Reproduce the results of the article "Explaining and Harnessing Adversarial Examples"

JIANG Chenao 1 CHENAO.JIANG@ETU.SORBONNE-UNIVERSITE.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

LIU Hanlin 2 HANLIN.LIU@ETU.SORBONNE-UNIVERSITE.FR
Master Systèmes Avancés et Robotiques
Sorbonne Université
Paris, France

Chen Yuwang 3 YUWANG.CHEN@ETU.SORBONNE-UNIVERSITE.FR
Master Ingénierie des Systèmes Intelligents
Sorbonne Université
Paris, France

WANG Haoyu 4 HAOYU.WANG@ETU.SORBONNE-UNIVERSITE.FR
Master Systèmes Avancés et Robotiques
Sorbonne Université
Paris, France

Editor: Machine Learning Avancé (2022-2023)

Lien github: Projet Machine learning avancé

Abstract

Szegedy et al. (2013) argue that the primary cause of neural networks' vulnerability to adversarial perturbation is their linear nature, and uses this view to generate a simple and fast method of generating adversarial examples - FGSM. In this project, we focus on generating adversarial examples and confirming their impact on neural networks using the FGSM method, and implementing adversarial training of linear models as well as deep networks. After the adversarial training is completed, we compare the robustness of the neural network to the adversarial interference before and after training and confirm the effectiveness of the adversarial training.In addition,we will discuss the ability of different architectures of neural networks to resist interference.

Keywords: Adversarial examples, FGSM, Adversarial training, GoogLeNet, Maxout, Linear Model.

1. Introduction

Szegedy et al. (2013) argue that the vulnerability of machine learning models to interference from adversarial examples is due to the extreme non-linearity of deep neural networks. In line with this view, Szegedy et al.devise a fast method for generating adversarial examples, FGSM, and show that adversarial training can indeed greatly improve the robustness and accuracy of neural networks. In their study, they first explain the existence of adversarial examples for linear models, and explain the principle of FGSM. Afterwards, Szegedy et al. implement adversarial training on a variety of linear models and deep networks, and demonstrate the effectiveness and practicality of the training.

The aim of this project is to re-implement the algorithms in the paper and to reproduce all the experimental results obtained. We have approached the reproduction of the paper in three main

directions : firstly, the existence of adversarial examples. Second, the impact of adversarial attacks. Third, the practicality of adversarial training.

To do this, we first need to understand the adversarial example and its existence : we study its rationale and its generation method, FGSM, and generate an adversarial example based on a neural network model. Secondly, we look at the effect of this adversarial example on linear neural network models, in particular logistic regression networks (*simple linear neural networks, softmax, etc.*), and implement the adversarial example training in these networks to see the effectiveness of the adversarial training. Thirdly, we will focus on adversarial training of deep networks and try to demonstrate that deep networks are more robust to adversarial examples than simple linear neural networks. In this section, we will work on generating adversarial examples and implementing adversarial training on Maxout, etc. Fourthly, we will compare the robustness of the adversarial examples and the results of the adversarial training on the above different architectures and hope to try more different approaches based on the adversarial training, such as *early stopping* and expanding the model to improve the accuracy of the model.

2. Presentation of the algorithm

The attacker uses perturbations that are not perceptible to human vision/audition, which are sufficient to cause a normally trained model to output false predictions with high confidence, a phenomenon that researchers call adversarial attacks.

FGSM approach is a white-box attack in which the threat model assumes that the attacker has complete knowledge of his target model, including the model architecture and parameters. The attacker can therefore create an adversarial example directly on the target model by any means. FGSM is a typical one-step attack algorithm who performs a one-step update along the direction of the gradient of the adversarial loss function $J(\theta, x, y)$(i.e. the sign) to increase the loss in the steepest direction.

The aim of the FGSM attack is to disrupt the classification of the network by modifying the pixel values of the input images without modifying the network parameters. Based on the above principles of neural network computation, the loss values can be passed back to the image and the gradient $J(\theta, x, y)$as well as the direction of the gradient $sign(J(\theta, x, y))$can be calculated.The reason for using the gradient direction rather than the gradient value is to control the size of the perturbation.

The classification model updates the parameters by subtracting the gradient from the parameters so that the loss value is reduced and the probability of a correct prediction is increased. Since the attack is intended to make the network misclassify the images, it is only necessary to make the loss increase. Therefore, adding the direction of the gradient to the input image to increase the value of the loss is sufficient to interfere with the network's classification. This is the principle of the FGSM algorithm, which calculates the gradient based on the image and adds the gradient when updating the image.

$$x' = x + \epsilon \cdot sign(\nabla_x J(\theta, x, y)) \tag{1}$$

$\eta = sign(\nabla_x J(\theta, x, y))$is the perturbation, ϵ is the weighting parameter that can be used to control the magnitude of the attack.

3. Data

The dataset used for most of the tests in the article is the Mnist dataset. The MNIST dataset (Mixed National Institute of Standards and Technology database) is a large database of handwritten numbers collected and organized by the National Institute of Standards and Technology, has become a standard test in the field of machine learning. It contains 60,000 training images and 10,000 test images, witch are black and white images, normalised and centred with 28 pixels on each side.

Handwriting recognition is a difficult problem, and a good test for learning algorithms. Secondly, unlike ImageNet, the dataset contains only 10 categories and the image size is smaller, which is more useful for visualising the neural network results. This is the reason why we believe the article makes extensive use of the Mnist dataset.

The MNIST dataset has only ten classes, we tried to use a light version of ImageNet dataset for testing. Tiny ImageNet contains 200 classes, each class contains 500 training images, 50 validation images and 50 test images. However, due to the need to import local data that cannot be used with GPU, and the amount of data is too large and slow to train with GoogLeNet network, we only used convolutional neural network for simple training and testing. The main reproduction is still implemented based on the MNIST dataset.

4. EXPERIMENT

4.1 Existence of adversarial examples

4.1.1 Description of the experience

Lien github: GoogLeNet

Explaining and Harnessing Adversarial Examples(Goodfellow et al., 2014), a classic paper in the field of adversarial examples, the most widely known of which is Szegedy et al.'s proof of the existence of adversarial examples and the validity of FGSM on GooLeNet.

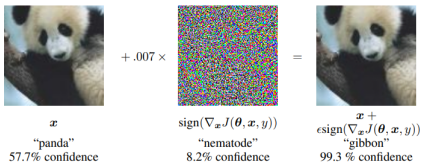

Figure 1 – A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet.

By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. (cf.FIGURE 1)

GoogLeNet is one of the most successful models of the earlier years of convolutional neural networks and is a type of convolutional neural network based on the Inception architecture. It utilises Inception modules, which allow the network to choose between multiple convolutional filter sizes in each block. As mentioned above, we do not currently have a license for ImagNet, so the entire test will be conducted on the Mnist dataset. After building and training the GooLeNet network, we obtained an accuracy of 98% on the valid set. (cf.FIGURE 2)

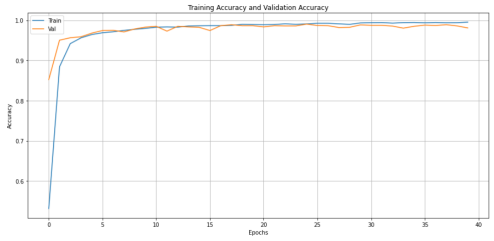

Figure 2 – Accuracy of GoogLeNet on the training set and valid set

We will then do two tests, firstly, to confirm that the adversarial examples do have an effect on the network, and to observe the confidence level of the network classification results. Secondly, observe the effect of different levels of interference on the accuracy of the network.

4.1.2 Results

After generating the model, we first generated a sample adversarial case using the FGSM method (cf.FIGURE3). We note that this adversarial examples is of the same form as the image in the article but of a different size, due to the size of the image in the database.

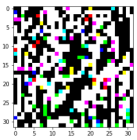

Figure 3 – Adversarial examples generated by GoogLeNet on Mnist

In Szegedy et al. (2013), a mere 0.007 ($\epsilon = 0.007$) interference on the imageNet dataset was used to give the network 99.3% confidence that its incorrect judgments were correct (cf.FIGURE1). Due to the different datasets used, the interference reached 0.2 ($\epsilon = 0.2$)before the network was nearly 100% confident that an incorrect decision was made. However, this still confirms that an adversarial examples of only 0.2 per cent can make the network completely wrong. (cf.FIGURE 4 and 5)

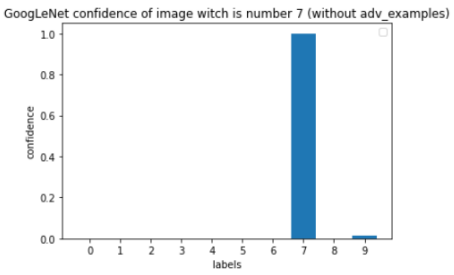

Figure 4 – GoogLeNet confidence without adv examples

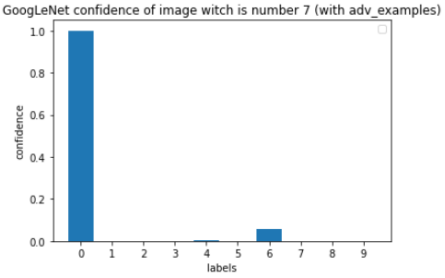

Figure 5 – GoogLeNet confidence with adv examples

Next we observed the effect of different magnitudes of interference on the accuracy of the network, accuracy does decrease with increasing interference. (cf.FIGURE 6) Only 0. 1 perturbation is needed to reduce the accuracy of the network to 45%.

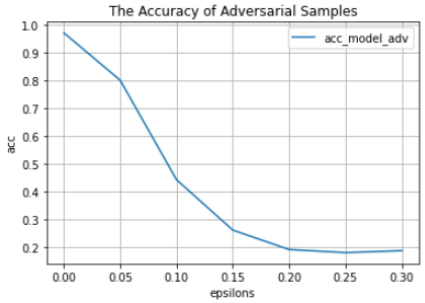

Figure 6 – Accurcy of GoogLeNet with increasing interference

4.1.3 Discussion

Through the above experiments, we learned about the existence of adversarial examples and succeeded in verifying the explanation in the paper : the accuracy of the input features of a neural network is finite, and if a perturbation is smaller than that accuracy, the classifier ignores the perturbation. But the change in activation caused by perturbation can grow linearly with dimensionality. For higher dimensional problems, if there are small changes to the input, then these add up to a large change to the output.

For a high performance network like GoogLeNet a perturbation of only 0.2 is needed to give 100% confidence in a wrong decision, while a perturbation of 0.3 gives an accuracy of only 0.2.

4.2 Linear Model

4.2.1 Description of the experience

Lien github: Softmax
Lien github: Logistic regression

After discussing the existence of adversarial examples and how the FGSM algorithm can quickly generate adversarial-like examples, the article first applies the FGSM method on the simplest logistic regression model so as to understand how to generate adversarial examples in a simple setting.

It is supposed in the article that we train a single model to recognize labels $y \in$-1,1 with $P(y = 1) = \sigma(\omega^T x+b)$ where $\sigma(z)$ is the logistic sigmoid function, then training consists of gradient descent on

$$\mathbb{E}_{x,y\sim p_{data}}\zeta(-y(\omega^T x + b))$$

where $\zeta(z) = log(1 + exp(z))$ is the softplus function.

We can derive a simple analytical form for training on **the worst-case** adversarial perturbation of x rather than x itself, based on gradient sign perturbation. Using the FGSM method for this model, the perturbations

$$\eta = \epsilon sign(\Delta_x J(\theta, x, y)) = \epsilon sign(\Delta_x \zeta(-y(\omega^T x+b))) = \epsilon sign(-\omega^T * \sigma(-(\omega^T x+b))) = \epsilon sign(-\omega) = -\epsilon sign(\omega)$$

and $\omega^T sign(\omega) = \|\omega\|_1$. The adversarial version of logistic regression is therefore to minimize

$$\mathbb{E}_{x,y\sim p_{data}}\zeta(-y(\omega^T \tilde{x} + b))$$

with $\tilde{x} = x + \eta = x - \epsilon sign(\omega)$

$$\mathbb{E}_{x,y\sim p_{data}}\zeta(-y(\omega^T x + b - \epsilon\|\omega\|_1))$$

The above equation is very similar to L1 regularization, but the most important difference is that the adversarial form is training by subtracting the L1 penalty instead of adding the L1 penalty. This means that the L1 penalty can eventually start to disappear if the model is trained with enough accurate predictive that ζ saturates. However, in the case of underfitting, adversarial training can worsen the underfitting.

4.2.2 Results

Since the dataset we use, MNIST, is a multiclassification problem and the logistic regression model is more suitable for a binary classification problem (we will implement a binary classification problem using the logistic regression model later), we currently construct a linear model using the softmax activation function to handle the multiclassification problem. However, the article points out that the L1 weight decay becomes more pessimistic in the case of multi-class softmax regression

because it treats each output of the softmax as independently perturbable, and its overestimates the possible damage from perturbations.

In our model, an L1 penalty with a weight decay factor of -0.3 is added for adversarial training (the maximum value of the perturbation parameter epsilon is 0.3). We add perturbations (gradually increasing the perturbation parameter epsilon) to the test set and compare the accuracy of the original model and the model that has undergone adversarial training on this test set (cf.FIGURE 7). We can see that the adversarial training model is more robust to perturbations, but this adversarial

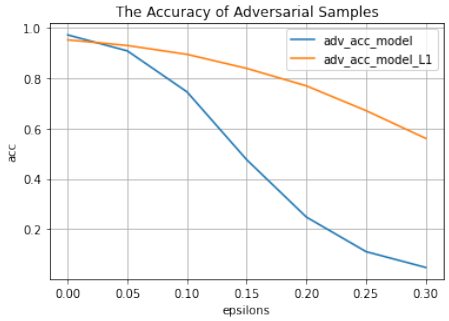

FIGURE 7 – Accuracy of the original model and the model 'Weight Decay'

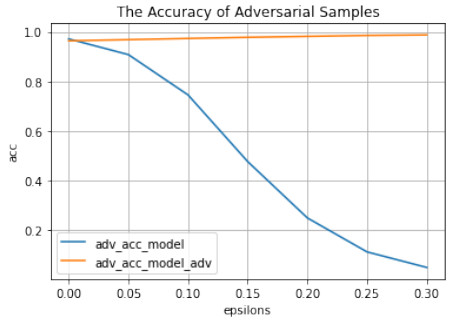

FIGURE 8 – Accuracy of the original model and the model 'adversarial examples'

training does not resist perturbations well (accuracy only 56%) because the weights decay in the softmax regression case overestimating the damage that perturbations may achieve.

We then used another adversarial training method to train the model by creating a training set containing both adversarial and clean examples. And compare the accuracy of the original model and the adversarial training model (cf.FIGURE 8) We can see that the model is robust to perturbations after training with adversarial examples, and even the stronger the perturbation, the higher the accuracy. The current ratio of adversarial examples to clean examples in the training set is 1 :1, and we speculate that the ratio will affect the accuracy, which we will verify in the follow-up.

For the logistic regression network, since there is no suitable binary database, we decided to sample the Mnist database by extracting images with 0 and 1 information to form a new database. The above experimental steps were repeated. We observed that the linear regression network is far more resistant to interference than the other networks, and therefore in that experiment, an L1 penalty with a weight decay factor of -0.5 is added for adversarial training (the maximum value of the perturbation parameter epsilon is 0.5).(cf.FIGURE 9) We were surprised to find that the

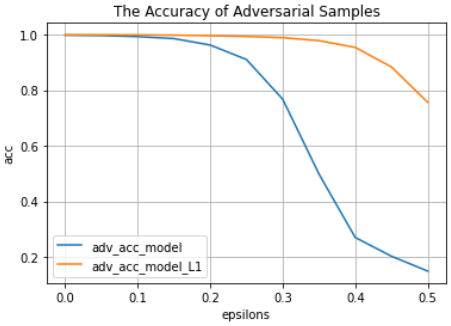

FIGURE 9 – Accuracy of the original model and the model 'Weight Decay'

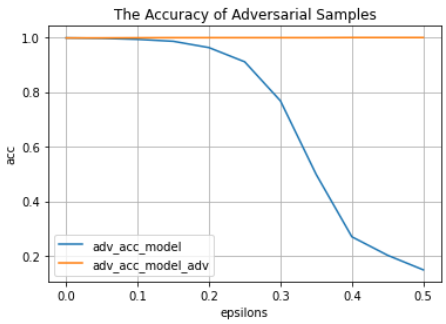

FIGURE 10 – Accuracy of the original model and the model 'adversarial examples'

adversarial training of the linear regression network was very effective, with an accuracy of 98% at a perturbation of 0.3, compared to 80% for the untrained network. At a perturbation of 0.5, the trained network achieves 63% accuracy, while the accuracy of the untrained network is already reduced to 14%. In addition, the linear regression network was much more resistant to perturbations than the other networks, still achieving 63% accuracy at a perturbation of 0.3, while the accuracy of the other networks had dropped to 10% at that point. (cf.FIGURE 9)

4.2.3 Discussion

Figure 11 – Image with the adversarial example

The principle of FGSM is to increase the loss in the steepest direc of the gradient. This is a very abstract explanation, but perturbations for binary networks explain the concept very intuitively. As shown above, the physical shape of the perturbation in image 1 is the opposite of the perturbation in image 2 (cf.FIGURE 11), since a perturbation for either class in a binary network is meant to mislead the neural network into classifying it into another class, so the form of the perturbation is reasonable.

4.3 Deep Neural Network

4.3.1 Description of the experience

Lien github: Maxout

It is clear that standard supervised training does not specify that the chosen loss function be resistant to adversarial examples. This has to be encoded in some way during the training process. In this section, we used two kinds of adversarial training on a simple network (maxout) to improve its robustness. First we added the adversarial loss to the loss function. As described in the paper, its form is as follows :

$$f_{adv}(\theta, x, y) = \alpha f(\theta, x, y) + (1 - \alpha) f(\theta, x + \epsilon sign(\nabla_x f(\theta, x, y), y)$$

As in the paper, we take α always as 0.5. Its role is similar to an effective regularizer. This means that we will constantly update the adversarial sample in our training. We generate adversarial samples at the end of each batch and calculate the losses caused by them. After that we use the obtained new losses to compute the gradient and propagate.

In the second approach, first we trained a model without adversarial training. Then, we generated the corresponding adversarial samples and added these samples to the training set. We used this training set with the adversarial samples to re-train our model. This approach is similar to data augmentation.

4.3.2 Résultats

In this section, we repeated the experiments in the paper and compare the results obtained with those in the original paper. In the first adversarial training we found a significant increase in training time because the number of samples in each batch is doubled in this approach. However, we did not

achieve the same results on the test set. First, in the maxout network paper(Goodfellow et al., 2013), the authors achieved an accuracy of 99.06 % using the simple structure of MLP+maxout+dropout. However, in the github code given by the authors, dropout is not used and the final accuracy is only 97.3%. The best accuracy we have found in other replications is 98.5%. Our accuracy is 98.4% after fifty epochs, which is similar to Maxime Vandegar.

By using eraly stopping, we had an average error rates of 1.35 % with adversarial training and the best error rate of 1.26 %, which are better than error rates without adversarial training (1.7%). After confirming that the adversarial regular term improves the performance of the model on the original training set, we studied whether the model is robust to adversarial samples. The results are shown in Figure 12.

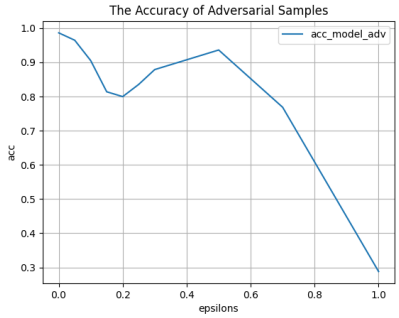

FIGURE 12 – Accuracy of model with adversarial training

We reduced the error rate from 89.4% to 19.6%, which is the highest error with $\epsilon = 0.2$ for all perturbations imperceptible to humans ($\epsilon < 0.5$).

4.3.3 DISCUSSION

We observed that weights changed significantly, and the weights of the models with adversarial training were significantly more limited and interpretative. We can considered adversarial training as adding noise. And our model mined hardly in noisy inputs in order to train more efficiently by considering only those noisy points that strongly resist classification. This regularizer is efficient because the derivative of the sign function is zero or undefined. As loss function of our model is based on gradient descent, it could not react to changes in the parameter. We can also apply adversarial perturbations on hidden layer, it's less efficient. And apply adversarial perturbations on last layer didn't work.

5. Conclusion

The paper "Explaining and Exploiting Adversarial Examples" proposes a new explanation for the existence of machine learning models, including neural networks, for adversarial examples, which is the linearity assumption. Also, the paper proposes a simple adversarial example generation method-FGSM-and then uses the adversarial examples generated by this attack method for adversarial training. By reproducing the results, we first implement the FGSM method and verify the existence of adversarial examples, and then implement several different effective defense methods for different networks. Overall, this paper provides a new perspective on the phenomenon of adversarial examples and offers new defense solutions to reduce their impact.

6. Bibliographie

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. 28(3) :1319–1327, 17–19 Jun 2013.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv :1412.6572, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. arXiv preprint arXiv :1312.6199, 2013

Kui Ren, Tianhang Zheng, Zhan Qin, Xue Liu.Adversarial Attacks and Defenses in Deep Learning[J].Engineering,2020,6(3) :346-360.

LIU Ximeng. Adversarial attacks and defenses in deep learning. Chinese Journal of Network and Information Security[J], 2020, 6(5) : 36-53 doi :10.11959/j.issn.2096-109x.2020071

Références

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. 28(3) :1319–1327, 17–19 Jun 2013.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv :1412.6572*, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv :1312.6199*, 2013.