Binary Iterative Method for Non-targeted Adversarial Attack

Naman Goyal*
Indian Intitute of Technology, Ropar

Milan Chaudhari*
Indian Intitute of Technology, Ropar

Abstract

This papers presents compares existing non-targeted various methods to generate non-targeted adversarial attack such as Fast Gradient Method, Basic Iterative Method, Virtual Adversarial Method using the InceptionV3 model. These attacks are then evaluated over pre-trained networks like InceptionV3, InceptionV2, ResNet V2 152 over classification task. To this end, we contribute a new method Binary Iterative Method which performs well over most of others methods on 1000 images sampled from the standard ImageNet dataset.

Introduction

Adversarial examples refer to the transforming the input by an adversary so that any general machine learning model misclassify i.e the perturbed input results in the model outputting an incorrect answer with high confidence.

The input is transformed by a small perturbation which looks like random noise, and can be so subtle that a human observer does not even notice the modification at all, yet the classifier still makes a mistake.

There are two versions of the problem targeted and non-targeted both of which work on the generally unknown machine learning classifier. While in non-targeted attack aim to just misclassify to any other class, in targeted one the aim is to misclassify to a particular class. The rest of the discussion is on non-targeted attack model.

The task is extremely relevant in current scenario where we can do adversarial training. Adversarial examples pose security concerns because they could be used to perform an attack on machine learning systems, even if the adversary has no access to the underlying model.

While the noise added looks completely random; it is carefully computed using a another machine learning model various methods. Our main focus is on analyzing these methods and evaluating these methods on some state-of-the neural networks.

(Kaggle 2017)

*These authors contributed equally

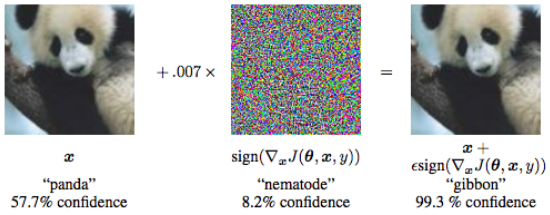

Figure 1: Explaining and Harnessing Adversarial Examples (Goodfellow, Shlens, and Szegedy 2014)

Related Work

Our project is based on adversarial attacks on machine learning models. This attacks were first time introduced by (Gu and Rigazio 2014) where they showed deep neural networks (DNNs) to be highly susceptible to well-designed, small perturbations at the input layer.

While the early attempts at explaining this phenomenon focused on nonlinearity and overfitting (Goodfellow, Shlens, and Szegedy 2014) argued instead that the primary cause of neural networks' vulnerability to adversarial perturbation is their linear nature i.e. piece-wise linearity among the various layers of neural networks. And came up with the idea of simple and fast method of generating adversarial examples.

The major solutions are now available through cleverhans (Papernot et al. 2016) where the main focus of attack is based on the idea of how the weight updates of a neural network work.

The basic idea is to attack the way a neural network updates is parameter i.e. back-propagation.

$$w_{t+1} = w_t - \alpha \nabla_{w_t} Loss \tag{1}$$

To increase the $Loss$ we instead transform each input image x by $\nabla_x Loss$

$$x_{perturb} = x + eta = x + \alpha \nabla_x Loss \tag{2}$$

then any neural network learned on the x when tested on $x_{perturb}$ would decrease the original output confidence for true class of x.

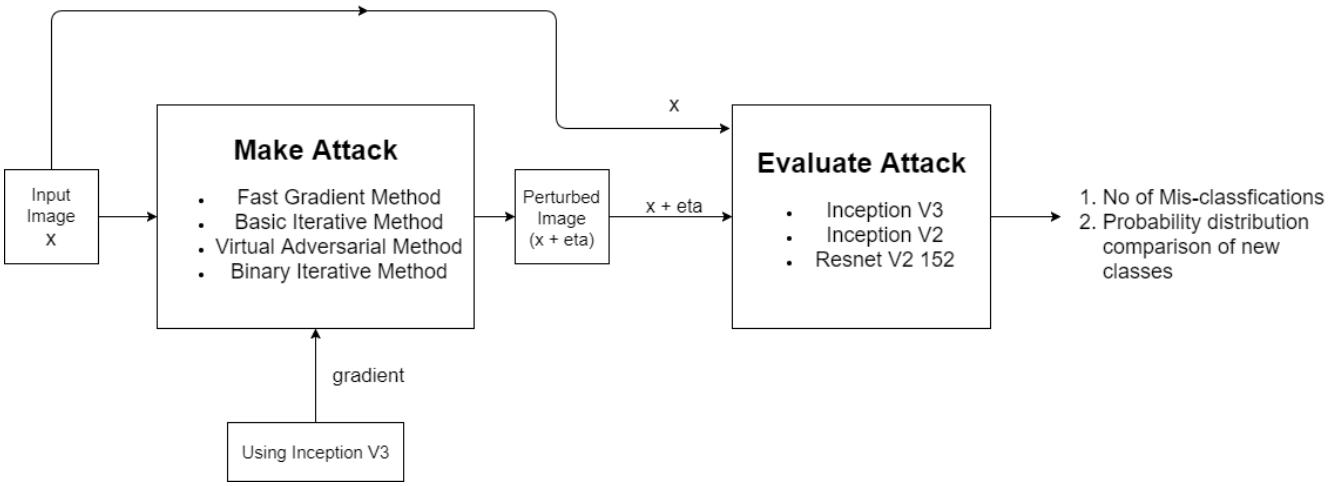

Figure 2: Method of transforming the image

Some of the current methods which are used to compare our results are

1. **Fast Gradient Method (FGM)** In this method we take a machine learning model and an image, calculate gradients of the loss function(the cross-entropy error) w.r.t. input image, calculate the normalized gradients (In this case the sign of gradients) and add it to the input image after multiplying with a small number (epsilon: the step-size).

$$x_{perturb} = x + eps * sign(\nabla_x Loss) \tag{3}$$

2. **Basic Iterative Method (BIM)** This the application of the Fast Gradient Method iteratively over the perturbed image by fixing a small step-size and iteratively update the image.

$$x^1_{perturb} = x \tag{4}$$

$$x^{t+1}_{perturb} = x^t_{perturb} + eps\_iter * sign(\nabla_{x^t_{perturb}} Loss) \tag{5}$$

3. **Virtual Adversarial Method (VAM)** (Miyato et al. 2015) This method is based on the careful update using KL loss based on logits (the model's unnormalized output or the input to the softmax layer)

Methodology

We basically updated the Basic Iterative Method and came up with **Binary Iterative Method (BinIM)**. where we made an underlying assumption that we are searching in a concave region of $\nabla_x Loss$ function.

We now need to find maximum of the same i.e. a point with zero gradient. Since the gradient in concave part of graph behaves like a array were values on the left side are positive and values on the left side are negative and they are in *sorted order* we need to search the value 0.

200	100	50	20	0	-19	-50	-100	-224

So we do a binary search where we start with a very high value of $eps\_iter^1 = eps$ then each iteration reduce the $eps\_iter^{t+1} = eps\_iter^t/2$ i.e

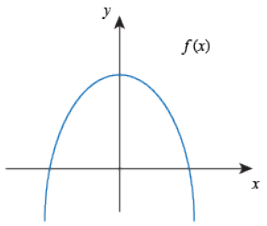

Figure 3: Maximising the error using a concave region assumption

Initialize as

$$\begin{cases} x^1_{perturb} = x \\ eps\_iter^1 = eps \end{cases} \tag{6}$$

Update as

$$\begin{cases} x^{t+1}_{perturb} = x^t_{perturb} + eps\_iter^t * sign(\nabla_{x^t_{perturb}} Loss) \\ eps\_iter^{t+1} = eps\_iter^t/2 \end{cases} \tag{7}$$

Experiment and Results

Our aim is to reduce the accuracy of classification. The approach to generate result is as described in figure2 i.e.

1. Sample 1000 images from the standard ImageNet dataset.
2. Transform the images using 3 available methods i.e. Fast Gradient Method, Basic Iterative Method, Virtual Adversarial Method and our method i.e. Binary Iterative Method using gradient of loss computed over Inception V3 network.

3. To validate our results we then evaluate the accuracy of classification over the original data set and 4 perturbed outputs over 3 different state-of-the networks for classification i.e. InceptionV3, InceptionV2, ResNet V2 152.

Model	Accuracy
Inception_V3	0.958
Inception_V2	0.855
Resnet_V2_152	0.906

Table 1: Accuracy over the original dataset

Model	**FGM**	**BIM**	**VAM**	**BinIM**
Inception_V3	0.356	0.0252	0.628	0.009
Inception_V2	0.882	0.839	-	0.683
Resnet_V2_152	0.835	0.828	-	0.668

Table 2: Accuracy over the modified dataset

FGM = Fast Gradient Method
BIM = Basic Iterative Method
VAM = Virtual Adversarial Method
BinIM = Binary Iterative Method *(our method)*

Observation The accuracy over Inception_V3 decreases the most over other models to evaluate the attack.

Explanation The gradient to make the attack was using Inception_V3 hence it is expected to make maximum classifications over the Inception_V3 itself.

Observation All the four methods are not much effective over the Resnet_V2 and Inception_V2, but still they are able to reduce confidence of true label.

Explanation "An intriguing aspect of adversarial examples is that an example generated for one model is often misclassified by other models, even when they have different architecures or were trained on disjoint training sets. Moreover, when these different models misclassify an adversarial example, they often agree with each other on its class. Explanations based on extreme non-linearity and over- fitting cannot readily account for this behavior. This behavior is especially surprising from the view of the hypothesis that adversarial examples finely tile space like the rational numbers among the reals, because in this view adversarial examples are common but occur only at very precise locations.", as stated by (Goodfellow, Shlens, and Szegedy 2014).

Observation The Binary Iterative Method (BinIM) causes the most number of misclassifciations.

Explanation The Binary Iterative Method (BinIM) is combination of techniques used in the FGM, BIM hence is expected to give better result i.e. more misclassfications.

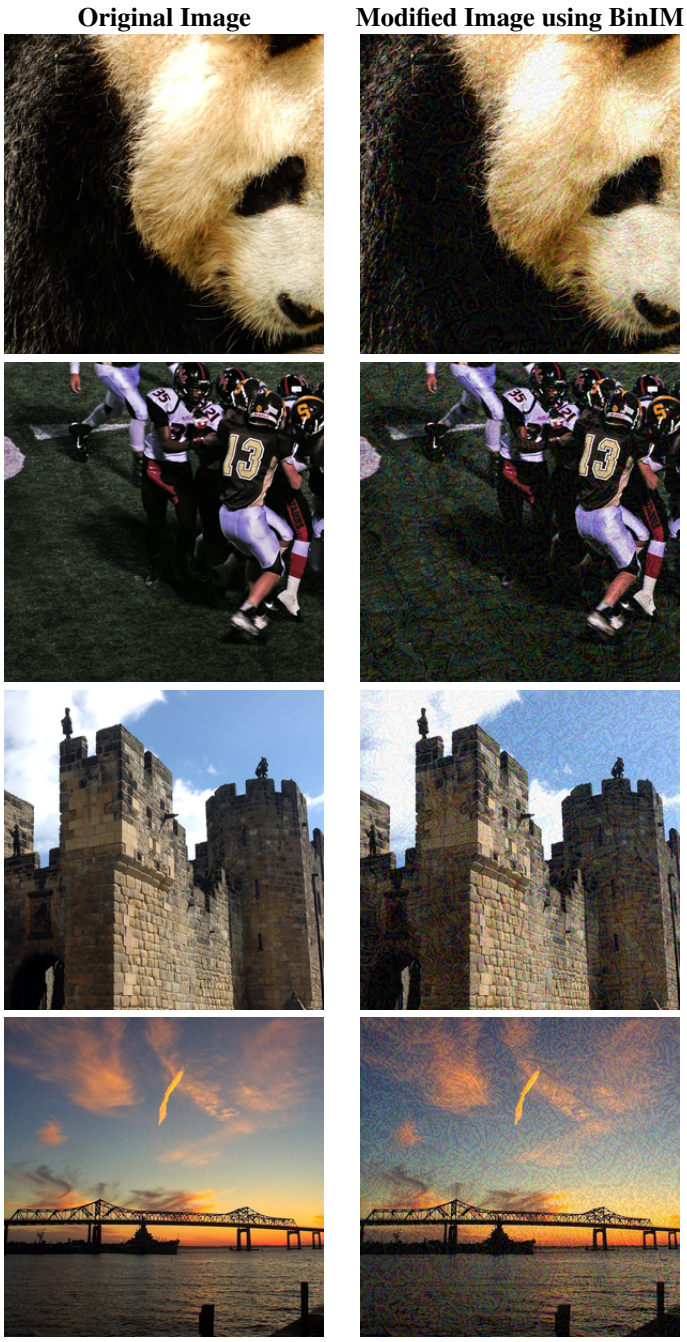

Table 3: Sample outputs generated using Binary Iterative Method

Observation Few the modified images are indistinguishable from the original one such as images 1 and 2; while the 3rd and 4th images are distinguishable.

Explanation This depends on the spatial features present in the images such as in both the 3rd and 4th images; the presence of sky causes a human observer to notice the change in the color intensity and the change in color intensity.

Initial Predicted label	Initial Prob	Modified Label	Modified Prob	Modified Prob for initial label	True Label
306	0.807618	303	0.9995	2.21524e-09	306
884	0.995941	873	1.0	6.9128e-17	884
244	0.946103	152	1.0	2.88032e-16	244
560	0.834364	542	1.0	6.43702e-20	560
439	0.542039	371	0.989378	3.6594e-08	439
991	0.832426	954	1.0	1.81221e-19	991
950	0.97152	114	1.0	7.20899e-17	950
854	0.988273	778	1.0	1.17832e-13	854
610	0.814898	657	0.999993	4.58456e-12	610
610	0.434363	437	0.999994	2.39441e-11	610
583	0.337361	455	0.999609	2.3094e-22	583

Table 4: Sample Transformation of Probabilities using Binary Iterative Method evaluated on Inception_V3

Observation The table 4 represents how the model 'Inception_V3' were initially classifying and how the classification changes after modification in the input images.

Consider the first example, initially the model predicted the label 306 with probability of 0.807, which is the true label.

While after modification in the image, the model predicts the label 303, which not only is wrong label, but model's is confident with probability is 0.9995. Moreover the probability of the true label has reduced to 2.21e-09 (≈ 0), which is quite low.

Explanation This shows the proposed Binary Iterative Method is able to generate a good adversarial exmaples and able to reduce the confidence of the orginal class to great extent.

Summary

The proposed Binary Iterative Methods is able to generate adversarial examples which reduce the confidence of the original label to a great extent which was further validated using results from different state-of-the neural network over 1000 image data set sampled from ImageNet.

The method can be used for adversarial training to reduce the impact of other Adversarial Attack.

Future Works

- We can train over adversarial examples then see if the model can learn to defend against adversarial examples generated by other methods. Hence look at the defence methods.
- We experimented with Fast Gradient, Basic Iterative and Virtual Adversarial Method. The community is always involving with new methods and there remains a large number of other attacks to try.
- We can further develop a more generic attack method which preforms well on various kinds of architectures. The Binary Iterative Method based on the sign of gradient worked well for Inception Network but not of Reset.
- We introduced a method for non-targeted attack, but can work for the targeted one.

References

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Gu, S., and Rigazio, L. 2014. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*.

Kaggle. 2017. NIPS 2017: Non-targeted Adversarial Attack. `https://www.kaggle.com/c/nips-2017-non-targeted-adversarial-attack/`. [Competition Ended].

Miyato, T.; Maeda, S.-i.; Koyama, M.; Nakae, K.; and Ishii, S. 2015. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*.

Papernot, N.; Goodfellow, I.; Sheatsley, R.; Feinman, R.; and McDaniel, P. 2016. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*.