ISBN 978-981-14-1684-2
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
(WCSE 2019 SUMMER)
Hong Kong, 15-17 June, 2019, pp. 117-123
doi: 10.18178/wcse.2019.06.017

Dropout in Testing Phase Makes Adversarial Samples Generation Difficult

Yuan Wang[+], Zhiming Wang, Xucheng Yin, Chao Zhu

School of Computer and Communication Engineering, University of Science and Technology Beijing, China

Abstract. Deep neural network (DNN) brings the rapid development of pattern recognition algorithm. However, experiments show the vulnerability of deep neural network. This paper studied the problem of generating adversarial samples when we adopt dropout in testing phase. Based on MNIST database, we test four adversarial generation algorithms, two types of adversarial samples, and dropout in different layers of DNN. Several conclusions are obtained: (1) Dropout in testing phase makes DNN more robust with tiny performance loss. (2) Dropout in fully connected layer is the most efficient manner to improve the robustness of DNN. (3) Dropout has different impact on different adversarial samples generation algorithms.

Keywords: Deep learning, Deep neural network, Dropout, Adversarial sample

1. Introduction

In recent years, deep neural network (DNN) has brought revolutionary changes in the field of machine learning and pattern recognition. DNN has demonstrated impressive, state-of-the-art, even human-competitive results on many pattern recognition tasks, especially vision classification problems. However, many researchers found that DNN has some vulnerability. That is, models learned by deep learning algorithms misclassify examples that are only slightly different from correctly classified examples.

Szegedy [1] first found that one can cause the network to misclassify an image by applying a certain very small and often imperceptible additive perturbation, which is found by maximizing the network's prediction error. These samples were called adversarial samples. To make matters worse, a wide variety of models with different architectures trained on different subsets of the training data often misclassify the same adversarial example. This suggests that there are some intrinsic blind spots in current deep neural network learned by backpropagation.

The basic definition of adversarial sample is that a misclassified sample x' which is obtained by 'slightly' perturbing a correctly classified sample x. The generating processing can be defined as:

$$\text{argmin}_{x' \in X} \Delta(x, x') \text{ Subject to: } f(x') != f(x) \tag{1}$$

where X is data space, and $\Delta(x, x')$ is some kind of distance such as L-1, L-2, or L-∞ norm, and f(*) is the trained classifier. The output of label for x' can be any label that different with the true label. This kind of adversarial is called *misclassification adversarial*. Further, a more difficult adversarial goal is *target adversarial*, which requires the classifier not only miss classification, but output a predefined target class label. This kind of adversarial samples can be defined as:

$$\text{argmin}_{x' \in X} \Delta(x, x') \text{ Subject to: } f(x') = t \text{ and } t != f(x) \tag{2}$$

where t is the target label.

Rozsa et al. [2] carried out experiments on three adversarial example generation approaches with eight deep convolutional neural networks. They found that adversarial examples are mostly transferable across

+ Corresponding author. Yuan Wang. Tel.: +86-13522656112.
E-mail address: LHMY599@163.com.

similar network topologies, and better machine learning models such as residual network or application of ensembles are less vulnerable to adversarial examples. Carlini and Wagner [3] carried out extensive adversarial attack experiments on 10 defense algorithms, and concluded that adversarial examples are significantly harder to detect than previously appreciated.

Dropout, first proposed by [4], is one of most significant invention in the field of deep learning in recent years. It prevents overfitting by 'dropping out' some unit activations in a given layer, that is, setting them to zero. Thus it prevents the feature co-adaptation of units during the training phase, and can also be seen as a method of network ensemble during the testing phase.

However, up to now, dropout is mainly used in training phase, rarely in testing phase. In other words, in testing phase, no connections or features were dropped out. In this paper, we try the possibility of using dropout in the testing phase. It means some randomly chosen unit activations were set to zero, or disconnected in forward computing. Intuitively, this will cause some of the information to be discarded, and will lead to performance decline and output results inconsistency at different running times. But on the other hand, it will make the generation of adversarial samples difficult, especially target adversarial samples. Because the small and often imperceptible additive perturbations that try to cheat DNN may also be discard in the forward computing process. A large number of experiments confirmed our conjecture.

2. Generating Adversarial Samples

In this section, we briefly introduce four typical adversarial sample generation algorithms.

2.1. L-BFGS

Szegedy et al. [1] proposed an attack algorithm to generate adversarial samples for deep neural network named L-BFGS. It solves the following box-constrained optimization problem by using a box-constrained L-BFGS algorithm:

$$\text{Minimize } \|r\|_2 \text{ subject to: } 1.\, f(x+r) = t \;\; 2.\, x + r \in [0,1]^m \tag{3}$$

where *t* is the target class label, and *r* is the perturbation. Starting from a randomly chosen direction, it aims to find the smallest perturbation in the input space that causes the perturbated image to be misclassified or classified as a predefined target label.

2.2. FGSM

In gradient based methods, gradients are computed by network training algorithms first. Then, instead of using these gradients to update network parameters as in network training phase, gradients are used to modify the original input itself, which is subsequently misclassified by DNNs. Goodfellow et al. [5] proposed fast gradient sign method (FGSM) based on linear behavior of models. It simply adds a smaller variable to the original sample which is a proportional to the gradient sign of loss function.

$$x' = x + \epsilon \text{sign}(\nabla_x J(\theta, x, y)) \tag{4}$$

where θ is the parameters of a model, *x* the input to the model, *y* the targets associated with *x* and $J(\theta, x, y)$ the cost used to train the neural network. Simple as it is, it reliably causes a wide variety of models to misclassify their input. The authors stated that rotating *x* by a small angle in the direction of the gradient also produces adversarial examples reliably.

2.3. JSMA

Papernot et al. [6] proposed a Jacobian based saliency map approach (JSMA) which minimizes a similar optimization problem:

$$\text{argmin}_{\delta_x} \|\delta_x\| \text{ s.t. } f(x + \delta_x) = y^* \tag{5}$$

where y^* is the desired adversarial output. Forward derivative for the given sample is computed by Jacobian of the function learned by DNN. Then, this derivative is used to construct adversarial saliency maps indicating the most sensitive input features to include in perturbation for producing adversarial samples. At last, benign sample is modified by selecting the feature with maximum saliency in each cycle. Their experimental results on MNIST show that any input sample can be perturbed to be misclassified as any target class with 97.10% success while perturbing on average 4.02% of the input features per sample.

2.4. Deepfool

Moosavi et al. [7] proposed an algorithm named 'Deepfool', which is based on an iterative linearization of the classifier to generate minimal perturbations that are sufficient to change classification labels. It approximates classification region with a polyhedron $\tilde{\mathbf{P}}_i$:

$$\tilde{\mathbf{P}}_i = \cap_{k=1}^{c}\{x: f_{\hat{k}}(x_i) - f_k(x_i) + \nabla f_{\hat{k}}(x_i)^T x - \nabla f_k(x_i)^T x \leq 0\} \quad (6)$$

Where c is the class number, $f_k(x)$ is the output of $f(x)$ that corresponds to the kth class, and $\hat{k}$ is true class of sample x. In generating adversarial samples, start from benign sample x_0, it iteratively computes the distance between sample x_i and the complement of the polyhedron $\tilde{\mathbf{P}}_i$, the perturbation vector that reaches the boundary of the polyhedron $\tilde{\mathbf{P}}_i$ is obtained and the current estimate is updated. The algorithm stops when the classifier finally changes its output label.

3. Dropout in testing phase

Conventionally dropout is used only during training phase and is turned off during testing phase. We try to improve the robustness of DNN by using dropout in testing phase. Dropout can be used in different layers of a DNN. For a conventional classification DNN, from low to high, they are input layer, convolution layer, and fully connected layer.

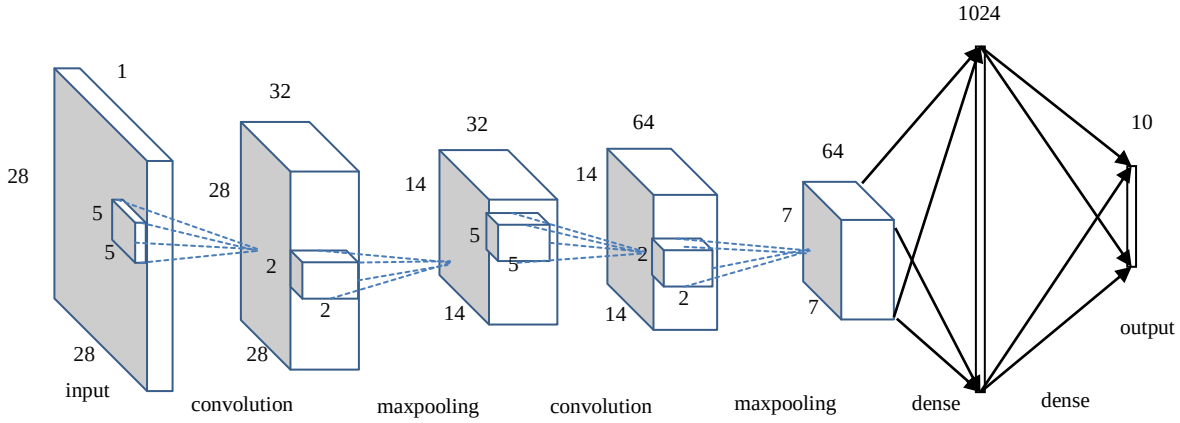


Fig. 1: Classification DNN for Adversarial Attack Test

Fig. 1 shows the classification DNN structure we used for adversarial attack experiments on MNIST database. The input is 28*28 gray hand-written images pixels. There are two convolution layers with 5*5*32 kernels and 5*5*64 kernels, every one followed by a RELU layer (not show) and a 2*2 max-pooling layer. Then, 7*7*64 features were extracted and feed into a 1024 nodes fully connected layers. Finally, the 1024 output were fully connected to a 10 nodes output layer.

3.1. Input dropout

Input dropout means we randomly discard some input data, that is, image pixels in image recognition. It's one of common data augmentation method in training phase. When we carry out input dropout in testing phase, some of pixels are directly neglected and they will contribute nothing to following feature extraction. If by chance these pixels values are contaminated in image acquisition or preprocessing, recognition result will be completely free from these disturbances. Of course, discard pixels will lead to loss of information, and finally results in network performance decline.

In the process of target adversarial sample generation, input dropout forces the attacker to modify more pixels because some of them may be discarded in prediction. However, different adversarial sample generation algorithms behave differently in our experiment (see experimental result in section IV).

3.2. Feature dropout

The function of convolution layer is to extract image features for specific application. Feature dropout means we randomly discard some convolutional results. Some features from some locations will be set to zero, and only partial features are used for subsequent classification.

Similar to input dropout, feature dropout also results performance decline, but it seems smaller (see experimental result in section IV). It also makes target adversarial sample generation difficult and makes some adversarial samples generated with or without dropout correctly recognized.

3.3. Full connection dropout

Fully connected layers act as classifiers in the convolutional neural network. Full connection dropout means we randomly discard some connections from one layer to another layer. Some of weights will be set to zero, and only partial features were used for classification. Similar to input dropout and feature dropout, fully connection dropout also result performance decline, but it seems more subtle (see section IV). It also makes target adversarial sample generation difficult and makes some adversarial samples correctly recognized.

Cheney et al. [8] studied the robustness of convolutional neural networks to perturbations to the internal weights and architecture of the network itself, such as remove nodes, remove synapses and modify synapse weights. Their experimental results show that convolutional networks are robust to a number of internal perturbations in the higher convolutional layers but the bottom convolutional layers are much more fragile. This is in agreement with our experimental results.

4. Experimental Results

4.1. Experiment setup

Our experiments are based on MNIST database and Foolbox [9], a Python package to generate adversarial perturbations and to quantify and compare the robustness of machine learning models. We first train a DNN with 50,000 train set and 40 epochs, which give 99.94% accuracy in training data, and 99.28% in testing data. As dropout has different effects on different layers, we set different drop ratio for different layer. Our adversarial experiments are carried out on 5 different dropout schemes, dropout ratio for input, convolutional and fully connected layers are set to [0, 0, 0], [0.1, 0, 0], [0, 0.2, 0], [0, 0, 0.5], [0.1, 0.2, 0.5] respectively. Table 1 gives the accuracy for different dropout scheme in testing phase.

Table 1: Accuracy for Different Dropout Scheme

Data Set	No Dropout(%)	Input 10% Dropout(%)	Feature 20% Dropout(%)	Fully Connection 50% Dropout(%)	All 3 Kinds of Dropout
Training	99.94	99.87	99.88	99.76	99.49
Testing	99.28	99.17	99.08	99.00	98.77

First, for each of 5 different dropout schemes we generate 1,000 target adversarial samples with L-BFGS, FGSM, and JSMA (as Deepfool not support target adversarial, it's only tested in top-k misclassification). The target label is generated randomly under the condition that it is different from the original label. If one algorithm failed to generate a target adversarial for a specified sample, we just skip it and input the next sample. Two metrics were computed to compare different algorithms and network robustness. One is mean absolute difference (MAD) between original samples and adversarial samples. The other is correct percentage of cross recognition on all 5,000 adversarial samples.

Second, we generate 1,000 top-k misclassification adversarial samples with L-BFGS, FGSM, JSMA, and Deepfool. Top-k misclassification means that the output for true label is out of the maximum *k* values. In our experiment k=3. The network was also tested in 5 different dropout schemes same as mentioned above.

4.2. Target adversarial

Table 2 show the MAD of different algorithms and different dropout schemes in our target adversarial attack experiment.

Table 2: Mean Absolute Difference between Target Adversarial and Original Samples

Attack Method	No Dropout	Input 10% Dropout	Feature 20% Dropout	Fully Connection 50% Dropout	All 3 Kinds of Dropout	Algorithm Mean
L-BFGS	11.286	16.038	16.734	18.512	43.132	21.140
FGSM	50.098	55.243	64.510	74.854	75.377	64.016
JSMA	10.563	11.184	10.899	10.574	11.000	10.844
Dropout mean	23.982	27.488	30.714	34.647	43.170	

The last row gives the mean MAD of all 3 algorithms on different dropout schemes. From it we can found that with more dropout in testing phase makes target adversarial samples generation more difficult for L-BFGS and FGSM. That is, adversarial samples have to change more from original samples. But JSMA is less affected. The last column gives mean MAD of every algorithm on all 5 dropout schemes. From it we can found that JSMA gives best performance with least MAD, followed by L-BFGS, and FGSM performs worst.

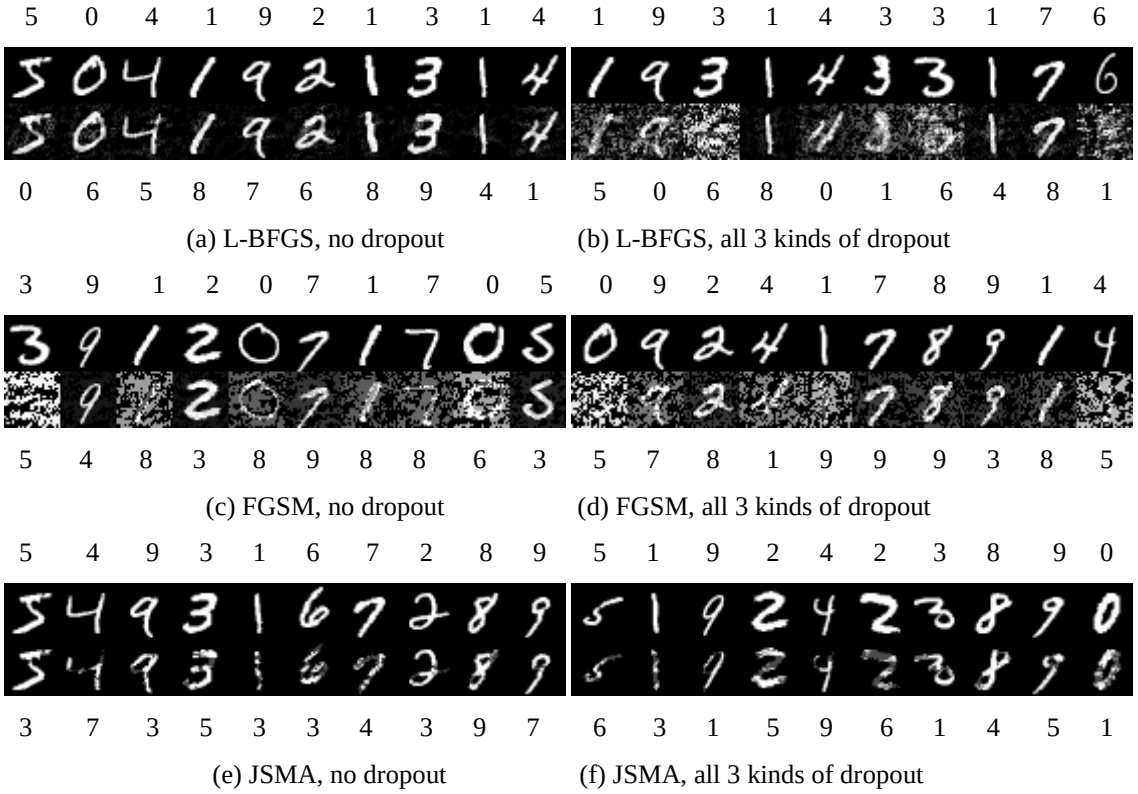


Fig. 2: Original samples and target adversarial samples, top is true label and bottom is target label

Figure 2 show some samples and its corresponding target adversarial samples for no dropout and all 3 kinds of dropout. From it we can see that JSMA give best adversarial samples with less modification.

Table 3 show the cross recognition accuracy of different dropout schemes with adversarial samples generated by different algorithms. Every column corresponding to different dropout test scheme on 1,000 adversarial samples generated with one dropout scheme. Every row corresponding to one test scheme performance on all 5,000 adversarial samples generated with different dropout scheme.

Table 3: Cross Recognition Accuracy on Target Adversarial Samples

Correct ratio(%):L-BFGS, FGSM, JSMA	No Dropout	Input 10% Dropout	Feature 20% Dropout	Fully Connection 50% Dropout	All 3 Kinds of Dropout	Dropout Scheme mean
No Dropout	0.0,0.0, 0.0	2.0, 0.4,5.3	3.9, 0.9,6.9	3.5, 6.3,4.1	13.9, 9.9,27.6	4.66, 3.48,8.78
Input 10% Dropout	1.6,54.0, 1.2	4.3, 5.8,7.8	10.7, 18.3,15.1	7.6, 37.9,13.6	13.1, 16.6,28.5	7.46, 26.52,13.24
Feature 20% Dropout	1.0,58.7, 5.7	5.1, 9.0,21.8	9.7, 12.2,21.2	8.4, 38.4,21.4	13.7, 18.0,37.3	7.58, 27.26,21.48
Fully Connection 50% Dropout	1.5,5.6, 27.8	7.1, 4.6,51.8	11.3, 9.3,57.1	7.1 15.8,34.7	14.2, 15.1,54.4	8.24, 10.08,45.16
All 3 Kinds of Dropout	9.8, 74.2,579,	11.3 27.4,63.1	16.5, 35.9,65.6	9.8 52.1,51.2	14.6, 23.9,52.5	12.40, 42.70,58.06
Data set mean	2.78, 38.50,18.52	5.96, 9.44,29.96	10.42, 15.32,33.18	7.28, 30.8,25.00	13.9, 16.70,40.06	

The last row gives the mean recognition correct ratio on adversarial samples generated with different dropout scheme. From it we can found that when there is more dropout in testing phase, the generated adversarial samples are easier to recognize. Or we can say that adversarial samples generated with less dropout were more reliable to cheat DNN, though pixel values changed less (from table 2). The last column gives mean recognition correct ratio with different dropout scheme. From it we can found that with more

dropout in testing phase makes DNN more robust. That is, by using dropout DNN can correctly recognize more adversarial examples. Mean recognition accuracy over data set (last row) is quite different for L-BFGS. This may be due to the different mechanism of generating adversarial samples by L-BFGS.

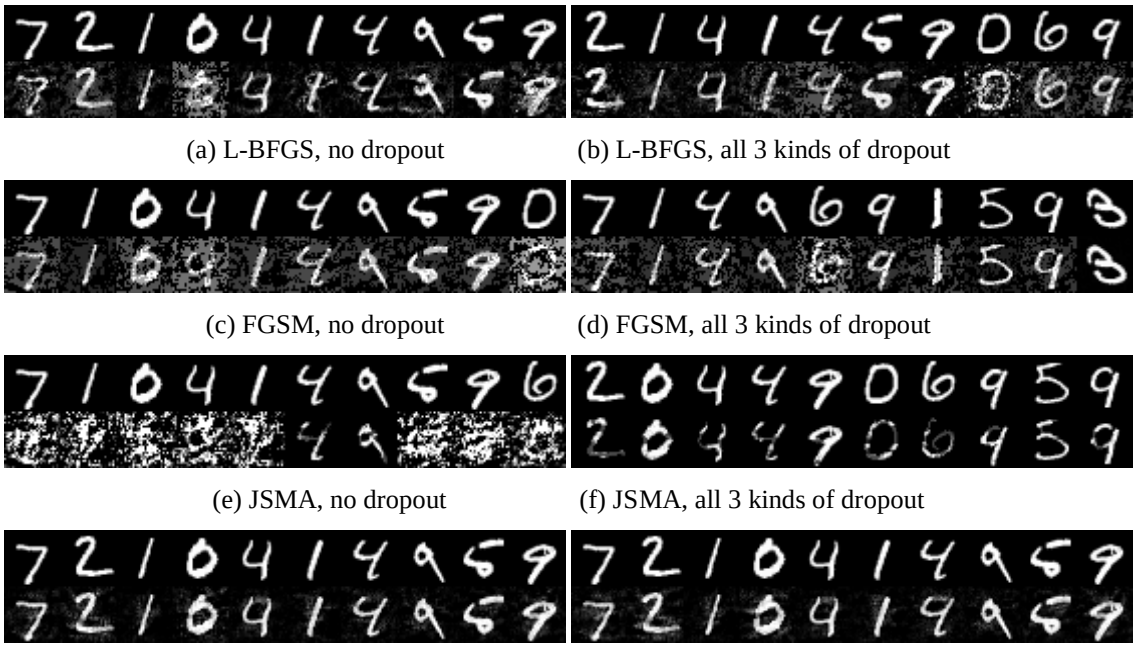
4.3. Top-k misclassification

Table 4 show the MAD of different algorithms and different dropout schemes in our top-3 misclassification adversarial attack experiment.

Table 4: Mean Absolute Difference between Top-3 Misclassification Adversarial and Original Samples

Attack Method	No Dropout	Input 10% Dropout	Feature 20% Dropout	Fully Connection 50% Dropout	All 3 Kinds of Dropout	Algorithm Mean
L-BFGS	27.572	27.966	30.757	24.231	27.203	27.546
FGSM	41.303	36.855	49.328	35.568	38.781	40.367
JSMA	74.074	40.496	36.805	27.73	16.061	39.033
Deepfool	11.867	12.954	13.381	12.026	13.369	12.719
Dropout mean	38.704	29.568	32.568	24.889	23.854	

The last row gives the mean MAD of all 4 algorithms on different dropout schemes. From it we can found that with more dropout in testing phase makes JSMA adversarial samples generation more easy, but there is no obvious trend for other algorithms. This is quite different with target adversarial sample generation. The last column gives mean MAD of every algorithm on all 5 dropout schemes. Deepfool gives best performance with least MAD, followed by L-BFGS, and JSMA and FGSM give poor performance.



(a) L-BFGS, no dropout (b) L-BFGS, all 3 kinds of dropout

(c) FGSM, no dropout (d) FGSM, all 3 kinds of dropout

(e) JSMA, no dropout (f) JSMA, all 3 kinds of dropout

(g) Deepfool, no dropout (h) Deepfool, all 3 kinds of dropout

Fig. 3: Original samples and top-3 misclassification adversarial samples

Fig. 3 show some hand written digital samples and its corresponding top-3 misclassification adversarial samples for no dropout and all 3 kinds of dropout. From it we can see that Deepfool give best adversarial samples with less modification. Table 5 show the cross recognition accuracy of different dropout schemes with top-3 misclassification adversarial samples generated by different algorithms.

Table 5: Cross Recognition Accuracy on Top-3 Misclassification Adversarial Samples

Correct ratio(%):L-BFGS, FGSM, JSMA, Deepfool	No Dropout	Input 10% Dropout	Feature 20% Dropout	Fully Connection 50% Dropout	All 3 Kinds of Dropout	Dropout Scheme mean
No Dropout	0.0,0,0, 0.0,0.0	4.2,2.0, 1.5,0.0	2.1,3.9, 2.9,0.0	9.9,3.5, 9.6,0.2	23.1,13.9, 26.0,1.8	7.86,4.66, 8.00,0.40
Input 10% Dropout	1.2,1.6, 0.6,5.5	5.4,4.3, 2.9,0.4	4.6,10.7, 5.7,0.6	13.1,7.6, 16.9,8.7	23.3,13.1, 25.1,3.5	9.52,7.46, 10.24,3.74
Feature 20% Dropout	2.3,1.0, 0.8,5.8	9.8,5.1, 7.2,1.5	3.8,9.7, 6.4,0.1	16.8,8.4, 23.1,10.8	25.0,13.7, 30.0,4.8	11.54,7.58, 13.50,4.60
Fully Connection 50% Dropout	1.9,1.5, 0.6,1.6	6.5,7.1, 3.6,0.4	4.7,11.3, 3.5,0.2	13.1,7.1, 12.3,2.5	24.1,14.2, 24.9,4.1	10.06,8.24, 8.98,1.76

All 3 Kinds of Dropout	4.2,9.8, 2.0,20.4	11.0,11.3, 9.3,5.7	8.8,16.5, 7.3,5.7	20.7,9.8, 22.8,24.1	24.1,14.6, 29.9,10.2	13.76,12.40, 14.26,13.22
Data set mean	1.92,2.78 0.80,6.66	7.38,5.96, 4.90,1.60	4.80,10.42, 5.16,1.32	14.72,7.28, 16.94,9.26	23.92,13.9, 27.18,4.88	

The last row of gives the mean recognition correct ratio on adversarial samples generated with different dropout scheme. Similar to target adversarial, when there is more dropout in testing phase, the generated adversarial samples are easier to recognize. Though mean recognition accuracy over data set is different for different algorithms, adversarial samples generated with less dropout were more reliable to cheat DNN. The last column also gives similar conclusion that with more dropout in testing phase makes DNN more robust.

Based on the above experimental results, we can find that different adversarial generating algorithms behave differently. Yet we can get some general conclusions: (1) Dropout in testing phase makes DNN more robust with tiny performance loss, that is, more adversarial examples generated with no dropout or less dropout network are correctly recognized. (2) Dropout in fully connected layer is the most efficient manner to improve the robustness of DNN, followed by feature dropout and input dropout. (3) Dropout in testing phase has different impact on different adversarial samples generation algorithms. It makes target adversarial samples generation of L-BFGS and FGSM difficult, but has less impact on JSMA. On the other hand, it makes top-k miss classification adversarial samples generation with JSMA easier, but has less impact on L-BFGS, FGSM and Deepfool.

5. Conclusion

How to make a DNN more robust to adversarial samples is an interesting question. We give intensive experiments of using dropout in testing phase. Experimental results show that dropout in testing phase can improve robustness of DNN with tiny performance loss. As dropout randomly discards some input data or internal features, the output of DNN will be uncertain when we use dropout in testing phase. This will result a little performance decline, but it also makes some adversarial samples be correctly recognized. Different attack algorithms behave quite different, and we need to devote more research into the mechanism of the adversarial sample generation.

6. References

[1] C. Szegedy, W. Zaremba, I. Sutskever, et al. Intriguing properties of neural networks. *International Conference on Learning Representations, 2014*.

[2] A. Rozsa, M. Günther, and Terrance E. Boult. Are Accuracy and Robustness Correlated. *15th IEEE International Conference on Machine Learning and Applications, 2016*.

[3] N. Carlini, D. Wagner. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. *Proceedings of the 10th ACM Workshop on artificial intelligence and security, 2017.*

[4] G. E. Hinton, N. Srivastava, A. Krizhevsky, et al. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580.*

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *Proceedings of the International Conference on Learning Representations*, 2015.

[6] N. Papernot, P. McDaniel, S. Jha, et al. The limitations of deep learning in adversarial settings. *IEEE European Symposium on Security and Privacy, 2016*.

[7] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *IEEE Conference on Computer Vision and Pattern Recognition, 2016.*

[8] N. Cheney, M. Schrimpf, G. Kreiman. On the Robustness of Convolutional Neural Networks to Internal Architecture and Weight Perturbations. *arXiv:1703.08245v1*.

[9] J. Rauber, W. Brendel, and M. Bethge. Foolbox v0.8.0: A Python toolbox to benchmark the robustness of machine learning models. *arXiv:1707.04131v1*.