Student's Declaration

I hereby declare that the work presented in the report entitled **"Robust Deep Learning"** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr. Saket Anand**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

Anish Madan **Place & Date: 15 November 2018**
...(student's name)...

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

..... **Place & Date: .....**
..(advisors' name)...

Abstract

Machine Learning models are deployed in various tasks including image classification, malware detection, network intrusion detection, etc. But recent work has demonstrated that even state-of-the-art deep neural networks, which excel at such tasks, are vulnerable to a class of malicious inputs known as Adversarial Examples. These examples are non-random inputs that are almost indistinguishable from natural data and yet are classified incorrectly.

In this report, I try to explain the reason for existence of adversarial examples, discuss some of the various attacks developed to exploit the weaknesses of deep neural networks over the years, and provide an analysis of such attacks over a subset of visually distinct classes of ImageNet.

Acknowledgments

I would like to thank Lokender Tiwari(Ph.D Scholar at IIIT-D) for his constant support and guidance to enable me to better understand concepts and different ideas. I would also like to thank my advisor, Dr. Saket Anand for giving me an opportunity to pursue this project and for his support throughout my project.

Work Distribution

Since I am working alone on this project, there was no distribution of work.

Contents

Chapter 1

Introduction

1.1 Why worry about Adversarial Examples?

Recent breakthroughs in Computer Vision have led to near human level performance in various tasks like Image Classification and Object Detection. These advances also call for widespread adoption of such methods for various problems by individuals and industry because of their accurate results. But this also means that such systems would naturally be applied in security-critical scenarios. It turns out that these "highly accurate classifiers" show an inherent weakness known as adversarial examples. Szegedy et al. [1] first noticed the existence of adversarial examples in the image classification domain, when he observed that on applying an imperceptible non-random perturbation to a test image, it was possible to change the network's prediction arbitrarily. These perturbations were found by optimizing the input to maximize the prediction error and were termed as **Adversarial Examples**.

Let's take an example to illustrate the importance of defenses required against adversarial attacks. State-of-the-art neural networks are extensively used in autonomous vehicles for Computer Vision tasks. Generally some decision regarding the path, speed, etc is taken after doing some computation based on the images taken by the camera. Say our car encounters a signal which says to stop. On an unperturbed input, our car would stop but if our input is perturbed (imperceptible but non-random perturbation), our network's prediction would be of some arbitrary class, say 'Go Right'. As you can imagine all autonomous cars on the road would behave in a similar manner, and it would lead to chaos.

Another example would be of a malicious file or malware. By changing some bits of the file which are benign, our systems might predict that it is a benign file (though it did not alter the malicious part of the file, so it still should be classified as malware) and would not block it, therefore posing a security risk to the computer.

1.2 Where do Adversarial examples come from?

Machine learning algorithms are usually designed under the assumption that models are trained on samples drawn from a distribution that is representative of test samples for which we would later make predictions [2]. However, this is not true in the case of adversarial examples. Let us define a Machine Learning Classifier and we will then try to understand the statistical properties of adversarial examples through the eyes of the classifier.

An input $x \in X$ with n features and $y \in Y$ be the label for that input. A classifier then

tries to learn a mapping $f : x- > y$ such that the function tries to minimize the empirical risk. The classifier outputs probabilities and then the label corresponding to the largest probability is chosen as the prediction of the model. There is an unknown distribution D_{real}^{Ci} for each class C_i and the training data X is obtained from this distribution, and the classifier tries to approximate this distribution during training, thereby learning D_{train}^{Ci}.

A notable result in ML is that any stable learning algorithms will learn the real distribution D_{real}^{Ci} up to any multiplicative factor given a sufficient number of training examples drawn from D_{real}^{Ci} [3]. But complete generalization is not possible due to finite number of training samples.

The existence of adversarial examples is a manifestation of the difference between the real distribution D_{real}^{Ci} and the learned training distribution D_{train}^{Ci}, i.e the adversary aims to find a sample from D_{real}^{Ci} whose behavior is not captured by the learned distribution D_{train}^{Ci}. But the adversary does not know the real distribution (if we would, then no need to learn anything) so it takes a sample from D_{train}^{Ci} and tries to perturb it to craft adversarial examples. Each such example made would belong to D_{adv}^{Ci} for the class. Since the perturbations applied are tiny in nature, we know that D_{adv}^{Ci} is consistent with D_{real}^{Ci} (since a sample in D_{adv}^{Ci} would also be in D_{real}^{Ci}) , but D_{adv}^{Ci} differs from D_{train}^{Ci}.

1.3 Linear Explanation of Adversarial Examples

Adversarial examples not only exist in big neural networks but also for shallow linear models. We explain the existence of such examples in linear models in this section.

Digital images are capable of storing 8 bits of information per pixel (i.e from 0-255) and changes below the order of 2^{-8} are not recognized by a computer let alone human vision. So maximum change per pixel should i.e $||\boldsymbol{\eta}||_\infty < \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon}$ is small enough to be discarded by the sensor. For linear models, we calculate the inner product between the weights of the network $\boldsymbol{w}$ and the adversarial example $\tilde{\boldsymbol{x}}$, where $\boldsymbol{w}$ is the unperturbed image :

$$\boldsymbol{w^T}\tilde{\boldsymbol{x}} = \boldsymbol{w^T x} + \boldsymbol{w^T \eta} \tag{1.1}$$

This equation tells us that the activation increases by $\boldsymbol{w^T \eta}$. Our goal is to increase this activation as much as possible, while having capped the maximum change per pixel. This can be achieved by assigning $\boldsymbol{\eta} = \boldsymbol{sign(w)}$. To analyze this behaviour, we assume w is of n dimensions and average value of elements of w is m. This implies that our activation grows by $\boldsymbol{\epsilon} mn$ on average, which tells us that activation can grow linearly with the amount of perturbation added unlike $||\boldsymbol{\eta}||_\infty$. So for large images(or high dimensional data), we could make very small changes to pixel which would add up to something substantial to increase our activation values.

This explanation shows that a simple linear model can have adversarial examples if its input has sufficient dimensionality.

Chapter 2

Different types of Adversarial Attacks

2.1 Fast Gradient Sign Method

Following the discussion from the previous section we see that such a linear nature of examples gives a hint to develop a fast method for this. Even for non linear networks like those using sigmoid activation, most of the time the values stay in the linear part of the sigmoid function and same goes for ReLU.

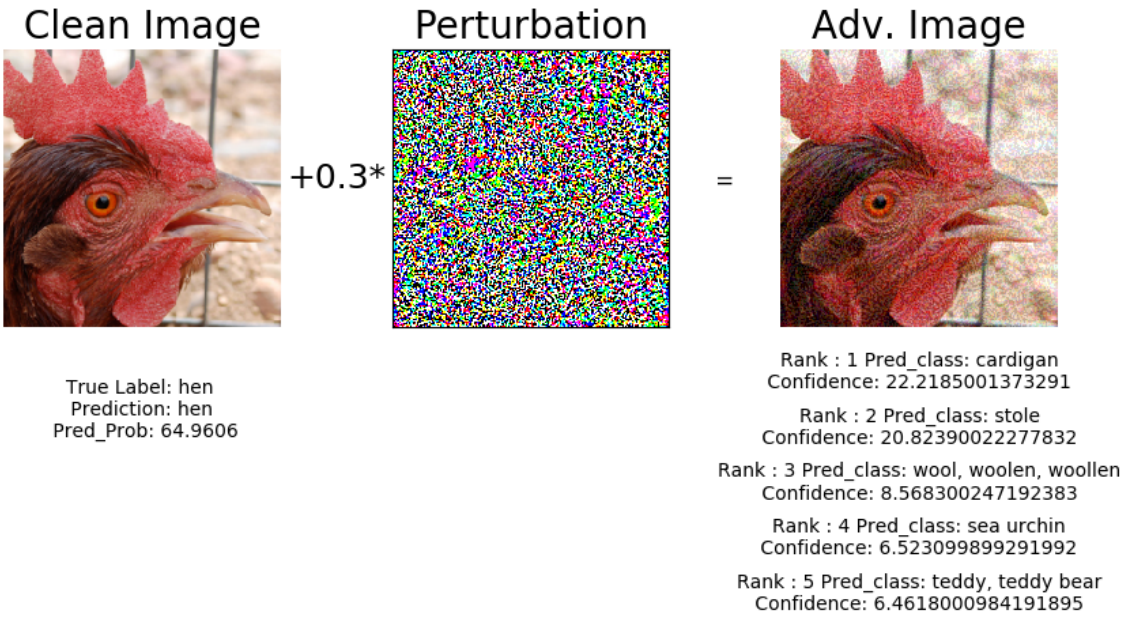

Figure 2.1: *FGSM performed on an image of hen as predicted by the unperturbed model. After adding some perturbation, we can see that the image is still a hen but our model does not predict it as hen even in its top five predictions. Here our ϵ of 0.3 corresponds to the magnitude of the smallest bit of an 8-bit image encoding.*

Let $\boldsymbol{\theta}$ be the parameters of the model, $\boldsymbol{x}$ is the input image, $\boldsymbol{y}$ is the true label for $\boldsymbol{x}$, and $\boldsymbol{J(\theta, x, y)}$ is the loss function for the network. We now linearize the loss function at the current value of $\boldsymbol{\theta}$ to obtain a perturbation of :

$$\boldsymbol{\eta = \epsilon sign(\nabla_x J(\theta, x, y))} \tag{2.1}$$

This method is known as the **Fast Gradient Sign Method [5]**(We will visit this method again in Madry's PGD Attack, when we will try to understand another interpretation of this method). One thing to notice is that we assume that parameters of the model is fixed and compute the gradient with respect to the input, thereby getting a matrix of the same size as that of the input. This cheap method is able to get high levels of misclassifications on datasets like MNIST, CIFAR-10 [4], etc.

An important property of adversarial examples which we haven't discussed is **transferability**. What makes these type of attacks especially interesting is that the adversarial examples generated by an algorithm on one architecture are often misclassified on other architectures trained or by models trained on a disjoint training set. Also, when they misclassify the models tend to agree on the misclassified class. This behaviour is best explained by the linear nature of adversaries, which states that adversarial examples occur in broad subspaces. We only need to get the direction right , i.e $\boldsymbol{\eta}$ should have a positive dot product with the gradient of the cost function and our step $\boldsymbol{\epsilon}$ be large enough in that direction. By using different values for $\boldsymbol{\epsilon}$ it was found that the adversarial examples live in contiguous regions of the 1-D subspaces defined by the method. This explains why adversarial examples on one architecture are misclassified on others too.

To explain this misclassification on models trained on disjoint training sets, we recall that machine learning models generalize, which means regardless of the disjoint nature of training sets we can assume that we would have approximately similar model weights (for a good classifier), hence it would misclassify to the same class.

This property implies that one doesn't even need to have the parameters for a model to generate its adversarial examples. Such type of attacks are known as **black box attacks**. These attacks widen the already gaping security hole that is adversarial examples.

2.2 Basic Iterative Method

The Basic Iterative Method [6] is simply an extension of the Fast Gradient Sign Method. We take multiple smaller steps of size $\boldsymbol{\alpha}$ instead of one step $\boldsymbol{\epsilon}$ like in FGSM.

$$\boldsymbol{X_0^{adv} = X}, \qquad \boldsymbol{X_{N+1}^{adv}} = Clip_{\boldsymbol{X},\boldsymbol{\epsilon}}\{\boldsymbol{X_N^{adv} + \alpha sign(\nabla_X J(X_N^{adv}, y_{true}))}\} \tag{2.2}$$

Also the image pixel values are clipped at every iteration to ensure that they are in $\boldsymbol{\epsilon}$-neighbourhood of the original image. The number of iterations are taken to be $\boldsymbol{min(\epsilon + 4, 1.25\epsilon)}$. This was chosen heuristically ; it is sufficient for the adversarial example to reach the edge of the max-norm ball but restricted enough to keep the computational cost of the experiments manageable.

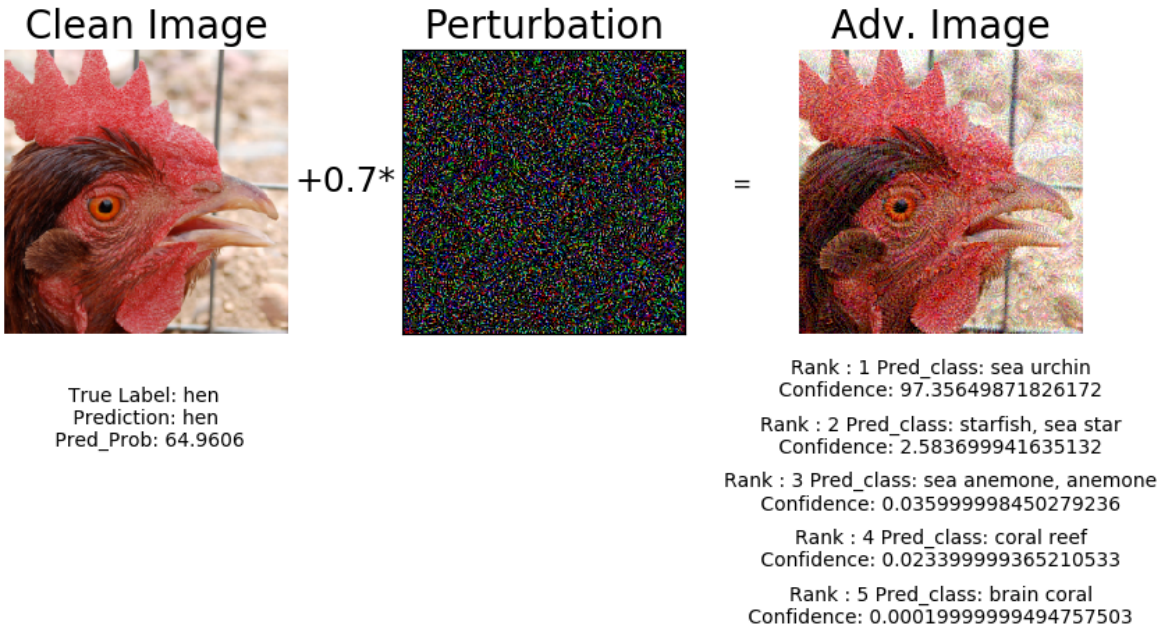

Figure 2.2: *BIM performed on an image of hen. On adding some perturbation iteratively,we see the misclassification by our model in the top-5 predictions. Here we used* $\epsilon = 0.7$ *and* $\alpha = 0.2$

2.3 Iterative Least Likely Class Method

Both the above methods we discussed are **untargeted** in nature, i.e they increase the cost of the correct class without choosing the target class for misclassification. Now we will discuss a method which targets a specific class for misclassifications. This would fall under a **targeted** attack. This attack is of interest on datasets with large number of classes, since on datasets like MNIST we have lower number of classes with higher distinction between classes. If we use untargeted attacks on ImageNet we can get uninteresting misclassifications like a change in breed of an animal as a misclassification. On the other hand it would indeed be interesting to see a misclassification of a spider as a towel.

This method tries to misclassify the original image to the class with the least probability according to prediction of original image. This class is found by :

$$y_{LL} = \underset{y}{\operatorname{argmin}}\{p(y|\boldsymbol{X})\} \tag{2.3}$$

To achieve this misclassification we try to maximize log $p(y_{LL}|\boldsymbol{X})$ by making iterative steps in the direction of sign$\{\nabla_X \log p(y_{LL}|\boldsymbol{X})\}$. This expression equals sign$\{-\nabla_X J(\boldsymbol{X}, y_{LL})\}$ for networks with cross entropy loss. Therefore we have the following method :

$$\boldsymbol{X_0^{adv}} = \boldsymbol{X}, \qquad \boldsymbol{X_{N+1}^{adv}} = Clip_{\boldsymbol{X},\epsilon}\{\boldsymbol{X_N^{adv}} - \boldsymbol{\alpha sign(\nabla_X J(X_N^{adv}, y_{LL}))}\} \tag{2.4}$$

We follow the same number of iterations and value of alpha as that in Basic Iterative Method.

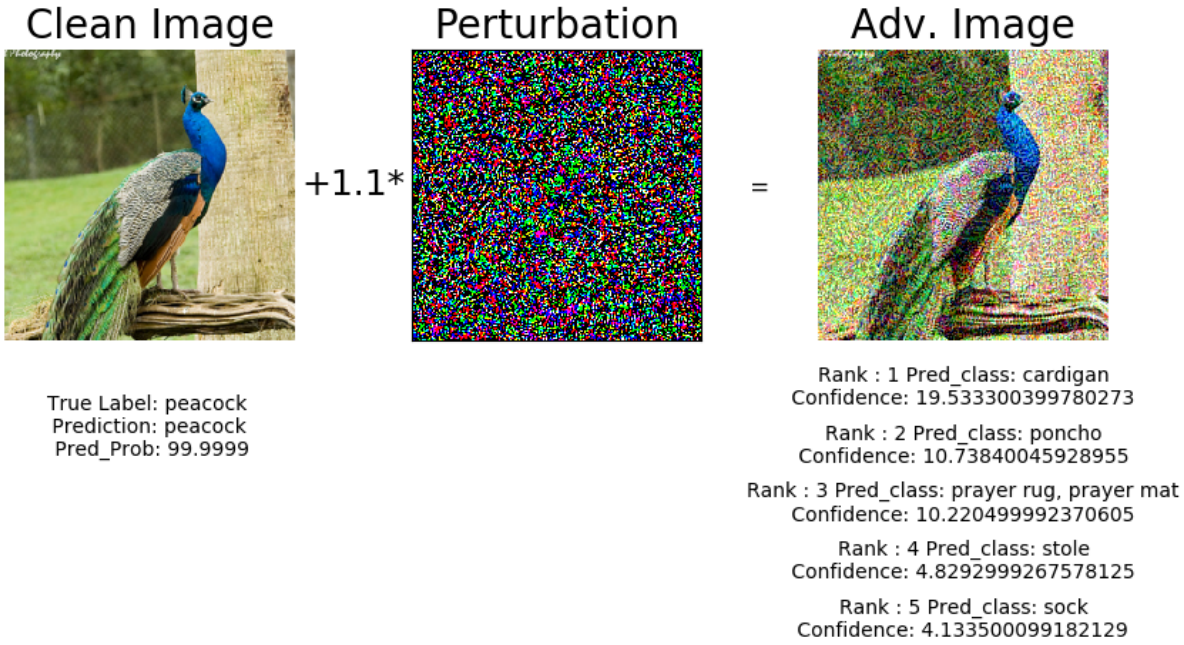

Figure 2.3: *We kept the target class as "hen" (instead of choosing the least likely class, we selected a random class) for this image but this method could not yield that prediction in this case. But we can observe that it has misclassified the image.*

It is not guaranteed that we would always have the corresponding adversarial example since the above methods are not exact. We provide an analysis by comparing the misclassifications per method vs ϵ values.

2.4 Carlini Wagner's L2 Attack

Carlini and Wagner(CW) [8] proposed a set of attacks based on L0, L2, $L\infty$ metric which which broke a defensive distilled model [9] - known for its robustness against previous attacks. CW focuses only on targeted attacks since they are much more powerful than the untargeted ones. Also till now we have only discussed methods for quick generation(computationally cheap) of adversarial examples which did not always give desired results. Let us now discuss CW's approach by defining the problem.
The problem formulation is same as that defined by Szegedy et al. [1]:

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

where x is fixed and we need to find δ to minimize $D(x, x+\delta)$. D is our distance metric and can be any of L0, L2, L_∞. There are many different approaches to solve this optimization problem, but first we need to re-write it in a form which is simpler since the constraint is highly non-linear.
Re-Formulation: We define an objective function f such that $C(x+\delta) = \text{t}$ iff $f(x+\delta) \leq 0$. Now we have an alternate formulation of the problem :

$$\text{minimize } D(x, x+\delta) + \text{c}.f(x+\delta)$$

$$\text{such that } x + \delta \in [0,1]^n$$

where $c \geq 0$ is a constant. After assigning an L_p norm to D the problem finally becomes:

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

1-D optimization techniques like binary search is used to find optimum c. Also we need to take care of the range of $x_i + \delta_i$, i.e the pixels should remain between 0 and 1. This constraint is known as box constraint. To solve this optimization problem we introduce change of variables, which introduces a new variable w, instead of optimizing over δ, we now optimize over w:

$$\delta_i = 0.5(tanh(w_i) + 1) - x_i \tag{2.5}$$

The advantage of this equation is that it automatically follows box constraints, i.e $-1 \leq tanh(w_i) \leq 1 => 0 \leq x_i + \delta_i \leq 1$

2.4.1 L2 Attack

We now use the equations discussed to formulate our L2 attack. For a given datapoint x and target class t (different from class of x),we try to find w which does the following:

$$\text{minimize } ||0.5(tanh(w) + 1) - x||_2^2 + c.f(0.5(tanh(w) + 1))$$

with f defined as

$$f(x') = max(max\{Z(x')_i : i \neq t\} - Z(x')_t)$$

where Z(x) gives the logits of x.

2.5 Madry's PGD Attack

Before discussing the attack, we will spend some time on a different way of interpreting adversarial attacks. Madry et al. [7] provided a way to understand adversarial robustness of classifiers through the eyes of optimization. A natural **saddle point(min-max)** formulation is used to study security against adversarial attacks since it allows us to devise the type of security guarantee we want to have, i.e the broad type of attacks we would like to guard against. This formulation is important since the defenses which exist only guard against specific attacks, so they aren't very helpful as people develop new stronger attacks. Also, this formulation helps us tie the attacks and defenses into one framework: in particular the different attacks correspond to solving the constrained optimization problem.

2.5.1 An Optimization View on Robustness

Consider a classifier with training data distribution **D** and with training samples as tuples (x,y) , where $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{d}}$ and $\boldsymbol{y} \in [\boldsymbol{k}]$ are the corresponding labels. Let us define the loss function as

$\boldsymbol{L(\theta, x, y)}$ which usually is the cross entropy loss and $\boldsymbol{\theta}$ are the weights of the network (classifier). As with all classifiers, we need to find $\boldsymbol{\theta}$ which minimises the risk $\mathbb{E}_{\boldsymbol{(x,y)\sim D}}[\boldsymbol{L(\theta, x, y)}]$.

To achieve this we would think to do the usual Empirical Risk Minimization(ERM). But in fact ERM does not produce adversarially robust models, so we will try to change our problem to solve. First, since we want to capture the general notion of adversaries rather than try to defend against individual attacks. So, we define an *"Attack Model"* ,i.e a precise notion of the attacks our models should be safe against. For each datapoint $\mathbf{x}$, we select a set of allowed perturbations $\boldsymbol{S} \subset \mathbb{R}^{\boldsymbol{d}}$ to allow a certain limit of power of the adversary. This is computed on the basis of $\boldsymbol{l_\infty}$ ball around $\mathbf{x}$. Now we define the risk using the notion of our adversary. We directly include the perturbations into the loss function which gives rise to the following saddle point problem:

$$\min_{\boldsymbol{\theta}} \boldsymbol{\rho(\theta), where} \quad \boldsymbol{\rho(\theta)} = \mathbb{E}_{\boldsymbol{(x,y)\sim D}}[\max_{\boldsymbol{\delta \in S}} \boldsymbol{L(\theta, x+\delta, y)}] \tag{2.6}$$

The above equation is a form of min-max equation and consists of an inner maximization and outer minimization problem. While the inner maximization corresponds to an adversary seeking high loss, the outer minimization problem represents minimizing adversarial loss while finding corresponding weights.

2.5.2 Landscape of Adversarial Examples and Generating Adversaries

We now discuss about the inner maximization part of the problem which corresponds to adversaries maximizing the data point's loss. This problem is highly non-concave and doesn't look tractable. So one way to solve this is by approximation, i.e apply Taylor Series Expansion to this problem. Now, we know that FGSM is a one-step process to generate adversaries. In fact, it is a first order expansion of this inner maximization problem, hence it gives good enough adversaries(computed very quickly), but it is certainly not the best(strongest) available method to do so.
A stronger variant of this attack would be $FGSM^k$ or Projected Gradient Descent (PGD) on the negative loss function [6]:

$$\boldsymbol{x^{t+1}} = \boldsymbol{\Pi_{x+S}(x^t + \alpha sgn(\nabla_x L(\theta, x, y)))} \tag{2.7}$$

A surprising result came out of Madry et al. [7], which stated that the inner maximization problem was tractable(atleast from the point of view of first order methods). They showed this by running PGD from different initialization points within the l_∞ norm ball of the data points. They found that the loss for the data point increased and plateaued very quickly and to almost the same values as observed when starting from rest of the initialization points. This was consistent with the belief that neural networks' loss surface has many local minima but they are not too different in values. Similarly it was found that *while there are many local maxima spread apart from one another within* $\boldsymbol{x+S}$*, they have very well-concentrated loss values.*

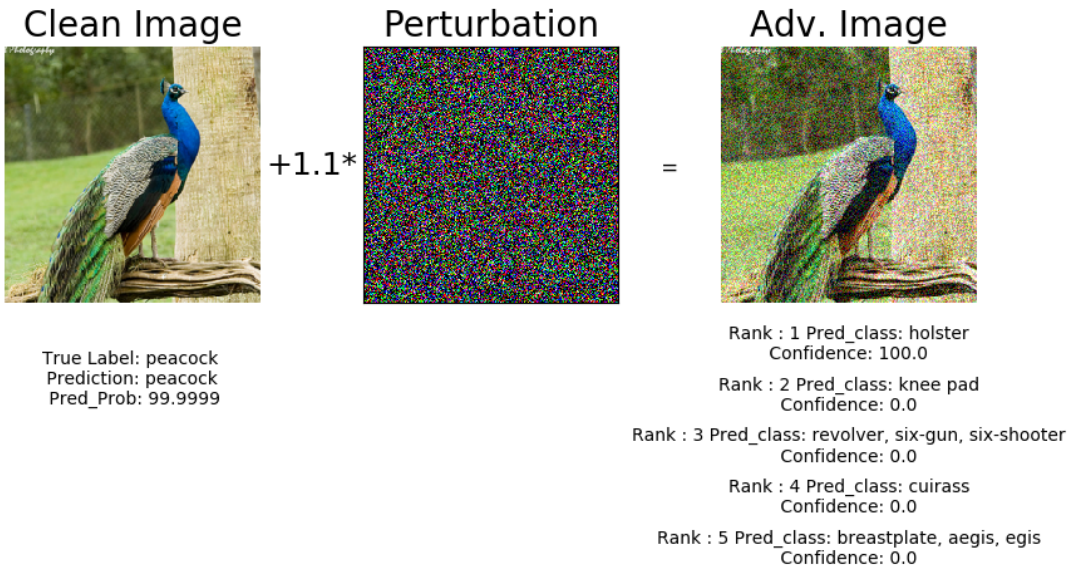

Figure 2.4: Projected Gradient Descent ran on an image of a peacock with ϵ=1.1 and step_size = 0.01 for 10 iterations and no restarts.

The experiments found that local maximas have similar loss values, so robustness against PGD adversaries must safeguard the model from other adversaries as well(first order ones). Hence they concluded that robustness against PGD adversaries would more or less ensure robustness against other first order adversaries.

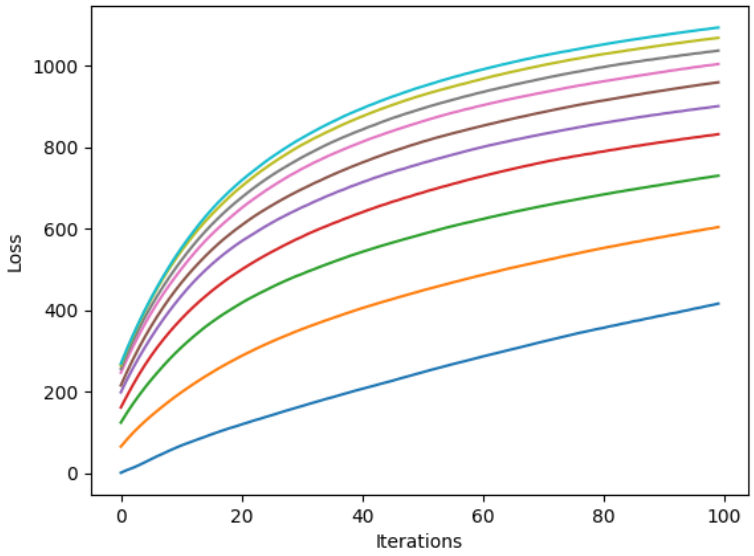

Figure 2.5: *Projected Gradient Descent was run on an image ϵ=1.1 and step_size = 0.01 for 100 iterations and 10 restarts, to analyse the behavior of loss. Each run starts at a uniformly random point in the L_∞ ball around the original datapoint. Madry et al. [7] reported this behavior on MNIST and CIFAR datasets. We have provided this analysis for a subset of ImageNet. We can clearly observe that the loss increases quite quickly for a datapoint but then starts to plateau with increasing iterations. This implies loss achieved by an adversary increases in a fairly consistent way and starts to plateau for randomly chosen starting points within x+S*

Chapter 3

Analysis of Different Attacks

In this section we will compare the various methods discussed and explore the behaviour of each attack. We have selected a VGG11 model which is pretrained on ImageNet. To analyze these attacks, we have selected 30 classes from ImageNet which are visually distinct from one another. For each class we choose 5 images randomly, so are test set now consists of 150 images. The images are chosen so as they classify correctly without adversaries. So in all cases of unperturbed models, we will get 100% accuracy in classification. This is done because we need to analyse performance of different attacks and that is possible only when we assume a classifier that works perfectly on given data. If not, then we would have wrongly classified images, so it doesn't make much sense crafting adversaries for such since it already misclassifies those(so in essence we could craft adversaries with $\epsilon = 0$. We first by analyzing the untargeted attacks.

3.1 Untargeted Attacks

Our list of untargeted attacks consists of the Fast Gradient Sign Method, Basic Iterative Method, and Projected Gradient Descent Attack. Untargeted attacks imply that the adversary aims to misclassify a given datapoint by perturbing it. The adversary doesn't target any specific class and only aims to increase the loss for that datapoint so that misclassification happens. These types of attacks are weaker than the targeted ones.

We now compare performances of the different attacks. We fix $\epsilon = 0.1$ for this analysis. We observe in Table 3.1 that FGSM degrades the performance the most out of all the attacks. We did not increase the steps for BIM since we are bound by the heuristic provided, which yields better results than FGSM for higher values of ϵ (since it corresponds to more number of steps. α was kept as $\epsilon/2$ for this analysis. Now PGD gives the worst performance of all yet it is claimed as being stronger than others. We can observe this when we increase the iterations to 50 and keep α=0.01. We see that top-1 accuracy reduces to 1.3% and top-5 accuracy to 14%. Similar trend can be observed for values in Table 3.2 .

Untargeted Attack for $\epsilon = 0.1$			
Method	No. of Steps	Top-1 Accuracy(%)	Top-5 Accuracy(%)
Natural	-	100	100
FGSM	1	2.6	19.3
BIM	1	2.6	26
PGD	6	30	72.6

Table 3.1: This table reports the various accuracies for different untargeted methods while fixing $\epsilon = 0.1$

Untargeted Attack for $\epsilon = 0.6$			
Method	No. of Steps	Top-1 Accuracy(%)	Top-5 Accuracy(%)
Natural	-	100	100
FGSM	1	2	7.9
BIM	2	0.6	10.6
PGD	6	12.6	33.9

Table 3.2: This table reports the various accuracies for different untargeted methods while fixing $\epsilon = 0.6$

We can observe the trend noted in the table for a range of values of ϵ in the figures 3.1 and 3.2. They capture the behavior of top-1 and top-5 accuracies for ϵ ranging from 0.05 - 0.6. Unlike the results from other papers, we don't provide this behavior for high values of ϵ, for eg. $\epsilon = 16$ This is done because allowing every pixel to change by $16(L_\infty$ norm) makes little sense as the image gets filled with too much distortion.

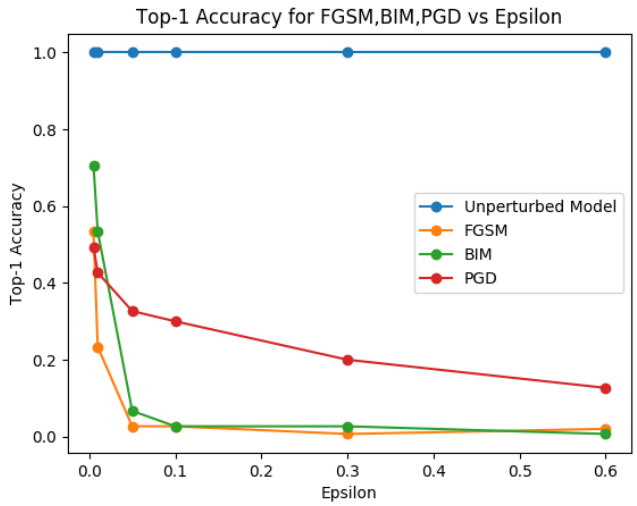

Figure 3.1: Comparing top-1 accuracy values for different epsilon values among various untargeted methods.

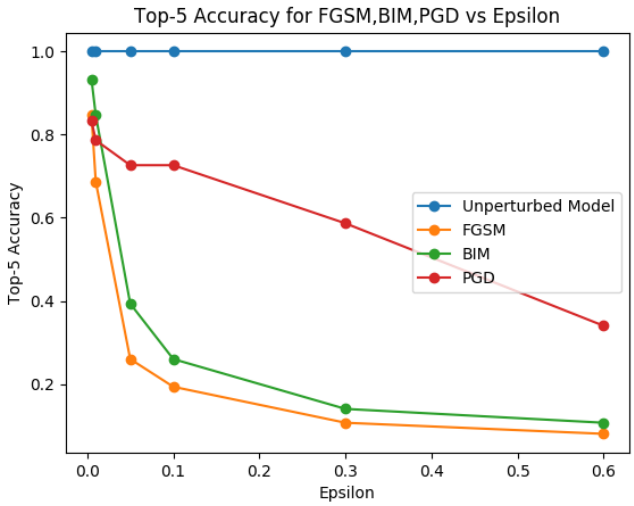

Figure 3.2: Comparing top-5 accuracy values for different epsilon values among various untargeted methods.

3.2 Targeted Attacks

In this section we compare the targeted attacks , namely Fast Gradient Sign Method (also works for targeted) and Iterative Least Likely Class(for any random class). We discuss Carlini Wagner's L2 Attack separately as it is quite distinct from these. We did not discuss Fast Gradient Sign Method for targeted attacks but it works similar to the Iterative Least Likely Class Method. The only difference is that it takes only one step to craft adversaries.

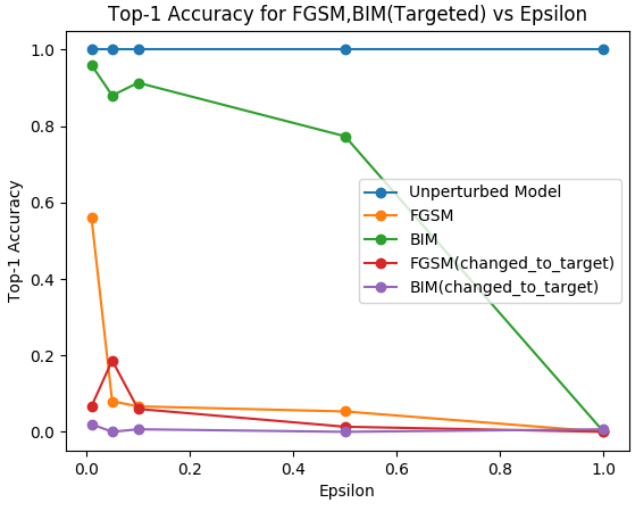

Figure 3.3: Comparing top-1 accuracy values for different epsilon values among various targeted methods.

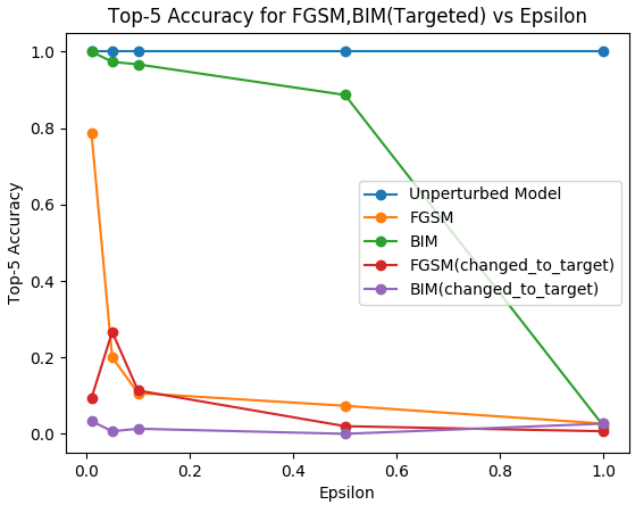

Figure 3.4: Comparing top-5 accuracy values for different epsilon values among various targeted methods.

We know that increasing the values of ϵ leads to stronger adversaries and the accuracy of the model decreases with it. On observing Figures 3.3 and 3.4, it is quite evident that the results are consistent with the above statement. We know that the unperturbed model will always have 100% accuracy. Now, the FGSM attack decreases model accuracy quite rapidly on increasing ϵ and so does BIM. But BIM is initially slow since it takes smaller steps and due to the choice of iterations via a heuristic, its true power increases with increase in number of iterations and that is only possible with increasing ϵ. FGSM(changed_to_target) and BIM(changed_to_target) represent the accuracy corresponding to the target label, i.e whether or not the predicted label was changed to the target label. As we can see that both the methods start very close to 0 and it is a gradual decrease towards 0 from there, which tells us that these methods are not quite good at targeted attacks, although they are good enough to cause misclassifications.

3.3 Carlini-Wagner's L2 Attack

This targeted attack is one of the stronger attacks and also the first ones to beat defensive distillation [9]. we analyse the adversarial images generated by this technique and also compare its performance on the ImageNet subset.

Targeted CW L2 attack on ImageNet subset		
Metric	Top-1	Top-5
Adv_target_acc	100%	100%
Accuracy	100%	51.73%

Table 3.3: This table reports the results obtained on applying CW -L2 attack on the ImageNet subset. Adv_target_acc refers to the accuracy with which original labels were successfully changed to the given target label. Misclassifications report the proportion of examples misclassified by the model on attack.

Figure 3.5: CW-L2 attack on an image of a candle with the target label **Tibetan terrier**. The attack successfully changes the original label to the target class without much distortion.

Figure 3.6: CW-L2 attack on an image of a bear with the target label **car**. The attack successfully changes the original label to the target class without much distortion.

We observe that this attack produces adversarial examples which are successfully converted to the target label's class. This is due to its optimisation objective which is much more exact when compared to other methods like fast gradient sign method and iterative least likely class. But this comes at a cost of large amount of computation time to generate each adversarial example. To generate each example, a binary search over the parameter c is carried out, and for value of c some fixed number of optimisation steps are carried out (typically about 1000) and if the attack is successful, the algorithm tries to minimise the value of c (via binary search). So all of this corresponds to higher computation time for each adversarial example.

Bibliography

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. ICLR, abs/1312.6199, 2014. URL `http://arxiv.org/abs/1312.6199`

[2] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (Statistical) Detection of Adversarial Examples. arXiv preprint arXiv:1702.06280 (2017).

[3] Olivier Bousquet and Andr Elisseeff. 2002. Stability and Generalization. The Journal of Machine Learning Research 2 (2002), 499526.

[4] Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. CoRR, abs/1412.6572, 2014. URL `http://arxiv.org/abs/1412.6572`.

[6] Alex Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. Technical report, arXiv, 2016. URL `https://arxiv.org/abs/1607.02533`.

[7] 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.

[8] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. arXiv preprint arXiv:1608.04644, 2016.

[9] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In Security and Privacy (SP), 2016 IEEE Symposium on, pages 582597. IEEE, 2016.