Cascade Adversarial Machine Learning Regularized with a Unified Embedding

Taesik Na, Jong Hwan Ko, and Saibal Mukhopadhyay
Georgia Institute of Technology
Atlanta, GA 30332

Abstract

Injecting adversarial inputs during training, known as adversarial training, can improve robustness against one-step attacks, but not for *unknown* iterative attacks. To address this challenge, (1) we first show iteratively generated adversarial images easily transfer between networks trained with the *same strategy*. Inspired by this observation, (2) we propose *cascade adversarial training*, which transfers the knowledge of the end results of adversarial training. We train a network from scratch by injecting iteratively generated adversarial images crafted from *already defended* networks in addition to one-step adversarial images from the network being trained. (3) We also propose to utilize embedding space for both classification and low-level (pixel-level) similarity learning to ignore *unknown* pixel level perturbation. During training, we inject adversarial images *without replacing* their corresponding clean images and penalize the distance between the two embeddings (clean and adversarial). Experimental results show that cascade adversarial training together with our proposed low-level similarity learning efficiently enhances the robustness against iterative attacks under both white box attack and black box attack scenarios.

1 Introduction

Deep neural networks and other machine learning classifiers are shown to be vulnerable to small perturbations to inputs [1, 2, 3, 4, 5]. Previous works [3, 5, 6] have shown that injecting adversarial examples during training (adversarial training) increases the robustness of a network against adversarial attacks. The networks trained with one-step methods have shown noticeable robustness against one-step attacks, but, limited robustness against iterative attacks at test time.

To address this challenge, we first (1) show iteratively generated adversarial images easily transfer between defended networks; (2) propose *cascade adversarial training* which transfers the knowledge of the end results of adversarial training; and (3) propose adversarial training regularized with a unified embedding for both classification and *low (pixel) level similarity learning* to efficiently ignore pixel level perturbation.

2 Proposed Approach

2.1 Transferability Analysis

We describe attack models and methodologies in Appendix A and first show transferability between purely trained networks and defended networks under black box attack. We use ResNet [7] models for CIFAR10 classification. We first train 20-layer ResNets with different methods (standard training, Kurakin's method [5]) and use those as target networks. We re-train networks (standard training and Kurakin's method) with the different initialization from the target networks, and use the trained

Table 1: CIFAR10 test results (%) under black box attacks for ϵ=16. Source networks share the same initialization which is different from the target networks. {Target: R20: standard training, R20$_{\boldsymbol{K}}$: $\boldsymbol{K}$urakin's, Source: R20$_{\mathbf{2}}$: standard training, R20$_{\boldsymbol{K2}}$: $\boldsymbol{K}$urakin's. }

Target	Source: step_FGSM		Source: iter_FGSM	
	R20$_{\mathbf{2}}$	R20$_{\boldsymbol{K2}}$	R20$_{\mathbf{2}}$	R20$_{\boldsymbol{K2}}$
R20	**16.2**	31.6	**2.7**	60.1
R20$_{\boldsymbol{K}}$	**66.7**	82.7	55.8	**28.5**

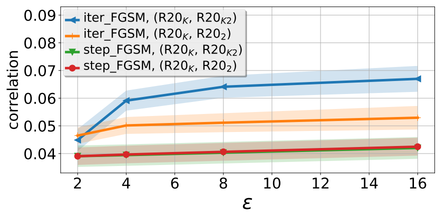


Figure 1: Correlation between adversarial noises from different networks for each ϵ. Shaded region shows $\pm$ 0.1 standard deviation of each line.

networks as source networks. Experimental details and model descriptions can be found in Appendix B and C.

In table 1, we report black box attack accuracies. We observe "iter_FGSM" attack remains very strong even under the black box attack scenario but *only between undefended networks or defended networks*. This is because iter_FGSM noises (X^{adv}-X) from defended networks resemble each other. As shown in figure 1, we observe higher correlation between iter_FGSM noises from a defended network (R20$_{\boldsymbol{K}}$) and those from another defended network (R20$_{\boldsymbol{K2}}$).

2.2 Cascade Adversarial Training

Inspired by the observation that iter_FGSM images transfer well between defended networks, we propose *cascade adversarial training*, which trains a network by injecting iter_FGSM images crafted from an already defended network. We hypothesize that the network trained with cascade adversarial training will learn to avoid such adversarial perturbation, enhancing robustness against iter_FGSM attack. In particular, we train a network by injecting iter_FGSM images crafted from already *defended* network in addition to the one-step adversarial images crafted from the network being trained.

2.3 Regularization with a Unified Embedding

We also advance the algorithm proposed in [5] by adding low level similarity learning. Unlike [5], we include the clean examples used for generating adversarial images in the mini batch. Once one step forward pass is performed with the minibatch, embeddings are followed by the softmax layer for the cross entropy loss for the standard classification. At the same time, we take clean embeddings and adversarial embeddings, and minimize the distance between the two with the distance based loss.

The distance based loss encourages two similar images (clean and adversarial) to produce the same outputs, *not necessarily the true labels*. By adding regularization in higher embedding layer, convolution filters *gradually learn* how to ignore such pixel-level perturbation.

We define the total loss as follows:

$$Loss = \frac{1}{(m-k)+\lambda k}\left(\sum_{i=1}^{m-k} L(\boldsymbol{X}_i|y_i) + \lambda \sum_{i=1}^{k} L(\boldsymbol{X}_i^{adv}|y_i)\right) + \lambda_2 \sum_{i=1}^{k} L_{dist}(\boldsymbol{E}_i^{adv}, \boldsymbol{E}_i)$$

$\boldsymbol{E}_i$ and $\boldsymbol{E}_i^{adv}$ are the resulting embeddings from $\boldsymbol{X}_i$ and $\boldsymbol{X}_i^{adv}$, respectively. m is the size of the mini batch, k ($\leq m/2$) is the number of adversarial images in the mini batch. λ is the parameter to control the relative weight of classification loss for adversarial images. λ_2 is the parameter to control the relative weight of the distance based loss L_{dist} in the total loss.

Pivot loss: we use pivot loss which minimizes the distance between the two embeddings by moving only the adversarial embeddings.

$$L_{dist}(\boldsymbol{E}_i^{adv}|\boldsymbol{E}_i) = ||\boldsymbol{E}_i^{adv} - \boldsymbol{E}_i||_2^2, \quad i = 1, 2, ..., k$$

In particular, we don't back-propagate through the clean embeddings for the distance based loss. The intuition behind the use of pivot loss is that the embedding from a clean image can be treated as the ground truth embedding.

Table 2: CIFAR10 test results (%) for 110-layer ResNet models. CW L_∞ attack is performed with 100 test samples (10 samples per each class) and the number of adversarial examples with $\epsilon > 2$ or 4 is reported. {R110$_{\boldsymbol{K}}$: $\boldsymbol{K}$urakin's, R110$_{\boldsymbol{P}}$: $\boldsymbol{P}$ivot loss, R110$_{\boldsymbol{E}}$: $\boldsymbol{E}$nsemble training, R110$_{\boldsymbol{K},\boldsymbol{C}}$: $\boldsymbol{K}$urakin's and $\boldsymbol{C}$ascade training, R110$_{\boldsymbol{P},\boldsymbol{E}}$: $\boldsymbol{P}$ivot loss and $\boldsymbol{E}$nsemble training, and R110$_{\boldsymbol{P},\boldsymbol{C}}$: $\boldsymbol{P}$ivot loss and $\boldsymbol{C}$ascade training}

Model	clean	step_ll		step_FGSM		iter_FGSM		CW	
		ϵ=2	ϵ=16	ϵ=2	ϵ=16	ϵ=2	ϵ=4	ϵ=2	ϵ=4
R110$_{\boldsymbol{K}}$	92.3	**88.3**	**90.7**	**86.0**	**95.2**	59.4	9.2	25	4
R110$_{\boldsymbol{P}}$ (Ours)	92.3	86.0	89.4	81.6	91.6	64.1	20.9	32	7
R110$_{\boldsymbol{E}}$	92.3	86.3	74.3	84.1	72.9	63.5	21.1	24	6
R110$_{\boldsymbol{K},\boldsymbol{C}}$ (Ours)	92.3	86.2	72.8	82.6	66.7	69.3	33.4	20	5
R110$_{\boldsymbol{P},\boldsymbol{E}}$ (Ours)	91.3	84.0	65.7	77.6	54.5	66.8	38.3	**38**	**16**
R110$_{\boldsymbol{P},\boldsymbol{C}}$ (Ours)	91.5	85.7	76.4	82.4	69.1	**73.5**	**42.5**	27	15

3 Experimental Results

White box attack analysis: We first compare a network trained with Kurakin's method and that with pivot loss. We train 110-layer ResNet models with/without pivot loss and report accuracy in table 2. We observe our low-level similarity learning further improves robustness against iterative attacks compared to Kurakin's adversarial training. However, the accuracy improvements against iterative attacks (iter_FGSM, CW) are limited, showing regularization effect of low-level similarity learning is not sufficient for the iterative attacks on complex color images like CIFAR10. This is different from MNIST test cases where we observed significant accuracy increase for iterative attacks only with pivot loss (Appendix D). We observe label leaking phenomenon reported in [5] happens even though we don't train a network with step_FGSM images. Additional analysis for this phenomenon is explained in Appendix E.3.

Next, we train a network from scratch with iter_FGSM examples crafted from the defended network, R110$_{\boldsymbol{P}}$. We use the same initialization used in R110$_{\boldsymbol{P}}$ since transfer rate is higher within networks sharing the same initialization as explained in [5]. In particular, iter_FGSM images are crafted from R110$_{\boldsymbol{P}}$ with CIFAR10 training images for ϵ= 1,2, ..., 16, and those are used randomly together with step_ll examples from the network being trained. We train cascade networks with/without pivot loss. We also train networks with ensemble adversarial training [8] with/without pivot loss for comparison. The implementation details for the trained models can be found in Appendix C.

We find several meaningful observations in table 2. First, ensemble and cascade models show improved accuracy against iterative attack although at the expense of decreased accuracy for one-step attacks compared to the baseline defended network (R110$_{\boldsymbol{K}}$). Additional data augmentation from other networks enhances the robustness against iterative attack, weakening label leaking effect caused by one-step adversarial training.

Second, our low-level similarity learning (R110$_{\boldsymbol{P},\boldsymbol{E}}$, R110$_{\boldsymbol{P},\boldsymbol{C}}$) further enhances robustness against iterative attacks including fully unknown CW attack (especially for ϵ=4). Additional knowledge learned from data augmentation through cascade/ensemble adversarial training enables networks to learn partial knowledge of perturbations generated by an iterative method. And the *learned iterative perturbations become regularized further with our low-level similarity learning* making networks robust against unknown iterative attacks.

Third, our low-level similarity learning serves as a good regularizer for adversarial images, but not for the clean images for ensemble/cascade models (reduced accuracy for the clean images for R110$_{\boldsymbol{P},\boldsymbol{E}}$ and R110$_{\boldsymbol{P},\boldsymbol{C}}$ in table 2). Compared to pure adversarial training with one-step adversarial examples, the network sees more various adversarial perturbations during training as a result of ensemble and cascade training. Those perturbations are prone to end up embeddings in the vicinity of decision boundary more often than perturbations caused by one-step adversarial training. Pivot loss pulls the vicinity of those adversarial embeddings toward their corresponding clean embeddings. During this process, *clean embeddings from other classes* might also be moved toward the decision boundary which results in decreased accuracy for the clean images.

Table 3: CIFAR10 test results (%) for 110-layer ResNet models under black box attacks (ϵ=16). {Target: same networks in table 2 and R110$_{\boldsymbol{K}}$: $\boldsymbol{K}$urakin's, Source: re-trained baseline, Kurakin's, cascade and ensemble networks with/without pivot loss. Source networks use the different initialization from the target networks. Additional details of the models can be found in Appendix C.}

Target	Source: iter_FGSM						
	R110$_{\boldsymbol{2}}$	R110$_{\boldsymbol{K2}}$	R110$_{\boldsymbol{E2}}$	R110$_{\boldsymbol{P2}}$	R110$_{\boldsymbol{K,C2}}$	R110$_{\boldsymbol{P,E2}}$	R110$_{\boldsymbol{P,C2}}$
R110$_{\boldsymbol{K}}$	70.5	73.2	**27.9**	77.0	67.3	54.6	80.8
R110$_{\boldsymbol{E}}$	77.9	79.5	55.8	79.0	68.2	**54.7**	82.7
R110$_{\boldsymbol{P}}$ (Ours)	75.9	75.6	**39.6**	78.5	68.3	61.3	83.3
R110$_{\boldsymbol{K,C}}$ (Ours)	**56.4**	80.2	61.1	79.5	67.4	62.6	82.1
R110$_{\boldsymbol{P,E}}$ (Ours)	78.2	82.1	**67.7**	81.7	73.4	68.4	83.8
R110$_{\boldsymbol{P,C}}$ (Ours)	71.9	80.4	**63.9**	80.1	71.1	64.2	83.0

Black box attack analysis: We finally perform black box attack analysis for the cascade/ensemble networks with/without pivot loss. We report black box attack accuracy with the source networks trained with the same method, but with different initialization from the target networks since adversarial examples transfer well between networks trained with the same strategy as observed in section 2.1. We re-train 110-layer ResNet models using Kurakin's/cascade/ensemble adversarial training with/without low-level similarity learning and use those networks as source networks for black-box attacks. Baseline 110-layer ResNet model is also included as a source network. Target networks are the same networks used in table 2. We found iter_FGSM attack resulted in lower accuracy than step_FGSM attack, thus, report iter_FGSM attack only in table 3.

We first observe that iter_FGSM attack from ensemble models (R110$_{\boldsymbol{E2}}$, R110$_{\boldsymbol{P,E2}}$) is strong (results in lower accuracy) compared to that from any other trained networks. [1] Since ensemble models learn various perturbation during training, adversarial noises crafted from those networks might be more general for other networks making them transfer easily between defended networks.

Second, transferability between defended networks becomes lower as the network becomes deeper (accuracy is higher when R110$_{\boldsymbol{K2}}$ -> R110$_{\boldsymbol{K}}$ than R20$_{\boldsymbol{K2}}$ -> R20$_{\boldsymbol{K}}$ (table 1)). Since the degree of freedom increases as the network size increases, adversarially trained networks prone to end up with different states, thus, making transfer rate lower.

Third, cascade adversarial training breaks chicken and egg problem (higher transfer rate between defended networks). Even though the transferability between one-step adversarially trained networks is reduced for deeper networks, cascade network (R110$_{\boldsymbol{K,C}}$) shows worst case performance against the attack not from a defended network, but from a purely trained network (R110$_{\boldsymbol{2}}$). Possible solution to further improve the robustness against various source networks would be to use more than one network as source networks (including pure/defended networks) for cascade adversarial training.

Finally, ensemble/cascade networks together with our low-level similarity learning (R110$_{\boldsymbol{P,E}}$, R110$_{\boldsymbol{P,C}}$) show better worst case accuracy under black box attack scenario. This shows that enhancing robustness against iterative white box attack also improves robustness against iterative black box attack.

4 Conclusion

We made an observation that iter_FGSM images transfer easily between networks trained with the *same strategy*. We exploited this and proposed cascade adversarial training, a method to train a network with iter_FGSM adversarial images crafted from already defended networks. We also proposed adversarial training regularized with a unified embedding for classification and low- level similarity learning by penalizing distance between the clean and their corresponding adversarial embeddings. Combining those two techniques (low level similarity learning + cascade adversarial training) improved robustness against iterative attacks for both white-box and black-box attacks.

[1]We also observed this when we switch the source and the target networks. Additional details can be found in Appendix G.

Acknowledgments

We thank Alexey Kurakin of Google Brain and Li Chen of Intel for comments that greatly improved the manuscript. The research reported here was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-17-2-0045. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA.

References

[1] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *ECML/PKDD (3)*, volume 8190 of *Lecture Notes in Computer Science*, pages 387–402. Springer, 2013.

[2] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[3] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[4] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. In *Proceeedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS'17)*, 2017.

[5] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[6] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *CoRR*, abs/1511.03034, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[8] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *CoRR*, abs/1705.07204, 2017.

[9] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Proceedings of the International Conference on Learning Representations (ICLR) Workshop*, 2017.

[10] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

A Background on Adversarial Attacks

One-step fast gradient sign method (FGSM), referred to as "step_FGSM", tries to generate adversarial images to minimize confidence on true label ([3]). Adversarial image $\boldsymbol{X}^{adv}$ is generated by adding sign of the gradients w.r.t. the clean image $\boldsymbol{X}$ multiplied by $\epsilon \in [0, 255]$ as shown below:

$$\boldsymbol{X}^{adv} = \boldsymbol{X} + \epsilon \ \text{sign}(\nabla_X J(\boldsymbol{X}, y_{true}))$$

One-step target class method tries to generate adversarial images to maximize the confidence on a target false label as follows:

$$\boldsymbol{X}^{adv} = \boldsymbol{X} - \epsilon \ \text{sign}(\nabla_X J(\boldsymbol{X}, y_{target}))$$

We use least likely class y_{LL} as a target class and refer this method as "step_ll" as in [9, 5].

Basic iterative method, referred to as "iter_FGSM", applies FGSM with small α multiple times.

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_N^{adv} = Clip_{X,\epsilon}\big\{\boldsymbol{X}_{N-1}^{adv} + \alpha \ \text{sign}(\nabla_{X_{N-1}^{adv}} J(\boldsymbol{X}_{N-1}^{adv}, y_{true}))\big\}$$

We use α = 1, number of iterations N to be $\min(\epsilon + 4, 1.25\epsilon)$. $Clip_{X,\epsilon}$ is elementwise clipping function where the input is clipped to the range $[\max(0, X - \epsilon), \min(255, X + \epsilon)]$.

Iterative least-likely class method, referred to as "iter_ll", is to apply "step_ll" with small α multiple times.

$$\boldsymbol{X}_0^{adv} = \boldsymbol{X}, \quad \boldsymbol{X}_N^{adv} = Clip_{X,\epsilon}\big\{\boldsymbol{X}_{N-1}^{adv} - \alpha \ \text{sign}(\nabla_{X_{N-1}^{adv}} J(\boldsymbol{X}_{N-1}^{adv}, y_{LL}))\big\}$$

Carlini and Wagner attack [10] referred to as "CW" solves an optimization problem which minimizes both an objective function f (such that attack is success if and only if $f(\boldsymbol{X}^{adv}) < 0$) and a distance measure between $\boldsymbol{X}^{adv}$ and $\boldsymbol{X}$.

Black box attack is performed by testing accuracy on a target network with the adversarial images crafted from a source network different from the target network. Lower accuracy means successful black-box attack. When we use the same network for both target and source network, we call this as white-box attack.

Adaptive attack is another form of black box attack where adversaries can collect input and output pairs for a target network by querying arbitrary inputs. This can be considered as stronger/weaker attack than pure black box/white box attack model. In this paper, we don't consider this since it will be covered by considering both white-box and black box attacks.

B Experimental Setup

We scale down the image values to [0,1] and don't perform any data augmentation for MNIST. For CIFAR10, we scale down the image values to [0,1] and subtract per-pixel mean values. We perform 24x24 random crop and random flip on 32x32 original images. We generate adversarial images with "step_ll" after these steps otherwise noted.

We use stochastic gradient descent (SGD) optimizer with momentum of 0.9, weight decay of 0.0001 and minibatch size of 128. For adversarial training, we generate k = 64 adversarial examples among 128 images in one mini-batch. We start with a learning rate of 0.1, divide it by 10 at 4k and 6k iterations, and terminate training at 8k iterations for MNIST, and 48k and 72k iterations, and terminate training at 94k iterations for CIFAR10. [2] We also found that initialization affects the training results slightly as in [5], thus, we pre-train the networks 2 and 10 epochs for MNIST and CIFAR10, and use these as initial starting points for different configurations.

We use λ = 0.3, λ_2 = 0.0001, m = 128, k = 64 for the experiments. As in [5], we used randomly chosen ϵ in the interval [0, max_e] with clipped normal distribution $\mathcal{N}(\mu = 0, \sigma = max\_e/2)$, where max_e is the maximum ϵ used in training. We use max_e = 0.3*255 and 16 for MNIST and CIFAR10 respectively.

[2] We found that the adversarial training requires longer training time than the standard training. Authors in the original paper [7] changed the learning rate at 32k and 48k iterations and terminated training at 64k iterations.

When we train networks with distance based loss, we also consider another type of loss (**bidirectional loss**) which minimizes the distance between the two embeddings by moving both clean and adversarial embeddings.

$$L_{dist}(\boldsymbol{E}_i^{adv}, \boldsymbol{E}_i) = ||\boldsymbol{E}_i^{adv} - \boldsymbol{E}_i||_2^2, \quad i = 1, 2, ..., k$$

C Model Descriptions

We summarize the model names used in this paper in table 4. For ensemble adversarial training, pre-trained networks as in table 6 together with the network being trained are used to generate one-step adversarial examples during training. For cascade adversarial training, pre-trained defended networks as in table 5 are used to generate iter_FGSM images, and the network being trained is used to generate one-step adversarial examples during training.

Table 4: Model descriptions

Dataset	ResNet	Initialization Group	Training	Model
MNIST	20-layer	A	standard training	R20M
			$\boldsymbol{K}$urakin's	R20M$_{\boldsymbol{K}}$
			$\boldsymbol{B}$idirection loss	R20M$_{\boldsymbol{B}}$
			$\boldsymbol{P}$ivot loss	R20M$_{\boldsymbol{P}}$
CIFAR10	20-layer	B	standard training	R20
			$\boldsymbol{K}$urakin's	R20$_{\boldsymbol{K}}$
			$\boldsymbol{E}$nsemble training	R20$_{\boldsymbol{E}}$
			$\boldsymbol{B}$idirection loss	R20$_{\boldsymbol{B}}$
			$\boldsymbol{P}$ivot loss	R20$_{\boldsymbol{P}}$
			$\boldsymbol{K}$urakin's & $\boldsymbol{C}$ascade training	R20$_{\boldsymbol{K},\boldsymbol{C}}$
			$\boldsymbol{P}$ivot loss & $\boldsymbol{E}$nsemble training	R20$_{\boldsymbol{P},\boldsymbol{E}}$
			$\boldsymbol{P}$ivot loss & $\boldsymbol{C}$ascade training	R20$_{\boldsymbol{P},\boldsymbol{C}}$
		C	standard training	R20$_{\boldsymbol{2}}$
			$\boldsymbol{K}$urakin's	R20$_{\boldsymbol{K2}}$
			$\boldsymbol{P}$ivot loss	R20$_{\boldsymbol{P2}}$
		D	standard training	R20$_{\boldsymbol{3}}$
		E	standard training	R20$_{\boldsymbol{4}}$
	56-layer	F	$\boldsymbol{K}$urakin's	R56$_{\boldsymbol{K}}$
			$\boldsymbol{P}$ivot loss	R56$_{\boldsymbol{P}}$
		G	$\boldsymbol{K}$urakin's	R56$_{\boldsymbol{K2}}$
	110-layer	H	standard training	R110
			$\boldsymbol{K}$urakin's	R110$_{\boldsymbol{K}}$
			$\boldsymbol{P}$ivot loss	R110$_{\boldsymbol{P}}$
			$\boldsymbol{E}$nsemble training	R110$_{\boldsymbol{E}}$
			$\boldsymbol{K}$urakin's & $\boldsymbol{C}$ascade training	R110$_{\boldsymbol{K},\boldsymbol{C}}$
			$\boldsymbol{P}$ivot loss & $\boldsymbol{E}$nsemble training	R110$_{\boldsymbol{P},\boldsymbol{E}}$
			$\boldsymbol{P}$ivot loss & $\boldsymbol{C}$ascade training	R110$_{\boldsymbol{P},\boldsymbol{C}}$
		I	standard training	R110$_{\boldsymbol{2}}$
			$\boldsymbol{K}$urakin's	R110$_{\boldsymbol{K2}}$
			$\boldsymbol{E}$nsemble training	R110$_{\boldsymbol{E2}}$
			$\boldsymbol{P}$ivot loss	R110$_{\boldsymbol{P2}}$
			$\boldsymbol{K}$urakin's & $\boldsymbol{C}$ascade training	R110$_{\boldsymbol{K},\boldsymbol{C2}}$
			$\boldsymbol{P}$ivot loss & $\boldsymbol{E}$nsemble training	R110$_{\boldsymbol{P},\boldsymbol{E2}}$
			$\boldsymbol{P}$ivot loss & $\boldsymbol{C}$ascade training	R110$_{\boldsymbol{P},\boldsymbol{C2}}$
		J	standard training	R110$_{\boldsymbol{3}}$
		K	standard training	R110$_{\boldsymbol{4}}$

Table 5: Ensemble model description

Ensemble models	Pre-trained models
R20$_{\boldsymbol{E}}$, R20$_{\boldsymbol{P,E}}$, R110$_{\boldsymbol{E}}$, R110$_{\boldsymbol{P,E}}$	R20$_{\boldsymbol{3}}$, R110$_{\boldsymbol{3}}$
R110$_{\boldsymbol{E2}}$, R110$_{\boldsymbol{P,E2}}$	R20$_{\boldsymbol{4}}$, R110$_{\boldsymbol{4}}$

Table 6: Cascade model description

Cascade models	Pre-trained model
R20$_{\boldsymbol{K,C}}$, R20$_{\boldsymbol{P,C}}$	R20$_{\boldsymbol{P}}$
R110$_{\boldsymbol{K,C}}$, R110$_{\boldsymbol{P,C}}$	R110$_{\boldsymbol{P}}$
R110$_{\boldsymbol{K,C2}}$, R110$_{\boldsymbol{P,C2}}$	R110$_{\boldsymbol{P2}}$

D Experimental Results on MNIST

We use ResNet [7] for MNIST classification. We train networks with different methods (standard training, Kurakin's adversarial training and adversarial training with our distance based loss). Experimental details can be found in Appendix B.

Table 7 shows the accuracy results for MNIST test dataset for different types of attack methods. As shown in the table, our method achieves better accuracy than Kurakin's method for all types of attacks with a little sacrifice on the accuracy for the clean images. Even though adversarial training is done only with "step_ll", additional regularization increases robustness against *unknown* "step_FGSM", "iter_ll", "iter_FGSM" and CW L_∞ attacks. This shows that our low-level similarity learning can successfully regularize the one-step adversarial perturbation and its vicinity for simple image classification like MNIST.

Table 7: MNIST test results (%) for 20-layer ResNet models (ϵ = 0.3*255 at test time). { R20M: standard training, R20M$_{\boldsymbol{K}}$: $\boldsymbol{K}$urakin's adversarial training, R20M$_{\boldsymbol{B}}$: $\boldsymbol{B}$idirectional loss, R20M$_{\boldsymbol{P}}$: $\boldsymbol{P}$ivot loss.} CW L_∞ attack is performed with 100 test samples (10 samples per each class) and the number of adversarial examples with $\epsilon > 0.3*255$ is reported. Additional details for CW attack can be found in Appendix H

Model	clean	step_ll	step_FGSM	iter_ll	iter_FGSM	CW
R20M	99.6	9.7	10.3	0.0	0.0	0
R20M$_{\boldsymbol{K}}$	99.6	96.7	94.5	89.0	60.2	46
R20M$_{\boldsymbol{B}}$ (Ours)	99.5	**97.3**	**96.2**	**97.2**	**88.5**	**81**
R20M$_{\boldsymbol{P}}$ (Ours)	99.5	**97.1**	**95.7**	**96.9**	**88.9**	**82**

D.1 Embedding Space Visualization

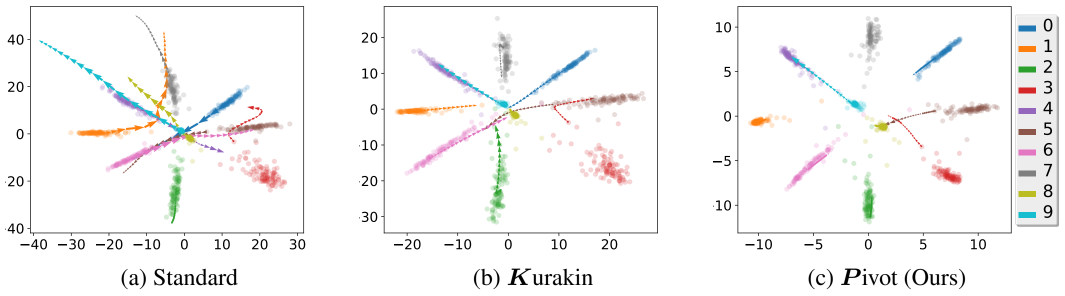


Figure 2: Embedding space visualization for modified ResNet models trained on MNIST. x-axis and y-axis show first and second dimension of embeddings respectively. Scatter plot shows first 100 clean embeddings per each class on MNIST test set. Each arrow shows difference between two embeddings (one from iter_FGSM images with ϵ and the other one from iter_FGSM images with ϵ+4). We draw arrows from ϵ = 0 to ϵ = 76 ($\approx$ 0.3*255) for one sample image per each class. We observe differences between clean and corresponding adversarial embeddings are minimized for the network trained with pivot loss.

To visualize the embedding space, we modify 20-layer ResNet model where the last fully connected layer (64x10) is changed to two fully connected layers (64x2 and 2x10). We re-train networks with standard training, Kurakin's method and our pivot loss on MNIST. [3] In figure 2, we draw embeddings (dimension=2) between two fully connected layers. As seen from this figure, adversarial images from the network trained with standard training cross the decision boundary easily as ϵ increases. With Kurakin's adversarial training, the distance between clean and adversarial embeddings are minimized

[3]Modified ResNet models showed slight decreased accuracy for both clean and adversarial images compared to original ResNet counterparts, however, we observed similar trends (improved accuracy for iterative attacks for the network trained with pivot loss) as in table 7.

compared to standard training. And our pivot loss further minimizes distance between the clean and adversarial embeddings. Note that our pivot loss also decreases absolute value of the embeddings, thus, higher λ_2 will eventually result in overlap between distinct embedding distributions. We also observe that intra class variation of the clean embeddings are also minimized for the network trained with our pivot loss as shown in the scatter plot in figure 2 (c).

E Analysis of Adversarial Training

In this section, we apply our low-level similarity learning on CIFAR10 dataset and provide in-depth analysis of adversarial training. We also use ResNet models and train those with different methods. Additional details for the experiments can be found in Appendix B.

E.1 Experimental Results on CIFAR10

Table 8 shows the accuracy results for 20-layer ResNet models on CIFAR10 test dataset. Again, we also observe our low-level similarity learning further improves robustness against all types of attacks compared to Kurakin's adversarial training. However, the accuracy improvements against iterative attacks (iter_FGSM, CW) are limited, showing regularization effect of low-level similarity learning is not sufficient for the iterative attacks on complex color images like CIFAR10.

Table 8: CIFAR10 test results (%) for 20-layer ResNet models. { R20: standard training, R20$_{\boldsymbol{K}}$: $\boldsymbol{K}$urakin's adversarial training, R20$_{\boldsymbol{B}}$: $\boldsymbol{B}$idirectional loss, R20$_{\boldsymbol{P}}$: $\boldsymbol{P}$ivot loss.} CW L_∞ attack is performed with 100 test samples (10 samples per each class) and the number of adversarial examples with $\epsilon > 2$ or 4 is reported.

Model	clean	step_ll		step_FGSM		iter_FGSM		CW	
		ϵ=2	ϵ=16	ϵ=2	ϵ=16	ϵ=2	ϵ=4	ϵ=2	ϵ=4
R20	90.9	44.5	11.5	28.7	12.2	14.0	0.4	8	0
R20$_{\boldsymbol{K}}$	91.0	85.8	84.4	78.9	81.5	50.6	9.8	13	2
R20$_{\boldsymbol{B}}$ (Ours)	91.0	86.1	**87.3**	78.7	**90.0**	**52.0**	**11.2**	**19**	**3**
R20$_{\boldsymbol{P}}$ (Ours)	90.9	86.2	**88.3**	79.7	**91.7**	**52.0**	**11.3**	**18**	**4**

E.2 Effect of λ_2

We train networks with pivot loss and various λ_2s for CIFAR10 dataset to study effects of the weight of the distance measure in the loss function. Figure 3 shows that a higher λ_2 increases accuracy of the iteratively generated adversarial images. However, it reduces accuracy on the clean images, and increasing λ_2 above 0.3 even results in divergence of the training. This is because embedding distributions of different classes will eventually overlap since absolute value of the embedding will be decreased as λ_2 increases. In this experiment, we show there exists clear trade-off between accuracy for the clean images and that for the adversarial images, and we recommend using a very high λ_2 only under strong adversarial environment.

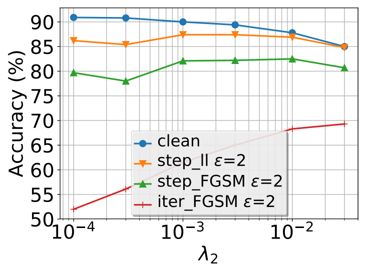
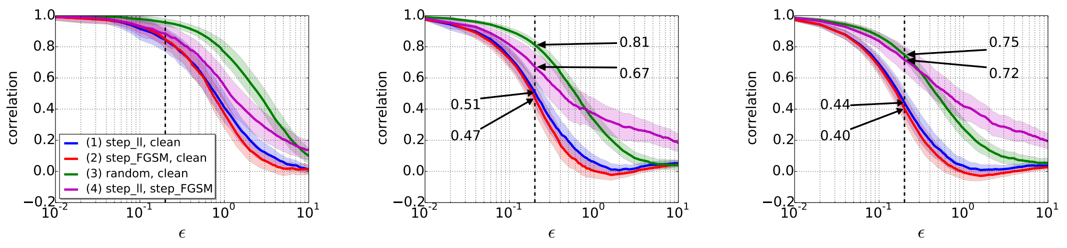


Figure 3: Accuracy vs. λ_2

E.3 Label Leaking Analysis

We observe accuracies for the “step_FGSM" adversarial images become higher than those for the clean images (“label leaking" phenomenon) by training with “step_FGSM" examples as in [5]. Interestingly, we also observe “label leaking" phenomenon even without providing true labels for adversarial images generation as shown in “R20$_P$" in table 8. We argue that “label leaking” is a natural result of the adversarial training.

To understand the nature of adversarial training, we measure correlation between gradients w.r.t. different images (i.e. clean vs. adversarial) as a measure of error surface similarity. We measure (1)



(a) R20: Baseline (b) R20$_{\boldsymbol{K}}$: Kurakin's (c) R20$_{\boldsymbol{P}}$: Pivot (Ours)

Figure 4: Averaged Pearson's correlation coefficient between the gradients w.r.t. two images. Trained networks are the same in table 8. Correlation were measured by changing ϵ for each adversarial image and averaged over randomly chosen 128 images from CIFAR10 test-set. Shaded region represents $\pm$ 0.5 standard deviation of each line.

clean vs. "step_ll" image, (2) clean vs. "step_FGSM" image, (3) clean vs. "random sign" added image, and (4) "step_ll" image vs. "step_FGSM" image for three trained networks (a) R20, (b) R20$_{\boldsymbol{K}}$ and (c) R20$_{\boldsymbol{P}}$ (Ours) in table 8. Figure 4 draws average value of correlations between randomly chosen clean images and adversarial images from CIFAR10 test-set for each case.

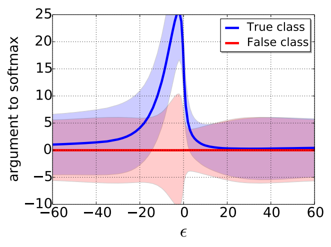
Meaning of the correlation: In order to make strong adversarial images with "step_FGSM" method, correlation between the gradient w.r.t. the clean image and the gradient w.r.t. its corresponding adversarial image should remain high since the "step_FGSM" method *only* use the gradient w.r.t. the clean image (ϵ=0). Lower correlation means perturbing the adversarial image at ϵ further to the gradient (seen from the clean image) direction is no longer efficient.

Results of adversarial training: We observe that (1) and (2) become quickly lower than (3) as ϵ increases. This means that, when we move toward the steepest (gradient) direction on the error surface, gradient is more quickly uncorrelated with the gradient w.r.t. the clean image than when we move to random direction. As a result of adversarial training, this uncorrelation is observed at a lower ϵ making one-step attack less efficient even with small perturbation. (1), (2) and (3) for our case are slightly lower than Kurakin's method at the same ϵ which means that our method is better at defending one-step attacks than Kurakin's.

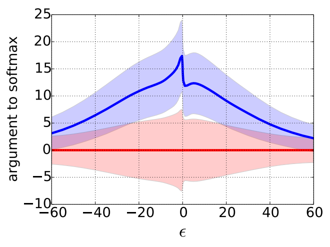
Error surface similarity between "step_ll" and "step_FGSM" images: We also observe (4) remains high with higher ϵ for all trained networks. This means that the error surface (gradient) of the "step_ll" image and that of its corresponding "step_FGSM" image resemble each other. That is the reason why we get the robustness against "step_FGSM" method only by training with "step_ll" method and vice versa. (4) for our case is slightly higher than Kurakin's method at the same ϵ and that means our similarity learning tends to make error surfaces of the adversarial images with "step_ll" and "step_FGSM" method to be more similar.

Analysis of label leaking phenomenon: Interestingly, (2) becomes slightly negative in certain range ($1 < \epsilon < 3$ for Kurakin's, and $1 < \epsilon < 4$ for Pivot (Ours)) and this could be the possible reason for "label leaking" phenomenon. For example, let's assume that we have a perturbed image (by "step_FGSM" method) at ϵ where the correlation between the gradients w.r.t. that image and the corresponding clean image is negative. Further increase of ϵ with the gradient (w.r.t. the clean image) direction actually decreases the loss resulting in increased accuracy (label leaking phenomenon). The reason why label leaking phenomenon is prone to happen for the networks trained with step_FGSM images is that the networks can be heavily optimized for the step_FGSM images. Due to the error surface similarity between "step_ll" and "step_FGSM" images and this negative correlation effect, however, label leaking phenomenon can always happen for the networks trained with *one-step* adversarial examples.

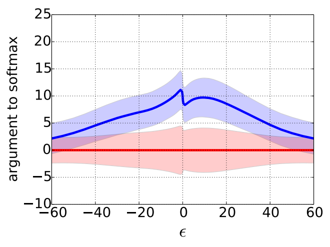
F Alternative Visualization on Embeddings



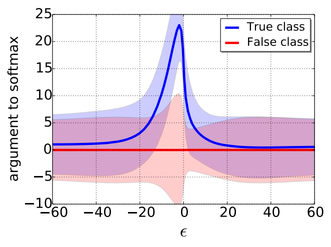
(a) R20, step_ll



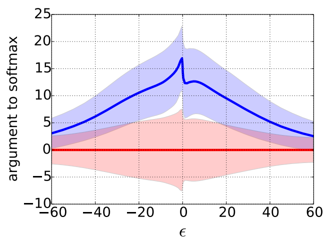
(b) R20$_{\boldsymbol{K}}$, step_ll



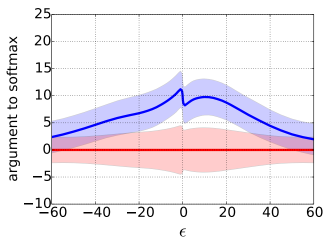
(c) R20$_{\boldsymbol{P}}$ (Ours), step_ll



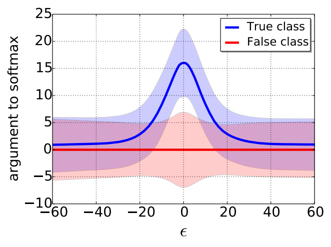
(d) R20, step_FGSM



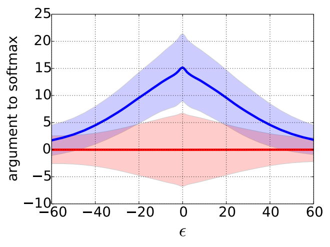
(e) R20$_{\boldsymbol{K}}$, step_FGSM



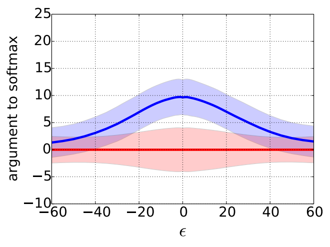
(f) R20$_{\boldsymbol{P}}$ (Ours), step_FGSM



(g) R20, random sign



(h) R20$_{\boldsymbol{K}}$, random sign



(i) R20$_{\boldsymbol{P}}$ (Ours), random sign

Figure 5: Argument to the softmax vs. ϵ in test time. For the trained networks in table 8, “step_ll", “step_FGSM" and "random sign" methods were used to generate test-time adversarial images. Arguments to the softmax were measured by changing ϵ for each test method and averaged over randomly chosen 128 images from CIFAR10 test-set. Blue line represents true class and the red line represents mean of the false classes. Shaded region shows $\pm$ 1 standard deviation of each line.

As shown in figure 5, we observe that embeddings for adversarial images with “step_ll" and those with “step_FGSM" resemble each other. As a result of adversarial training, even though we don’t inject random sign added images, we observe improved results (broader margins than baseline) for ’random sign’ added images. Overall shape of the argument to the softmax layer in our case becomes smoother than Kurakin’s method suggesting our method is good for pixel level regularization.

G Additional Black Box Attack Results

Table 9: CIFAR10 test results (%) under black box attacks for ϵ=16. {Target and Source: same networks in table 8}

Target	Source: step_FGSM			Source: iter_FGSM		
	R20	R20$_{K}$	R20$_{P}$	R20	R20$_{K}$	R20$_{P}$
R20	**12.2**	27.4	27.5	0.0	45.9	44.7
R20$_{K}$	**65.7**	81.5	81.8	51.5	0.0	**18.2**
R20$_{P}$	**58.1**	89.3	91.7	48.9	**13.4**	0.0

Table 9 shows that black box attack between trained networks with the same initialization tends to be more successful than that between networks with different initialization as explained in [5].

Table 10: CIFAR10 test results (%) under black box attacks for ϵ=16. {Target and Source networks are switched from the table 1}

Target	Source: step_FGSM			Source: iter_FGSM		
	R20	R20$_{K}$	R20$_{P}$	R20	R20$_{K}$	R20$_{P}$
R20$_{2}$	**17.9**	33.9	34.5	**4.1**	54.8	54.3
R20$_{K2}$	**65.0**	84.6	84.5	61.2	**25.3**	**30.4**
R20$_{P2}$	**66.4**	88.2	87.2	61.6	**27.7**	**36.1**

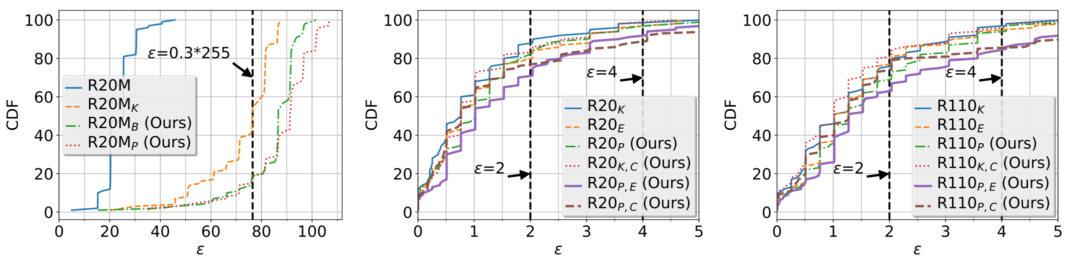
In table 10, our method (R20$_{P2}$) is always better at one-step and iterative black box attack from defended networks (R20$_{K}$, R20$_{P}$) and undefended network (R20) than Kurakin's method (R20$_{B2}$). However, it is hard to tell which method is better than the other one as explained in the main paper.

Table 11: CIFAR10 test results (%) for cascade networks under black box attacks for ϵ=16. {Target and Source: Please see the model descriptions in Appendix C.}

Target	Source: iter_FGSM						
	R110	R110$_{K}$	R110$_{E}$	R110$_{P}$	R110$_{K,C}$	R110$_{P,E}$	R110$_{P,C}$
R110$_{K2}$	80.5	72.7	49.3	68.0	49.6	**41.0**	67.9
R110$_{E2}$	82.7	59.1	**39.5**	59.6	51.5	40.3	69.6
R110$_{P2}$ (Ours)	80.3	75.9	54.2	72.2	54.9	**44.3**	72.4
R110$_{K,C2}$ (Ours)	62.1	74.7	61.5	72.3	46.5	**39.0**	67.9
R110$_{P,E2}$ (Ours)	81.5	79.0	50.0	77.3	56.9	**45.5**	75.2
R110$_{P,C2}$ (Ours)	72.2	76.4	60.6	73.8	51.0	**40.9**	72.0

In table 11, we show black box attack accuracies with the source and the target networks switched from the table 3. We also observe that networks trained with both low-level similarity learning and cascade/ensemble adversarial training (R110$_{P,C2}$, R110$_{P,E2}$) show better *worst-case* performance than other networks. Overall, iter_FGSM images crafted from ensemble model families (R110$_{E}$, R110$_{P,E}$) remain strong on the defended networks.

H Implementation Details for Carlini-Wagner L_∞ Attack



(a) MNIST (b) CIFAR10, 20-layer (c) CIFAR10, 110-layer

Figure 6: Cumulative distribution function vs. ϵ for 100 test adversarial examples generated by CW L_∞ attack. Lower CDF value for a fixed ϵ means the better defense.

Carlini and Wagner (CW) L_∞ attack solves the following optimization problem for every input $\boldsymbol{X}$.

$$minimize \quad c \cdot f(\boldsymbol{X} + \boldsymbol{\delta}) + \sum_i [(|\delta_i| - \tau)^+]$$

$$such\ that \quad \boldsymbol{X} + \boldsymbol{\delta} \in [0, 1]^n$$

where, the function f is defined such that attack is success if and only if $f(\boldsymbol{X} + \boldsymbol{\delta}) < 0$, $\boldsymbol{\delta}$ is the target perturbation defined as $\boldsymbol{X}^{adv} - \boldsymbol{X}$, c is the parameter to control the relative weight of function f in the total cost function, and τ is the control threshold used to penalize any terms that exceed τ.

Since CW L_∞ attack is computationally expensive, we only use 100 test examples (10 examples per each class). We search adversarial example $\boldsymbol{X}^{adv}$ with $c \in \{0.1, 0.2, 0.5, 1, 2, 5, 10, 20\}$ and $\tau \in \{0.02, 0.04, ..., 0.6\}$ for MNIST and $c \in \{0.1, 0.3, 1, 3, 10, 30, 100\}$ and $\tau \in \{0.001, 0.002, ..., 0.01, 0.012, ..., 0.02, 0.024, ..., 0.04, 0.048, ..., 0.08\}$ for CIFAR10. We use Adam optimizer with an initial learning rate of $0.01/c$ since we found constant initial learning rate for $c \cdot f(\boldsymbol{X} + \boldsymbol{\delta})$ term is critical for successful adversarial images generation. We terminate the search after 2,000 iterations for each $\boldsymbol{X}$, c and τ. If $f(\boldsymbol{X} + \boldsymbol{\delta}) < 0$ and the resulting $||\boldsymbol{\delta}||_\infty$ is lower than the current best distance, we update $\boldsymbol{X}^{adv}$.

Figure 6 shows cumulative distribution function of ϵ for 100 successful adversarial examples per each network. We report the number of adversarial examples with $\epsilon > 0.3$*255 for MNIST and that with $\epsilon > 2$ or 4 for CIFAR10. As seen from this figure, our approaches provide robust defense against CW L_∞ attack compared to other approaches.