Analysing adversarial inputs to Deep Neural Networks

Prasanna Patil **Mohit Gupta**

Abstract

Deep neural networks have far reaching application and have far complex architectures. However, in past, it has been shown that Deep neural networks which are performing much better in terms of generalization are flawed when a small amount of perturbation is introduced in an image. This perturbed images are known as adversarial inputs as they break functionality of neural networks. Much research has been devoted to the study of attacks and defences against adversarial inputs. However, adversarial inputs to deep neural networks still remains a viable threat. In this project, we review a few methods for generating adversarial inputs for deep neural networks. We also experiment two probable defences against adversarial attacks. The experimental results obtained from these experiments give us a few more insights regarding how adversarial inputs work.

1. Introduction

Deep neural networks have complex architecture which makes it difficult to analyse it theoretically but as it has been shown by previous attempts (Krizhevsky et al., 2017; Szegedy et al., 2014; He et al., 2015) that they are far better at modelling tasks such as object recognition, image recognition, speech processing and natural language processing. It has also been shown that they are highly prone to small input perturbations (Szegedy et al., 2013; Goodfellow et al., 2014; Moosavi-Dezfooli et al., 2015). Previous research has also proposed viable defences (Papernot et al., 2015b; Madry et al., 2017; Sharma & Chen, 2017) and there have been attacks against those defences (Carlini & Wagner, 2016; Sharma & Chen, 2017; Carlini & Wagner, 2017).

In this project, we try to come up with new defences against adversarial attack and in the process discover an intuition regarding how adversarial attacks work.

Our first proposed defence is a Teacher Student framework where a set of teacher models are used to make a student model robust against adversaries. Here Teacher performs job of adversarial knowledge transfer. This approach is similar to the defence proposed in (Krizhevsky et al., 2017). However, in (Papernot et al., 2015b), no adversarial training was used which is different from what we have done.

Our second proposed defence is to randomize the subset of pixels of input images and evaluate robustness of such neural network. We also use concept of a teacher model here, however it is different from how teachers work in 1st framework. Here, teacher model isn't used to provide any sort of adversarial training to the student model.

All of our experiments are performed on DNN architecture that was used in (Madry et al., 2017) and training data is MNIST (LeCun & Cortes, 2010). Here, we are considering BlackBox adversaries which has access to training data and DNN architecture but do not have access to the trained model used by user.

Both of our experiments are implemented using TensorFlow (Abadi et al., 2015). For the purpose of generating adversarial inputs we are utilizing cleverhans (Papernot et al., 2018) which implements a broad class of attacks with many tunable features for each attack.

The outline of report is as follows: Section 2 gives a background on adversarial attacks. Section 3 describes related work in defences against neural network. Section 4 details the two experiments we have done and section 5 contains their experiment evaluations. Section 6 proposes future work that can be done based upon outcomes of our experiments and section 7 concludes the report.

2. Adversarial attacks

In this section we describe a few of the adversarial attacks that we have used in our experiments. Note that this list is not a complete list of adversarial attacks. It is for the purpose of review of attacks used in our experiment.

2.1. L-BFGS

This attack is the very first attack against DNNs. This was introduced in (Szegedy et al., 2013). This attack tries to produce adversarial input by performing a constraint optimization as below:

$$\begin{array}{ll} \text{minimize} & ||x - x'||_2^2 \\ \text{such that} & C(x') = l \\ & x' \in [0, 1] \end{array}$$

C in above equation represents the classifier function. This above problem is difficult to solve as it requires x' without any instructions on how to find such value. However, the problem is solved using an alternative formulation as shown in (Szegedy et al., 2013). The alternative formulation is as shown below:

$$\begin{array}{ll} \text{minimize} & c \cdot ||x - x'||_2^2 + loss_{Fl}(x') \\ \text{such that} & x' \in [0, 1]^p \end{array}$$

$loss_{Fl}(x')$ is loss of classifier for input x' in direction of class l. Hence, the formulation is trying to solve the problem by minimizing the cost of input x' in direction of class l and at the same time minimuizing L_2 distance between actual input and adversarial input.

Generating adversarial inputs using this attack is heavy because of the L-BFGS method. Hence, we have not used this in our experiments. This is a targeted attack in which adversarial input is generated so that DNN misclassifies the adversarial image into pre-specified target class.

2.2. Fast Gradient Sign Method

Fast Gradient Sign Method or FGSM was first proposed in (Goodfellow et al., 2014). This attack is also a targeted attack. The formulation of FGSM and L-BFGS is similar. However FGSM tries to find adversarial input using first order gradient optimization methods which are also used to train neural network. This specific feature of this attack makes it quite feasible and quick to generate adversarial inputs. The attack tries to find adversarial input x' from actual input x and target class $y \neq C(x)$ by following procedure as below where J is the classifier loss function:

$$\eta = \epsilon \cdot sign(J(X, y, \theta)$$
$$x' = x + eta$$

FGSM attack uses $||l_\infty||$ norm to measure distance between adversarial input and actual input. Hence, adversary's powers are limited by ensuring that the maximum perturbation that can be introduced is limited to a predefined ϵ. Sometimes adversarial images created by FGSM are highly perturbed and they belong in the region of input space which are actually far away from the input space. Due to this reason, perturbation done by FGSM turn out to be quite high as compared to other methods. *Iterative sign method* is extension of FGSM by applying it iteratively until an adversarial image that classifies as target class y is obtained.

2.3. DeepFool

DeepFool is an untargeted attack that was proposed in (Moosavi-Dezfooli et al., 2015). It is an non-targeted attack in that it doesn't try to create adversarial input that causes classifier to give a certain prediction in the output. On the other hand, it tries to modify the adversarial input such that classifier classifies the input as instance of some other class than its true class.

On the high level, idea of DeepFool is simple. Deep neural networks can be thought of as modelling hyperplanes in high dimensions, where each hyperplanes forms a boundary that separates instance of one class from the rest of the classes. DeepFool simply tries to move input in the direction of the hyperplane that is closest to input but the hyperplane separates inputs from class other than the true class of input. Suppose that, true class of an input x is k, then DeepFool looks at all the hyperplane boundaries nearby x and picks a boundary that corresponds to some class other than k and is closest to x. Due to such formulation of attack DeepFool creates adversarial images which have very small amount of perturbation.

2.4. JSMA

JSMA is Jacobian Saliency Map based Attack which was first introduced in (Papernot et al., 2015a). JSMA is a targeted attack like L-BFGS and FGSM. JSMA attack prepares a Saliency map for input features and tries to determine which how input features have effect classifier's decision of particular input. This is determined by differentiating the classifier function C in forward direction. Once derivative of classifier's output w.r.t. input features is determined, we can look for features which have high value derivative. As value of derivative tells us what will be the effect on the classifier function C if we modify corresponding input feature by some amount.

A slightly modified version of this heuristic is applied in JSMA to pick an input feature which leads to larger modification in output of classifier. This procedure is applied iteratively until the input has been classified into desired class. JSMA keeps track of unmodified input features which is a set containing potential features which can be modified. At each iteration it removes currently selected features from this set and goes to next iteration until input is misclassified as desired target class.

JSMA also has other input parameters such as how much distortion should be done in input feature and how many input features can be modified at most to generate adversarial input. The number of features modified by JSMA corresponds to robustness of neural network. That is, if more features need to be modified then neural network is

more robust. The robustness parameter are discussed later in section 5.

2.5. Carlini Wagner

Carlini and Wagner proposed a set of attacks in their paper (Carlini & Wagner, 2016). It is one of the strongest attack known till date. The authors proposed 3 different attacks, each differing in the distance metric used to compare adversarial image and actual image. The L_2 attack proposed by Carlini and Wagner is used in our experiments. Given an input image x and target label $t \neq C(x)$ the attack tries to produce adversarial image w as below:

$$\begin{aligned} &\text{minimize} && ||\tfrac{1}{2}(tanh(w)+1) - x||_2^2 + c \cdot f(\tfrac{1}{2}(tanh(w)+1)) \\ &\text{where} && f(\text{x}) = \max(\max\{Z(x)_i, i \neq t\} - Z(x)_t, -k) \end{aligned}$$

Here $Z(x)$ is the output of classifier's final layer on input x before applying the softmax function. These values are also called as logits. The loss function can be seen as modifying the input w such that the difference between the target class t and any other class $i \neq t$ is at least k.

The authors also proposed L_0 attack method which is very similar to how JSMA works. However, this method also iteratively rejects input features which would not lead to any change in classifier output. At each iteration this method uses the L_2 method described above to figure out which input features do not lead to any significant change in classifier output.

Third method proposed by authors is L_∞ method, which is similar to how L_2 method works. The only difference is that it tries to minimize L_∞ norm between adversarial input and actual input. The method is formulated as below:

$$\text{minimize} \quad c \cdot f(x+\delta) + \textstyle\sum_i (\delta_i - \tau)^+$$

Here $e^+ = \max(e, 0)$. The second term considers sum of all values which are greater than some threshold τ as the norm being minimized rather than just considering the maximum value of δ. The reason is, sometimes by considering only maximum value the optimization function gets stuck in plateau region where even if second largest value is not much different from largest value, the optimizer function ignores the second largest value. This attack also works iteratively and value of τ is updated after each iteration if none of the values in δ are more than τ.

The value of the c used in the two attacks described above is a hyper parameter and it must be selected carefully to obtain good adversarial examples. Sometimes a binary search is applied and attack is executed multiple times to reduce the distance between adversarial image and actual image by finding a good value of c.

2.6. Projected Gradient Decent

Projected Gradient Decent or PGD is a targeted attack which was used in (Madry et al., 2017) to make DNN robust against adversarial attacks. PGD attack is similar to FGSM attack, such that it also uses the classifier loss function to find out direction in which input should be changed so that output class given by classifier is desired target class t. However, how this input is changed is slightly different. PGD is also based on $||L_\infty||$ norm metric.

Just like FGSM, PGD also finds a direction to change input, however in PGD the input is projected back into the $||L_\infty||$ radius ball only when it goes out of the ball. In contrast, FGSM directly multiplies the sign of gradient with the maximum value of ϵ. Projecting input back into $||L_\infty||$ radius ball is done by normalizing the input and then multiplying it with ϵ. Here ϵ is the maximum value of perturbation that can be done. Hence, ϵ determines the radius of $||L_\infty||$ ball.

2.7. Elastic-Net Attacks

Elastic-Net Attacks was first introduced in (Chen et al., 2017). This attack was also used to successfully attack the defence developed in (Madry et al., 2017). The key idea of this attack is to combine the L_1 and L_2 distance metric into single loss function so as to obtain better adversarial images. Given an input x and target class $t \neq C(x)$ the goal of this attack is to find adversarial image x' by solving:

$$\begin{aligned} &\text{minimize} && c \cdot f(x') + \beta||x'-x||_1 + ||x'-x||_2^2 \\ &\text{subject to} && x' \in [0,1]^p \end{aligned}$$

Here β is the regularization parameter and f is same as the one described in Carlini Wagner attack. By changing value of β one can regularize the adversarial image generation process. If the value of β = 0 then the attack is same as the Carlini Wagner attack.

3. Related work

This section describes previous defences against adversarial attacks and attacks against those defences. It also serves as our intuition behind the experiments we conducted.

3.1. Defensive distillation

Defensive distillation is one of the very first proposed defence against adversarial images. It was first proposed by (Papernot et al., 2015b). Defensive distillation tries to make neural network robust by transferring knowledge from one neural network to other. Key feature of this defence was that it does not rely on any kind of adversarial training.

Transferring knowledge from one DNN to other for reducing

the computational complexity of a DNN was first proposed in (Hinton et al., 2015). There the idea was applied to transfer knowledge learnt by a big neural network into a smaller network so that predictions can be made faster. Defensive distillation uses the same approach at its core but not to reduce computational complexity but to make DNN robust against adversarial inputs.

The core idea behind Defensive distillation uses two neural network having same architecture. The first neural network is initially trained on the training data until sufficient accuracy is obtained. Then, outputs from first DNN are extracted for each input in training data. The second neural network is trained on the output class probabilities obtained from first neural network. Hence, training data for second DNN contains the actual outputs rather than one hot vectors from original training data.

Defensive distillation also uses a temperature parameter T in calculation of softmax function. The formula used for softmax is shown below. The value of temperature T was kept by author.

$$F_t(x) = \frac{e^{Z_t/T}}{\sum_{n=1}^{N} e^{Z_t/T}}$$

The intuition behind this approach is that, using the class probability vector as training output rather than one hot vectors helps in smoothing the probability function modelled by DNN and hence improves robustness of DNN in the input region space close to actual inputs. Moreover, using high value of T prevents the first model from over fitting the output probabilities so that the output probability vectors does not become close to one hot vector. Also, at the time of evaluation and adversarial input generation the temperature was set to 1.

However, later it was shown (by (Carlini & Wagner, 2016)) that robustness achieved by Defensive distillation is because of absence of methods for crafting adversarial examples efficiently. The use of high temperature causes the outputs to be smoothed out but when it is reset to 1 during evaluation and adversarial input generation causes the outputs get very close to one hot vectors. Because of this, the methods used for generating adversarial examples in (Papernot et al., 2015b) were useless because the target class probability becomes nearly 0 and floating point numbers are unable to capture to such precision. Hence, the attack methods were stuck in plateau region of the loss function.

Authors of (Carlini & Wagner, 2016) also demonstrated that if adversarial inputs were to be crafted by keeping value of temperature same as the training time then it is possible to generate adversarial examples. Hence, authors in (Carlini & Wagner, 2016) proposed a new approach for generating adversarial images by considering the value of logits rather than probabilities.

The Teacher Student framework that we experimented is inspired by this approach. The original Defensive distillation didn't use adversarial training, hence, we experimented to see what happens if we actually used adversarial training.

3.2. Madry defence

Madry defence was first proposed in (Madry et al., 2017). The Madry defence is models adversarial input crafting and robustness as an optimization problem which is formulated as below. Given parameter θ of network and loss function L, Dataset D, we try to find parameter θ as below:

$$\begin{aligned} &\min_\theta \quad \rho(\theta) \\ &\text{where,} \quad \rho(\theta) = E_{(x,y)\sim D}\left[\max_\delta L(\theta, x+\delta, y)\right] \end{aligned}$$

The inner maximization problem allows to find out the maximum loss by figuring out radius δ in L_∞ radius ball such that $x+\delta$ and x has same output class y. The outer minimization problem finds out the parameter θ such that $\rho(\theta)$ is minimized. Hence, the largest radius δ being minimized.

The authors of the paper attempt to solve above problem using first order optimization methods. They use adversarial attack methods such as PGD to solve inner maximization problem. Multiple such attacks can be used since the larger the sample size the more accurate solution to inner maximization problem will be obtained. Once inner problem is solved, outer minimization problem is solved which is same as training the DNN on the adversarial images generated by inner maximization problem and hence changing parameters such that outer cost is minimized. Training network in such manner causes network to be robust in the L_∞ radius ball because loss for all adversarial perturbation is small in such region as result of training.

One important aspect of this defence is the way the authors modelled the perceptual similarity between two images. The most widely used notion for perceptual similarity is the L_∞ distance between adversarial image and actual image. Hence, the authors limited the adversary in terms of how much distortion an adversary can introduce in terms of $||L_\infty||$ norm. However authors in (Sharma & Chen, 2017) demonstrated that L_∞ norm is not an ideal way to measure perceptual similarity between images as the adversarial inputs generated by EAD attack which uses both L_1 and L_2 norms are also perceptually similar. Moreover, the authors were able to attack the Madry defence successfully by generating adversarial images using EAD attack.

3.3. MagNet

MagNet defence against adversarial neural network was first proposed in (Meng & Chen, 2017). MagNet used concept of detection and reformation of classifier input before running

the actual classifier on the predicted input. The key idea used in MagNet was the use of detection method that was independent of the underlying classifier model.

MagNet used an auto-encoder network on the training dataset for detection of adversarial input. The key idea was that if the output of the classifier and input of the classifier differ significantly then there is high chance that input had adversarial perturbation. That is, if reconstruction error is small then there is high chance that input actually belongs to the input dataset. However this method fails in detecting adversarial inputs for which reconstruction error is not that high. Thus authors use the output probabilities given by the underlying classifier and measure the Jensen-Shannon divergence between $C(x)$ and $C(ae(x))$ where $ae(x)$ is output obtained from auto-encoder. Note that this detection is now dependent on target classifier, however, it still does not need to extract any special knowledge from target classifier during training.

MagNet also used a reformer network which reconstructs the adversarial inputs and projects it back into the input space before applying the target classifier on input. This reformer network also uses auto-encoder and it is trained on adversarial examples obtained from target classifier.

The authors demonstrated that their framework is robust against adversarial attacks. They also utilised an ensemble of detectors to evade adversarial attacks that are successful only on one specific detector. However, authors of (Carlini & Wagner, 2017) later shown that it is possible to break the defence of detectors in MagNet by using an ensemble attack on larger set of detector ensembles that adversary trains independently. This allows an adversary to generate adversarial inputs which have high probability of success against ensemble used by MagNet.

4. Proposed approach

This section details our experiments and models.

4.1. Teacher student framework

Teacher student model is similar to the one that was used in defensive distillation. It tries to combine an ensemble of teachers with a student and uses knowledge extraction methods to see if it improves robustness of DNN. Architecture of all models are same.

First we train a set of teacher models. Each teacher model is trained on same dataset. Student is also trained on the same dataset separately. We call this phase natural training where all models are trained on original dataset consisting of only actual inputs.

Next phase is adversarial training, here we train teachers by generating adversarial inputs on each teacher model separately. Each teacher model utilizes different attack. The attacks that we used in this phase are Madry Attack, EAD, FGSM, Carlini and Wagner L_2 variant, DeepFool.

Teachers and students are trained on adversarial inputs on sequential manner. Initially, we extract a set of adversarial inputs from all teacher models, then we train student on these adversarial examples. We don't use one hot vector but we use probability vectors of actual image corresponding to adversarial image, which we extract from teachers models. Next, teachers are trained on same adversarial inputs. However, training of teachers is limited to the adversarial examples extracted from that teacher model.

This is done multiple time. At the end of every 3 iterations, we retrain all models on original dataset. At the end of each iteration we test adversarial robustness of both teachers and student.

4.2. Random permutation model

In this experiment, we again use concept of knowledge transfer among models but in slightly different manner. Here, we first train a single teacher model, which is trained on actual dataset.

We use a key matrix which is used to permute the pixels in the original images. We consider different settings where we permute different subsets of pixels. Here, which pixels should be permuted are chosen uniformly randomly. Once we have our key, we permute inputs in our training dataset using this key.

We, then, train another model, called RandPerm, model on this permuted images. Here architecture of student and teacher is same. We train RandPerm model on feature vector layer rather than output layer. Feature vector layer is the layer just before inputs. We extract this feature vectors for original non-permuted images from teacher model and train student upto this layer.

During the time of prediction, we use the same weights used by teacher model for generating output from feature vector layer. Training on feature vector layer is done because we want DNN to have predictable structure when inputs are permuted. This way we can be sure that the outputs generated at feature vector layer by RandPerm model matches the output generated by teacher model. This incorporates a reconstruction behaviour.

4.3. Attacks

In both of our experiments, we have only considered Black-Box adversaries which work upon transferability of neural networks. Our threat model assumes that adversary knows the architecture and training dataset. Our break model is situation where adversary has succeeded in transferring a

significant number of inputs.

We evaluate our models on both target and non-targeted attacks. Recent research (Liu et al., 2016) has shown that getting high transferability in targeted attack is a computationally hard task. We have also confirmed this hypothesis during our evaluation. However, we also noticed that transferability is very high when non-targeted attacks are being used.

5. Evaluation

This section contains evaluation of our experiments and various evaluation parameters that we have considered in our experiment.

5.1. Experiment setting and Evaluation Metrics

All of our experiments are done using MNIST (LeCun & Cortes, 2010) dataset. We have used the DNN model used in (Madry et al., 2017) as it is. We have used various attacks in our experiments. It is not possible to enumerate all parameters of all attacks, however following are key parameters that we have used in our experiment. In L_∞ attack we allow maximum distortion of 1.0 in both Madry and FGSM attack. For EAD and Carlini and Wagner attack we have used 2 binary search steps and maximum iterations of 100.

We have used several parameters to measure robustness of out DNNs.

Robustness:- Robustness is usually measured mean as amount of perturbation required to generate an adversarial input from original image. The distortion is generally measured as distance between original image and adversarial image. Different attacks use different distance metric and so we have used several distance metric for measuring robustness.

For Carlini Wagner and EAD attack we use following formulation:

$$\rho(x) = E_{x \sim D} \frac{||x - x'||_2^2}{\#pixels}$$

For Madry attck, FGSM we use following formulation:

$$\rho(x) = E_{x \sim D} \frac{||x - x'||_\infty}{\#pixels}$$

Along with robustness we also measure successful attack perturbation which is same as robustness but average is taken only over those inputs which are successful in adversarial attack.

Sometimes we also refer to adversarial attack success as robustness. We have explicitly mentioned when we mean it as accuracy and when it is used to refer mean perturbation.

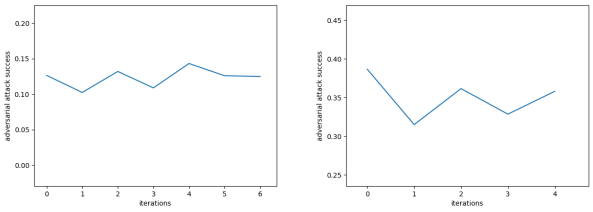
5.2. Teacher student framework

We trained teacher student model in targeted as well as non-targeted setting as described in experiment section. We trained all models on MNIST dataset and all the models had same architecture. We evaluated our models after every iteration.

All models are trained for 5 epochs initially on original training dataset. Adversarial training is done for 3 epochs only. Retraining on original dataset is done for only 1 epoch.

We observed that most of the time targeted attacks weren't able to transfer across the models. We found that combined average of transferability 10% from all of our teacher models. This is consistent with the observations mentioned in (Liu et al., 2016). Hence, we figured that training students by extracting knowledge from targeted setting is unfruitful. Hence, we next evaluated our framework on non-targeted attacks.

On non-targeted attacks we found transferability of 35% on an average. This scenario was better than targeted as some knowledge was being extracted from teachers and being transferred to students. However, we found that even after running the framework for 5 iterations student models robustness didn't improve much significantly.



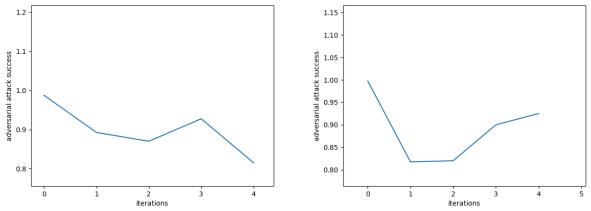
(a) Figure 1 (b) Figure 2

Figure 1. Figure (a): Attack accuracy of student model in targeted setting. Figure (b): Attack accuracy of student model in non-targeted setting.

From the graph shown above it is clear that even though student does improve a little, its not much significant. However, it is completely possible that student's robustness improves as teachers become better and better with their own training on adversarial inputs. However, this would require much larger training iterations for which we didn't have adequate resources.

However, even after that, it is quite possible that student will be vulnerable against one or other attack. This is because recently a research (He et al., 2017) has shown that it is quite possible to break an ensemble of models by crafting adversarial inputs on ensemble of models and performing black box attack. Hence, intuitively it makes sense that

if teacher models are vulnerable then student would also be vulnerable because student only trains on knowledge extracted from teachers.



(a) Figure 1 (b) Figure 2

Figure 2. Figure (a): Attack accuracy on Madry attack model. Figure (b): Attack accuracy on FGSM attack model.

From our experiments we also noticed that the adversarial attack accuracy decreased only for Madry attack and FGSM attack. That is, teachers who were using Madry and FGSM training as their attack method were the only ones becoming robust. Their graph of adversarial attack robustness is shown above.

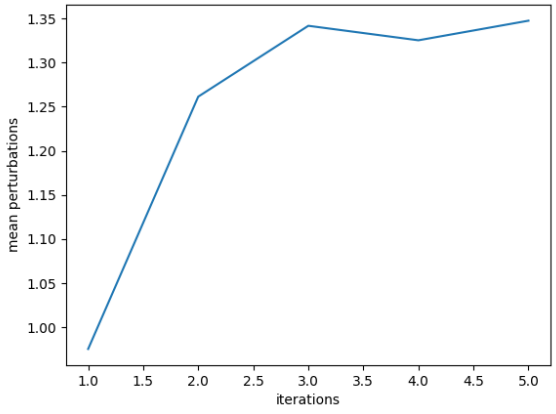


Figure 3. Mean perturbation required for adversarial attack

The graph of mean perturbation is shown above. From this graph we conclude that even though adversarial attack accuracy is not changing much the mean distortion required for attack is indeed increasing. Hence, it seems that if we perform training for more iteration attack accuracy is likely to decrease by increasing robustness. Note that here we have used L_2 distance for measuring mean perturbation.

Hence, we conclude our teacher student experiment as inconclusive because it seems that student will become more robust as teachers become robust over the time.

5.3. Random permutation model

Random permutation model is model for which we use randomly shuffled input images as training data. Here, also, we have used MNIST dataset and architecture of both our models have been kept same. We trained all models for 5 epochs only.

The attack that we utilised here is Madry attack proposed in (Madry et al., 2017). We found it to be most effective against for a black box adversary because it had very high transferability in both targeted and non-targeted setting. Here we have considered only black box attacks. The reason for this is that the key used for random permutation is supposed to be kept secret and adversary doesn't have a way to obtain the key used for training a particular model. We also considered Carlini and Wagner L_2 attack proposed in (Carlini & Wagner, 2016), however we observed that crafting transferable adversarial inputs is computationally costly and requires carefully tuned hyper-parameters of attack even in non-targeted adversarial setting.

We performed this experiment in 3 different settings. First we modifies only a subset of input pixels, 50 and 100 respectively. In the last trial we modified entire image and looked at the results. Again we ran experiment in both targeted and non-targeted setting. To be able to compare improvements by RandPerm model, we train a third model called as natural model which is also trained on same dataset. This model represents effectiveness of attack against models which uses same training data as black box adversary. We craft adversarial examples from BB (black box) model and apply it to all three types of models that we have considered here. We also apply same inputs to natural model.

Table 1. Non-targeted adversarial attack success on different RandPerm models

Model	Test accuracy	Attack success
BB model	98.4	100
Natural model	-	97.8
50-model	98.7	97.6
100-model	98.9	97.5
Full-model	95.8	**82**

First we did experiment in non-targeted setting. Above table shows that when we use only a 100 or 50 pixels and shuffle them, it doesn't really decrease attack accuracy. However, when complete shuffling of input is done, the adversarial attack success indeed decreases, but still not as significant as we would want it to be. However, test accuracy of 95.8% on RandPerm model which uses full permutation does show that neural network is able to find patterns even when it doesn't make sense to a human.

Next we did experiment in targeted setting. Above table

Table 2. Targeted adversarial attack success on different RandPerm models

Model	Test accuracy	Attack success
BB model	99	100
Natural model	-	43.7
50-model	98.9	40
100-model	99	42
Full-model	95.5	**18**

shows the adversarial robustness of different RandPerm models. Our results in above table shows that randomizing model input clearly decreases success of adversarial attacks. However change here is still not significant as we would want it to be. Moreover, we have no idea on what kind of patterns are learnt by DNN in full permutation setting of experiment. Note that by modifying 50 or 100 pixels we don't see any effect on adversarial attack success.

5.3.1. Are targeted attacks being prevented?

Since only 43.7% of targeted attacks are transferring across black box model to natural model, we believe that the data is not sufficient to answer the question regarding whether targeted attacks are truly being mitigated.

However, non-targeted attacks are highly transferable. Therefore we generated non-targeted attack adversarial images and treated them as if they were targeted attack. This means that once we obtained non-targeted attack images we obtained their outputs on Black Box model and used those outputs as target class to which our model should classify for adversarial success. This can simply be obtained by comparing output of BB model against output of natural model and RandPerm model.

We performed this evaluation for full permutation model. We found that 74% of the images had same label for BB model and natural model. However this number dropped to 32% for full permutation model. This shows that indeed it is possible to avert targeted attack using full permutations. However best we can do seems to be half of the maximum success attack obtained against the natural model.

Intuitively, it seems that number of input features is the limiting factor here. Since there are only 784 input features it is quite possible that all of the input features get modified. We did not perform this experiment for tasks that require images to be highly structured and where permuting input images might make DNN to fail to train.

6. Future work

In future, we could run the teacher student framework model for large number of iterations and check how student improves as teachers become more and more robust. One more thing we can do is to generate adversarial inputs on student and test how many of them transfer back to teachers. Recently there have been many approaches to ensemble attacks ((Carlini & Wagner, 2017), (He et al., 2017)). This could give us a new way to perform ensemble attack using a single model.

We could also try and modify the training paradigm used in teacher student framework and use a paradigm that is similar to the one proposed in Madry (Madry et al., 2017) and adopt it for ensemble setting. Other thing that could be done is to test whether RandPerm model works for other tasks which identification of complex and hierarchical patterns in training data.

Recently there have been many proposed attacks on ensemble models (Carlini & Wagner, 2017; He et al., 2017; Liu et al., 2016). It would be interesting to see their effect on teacher student framework. Training student model by extracting adversarial inputs by performing ensemble attack on teachers can certainly improve learning speed of student.

One could also try to generate the random permutation key used in RandPerm model according to some distribution other than the uniform distribution. One approach worth looking is to generate different permutation keys for each output class based on the class activation maps (Zhou et al., 2016) extracted from original teacher model.

7. Conclusion

We conclude our experiments by summarizing our learnings from them.

1. Teacher student framework shows promise of becoming robust with time. We also learnt that transferability of targeted attacks is low compared to that of non-targeted. If we iterate teacher student framework till all teachers become robust then it seems that student will also start becoming robust to adversarial attacks.

2. RandPerm model shows that adversarial image generation does not only depend on dataset and the model but also on how this data is being fed in model and we can introduce some randomization in this aspect and hope to achieve some adversarial robustness. We also see that we can hypothesize how DNN will behave when subjected to targeted attack by comparing outcomes on non-targeted attacks.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat,

Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Carlini, Nicholas and Wagner, David A. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016. URL http://arxiv.org/abs/1608.04644.

Carlini, Nicholas and Wagner, David A. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *CoRR*, abs/1711.08478, 2017. URL http://arxiv.org/abs/1711.08478.

Chen, Pin-Yu, Sharma, Yash, Zhang, Huan, Yi, Jinfeng, and Hsieh, Cho-Jui. EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples. *ArXiv e-prints*, art. arXiv:1709.04114, September 2017.

Goodfellow, Ian J., Shlens, Jonathon, and Szegedy, Christian. Explaining and Harnessing Adversarial Examples. *ArXiv e-prints*, art. arXiv:1412.6572, December 2014.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

He, Warren, Wei, James, Chen, Xinyun, Carlini, Nicholas, and Song, Dawn. Adversarial example defenses: Ensembles of weak defenses are not strong. *CoRR*, abs/1706.04701, 2017. URL http://arxiv.org/abs/1706.04701.

Hinton, Geoffrey, Vinyals, Oriol, and Dean, Jeff. Distilling the Knowledge in a Neural Network. *ArXiv e-prints*, art. arXiv:1503.02531, March 2015.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL http://doi.acm.org/10.1145/3065386.

LeCun, Yann and Cortes, Corinna. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Liu, Yanpei, Chen, Xinyun, Liu, Chang, and Song, Dawn. Delving into transferable adversarial examples and black-box attacks. *CoRR*, abs/1611.02770, 2016. URL http://arxiv.org/abs/1611.02770.

Madry, Aleksander, Makelov, Aleksandar, Schmidt, Ludwig, Tsipras, Dimitris, and Vladu, Adrian. Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv e-prints*, art. arXiv:1706.06083, June 2017.

Meng, Dongyu and Chen, Hao. Magnet: a two-pronged defense against adversarial examples. *CoRR*, abs/1705.09064, 2017. URL http://arxiv.org/abs/1705.09064.

Moosavi-Dezfooli, Seyed-Mohsen, Fawzi, Alhussein, and Frossard, Pascal. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015. URL http://arxiv.org/abs/1511.04599.

Papernot, Nicolas, McDaniel, Patrick D., Jha, Somesh, Fredrikson, Matt, Celik, Z. Berkay, and Swami, Ananthram. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015a. URL http://arxiv.org/abs/1511.07528.

Papernot, Nicolas, McDaniel, Patrick D., Wu, Xi, Jha, Somesh, and Swami, Ananthram. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015b. URL http://arxiv.org/abs/1511.04508.

Papernot, Nicolas, Faghri, Fartash, Carlini, Nicholas, Goodfellow, Ian, Feinman, Reuben, Kurakin, Alexey, Xie, Cihang, Sharma, Yash, Brown, Tom, Roy, Aurko, Matyasko, Alexander, Behzadan, Vahid, Hambardzumyan, Karen, Zhang, Zhishuai, Juang, Yi-Lin, Li, Zhi, Sheatsley, Ryan, Garg, Abhibhav, Uesato, Jonathan, Gierke, Willi, Dong, Yinpeng, Berthelot, David, Hendricks, Paul, Rauber, Jonas, and Long, Rujun. Technical report on the cleverhans v2.1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2018.

Sharma, Yash and Chen, Pin-Yu. Attacking the Madry Defense Model with L_1-based Adversarial Examples. *ArXiv e-prints*, art. arXiv:1710.10733, October 2017.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian J., and Fergus, Rob. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. URL http://arxiv.org/abs/1312.6199.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott E., Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. *CoRR*,

abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

Zhou, B., Khosla, A., A., Lapedriza., Oliva, A., and Torralba, A. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.