On Evaluating Adversarial Robustness

Jianing (Julia) Chen
Viterbi School of Engineering, University of Southern California, Los Angeles, California 90089, USA
(Dated: May 1, 2021)

Machine learning's current driving force is to generate more accurate models rather than focusing on models' security and robustness. In reality, many machine learning models are vulnerable to different adversarial attacks. This paper will discuss how to evaluate model robustness based on the three threat models and three defense models. We will also introduce evaluation metrics and give numerical examples using MNIST handwriting, MNIST fashion, and the CIFAR-10 dataset. In our attack model experiment, we confirm that ℓ_2 will bring stronger perturb on images. The range of epsilon also will be varied on the nature of the image and the distance metrics. We also confirm that both adversarial training and pixel defend models are helpful in the image classification task from the defense model experiment.

Keywords: Machine Learning, Model Robustness,Adversarial Training, Gradient Masking, Reactive Defense

I. INTRODUCTION

Neural networks have successfully implemented machine learning systems for computer vision, reinforcement learning, and many other fields. Later on, people realize that a small feature perturbation will result in an incorrect output from the machine, i.e., carefully perturbed images can make the image recognition fail; well-designed spam makes the spam detector fails to classify. Those examples can be defined as adversarial examples. The adversarial examples represent some machine learning concerns, and those examples will eventually break the neural networks. Therefore, analyzing our model's vulnerability and robustness, indeed to improve our models, will be an essential part. In this project, we aim to study the attack and defense algorithms. Our goal is to give a comprehensive overview of the mainstreaming models, both theoretically and numerically. From the theoretical perspective, we want to:

- understand the principles that why the attack and defense work in a certain problem setup.
- understand the advantages and disadvantages of different defense models.

We will implement and compare the methods from the numerical perspective, aiming to demonstrate our previous theoretical analysis

II. EVALUATION METRICS

Before we dive into how to boost the robustness of a neural network, we must answer what robustness is. We want to make sure that a network classifier can lead to the correct decision when the input data is slightly perturbed. Specifically, ideally, the perturbed data should be invisible for the human being to detect but sensitive to the classifier. Then we come across the question of how to measure the perturbation that humans cannot detect but fool the machine.

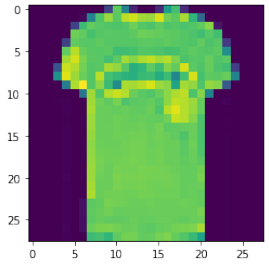

FIG. 1. Fashion MNIST Image. Original T-shirt image.

We will use ℓ_p norm to determine the perturbations on the input data. Evaluating the norm itself is an open question, and choosing the different norm will have a noticeable effect on adversarial examples and model robustness. This section will introduce different ℓ_p norm evaluation metrics, also known as distance metrics.

Now, consider an adversarial examples as $\tilde{\boldsymbol{x}}$, and the origin data as $\boldsymbol{x}$ respect to different ℓ_p norm metrics. Then metrics can be written as $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_p$.

- ℓ_1 norm is the absolute distance between adversarial examples and the real pixels. It yields sparse adversarial attacks so that only a few pixels will be perturbed.
- ℓ_2 norm measures the Euclidean distance between real input pixel and the adversarial example. It can remain small when there are many small changes to many pixels.
- ℓ_∞ norm is the most commonly used one. It measures the maximum distance between the real input pixel and the adversarial example.

The FIG. 2 shows adversarial examples with different norm constraints formed via the fast gradient sign method. Compared to FIG. 1 which we cannot tell much difference from ℓ_1, ℓ_∞ and the original T-shirt image, but the ℓ_2 perturbation is visible. It is because it limits the absolute change that can be made to any pixels.

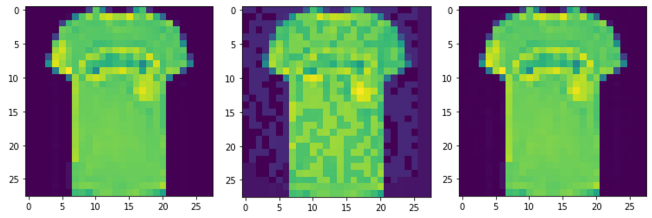

FIG. 2. Fashion MNIST Image. Left: ℓ_1 norm with 0.01 ϵ perturbations. Middle: ℓ_2 norm with 0.01 ϵ perturbations. Right: ℓ_∞ norm with 0.01 ϵ perturbations.

III. THREAT MODEL: SEARCHING ADVERSARIAL EXAMPLES

Now we will move to how to generate the adversarial examples. Learning the formation of adversarial examples will help us prevent them when we design networking to improve the model robustness.

The adversarial example concept was first introduced in the paper of Szegedy et al.[8] and defined as the perturbations found by optimizing the input to maximize the prediction error. For example, add some perturbations to a cat's picture, then the machine will recognize it as a dog. Study on the threat models is basically referred to fabricate the perturbations. In this section, we introduce three popular threat models to a neural network.

A. Fast Gradient Sign Method [3]

Fast Gradient Sign Method (FGSM) for generating adversarial images was invented by Goodfellow et al. in 2014, which adds the error to cost function by using the neural network's gradients. The original data or images multiplied by the error (ϵ) to ensure the perturbations are small. The direction of the error is the same as the gradient of the cost function. That is to say, we intentionally add errors in the direction of the gradient to make the machine fail by maximizing the loss.

This can be summarised using the following expression below:

$$\tilde{J}(\boldsymbol{\theta}, \boldsymbol{x}) \approx J(\boldsymbol{\theta}, \boldsymbol{x}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\intercal \nabla_{\boldsymbol{x}} J(\boldsymbol{x}) \tag{1}$$

$$\begin{aligned} &\text{maximize } J(\boldsymbol{\theta}, \boldsymbol{x}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\intercal \nabla_{\boldsymbol{x}} J(\boldsymbol{x}) \\ &\text{subject to } \|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|_\infty \leq \epsilon \end{aligned}$$

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x})) \tag{2}$$

Where θ is the model parameters, $\boldsymbol{x}$ is the input data or images, $\tilde{\boldsymbol{x}}$ is the generated adversarial examples, ϵ is the error. Therefore, the equation (1) represents the loss function of data. The equation (2) shows how we can generate the adversarial examples.

B. Projected Gradient Descent [6]

Projected Gradient Descent (PGD) attack approach considered an inner saddle point formulation. PGD attack model has a very similar model representation as FGSM. The idea behind this is to make sure that points are feasible by projecting onto the set of input data, so by using the PGD method, we might maximize the inner objective a bit more carefully than using FGSM.

We can generate the adversarial examples by using the expression (3) below:

$$\tilde{\boldsymbol{x}} = \Pi_{\boldsymbol{x}+S}(\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}))) \tag{3}$$

Where $\Pi_{\boldsymbol{x}+S}$ denotes the projection onto the input data set $\boldsymbol{x}$, and call out a set of allowed perturbation $S \subseteq \mathcal{R}^d$. The whole attack representation is saying for each input data $\boldsymbol{x}$, we will introduce perturbation to the input first, then project it onto $\boldsymbol{x} + S$.

C. Carlini & Wagner method [1]

In 2013, Szegedy et al. [8] used optimization problem to craft adversarial examples which can be formulated as:

$$\begin{aligned} &\text{minimize } \mathcal{D}(\boldsymbol{x}, \boldsymbol{x} + \delta) \\ &\text{such that } C(\boldsymbol{x} + \delta) = t \text{ and } \boldsymbol{x} + \delta \in [0, 1]^n \end{aligned}$$

Again, $\boldsymbol{x}$ is the input image data, δ is the perturbations (δ equals to $\tilde{\boldsymbol{x}} - \boldsymbol{x}$) and $\mathcal{D}(\boldsymbol{x}, \boldsymbol{x} + \delta)$ is the distance metric which measures the distance between the adversarial example and real input image. The formulation has two constrain functions, the first one (C) is a classifier function on adversarial image and real image. It will be hard to solve because it is not a straight forward linear function. Therefore, Carlini and Wagner express the constraint in a different form:

$$\begin{aligned} &\text{minimize } \mathcal{D}(\boldsymbol{x}, \boldsymbol{x} + \delta) \\ &\text{such that } f(\boldsymbol{x} + \delta) \leq 0 \text{ and } \boldsymbol{x} + \delta \in [0, 1]^n \end{aligned}$$

Noticed that the f is the objective function. Carlini and Wagner, then, reformulated the original optimization problem by moving the objective function from constaint function into the minimization function. It will not change the final result, but will only have one constraint function left and they also compose the distance metric $\mathcal{D}(\boldsymbol{x}, \boldsymbol{x} + \delta)$ into $\|\delta\|_p$, so the the problem becomes to:

$$\begin{aligned} &\text{minimize } \|\delta\|_p + c \cdot f(\boldsymbol{x} + \delta) \\ &\text{such that } \boldsymbol{x} + \delta \in [0, 1]^n \end{aligned}$$

Solve the equation above, we will get:

$$\tilde{\boldsymbol{x}} = \min_{\boldsymbol{x} \in [0,1]^n} c \cdot f(\boldsymbol{x} + \delta) + \|\delta\|_p \tag{4}$$

Where c is the constant and should be greater than 0. Carlini and Wagner also point out the value of c will determine the effectiveness of the attack. So they resort to using binary search to figure out a value of c. In addition, Carlini and Wagner also proposed three models under the different distance measures.

The Carlini & Wagner method (C-W method) is more robust and has more elaborate effects. The creation of this method is purposed to generate adversarial examples with less distortion.

IV. DEFENSE MODEL

The finding of adversarial examples essentially helps us to improve our model. We will call those protective models defense models. The mainstream method is to incorporate adversarial examples during the training process, known as adversarial training. But there are more defense models. After Szegedy et al.'s paper was published, many defense models have proposed solidifying networks against attacks. Still, these defenses are quickly broken as the new attack models come out. For instance, adversarial training with FGSM examples[3] was shown to be vulnerable to multiple stage attacks[4]; and the defensive distillation does not help to increase the robustness of machine learning models in some case[2].

Adversarial examples are hard to defend because of the difficulty of constructing a theoretical model, describing the solutions for non-linear and non-convex optimization problems. This section introduces the popular defense models that have shown to benefit neural network robustness and their limitations.

A. Adversarial Training[3]

Adversarial training by Goodfellow is (possibly) the most popular robust training method, and it successfully thwarted the fast gradient sign method (FGSM). It solves a min-max robust optimization problem to minimize the worse-case loss with perturbed data. So we can define our problem as:

$$\min_{\theta} \mathop{\mathbb{E}}_{x \in S} \left(\max_{\|\delta\| \leq \epsilon} J(\boldsymbol{\theta}, \boldsymbol{x} + \delta) \right) \tag{5}$$

Which is equivalent to a regularization problem, and we will have to a minimum the overall loss function as incorporating the adversarial example. So it becomes:

$$\min_{\theta} \big(\alpha \cdot J(\boldsymbol{\theta}, \boldsymbol{x}) + (1 - \alpha) \cdot J(\boldsymbol{\theta}, \tilde{\boldsymbol{x}}) \big) \tag{6}$$

$$\text{where } \tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \text{sign}\big(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x})\big)$$

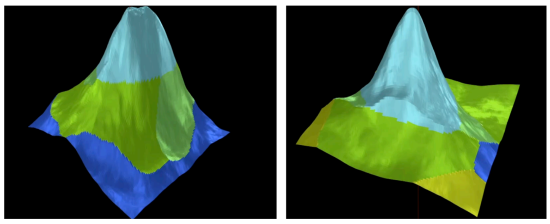

FIG. 3. Loss surface. Left: a standard neural network. Right: a neural network trained with gradient masking.

We can solve the optimization expression (6) by repeatedly computing the gradient with respect to parameters $\boldsymbol{\theta}$.

The idea of adversarial training is very straightforward, it requires us to find the adversarial example first. Now adversarial training based defense is susceptible to multiple stage attacks[4] and blind-spot attack [9].

B. Gradient Masking [5]

Liu et al.in 2018 proposed to add randomness noise which is generated by Gaussian distribution into each neural network layer. Use the convolutional neural network as an example; the noise layer before the first convolution layer the "init-noise", another noise layer is called "inner-noise". The idea of this algorithm can be formulated as:

$$\min_{\theta} J_{\epsilon}(\boldsymbol{\theta}, \boldsymbol{x}) \tag{7}$$

The original loss function is $J(\boldsymbol{\theta}, \boldsymbol{x})$, after add the noise ($\epsilon$) into each layer during the training process.

The gradient masking methods actually defend the searching of adversarial example by noising the loss surface. As shown in FIG. 3.

The loss surface becomes more grained with the noise layer. Therefore, when the attacker uses gradient-based methods to search the adversarial examples, the attacker will likely fall into the local maximum and fail to find real sensitive examples.

However, Nicolas [1] has shown that this kind of gradient masking dense method is "broken". The defense can be easily fooled by re-smoothing the loss surface. Nicolas proposed a method to train a new network to approximate the one with noisy layers. Then the adversarial examples searched from the new network can easily fool the one with noisy layers. Since the authors have not published their algorithm code, we will introduce it, but we exclude this defense model in the experiment.

C. Reactive Defense(Pixel Defend) [7]

A reactive defense method tries to reject or pre-process the input data that may cause misclassification. Song et al. in 2018 proposed an image purification method (PixelDefend) that can approximate the training image distribution using a PixelCNN model. PixelCNN model is a generative model with the tractable likelihood for images. This method can work with other adversarial defense techniques and will not change the classification model. Based on the distribution of images and adversarial examples, we can notice that most adversarial examples lie in low probability regions. The detailed algorithm for this defense are:

Algorithm 1: PixelDefend

Result: Purified Image $\boldsymbol{x^*}$
for *Every pixel* $\boldsymbol{x}$ **do**
  Set range R for purifying a pixel;
  Approximate the clean pixel distribution via a generative model $p(x^*)$;
  Update the purified pixel $\boldsymbol{x^*}$ based on R and $p(x^*)$.
end

The limitation of PixelDefend also very obvious since we have to manipulate every pixel on the image, so the algorithm will be pretty computational.

V. NUMERICAL EXAMPLES

The idea of my experiment to measure each attack model and defense model. I will test the attack model on three different datasets from easy to complex, which means from easy structured images to harder structured, such as more complicated patterns and colorful images. Due to the hardness of designing the algorithm, I used adversarial-robustness-toolbox (ART) for both attack and defense models. Besides, the baseline model was built by using TensorFlow Keras.

A. Data set

- MNIST handwriting dataset, it is the easiest dataset for the neural network to classify. In our case, we will only choose the samples with number 0 and number 1.
- MNIST fashion dataset, which consists of 60,000 28x28 grayscale images. We will choose two classes, that is the T-shirts and pants.
- CIFAR-10 dataset consists of 32x32 colorful images. We will also choose two classes of data from it. So we have one class of automobiles and one class of airplanes.

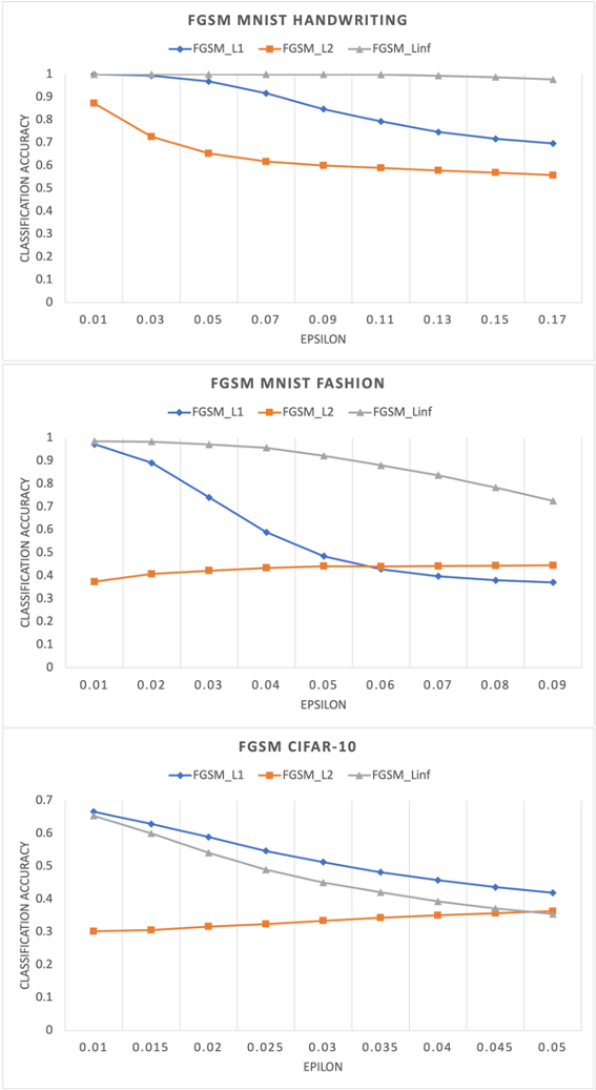

FIG. 4. FGSM Attack on Three Image Datasets.

B. Experiment on Attack Models

The main goal of studying the three attack models is to learn the epsilon(ϵ), which is the perturbations, and learn the model's accuracy by using epsilon under different norm evaluations metrics. At the beginning of the experiment, we first will use a simple convolutional neural network (CNN) to build an image classifier based on the training image. Then we will apply each attack model technique to disturb the testing image pixels. The measurement metric we will use here is the classifier's accuracy fitted on the manipulated testing image.

On FGSM attack and PGD attack, I use ℓ_1 norm, ℓ_2 norm and ℓ_∞, but on C-W attack, I only apply the ℓ_∞ norm because the ℓ_2 norm evaluation does not take the epsilon into account. According to the paper[1], C-W ℓ_2 attack should be the most effective one. We exclude this in the experiment only because it does not have epsilon,

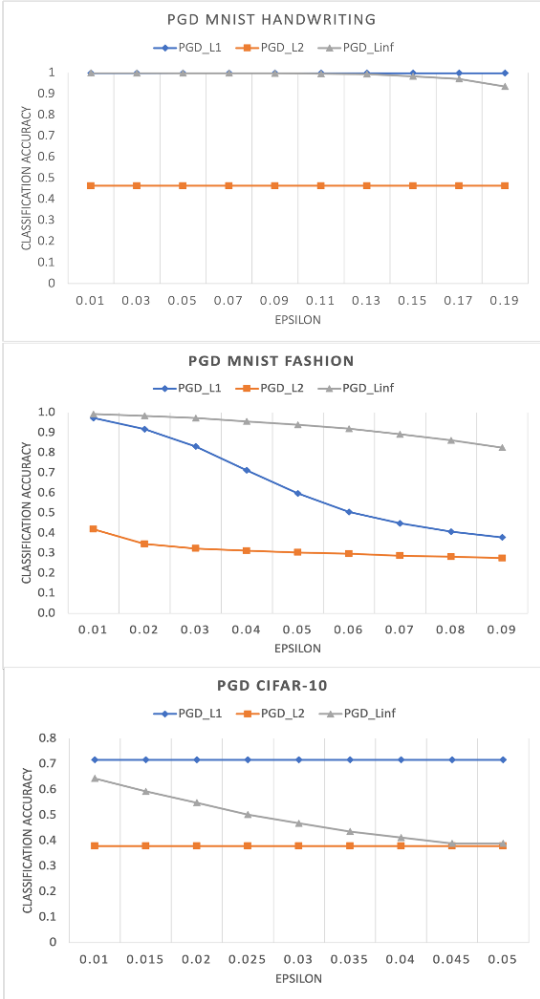

FIG. 5. PGD Attack on Three Image Datasets.

MNIST Handwriting					
eps	PGD_L1	PGD_L2	PGD_Linf	eps	PGD_L2
0.01	0.9986	0.4643	0.9995	0.001	0.9981
0.03	0.9986	0.4643	0.9991	0.0014	0.9953
0.05	0.9986	0.4643	0.9986	0.0018	0.9882
0.07	0.9986	0.4643	0.9986	0.0022	0.974
0.09	0.9986	0.4643	0.9981	0.0026	0.9409
0.11	0.9986	0.4643	0.9957	0.003	0.8463
0.13	0.9986	0.4643	0.9939	0.0034	0.643
0.15	0.9986	0.4643	0.9835	0.0038	0.5083
0.17	0.9986	0.4643	0.9712	0.0042	0.4723
0.19	0.9986	0.4643	0.9352	0.0046	0.4709

FIG. 6. PGD Attack on MNIST Handwriting Image Accuracy

so it will not help learn the relationship between epsilon and classification accuracy.

C. Results on Attack Models

The result of the FGSM attack method shown on FIG. 4 and the result of the PGD attack method shown on FIG. 5. Few things we can conclude from this figure are:

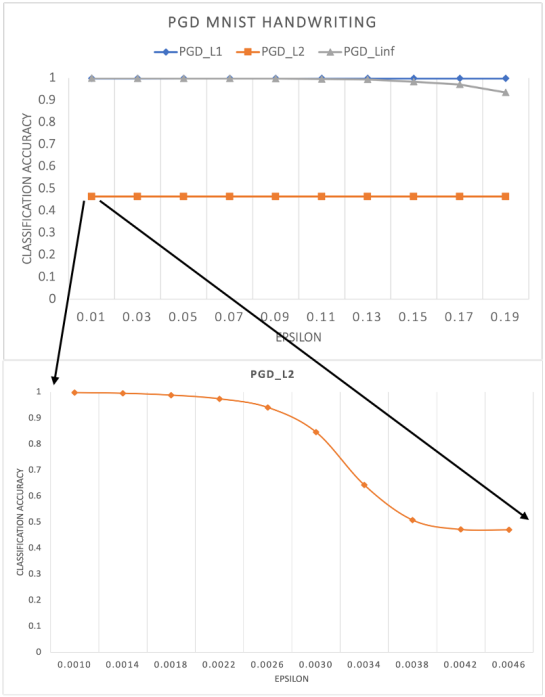

FIG. 7. Visualization of ℓ_2 norm on PGD Attack on MNIST Handwriting Image Accuracy

- For both FGSM and PGD attack models, by looking at the x-axis, as the difficulty of classifying image increase, the epsilon will decrease to ensure the perturbations are small. In the MNIST handwriting dataset, the range of epsilon is [0.01, 0.17], in MNIST fashion, the range of epsilon is [0.01, 0.09], and in CIFAR-10, the range of epsilon is [0.01, 0.05].

- On FGSM (FIG. 4) MNIST handwriting table, we can see the ℓ_2 norm is keeping decrease, but on other two datasets, the ℓ_2 norm appears to increase. Because on MNIST fashion and CIFAR-10 datasets, the first ℓ_2 is too large, which already broken the CNN model to distinguish the labels. Therefore the model starts guessing the label later on. What we can observe from on PGD (FIG. 5), even though we will not see the increase of accuracy on ℓ_2 norm, but we can see at every beginning, accuracy has been reached its minimum.

- On the PGD method, it is harder to make the same range of epsilon to capture the change of accuracy for each norm. For example, FIG. 6, on MNIST handwriting dataset, range of epsilon [0.01,0.19] on ℓ_1 norm have no effect on the classification accuracy at all (0.9986 accuracy), but the same range of epsilon on ℓ_2 norm has the lowest accuracy of 0.4643. But with ℓ_∞ norm, the accuracy is keeping decrease. In addition to see how accuracy changes, I add another series of epsilon On FIG. 6. What we can see that from 0.001 to 0.0046, the accuracy

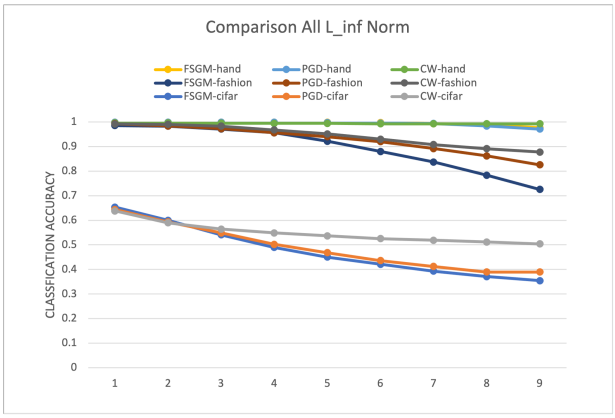

FIG. 8. Visualization of All ℓ_∞ Norm

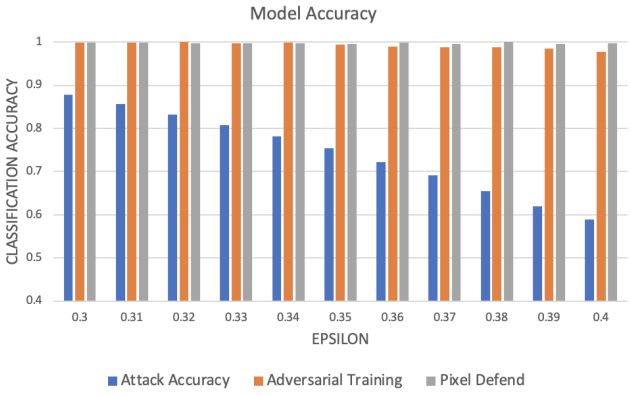

FIG. 9. Visualization of Defense Models: Adversarial Training and Pixel Defend

decrease directly from 0.9981 to 0.4709. (also visualized on FIG . 7)

- The C-W method with ℓ_∞ norm has smaller distraction on images compared with other two methods (refer FIG. 8). But it does not say the method is not good. Oppositely, the smaller and invisible attack is what we are looking for.

In summary, from both the FGSM attack and PGD attack, we can conclude that the ℓ_2 norm will bring a more substantial effect than the other two distance metrics. While on PGD, we saw that both ℓ_2 , ℓ_∞ between pixel values could easily use to cause an in-distinguishable change on harder images. Conversely, the ℓ_1 distance corresponds to a sparse attack in which only a few pixels are significantly manipulated, yielding the worst result.

D. Experiment on Defense Models

Now we have the perturbed images from previous experiments. Then we will work on the FGSM ℓ_∞ norm model on MNIST handwritten dataset, and apply both adversarial training and reactive defense to see the results.

E. Results on Defense Models

The result for two Defense Models shows on 9. What we can see is that as the epsilon increase, the model accuracy will decrease. Two defense models are built based on a distracted model, and we observe that model accuracy a lot.

On the adversarial training algorithm, there is a parameter called ratio. The ratio measures the proportion of samples in each batch to be replaced with their adversarial counterparts. Setting this value to 1 allows training only on adversarial samples. In my experiment, I set this to be 0.1, which means my adversarial training model will be training more on the entire image.

Another thing we can see from the plot, but not very obvious, is that as the image has perturbed worse, the defense model, even though it will increase the accuracy, but the defense accuracy has been decreasing over time.

VI. CONCLUSION

In the final analysis, in the theoretical part, I have shown three attack models: Fast Gradient Sign Method, Projected Gradient Descent from a mathematical perspective. Then I also introduce three defense models which can potentially increase the robustness of image classification: Adversarial Training, Gradient Masking, and Pixel Defend. Due to the lack of source on Gradient Masking, I only incorporate the other two methods.

There are few things we can conclude from the attack models experiment:

- the range of epsilon also will be varied on the nature of the image and the distance metrics.
- the distance metric ℓ_2 norm measures the Euclidean distance between real input pixel and the adversarial example is stronger than ℓ_1 and ℓ_∞.

From the defense model experiment, we can confirm that both Adversarial Training and Pixel Defend are helpful to increase the robustness of original models. Still, it is hard to tell which method will be better in my case.

VII. CODE AVAILABILITY

Work can be find on Github.

VIII. ACKNOWLEDGMENTS

I would like to give my special thanks to Professor Marcin Jaroslaw Abram and TA and CPs for their

tremendous help in this class and teaching assistants for their support. Also, thank the classmates who did the peer review on my project and I would very appreciate the comment.

---

[1] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[2] Nicholas Carlini and David A. Wagner. Defensive distillation is not robust to adversarial examples. *CoRR*, abs/1607.04311, 2016.

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

[4] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[5] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 369–385, 2018.

[6] 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.

[7] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *CoRR*, abs/1710.10766, 2017.

[8] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[9] Huan Zhang, Hongge Chen, Zhao Song, Duane Boning, Inderjit S. Dhillon, and Cho-Jui Hsieh. The limitations of adversarial training and the blind-spot attack, 2019.