6.883 Science of Deep Learning – Spring 2018 March 14, 2018

Lecture 11: Adversarial Examples and Misclassification Attacks

Lecturer: Aleksander Mądry *Scribe: Anastasiya Belyaeva, Ryan Chung, Samuel Finalyson*

1 Introduction: Security threats in the Deep Learning Pipeline

The dramatic success of AlexNet in the 2012 ImageNet challenge [7] ushered in – seemingly overnight – a veritable deep learning frenzy. In the years since, there has been impressive progress on challenging benchmarks, often surpassing human performance on these tasks. This progress has lead many to believe that Deep Learning is currently at a state to be applied robustly and reliably on virtually any task.

Nevertheless, current deep learning systems exhibit considerable brittleness, and are vulnerable to manipulation at all stages of development and deployment. In this next phase of the course, we will discuss several key limitations to current deep learning systems, and attempt to provide principled formulations for why they exist and how we might address them.

- Training: data poisoning
- Inference: adversarial examples
- Profitable deployment: model stealing

2 Adversarial examples 101

2.1 History and examples

Adversarial examples are inputs to machine learning models that an attacker has crafted to cause the model to misclassify them, despite appearing natural to a human observer. Adversarial examples come in a variety of forms and demonstrate that ML models are not at a human level of robustness, and have been shown to exist for a wide range of ML systems [3].

In the context of deep neural networks, adversarial examples where first observed by Szegedy et al. [10]. They show that imperceptible changes to the image cause the network to misclasssify it (see Figure 1 for an illustration). Adversarial attacks have also been successfully translated into the physical world. It was shown that they continue to be adversarial even when printed and photographed [8]. Recently, it was shown that one can 3D-print physical models that are misclassified from many different angles and lighting conditions [5].

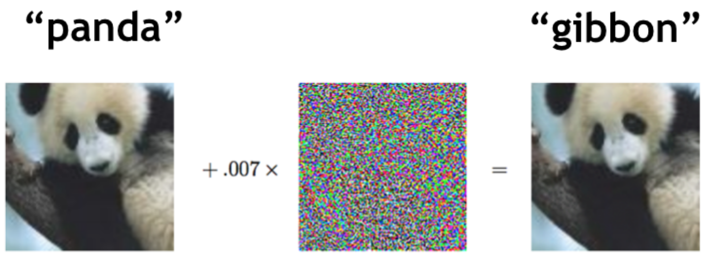


Figure 1: Image of a panda to which small and imperceptible to the human eye noise has been added, which caused misclassification of the image into a gibbon [6].

2.2 Mathematical formulation

Let's try to formulate adversarial examples more mathematically, from the perspective of optimization. Recall that our fundamental problem of supervised learning is to provide an accurate mapping from an input x_i to an output y_i by optimizing over some set of model parameters θ. This can be formulated as a minimization problem

$$\min_{\theta} \text{loss}(\theta, x_i, y_i)$$

This minimization problem is solved by performing stochastic gradient descent on the model parameters (a.k.a. model training). The above approach has been the workhorse of supervised learning. A very similar approach can be taken in order to cause the model to misclassify a specific input. To execute an adversarial attack, we consider our model parameters θ as fixed and instead optimize over the *input space*. More specifically, we search for a perturbation δ that can be added to x_i to *maximize* the resulting model loss:

$$\max_{\delta \in \Delta} \text{loss}(\theta, x_i + \delta, y_i).$$

In order to ensure that the adversarial perturbation is not completely changing the image, we restrict it to be "small". We constrain the δ to be from a class of appropriate perturbations Δ.

The choice of how to define Δ is domain-specific and should thus be chosen after carefully considering the problem at hand. Common approaches include:

- ℓ_p-norm bound: δ is small with respect to some norm, i.e. $\|\delta\|_p \leq \epsilon$ for some ϵ. A common choice is $p = \infty$ which prevents any pixel from being changed by more that ϵ.
- VGG feature similarity: δ cannot change the VGG features (a specific embedding to feature space, obtained through a DNN) of the image by too much.
- Small rotations, translations, and pixel displacements.

During the rest of the lecture, we will assume that $\Delta = \{\delta \mid ||\delta||_\infty \leq \epsilon\}$.

3 Properties of Adversarial Examples

3.1 Adversarial Examples are Present in Linear Models

Adversarial examples are not unique to sophisticated DNN models. Let's consider a simple linear regression model

$$f(x) = w^T x.$$

If we apply a perturbation δ to the input of this model, we have

$$f(x + \delta) = w^T(x + \delta) = w^T x + w^T \delta.$$

To maximize the effect of this perturbation subject to having $\|\delta\|_\infty \leq \epsilon$, we set $\delta_i = \text{sign}(w_i)\epsilon$, then

$$f(x + \delta) = w^T x + \epsilon \|w\|_1.$$

In the case that w has high dimensionality (and thus usually large ℓ_1-norm), the small perturbation can be magnified by applying a bias in the direction away from this image's true class.

Finally, while deep neural networks possess many nonlinearities, they are *piecewise linear* or locally linear. As such, linear perturbations just like the above are thought to drive adversarial examples in deep networks [6].

3.2 Adversarial examples are not random

While it may be tempting to assume that adversarial examples are a product of the models poor robustness to random noise, empirical studies have demonstrated that this is not the case. In fact [6], adding random noise to the inputs produces adversarial examples a small fraction of the time (Figure 2), whereas adding noise in the direction of a calculated attack (the FGSM attack in this case, see below for details), adversarial examples abound (Figure 3).

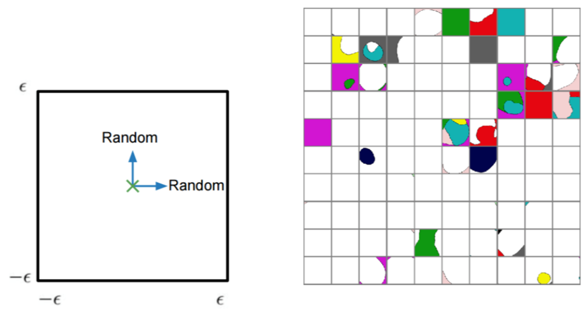


Figure 2: Adversarial examples are not noise.

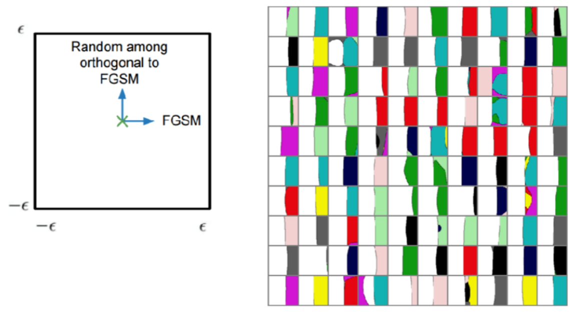


Figure 3: Adversarial examples are plenty in certain directions.

Precisely how large is the space of adversarial examples? Tramer et al. [11] sought to quantify the dimensionality of the adversarial subspaces. They found that only approximately 50 directions were adversarial (Figure 4).

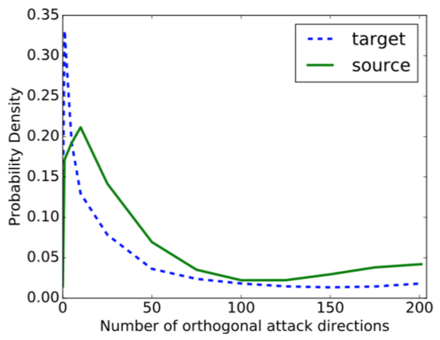


Figure 4: Number of successful orthogonal adversarial perturbations on the source DNN model and of the number of perturbations that transfer to the target DNN model.

Despite the comparatively small dimensionality of the adversarial subspaces, however, other studies have shown that models can be "wrong almost everywhere." [6]. In other words, if you choose *random* inputs to the neural network, you can still perturb the images into any any class you desire in most cases (Figure 5).

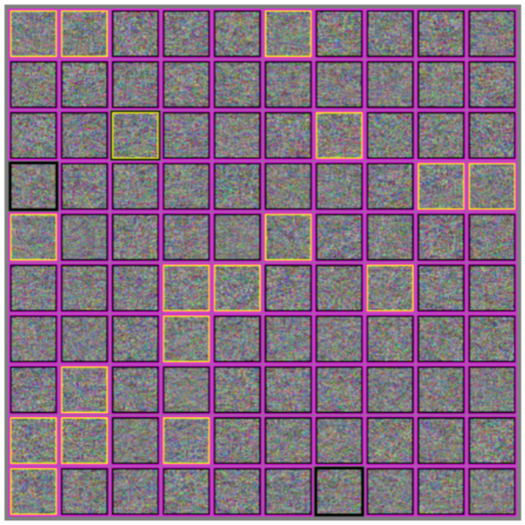


Figure 5: Randomly generated fooling images for a convolutional network trained on CIFAR-10. These examples were generated by drawing a sample from an isotropic Gaussian, then taking a gradient sign step in the direction that increases the probability of the airplane class. Yellow boxes indicate samples that successfully fool the model into believing an airplane is present with at least 50% confidence.

3.3 A word of caution

A cottage industry has sprung up generating various ad hoc defenses to specific adversarial attacks. Time and time again, these defenses have been repeatedly broken, often extremely quickly [4, 1]. This emphasizes the need to evaluate proposed defenses in a rigorous manner, considering adaptive adversaries. Relying on "security through obscurity" only adds noise to the field.

4 A principled approach to avoiding adversarial examples

4.1 Why we shouldn't be surpised that adversarial examples arise from our typical formulations of supervised learning

As we attempt to avoid adversarial examples, it is first important to note that the phenomenon as it currently exists does not necessarily demonstrate a *failure* of machine learning models per se. Rather, adversarial examples reflect a disconnect between what we explicitly optimize our models to do and the behavior that we (often mistakenly) expect our models to implicitly learn along the way.

To make this statement more precise, let's consider the problem we generally hope to solve in supervised learning from a statistical perspective. In learning our model, we generally seek to minimize our expected model loss over the true data distribution (x, y) $\mathcal{D}$:

$$\min_{\theta} E_{(x,y)\sim\mathcal{D}}\left[\text{loss}(\theta, x, y)\right].$$

Likewise, our training procedure can be framed as seeking to approximate this true minimum expected loss by finding the minimum empirical loss over the training data:

$$\min_{\theta} \hat{E}\big[\text{loss}(\theta, x_i, y_i)\big].$$

Both of the formulations above are designed to solve exactly the task of loss minimization and neither of the above formulations explicitly account for adversarial examples. Thus, requiring that the system is robust to adversarial examples, which as we saw are not random perturbations (they are essentially measure zero) is completely orthogonal to the empirical minimization problem above. Since, we never asked our machine learning algorithm not to have adversarial examples, we should not be surprised to find adversarial examples

4.2 Towards a paradigm for adversarially robust generalization

Once an understanding has been established of the task that our machine learning algorithm is solving and not solving, we can redefine our machine learning optimization task to suit our particular goals. Specifically, we can build supervised learning models that are robust to adversarial examples by modifying our learning problem to account for adversarial perturbations. This can be achieved by finding parameters θ that minimize the adversarial loss, which is the highest loss of a given data point after a perturbation δ out of all allowed perturbations Δ:

$$\min_{\theta} E_{(x,y)\sim\mathcal{D}}\big[\max_{\delta\in\Delta} \text{loss}(\theta, x+\delta, y)\big].$$

Formulating and solving such min-max problems has a rich history in the field of *robust optimization* [2] (also known as *adversarial training* [6] in this context, not to be confused with GANs). We can abstract the inner problem by defining

$$\text{loss}_{\text{adv}}(\theta, x_i, y_i) = \max_{\delta\in\Delta} \text{loss}(\theta, x_i+\delta, y_i),$$

in which case adversarial training can be seen as standard training on the adversarial loss (loss_{adv}). For each model parameter update: a batch of images is sampled, the worst case perturbation for each of the images is found and gradient descent is computed on the perturbed image to update model parameters. This solution to the adversarial learning problem should not be confused with GANs. Instead, this approach is akin to data augmentation, where the model is trained not on the data itself but on modified data and in this particular case, adversarially modified data. More generally, data augmentation can be thought of as replacing the max in the training equation above with an expectation over random perturbation.

Although the formulation above is non-convex, we hope that our deep learning methods for solving highly non-convex problems will perform well for this task.

5 Adversarial Attacks

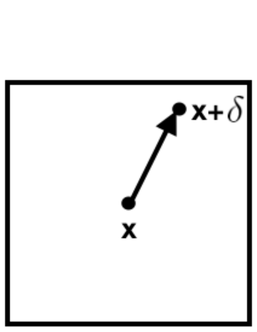
The adversarial problem boils down to finding the optimal direction to move within the box of radius δ surrounding each input x_i that will maximize the probability the model assigns to an incorrect class. Single-step and iterative approaches have been proposed [9].

5.1 Fast Gradient Sign Method (FGSM)

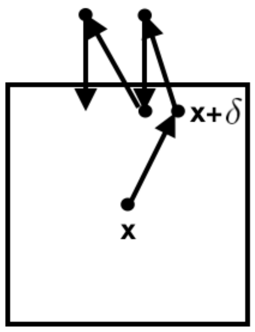
The fast gradient sign method (FGSM) (Figure 6a) was the first simple adversarial attack proposed and represents the optimal single step attack

$$\delta_{\text{FGSM}} = \max_{\delta\in\Delta} \langle \nabla_x \text{loss}(\theta, x, y), \delta\rangle$$

In this case, the set of perturbations is constrained such that $||\delta||_\infty \leq \epsilon$, resulting in the following gradient update:



(a) Fast Gradient Sign Method (FGSM)



(b) Projected Gradient Descent (PGD)

Figure 6: Gradient updates for FGSM (a) vs. PGD (b) where x is the training example, the box is the space of perturbations and δ is a particular perturbation. (CORRECTION, FGSM should go to a corner of the box)

$$\delta = \epsilon \cdot \text{sign}\Big(\nabla_x \text{loss}(\theta, x_i, y_i)\Big)$$

5.2 Projected Gradient Descent Methods (PGD)

Projected Gradient Descent (PGD) methods involves using FGSM with multiple small steps. Additionally, we constrain the perturbation to within the permitted box around x_i. To ensure this, each time PGD takes a step, it checks if it has moved out of the box, and applies a projection back into the box if necessary (Figure 6b). PGD is naturally more expensive than FGSM, but allows for more effective attacks.

6 Properties of Adversarial Attacks

Through experiments using MNIST and CIFAR10, the following properties of adversarial training have been observed [9]:

- The choice of the attack method is important
- The loss of the final solution is a well-concentrated distribution
- Model capacity matters

Comparison of FGSM vs. PGD revealed that PGD results in much better performance than FGSM (Figure 7), which linearizes the inner maximization problem.

Since PGD solves a non-convex problem, the solution given by PGD is one of possible local maxima. However, it is possible that much larger local maxima exist that PGD is unable to find. Madry et al. have investigated this behavior by performing many restarts. As shown in Figure 8, the final loss follows a well-concentrated distribution.

Training against strong adversarial examples requires a stronger classifier with more capacity to distribguish one class from another. Figure 9 shows that a simple linear decision boundary is unable to separate all adversarial examples within ℓ_∞ of the training point because it is not complex enough. However, when a more complicated decision boundary can be learned, smaller number of adversarial examples will be misclassified.

This intuition has been captured via experimental results that trained convolutional networks on natural examples, FGSM examples and PGD examples while doubling the size of network. As shown in Figure 10, small capacity networks have larger loss even on natural examples, indicating that capacity

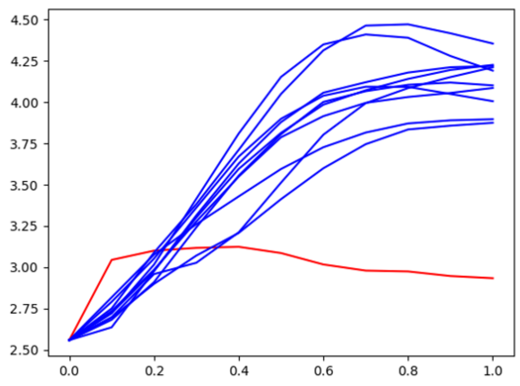


Figure 7: The choice of attack method - FGSM (red) vs. PGD (blue) matters.

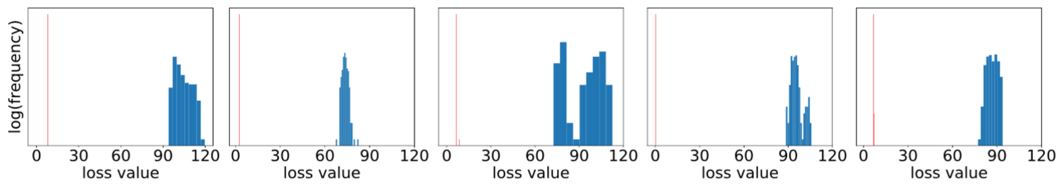


Figure 8: Values of the local maxima given by the cross-entropy loss for five examples from the MNIST and CIFAR10 evaluation datasets. For each example, PGD is started uniformly at random around the example and iterated until the loss plateaus.The blue histogram corresponds to the loss on a naturally trained network, while the red histogram corresponds to the adversarially trained counterpart. The loss is significantly smaller for the adversarially trained networks, and the final loss values are very concentrated without any outliers.

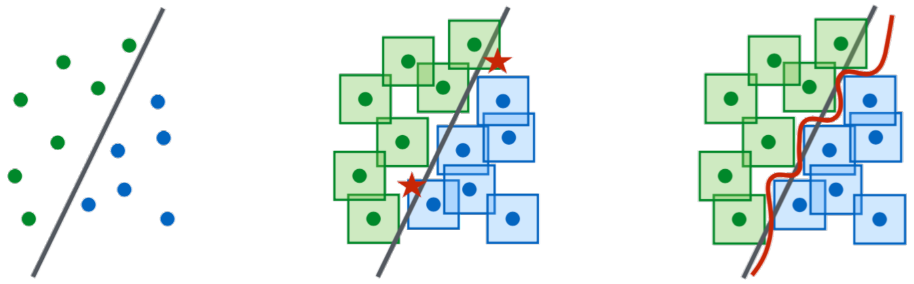


Figure 9: Natural classification (left) vs. adversarial boundaries (right) corresponding to ℓ_∞ ball around training points.

alone increases accuracy. When adversaries like PGD are added, for small capacity networks PGD fails to learn a meaningful decision boundary and performance is sacrificed for robustness. On the other hand, for large capacity networks a robust and accurate solution can be achieved with PGD adversary.

The PGD adversary was trained for both MNIST and CIFAR10 and it has been shown that there is a steady decrease in the training loss of adversarial examples (Figure 11) showing an indication that the original adversarial training optimization problem is indeed being solved during training.

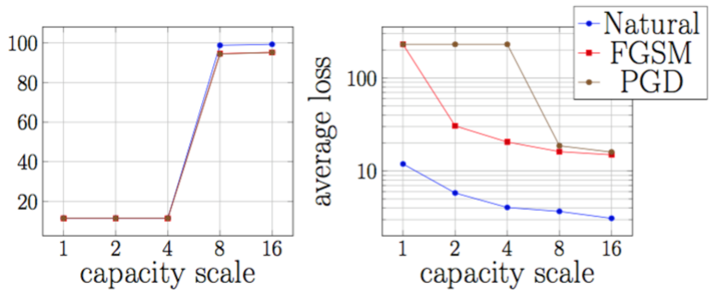
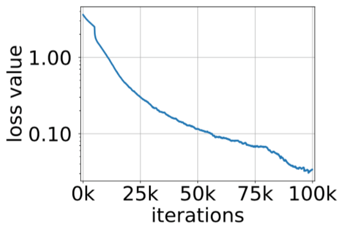
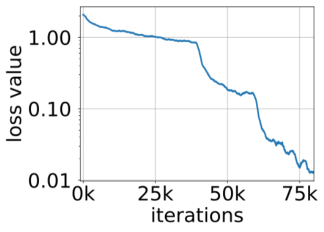


Figure 10: Average accuracy and average loss vs. capacity of convolutional neural networks trained with natural examples, FGSM adversary and PGD adversary.



(a) MNIST



(b) CIFAR10

Figure 11: Value of adversarial loss function on MNIST and CIFAR10 datasets during training.

7 Conclusions

Deep learning models are susceptible to adversarial examples where the differences between training and adversarial images are indistinguishable to the human eye. It has been shown that adversarial examples are not random noise, but are a subspace of perturbations. A variety of approaches that come up with new attack and defense methods have been coming out. Recently a new paradigm that explicitly optimizes for robustness to attacks has been developed.

References

[1] A. Athalye, N. Carlini, and D. Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. *arXiv*, 2018.

[2] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization.* Princeton University Press, 2009.

[3] B. Biggio and F. Roli. Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning. *arXiv*, pages 32–37, 2017.

[4] N. Carlini and D. Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *arXiv*, 2017.

[5] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry. A Rotation and a Translation Suffice: Fooling CNNs with Simple Transformations. *arXiv*, 2018.

[6] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. 12 2014.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. 25, 01 2012.

[8] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *ICLR*, pages 1–14, 2017.

[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv*, pages 1–27, 2017.

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv*, pages 1–10, 2014.

[11] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. The Space of Transferable Adversarial Examples. *arXiv*, pages 1–15, 2017.