IFT 6164 - Lecture 2
Adversarial Examples (Part 2)

This version of the notes has not yet been thoroughly checked. Please report any bugs to the scribes or instructor.

Scribes
Winter 2022: [Mehrnaz Mofakhami]
Winter 2021: [Mathieu Godbout, François Mercier, Sharath Chandra Raparthy]

Instructor: Gauthier Gidel

1 Summary

In the previous lecture, we talked about adversarial examples—both targeted and non-targeted—and learned how to compute them to attack a model. We also learned about different thread models and what kind of information the attacker has in each of them. In this lecture, we will discuss an interpretation of the source of the vulnerability of neural networks to adversarial attacks introduced in the *Adversarial Examples Are Not Bugs, They Are Features* paper [4], which proposes that adversarial examples exist due to *non-robust features* which are highly predictive but imperceptible to human eyes [2]. In the end, we will learn about three defense mechanisms against adversarial attacks.

2 Generalization vs. Robustness

In empirical risk minimization we find $\hat{\theta}$ such that:

$$\hat{\theta} \in \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i)$$

Generalization is about having a small error on new datapoints that are from the *same* distribution as the training set, while *Robustness* is about having a small error on datapoints that are close to the training set. In other words, we have good generalization when $\mathbb{E}_{(x,y)\sim p_{data}} \ell(f_{\hat{\theta}}(x), y)$ is relatively low, and we have good robustness when $\frac{1}{n}\sum_{i=1}^{n} \max_{\|x_i' - x_i\| \leq \epsilon} \ell(f_{\hat{\theta}}(x_i'), y_i)$ is relatively small.

There is a trade-off between generalization and robustness because of the existence of non-robust features, that are highly predictive—thus useful for generalization to unseen data—but have poor performance in terms of robustness. For example, there might be a small patch in images in the train and test set that is imperceptible to humans but indicates the class, so if the model only cares about having low test error, it will use the patch to classify the images and can have very high accuracy on test data. However, an attacker can manipulate the images by changing that small patch, hence generating adversarial examples that are very close to the original samples but will fool the classifier. So in order to increase robustness, we should remove these non-robust features, which will in turn decrease test accuracy as they are correlated with the label. We will talk about non-robust features in more detail in the following sections.

3 Two perspectives for adversarial examples

3.1 Perspective 1: Adversarial examples are bugs!

In the usual perspective to explain the existence of adversarial examples, the features of the model are twofold: *Useful* features that are responsible for good classification and *Useless* features that are not predictive of the label but the

model is unreasonably sensitive to them and can be manipulated to create adversarial examples. From this perspective, adversarial examples are bugs, because they are the consequence of useless features learned for classification.

Hypothesis 1: Adversarial examples are bugs and correspond to features that are useless for learning.

Experiment 1: Now let's look at an experiment to test this hypothesis:
Consider a binary classification task on a cat/dog dataset. For each image in the original dataset, we compute the corresponding adversarial example and change the label. So for each image of a dog, the corresponding sample in the new dataset (call it $\hat{\mathcal{D}}_{NR}$) is the adversarial example generated from that dog that is labeled as a cat, so it looks very similar to a dog but has small perturbations that fool the model to classify it as a cat. We then train a new classifier on this new train set but evaluate it on the original test set (with dogs labeled as dogs and cats labeled as cats). Figure 1 shows two possible results. Blue bars show the accuracy of the model on the test set and red bars show the accuracy of the model when attacked by an adversary. If the hypothesis is true, we should observe figure 1. (a) since the only (possibly)-useful information in dog-looking images labeled as cats lies within the adversarial perturbations which we are assuming to be useless for learning, so the model should not learn anything. However, what we actually observe is figure 1. (b) which shows high accuracy on *original, unmodified* test set (blue bar), though it is not robust (red bar).

Conclusion: The above experiment demonstrates that adversarial perturbations correspond to features that are meaningful for classification, which we call *non-robust* features. The model learns to leverage the non-robust features of a cat (that exist in adversarial examples generated from images of dogs) to label the image as a cat, so it will work well on the original test set. However, because it focuses on non-robust features it is not robust against attacks. Figure 2 summarises this experiment.

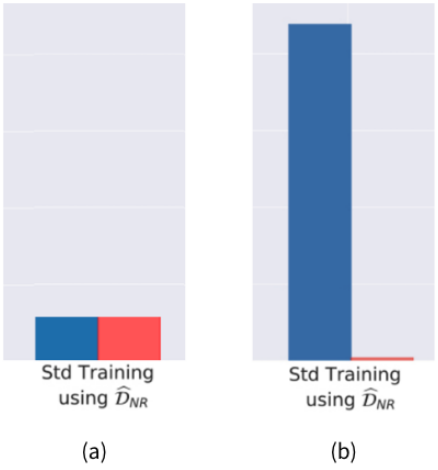


Figure 1: Two options for Experiment 1

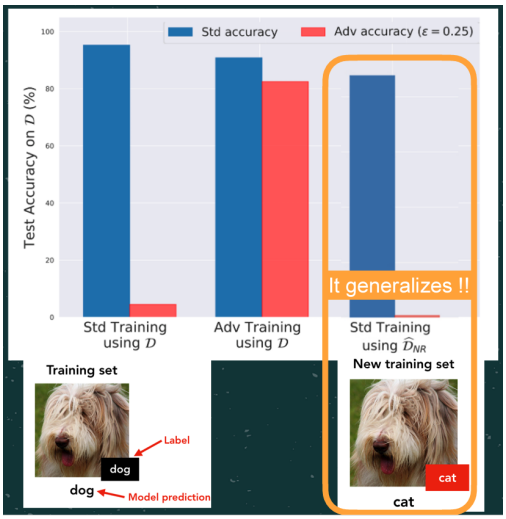


Figure 2: (Left) Results of training using the original dataset. (Middle) Results of adversarial training which will be discussed in section 5.2 (Right) Results of training on the new training set, $\hat{\mathcal{D}}_{NR}$. We can see that with training on $\hat{\mathcal{D}}_{NR}$, the accuracy on the test set drops a little because the model is using only the non-robust features. Also, the accuracy on the adversarial dataset is lower using $\hat{\mathcal{D}}_{NR}$ since non-robust features are very sensitive to perturbations.

3.2 Perspective 2: Adversarial examples are not bugs, they are features!

In an alternative perspective motivated by experiment 1, the features of a model are divided into three groups: *Useless*, *Robust* and *Non-robust*, as depicted in figure 3. Robust features are the ones correlated with the label that remain in the image even in the presence of an adversary, but non-robust features are patterns that have no meaningful information to humans but are actually highly predictive of the label. These non-robust features, however, can be manipulated by an adversary.

Remark 1. From this perspective, non-robust features are useful for increasing test accuracy, so if we eliminate non-robust features to make the model robust, the test accuracy will drop and this highlights the trade-off between generalization and robustness.

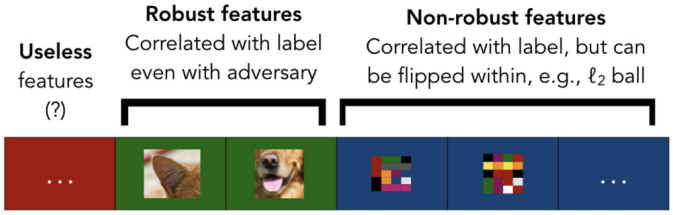


Figure 3: The perspective in which the adversarial examples are not bug, but features!

An additional experiment shows the behavior of the model when trained only on *Robust features* [4]:

Experiment 2: In this experiment, we construct a new dataset $\hat{\mathcal{D}}_R$ from the original dataset $\mathcal{D}$ that only contains robust features of images and train a network on $\hat{\mathcal{D}}_R$. Std Training using $\hat{\mathcal{D}}_R$ in Figure 4 shows the result of this experiment; the model is good at generalization to test data (high blue bar) and significantly more robust (shown by the red bar) than the original classifier (Std Training using $\mathcal{D}$) and the classifier trained only using the non-robust features (Std Training using $\hat{\mathcal{D}}_{NR}$).

Figure 5 summarizes these two experiments. For information on how to generate images that only contain robust or non-robust features, look at section 3 of [4].

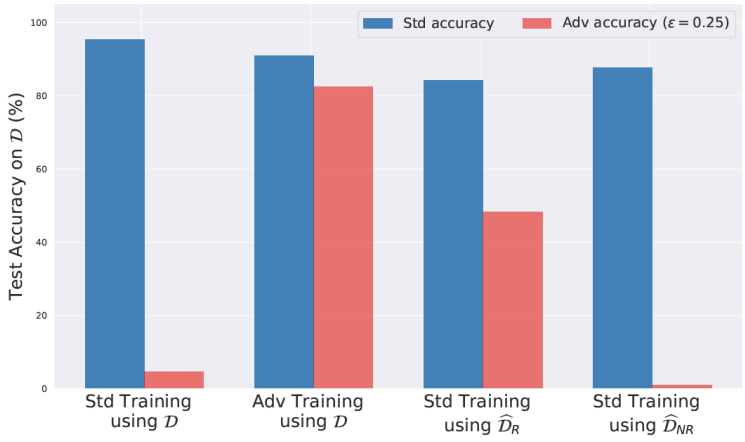


Figure 4

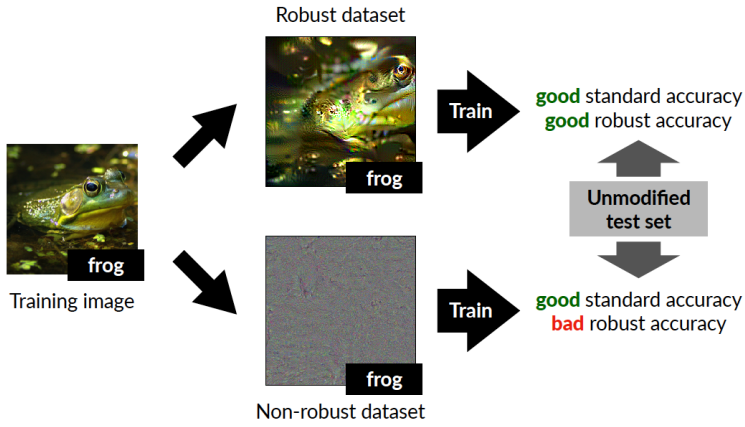


Figure 5: Summarizing experiments 1 and 2

4 Transferability can arise from non-robust features

One of the most intriguing properties of adversarial examples is that they transfer across models with different architectures and independently sampled training sets. Here, we show that this phenomenon can in fact be viewed as a natural consequence of the existence of non-robust features. Recall that, according to the main hypothesis, adversarial examples can arise as a result of perturbing well-generalizing, yet brittle features. Given that such features are inherent to the data distribution, different classifiers trained on independent samples from that distribution are likely to utilize similar non-robust features. Consequently, an adversarial example constructed by exploiting the non-robust features learned by one classifier will transfer to any other classifier utilizing these features in a similar manner [4]. Figure 6 illustrates the transferability of adversarial examples generated by ResNet-50 to different architectures. The x-axis shows the test accuracy of models when trained on a dataset containing only non-robust features of Resnet-50. This experiment suggests that architectures that learn better from this training set (i.e. have higher performance on the standard test set) are more likely to learn similar non-robust features to the original classifier (ResNet-50), so are more susceptible to transfer attacks, and this supports the hypothesis that adversarial transferability arises from utilizing similar *non-robust features*.

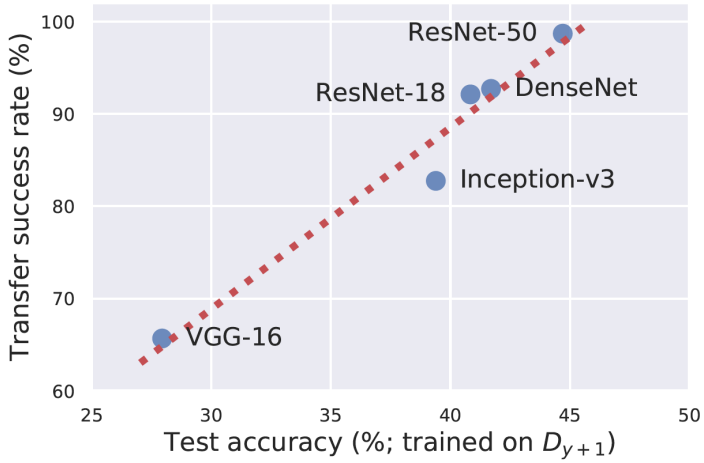


Figure 6: Transfer rate of adversarial examples from a ResNet-50 to different architectures alongside test set performance of these architecture when trained on a dataset containing only non-robust features of Rensen-50.

5 Defences

5.1 Gradient Masking

Gradient Masking is a defense mechanism that constructs a model that does not have useful gradients—either by making the model discontinuous so the gradient would not be computable or by creating ”bad” local maxima—so attacks that use gradients will not work. The idea is to create a local maximum that is very close to the benign example (point (0,0) in Figure 7) such that if we do local maximization on the loss function (shown by the blue surface) we will end up with an adversarial example that has a loss close to the benign example. However, gradient masking can be beaten in the following ways:

1. One way to get away from the local maxima is to first take a random step from the initial point and then do gradient ascent as shown in Figure 8, so we will not get stuck in the local maxima.

2. If the defended model is piece-wise constant and not differentiable, we can find a smooth surrogate and compute gradient on this smooth approximation to find the adversarial example. (Figure 9)

These methods show gradient masking is not a very effective defense. Moreover, from the point of view that states transferability can arise from non-robust features, we can craft adversarial examples using another model and transfer them to the model with masked gradients.

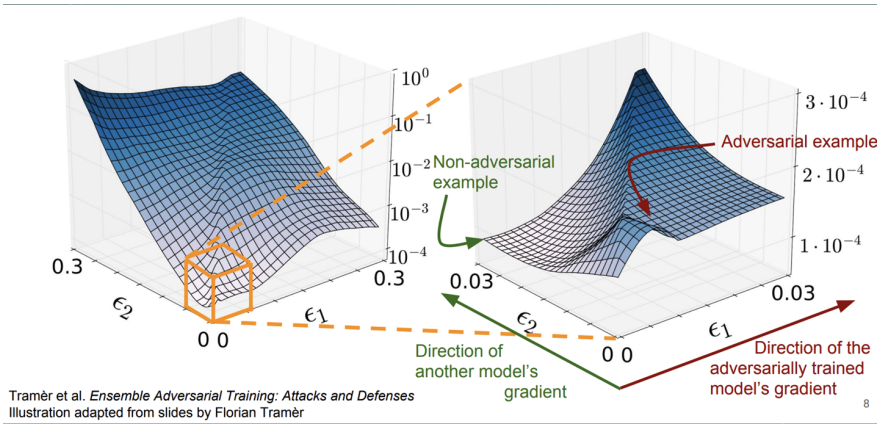
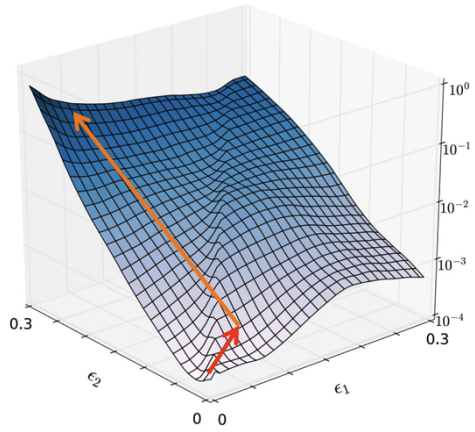


Figure 7



Tramèr et al. *Ensemble Adversarial Training: Attacks and Defenses*
Illustration adapted from slides by Florian Tramèr

Figure 8

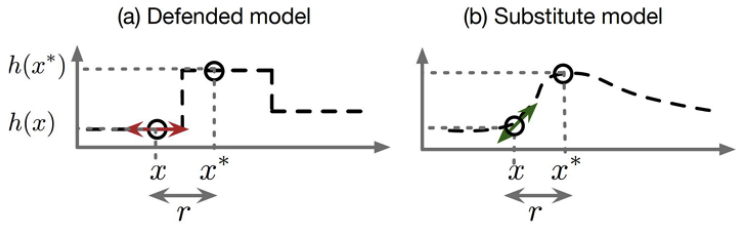


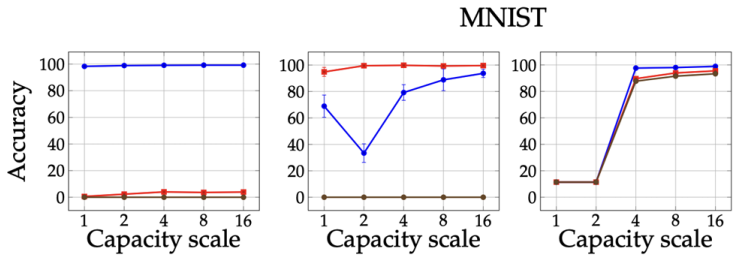
Figure 9

5.2 Adversarial Training

The most popular defense against adversarial attacks is adversarial training, whose idea is to train the model against its own adversarial examples (equation 1).

$$f^* \in \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(x,y)\sim p_{data}}\big[\max_{\|x'-x\| \leq \epsilon} \ell(f(x'), y)\big] \tag{1}$$

At each iteration within adversarial training, we sample a datapoint, create an adversarial example against the current model, and train the model on that. So as we train the model, the adversarial examples change.
In adversarial training, we ask the model to fit the original data and all the datapoints in a small neighborhood around it, so we need more capacity for a robust model (for more information, see [1]). Figure 10 shows an interesting experiment on MNIST confirming the importance of capacity in robust models. Blue lines show the accuracy of the model on standard test data, red lines show robust accuracy against FGSM attacks, and brown lines show robust accuracy against PGD attacks. Recall that FGSM follows $x' = x + \epsilon\,\text{sign}(\nabla_x l(f(x), y))$ and PGD computes several steps of this update to find the adversarial example. In standard training (a), the model has nearly 100% accuracy on the standard test set, but zero accuracies on adversarial examples crafted with FGSM or PGD. If we use the model that has been trained adversarially against adversarial examples crafted with the FGSM attack (b), the model has good standard accuracy and robust accuracy against FGSM, but very low performance against the PGD attack since PGD is much stronger than FGSM. The blue line is lower in (b) than in (a) because we have to sacrifice a bit the accuracy on test data to have a robust model. We can also see that as the capacity increases, the model gets better. If we train against PGD (c), the model performs well on the standard test set and at the same time is robust against FGMS and PGD attacks with high enough capacity. For models with low capacity (1 or 2), the attack is too strong that the model cannot learn anything, but as the capacity increases the behavior shifts as seen in the plots.



CIFAR10

	Simple	Wide	Simple	Wide	Simple	Wide
Natural	92.7%	95.2%	87.4%	90.3%	79.4%	87.3%
FGSM	27.5%	32.7%	90.9%	95.1%	51.7%	56.1%
PGD	0.8%	3.5%	0.0%	0.0%	43.7%	45.8%
	(a) Standard training		(b) FGSM training		(c) PGD training	

Figure 10

Regularization perspective In regularization perspective proposed by Goodfellow et. al [3], the following objective is minimized:

$$\tilde{J}(\theta, \boldsymbol{x}, y) := \alpha J(\theta, \boldsymbol{x}, y) + (1-\alpha) J(\theta, \boldsymbol{x} + \epsilon \operatorname{sign}(\nabla_{\boldsymbol{x}} J(\theta, \boldsymbol{x}, y)), y) \tag{2}$$

for $J(\theta, \boldsymbol{x}, y) := \ell(f_\theta(\boldsymbol{x}), y)$. $\alpha = 1$ corresponds to standard empirical risk minimization, and $\alpha = 0$ corresponds to adversarial training, i.e. minimization only on adversarial examples.
In the case of logistic regression, adversarial training is equivalent to adding some regularization to the cost function. In logistic regression, we solve:

$$\min_w \mathbb{E}_{(x,y)\sim p_{data}} [\log(1 + e^{-yw^\top x})]$$

where $y \in \{-1, 1\}$. We can show that adversarial training with a logsitic regression model is equivalent to the following object with a ℓ_1 regularization term in the loss function:

$$\min_w \mathbb{E}_{(x,y)\sim p_{data}} \log\left(1 + e^{\epsilon \|w\|_1 - yw^\top x}\right) \tag{3}$$

Important notes Solving $\max_{\|x-x'\|\leq\epsilon} \ell(f(x'), y)$ exactly or even approximately to find adversarial examples is in general hard because it is a non-concave maximization problem. That is why having the right optimization method can make a significant difference in practice. For example, when Madry et. al [5] proposed PGD (several steps of FGSM), it made a huge difference in adversarial training. And they were so confident about the robustness of their models that they created a NeurIPS competition that asks people to attack their models. Interestingly, nobody succeeded to produce attacks that significantly break their robust models, especially on MNIST.

5.3 Ensemble Adversarial Training

Ensemble Adversarial Training is a technique that augments training data with perturbations transferred from other models [6]. In this method, we craft adversarial examples using several pre-trained classifiers and use them to adversarially train our main model. Let's say we have four models (Figure 11). Models B, C, and D are pre-trained classifiers with possibly different architectures which we use to craft adversarial examples against our main model, A. So at each training step, we load our data, craft adversarial examples either using B, C, or D with FGSM, and then we train A by those adversarial examples. This approach decouples adversarial example generation from the parameters of the trained model and increases the *diversity* of perturbations seen during training.
Figure 12 shows the error rate of models trained with adversarial training (blue model) and ensemble adversarial training (red model), both with examples generated using FGSM. Both models are robust, and hence have low error rates on clean data. The blue model has been trained with FGSM against itself, so it makes sense that it has a lower error rate on White-Box FGSM (examples crafted with FGSM on the main model) than the red model that has been trained with attacks coming from other models. However, if we do FGSM attack that comes from other classifiers (Black-Box

FGSM Attack), the blue model performs much worse than the red one because it was only trained on attacks against itself, but ensemble adversarial training makes the model robust against transferrable attacks. It is also important to note that the blue model performs better against white-box FGSM attack than black-box since in the former case, it has been trained and tested on similar distribution of examples. In conclusion, this plot shows that we can achieve better robustness against transfer attacks using ensemble adversarial training.

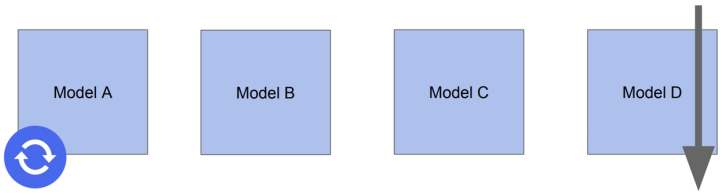


Figure 11: Ensemble Adversarial Traning

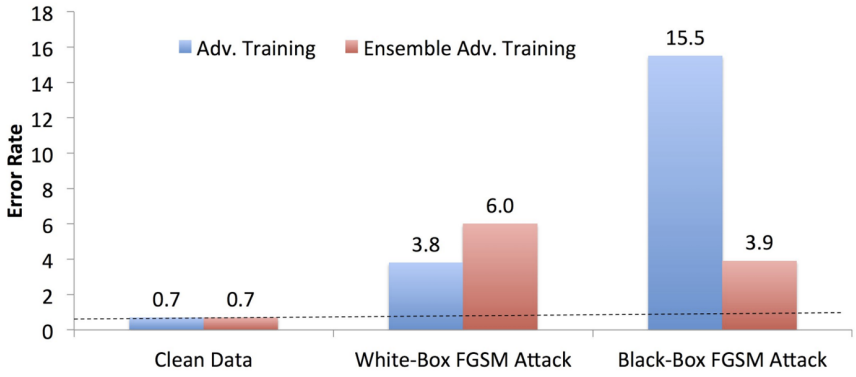
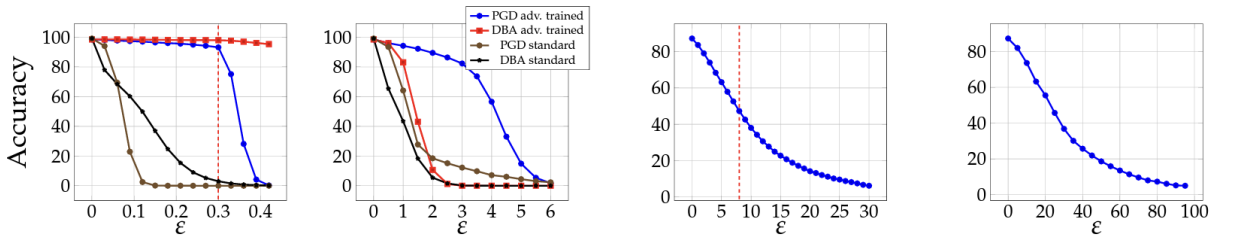


Figure 12: Source: Slides by Nicolas Papernot

Figure 13 from [5] shows the impact of ϵ by illustrating the performance of adversarially-trained models against PGD adversaries of different strengths. The MNIST and CIFAR10 networks were trained against PGD ℓ_∞ adversaries with $\epsilon = 0.3$ and $\epsilon = 8$ respectively (the training ϵ is denoted with a red dashed lines in the ℓ_∞ plots). We can see that if the epsilon is too large, the accuracy drops significantly because with very large ϵ, the attacker has too much freedom and can basically transform the image into random noise, making it impossible for the model to learn them, so it is important to set up this experiment to find the reasonable value for ϵ. For MNIST with ℓ_∞ ball, the standard value for ϵ is now considered to be 0.3.



(a) MNIST, ℓ_∞-norm (b) MNIST, ℓ_2-norm (c) CIFAR10, ℓ_∞-norm (d) CIFAR10, ℓ_2-norm

Figure 13

References

[1] S. Bubeck and M. Sellke. A universal law of robustness via isoperimetry, 2021. URL https://arxiv.org/abs/2105.12806.

[2] L. Engstrom, J. Gilmer, G. Goh, D. Hendrycks, A. Ilyas, A. Madry, R. Nakano, P. Nakkiran, S. Santurkar, B. Tran, D. Tsipras, and E. Wallace. A discussion of 'adversarial examples are not bugs, they are features'. *Distill*, 2019. doi: 10.23915/distill.00019. https://distill.pub/2019/advex-bugs-discussion.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2015.

[4] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features, 2019. URL https://arxiv.org/abs/1905.02175.

[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks, 2017. URL https://arxiv.org/abs/1706.06083.

[6] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. Ensemble adversarial training: Attacks and defenses, 2017. URL https://arxiv.org/abs/1705.07204.