Additional Experiment Results

An Autoencoder-based Method for Targeted Attack on Deep Neural Network Models

Duc-Anh Nguyen, Do Minh Kha,
Pham Thi To Nga, Pham Ngoc Hung

February 24, 2021

Contents

1 Overview

This document aims to give readers a more comprehensive overview of the proposed method, named *AE4DNN*. In the paper, we used a good enough autoencoder to show the effectiveness of *AE4DNN* against a robust CNN model. This report continues to investigate this capability with different architectures of autoencoder while still using this CNN model. The proposed method is compared with AAE [1]. Specifically, we experimented with a simpler one and a more complex one in terms of architecture. The additional experiment has shown the effectiveness of *AE4DNN* to deploy in practice to test the robustness of DNN.

2 Research Questions

In order to show the effectiveness of *AE4DNN*, the report focuses on addresses the following research questions:

- *Compared to AAE, is AE4DNN more effective in dealing with a set of new input vectors? (generalization ability)* (RQ2)
- *Compared to AAE, is AE4DNN more effective in attacking other models? (transferable ability)* (RQ3)

This report ignores RQ1 and RQ4 because it is out of scope of this report.

3 Attacked CNN model & dataset

Dataset MNIST. The research chooses MNIST [2] which is a popular publicly-available dataset for evaluation. The training set contains 50,000 samples. The test set has 10,000 samples. Each sample on the dataset is an image with 28 pixels in width and 28 pixels in height. The value of each pixel is in range of 0 and 255, which indicates the lightness or darkness of that pixel. However, before training a model on MNIST, every pixel is normalized in the range of [0, 1] for the training purpose. Adversarial example in this experiment is called *adversarial image* for simplicity.

Attacked model. MNIST dataset is trained with the CNN architecture described in [3]. Table 1 presents the architecture of this CNN. The main reason why this experiment chooses this architecture is that this CNN architecture is proven the robustness against adversarial attack. This training uses the proposed configuration for training this CNN: learning rate = 0.1, momentum = 0.9, dropout = 0.5, batch size = 128 and epochs = 50. The CNN model achieves 99.5% accuracy on MNIST, comparable to the state-of-the-art models.

Table 1: The architecture of the CNN used to train MNIST dataset [3].

Layer type	**Description**
Convolution + ReLU	3 x 3 x 32
Convolution + ReLU	3 x 3 x 32
Max pooling	2 x 2
Convolution + ReLU	3 x 3 x 64
Convolution + ReLU	3 x 3 x 64
Max pooling	2 x 2
Fully Connected + ReLU	200
Fully Connected + ReLU	200
Softmax	10

4 Attack

This part investigates the result of attack with different architectures of autoencoder. The conclusion of the experiment is that the architecture of autoencoder plays an important role in the transferability and generalization. In most of the cases, the proposed method produces a better transferability and generalization rate.

4.1 Autoencoder

The experiment chooses three architectures of autoencoder, namely A0 (the autoencoder used in the paper), A1, and A2. We choose these autoencoder models in additional experiment to have a clear comparison between AE4DNN and AAE. Autoencoder A1 is created by 4 layer types: InputLayer, Conv2D, MaxPooling2D, UpSampling2D (in Table 2). Its architecture is shallow and it reconstructs the first 1000 MNIST training images with a MSE loss of 0.05. This model is assumed to be ineffective when attacking the chosen model.

Table 2: The architecture of autoencoder used to attack the CNN model (named A1).

Layer	Layer Type	Input	Output Shape
input_1	InputLayer	(28, 28, 1)	(28, 28, 1)
conv2d	Conv2D	(28, 28, 1)	(28, 28, 16)
max_pooling2d	MaxPooling2D	(28, 28, 16)	(14, 14, 16)
conv2d_1	Conv2D	(14, 14, 16)	(14, 14, 8)
max_pooling2d_1	MaxPooling2	(14, 14, 8)	(7, 7, 8)
conv2d_2	Conv2D	(7, 7, 8)	(7, 7, 8)
max_pooling2d_2	MaxPooling2	(7, 7, 8)	(4, 4, 8)
conv2d_3	Conv2D	(4, 4, 8)	(4, 4, 8)
up_sampling2d	UpSampling2D	(4, 4, 8)	(8, 8, 8)
conv2d_4	Conv2D	(8, 8, 8)	(8, 8, 8)
up_sampling2d_1	UpSampling2	(8, 8, 8)	(16, 16, 8)
conv2d_5	Conv2D	(16, 16, 8)	(14, 14, 16)
conv2d_6	Conv2D	(14, 14, 16)	(28, 28, 1)

Autoencoder A2 is created by 7 layer types: InputLayer, Conv2D, BatchNormalization, Flatten, Dense, Reshape, Conv2DTranspose (in Table 3). Its architecture

is relatively deep and it can reconstruct the first 1000 MNIST training images with a MSE loss of 0.009. This model is assumed to be effective when attacking the chosen model.

Table 3: The architecture of autoencoder used to attack the CNN model (named A2).

Layer	Layer Type	Input	Output Shape
input_4	InputLayer	(28, 28, 1)	(28, 28, 1)
conv2d_5	Conv2D	(28, 28, 1)	(14, 14, 32)
batch_normalization_8	BatchNormalization	(14, 14, 32)	(14, 14, 32)
conv2d_6	Conv2D	(14, 14, 32)	(7, 7, 64)
batch_normalization_9	BatchNormalization	(7, 7, 64)	(7, 7, 64)
flatten_2	Flatten	(7, 7, 64)	(3136,)
dense_4	Dense	(3136,)	(16,)
dense_5	Dense	(16,)	(3136,)
reshape_2	Reshape	(3136,)	(7, 7, 64)
conv2d_transpose_5	Conv2DTranspose	(7, 7, 64)	(14, 14, 64)
batch_normalization_10	BatchNormalization	(14, 14, 64)	(14, 14, 64)
conv2d_transpose_6	Conv2DTranspose	(14, 14, 64)	(14, 14, 32)
batch_normalization_11	BatchNormalization	(14, 14, 32)	(14, 14, 32)
conv2d_transpose_7	Conv2DTranspose	(14, 14, 32)	(28, 28, 1)

The loss of autoencoder seems to have a significant impact on the quality of attack.

4.2 The training process of autoencoders

Each target label requires a separate training process. Therefore, if machine learning testers want to perform targeted attacks for k labels, these testers need to train k autoencoders. For simplicity, this research performs a targeted attack with a target label 7.

Rather than training on the whole dataset MNIST, the experiment selects 1,000 first images on MNIST, denoted by $\mathbf{S}$, to generate adversarial examples. There are two reasons leading to this decision. Firstly, the statistical information on $\mathbf{S}$ satisfies that the numbers of images belong to k labels are nearly equivalent. Therefore, the autoencoder might learn enough necessary transformation to convert an image classified as any label to the target label. Secondly, the experiment aims to show the generalization of the trained autoencoder to generate new adversarial examples. Assume that the autoencoder is only trained on $\mathbf{S}$, which is a balanced subset of the training set $\mathbf{X}$, it still shows its usefulness to generate adversaries from new input images which are never learnt by the autoencoder.

The proposed autoencoder is trained up to 400 epochs with $batch\_size = 256$. We applied early stopping strategy to stop the training process when there is no decrease in the loss over a number of continuous epochs. After the training process, each image of $\mathbf{S}$ would be put into this autoencoder to generate the corresponding candidate adversary. This candidate adversary is then predicted by the attacked CNN model to detect if it is a valid adversary.

There is a problem that *AE4DNN* and AAE use different configuration (i.e. β in *AE4DNN* and ϕ in AAE) to generate adversaries. Therefore, in order to make a comparison, the experiment grouped equivalent configurations. A pair of configuration is equivalent when *AE4DNN* and AAE produce approximate values of $||L||_2$ distance. As a result, autoencoder A0, A1, and A2 have 5, 3, and 3 pairs of configuration, respectively.

4.3 Generalization ability

To show the generalization ability, this experiment used three sets of new input vectors with the size 10,000 images (denoted by *10k-attack*), 20,000 images (*20k-attack*), and 40,000 images (*40k-attack*) for evaluation. These evaluations do not need to train the autoencoder. Because the result of *10k-attack*, *20k-attack*, and *40k-attack* are nearly the same, we compute the average of comparable criterion for simplicity, which is shown in Table 4. Here, *average adversarial rate* $\in [0, 1]$ is the average proportion of the number of adversarial images to the number of input images (i.e. higher value is better).

Table 4: Generalization comparison between A0, A1, and A2. Target label is 7. Better values are marked in bold. δ_1 is the difference between AE4DNN and AAE in Average $||L||_2$, δ_2 is the difference between AE4DNN and AAE in Average Adversarial rate

Config	Average $\|\|\|L\|\|_2$			Average adversarial rate (%)		
	AE4DNN	AAE	δ_1	*AE4DNN*	AAE	δ_2
A0-A	**4.65**	4.76	0.11	**7.5**	7.1	0.4
A0-B	**4.69**	4.76	0.07	**8.5**	7.1	1.4
A0-C	4.99	**4.76**	0.23	**11.7**	7.1	4.6
A0-D	6.49	**6.28**	0.21	**53.5**	39	14.5
A0-E		**6.28**	0.1	**67.7**	39	28.7
A1-A	**6.49**	6.57	0.08	82.8	**82.9**	0.1
A1-B	**6.49**	6.6	0.11	82.8	**84.4**	1.6
A1-C	**6.7**	6.76	0.06	88.7	**89.1**	0.4
A2-A	**4.99**	5.14	0.15	**55.35**	16.37	38.98
A2-B	7.85	**7.62**	0.23	**89.47**	78.32	11.15
A2-C	7.67	**7.62**	0.05	**89.09**	78.32	10.77

In A1-A, A1-B, A1-C set, although average adversarial rate of AE4DNN is smaller than AAE, those distances (denoted as δ_2) are so small in comparison to other values (roughly 2%). It shows that, in general, AE4DNN can generate adversarial images having more effective than ones generated by AAE.

4.4 Transferable ability

The procedure of transferable attack is as follows. Initially, adversaries would be produced by attacking the trained CNN model previously with 1,000 first images of MNIST. After that, these adversaries are put into different DNN models including VGG-13 model, VGG-16 model, LeNet-5 model, and AlexNet model. If an adversary

makes the new model predict as the target label y^*, this adversary could be used to attack this new model. The detail of comparison is shown in Table 5.

Table 5: Transferable rate comparison between A0, A1, A2. Target label is 7. Better values are marked in bold. δ_3, δ_4, δ_5, δ_6 are denoted as the difference of transferable rate between AE4DNN and AAE for attacking VGG-13, VGG-16, LeNet-5, AlexNet respectively

Config	VGG-13 (%)			VGG-16 (%)			LeNet-5 (%)			AlexNet (%)		
	AE4DNN	AAE	δ_3	*AE4DNN*	AAE	δ_4	*AE4DNN*	AAE	δ_5	*AE4DNN*	AAE	δ_6
A0-A	**6.3**	4.9	1.4	**3.7**	3.6	0.1	**1.2**	0.9	0.3	**7.3**	6.1	1.2
A0-B	**7.4**	4.9	2.5	**6**	3.6	2.4	**1.2**	0.9	0.3	**8**	6.1	1.9
A0-C	**31.8**	4.9	26.9	**27.7**	3.6	24.1	**10**	0.9	9.1	**20.4**	6.1	14.3
A0-D	**46.4**	38.5	7.9	**47.1**	39.1	8	**18.1**	6.4	11.7	**39.4**	31.1	8.3
A0-E	**46.4**	38.5	7.9	**47.1**	39.1	8	**18.1**	6.4	11.7	**39.1**	31.1	8
A1-A	**18**	13.8	4.2	**21.2**	11	10.2	**11.8**	0.8	11	0.2	**2.8**	2.6
A1-B	18	**36.4**	18.4	21.2	**43.5**	22.3	**11.8**	11.7	0.1	0.2	**2.6**	2.4
A1-C	27	**51.6**	24.6	30.4	**36.4**	6	2.9	**4.9**	2	**3.9**	0.9	3
A2-A	**18.5**	12.3	6.2	**43.8**	16	27.8	**4.3**	3.3	1	**18.1**	12.2	5.9
A2-B	**67.3**	44.1	23.2	**77.5**	54.1	23.4	**10.8**	5.7	5.1	**59.5**	44.6	14.9
A2-C	40.8	**44.1**	3.3	**81.8**	54.1	27.7	4.5	**5.7**	1.2	**55.25**	44.6	10.65

As shown in Table 5, AAE has a higher transferable ability than AE4DNN in 9 out of 33 cases (27.27 %). Nevertheless, in the rest, there are 24 out of 33 cases (72.73 %) show that AE4DNN has a higher transferable ability than AAE in transferability. Consequently, in general, AE4DNN can attack more effectively than AAE do in black-box setting.

5 Conclusion

In this experiment, besides implementing with a autoencoder in our paper, we additionally implemented our proposed method (AE4DNN) and AAE with 2 more autoencoder models, one seems weak and one seems strong to attack selected classifier. We practically set some pairs of (β, ϕ) for comparison between AE4DNN and AAE fairly. On the whole, the results show that AE4DNN can be used to attack more powerfully than AAE with MNIST classifiers. Besides these results, AE4DNN need to be tested with other complicated datasets to be able to be used in real world in the future.

References

[1] S. Baluja and I. Fischer, "Adversarial transformation networks: Learning to generate adversarial examples," 2017.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[3] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," *CoRR*, vol. abs/1511.04508, 2015. [Online]. Available: http://arxiv.org/abs/1511.04508