Cascade Adversarial Machine Learning Regularized with a Unified Embedding

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

Abstract

Deep neural network classifiers are vulnerable to small input perturbations carefully generated by the adversaries. 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, we 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). This additional regularization encourages two similar images (clean and perturbed versions) to produce the same outputs, *not necessarily the true labels*, enhancing classifier's robustness against pixel level perturbation. Next, we show iteratively generated adversarial images easily transfer between networks trained with the *same strategy*. Inspired by this observation, we also 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. Experimental results show that cascade adversarial training together with our proposed low-level similarity learning efficiently enhance the robustness against iterative attacks, but at the expense of decreased robustness against one-step attacks. We show that combining those two techniques can also improve robustness under the *worst case* black box attack scenario.

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 adversarial examples can be generated by perturbing the inputs in one step or iteratively to either minimize confidence on true labels or increase confidence of a target false label (section 2). The networks trained with one-step method have shown noticeable robustness against one-step attacks, but, limited robustness against iterative attacks at test time. In [5], authors explained that using iterative methods for training didn't help improve robustness against iterative attacks at test time.

To address this challenge, we propose adversarial training regularized with a unified embedding for both classification and low-level (pixel-level) similarity learning to efficiently ignore pixel level perturbation. We inject adversarial images in the mini batch *without replacing* their corresponding clean images and penalize distance between embeddings from the clean and the adversarial examples. Thus, our approach differs from prior works where adversarial images were used *only* for data augmentation by *replacing* their corresponding clean images.

There are past examples of using the embedding space for learning similarity of high level features like face similarity between two different images [7, 8, 9]. Instead, we use the embedding space for learning similarity of the *pixel level differences between two similar images*. As the additional regularization promotes two similar images to produce similar embeddings, the network can be regularized for the additive pixel level perturbation (both known and unknown perturbations).

We train ResNet model [10] on MNIST [11] and CIFAR10 dataset [12] using the proposed adversarial training. The experimental results demonstrate improved robustness of the network against adversarial images generated by *one-step* and *iterative* methods compared to the prior work.

Our approach is also less sensitive to the randomness of the training (random initialization and shuffling of the training data) and shows less variation in the accuracies for adversarial examples at test time. We also show that modifying the weight of the distance measure in the loss function can help control trade-off between accuracies for clean and adversarial examples.

Next, we provide fruitful insights of adversarial training by analyzing correlation between gradients w.r.t. the clean images and gradients w.r.t. their corresponding adversarial images as a measure of error surface similarity. We show negative correlation between the gradient w.r.t. the clean image and the gradient w.r.t. its corresponding adversarial image is the reason for the label leaking phenomenon, reported in the previous work [5]. We argue that label leaking phenomenon can always happen as long as we train networks by injecting *one-step* adversarial images due to the negative correlation effect and the similarity between gradients w.r.t. one-step adversarial images.

We also analyze transferability between defended networks, and between pure and defended networks by testing accuracies for the target network with the adversarial examples crafted from different networks (black box attack). We show one-step adversarial images from defended networks become weaker than those from undefended networks since adversarial training makes gradient seen from the clean image not to point strong adversarial image by distorting the error surface. We also make interesting observation that iter_FGSM images (section 2) transfer more between networks when the source and the target networks are trained with the *same training method*.

Inspired by this observation, we propose *cascade adversarial training* which transfers the knowledge of the end results of adversarial training. [1] 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. Together with cascade adversarial training and low-level similarity learning, we achieve significant accuracy increase against unknown iterative attacks, but at the expense of decreased accuracy for one-step attacks. Note that iterative attack cannot be considered as known attack since we didn't use the current state of the network during training for adversarial example generation, instead we use saved version from already defended network.

Finally, we show our *cascade adversarial training* gives much better robustness against black box attack scenario. Only by combining low level similarity learning and cascade adversarial training, networks can become even more robust against both white box and black box attacks.

2 Background on Attack Methods

We consider *one-step* and *iterative* methods for adversarial example generation as described in [5] and follow the same terminologies. We use L_∞ metric for adversarial images generation where the maximum allowed perturbation is defined as ϵ. Other metrics like L_0 and L_2 exist, but, there is no standard for the use of those metrics. Please see the details in [14].

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:

$$\boldsymbol{X}^{adv} = \boldsymbol{X} + \epsilon \operatorname{sign}(\nabla_X J(\boldsymbol{X}, y_{true})) \tag{1}$$

---

[1] After we made the initial version of this paper before [13], we found similar concept with the cascade adversarial training is proposed as ensemble adversarial training in [13]. The concept of using already defended networks for adversarial training is very similar, however, the details are different. We initially started from an observation of high transferability of iter_FGSM images between defended networks, thus, we proposed to use defended networks for iter_FGSM images generation instead of purely using defended networks as other source networks for one-step adversarial examples generation.
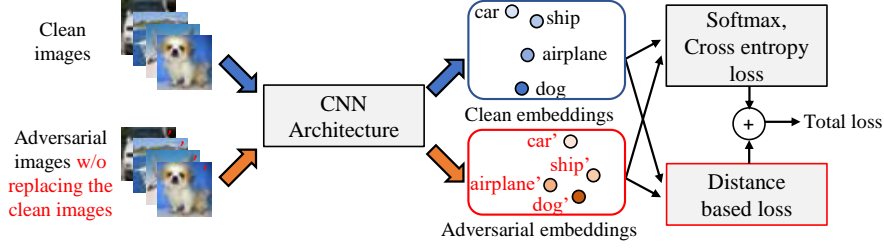
Figure 1: Adversarial training framework regularized with a unified embedding. Adversarial images are used without replacing the clean images. Distance based loss is added to ensure the distance between the two embeddings per each image minimized.

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

$$\boldsymbol{X}^{adv} = \boldsymbol{X} - \epsilon \, \text{sign}(\nabla_X J(\boldsymbol{X}, y_{target})) \tag{2}$$

We use least likely class y_{LL} as a target class and refer this method as "step_ll" as in [15, 5].

Basic iterative method, referred to as "iter_FGSM", applies FGSM with small α multiple times. [2]

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

We use $\alpha = 1$, number of iterations to be $\min(\epsilon + 4, 1.25\epsilon)$. $Clip_{X,\epsilon}$ is elementwise clipping function where the input is clipped to the range $[\max(0, X - \epsilon), \min(255, X + \epsilon)]$. [3]

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

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

Other iterative methods L_∞ attack proposed by Carlini and Wagner [14] is also known to be strong attack, however, it shows comparable results with iter_FGSM attack as shown in [14]. In this paper, we only consider Kurakin's iterative attacks as examples of unknown strong attack.

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.

3   Low Level Similarity Learning

3.1   Regularization with a Unified Embedding

Basic idea of adversarial training proposed in [5] is to inject adversarial examples during training. In this method, k examples are taken from the minibatch B (size of m) and the adversarial examples are generated with one of the attack methods discussed in section 2. The k adversarial examples *replaces* the corresponding clean examples. Below we refer this adversarial training method as "Kurakin's"

We 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 as shown in figure 1. Once one step forward pass is performed with the minibatch, embeddings are followed by

---

[2]In [5], it is referred as "iter. basic". We use "iter_FGSM" term since it is more intuitive.

[3]When the image is scaled to [0, 1] and mean value subtracted, we use mean value added image with scaled to [0,255] for the input of the clipping function.
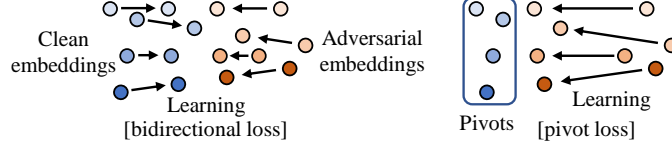
Figure 2: (**Left**) Bidirectional loss. (**Right**) Pivot loss.

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*. Thus, low-level similarity learning can be considered as an unsupervised learning. By adding regularization in higher embedding layer, convolution filters *gradually learn* how to ignore such pixel-level perturbation. We have applied regularization on lower layers with an assumption that low level pixel perturbation can be ignored in lower hierarchy of networks. However, adding regularization term on higher embedding layer right before the softmax layer showed best performance. *The more convolutional filters have chance to learn such similarity, the better the performance.* Note that cross entropy doesn't encourage two similar images to produce the same output labels. Standard image classification using cross entropy compares ground truth labels with outputs of a network *regardless of* how similar training images are. The entire procedure is described in algorithm 1.

Algorithm 1 Adversarial training with distance based loss
m: size of mini batch, k: size of adversarial images

Require: $k \leq m/2$
1: **repeat**
2:     Get mini batch $B = \{X_1, ..., X_{m-k}\}$ from training data set.
3:     Generate k adversarial examples $\{X_1^{adv}, ..., X_k^{adv}\}$ from corresponding clean examples $\{X_1, ..., X_k\}$ using current state of the network N.
4:     Make new mini batch $B' = \{X_1^{adv}, ..., X_k^{adv}, X_1, ..., X_{m-k}\}$.
5:     Perform one step forward pass with B'.
6:     Formulate the cross entropy loss with entire embeddings $\{E_1^{adv}, ..., E_k^{adv}, E_1, ..., E_{m-k}\}$.
7:     Formulate the distance based loss with adversarial embeddings $\{E_1^{adv}, ..., E_k^{adv}\}$ and corresponding clean embeddings $\{E_1, ..., E_k\}$.
8:     Perform one step backward pass and update the parameters in N.
9: **until** training converged

We define the total loss as follows:

$$Loss = \frac{1}{(m-k) + \lambda k} \left(\sum_{i \in CLEAN} L(X_i|y_i) + \lambda \sum_{i \in ADV} L(X_i^{adv}|y_i) \right) + \lambda_2 \sum_{i=1}^{k} L_{dist}(E_i^{adv}, E_i)$$

E_i and E_i^{adv} are the resulting embeddings from X_i and X_i^{adv} respectively. λ_2 is the parameter to control the relative weight of the distance based loss L_{dist} in the total loss. We use $\lambda = 0.3$, $\lambda_2 = 0.0001$, $m = 128$, $k = 64$ for the experiments. We only use one step methods when we generate adversarial examples from the current state of the network and tested the network for all the examples generated by one step or iterative attack methods. As in [5], we used randomly chosen ϵ in the interval $[0, max\_e]$ with clipped normal distribution $N(\mu = 0, \sigma = max\_e/2)$, where max_e is the maximum ϵ used in training.

3.2 Distance Based Loss

Bidirectional loss minimizes the distance between the two embeddings by moving both clean and adversarial embeddings as shown in the left side of the figure 2

$$L_{dist}(E_i^{adv}, E_i) = |E_i^{adv} - E_i|^N, \quad N = 1, 2, ..., \quad i = 1, 2, ..., k$$

4

Table 1: MNIST test results (%). {Baseline: standard training, Kurakin's: adversarial training w/o distance based loss, Bidirection: w/ bidirectional loss, Pivot: w/ pivot loss.}

		step_ll		step_FGSM		iter_ll		iter_FGSM	
ϵ	0	16	32	16	32	16	32	16	32
Baseline	99.6	95.7	30.2	92.6	29.7	98.1	66.2	86.6	7.5
Kurakin's	99.6	99.2	96.7	98.5	94.1	99.5	97.9	98.2	87.9
Bidirection (Ours)	99.5	99.3	**98.8**	99.1	**98.4**	99.4	**99.1**	98.9	**96.5**
Pivot (Ours)	99.6	99.4	**98.8**	99.1	**98.1**	99.5	**99.3**	98.8	**96.4**

We tried $N = 1, 2$ and found not much difference between the two. We report the results with $N = 2$ for the rest of the paper otherwise noted. When $N = 2$, L_{dist} becomes L2 loss.

Pivot loss minimizes the distance between the two embeddings by moving only the adversarial embeddings as shown in the right side of the figure 2

$$L_{dist}(E_i^{adv}|E_i) = |E_i^{adv} - E_i|^N, \quad N = 1, 2, ..., \quad i = 1, 2, ..., k$$

In this case, clean embeddings (E_i) serve as pivots to the adversarial embeddings. [4] The intuition behind the use of pivot loss is that the embedding from clean image can be treated as ground truth embedding.

We found not much difference between two distance based losses. However, we introduce these two losses for those who might be interested in applying our proposed method to give possible knobs for finetuning.

3.3 Experimental Results

We use 20-layer ResNet [10] for MNIST and CIFAR10 classification. 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. [5] We generate adversarial images after these steps.

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 terminated training at 94k iterations for CIFAR10. [6]

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 max_e = 32 and 16 for MNIST and CIFAR10 respectively.

MNIST dataset: Table 1 shows the accuracy results for MNIST test dataset for different types of attack methods. For brevity, we only show results for training with "step_ll" method. As shown in the table, our method achieves 2 - 8 % better accuracy than Kurakin's method. Even though adversarial training is done only with "step_ll", additional regularization increases robustness against *unknown* "step_FGSM", "iter_ll" and "iter_FGSM" attacks. This shows that our low-level similarity learning can successfully regularize the pixel level perturbation on the simple images like MNIST.

CIFAR10 dataset: For CIFAR10 dataset, we compare Kurakin's method and our method in various configurations. First, we train the network with "step_ll" or "step_FGSM" method (step_ll / step_FGSM). We also train the network using both "step_ll" and "step_FGSM" methods to generate adversarial examples, referred to as "step_both". In this case, we generate 32 adversarial examples per each method.

---

[4]In Tensorflow, we create non-trainable variable and update this with clean embedding E_i at every training step. Then, adversarial embedding E_i^{adv} is compared with the non-trainable variable.

[5]Original paper [10] performed 32x32 random crop on zero-padded 40x40 images.

[6]We found that the adversarial training requires longer training time than the standard training. Authors in the original paper [10] changed the learning rate at 32k and 48k iterations and terminated training at 64k iterations.

Table 2: CIFAR10 test results (%) for various adversarial attacks. Adversarial training used either step_ll, step_FGSM, or step_both. We emphasize the numbers in **bold** and <u>underline</u> if those are above and below the "step_ll, Kurakin's" case, respectively.

			step_ll attack				step_FGSM attack			
ϵ		0	2	4	8	16	2	4	8	16
Baseline		90.9	44.5	26.0	14.2	11.5	28.7	20.6	15.7	12.2
step_ll	Kurakin's	91.0	85.8	86.7	87.3	84.4	78.9	83.3	85.2	81.5
	Bidirection (Ours)	91.0	86.1	87.1	88.3	**87.3**	78.7	82.8	**88.0**	**90.0**
	Pivot (Ours)	90.9	86.2	88.0	89.0	**88.3**	79.7	84.8	**90.0**	**91.7**
step_FGSM	Kurakin's	91.1	**87.9**	89.0	89.3	88.0	**88.1**	**92.2**	**95.1**	**93.9**
	Pivot (Ours)	90.9	**87.8**	88.7	89.4	**89.1**	**86.0**	**90.3**	**93.9**	**93.9**
step_both	Kurakin's	91.3	**87.4**	88.6	89.2	87.6	83.3	88.5	93.3	92.0
	Pivot (Ours)	90.8	**87.2**	88.5	89.1	**88.2**	82.2	87.1	91.8	92.3
			iter_ll attack				iter_FGSM attack			
Baseline		90.9	47.7	33.3	12.5	9.1	14.0	0.4	0.0	0.0
step_ll	Kurakin's	91.0	78.2	65.1	35.3	16.8	50.6	9.8	0.4	0.0
	Bidirection (Ours)	91.0	<u>76.6</u>	**69.0**	**41.8**	**22.9**	52.0	11.2	0.2	0.0
	Pivot (Ours)	90.9	<u>76.3</u>	**69.4**	**41.3**	**22.2**	52.0	11.3	0.2	0.0
step_FGSM	Kurakin's	91.1	<u>65.0</u>	<u>51.5</u>	<u>24.2</u>	<u>14.9</u>	<u>41.2</u>	<u>2.3</u>	0.0	0.0
	Pivot (Ours)	90.9	<u>69.5</u>	<u>63.3</u>	38.4	23.4	49.9	6.8	0.1	0.0
step_both	Kurakin's	91.3	<u>70.5</u>	<u>58.9</u>	<u>27.8</u>	<u>15.4</u>	<u>45.7</u>	<u>5.1</u>	0.1	0.0
	Pivot (Ours)	90.8	<u>76.6</u>	**69.7**	**44.0**	**25.5**	**55.0**	**13.1**	0.2	0.0

Table 2 shows the accuracy results for CIFAR10 test dataset for different types of attack methods. In the proposed approach, combining "step_ll" and "step_FGSM" methods with pivot loss shows best results for both step attacks and unknown iterative attacks. In general, except "step_ll" case, Kurakin's method and ours shows similar results against "one step" methods. However, our method shows better results against iterative attacks than Kurakin's method in most cases. The "iter_FGSM" attack is the hardest attack for all trained networks which differs from the results of the training reported in [5]. Possible reason would be CIFAR10 dataset is small compared to ImageNet, thus, direct gradient direction results in stronger attack than the direction for least likely class. We also observe increased accuracy for the clean images for some cases compared to the baseline standard training as in [3, 16], and the authors in [5] explains this might be because the adversarial training acts as a regularizer for overfitted model.

3.4 Effect of λ_2

We train networks with "step_ll, Pivot" and various λ_2s to study effects of the weight of the distance measure in the loss function. Figure 3 shows that a higher λ_2 reduces accuracy of the clean images, and increasing λ_2 above 0.3 even results in divergence of the training. On the contrary, as λ_2 increases, accuracy of the iteratively generated adversarial images increases. Too much regularization for the pixel level perturbation gives higher robustness against iterative attacks, but at the expense of accuracy loss in the clean examples. Hence, 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.
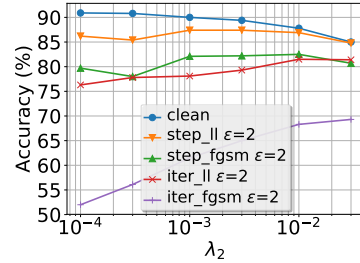
Figure 3: Accuracy vs. λ_2

3.5 Effect of Variations in Initial Condition

To see how the training is sensitive to the initialization, we train 20 networks [7] with Kurakin's method and ours with pivot loss ("step_ll") and report mean accuracy and the standard deviation. As shown in

---

[7]We found adversarial training without distance based loss sometimes ended up with poor results. We, thus, schedule a learning rate at 60k and 90k iterations, and terminate training at 120k iterations for this experiment.

Table 3: CIFAR10 test results (%) for 20 trained networks (Kurakin's and ours w/ pivot loss).

			step_ll attack		iter_ll attack	
	ϵ	0	4	16	4	16
mean $\pm$ std	Kurakin's	90.9 $\pm$ 0.25	86.1 $\pm$ 0.98	84.6 $\pm$ 2.89	64.2 $\pm$ 2.89	16.3 $\pm$ 2.02
	Pivot (Ours)	90.9 $\pm$ 0.25	87.3 $\pm$ **0.79**	87.8 $\pm$ **0.82**	69.4 $\pm$ **1.61**	21.9 $\pm$ **1.85**

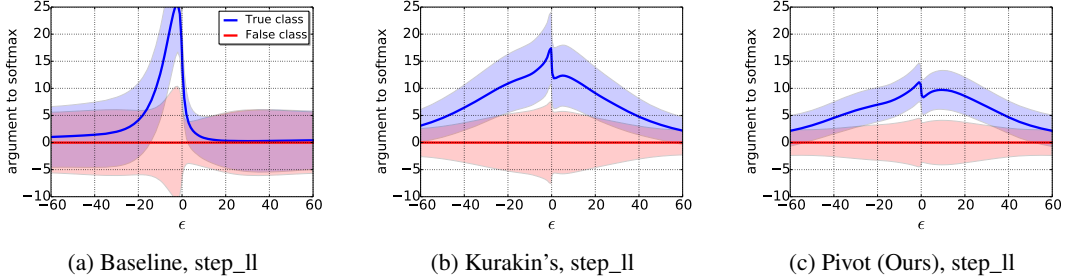(a) Baseline, step_ll (b) Kurakin's, step_ll (c) Pivot (Ours), step_ll

Figure 4: Argument to the softmax vs. ϵ in test time. Kurakin's and Pivot (Ours) were trained with "step_ll". "step_ll" method is used to generate test-time adversarial images. Arguments to the softmax were measured by changing ϵ 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.

table 3, our method produces higher mean accuracies and lower standard deviation of accuracy than Kurakin's method. This suggests that the low-level similarity learning also reduces uncertainties in training, thus, makes networks to be converged similarly. We also found *networks robust to one-step attacks tend to be less robust to iterative attacks and vice versa* for both methods (Kurakin's and Ours), however, this variation was less in our case than in Kurakin's case. Difficulty of optimizing networks for both one-step attacks and iterative attacks can also be found in the following section 6.

4 Analysis of Adversarial Training

4.1 Alternative Embedding Space Visualization

We draw average value of the argument to the softmax layer for the true class and the false classes to visualize how the adversarial training works as in figure 4. Standard training, as expected, shows dramatic drop in the values for the true class as we increase ϵ in "step_ll" directions. [8] With adversarial training, we observe that the value drop is limited at small ϵ and our method even increases the value in certain range upto ϵ=10. Note that adversarial training is not same as the gradient masking [17]. As illustrated in figure 4, it exposes gradient information, however, quickly distort gradients along the sign of the gradient ("step_ll") direction. 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. Even though actual value of the embeddings for the true class in our case is smaller than that in Kurakin's, the standard deviation of our case is less than Kurakin's, making better margin between the true class and false classes.

4.2 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" as in [5]. Interestingly, we observe "label leaking" phenomenon even without providing true labels for adversarial images generation as shown in "step_ll, Pivot" in table 2. We argue that "label leaking" is a natural result of the adversarial training.

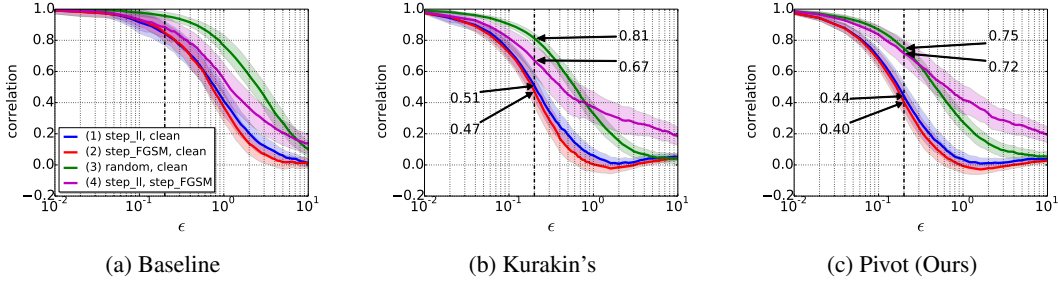[8]"step_FGSM" showed similar phenomenon and additional figures can be found in the Appendix A.

Figure 5: Averaged Pearson's correlation coefficient between the gradients w.r.t. two images. (b) and (c) were trained with "step_ll". 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.

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) 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) Baseline: standard training, (b) Kurakin's and (c) Pivot (Ours). Figure 5 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 between gradient w.r.t. clean and gradient w.r.t. adversarial image (with ϵ) 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 of 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 surfaces (gradients) of the "step_ll" image and "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.

Table 4: CIFAR10 test results (%) under black box attacks. {B: Baseline, K: Kurakin's, P: Pivot (Ours) trained with "step_ll". Target: B1, K1 and P1, source: B2, K2 and P2.} Models with the same number in their name share the same initialization.

| | | step_FGSM attack | | | iter_ll attack | | | iter_FGSM attack | | |
| | | Source | | | Source | | | Source | | |
ϵ	Target	B2	K2	P2	B2	K2	P2	B2	K2	P2
	B1	**16.2**	31.6	29.1	69.7	82.6	78.3	**2.7**	60.1	54.4
16	K1	**66.7**	82.7	79.4	87.7	83.5	82.5	55.8	**28.5**	**28.5**
	P1	**57.2**	89.2	85.4	87.8	83.3	82.3	52.0	**24.3**	**21.7**

5 Black Box Attack Analysis

In table 4, we report black box attack accuracies (defined in section 2) to analyze the transferability between the networks trained with different methods (B: Baseline, K: Kurakin's and P: Pivot loss, "step_ll" is used for training). [9] We haven't included black box attack accuracies between networks with the same initialization in the table 4, since the same initialization assumption for the black box attack scenario isn't realistic. Actually, we have tested black box attacks between the networks with same initialization, and observed slightly lower accuracy (higher transfer rate) than with different initialization which is consistent with the results in [5]. We included the results in the Appendix B.

Transferability (step attack): We first observe that high robustness against one-step attack between defended networks (K2/P2 -> K1/P1), and low robustness between undefended networks (B2 -> B1). This observation shows that error surfaces of neural networks are *driven by the training method* and networks trained with the same method end up similar optimum states. The network once showed robustness against white-box attack (K1, P1) tends to show similar robustness against black-box attacks from networks trained with the same method (K2, P2). Similarly, undefended network originally poor at white-box attack (B1) tends to be poor at black-box attack from another undefended network (B2).

It is noteworthy to observe that the accuracies against step attack from the undefended network (B2) are always lower than those from defended networks (K2, P2). Possible explanation for this would be that adversarial training tweaks gradient seen from the clean image to point toward weaker adversarial point along that gradient direction. As a result, one-step adversarial images from defended networks (K2, P2) become weaker than those from undefended network (B2). Thus, when we mount black box attack with one-step method, it is more efficient to use undefended network as a source network than a defended network. Note that defended network in this case means the network trained with *one-step adversarial images*. There is no guarantee that undefended networks are always better source for black box one-step attack when the target is not trained with one-step adversarial examples.

Another observation here is that, even though our method showed better performance than Kurakin's for white-box attack, our method didn't help increase robustness against black box attacks. Table 4 shows that our method (P1) is better or worse than Kurakin's method (K1) at one-step black box attack when the source networks are defended networks (K2, P2) or undefended network (B2) respectively. However, it is a statistical variation since we found not much difference when we switch the source and the target. Additional details can be found in the Appendix B.

Transferability (iterative attack): First, we also didn't find any meaningful differences between our method and Kurakin's method for black box attacks with iterative methods. This is because the changes that similarity learning makes are subtle in terms of black box attacks. In the following, we treat our method and Kurakin's method as the same training method when we consider the black box attack scenario.

We observe the accuracies for "iter_ll" black box attacks remain high (less transferability) which is consistent with the previous results in [5]. Least likely targeted iterative method seems to be too specific for the source network as explained in [5] since we change the target every iteration and there are many possible choices for each step.

---

[9]"step_ll" is not included due to the limited space and the results were similar with "step_FGSM".

Table 5: CIFAR10 test results (%) for cascade networks {K: Kurakin's, P: Pivot (Ours)} (step_ll: 64 "step_ll" examples, step_ll, saved_P2: 32 "step_ll" examples, 32 images generated by "iter_FGSM" method from P2 in table 4 (P2_iter_FGSM_train) for $k = 64$). P2_iter_FGSM_train images were generated by "iter_FGSM" method with center cropped train images from P2 for $\epsilon= 1,2, ..., 16$. Then, those were used randomly during training. We didn't perform random crop for these images during training to maximize the impact of the adversarial images.

layers	ϵ		0	step_ll attack				step_FGSM attack			
				2	4	8	16	2	4	8	16
20	step_ll	P (Ours)	90.9	86.2	88.0	89.0	88.3	79.7	84.8	90.0	91.7
20	step_ll, saved_P2	P (Ours)	90.7	84.6	<u>82.8</u>	<u>79.4</u>	<u>73.9</u>	80.5	<u>77.6</u>	<u>72.7</u>	<u>65.7</u>
		K	91.3	85.3	<u>83.6</u>	<u>80.7</u>	<u>73.8</u>	80.2	<u>79.1</u>	<u>75.4</u>	<u>67.6</u>
				iter_ll attack				iter_FGSM attack			
20	step_ll	P (Ours)	90.9	76.3	69.4	41.3	22.2	52.0	11.3	0.2	0.0
20	step_ll, saved_P2	P (Ours)	90.7	**82.2**	**75.8**	**67.1**	**58.7**	**67.0**	**37.9**	**20.1**	**9.7**
		K	91.3	**81.4**	69.0	41.3	19.2	**60.7**	**18.9**	1.7	0.1

Interestingly, "iter_FGSM" attack remains very strong even under the black box attack scenario but *only between undefended networks or defended networks*. Possible explanation would be, even though "iter_FGSM" changes its direction at every step, since it doesn't change the target and the networks trained with the same policy end up similar optimum, the resulting adversarial images become globally difficult examples *within the networks trained with the same strategy*.

Difficulty of defense/attack under the black box attack scenario: As seen from the previous observation, iter_FGSM images crafted from a network remain strong *only on the networks trained with the same strategy*. Thus, it is efficient to attack an undefended/defended network with iter_FGSMl examples crafted from another undefended/defended networks. The problem is we (as attackers/defenders) don't have information about the target/source network under the black box attack scenario. Defenders always have to consider the worst case scenario, and this makes the problem much more difficult. We recommend reporting accuracies for the adversarial examples crafted from other networks trained with the *same strategy* under the black box scenario.

6 Cascade Adversarial Training

Inspired by the observation that adversarial images generated with iter_FGSM method transfer between defended networks, we propose *cascade adversarial training*, which trains a network by injecting pre-saved iter_FGSM images crafted from an already defended network to improve the robustness against iter_FGSM attack. The intuition behind this proposed method is that we *transfer the knowledge of the end results of adversarial training*.

Experimental results: We train networks from scratch with/without low-level similarity learning by injecting iter_FGSM images (P2_iter_FGSM_train images) crafted from the already defended network P2 (in table 4) in addition to one-step adversarial images generated from the network being trained. We chose P2 since the transfer rate was high among the networks differently initialized from K1 and P1. [10]

We observe increase in accuracy against iterative attacks, although at the expense of decreased accuracy for "one step" attacks as shown in table 5. Again, this observation also supports an idea that networks robust against one-step attacks tend to be less robust against iterative attacks and vice versa as previously seen from the section 3.5. Interestingly, the cascade network trained with low-level similarity learning (step_ll, saved_P2, P (Ours)) shows superior results than the cascade network trained without similarity learning (step_ll, saved_P2, K) especially for the iterative attacks. This suggests that our low-level similarity learning successfully regularize *unknown* pixel level perturbations together with cascade adversarial training.

---

[10]We also trained the networks with saved adversarial images from K2 and even P1, and found not much difference on the accuracy.

Table 6: CIFAR10 test results (%) for cascade 110-layer ResNet models. The same training methods were used as in table 5. We used saved version of iter_FGSM images generated from the 20-layer network (P2 in table 4).

				step_ll attack				step_FGSM attack			
layers	ϵ		0	2	4	8	16	2	4	8	16
110	step_ll	P (Ours)	92.3	86.0	86.8	89.3	89.4	81.6	84.5	90.1	91.6
110	step_ll, saved_P2	P (Ours)	91.4	85.6	83.9	81.4	76.8	81.2	79.3	74.3	67.2
		K	92.2	85.4	83.7	80.6	73.9	81.4	80.1	75.8	66.8
				iter_ll attack				iter_FGSM attack			
110	step_ll	P (Ours)	92.3	79.6	66.2	48.6	34.1	64.1	20.9	2.6	0.1
110	step_ll, saved_P2	P (Ours)	91.4	**83.8**	**77.8**	**69.8**	**61.6**	**69.9**	**45.9**	**30.4**	**19.5**
		K	92.2	**84.6**	**75.6**	**56.4**	**33.5**	**69.9**	**35.2**	**9.8**	1.2

Table 7: CIFAR10 test results (%) for cascade models under black box attacks {B: Baseline, K: Kurakin's, P: Pivot (Ours), K_C: Kurakin's with cascade adversarial training, P_C: Pivot (Ours) with cascade adversarial training}. 1 and 2: 20-layer ResNet, 3 and 4: 110-layer ResNet. Models with the same number in their name share the same initialization.

		step_FGSM attack							iter_FGSM attack						
		Source							Source						
ϵ	Target	B2	K2	P2	$K2_C$	$P2_C$	$K4_C$	$P4_C$	B2	K2	P2	$K2_C$	$P2_C$	$K4_C$	$P4_C$
16	$K1_C$	60.2	65.7	58.7	72.6	73.6	63.3	69.0	**42.5**	69.1	68.5	51.3	61.7	76.4	81.0
	$P1_C$	61.3	68.3	65.1	71.2	73.9	66.6	71.5	50.5	69.0	71.0	**50.3**	62.4	77.1	81.5
	$K3_C$	73.4	73.3	70.8	75.8	77.7	69.3	74.6	62.8	73.8	75.4	**58.3**	71.4	78.4	83.0
	$P3_C$	71.8	74.8	72.7	76.6	75.4	68.3	74.4	67.5	77.5	78.1	**66.8**	68.8	80.2	83.5

Deeper networks: We study the application of similarity learning and cascade adversarial training to higher capacity (110-layer ResNets) networks to see whether we can recover the accuracy against one-step attacks for the network with higher capacity. We first train a network only with our similarity learning. As expected from [5], we observe that higher network capacity improves accuracy (see rows "step_ll" in table 6).

Next, we use the same adversarial images (P2_iter_FGSM_train) generated from 20-layer ResNet (P2 in table 4) for training the 110-layer ResNet (rows step_ll, saved_P2). Interestingly, accuracy against one-step and iterative attacks are improved as shown in table 6 compared to 20-layer counterparts. The results show that (1) deeper networks with our methods (low-level similarity learning and cascade adversarial training) can help further increase robustness against both one-step and iterative adversarial attacks and (2) we can even transfer the knowledge learned from one network to another network regardless of the network architecture *if we train a network with the same strategy*.

Black box attack for the knowledge transferred networks: We finally perform black box attack analysis for the networks trained with cascade adversarial training. We re-train 20-layer and 110-layer ResNet models using cascade adversarial training (with P2_iter_FGSM_train images) with/without low-level similarity learning and use those networks as source networks for black-box attacks. 20-layer ResNet models without cascade adversarial training are also included as source networks. Target networks are the same networks used in table 5 and 6.

As shown in table 7, we achieve huge accuracy increase under the various black box attack scenarios. We observe that the hardest attacks are the iter_FGSM attacks from a source network trained with cascade adversarial training ($K2_C$) except B2 -> $K1_C$ case. Possible solution to further improve the robustness against various source networks would be to use more than one network as source networks (including pure networks, defended networks with/without cascade learning) for iter_FGSM images generation for cascade adversarial training.

We also observe networks trained with both low-level similarity learning and cascade adversarial training ($P1_C$ and $P3_C$) show better *worst-case* performance than networks trained without low-level

similarity learning (K1_C and K3_C). [11] This observation differs from the results in the previous section 5 where we didn't find meaningful difference between networks trained with/without low-level similarity learning under the black box attack. This suggests that *together with cascade adversarial training*, regularization effect of low-level similarity learning further enhance robustness against perturbations caused by an iterative method *even under the black-box attack scenario*. Without cascade adversarial training, iterative attacks are *fully unknown* attacks, however, cascade learning enables networks to learn partial knowledge of perturbations generated by an iterative method from other networks. And the learned perturbations become regularized further with our low-level similarity learning.

7   Conclusion

We 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. The networks trained with low-level similarity learning showed higher robustness against *one-step* and *iterative* attacks under white box attack.

We showed our approach is less sensitive to the randomness of the training. We also showed that modifying the weight of the distance measure can help control trade-off between accuracies for clean and adversarial examples. We analyzed error surface similarities between the clean and their corresponding adversarial images and found label leaking phenomenon is a natural result of adversarial training.

We performed through transfer analysis and showed 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. Combining those two techniques (low level similarity learning + cascade adversarial training) with deeper networks further improved robustness against iterative attacks for both white-box and black-box attacks.

However, there is still a gap between accuracy for the clean images and that for the adversarial images. Improving robustness against both one-step and iterative attacks still remains challenging since it is shown to be difficult to train networks robust for both one-step and iterative attacks simultaneously. Future research is necessary to further improve the robustness against iterative attack *without* sacrificing the accuracy for step attacks or clean images under both white-box attack and black-box attack scenarios.

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.

---

[11] We also observed this when we used K2_C, P2_C, K4_C and P4_C for target networks. Additional details can be found in the Appendix B.

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

[7] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[8] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2015.

[9] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII*, pages 499–515, 2016.

[10] 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.

[11] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[12] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

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

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

[15] 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.

[16] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing by virtual adversarial examples. *CoRR*, abs/1507.00677, 2015.

[17] Nicolas Papernot, Patrick D. McDaniel, Arunesh Sinha, and Michael P. Wellman. Towards the science of security and privacy in machine learning. *CoRR*, abs/1611.03814, 2016.
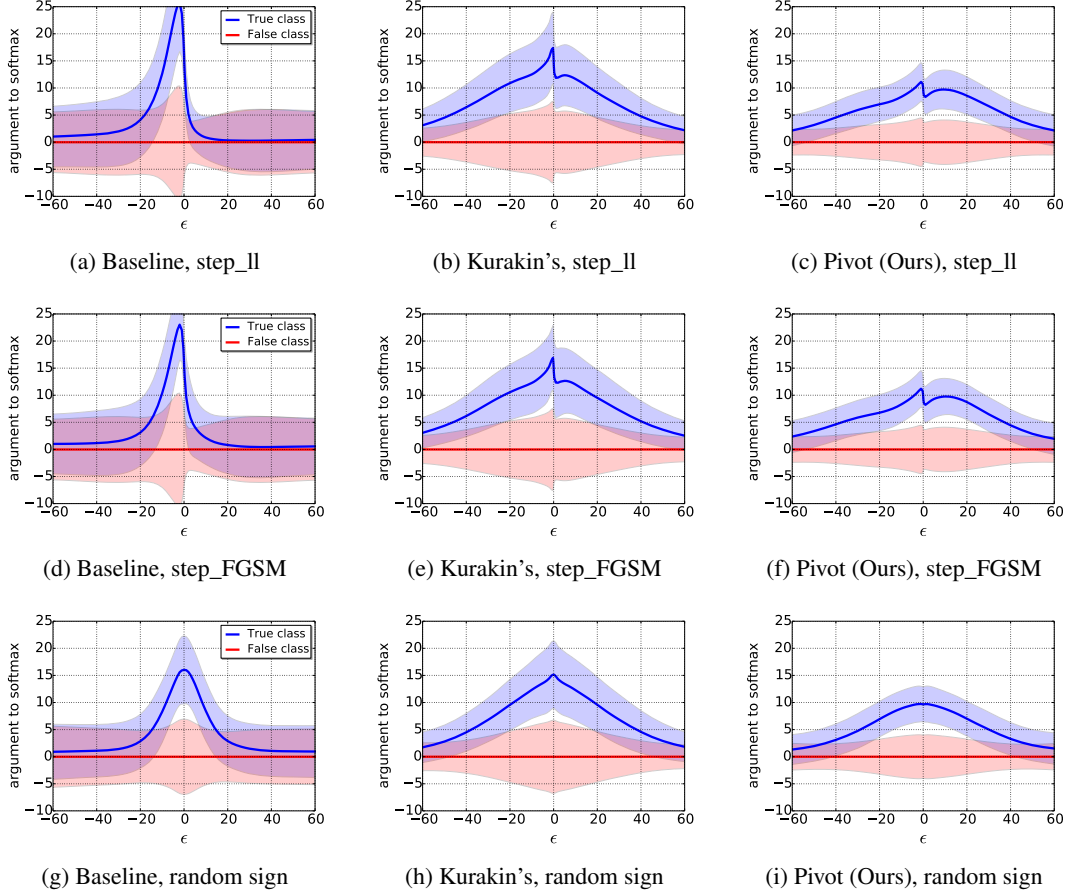
A More Figures on Embeddings

(a) Baseline, step_ll   (b) Kurakin's, step_ll   (c) Pivot (Ours), step_ll

(d) Baseline, step_FGSM   (e) Kurakin's, step_FGSM   (f) Pivot (Ours), step_FGSM

(g) Baseline, random sign   (h) Kurakin's, random sign   (i) Pivot (Ours), random sign

Figure 6: Argument to the softmax vs. ϵ in test time. Kurakin's and Pivot (Ours) were trained with "step_ll". "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 6, 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.

B   Additional Black Box Attack Results

Table 8: Test accuracy (%) for adversarial examples with different attack methods from different networks {B: Baseline, K: Kurakin's, P: Pivot (Ours) trained with "step_ll". Target networks: B1, K1 and P1, source networks: B1, K1 and P1.} B1, K1 and P1 used the same initialization.

		step_FGSM attack			iter_ll attack			iter_FGSM attack		
		Source			Source			Source		
ϵ	Target	B1	K1	P1	B1	K1	P1	B1	K1	P1
	B1	**12.2**	27.4	27.5	9.3	78.6	80.7	0.0	45.9	44.7
16	K1	**65.7**	81.5	81.8	87.7	16.8	81.8	51.5	0.0	**18.2**
	P1	**58.1**	89.3	91.7	88.1	81.9	22.3	48.9	**13.4**	0.0

Table 8 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 9: Test accuracy (%) for adversarial examples with different attack methods from different networks {B: Baseline, K: Kurakin's, P: Pivot (Ours) trained with "step_ll". Target networks: B2, K2 and P2, source networks: B1, K1 and P1.} B1, K1 and P1 used the same initialization; B2, K2 and P2 used the same initialization, but different from B1, K1 and P1.

		step_FGSM attack			iter_ll attack			iter_FGSM attack		
		Source			Source			Source		
ϵ	Target	B1	K1	P1	B1	K1	P1	B1	K1	P1
	B2	**17.9**	33.9	34.5	73.9	82.2	82.8	**4.1**	54.8	54.3
16	K2	**65.0**	84.6	84.5	87.6	81.5	82.3	61.2	**25.3**	**30.4**
	P2	**66.4**	88.2	87.2	88.3	83.5	83.9	61.6	**27.7**	**36.1**

In table 9, our method (P2) is always better at one-step black box attack from defended networks (K1, P1) and undefended network (B1) than Kurakin's method (B2). However, it is hard to tell which method is better than the other one as explained in the main paper.

Table 10: Test accuracy (%) under black box attack {B: Baseline, K: Kurakin's, P: Pivot (Ours), K_C: Kurakin's with cascade adversarial training, P_C: Pivot (Ours) with cascade adversarial training}, 1 and 2: 20-layer ResNet, 3 and 4: 110-layer ResNet. Models with the same number in their name share the same initialization.

		iter_FGSM attack								
		Source								
ϵ	Target	B1	$K1_C$	$P1_C$	$K2_C$	$P2_C$	$K3_C$	$P3_C$	$K4_C$	$P4_C$
	$K2_C$	**41.2**	58.4	63.6	-	-	50.0	58.3	75.0	82.0
16	$P2_C$	**51.7**	62.9	62.5	-	-	53.7	64.0	76.8	81.7
	$K4_C$	62.7	68.2	70.3	**52.4**	68.7	55.6	65.8	-	-
	$P4_C$	76.4	69.9	71.3	**65.1**	69.0	61.7	67.7	-	-

In table 10, we show black box attack accuracies with the source and the target networks switched from the main paper. In this case, we also observe that networks trained with both low-level similarity learning and cascade adversarial training ($P2_C$ and $P4_C$) show better *worst-case* performance than networks trained without low-level similarity learning ($K2_C$ and $K4_C$). After cascade adversarial training, iteratively generated adversarial examples from a network trained with the same strategy become sometimes no longer strongest attacks (i.e. B1 -> $K2_C$ and B1 -> $K2_C$). This also suggests that the necessity of using various source networks (including pure networks, defended networks with and without cascade learning) for iter_FGSM images generation for cascade adversarial training.