Developing Robust and Lightweight Adversarial Defenders by Enforcing Orthogonality on Attack-Agnostic Denoising Autoencoders Supplementary Material

Aristeidis Bifis
University of Patras,
Patras, GR
bifis@ceid.upatras.gr

Emmanouil Z. Psarakis
University of Patras,
Patras, GR
psarakis@ceid.upatras.gr

Dimitrios Kosmopoulos
University of Patras,
Patras, GR
dkosmo@upatras.gr

1. Experimental setup

We implemented our networks in PyTorch framework and trained them on an Nvidia RTX 2080Ti. For our DAE trained on the MNIST dataset, we used a simple 3-layered fully connected architecture for the encoder and its mirrored architecture for the decoder. We considered a 128-dimensional latent space. Additionally we used Leaky ReLU activation functions with the slope of the negative part of the activations to be learnable by the network with a non-negativity constraint. We also used Adam as our solver with a learning rate of 10^{-3}.

There are two variations of the proposed architecture in the experiments. The first one is a DAE which has no constraints on the weights of the network. The second one, and the best performing based on our experiments, has tied-weights between the encoder and the decoder, meaning that the encoder's weights are used (in their transposed version) as the decoder's.

We used a separate Adam solver for the Lagrangian multipliers of the orthogonality constraint terms of the loss function (to maximize the parameters, based on the objective, instead of minimizing them), with a learning rate of 10^{-7}.

For the DAEs trained on the fashion-MNIST dataset we expanded our architecture by adding convolutional layers before our encoder and additional transposed convolutional layers after our decoder. The details for the two architectures are shown in Tables 1 & 5.

Table 1. Architecture of our DAEs for the MNIST dataset.

DAE
Dense(456)
LeakyReLU()
Dense(292)
LeakyReLU()
Dense(128)
Dense(292)
LeakyReLU()
Dense(456)
LeakyReLU()
Dense(784)

1.1. Evaluating the denoising capabilities of the DAEs

In this experiment we train our DAEs in the task of denoising. During the training phase, the networks take as input data with added samples from various noise distributions. The samples in this phase come from Normal, Uniform, or the combination of both distributions with randomized mean and standard deviation values each time. This way we can avoid the pitfall of our network learning to handle well on one type of noise with specific statistics. Our goal is to produce a network that can perform well on denoising, independently of the statistics of the noise distribution. In each batch:

- first we sample the mean and standard deviation values from a uniform prior, and
- then we randomly make a selection between normal or uniform noise distribution and inherit the selected statistics
- finally we sample from this generated distribution and add the noise samples to our train data.

For the classifiers, we used a convolutional architecture similar to the one in [1]. With this architecture we achieved a classification accuracy of 99.1% . For the substitute model, following [3] & [2], we utilized the architecture A from [3].

To evaluate our denoising results, we performed the classification task on the reconstructed data for various mean and standard deviation values and we obtained the results depicted on the box plot of Figure 2.

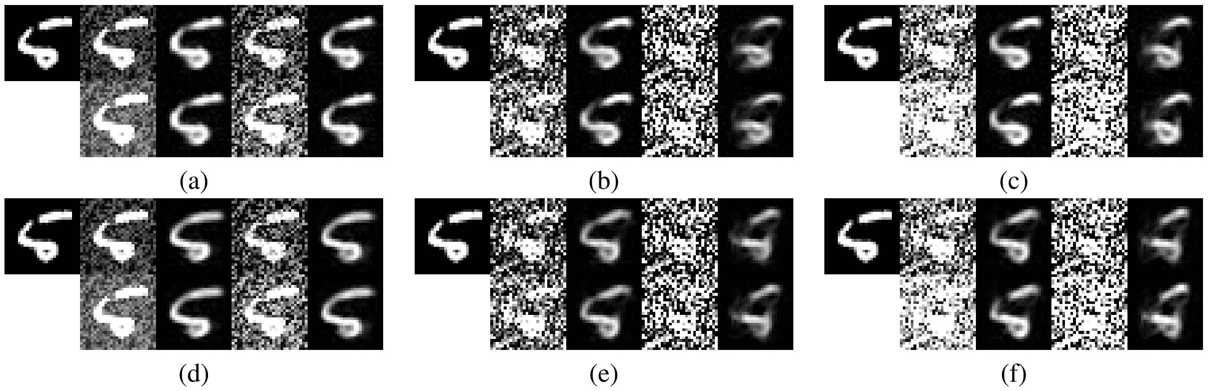

Figure 1. Denoising results of the proposed network with tied-weights ((a),(b),(c)) and full weights ((d),(e),(f)) when applying ((a),(d)) uniform noise, ((b),(e)) gaussian noise and ((c),(f)) a mixture of the two noises, from the combinations of mean = (0.2 , 0.4) (rows) and std = (0.5,1) (columns)

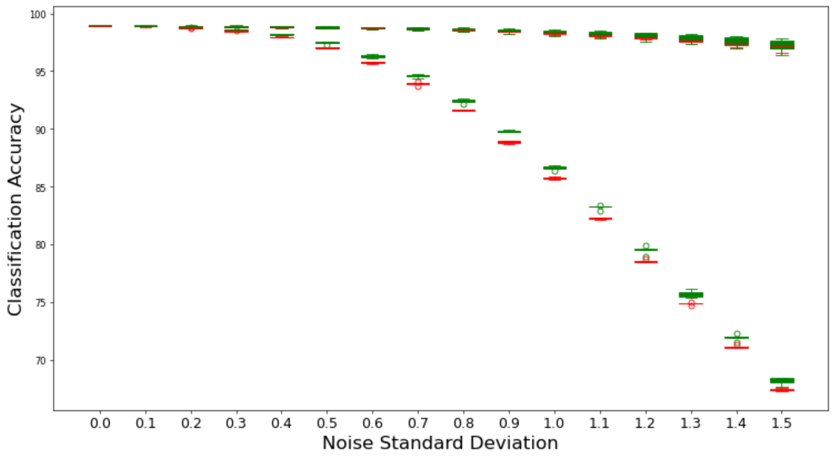

Figure 2. Boxplot depicting the classification accuracies of the classifier, when fed with the DAE reconstructions of data with added noise, sampled from uniform (top curves) and normal (bottom curves) distributions, for the tied-weights (green) and full DAE (red) networks.

Table 2. Classification accuracies under various black-box attacks of the proposed DAEs on Fashion-MNIST

Attack	FGSM	R-FGSM	PGD
Full DAE	78.19	80.53	80.03
Full DAE with Constraints	**78.53**	**80.93**	80.21
Tied-weights DAE	76.99	79.92	81.82
Tied-weights DAE with Constraints	78.06	80.49	**82.23**

Clearly, the classification accuracy is high, even in the cases when the added noise is strong.

To test our trained networks, we sample mean and standard deviation values in a similar way and produce noise samples which are then added to the test data. We first pass the noisy test data through the trained DAEs and perform our classification task on the reconstructed outputs. As we can see from Figure 1, our networks both perform well on the denoising task, since they are able to produce high fidelity images even when the added noise is a mixture of the two distributions where our DAEs have not seen during training (Figure 1.(c) -1.(f) for the tied-weights and full networks respectively).

1.2. Evaluating the DAEs against adversarial attacks on the fashion-MNIST dataset

To test our proposed constraints, we evaluated our setups on different scenarios of adversarial attacks, namely white, semi-white (gray) as well as black box attacks. In this setup we utilized the architecture in Table 6 as our classifica-

Table 3. Classification accuracies under various gray-box attacks of the proposed DAEs on Fashion-MNIST

Attack	FGSM	R-FGSM	PGD
Full DAE	78.7	79.2	79.63
Full DAE with Constraints	**79.62**	**80.36**	80.42
Tied-weights DAE	77.12	78.09	81.35
Tied-weights DAE with Constraints	78.85	79.86	**82.21**

Table 4. Classification accuracies under various white-box attacks of the proposed DAEs on Fashion-MNIST

Attack	FGSM	R-FGSM	PGD
Full DAE	56.06	34.78	35.94
Full DAE with Constraints	**57.02**	**36.83**	38.78
Tied-weights DAE	53.06	32.42	50.42
Tied-weights DAE with Constraints	53.85	32.82	**51.51**

Table 5. Architecture of our DAEs for the fashion-MNIST dataset.

DAE
Conv2d(16,(3,3))
ReLU()
Conv2d(32,(3,3))
ReLU()
Conv2d(64,(5,5))
Dense(352)
LeakyReLU()
Dense(240)
LeakyReLU()
Dense(128)
Dense(240)
LeakyReLU()
Dense(352)
LeakyReLU()
Dense(576)
Conv2dTranspose(32,(5,5))
ReLU()
Conv2dTranspose(16,(3,3))
ReLU()
Conv2dTranspose(1,(3,3))

tion model. Our results can be found in Tables 2, 3 & 4. Our results suggest that the constraints enforced during the training phase do in fact optimize our defence capabilities. On the other hand although our full weights architecture slightly outperforms the tied-weights counter part in some cases, we can justify the tied-weights preference by the fact that the results are pretty close to each other. Also a more fair comparison would utilize the same number of parameters for the two architectures, with more layers for the tied-weights counterpart. The trade-off of the slightly smaller accuracies compared to the significantly smaller number of parameters seems minor.

Table 6. Architecture of our CNN classifier for the fashion-MNIST dataset.

CNN Classifier
Conv2d(32,(3,3))
ReLU()
MaxPool()
Conv2d(64,(3,3))
ReLU()
MaxPool()
Conv2d(128,(3,3))
ReLU()
MaxPool()
Dense(10)
Softmax()

References

[1] Yassine Bakhti, Sid Ahmed Fezza, Wassim Hamidouche, and Olivier Déforges. Ddsa: A defense against adversarial attacks using deep denoising sparse autoencoder. *IEEE Access*, 7:160397–160407, 2019. 1

[2] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, page 506–519, New York, NY, USA, 2017. Association for Computing Machinery. 1

[3] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017. 1