Supplemental Material

We show more details about our experiments in this appendix. Our experiments mainly contain two parts, the first one is to examine whether the new triplet loss function can improve the performance of adversarial training or not, and the performance when triplet loss is taken as a regularization. The second one is to apply our new triplet loss function to current defense methods. We do experiments over three datasets for the first part, Cats vs. Dogs, MNIST and CIFAR10. For the second part, we follow the setting of the original papers of these defense methods. Our new triplet loss is defined as follows,

$$\begin{aligned}\hat{\ell}(\mathbf{x}, y) =& \frac{1}{(1+\lambda_1)k}\left(\sum_{i\in k}\ell(\mathbf{x}_i, y) + \lambda_1\sum_{i\in k}\ell(\mathbf{x}_i^{adv}, y)\right) \\ &+ \frac{\lambda_2}{k}\sum_i \max\Big\{\|f(\mathbf{x}_i^{adv}) - f(\mathbf{x}_i)\| \\ &- \|f(\mathbf{x}_i^{adv}) - f(\mathbf{x}_i^{n})\| + \alpha, 0\Big\}\end{aligned}$$

1 Adversarial Training with Triplet Loss (AT^{2}L)

In this section, we show results over three datasets, Cats vs. Dogs, MNIST and CIFAR10. The attack methods used in this part are FGSM, I-FGSM, LL, I-LL and C&W. We set ϵ as 0.3, which is the scale of FGSM, I-FGSM, LL and I-LL. For iteration algorithms like I-FGSM and I-LL, we set the number of iteration as 10 and $\epsilon = 0.03$ for each iteration. The max-iteration of C&W is set to be 1000. The initial constant c is set to be 1e$-$3 and the largest value of c to go up to before giving up is 2. The rate at which we increase constant is 2 and the learning rate of C&W attack is 5e$-$3. The performance of our algorithm is stable in a large range of the hyper-parameters. λ_1 and λ_2 are designed to adjust the weight of the original loss, adversarial loss and triplet loss. α is designed to tune the margin between different classes. In our experiments, with fixed λ_2, the robustness of the model is stable when λ_1 varies from 0.2 to 1.5. With fixed λ_1, the robustness is stable when λ_2 varies from 0.1 to 5. As for α in the triplet loss, the robustness is stable when varying from 0.1 to 2.0 with step-size 0.1.

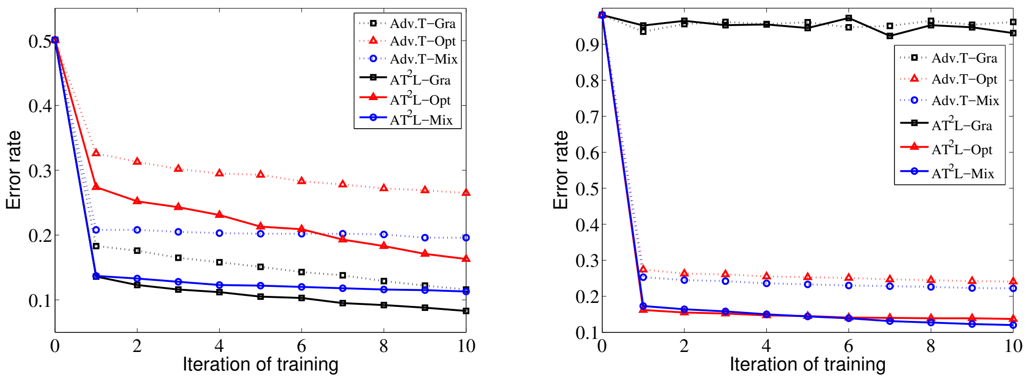
1.1 Cats vs. Dogs

We first do experiments on Cats vs. Dogs dataset. The model structures we chose are VGG16, VGG19, inceptionv3, resnet50. Here we use FGSM, I-FGSM, LL, and I-LL to generate adversarial examples and notate these methods as gradient training methods and we select C&W as the optimization attack method for the purpose of adversarial training. The margin α and λ is set to be 0.5 and 0.3. We train an ensemble model which covers VGG16, VGG19, inceptionv3, resnet50 because each picture of this dataset is 224x224x3 and simple networks may not work well. The accuracy of the model over clean data is 91.2%.
The known type of attack is proposed to train an ensemble model over model VGG16, VGG19, inceptionv3 and resnet50 when the original model $f(\cdot)$ is VGG16. We then attack the model with the adversarial examples generated against model VGG16. The unknown type of attack is designed to train the ensemble model over model VGG19, inceptionv3 and resnet50 when the original model $f(\cdot)$ is VGG19. We then attack the model with the adversarial examples generated against model VGG16 so that the attackers are not ware of the model's structure during the training process.
The accuracy of adversarial training with triplet loss against gradient-based adversarial examples is 4.7%, which is better than normal adversarial training without triplet loss. Even after 10 iterations of adversarial training, the algorithm of adversarial training with triplet loss is still better. When trained with optimization-based algorithm like C&W, the robustness against gradient-based adversarial examples is worse than when trained with gradient-based algorithm. However, the mixed version of both gradient-based and optimization-based algorithm works well regardless of the type of the attack.

The parameters in the triplet loss function are set to be $\lambda_1 = 0.3, \lambda_2 = 1, \alpha = 1.0$, and the results are shown in Figure 1, Figure 2, Table 1, and Table 2.

Attack method	Training method	Adv. Train(1)	Adv. Train(10)	AT^{2}L(1)	AT^{2}L(10)
FGSM	Gradient-based	18.3	11.6	13.6	**8.3**
	Optimization-based	32.6	26.5	27.4	16.3
	Mixed	20.8	19.6	13.7	11.3
C&W	Gradient-based	93.5	96.2	95.2	93.1
	Optimization-based	27.4	24.1	16.2	13.7
	Mixed	25.3	22.2	17.3	**12.0**

Table 1: Error rate of known type of attack on Cats vs. Dogs. We train over model VGG16, VGG19, inceptionv3, resnet50, and we test the error rate over model VGG16. Baseline is traditional adversarial training. We use different algorithm to generate adversarial examples for training.



(a) Attack by FGSM (b) Attack by C&W

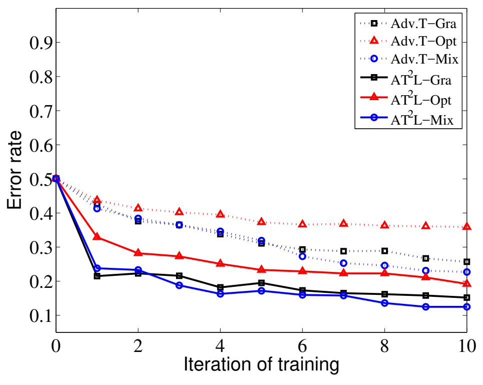
Figure 1: Results on Cats vs. Dogs dataset. 'Adv.T' means traditional adversarial training. '-Gra' means the training process use the gradient-based algorithms to generate adversarial examples. '-Opt' means optimization-based algorithm C&W and '-Mix' means mixed version of algorithm.

Attack method	Training method	Adv. Train(1)	Adv. Train(10)	AT^{2}L(1)	AT^{2}L(10)
FGSM	Gradient-based	42.6	25.7	21.5	15.2
	Optimization-based	43.7	35.9	32.9	19.2
	Mixed	41.3	22.7	23.8	**12.5**
C&W	Gradient-based	99.3	95.4	92.1	94.6
	Optimization-based	32.8	30.3	18.3	15.2
	Mixed	27.6	24.6	19.5	**13.5**

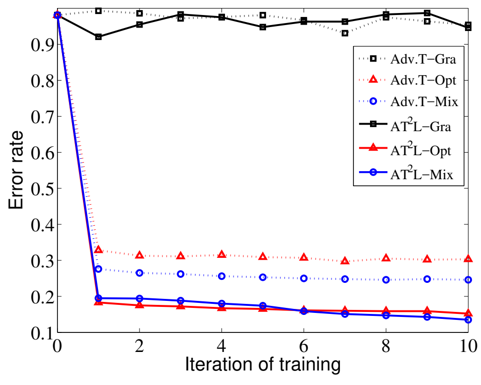
Table 2: Error rate of unknown type of attack on Cats vs. Dogs. We train over model VGG16, VGG19, inceptionv3, resnet50, and we test the error rate over model VGG16. Baseline is traditional adversarial training. We use different algorithm to generate adversarial examples for training.

1.2 MNIST

We then tested our algorithm on MNIST. The size of instance of MNIST is not as large as the instance of Cats vs. Dogs, but the number of classes is a bit larger. Each picture of MNIST is 28x28x1, so we construct 4 different models for the ensemble training process (which are shown in the Table 6). Model A, B, C are CNNs with different constructions, and model D only contains multiply dense layers and dropout layers. We perform a known type of attack and a unknown type of attack on this dataset.
The known type of attack is proposed to train an ensemble model over model A, B, C and D (M in algorithm 2 is set to be [A, B, C, D]) and then attack the model with the adversarial examples generated against model A. Here we use FGSM, I-FGSM, LL, I-LL and a combination of these 4 algorithms to generate adversarial examples for training. The model we pre-trained (Step 1 of



(a) Attack by FGSM



(b) Attack by C&W

Figure 2: Results on Cats vs. Dogs dataset. 'Adv.T' means traditional adversarial training. '-Gra' means the training process use the gradient-based algorithms to generate adversarial examples. '-Opt' means optimization-based algorithm C&W and '-Mix' means mixed version of algorithm.

Algorithm 2) reaches up to 99.1% accuracy. The parameter k is 32, λ_1 is 0.3, $\lambda_2 = 1.0$ and α is set to be 1.0. The model after normal adversarial training or adversarial training with triplet loss does not lose the accuracy, its accuracy is still over 99% on average. We also do more experiments about the performance of different gradient-based attack methods and their combination. The result shows that AT^{2}L can increase the robustness of the model. When training with the mixed version, the robustness against corresponding attack methods shows the best performance.
The unknown type of attack is proposed to train an ensemble model over model A, B and D (M in algorithm 2 is set to be [A, B, D]) and then attack the model with the adversarial examples generated against model C. Other settings are the same as the experiments of known type of attack. The result is quite similar as known type of attack, and the mixed version of AT^{2}L which is trained by adversarial examples against both gradient-based algorithms and optimization-based algorithms shows the lowest error rate.
The parameters in the triplet loss function are set to be $\lambda_1 = 0.3, \lambda_2 = 1, \alpha = 1.0$, and the results are shown in Figure 3, Figure 4, Table 4, and Table 5. The model structures we used in the experiments are shown in Table 3.

A	B	C	D
Input	Input	Input	Input
Conv(64,5,5)	Conv(64,8,8)	Conv(128,3,3)	Dense(300)
ReLu	ReLu	ReLu	ReLu
Conv(64,5,5)	Conv(128,6,6)	Conv(64,3,3)	Dropout
ReLu	ReLu	ReLu	Dense(300)
Dropout	Conv(128,5,5)	Dropout	ReLu
Dense(128)	ReLu	Flatten	Dropout
ReLu	Dropout	Dense(128)	Dense(300)
Dropout	Flatten	ReLu	ReLu
Dense(10)	Dense(10)	Dropout	Dropout
		Dense(10)	Dense(10)

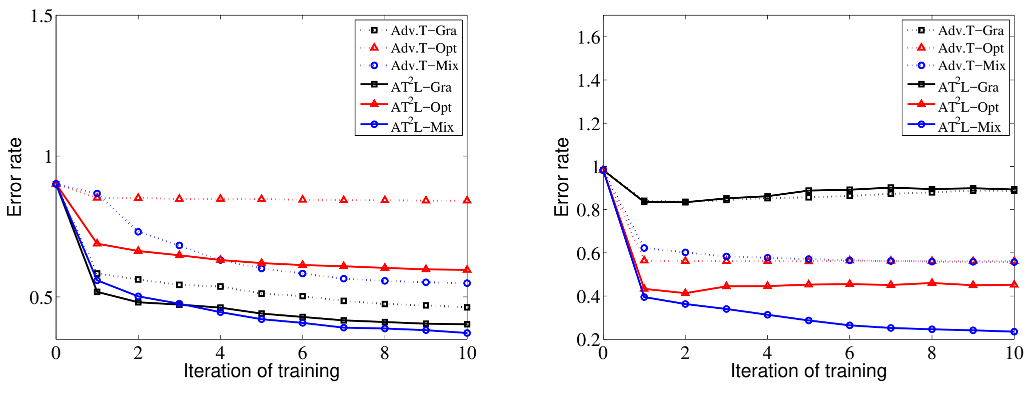
Table 3: Structures of models on MNIST.

1.3 CIFAR10

MNIST is a standard dataset in the research of adversarial examples, but it's a small dataset and we further do more experiments over CIFAR10. The models we choose for CIFAR10 are the same as the Cats vs. Dogs dataset: VGG16, VGG19, inceptionv3 and resnet50.
We also do a known type of attack and a unknown type of attack on CIFAR10. The known type of attack proposes to train an ensemble model over model VGG16, VGG19, inceptionv3 and

Attack method	Training method	Adv. Train(1)	Adv. Train(10)	AT^{2}L(1)	AT^{2}L(10)
FGSM	FGSM	56.9	42.0	54.0	**34.8**
	iter_FGSM	83.3	88.3	66.6	38.3
	LL	56.7	48.1	46.2	42.9
	iter_LL	87.9	83.8	53.0	44.1
	Gradient-based	58.3	46.3	51.8	40.3
	Optimization-based	85.2	84.2	68.9	59.6
	Mixed	86.7	54.9	55.9	37.2
C&W	FGSM	91.6	86.8	91.1	88.6
	iter_FGSM	83.0	88.9	77.9	85.2
	LL	82.6	91.3	86.8	88.5
	iter_LL	79.8	84.2	89.9	85.2
	Gradient-based	84.1	88.7	83.5	89.3
	Optimization-based	56.4	56.1	43.3	45.2
	Mixed	62.2	55.7	39.5	**23.5**

Table 4: Error rate of known type of attack on MNIST. We train over model A,B,C,D, and we test the error rate over model A. Baseline is traditional adversarial training. We use different algorithm to generate adversarial examples for training. We then use FGSM or C&W for testing the robustness.

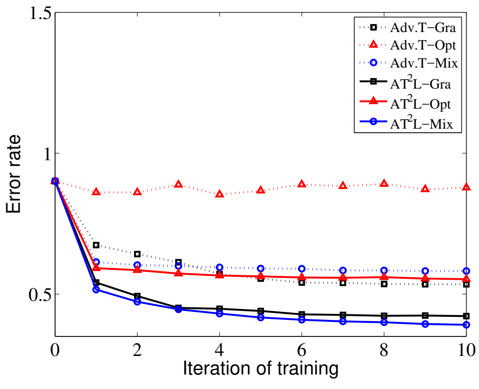


(a) Attack by FGSM (b) Attack by C&W

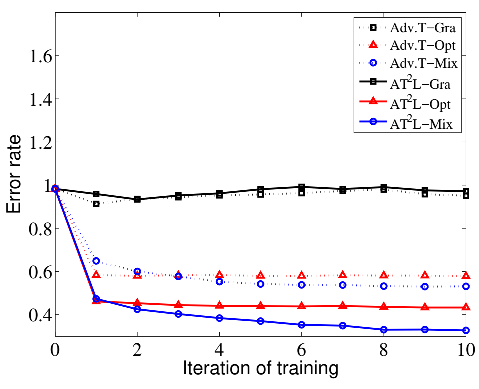
Figure 3: Results on MNIST dataset of known type of attack. 'Adv.T' means traditional adversarial training. '-Gra' means the training process use the gradient-based algorithms to generate adversarial examples. '-Opt' means optimization-based algorithm C&W and '-Mix' means mixed version of algorithm.

Attack method	Training method	Adv. Train(1)	Adv. Train(10)	AT^{2}L(1)	AT^{2}L(10)
FGSM	FGSM	59.3	52.6	52.1	45.2
	iter_FGSM	83.5	85.8	63.6	49.1
	LL	62.1	54.7	45.3	**38.5**
	iter_LL	82.5	86.1	51.9	45.1
	Gradient-based	67.4	53.5	54.1	42.2
	Optimization-based	86.1	87.8	59.2	55.3
	Mixed	61.4	58.2	51.6	39.1
C&W	FGSM	95.1	97.5	95.2	92.9
	iter_FGSM	95.8	94.2	87.2	89.1
	LL	94.1	93.2	93.9	86.6
	iter_LL	97.9	94.7	98.5	83.1
	Gradient-based	91.3	95.1	95.9	97.2
	Optimization-based	58.3	57.8	46.1	43.3
	Mixed	64.9	53.1	47.3	**32.7**

Table 5: Error rate of unknown type of attack on MNIST. We train over model A,B,D, and we test the error rate over model C. Baseline is traditional adversarial training. We use different algorithm to generate adversarial examples for training. We then use FGSM or C&W for testing the robustness.



(a) Attack by FGSM



(b) Attack by C&W

Figure 4: Results on MNIST dataset of unknown type of attack. 'Adv.T' means traditional adversarial training. '-Gra' means the training process use the gradient-based algorithms to generate adversarial examples. '-Opt' means optimization-based algorithm C&W and '-Mix' means mixed version of algorithm.

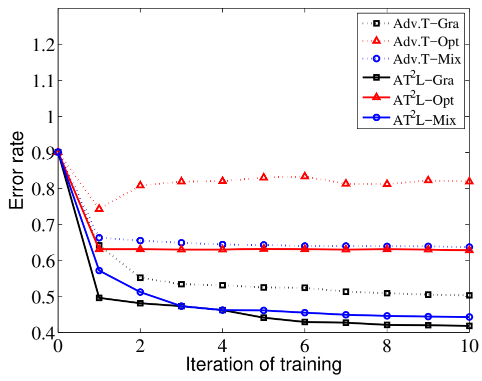
resnet50 (M in algorithm 2 is set to be [VGG16, VGG19, inceptionv3, resnet50]) and then attack the model with the adversarial examples generated against model VGG16. The unknown type of attack is proposed to train over models mentioned above without VGG16 and attack the model against VGG16. The setting of parameters over both attacks are the same: $k = 32, \alpha = 1.0, \lambda_1 = 0.6, \lambda_2 = 1.0$. The accuracy of the pre-trained model (Step 1 of Algorithm 2) reaches up to 92.6% and the accuracy does not decrease much after adversarial training (near 92%). The results of unknown type of attack are a little worse than the known type of attack, but the robustness of the model indeed increases. The error rate of the original model against FGSM is 90.1% and that of the original model against C&W is 99.2%. From the results we can see that our AT^{2}L trained over ensemble models and mixed attack methods shows the best performance against both gradient-based and optimization-based attack. The triplet loss can increase the robustness of the model better than the normal adversarial training without triplet loss.
The model structures used to generate adversarial examples are the same as the setting of dataset Cats vs. Dogs. We experiment both known type of and unknown type of settings and the result shows that our special triplet loss can improve the robustness of the model better than traditional adversarial training process. The results are shown in Figure 5, Figure 6, Table 6, and Table 7.

Attack method	Training method	Adv. Train(1)	Adv. Train(10)	AT^{2}L(1)	AT^{2}L(10)
FGSM	Gradient-based	64.2	50.3	49.6	**41.8**
	Optimization-based	74.3	81.9	63.1	62.8
	Mixed	66.3	63.7	57.2	44.3
C&W	Gradient-based	97.1	98.8	96.3	97.9
	Optimization-based	53.9	43.0	46.1	**35.7**
	Mixed	62.5	61.1	53.9	44.4

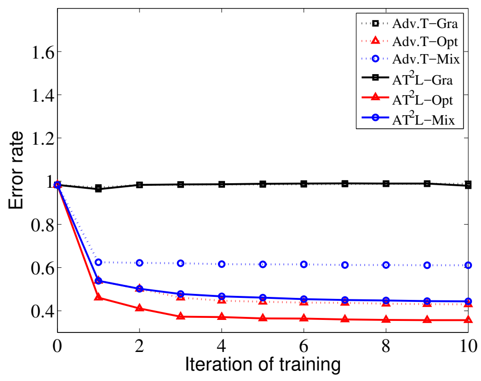
Table 6: Error rate of known type of attack on CIFAR10. We train over model VGG16, VGG19, inceptionv3, resnet50, and we test the error rate over model VGG16. Baseline is traditional adversarial training. We use different algorithm to generate adversarial examples for training. We then use FGSM or C&W for testing the robustness.

2 Apply to current defense

The second part of our experiment is to apply our new loss to the existence defense methods. We choose three typical defense methods described in the paper, thermometer encoding, mitigating through randomization, and Defense-GAN. We propose to improve the robustness of the model based on these defenses.



(a) Attack by FGSM

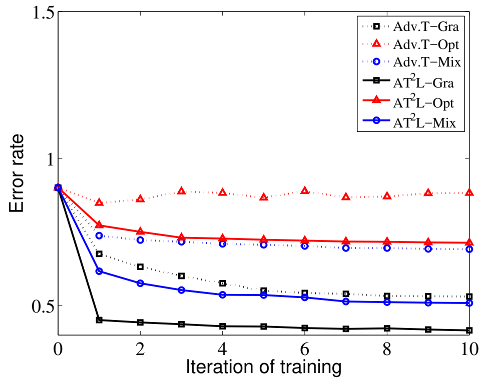


(b) Attack by C&W

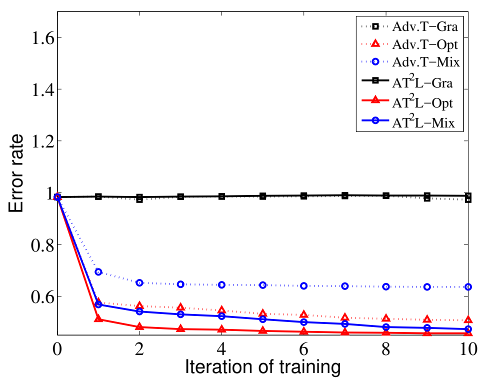
Figure 5: Results on CIFAR10 dataset of known type of attack. ‘Adv.T’ means traditional adversarial training. ‘-Gra’ means the training process use the gradient-based algorithms to generate adversarial examples. ‘-Opt’ means optimization-based algorithm C&W and ‘-Mix’ means mixed version of algorithm.

Attack method	Training method	Adv. Train(1)	Adv. Train(10)	AT^{2}L(1)	AT^{2}L(10)
FGSM	Gradient-based	67.6	53.1	45.1	**41.6**
	Optimization-based	84.9	88.3	77.3	71.4
	Mixed	73.8	69.2	61.7	50.9
C&W	Gradient-based	98.5	97.3	98.5	98.8
	Optimization-based	57.6	50.7	51.1	**45.7**
	Mixed	69.4	63.6	56.8	47.3

Table 7: Error rate of unknown type of attack on CIFAR10. We train over model VGG19, inceptionv3, resnet50, and we test the error rate over model VGG16. Baseline is traditional adversarial training. We use different algorithm to generate adversarial examples for training. We then use FGSM or C&W for testing the robustness.



(a) Attack by FGSM



(b) Attack by C&W

Figure 6: Results on CIFAR10 dataset of unknown type of attack. ‘Adv.T’ means traditional adversarial training. ‘-Gra’ means the training process use the gradient-based algorithms to generate adversarial examples. ‘-Opt’ means optimization-based algorithm C&W and ‘-Mix’ means mixed version of algorithm.

2.1 Thermometer encoding

Thermometer encoding used a phenomenon called Gradient Shattering. This defense is designed to encode the original input to a non-differentiable image and the original defense is combined with a traditional adversarial training process. So our triplet regularization can be easily inserted into this defense. We only need to change the loss function during the adversarial training process to our new loss function and we do not need to generate different types of adversarial examples or against different model structures. In this experiment, we view the image after the thermometer encoding as the adversarial examples. In the known type of setting, we use the same model structure in both training and testing process and in the unknown type of setting we apply different structures. We show the results of the first round of adversarial training and the results after 7 rounds of training.
The parameters in the triplet loss function are set to be $\lambda_1 = 0.3, \lambda_2 = 0.7, \alpha = 1.5$, and the results are shown in Table 8-11. In each table, we test our model with three types of examples, i.e. clean data, examples generated by FGSM, and examples generated by PGDLS-PGA.

	Ori	Th. En.(1)	Th. En.(7)	Th. En.+ Tri. Reg(1)	Th. En.+ Tri. Reg(7)
Clean	**0.8**	**0.8**	0.9	**0.8**	**0.8**
FGSM	100.0	31.6	4.2	28.7	**3.5**
PGD/LS-PGA	100.0	25.0	5.9	23.0	**2.7**

Table 8: Error rate of known type of attacks on MNIST over Thermometer Models. ‘Th. En.’ mean Thermometer Encoding. ‘Tri. Reg’ means applying our triplet regularization.

	Ori	Th. En.(1)	Th. En.(7)	Th. En.+ Tri. Reg(1)	Th. En.+ Tri. Reg(7)
Clean	**5.8**	7.6	10.1	6.1	7.2
FGSM	51.5	37.1	20.0	27.6	**15.1**
PGD/LS-PGA	49.5	39.3	20.9	29.7	**13.4**

Table 9: Error rate of known type of attacks on CIFAR10 over Thermometer Models. ‘Th. En.’ mean Thermometer Encoding. ‘Tri. Reg’ means applying our triplet regularization.

	Ori	Th. En.(1)	Th. En.(7)	Th. En.+ Tri. Reg(1)	Th. En.+ Tri. Reg(7)
Clean	**3.5**	5.1	5.5	5.0	9.2
FGSM	89.1	73.7	67.0	53.1	**47.9**
PGD/LS-PGA	88.2	77.1	71.9	62.8	**55.3**

Table 10: Error rate of unknown type of attacks on MNIST over Thermometer Models. ‘Th. En.’ mean Thermometer Encoding. ‘Tri. Reg’ means applying our triplet regularization.

	Ori	Th. En.(1)	Th. En.(7)	Th. En.+ Tri. Reg(1)	Th. En.+ Tri. Reg(7)
Clean	**11.5**	13.6	20.1	16.2	17.7
FGSM	46.5	43.8	39.1	38.6	**33.0**
PGD/LS-PGA	55.0	52.9	49.3	51.9	**43.0**

Table 11: Error rate of unknown type of attacks on CIFAR10 over Thermometer Models. ‘Th. En.’ mean Thermometer Encoding. ‘Tri. Reg’ means applying our triplet regularization.

2.2 Mitigating through randomization

Mitigating through randomization uses Stochastic Gradients which are caused by randomized defenses. The input is randomly transformed before being fed to the classifier, causing the gradients to become randomized. The target models and the defense models are exactly the same except for the parameter settings of the randomization layers, i.e., the randomization parameters of the target models are predefined while randomization parameters of the defense models are randomly generated at test time.
The traditional adversarial training process is also mentioned in the original paper. So our special triplet regularization can be easily applied to this defense. We also drop the use of

different types of adversarial examples and different model structures used to improve the effect of adversarial training. We select two attack scenarios, the vanilla attack scenario and the ensemble-pattern attack scenario in our experiment.
The parameters in the triplet loss function are set to be $\lambda_1 = 0.3, \lambda_2 = 1, \alpha = 2.0$, and the results are shown in Table 12-13. We use three different attack methods, i.e., FGSM, Deepfool, and C&W, to test the effect of the defense.

Models	Inception-v3			ResNet-v2-101			Inception-ResNet-v2			Ens-adv-Inception-ResNet-v2		
	Ori	Rand	Rand+ Tri. Reg	Ori	Rand	Rand+ Tri. Reg	Ori	Rand	Rand+ Tri. Reg	Ori	Rand	Rand+ Tri. Reg
FGSM	66.8	36.2	**30.5**	73.7	28.2	**21.7**	34.7	19.0	**8.3**	15.6	**4.3**	4.6
Deepfool	100.0	1.7	**1.1**	100.0	2.3	**1.5**	100.0	1.8	**0.8**	99.8	0.9	**0.7**
C&W	100.0	3.1	**2.6**	100.0	2.9	**1.2**	99.7	2.3	**1.3**	99.1	1.2	**0.9**

Table 12: Top-1 classification error rate under the vanilla attack scenario. 'Ori' means the original model. 'Rand' means adding some randomization layers. 'Tri. Reg' means applying our triplet regularization.

Models	Inception-v3			ResNet-v2-101			Inception-ResNet-v2			Ens-adv-Inception-ResNet-v2		
	Ori	Rand	Rand+ Tri. Reg	Ori	Rand	Rand+ Tri. Reg	Ori	Rand	Rand+ Tri. Reg	Ori	Rand	Rand + Tri. Reg
FGSM	62.7	58.8	**37.3**	60.8	55.1	**45.1**	28.5	25.7	**23.5**	13.8	11.1	**8.9**
Deepfool	99.4	18.7	**17.7**	99.1	**19.5**	22.2	99.1	**30.6**	41.6	98.4	**6.5**	9.0
C&W	99.4	37.1	**23.1**	99.0	25.7	**20.0**	98.4	31.7	**21.7**	94.2	13.9	**7.4**

Table 13: Top-1 classification error rate under the ensemble-pattern attack scenario. Similar to vanilla attack and single-pattern attack scenarios, we see that randomization layers increase the accuracy under all attacks and networks. This clearly demonstrates the effectiveness of the proposed randomization method on defending against adversarial examples, even under this very strong attack scenario. 'Ori' means the original model. 'Rand' means adding some randomization layers. 'Tri. Reg' means applying our triplet regularization.

2.3 Defense-GAN

Defense-GAN used Vanishing & Exploding Gradients. This defense do not affect the training and testing of classifier. The adversarial training process was even used in the original paper and got a well performance. So we apply our triplet regularization under the same setting of parameters and the result shows that the new loss can improve the robustness based on the original defense.
The parameters in the triplet loss function are set to be $\lambda_1 = 0.3, \lambda_2 = 0.5, \alpha = 2.0$, and Defense-GAN has $L = 200$ and $R = 10$. The results are shown in Table 14 and the model structures are listed in Table 15.

Attack	Model	No Attack	No Defense	Defense-GAN-Rec	Adv. Train	AT^{2}L	Defense-GAN-Rec + Tri. Reg
FGSM	A	0.3	88.3	1.2	34.9	23.6	**1.1**
	B	3.8	97.8	4.4	94.0	96.8	**0.7**
	C	0.4	67.9	1.1	21.4	11.9	**0.8**
	D	0.8	96.2	2.0	26.8	15.2	**1.6**
C&W	A	0.3	85.9	**1.1**	92.3	96.5	1.4
	B	3.8	96.8	8.4	72.0	70.7	**4.7**
	C	0.4	87.4	**1.1**	96.9	93.7	**1.1**
	D	0.8	96.8	1.7	99.0	87.7	**1.4**

Table 14: Classification error rates of different classifier models using various defense strategies on the MNIST datasets, under FGSM and C&W known type of attacks. 'Adv. Train' mean a traditional adversarial training process. 'Tri. Reg' means applying our triplet regularization.

A	B	C	D	Generator	Discriminator
Conv(64,5x5,1)	Dropout(0.2)	Conv(128,3x3,1)	FC(200)	FC(4096)	Conv(64,5x5,2)
ReLU	Conv(64,8x8,2)	ReLU	ReLU	ReLU	LeakyReLU(0.2)
Conv(64,5x5,2)	ReLU	Conv(64,3x3,2)	Dropout(0.5)	ConvT(256,5x5,1)	Conv(128,5x5,2)
ReLU	Conv(128,6x6,2)	ReLU	FC(200)	ReLU	LeakyReLU(0.2)
Dropout(0.25)	ReLU	Dropout(0.25)	ReLU	ConvT(128,5x5,1)	Conv(256,5x5,2)
FC(128)	Conv(128,5x5,1)	FC(128)	Dropout(0.5)	ReLU	LeakyReLU(0.2)
ReLU	ReLU	ReLU	FC(10)+Softmax	ConvT(1,5x5,1)	FC(1)
Dropout(0.5)	Dropout(0.5)	Dropout(0.5)		Sigmoid	Sigmoid
FC(10)+Softmax	FC(10)+Softmax	FC(10)+Softmax			

Table 15: Neural network architectures used for classifiers, substitute models and GANs.