration attack is valid [2]; however, we still include the one-step gradient attack algorithm FGSM to compare the effectiveness by using generator to estimate the adversarial noise. The max-iteration of PGD attack is set to 100. The results are plotted in Figure 3.

From the plot, all the models trained with GANs framework significantly improved in terms of robustness against different levels of adversarial noise. The discriminators trained against the generators that included target image x as an input option (G(x) and G(x, z)) generally gained similar robustness against FGSM and PGD. The accuracy against FGSM was slightly higher compared to against PGD under the highest noise norm setting (16/255). This result means that the discriminators (classifiers) trained against G(x) and G(x, z) formulations had improved adversarial robustness but also had a more obvious loss gradient compared to other models.

One step of gradient descent is sufficient to discover adversarial samples similar to those computed from multi-step gradient descents. We also observe that the formulations, including gradient inputs, generally gain better robustness than other formulations. Formulations G(∇) and G(∇,x) generally had a similar performance under all attack settings. However, the performance under PGD attacks downgrades when noise norm increases compared to FGSM attacks, but the accuracies stay higher than other formulations. Furthermore, the clean data accuracy of the baseline model is higher than all the formulations. The generator that considered both gradient and image inputs had the lowest clean accuracy.

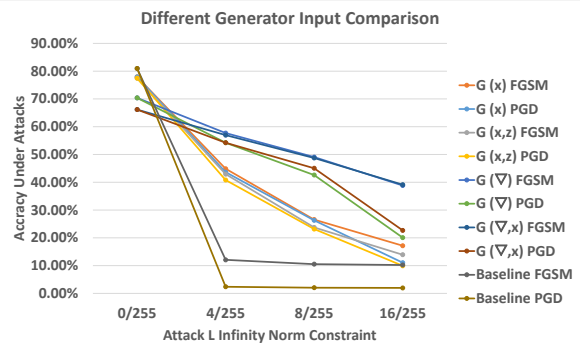


Figure 3: The discriminator robustness comparison between different formulations. 0/255 indicates no perturbation is added to the images.

From this experiment, we observe that the GAN architectures help increase the robustness of a classifier. The formulation of the generator has an impact on the adversarial noise estimation capacity. The generator with initial gradient information from the classifier had better training results in GANs frameworks. We suggested that with the first step gradient information, the initial direction of the perturbation is available to the generator. The generator only needs to discover the perturbation's further direction compared to the full perturbation direction. This formulation decreases the difficulty of the generator optimization; hence the generator can discover a worse-case perturbation more efficiently. The classifier trains against this generator formulation should yield a better robustness result. However, there is still a limitation on the generator capacity to find the worst-case adversarial noise. The robustness performances under the highest norm constraints (16/255) only reached around 20% accuracy at best. Furthermore, the clean data accuracies downgrade significantly with GANs frameworks. The frameworks with gradient input generators have the worst clean data accuracies. We observed overfitting effects with all GANs frameworks' discriminators, with the training set data accuracy could reach over 90% after GANs training, but testing set accuracy is only around 70%. With the same discriminator architecture, the conventional training accuracy is around 81% for both training and test set data. We tried adding L2 regularization for each layer or using an Adam optimizer with weight decay (AdamW). However, the regularization helped little to the results, and the performance downgraded when the weight decay was above 0.0001. Currently, we have no idea why the overfit is happening. We suspect the synthetic data from the generator has a different data distribution that shifts the decision boundaries of the discriminator.

4.2. Training Epochs

In this section, we evaluate the discriminators' accuracy performance with different training epochs. The architecture selected for this evaluation is the G(∇) formulation generator with "×2" filter numbers and the same discriminator, shown in Table 1. The other setting for the experiment is consistent with the previous section. Figure 4 shows the results of the experiment.

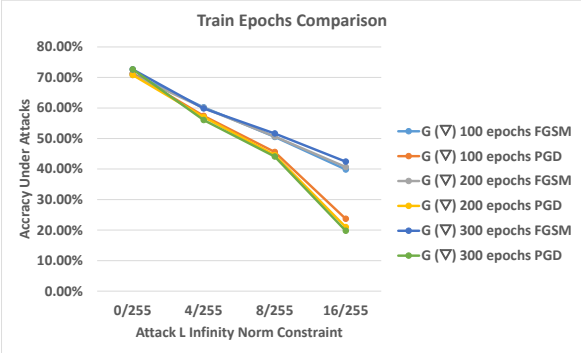


Figure 4: Discriminator robustness comparison between the different training epochs.

With increasing the training epochs, the clean data accuracies did not deviate too much. We observed that the accuracies under FGSM attacks increased with more training epochs; however, the accuracies under PGD slightly decreased with more training epochs. This effect is more significant with larger noise norm sizes. Usually, we expect an improved robustness for both FGSM and PGD with same training iterations; however, this plot is inconsistent with our expectations. The reason behind this performance downgrade is not apparent. We hypothesize that the generator started to overly fit to FGSM perturbation and produce the one perturbation direction with later

training iterations, and the discriminator learned to classify the images with this perturbation direction; however, the gradient was saturated for the generator to learn a new adversarial direction.

4.3. Results for PGD 1000 Iterations

In previous sections, we used PGD with 100 iterations to evaluate our models. However, the results obtained by 100 iterations may not describe the overall robustness. Therefore, in this section, the accuracy results from 1000 iterations are collected and shown in Table 2. All parameters and settings are the same to the last section 4.2. The model trained with 100 training epochs was selected for this experiment. From the table, we did not observe a significant accuracy difference when iteration increases. With all norm constraints, the accuracy of the model stays relatively identical. This result shows that the robustness evaluation for this model is sufficient by using the PGD 100 iterations. Therefore, we suggest that the evaluations from previous sections are valid and sufficient to conclude the overall robustness of the model.

Table 2

The model accuracies under PGD 100 and 1000

L infinity norm size	4/255	8/255	16/255
Accuracy PGD 100	57.41%	45.6%	23.71%
Accuracy PGD 1000	57.33%	45.28%	23.31%

4.4. Visualization of Adversarial Directions

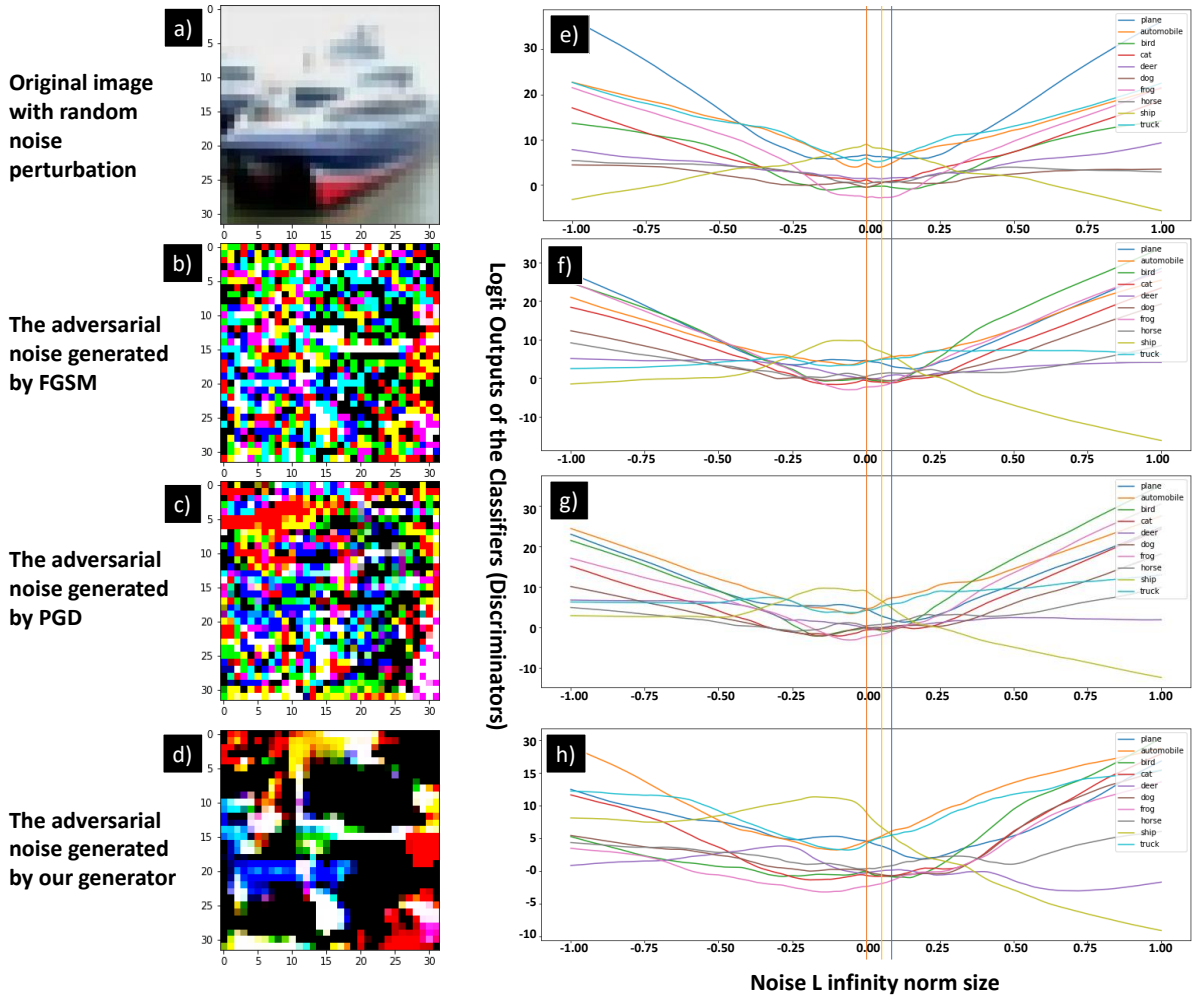


Figure 5: Adversarial noise visualization with model's logit outputs. The left-hand side shows the figures of the original image (a), adversarial noise from the FGSM attack (b), adversarial noise from the PGD attack (c), and adversarial noise from our generator (d). The right-hand side shows the pre-SoftMax output values trending when the original image is perturbated by a random noise (e), the FGSM adversarial noise (f), the PGD adversarial noise (g), and the generator adversarial noise(h). The labels of the right-hand side plots are shared.

In this section, we visualize the perturbation generated by the generator from our GANs frameworks. We chose the generator with G(∇) formulation and "×2" filter numbers (Table 1). The perturbations were generated with the framework trained after 300 epochs. We chose the maximum epochs from our experiments since we want to investigate the maximum effect of training with the framework. The FGSM and PGD noises were generated based on the classifier's (discriminator's) state after the 300 epochs of training. The visualization is shown in Figure 5.

The target image (a) is the top left image with a correct label of "ship". The noise images under the target image are the perturbation vectors generated by FGSM (b), PGD (c), and our generator (d), respectively. The right-hand side plots (e, f, g, h) show the classifier's logits outputs trending regarding the different norm size by adding the respective noise vector onto the target image. The top-right plot (e) shows the logits output trending when the classifier is perturbed by a random Gaussian noise vector with zero mean and one standard deviation. We do not show the Gaussian noise since the plot indicates the model's behavior in standard non-adversarial condition and there is no significant information within the Gaussian noise vector. Furthermore, the negative norm size in the x axis represents the opposite perturbation direction by the noise vector, which means subtracting the noise vector from the target image.

When the target image is perturbed by random noise (e), the logit distribution shows a relative symmetry shape whether the noise norm increases in either direction. However, with adversarial noise perturbation (f, g, h), the logit output regarding the correct class label decreases more dramatically with increasing norm size in a positive direction. We labeled the cross points when another class label output value started surpassing the correct class label's output. The surpassed point for FGSM is labeled with the blue line, and the surpassed point for PGD is labeled with the yellow line. The orange line indicates the position where the noise norm is zero, which means no perturbation is added to the image. As expected, the surpassed point of PGD is closer to the zero-norm point compared to the FGSM's surpassed point since PGD could find an adversarial sample with smaller perturbation with its iterative gradient descent. The surpass point for the noise of the generator stays closer to the zero-norm point compared to the FGSM noise and stay further compared to the PGD noise. These results suggest that the adversarial noise from the generator is stronger than the noise from a single-step adversarial attack but weaker than the noise from an iterative adversarial attack. We also observed that the noise from the generator is more uniform compared to the noisier noise from the gradient descent attacks. This more uniform noise may relate to the convolution generator's properties and capacities.

4.5. Discussion

With the experiments illustrated in this paper, we showed that the framework of GANs could be used to improve the adversarial robustness of the deep learning classifier. The discriminator could be trained with increasing robustness against the gradient-based adversarial attack algorithms. However, the trade-off still presents between clean data accuracy and robustness of the model, especially with a high dimensional dataset. In the current state, we need more experiments to analyze the training dynamics of GANs and make further improvements. We also found that the structure of the generator could affect its performance in perturbation generation. This finding is different from the previous work [7]. Furthermore, using the signed gradient from one-step backpropagation is the best option using for the generator input to approximate the strong adversarial perturbation. This formulation helps to bridge the gap between the computation overhead of the one-step gradient attack algorithm and the multi-step attack algorithm during adversarial training.

5. Conclusion and Future Work

This work intends to verify the possibility of GANs contributing to adversarial attack defenses. We suggest that the properties of GANs could be utilized further for defending against adversarial attacks. We might be able to improve the formulation of the framework to make it more applicable to real-life implementation scenarios. There are a lot of hidden properties to be discovered from GANs and the phenomenon of adversarial samples. We believe the synthetic adversarial sample could explain more about the properties of deep learning models. In the next step, we hope to extend this framework to other types of deep learning models, such as attention-based models, and evaluating how they perform under the GANs training. Beside image models,

GANs can also train deep learning models used in other application domains, including text recognition models, malware detection models, and reinforcement learning models. We are also interested in exploring the different application domains with GAN robustness training architecture to address the problem of adversarial samples.

6. References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, Explaining and Harnessing Adversarial Examples, arXiv:1412.6572, 2015.

[2] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, Towards Deep Learning Models Resistant to Adversarial Attacks, arXiv:1706.06083, 2019.

[3] N. Carlini and D. Wagner, Towards Evaluating the Robustness of Neural Networks, arXiv, 2017. doi: 10.48550/arXiv.1608.04644.

[4] J. Chen, M. Su, S. Shen, H. Xiong, and H. Zheng, POBA-GA: Perturbation optimized black-box adversarial attacks via genetic algorithm, volume 85 of Computers & Security, 2019, pp. 89–106. doi: 10.1016/j.cose.2019.04.014.

[5] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks, in: IEEE Symposium on Security and Privacy (SP), 2016, pp. 582–597. doi: 10.1109/SP.2016.41.

[6] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, Mitigating Adversarial Effects Through Randomization, arXiv, 2018. doi: 10.48550/arXiv.1711.01991.

[7] H. Wang and C.-N. Yu, A Direct Approach to Robust Deep Learning Using Adversarial Networks, arXiv, 2019. doi: 10.48550/arXiv.1905.09591.

[8] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. E. Ghaoui, and M. Jordan, Theoretically Principled Trade-off between Robustness and Accuracy, in: Proceedings of the 36th International Conference on Machine Learning, 2019, pp. 7472–7482.

[9] T. Karras, T. Aila, S. Laine, and J. Lehtinen, Progressive Growing of GANs for Improved Quality, Stability, and Variation, arXiv, 2018. doi: 10.48550/arXiv.1710.10196.

[10] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, Generating Adversarial Examples with Adversarial Networks, arXiv, 2019. doi: 10.48550/arXiv.1801.02610.

[11] S. Shen, G. Jin, K. Gao, and Y. Zhang, APE-GAN: Adversarial Perturbation Elimination with GAN, arXiv, 2017. doi: 10.48550/arXiv.1707.05474.

[12] P. Samangouei, M. Kabkab, and R. Chellappa, Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models, arXiv, 2018. doi: 10.48550/arXiv.1805.06605.

[13] F. Yu, L. Wang, X. Fang, and Y. Zhang, The Defense of Adversarial Example with Conditional Generative Adversarial Networks, volume 2020 of Security and Communication Networks, p. e3932584, 2020, doi: 10.1155/2020/3932584.

[14] A. ArjomandBigdeli, M. Amirmazlaghani, and M. Khalooei, Defense against adversarial attacks using DRAGAN, in: 2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), 2020, pp. 1–5. doi: 10.1109/ICSPIS51611.2020.9349536.

[15] G. K. Santhanam and P. Grnarova, Defending Against Adversarial Attacks by Leveraging an Entire GAN, arXiv, 2018. Doi: 10.48550/arXiv.1805.10652.

[16] R. Bao, S. Liang, and Q. Wang, Featurized Bidirectional GAN: Adversarial Defense via Adversarially Learned Semantic Inference, arXiv, 2018. doi: 10.48550/arXiv.1805.07862.

[17] Q. Liang, Q. Li, and W. Nie, LD-GAN: Learning perturbations for adversarial defense based on GAN structure, volume 103 of Signal Processing: Image Communication, p. 116659, 2022, doi: 10.1016/j.image.2022.116659.

[18] G. Liu, I. Khalil, and A. Khreishah, GanDef: A GAN Based Adversarial Training Defense for Neural Network Classifier, in: ICT Systems Security and Privacy Protection, Cham, 2019, pp. 19–32. doi: 10.1007/978-3-030-22312-0_2.

[19] D. J. Im, H. Ma, C. D. Kim, and G. Taylor, Generative Adversarial Parallelization, arXiv, 2016. doi: 10.48550/arXiv.1612.04021.

[20] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks, 2017, pp. 2223–2232.