Supplementary to "Improving Adversarial Robustness by Enforcing Local and Global Compactness"

1 Hyperparameters

The hyperparameters for our experiments as Table 1. The hyperparameters of local compactness, global compactness, and smoothness are set to be either 1 or 0, meaning they are switched ON/OFF. Although finer tuning of these parameters can lead to better results, our method outperforms the baselines in these initial settings, which demonstrates the effectiveness of those components.

Table 1. Hyper-parameter setting for the experiment section

	λ_{com}^{lc}	λ_{com}^{gb}	λ_{smt}	λ_{conf}
MNIST	1.	1.	1.	0.
CIFAR-10-CNN	1.	1.	1.	1.
CIFAR-10-ResNet	1.	1.	1.	0.

2 Model architectures and experimental setting

We summarize the experimental setting in Table 2.

For the MNIST dataset, we used the standard CNN architecture with three convolution layers and three fully connected layers described in [2]. For the CIFAR-10 dataset, we used two architectures in which one is the standard CNN architecture described in [2] and another is the ResNet architecture used in [6]. The ResNet architecture has 5 residual units with (16, 16, 32, 64) filters each. We choose the convolution layers as the Generator and the last fully connected layers as the Classifier for ResNet architecture. The standard CNN architectures are redescribed as follow:

CNN-4C3F(32) Generator: 2×Conv(32)→ Max Pooling→ 2×Conv(32)→ Max Pooling→ Flatten

CNN-4C3F(32) Classifier: FC(200)→ ReLU→ Dropout(0.5)→ FC(200)→ ReLU→ FC(10)→ Softmax

CNN-4C3F(64) Generator: 2×Conv(64)→ Max Pooling→ 2×Conv(64)→ Max Pooling→ Flatten

CNN-4C3F(64) Classifier: FC(256)→ ReLU→ Dropout(0.5)→ FC(256)→ ReLU→ FC(10)→ Softmax

3 Choosing the intermediate layer

The intermediate layer for enforcing compactness constraints immediately follows on from the generator. We additionally conduct an ablation study to investigate the importance of choosing the intermediate layer and report natural accuracy and robust accuracy against non-targeted/multiple-targeted attacks respectively. We use the standard CNN architecture (which has 4 Convolution layers in Generator and 3 FC layers in Classifier), with four additional variants corresponding to different choices of the intermediate layer (right after the generator). We use PGD ($k = 100, \epsilon = 0.3, \eta = 0.01$ for MNIST, $k = 100, \epsilon = 0.031, \eta = 0.007$ for CIFAR-10) to evaluate these models. It can be seen from the results as showing in Table 3 that the performance slightly downgrades if choosing shallower layers. The higher impact is expected on a larger architecture (i.e., Resnet), which can be investigated in future.

4 The performance of TRADES

TRADES aims to find the most divergent adversarial examples, while ADV aims to find the worst-case examples to improve a model (see Sec. 2.2 in our paper for more detail). Hence theoretically, there is no guarantee that TRADES outperforms ADV. In practice, the performance of TRADES is influenced by the classifier architectures and parameter tunings. The works [7,4] also reported that TRADES cannot surpass ADV all the time (Table 1 and footnote 8 in [7], Table 1 in [4]), which is in line with the findings in our paper.

5 Further experiments

We conduct an additional evaluation with further state-of-the-art attack methods (e.g., the Basic Iterative Method - BIM [5] and the Momentum Iterative Method - MIM [3]) to convince that our method indeed boots the robustness rather than suffers the gradient obfuscation [1]. Three attack methods PGD, BIM and MIM share the same setting, i.e., $\{k = 100, \epsilon = 0.3, \eta = 0.01\}$ for MNIST and $\{k = 100, \epsilon = 0.031, \eta = 0.007\}$ for CIFAR-10. The result as in Table 4 show that our components can improve the robustness of the baseline framework against all three kind of attacks which again proves the efficacy of our method.

5.1 Loss surface of adversarial examples

We separate adversarial examples into two classes: positive adversarial example which successfully fools a defense method and negative adversarial example which is an unsuccessful attack. The loss surface of positive adversarial example as Figure 1. In particular, both ADV (ADR-None) and our method (ADR+LC) predicted x_a with the label 8, whereas its true label is 3. From Figure 1, it is

Table 2. Experimental settings for our experiments. The model architectures are from [2] [6] and redescribed in the supplementary material.

	MNIST	CIFAR-10 (CNN)	CIFAR-10 (Resnet)
Architectures	CNN-4C3F(32)[2]	CNN-4C3F(64)[2]	RN-34-10[6]
Optimizer	SGD	Adam	SGD
Learning rate	0.01	0.001	0.1
Momentum	0.9	N/A	0.9
Training stratery	Batch size 128, 100 epochs	Batch size 128, 200 epochs	Batch size 128, 200 epochs
Perturbation	$k = 20, \epsilon_d = 0.3, \eta_d = 0.01, l_\infty$	$k = 10, \epsilon_d = 0.031, \eta_d = 0.007, l_\infty$	$k = 10, \epsilon_d = 0.031, \eta_d = 0.007, l_\infty$

Table 3. Performance comparison on different choices of the intermediate layer. The results in each setting are natural accuracy and robust accuracy against non-targeted/multiple-targeted attacks respectively.

	MNIST	CIFAR10
G=2Conv, C=2Conv+3FC	99.52/93.88/92.78	68.78/36.46/21.99
G=3Conv, C=1Conv+3FC	99.44/94.38/93.59	69.17/37.05/22.44
CNN (G=4Conv, C=3FC)	99.48/95.06/94.26	69.08/37.06/22.44
G=4Conv+1FC, C=2FC	99.51/94.38/93.47	69.39/37.31/22.87
G=4Conv+2FC, C=1FC	99.52/94.26/93.45	69.13/37.31/22.57

Table 4. Robustness comparison on the MNIST and CIFAR-10 datasets using Standard CNN with higher attack iteration (i.e., $k = 100$). The results in each setting are natural accuracy and robust accuracy against non-targeted/multiple-targeted attacks respectively.

	Dataset	ADV	ADR-ADV
PGD	MNIST	99.43/93.13/92.09	99.48/95.06/94.26
BIM	MNIST	99.43/93.00/91.70	99.48/94.86/93.99
MIM	MNIST	99.43/94.05/92.63	99.48/95.41/94.56
PGD	CIFAR-10	67.61/32.87/18.74	69.16/36.85/22.71
BIM	CIFAR-10	67.61/32.89/18.71	69.16/36.82/22.69
MIM	CIFAR-10	67.61/33.00/18.59	69.16/36.96/22.56

evident that for ADV, that most of its neighborhood region is non-smooth, resulting in incorrect predictions in almost all of the grid. By contrast, for our method (ADR+LC), the loss surface w.r.t. the input is smoother, resulting in more correct predictions in this neighborhood region. In addition, in our method, the prediction surface w.r.t. the latent feature in the intermediate representation layer is smoother than that w.r.t. input. This means that our local compactness makes the local region more compact, hence improving adversarial robustness.

We provide the loss surface of negative adversarial examples from adversarial training method and adversarial training with our components as Figure 2. Both examples show that the loss function smooth in local region of an adversarial example.

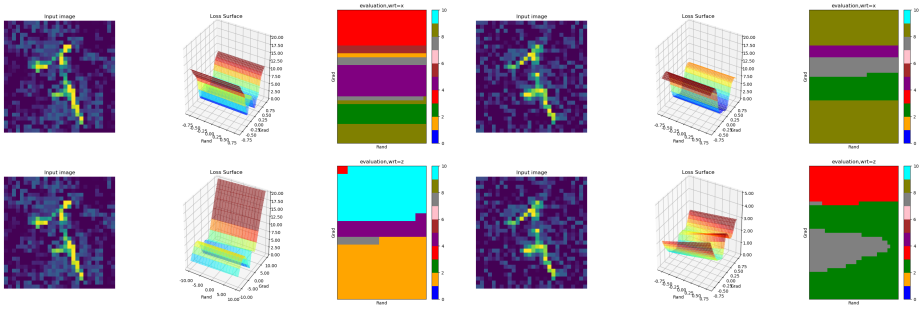

Fig. 1. Loss surface at local region of a positive adversarial example. Top-left: ADR-None w.r.t input. Top-right: ADV+LC w.r.t input. Bottom-left: ADR-None w.r.t latent. Bottom-right: ADV+LC w.r.t latent

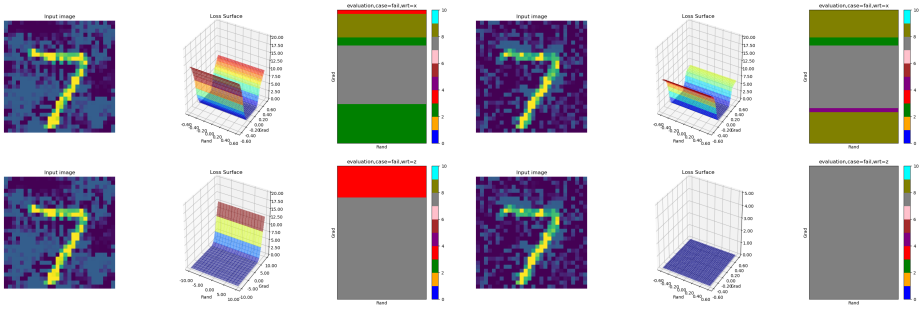

Fig. 2. Loss surface at local region of a negative adversarial example. Top-left: ADR-None w.r.t input. Top-right: ADV+LC w.r.t input. Bottom-left: ADR-None w.r.t latent. Bottom-right: ADV+LC w.r.t latent

5.2 T-SNE visualization of adversarial examples

In addition to positive adversarial examples, we provide the t-SNE visualization of the negative adversarial examples from adversarial training (ADR-None) and adversarial training with our components (ADR+LC/GB) as Figure 3. In adversarial training method, the unsuccessful attacks have been mixed insight the natural/clean data. In contrast, in case adversarial training with our components, the attack representation consistently is separated from those from natural data, similar to positive adversarial examples. Additionally, the unsuccessful attacks in adversarial training have the same confidence level with natural data, while those in our methods are totally different levels. In summary, our method can produce a better latent representation which is well separated between natural data and adversarial example (both positive and negative). This feature can be used for adversarial detection.

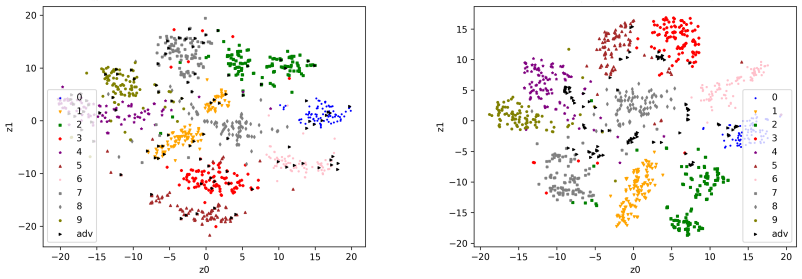

Fig. 3. T-SNE visualization of latent space. Black triangles are (negative) adversarial examples while others are clean images. Left: ADR-None. Right: ADR+LC/GB

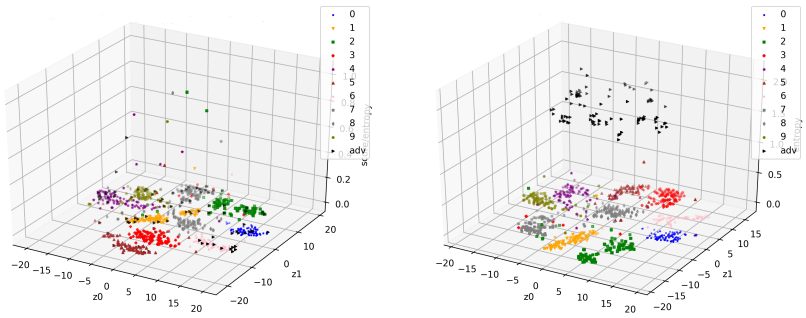

Fig. 4. T-SNE visualization with entropy of prediction with entropy of prediction probability. Black triangles are (negative) adversarial examples while others are clean images. Left: ADR-None. Right: ADR+LC/GB

References

1. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. arXiv preprint arXiv:1802.00420 (2018)
2. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 ieee symposium on security and privacy (sp). pp. 39–57. IEEE (2017)
3. Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., Li, J.: Boosting adversarial attacks with momentum. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 9185–9193 (2018)
4. Jalal, A., Ilyas, A., Daskalakis, C., Dimakis, A.G.: The robust manifold defense: Adversarial training using generative models. arXiv preprint arXiv:1712.09196 (2017)
5. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
6. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
7. Qin, C., Martens, J., Gowal, S., Krishnan, D., Dvijotham, K., Fawzi, A., De, S., Stanforth, R., Kohli, P.: Adversarial robustness through local linearization. In: Advances in Neural Information Processing Systems. pp. 13824–13833 (2019)