Generative Adversarial Networks for Adversarial Training

Guneet S. Dhillon *, **Brahma S. Pavse** *
guneetdhillon@utexas.edu, brahmasp@utexas.edu

Abstract

Following recent work, neural networks are widely-known to be vulnerable to adversarial examples. Carefully chosen perturbations to real images, while imperceptible to humans, induce misclassification, threatening the reliability of deep learning in the wild. In this paper, we cast the problem as a minimax zero-sum game between the adversary and the defender to show that there are better methods than the *fast gradient sign method* to create adversarial examples and to enhance robustness against adversarial examples. Taking inspiration from this, we propose training the model in a *generative adversarial networks* framework, where both the adversary and the defender learn model parameters simultaneously. We test this approach on the MNIST dataset, and compare it against other existing methods of training the model, using accuracies and calibration plots as metrics.

1 Introduction

While deep neural networks have gained nearly uncontested primacy in supervised learning, they remain vulnerable to adversarial examples [Szegedy et al., 2013]. Small, carefully chosen perturbations to input data can induce misclassification with high probability. In the image domain, even perturbations so small as to be imperceptible to humans can fool powerful convolutional neural networks [Szegedy et al., 2013, Goodfellow et al., 2014b]. This fragility presents an obstacle to using machine learning in the wild. For example, a vision system vulnerable to adversarial examples might be fundamentally unsuitable for a computer security application. Even if a vision system isn't explicitly used for security, these weaknesses might be critical. Moreover these problems seem unnecessary. If these perturbations are not perceptible to people, why should they fool a machine?

Since this problem was first identified, several papers have investigated techniques both for generating and guarding against adversarial attacks. One way of generating adversarial examples was introduced in Goodfellow et al. [2014b], called the "fast gradient sign method". To produce an adversarial example, we take one step in the direction of the sign of the gradient of the loss with respect to the inputs.

Many papers attempt to defend against adversarial examples by training the neural network on adversarial examples, either using the same model as in Goodfellow et al. [2014b], Madry et al. [2017], or using an ensemble of models as in Tramèr et al. [2017]. Taking a different approach, Nayebi and Ganguli [2017] draw inspiration from biological systems. They propose that to harden neural networks against adversarial examples one should learn flat, compressed representations that are sensitive to a minimal number of input dimensions.

In this paper, we introduce a method for training the model in a *generative adversarial networks* (GAN) framework, with an adversary and a defender. The adversary learns to create perturbations to fool the defender, while the defender learns to become robust against such perturbations. Both the adversary and the defender are trained simultaneously.

*Indicates equal contributions.

2 Preliminaries

We denote an n-layered deep neural network $h : \mathcal{X} \to Y$ as a chain of functions $h = h_n \circ h^{n-1} \circ \ldots \circ h^1$ where $h^i \in \mathcal{F}, \forall i \in \{1, \ldots, n\}$. In this work, we consider $\mathcal{F}$ to be a set of nonlinear functions ϕ, e.g. ***Sigmoid, ReLU, MaxPool, etc.***, applied to a linear transformation of the input, $i.e.$ $\mathcal{F}(z) = \{\phi(Wz) \mid \forall \phi, \forall W\}$ and denote $h^i(h^{i-1}) := \phi^i(W^i h^{i-1})$. Given a set of nonlinearities and weight matrices, a deep neural network provides a nonlinear mapping from inputs x in an input space $\mathcal{X}$ to outputs $\hat{y}$ in an output space Y, i.e.

$$\hat{y} := h(x) = \phi^n(W^n \phi^{n-1}(W^{n-1} \phi^{n-2}(\ldots \phi^1(W^1 x))))$$

In supervised classification and regression problems, we are given a data set $\mathcal{D}$ of pairs (x, y), where each pair is drawn from an unknown joint distribution. For classification, y is a categorical variable, and for regression, y is a real-valued vector. Given a dataset, network architecture, and a loss function $J(y, \hat{y})$ (e.g. cross entropy, $L2$-loss), a learning training algorithm produces a set of learned parameters $\theta := \{W^i\}_{i=1}^n$. We denote $J(\theta, x, y)$ as the loss of a learned network parameterized by θ on a data point (x, y). While we apply our methods more broadly, for simplicity in notation, we focus on classification now for introducing the method.

2.1 Adversarial Examples

Consider an input x that is correctly classified by neural network h. An adversary seeks to apply a small perturbation to x, $x^{adv} = x + \Delta x$, such that $h(x) \neq h(x^{adv})$. By ***small*** we mean that the perturbation should be minimally perceptible to a human. For perturbations applied to images, the l_∞-norm is considered a better measure of human perceptibility than more common norms like l_2, and has been studied as a natural notion of such perturbations, as put forward in Goodfellow et al. [2014b]. We assume that the adversary can access our neural network h while the manipulative power of adversary, the perturbation Δx, is of bounded norm $\|\Delta x\|_\infty \leq \lambda$.

Given a classifier, one common way to generate an adversarial example is to perturb the input in the direction that increases the cross-entropy loss. This is equivalent to minimizing the probability assigned to the true label. Given the neural network h, network parameters θ, input data x, and corresponding true output y, the adversary determines the perturbation Δx as follows

$$\Delta x = \arg\max_{r \sim \rho} J(\theta, x + r, y) \tag{1}$$

where ρ is the adversary's policy, a deterministic mapping from input x to the space of bounded (allowed) perturbations $R\ (\leq \lambda)$. Due to nonlinearity of the underlying neural network, and therefore that of the objective function J, solving the optimization Eq. 1 is difficult. Following Madry et al. [2017], Goodfellow et al. [2014b], we use the Taylor expansion of the loss function up to the first order term to get

$$\Delta x = \arg\max_{r \sim \rho} [J(\theta, x, y) + r^\top \mathcal{J}(\theta, x, y)] \qquad \text{where } \mathcal{J} = \frac{\partial J}{\partial x}$$

The first term in the optimization does not depend on the adversary's policy, therefore

$$\Delta x = \arg\max_{r \sim \rho} r^\top \mathcal{J}(\theta, x, y)$$

Given that input, label and a fixed model, the adversary finds a deterministic policy and chooses r to be in the direction of $\mathcal{J}(\theta, x, y)$, i.e. $\Delta x = \lambda \text{sign}(\mathcal{J}(\theta, x, y))$. This is the "fast gradient sign method" introduced by Goodfellow et al. [2014b]. Fig. 1 is an example of the perturbation this method creates and the misclassification it results in.

2.2 Generate Adversarial Networks

Generative Adversarial Networks (GANs), [Goodfellow et al., 2014a], are a framework for estimating generative models via an adversarial process, where a generator and a discriminator are trained simultaneously. The generator, G, aims to produce a data distribution and the discriminator, D, attempts to discriminate between the training data and the sample produced by G. In other words, the

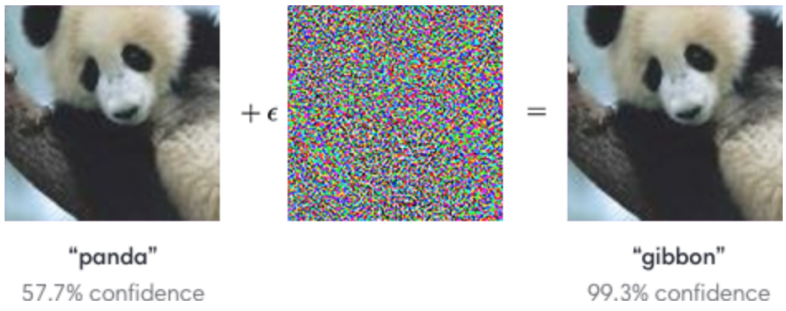


Figure 1: A demonstration of adversarial example generation using *FGSM*. The resulting change in the image is imperceptible to humans but induces a misclassification by the model.

generator attempts to produce a sample that will "fool" the discriminator, and the discriminator aims to avoid getting fooled. This framework corresponds to a *minimax* zero-sum game.

The generator, G, and discriminator, D, are both multilayer perceptrons. In order to learn the generator's data distribution p_g over some input data $\mathbf{x}$, a prior on the input noise variables, $p_z(\mathbf{z})$ is defined. G produces a data distribution based on $G(\mathbf{z}, \theta_g)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. Furthermore, the second multilayer perceptron is defined by $D(\mathbf{x}, \theta_d)$ where D is represented by parameters θ_d. $D(\mathbf{x})$ is the probability that $\mathbf{x}$ came from the data rather than from p_g. In this framework, D is trained to maximize the probability of assigning the correct label to both training examples and samples from G. Simultaneously, G is trained to minimize $\log(1 - D(G(\mathbf{z})))$. This results in a *minimax* zero-sum game with the value function $V(G, D)$ as follows

$$\min_G \max_D V(G, D) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z}))] \tag{2}$$

3 GAN Adversarial Training

Considering the defense problem from a game-theoretic perspective, the defender aims to change the model in order to minimize its maximum possible loss against the best policy of the adversary. Therefore, we can rewrite the Eq. 1 as follows:

$$\pi^* \ , \ \rho^* := \arg \min_{p \sim \pi} \max_{r \sim \rho} \mathbb{E}_{\pi, \rho} [J(M_p(\theta), x + r, y)] \tag{3}$$

where the adversary tries to maximize the loss of the defender by perturbing the input under a strategy ρ and the defender tries to minimize the loss by changing model parameters θ to a new $M_p(\theta)$ under a strategy π. The optimization problem in Eq. 3 is a *minimax* zero-sum game between the adversary and defender. Recall that the set of policies for the adversary is limited to perturbations with constrained l_∞ norm. For the defender, the policy set is constraint to those policies which preserve the overall accuracy of the model.

Using the *minimax* zero-sum game formulation, we take inspiration from GANs, and try to change it to a setting of adversarial training. We would want the GAN to consist of two parts, the adversary, A, and the defender, D. The adversary would learn to create noise for a given input data, and the defender would learn to be robust to the perturbed data.

3.1 Adversary

The adversary would be a deep neural network that learns to produce noise, given an input data, such that it is able to fool the defender. So the dimensions of the inputted data and outputted perturbed data of this network would be identical. Given the input data x, one important decision to be made is whether the adversary should create a perturbed image ($x + \Delta x$), or just the perturbation for that input data (Δx).

A crucial trait of adversarial examples is that the perturbation added is very small (bounded by the l_∞-norm in the case of images). This results in a perturbation that is imperceptible to humans. If

the perturbation that the adversary could add to the input data was not bounded, the adversary could completely change the input data, and humans might also be able to identify a change in the data.

In order to limit this power of the adversary in the GAN framework, we allow the adversary to create perturbations for a given input data with bounded l_∞-norm, which is then added to the original input data. We did this in two ways - using the *tanh* function, or centering and scaling the values. In both cases, the values returned by the adversary were in the range $[-1, +1]$. This limits the l_∞-norm of the perturbation to 1. When multiplied with a constant λ, the l_∞-norm would be limited to λ.

For input data x, true label y, loss function J, and model parameters θ, the optimization function for the adversary is

$$\max_{\Delta x} \mathbb{E}\left[J(\theta, x + \Delta x, y)\right] \tag{4}$$

where θ is adversary parameters learnt by the adversary.

3.2 Defender

The defender would be a deep neural network that learns to correctly classify input data, even if it is perturbed. The architecture of the defender would remain the same as the model that we want to improve. Here we want to essentially learn a robust defender such that it can be used in the wild reliably.

Since we want the defender to learn how to classify clean data and perturbed data, train it on both clean and perturbed data. This is similar to what happens in adversarial training.

For input data x, true label y, loss function J, and model parameters θ, the optimization function for the defender is

$$\min_{\theta} \mathbb{E}\left[J(\theta, x + \Delta x, y)\right] \tag{5}$$

where Δx is either the perturbation created by the adversary, or is 0. This is because we want the defender to learn from both clean data and perturbed data.

4 Experiments

Our experiments to evaluate the GAN addresses image classification on the MNIST dataset. To create adversarial examples in our evaluation, we use the *fast gradient sign method*, $\Delta x = \lambda \text{sign}(\mathcal{J}(M_p(\theta), x, y))$ and the adversary from our GAN after training. All perturbations are applied to the pixel values of images, which take values in the range $[0, 1]$. So the fraction of perturbation with respect to the data's dynamic range would be λ. To ensure that all images are valid, even following perturbation, we clip the resulting pixel values so that they remain within the range $[0, 1]$. In all plots, we consider perturbations of the following magnitudes $\lambda = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Values higher than this would result in drastic changes in the input data.

To evaluate models in the image classification domain, we look at two aspects: the model accuracy for varying values of λ, and the calibration of the models. When analyzing the model's accuracy, we compute the accuracy as an average over multiple forward passes. In case of the models' calibration, linear calibration is ideal, as it suggests that the model gets an accuracy that matches its confidence level.

For all different approaches, we used the same model (or defender) architecture. We used a multi-layered perceptron with 4 layers of sizes 784, 128, 64, 10, where 784 is the size of the input data, and 10 is the total number of classes. Each layer is a fully-connected layer, using *batch-norm* and *leaky-ReLU* activation. However, the last layer is a fully-connected layer, with a *softmax* output. A *cross-entropy* loss function was used.

For learning the parameters, we used *SGD*. We started with a learning rate of 1.0, momentum of 0.9, and weight decay of 0.0004. If the loss function increased at a certain epoch, we halved the learning rate. These models were run till convergence.

4.1 Vanilla model

This was the model trained on clean data, without having any condition to do well against perturbed data. It achieved an accuracy of 98.26% on the clean validation dataset.

4.2 Random noise model

This was the model trained on randomly perturbed data. The perturbations were completely random, with a bounded l_∞-norm of 0.1. It achieved an accuracy of 98.44% on the clean validation dataset.

4.3 Adversarially trained model

Adversarial training [Goodfellow et al., 2014b] has emerged as a standard method for defending against adversarial examples. It has been adopted by Madry et al. [2017], Tramèr et al. [2017] to maintain high accuracy levels even for large λ values.

At every epoch, the model was trained on 50% clean data, and 50% perturbed data computed using the *FGSM* method, with $\lambda = 0.1$. It achieved an accuracy of 98.50% on the clean validation dataset.

4.4 GAN model

This model was trained using the proposed method. The adversary for the GAN was also a multi-layered perceptron with 4 layers of sizes 784, 512, 512, 784, where 784 is the size of the input and output data (as that is the per-pixel noise generated by the adversary). Each layer is a fully-connected layer, using *batch-norm* and *leaky-ReLU* activation. However, the last layer is a fully-connected layer. As mentioned before, we looked at two ways of changing the dynamic range of this output to $[-1, +1]$. The *tanh* method, that applies the tanh function to every pixel individually. And the *scaled* method, where the pixel values are centered to be around 0, and then scaled to be in the range $[-1, +1]$.

In the case of the GAN, since both the adversary and the defender and trying to learn simultaneously, we lower the learning rate of both the adversary and the defender by half every 10 epochs. We run this model till convergence.

The *tanh* model and the *scaled* models achieved an accuracy of 98.23% and 98.29% on the clean validation dataset respectively.

4.5 Initialized GAN model

We also tried initializing the adversary of the GAN with a good set of weights. We trained the adversary alone to come up with perturbations against the learnt vanilla model. The *tanh* and *scaled* GANs were able to learn adversaries that dropped the accuracy of the vanilla model to 20.41% and 46.28% respectively. Note that this is lower than the accuracy of the vanilla model against the *FGSM* method, with the same $\lambda = 0.1$, which was 85.42%. Note that this means that we can learn better ways of perturbing data compared to the *FGSM* method, and hence the encouragement of using a GAN framework.

Initializing the GAN adversary to these set of weights, the *tanh* model and the *scaled* models achieved an accuracy of 98.23% and 98.19% on the clean validation dataset respectively.

4.6 Comparisons

We compare all the models based on the accuracies they achieve against the *FGSM* method, for varying values of λ. Fig. 2 illustrates this. While the adversarially trained model does the best almost throughout, the GAN models do better than the vanilla model and the random noise model. This is expected as the GAN adversary learns to perturb images in a way that would fool the defender, and not just return random noise. Note that initializing the GAN adversary does better than the

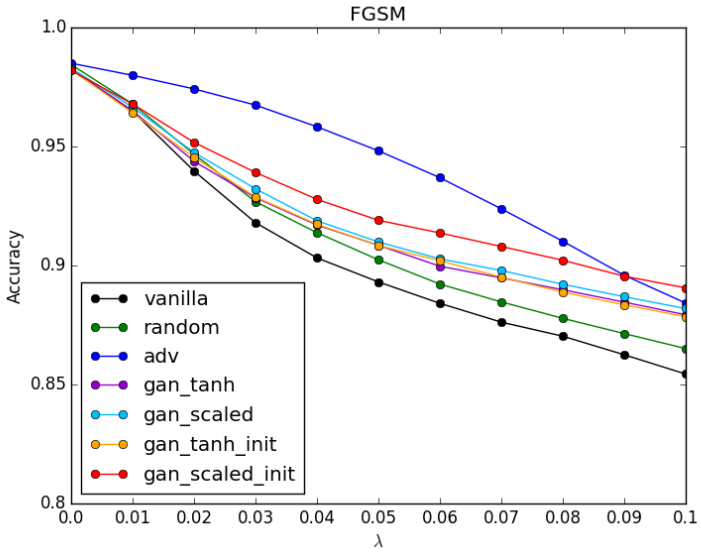


Figure 2: Accuracy-λ plots for the different models, against adversarial examples generated using *FGSM*.

non-initialized adversaries. Also note that for high λ values, the accuracies of the GAN models and the adversarially trained model is almost similar; the *scaled* initialized GAN model performs better than the adversarially trained model for $\lambda = 0.1$.

We also compare these models based on their calibration plots. Fig. 3 illustrates this. Again, we see that the adversarially trained model has the closest to linear calibration. The GAN models are more linear than the vanilla model and the random noise model. But there is no real distinction that can be made as the plots are very erratic.

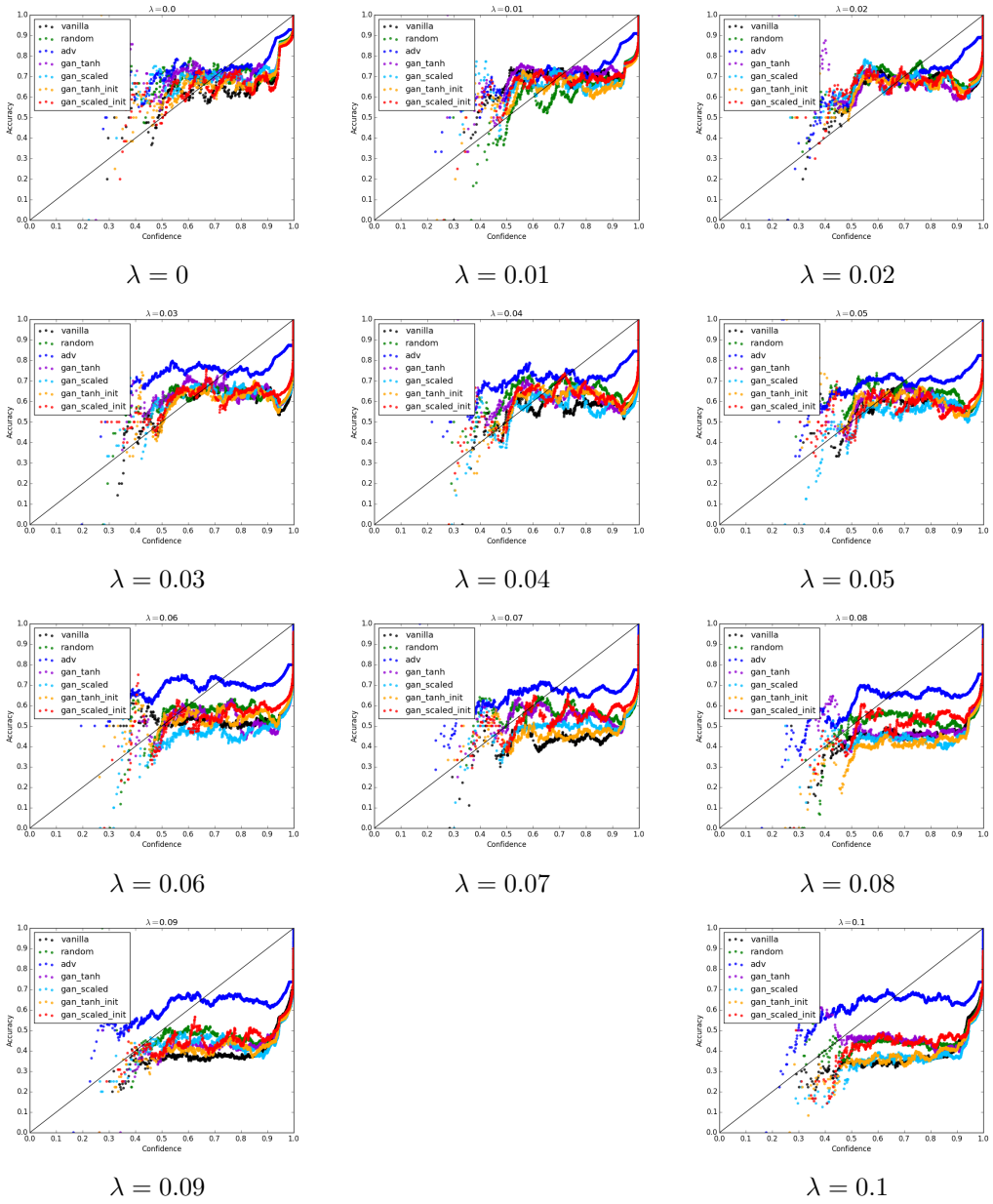
An advantage of using the GAN framework was that we could also learn an adversary that would be able to create adversarial examples. However, for all the 4 GAN models, the adversary only drop down the accuracies to $\approx 97\%$ for the different kinds of defenders.

5 Related Work

Recently, the paradigm of adversarial attacks and defenses to machine learning models have been addressed as a serious problem in the field of machine learning where creating such attacks and finding ways to defend against them are the essential core this paradigm. Goodfellow et al. [2014b] introduced the fast gradient sign method, which involves taking a step in the direction of the gradient of the loss function with respect to the input data. Kurakin et al. [2016] proposed an iterative method where fast gradient sign method is used for smaller step sizes, which leads to a better approximation of the gradient. Papernot et al. [2017] observed that adversarial examples could be transferred to other models as well. Madry et al. [2017] introduce adding random noise to the image and then using the fast gradient sign method to come up with adversarial examples.

Being robust against adversarial examples has primarily focused on training on the adversarial examples. Goodfellow et al. [2014b] use the fast gradient sign method and inject data perturbed in that way into their training dataset. Madry et al. [2017] use an iterative fast gradient sign method approach of creating adversarial examples to train on. Madry et al. [2017] introduced an ensemble adversarial training method of training on the adversarial examples created on the model itself and an ensemble of other pre-trained models. These works have been successful, achieving only a small drop in accuracy form the clean and adversarially generated data.

Other work on making models robust against adversarial examples involve training a new model, with a new objective function. One such example is the technique introduced due to Nayebi and



$\lambda = 0$ $\lambda = 0.01$ $\lambda = 0.02$

$\lambda = 0.03$ $\lambda = 0.04$ $\lambda = 0.05$

$\lambda = 0.06$ $\lambda = 0.07$ $\lambda = 0.08$

$\lambda = 0.09$ $\lambda = 0.1$

Figure 3: Calibration plots for the different models, against adversarial examples generated using *FGSM*.

Ganguli [2017], where the activation are encouraged to be saturated. In doing so, small perturbations in the input data does not effect the output much.

6 Discussion

We introduced GAN adversarial training as a method to learn a good defender and adversary in the adversarial examples problem.

The experiments demonstrate that the defender trained in this way does better than the vanilla and random noise models, but is not able to do as well as the adversarially trained model, at least for low values of λ.

Experiments also show that the adversary is not able to learn how to fool the defenders well. This result was surprising as the GAN optimization should make both the adversary and the defender better, not just the defender. Another reason for the failure of the GAN adversary is that it is catered to create perturbations for a single defender. Using it to create perturbations against other defenders might be similar to random noise perturbations.

The reason why we might not have learnt a good adversary in the GAN framework might be because of the difference in model capacities of the adversary and the defender (the adversary has a lot more parameters to learn compared to the defender). In our experiments, we tried to tackle this problem by using different learning rates for the two models, or by updating the adversary more often than the defender, but the results did not improve. This requires further research.

If we are able to improve the GAN adversary, it would in-turn also help the defender be more robust to perturbations. So this would benefit both the adversary and the defender.

References

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014a.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014b.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Aran Nayebi and Surya Ganguli. Biologically inspired protection of deep networks from adversarial attacks. *arXiv preprint arXiv:1703.09202*, 2017.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.