Maximising robustness and diversity for improving the deep neural network safety

Bardia Esmaeili,1,2* Alireza Akhavanpour,2 and Mohammad Sabokrou3
1School of Computer Science, University of Guelph, Guelph, ON, Canada
2Shenasa.ai, Tehran, Iran
3School of Computer Science, Institute For Research in Fundamental Sciences (IPM), Tehran, Iran
E-mail: bardia@uoguelph.com

This article proposes a novel yet efficient defence method against adversarial attack(ers) aimed to improve the safety of deep neural networks. Removing the adversarial noise by refining adversarial samples as a defence strategy is widely investigated in previous works. Such methods are simply broken if an attacker has access to both main and refiner networks. To cope with this weakness, the authors propose to refine the input samples relying on a set of encoder–decoders, which are trained in such a way to reconstruct the samples on completely different feature spaces. To this end, the authors learn several encoder–decoder networks and force their latent spaces to have a maximum diversion. In this way, if attacker gets access to one of the refiner networks, other ones can play as a defence network. The evaluation of the proposed method confirms its performance against adversarial samples.

Introduction: Over the past decade, deep neural networks especially conventional neural networks (CNNs) have achieved state-of-the-art performance in a wide range of applications. However, their vulnerability against adversarial samples makes them insecure for critical tasks. They are prone to a class of attacks conducted in order to sabotage their performance referred to as adversarial attacks, which could cause misclassification or confidence reduction in sensitive tasks such as malware/anomaly detection [1, 2] or autonomous driving [3] hence, a measure to face these attacks is of utmost importance. Generally, three strategies are presented: (1) adversarial training [4], (2) defining a robust loss function [5], and (3) refining [6].

The earlier approaches were to learn the targeted CNN on both clean and adversarial samples as training set. This will make the CNN robust against the adversarial samples in the training data; however, as one can speculate, this training approach can only work against defined attacks in training duration. Hence, the performance of such models drop down when facing unseen or rare attacks. The second class of defence is to optimise the reconstruction error, is expected to remove the disruptive noise from its input and deliver a clean version of it. This is the basis of our proposed approach.

An encoder–decoder network, which is learned on clean samples by optimising the reconstruction error, is expected to remove the disruptive noise from its input and deliver a clean version of it.

The idea of using a refiner method (network) such as an encoder–decoder is valid, as long as the attacker does not access the refiner network. Having access to the model and the refiner network allows the attacker able to produce adversarial samples that can trick both the model and the refiner network (See Figure 1). To avoid this weakness, [7] suggests to use several encoder–decoder networks, and in each step exploit one of them as the refiner, randomly. The intuition behind this idea is that the fact that a random procedure is safer than a pre-determined one. The problem is that if several encoder–decoder networks are trained with the same loss function, they follow up each other. This means that the attacker merely needs to access one of the encoder–decoder networks to compromise all. To cope with the mentioned challenges, we propose an efficient solution by taking advantage of randomness and diversity concepts. In contrary to previous methods [6–8], we propose to learn several encoder–decoder networks as refiner networks and force them to repre...

---

Problem definition: It is investigated that a very small but targeted noise such as \( \epsilon \) can compromise the safety of CNNs. Contaminating images \( X \) with such a noise can confuse the CNN \( M \) and makes the CNN misclassify its input (i.e. \( M(X) \neq M(X+\epsilon) \)) where \( \epsilon \approx 0 \). Proposing a defence approach against such targeted noise will extremely help with the safety of CNNs. The duty of this article is to find a (or a set of) Refiner network(s) aiming to remove such targeted noise from samples before feeding them to the main network. As the attacker \( A \) generates targeted noise by having only one of the refiner networks that is, \( (M, R, X) \Rightarrow \epsilon \), and other ones are completely different from that one, the attacker is not to compromise the whole system, even when the attacker has access to both \( M \) and one of \( R \) networks, that is, \( M(R, X) \neq M(R, X+\epsilon) \) where \( R \) is a randomly selected refiner network.

Proposed method: As mentioned, previous proposed techniques for refining the adversarial samples are not robust against those attackers which know the refiner network parameters. Inspired with cryptography methods, we intended to add diversity and randomness to the procedure of defence aiming to make it more robust. Randomness is added by considering several refiner \( R \) networks. These networks in training procedure are impelled to be completely independent from each other. In this way, the attacker \( A \) cannot compromise the targeted network \( M \) by accessing one of the refiners. The proposed approach consists of a classifier \( M(x; \theta) \) as the main network that should be defended against attacks and K refiner (i.e. encoder–decoder) networks \( R_k(x; \theta_k) \). The parameters of \( R_k \) network, that is, \( \theta_k \) are learned in such a way to map the clean input sample \( X \) to a latent space \( l_k \) and recover the sample by decoding such a latent space. Independence among refiner networks is achieved by distorting each \( l_k \) from other latent spaces. Each of samples, before being fed to the \( M \) network, will be processed by \( R_k \), \( (k \in [1, \ldots, K]) \) is randomly selected). As the \( R_k \) is trained to map the input samples to a clean version of them, it can act as a de-nosing network to refine the samples which were contaminated with adversarial noise such as \( \epsilon \).

---

Fig. 1 Vulnerability of encoder–decoder networks as a defense method. \( M \): main convolutional neural network. \( R \): Refiner network. The targeted network for attack is bordered by a green box. As can be seen, \( R \) works efficiently when the attacker has access to the \( M \) network only, not both of them.
As mentioned previously, having only several refiner networks learned in the same way does not work efficiently. To take advantage of all refiner networks, we have suggested those networks refine the input samples based on completely different features spaces. The intuition behind this idea is, the adversarial noise is highly dependent on the learned features by the target neural network. To this end, all of the refinery networks lean on the same training samples, but a new constraint is imposed on their latent spaces (i.e. bottleneck of encoder–decoder networks) to maximise the distance between the learned features for the same inputs.

R: Diverse feature learning: In previous works, encoder–decoder networks are widely used for removing the adversary noise. Similarly, we also learn an encoder–decoder network \( R_1 \) by minimising the reconstruction error on available training samples. Such a network is trained by optimising \( L_1 \) (see Equation (1)):  
\[
L_1 = ||X - \hat{X}||^2, \tag{1}
\]

where \( X \) refers to the training samples, and \( \hat{X} = R_1(X) \). Learning more encoder–decoder networks and selecting one of them (Similar to [7]) by optimising the same loss function (i.e. \( L_1 \)) results in a set of networks with the same vulnerability. As mentioned, to cope with this weakness, we force \( R_i \), reprenting and reconstructing the samples using different features. To this end, we iteratively learn the encoder–decoder networks. Firstly, \( R_1 \) is learned by optimising the \( L_1 \) loss function. Afterwards, the next encoder–decoder, that is, \( R_2 \) is considered to be learned. Not only does \( R_2 \) learn to map adversarial samples to clean ones by optimising \( L_1 \), but also its latent space for the same samples is impelled to be different from the latent space of \( R_1 \). To this end, in training duration, the cosine similarity \( L_2^{j \rightarrow k} \) (between the latent space of \( R_1 \) and \( R_2 \)) is considered to be minimised by optimising \( L_2 \) loss function (See Equation (2)).  
\[
L_2^{j \rightarrow k} = \frac{l_1 \cdot l_2}{||l_1|| \times ||l_2||}, \tag{2}
\]

where (.) is dot product operation and \( || . \| \) is L2 norm. In order to train the third encoder–decoder network, its latent space must not be similar to the latent spaces of previous learned encoder–decoder networks, that is, \( l_1 \) and \( l_2 \). Generally, for training the kth network, in addition to minimising \( L_1 \), its latent space for the same samples must be dissimilar to \( l_1 \cdots l_{k-1} \). In summary, the parameters of the kth encoder–decoder network that is, \( \theta_k \) are learned by optimising the \( L_1 \) loss function. (See Equation (3))  
\[
L_1 = L_1 + \sum_{j=1}^{j=k-1} L_2^{(j \rightarrow k)}, \tag{3}
\]

Experimental results: In this section, we have evaluated the performance of the proposed method on a standard dataset for classification task. The experimental results confirm that the proposed method works better than the other considered methods which are based on refiners.

Dataset: The task we examined was digit classification on the MNIST dataset [9] for training, attacks and evaluation. There are 48,000, 12,000 and 10,000 images for training, validation and evaluation, respectively. All experiments have been done with Python programming language and Tensorflow machine learning platform. For generating adversarial samples, we used a class of white-box attacks known as FGSM from Cleverhans library, which is a well-known Tensorflow library to implement adversarial attacks. White-box attacks are a class of attacks made assuming the attacker has access to the parameters of a network, while black-box attacks do not.

Experimental setup: The main network \( M \) which is considered here for classification task is a LeNet[10] trained on MNIST with Stochastic Gradient Descent (SGD) optimiser, learning rate of \( 10^{-4} \) and batch size of 128. The architecture (a fully connected encoder–decoder network with 400 neurons in its hidden layer) for all of the refiner networks, that is, \( R_{1, \ldots, n} \), are the same. For simplification, \( k \) is selected to be equal to 3. \( R_{1, \ldots, n} \) are trained for 400 epochs with Adam optimiser, a learning rate of \( 10^{-4} \) and batch size of 256. \( R_1 \) is trained by optimising the \( L_1 \), while \( R_2 \) and \( R_3 \) are trained to optimise the \( L_2 \) loss function.

### Table 1. Performance of \( M \) (the classifier) Before and After attacks with a range of value for \( \epsilon \) without any refiner. For \( S_i \), Before refers to attack only on \( M \) (the classifier) part of the sequence, while attack on the entire sequence is defined as After

| \( \epsilon \) | Model | Before | After |
|---|---|---|---|
| 0.1 | \( M \) | 0.9799 | 0.5941 |
| \( S_1 \) (\( R_1 \) trained with \( L_1 \)) | 0.87 | 0.5629 |
| 0.2 | \( M \) | 0.9799 | 0.0659 |
| \( S_1 \) (\( R_1 \) trained with \( L_1 \)) | 0.441 | 0.0868 |
| 0.3 | \( M \) | 0.9799 | 0.0187 |
| \( S_1 \) (\( R_1 \) trained with \( L_1 \)) | 0.0712 | 0.0238 |
| 0.4 | \( M \) | 0.9799 | 0.0184 |
| \( S_1 \) (\( R_1 \) trained with \( L_1 \)) | 0.0228 | 0.0188 |

### Table 2. Average accuracy of \( S_1, S_2 \) and \( S_3 \) after adversarial attack on \( S_1 \) with a range of values for \( \epsilon \) when (1) \( R_1, R_2 \) and \( R_3 \) are trained solely with \( L_1 \) and (2) are trained with \( L_1 \)

| \( \epsilon \) | Models Average | Avg \( S \) \( (R_1 \) trained with \( L_1 \)) | Avg \( S \) \( (R_1 \) trained with \( L_1 \)) |
|---|---|---|---|
| 0.1 | 0.17 | 0.2 | 0.3 | 0.4 | 0.5 |
| \( S_1 \) (\( R_1 \) trained with \( L_1 \)) | 0.703 | 0.332 | 0.209 | 0.045 | 0.0209 | 0.017 |
| \( S_2 \) (\( R_2 \) trained with \( L_1 \)) | 0.638 | 0.388 | 0.318 | 0.182 | 0.113 | 0.075 |

### Table 3. Accuracy for \( S_1, S_2 \) and \( S_3 \) after adversarial attack with \( \epsilon = 3 \) on \( S_1 \) when \( R_1, R_2 \) and \( R_3 \) are trained with \( L_1 \) and also for when \( R_2 \) and \( R_3 \) trained with \( L_3 \)

| \( \epsilon \) | Models Average | \( S_1 \) (\( R_1 \) trained with \( L_1 \)) | \( S_2 \) (\( R_2 \) trained with \( L_1 \)) | \( S_3 \) (\( R_3 \) trained with \( L_1 \)) |
|---|---|---|---|---|
| 0.3 | \( S_1 \) (\( R_1 \) trained with \( L_1 \)) | 0.0124 | \( S_2 \) (\( R_2 \) trained with \( L_1 \)) | 0.0516 | \( S_3 \) (\( R_3 \) trained with \( L_1 \)) | 0.063 |
| \( S_2 \) (\( R_2 \) trained with \( L_1 \)) | 0.2604 | \( S_2 \) (\( R_2 \) trained with \( L_1 \)) | 0.2662 |
the refiner and the main network (\(S_1\)) is attacked, presence of \(R_1\) does not improve the performance. Further results in Tables 2 and 3 support the validity of having multiple refiners and using \(L_1\) as the loss function for \(R_2\) and \(R_3\). The reason is that as you can see in Table 2, when \(S_1\) is under attack, \(S_2\) and \(S_3\), where \(R_2\) and \(R_3\) are trained with \(L_1\), have higher accuracy than \(S_1\) which supports adding multiple refiners. Additionally, \(S_2\) and \(S_1\), where \(R_2\) and \(R_3\) are trained with \(L_3\), have higher accuracy than when \(R_2\) and \(R_3\) are trained with \(L_1\), which supports using \(L_1\). These results confirm that adding multiple refiners with our proposed loss function and adding one of them in a random manner can reduce the effect of adversarial attacks. Our results were reported on FGSM attack, however, it could be generalised to other attacks as well.

The results for every value of \(\epsilon\) prove that adding a second refiner (\(R_2\)) with Equation (3) as the loss function improves the performance after the attack and adding a third refiner (\(R_3\)) with Equation (3) as the loss function improves the performance more than \(R_2\).

Conclusion: We examined the idea of adding randomness and diversity to refiner networks which defend a main network (i.e. classifier) against a class of attacks called adversarial attack. These attacks target the main network by adding noise to input. Adding more refiners and randomly choosing one of them results in randomness while adding a proposed loss function brings about diversity. Our results consistently approve of our proposed method by showing better accuracy for refiners which were trained based on our proposed loss function.

© 2021 The Authors. *Electronics Letters* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

Received: 10 August 2020  Accepted: 8 September 2020 doi: 10.1049/ell2.12070

References

1 Sabokrou, M., et al.: Deep end-to-end one-class classifier. *IEEE Trans. Neural Networks Learn. Syst.* 1–10 (2020)
2 Pourreza, M., et al.: G2d: Generate to detect anomalies. Preprint, arXiv:2006.11629 (2020)
3 Chernikova, A., et al.: Are self-driving cars secure? Evasion attacks against deep neural networks for steering angle prediction. *IEEE Security and Privacy Workshops*, May 2019, pp. 132–137. IEEE, Piscataway, NJ (2019)
4 Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. Preprint, arXiv:1412.6572 (2014)
5 Mustafa, A., et al.: Adversarial defense by restricting the hidden space of deep neural networks. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3385–3394. IEEE, Piscataway, NJ (2019)
6 Sabokrou, M., Khalooei, M., Adeli, E.: Self-supervised representation learning via neighborhood-relational encoding. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8010–8019. IEEE, Piscataway, NJ (2019)
7 Meng, D., Chen, H.: Magnet: A two-pronged defense against adversarial examples. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 135–147. ACM, New York, NY (2017)
8 Samangouei, P., Kabkab, M., Chellappa, R., Defense-gan: Protecting classifiers against adversarial attacks using generative models. Preprint, arXiv:1805.06605 (2018)
9 LeCun, Y.: The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/ (1998). Accessed January 11, 2020
10 LeCun, Y., et al.: Gradient-based learning applied to document recognition. *Proc. IEEE* 86(11), 2278–2324 (1998)