Defending Against Adversarial Attacks

Brett Gohre, Rene Gutierrez & Keller Jordan

Abstract

Adversarial attacks have been a growing concern as the success of Neural Networks increases. Of particular interest, are attacks that can devised without knowing the particular model used, but only the data distribution, the so called Black Box attacks. In this project we test several ideas to fend off Black Box attacks, Random Compression Matrices, Multi-task Training, Detection using Reconstruction Error and Capsule Networks.

KEY WORDS: Adversarial Examples, Black-Box Attacks, Random Projection Matrix, Multi-Task Training, Reconstruction Error, Capsule Networks, Convolutional Neural Networks.

1. Introduction

There is an increasing concern that Neural Networks are very susceptible to Black Box Adversarial Attacks, with implications that go from identity theft, as in biometrics recognition to safety in the self-driving car industry. So it is of the utmost importance to try to create defenses against such attacks, or at the very least study their characteristics. Here we try several alternatives with varying degrees of success, however even in failure, new insights are gained in exploration.

2. Random Projection Matrices

Idea

Random compression matrices have been found to be able to perform dimensionality reduction exploiting the input data structure, in particular when the data could belong to a manifold. We can make further use of this in two ways, first the dimensionality reduction can help create smaller models that could help robustness. Furthermore, the random nature of the matrix can transform the attack and make it less or not powerful at all.

Method

For this idea we worked with the MNIST dataset. First created 2 sets of adversarial examples based on different architectures. The first set was based on Logistic Regression and consists of 758 adversarial images. The second one was based on a Convolutional Neural Network optimized to 95% accuracy in the validation set, and had 440 adversarial images. Then we tried to attack these architectures plus a Random Compression Matrix architecture, a Random Expansion Matrix and 2 CNN optimized to 90% and 99% accuracy.

The Random Projection architectures where basically a Logistic Regression for which a Random Projection Matrix Φ is applied to the input x, but remains fixed through training. The idea was that since adversarial attacks are based on small perturbations ϵ to the input, then when applying the random matrix the perturbation would be rendered meaningless, that is $\Phi\epsilon$ would not be able to attack as effectively. Furthermore, since the attacker would not know Φ, it would not be able to train attacks specific to it. In this way our random projection matrix can be thought as some kind of code.

Figure 1: Examples of the Adversarial Attacks created under the 2 Architechtures. The top row shows the attacks created with the Logistic Regression, while the bottom was created using a CNN.

(a) Adversarial Attack for 3 (b) Noise

(c) Adversarial Attack for 3 (d) Noise

Results

In figure 5 we can see the corresponding confusion matrices for the true class on the left and for the adversarial class on the right for the examples generated from the Logistic regression architecture. As we can see the adversarial attacks not only make the expansion and compression defenses fail to predict correctly, but are completely fooled by the examples.

Furthermore, in table 2 we can see that the adversarial examples generated by different architectures transfer successfully across architectures. Even fooling completely the randomness induced by the projecting matrices. However it seems that training the CNN to until it achieves higher accuracy provides the best protection,

Table 1: Accuracy for the 5 different methods, using the test set and the adversarial examples generated by Logistic Regression and CNN architectures.

Method	*Test Set*	*Logistic Arch Adversarial*	*CNN Arch Adversarial*
Logistic Regression	92.00%	0.00%	3.41%
CNN 90%	89.82%	8.71%	5.23%
CNN 95%	94.64%	9.50%	9.50%
CNN 99%	98.63%	60.95%	11.36%
Compression Logistic to 196 dim	89.56%	0.00%	5.00%
Expansion Logistic to 1960 dim	91.07%	0.00%	3.64%

Figure 2: Confusion Matrices for the True Target and Predicted Target on the Left and the Adversarial Target and Predicted Target on the Right. The top row shows the results using the a compression matrix, while the bottom row shows the results on an expansion matrix. Both results are for the adversarial examples generated with the Logistic Regression architecturee.

(a)(b) Confusion Matrix for True Target Class Class

(c)(d) Confusion Matrix for True Target Class Class

however it still performs poorly and offers little protection for examples generated on the CNN.

Multi-task Training with Wingman Distribution

White box attacks on a separate classifier trained on (approximately) the same data distributions as a black box classifier target. This suggests the data distribution itself contains some weaknesses that can be independently fleshed out with a variety of different types of classifiers. We seek to explore the outcome of training on an additional new classification task with unrelated data. We will train our model to correctly classify the relevant data samples, while simultaneously training the model to correctly classify the irrelevant data samples.

The intuition behind our motivation is that the weight updates may take on a vastly different trajectory than the weight update trajectory of the vanilla case. Bregmann divergences leads to different update rules and they may lead to varying degrees of robustness if only to attacks based on different divergences. We are curious to find out if multi-task training with the wingman distribution achieve a similar outcomes of defending against attacks.

The assumption is that the attacker does not know what our secret additional domain and task we are training for, and therefore has a worse estimate of the steepest direction toward classifying the input as differently.

Detection of Adversarial Examples using Reconstruction Error

Idea

In the paper introducing Capsule Networks [https://arxiv.org/abs/1710.09829], the capsule network is trained alongside a separate network that attempts to reconstruct the input image from the vector output. The L2 distance between the original image and reconstruction is then added to the loss function and used to train both networks.

When the network is fooled by an adversarial example, the reconstruction produced by the final vector is extremely far from the input image. As a result, it should be possible to use high reconstruction loss as a linear classifier to discern between real inputs and adversarial examples.

Method

We generated another set of adversarial examples using gradient descent. This time we generated 50,000 for MNIST, we show one of such examples in figure 10. We compute the ROC using the Mean Square Error as a threshold using 50,000 real and 50,000 adversarial examples.

Results

As you can see in table the reconstruction error can be used not only to warn against adversarial attacks, but use it also as a linear predictor for adversarial attacks. Finding that setting the threshold for mean square loss at 93 provides a satisfactory defense.

Figure 3: EAdversarial Example and Reconstruction. The leftmost image represents the real image, next to it it's corresponding reconstruction. Then we show the adversarial example and finally the corresponding reconstruction on the right.

(a) Adversarial and Real Reconstruction

GD vs ED

Idea

Using the square loss results in iterative optimization process as gradient descent, however using the relative entropy loss results in an iterative Exponential gradient update iterative process. In this way we propose using different iterative optimization updates to defend against adversarial attacks generated with the most commonly used gradient descent.

Method

Again we generate adversarial examples comming from both updating mechanism.

Results

As we can see in figure .. using EG instead of GD provides robustness to adversarial attacks.

Abstract

Adversarial attacks have been a growing concern as the success of Neural Networks increases. Of particular interest, are attacks that can devised without knowing the particular model used, but only the data distribution, the so called Black Box attacks. In this project we test several ideas to fend off Black Box attacks, Random Compression Matrices, and Capsule Networks with Reconstruction Error based detection.

KEY WORDS: Adversarial Examples, Black-Box Attacks, Random Projection Matrix, Reconstruction Error, Capsule Networks, Convolutional Neural Networks.

3. Introduction

There is an increasing concern that Neural Networks are very susceptible to Black Box Adversarial Attacks, with implications that go from identity theft, as in biometrics recognition to safety in the self-driving car industry. So it is of the utmost importance to try to create defenses against such attacks, or at the very least study their characteristics. Here we try several alternatives with varying degrees of success, however even in failure, new insights are gained in exploration.

4. Random Projection Matrices

Idea

Random compression matrices have been found to be able to perform dimensionality reduction exploiting the input data structure, in particular when the data could belong to a manifold. We can make further use of this in two ways, first the dimensionality reduction can help create smaller models that could help robustness. Furthermore, the random nature of the matrix can transform the attack and make it less or not powerful at all.

Method

For this idea we worked with the MNIST dataset. First, we created 2 sets of adversarial examples based on different architectures. The first set was based on Logistic Regression and consists of 758 adversarial images. The second one was based on a Convolutional Neural Network optimized to 95% accuracy in the validation set, and had 440 adversarial images. Then we tried to attack these architectures plus a Random Compression Matrix architecture, a Random Expansion Matrix and 2 CNN optimized to 90% and 99% accuracy.

The Random Projection architectures where basically a Logistic Regression for which a Random Projection Matrix Φ is applied to the input x, but remains fixed through training. The idea was that since adversarial attacks are based on small perturbations ϵ to the input, then when applying the random matrix the perturbation would be rendered meaningless, that is $\Phi\epsilon$ would not be able to attack as effectively. Furthermore, since the attacker would not know Φ, it would not be able to train attacks specific to it. In this way our random projection matrix can be thought as some kind of code.

Results

In figure 5 we can see the corresponding confusion matrices for the true class on the left and for the adversarial class on the right for the examples generated from the Logistic regression architecture. As we can see the adversarial attacks not only make the expansion and compression defenses fail to predict correctly, but are completely fooled by the examples.

Furthermore, in table 2 we can see that the adversarial examples generated by different architectures transfer successfully across architectures. Even fooling completely the randomness induced by the projecting matrices. However it seems that training the CNN to until it achieves higher accuracy provides the best protection,

Table 2: Accuracy for the 5 different methods, using the test set and the adversarial examples generated by Logistic Regression and CNN architectures.

Method	*Test Set*	*Logistic Arch Adversarial*	*CNN Arch Adversarial*
Logistic Regression	92.00%	0.00%	3.41%
CNN 90%	89.82%	8.71%	5.23%
CNN 95%	94.64%	9.50%	9.50%
CNN 99%	98.63%	60.95%	11.36%
Compression Logistic to 196 dim	89.56%	0.00%	5.00%
Expansion Logistic to 1960 dim	91.07%	0.00%	3.64%

Figure 4: Examples of the Adversarial Attacks created under the 2 Architectures. The top row shows the attacks created with the Logistic Regression, while the bottom was created using a CNN.

(a) Noise (b) Adversarial Attack for 3

(c) Noise (d) Adversarial Attack for 3

however it still performs poorly and offers little protection for examples generated on the CNN.

To analyze the weights obtained after compression and expansion, and compared them to the weights obtained under regular logistic regression, we apply the pseudo inverse of the projection matrix Φ^+. In figure 6 we can see the weights re-expanded or re-compressed to the original size. The weights for the Expansion weights are very close to the regular weights, so it comes as no surprise that the Expansion Logistic Regression fails to defend against adversarial attacks. However, even though the weights from the Compressed Logistic are significantly different this did not prevent it from getting fooled. Also, the weights seem to converge to the regular weights as the level of compression decreases, as we can see in figure 7, giving us a hint of why the network still gets fooled.

GD vs ED

Idea

Using the square loss results accommodates Gradient Descent methods as a good iterative optimization methods,

Figure 5: Confusion Matrices for the True Target and Predicted Target on the Left and the Adversarial Target and Predicted Target on the Right. The top row shows the results using the a compression matrix, while the bottom row shows the results on an expansion matrix. Both results are for the adversarial examples generated with the Logistic Regression architecture.

(a) Confusion Matrix for True Class (b) Confusion Matrix for Target Class

(c) Confusion Matrix for True Class (d) Confusion Matrix for Target Class

Figure 6: Weights for "Neuron 2", for the Compression, Expansion and Regular Logistic Regression.

(a) Compression (b) Expansion (c) Reg

Figure 7: Weights for 2, for the Compression t0 196, Compression to 392, Compression to 588 and Regular Logistic Regression.

(a) 196 (b) 392 (c) 588 (d) Reg

resulting on an updating rule given by:

$$w_{t+1} \doteq w_t - \eta \nabla L(w_t).$$

Figure 8: Adversarial Attacks examples for the SGD on the Center and EG on the right. On the left, we have the true input, and we show the added perturbation next to each adversarial attack.

Table 3: Number of iterations that it takes to fool the two different optimizers SGD and EG.

Method/Optimizer	*SGD*	*EG*
Gradient Ascent	60.9(±32.3)	85.1(±40.5)
Fast Gradient Sign	52.0(±26.1)	91.0(±43.5)

However using the relative entropy loss results in an iterative process called Exponential Gradient, with updating rule given by:

$$w_{t+1} \doteq w_t e^{-\eta \nabla L(w_t)}.$$

In this way we propose using different iterative optimization updates or different losses that could be unknown to the attacker to defend against adversarial attacks generated with the most commonly used square loss.

Method

Again we generate adversarial examples based on a fully connected neural network with 2 layers of size 100 and 10, using GD and EG on MNIST. Then we generated adversarial attacks using to methods; Gradient Ascent (GA) and Fast Gradient Sign (FGS).

Results

In table 3 we can see that generating adversarial attacks is harder for for EG than SGD. Since as the number of iterations increases the magnitude of the perturbation increases, then not only is more computationally expansive, but is easier to observe the perturbation. Again, we can observe this phenomena in figure 8, where the perturbation is clearly bigger for the EG examples.

Additionally in figure 9 we can observe that on average 'fooling' an EG trained FC network requires bigger, but also more structured perturbations. As we can see in the image, it is easy to see that a 3 is captured by the EG perturbation.

Finally, in table 4 we can see that attacks generated with the same number of iterations transfer better from EG to SGD than the other way around. Frustratingly, both sets of attacks transfer successfully to the other optimizer.

Figure 9: Average perturbation to fool a 3. SGD on the Left and EG on the Right.

Table 4: Percentage of successful attacks with examples created from one optimizer to th other. Attacks generated with 200 iterations.

Method/ Src→Dst	*SGD→EG*	*EG→SGD*
Gradient Ascent	67.4%	99.0%
Fast Gradient Sign	88.2%	99.8%

Detection of Adversarial Examples using Reconstruction Error

Idea

In the paper introducing Capsule Networks [https://arxiv.org/abs/1710.09829], the capsule network is trained alongside a separate network that attempts to reconstruct the input image from the vector output. The L2 distance between the original image and reconstruction is then added to the loss function and used to train both networks.

When the network is fooled by an adversarial example, the reconstruction produced by the final vector is extremely far from the input image. As a result, it should be possible to use high reconstruction loss as a linear classifier to discern between real inputs and adversarial examples.

Method

We created a Capsule Network to perform reconstruction. The structure of the network can be observed in figure 9. Using the output reconstruction we compute the Mean Square Error between the original input and the reconstruction. Our hope is that adversarial attacks will have reconstructions that will be more different than the non adversarial ones.

Figure 10: Capsule Network Architecture

In figure 11 we can observe such examples. As we can see the reconstruction for non adversarial attacks seems closer than the adversarial reconstruction. In fact the adversarial reconstruction is a different number. The adversarial reconstruction is not always the same, as we can see in figure 12, the reconstruction for adversarial ones gets reconstructed as 2 or 7 in this particular examples.

Figure 11: Reconstruction Examples. Non adversarial on the Left and adversarial on the right.

Results

In figure 13 we can observe the ROC for attack detection. Surprisingly, we can detect around 75% of the attacks while incurring in a 5% false positive rate, which is a promising result that could get better with finer tune-up.

Figure 12: Reconstruction Examples. Non adversarial on the Left and adversarial on the right.

(a) Re-con-structed to 2

(b) Re-con-structed to 7

Figure 13: Receiving Operating Characteristic Curve for attack detection using different thresholds of MSE.

Bibliography

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. *Dynamic routing between capsules.* In Advances in Neural Information Processing Systems, pp. 38593869, 2017.

Kivinen, J Warmuth, M (1997). *Additive versus exponentiated gradient updates for linear prediction.* Journal of Tnformation and Computation 132, 1-64.