# GTAT: Adversarial Training with Generated Triplets

Baoli Wang[1,2], Xinxin Fan[1], Quanliang Jing[1,2], Yueyang Su[1,2], Jingwei Li[3], Jingping Bi[1]
*Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China*[1]
*University of Chinese Academy of Sciences, China*[2]
*Department of Computer Science and Engineerting University of Buffalo, SUNY, US*[3]
{wangbaoli, fanxinxin, jingquanliang, suyueyang19b, bjp}@ict.ac.cn, jli379@buffalo.edu

***Abstract*—To circumvent the grave problem of present adversarial training methods, i.e. distortion of classification surface, we in this paper propose a generated Triplet-based adversarial training method-GTAT, in which a Generator generates a semi-hard Triplet by design, rather than directly invoking the existing clean examples and adversarial examples. Through this kind of generated semi-hard Triplet constraint, GTAT can reshape the classification boundaries appropriately across various classes, arising from two-facet synergies: i) pull the intra-class examples together with tight distances; and ii) push away the inter-class examples with broad distances. This synergy will simplify and broaden the classification surfaces across different classes. Extensive experiments on the popular MNIST and CIFAR-10 datasets show that our proposed GTAT significantly outperforms other state-of-the-art adversarial training methods. We believe GTAT opens a door for the adversarial training from a new horizon of rationally generating semi-hard Triplet-satisfied adversarial training (retraining) examples, instead of straightly performing retraining on the generated adversarial examples and existing clean examples, or on the generated adversarial examples only.**

***Index Terms*—adversarial training, adversarial attack, triplet, adversarial robustness**

## I. Introduction

Adversarial attack has attracted much attention in academics and industrial community, arising from deliberately-crafted perturbation on the original/clean examples with human-imperceptible [28]. To date, a serial of representative adversarial attacks has been figured out from several perspectives, such as gradient-based FGSM [6], BIM [10], RFGSM [23], MIM [4], optimization-based L-BFGS [22], CW [2], DeepFool [15], and Generated Adversarial Network (GAN)-based AdvGAN [25], AdvGAN++ [7], GAP [18], AdvCGAN [24]. Accompanied with the adversarial attacks, a line of research work focusing on the robustness is proposed to defend various adversarial attacks like an arm race. At present, the proposed defense countermeasures can be divided into gradient masking [16], feature denoising [26], network distillation [17], data compression [5] [8], GAN-based detector [12], [21] and adversarial training. Nevertheless, recent studies witnessed an upset fact that several more-powerful adversarial attacks, e.g. CW [2], PGD [13], BPDA [1], had broken a suite of allegedly robust defenses, only render adversarial training somewhat resistant, i.e. leverage the adversarial examples to independently retrain the target model, or jointly retrain together with original examples.

In our work, aiming at padding the gap of overfitting and generalization of adversarial training, we concentrate on how to reshape the classification surface to enhance the robustness through mapping the clean examples and adversarial examples into a vector space to moderately harness the intra/inter-classification distance. Prior that, we first exhibit the essential spirit of adversarial training referring to previous study [13] (Fig. 1 (a)-(c)), along with our core-intent to achieve an ideal adversarial training fashion (Fig. 1 (d)). Fig. 1 (a) shows that for clean examples, the neural network model is easy to learn an accurate classification surface. However, when facing the adversarial examples, the previously-learned classification surface is error-prone as shown in Fig. 1 (b), wherein the rectangular box centered on a certain side sample will be perturbed to walk out of the previously-learned classification bounds, such as the part marked by the red five-pointed stars. To defend the adversarial attacks, Fig. 1 (c) shows the traditional adversarial training method can reshape the classification surface by adding adversarial examples into the training dataset, as a result, a renewed classification surface is generated to improve the robustness to some extent. However, one severe problem will be accompanied simultaneously if straightly conflating the adversarial examples with the original ones, namely this newborn classification surface is too complicated, and easily cause the overfitting phenomenon [19]. Thereby, an ideal solution, as depicted in Fig. 1 (d), is to appropriately thin the classification surface with three basic purposes: i) simplify the fitting functions to promote the generalization capability of DL models; ii) enlarge the inter-classification distance to address overfitting problem; and iii) draw back the adversarial examples into the correct classification from the misclassified clusters to improve the accuracy.

Recently, Madry et. al. [13] used a saddle point (outer-min-inner-max) formulation to capture the notion of security against adversarial attacks in a principled manner, and performed adversarial training in consideration of such ”outer-min-inner-max” constraint through directly using the adversarial examples produced by PGD-K attacks. This method at present gains the state-of-the-art against those first-order

This work has been supported by the National Natural Science Foundation of China under Grant No.: 62077044, 61702470, 62002343. Jingping Bi (bjp@ict.ac.cn) is the corresponding author.

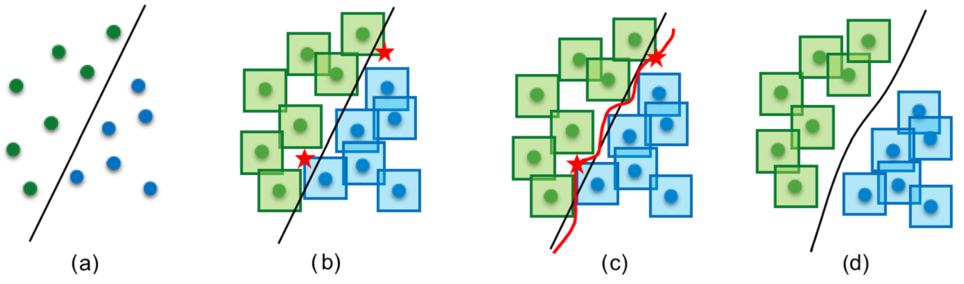

Fig. 1: A conceptual illustration of adversarial training methods. (a) A set of points that can be easily separated by a simple (in this case, linear) decision boundary. (b) The simple decision boundary does not separate the $l_\infty$-balls (here, squares) around the data points, hence there are adversarial examples (the red stars) that will be misclassified. (c) The points with $l_\infty$-balls can be separated by a more complicated decision boundary. (d) The points with $l_\infty$-balls can be separated by a simple (compared to the one in (c) decision boundary, the distance within the category is closer and the distance between the categories is larger.

adversarial attacks, and well matches the description of Fig. 1 (c). In what follows, scholars strives to make the adversarial training reach the ideal status of Fig. 1 (d).

Since the PGD adversarial training method only uses adversarial examples as the training dataset, it is more suitable to train the model from scratch rather than the already trained model. Normally, an intuitive and straightforward improvement is from the retraining operation using the blend dataset of the original clean examples and adversarial examples, which we here call M-PGD as ALP [9] does. Base on M-PGD, ALP [9] performed adversarial training through adding an logit loss to constrain the distance between clean examples and PGD-K adversarial examples. Differently, $AT^2L$ [11] adversarial training added a Triplet loss to constrain the distance between intra-class and inter-class. Instead of randomly choosing negative example like $AT^2L$ does, TLA [14] uses the hardest one from current mini-batch as the negative example in Triplet-tuple. All above methods can indeed improve the efficiency of adversarial training to some extent, however, their attention is mainly paid to the additional loss function to optimize the object function, but ignores whether the PGD adversarial examples for retraining are suitable or not to generate new and moderate classification surfaces.

To this end, we propose a new adversarial training framework GTAT (Generated Triplet Adversarial Training), through restrictively generating adversarial examples using GAN with the requirement of meeting the semi-hard Triplet constraint at the same time, in this way, the newborn classification surface would become more attack resilient and general to various adversarial attacks. In a nutshell, our main contributions can be summarized below:

- First, we provide in-depth analytics and exploration on the principle and working mechanisms behind adversarial training, and point out the rooted-caused vulnerabilities of the present studies. To circumvent this gap, an ideal classification surface (Fig. 1 (d)) that the retrained models should fit is advocated as the retraining goal for the following adversarial training methods.
- Second, we propose a generated Triplet adversarial training framework GTAT to achieve the robustness and generalization against diversities of adversarial attacks. In GTAT, we utilize the GAN's Generator to accommodate the constraint space of Triplet into a semi-hard state with the purpose of reshaping the classification surface appropriately.
- Finally, we validate our GTAT in terms of robustness and generalization against diverse adversarial attacks on two commonly-used datasets MNIST and CIFAR-10. Compared to the state-of-the-art baselines, multi-facet experimental results show our GTAT significantly has a superior performance on classification accuracy and generalization capability.

## II. Related work

### A. Adversarial Training

The conception of adversarial examples and the idea of adding adversarial examples to the training set for retraining to improve the robustness were first introduced in L-BFGS [22]. Madry et al. [13] proposed to train the target network using the multi-step PGD adversarial examples, it achieves the state-of-the-art robustness levels against $l_\infty$ attacks on MNIST and CIFAR-10 datasets. Since then, the adversarial training fashion has become the foundation to enhance the robustness to date, replacing other gradient-masking approaches. Accompanied with the development of various adversarial attacks, some adversarial training methods, on the basis of inheriting PGD adversarial training, additionally introduce extra-loss modules to leash the objective loss function, such as M-PGD, ALP [9], $AT^2L$ [11], TLA [14], by which the adversarial training performance can be promoted to some extent. To be mindful, not all target model are suitable to execute the adversarial training, as stated in the work [13], the authors pointed out that when the capacity of target models was weak or insufficient, the adversarial training would seriously weaken the performance, this is because the presence of adversarial examples makes the decision boundary problem more complicated.

Through bringing in the distance-metric intent, $AT^2L$ [11] leveraged the Triplet loss to implement adversarial training, it can validly smooth the classification boundary. Nevertheless, given the restriction on the generation of adversarial examples by the $p$-norm constraint, it is usually difficult to obtain hard Triplet examples through resorting to the existing examples in the current mini-batch, the adversarial examples and the original clean examples. Differently from the current research, our work employs a Generator $G$ to enforce the three-tuple (anchor/positive/negative) examples into a semi-hard Triplet, by which the retrained target model can be enhanced to improve its robustness.

## B. *Triplet Loss*

Triplet loss is a loss function for ML algorithms wherein a baseline (anchor) input is compared to a positive (truthy) input and a negative (falsy) input [3] [20]. Given a three-tuple Triplet $(x^a, x^p, x^n)$, $(x^a, x^p)$ is referred as a positive (relevant) pair and $(x^a, x^n)$ as a negative (irrelevant) pair. According to the pairwise distance among the three tuples, Triplet can be divided as three assortments: i) easy Triplet, $d(x^a, x^p) + m < d(x^a, x^n)$; ii) hard Triplet, $d(x^a, x^n) < d(x^a, x^p)$; iii) semi-hard Triplet, $d(x^a, x^p) < d(x^a, x^n) < d(x^a, x^p) + m$. Here $m$ is a margin value. The original objective function of Triplet loss can be formulated as follows:

$$\frac{1}{N}\sum_{i=1}^{N} \max\{||f(X_i^a) - f(X_i^p)|| - ||f(X_i^a) - f(X_i^n)|| + m, 0\}. \tag{1}$$

It is worth noting that for standard training, the difficulty of using Triplet loss lies in how to mine more hard-type Triplet, but for adversarial training, the attack characteristics of adversarial examples often make the formed Triples too hard, which make it difficult to train the model to gain revenue. Therefore, in the adversarial training scenario, our goal is to find semi-hard type Triplets to improve the adversarial training effect with an ideal output status as shown in Fig. 1 (d).

# III. Methodology

## A. *Problem Definition*

The adversarial training problem is typically cast as the following "outer-min-inner-max" optimal function to achieve the significant robustness (class boundary):

$$\min_{\theta} \sum_{i} \max_{\delta \in \Delta} \ell\left(f_{\theta}\left(\boldsymbol{x}_i + \delta\right), y_i\right), \tag{2}$$

where $\Delta = \{\delta : ||\delta||_{\infty} \leq \epsilon\}$ for some $\epsilon > 0$. The procedure of adversarial training is firstly to use adversarial attacks to approximate the inner maximization over $\Delta$, subsequently followed by some variation of gradient descent on the model parameters $\theta$.

The core operation of our method is at the moment that after the adversarial examples are generated and before the model parameters are updated, we will first send the generated adversarial examples to a Generator for processing, and then as input into the model to update the parameters, the process can be formulated as follows:

$$\max_{\delta \in \Delta} \sum_{i} \ell\left(f_{\theta}\left(\boldsymbol{x}_i + \delta\right), y_i\right), \tag{3}$$

$$\min_{\theta} \sum_{i} \ell\left(G\left(f_{\theta}\left(\boldsymbol{x}_i + \delta\right)\right), y_i\right), \tag{4}$$

where $G$ is a Generator, which takes the feature of the adversarial example $(x + \delta)$ extracted by the target model $f_\theta$ as its input, and outputs the modified feature that becomes more suitable for updating the parameters of the target model under Triplet standard, by which the newborn classification surfaces can be fitted appropriately to leash the inter/intra distances among clean (anchor), adversarial (positive) and non-identical label (negative) examples even with the presence of sophisticated adversarial attacks, e.g. CW or PGD.

## B. *Overall Framework*

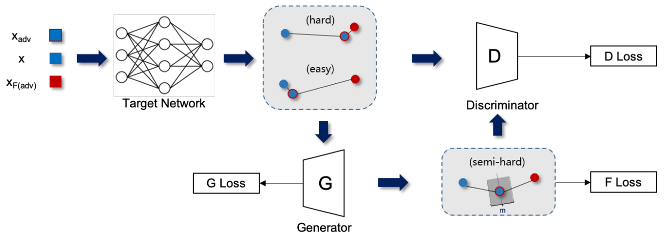

Fig. 2: Overall Architecture of GTAT.

As shown in Fig. 2, the architecture of our proposed framework GTAT mainly involves three core-components: i) target network $F$, it is the target model; ii) Generator $G$, it realizes the function that generating appropriate retraining examples to harness the adversarial training fashion towards the correct function fitting; iii) Discriminator $D$, it guides the loss function to identify correct labels during the course of adversarial training. Here it is worth noting that both the Generator $G$ and Discriminator $D$ are no longer the two basic parts in the regular GANs, but endowed with specific-functions for adversarial training purpose. Our method is inspired by the article HTG [27], which mainly focuses on how to construct more difficult Triplets on regular face recognition tasks, but different from it, our GTAT aims at reshaping the generated adversarial examples to make it more suitable for model parameter updates.

As defined previously, our designed generated Triplet is comprised of three elements i.e. anchor example, positive example and negative example. As described in Fig. 2, the blue rectangular box represents clean examples $x$ in the training data set, and the blue rectangular box with red border represents the adversarial examples $x_{adv}$ corresponding to the clean examples $x$, and the red rectangular box represents another group of clean training examples $x_{F(adv)}$ with different class from the clean examples $x$, but with same class into which the adversarial examples $x_{adv}$ are misclassified by the target network $F$. The three colored circular dots represent the features of the corresponding rectangular-box examples extracted by the target network $F$ with the further handling of reshaping and accommodation by the generator $G$, which can be denoted as feat$_x$, feat$_{adv}$, feat$_{F(adv)}$ respectively. As defined previously, the relative position of the three-tuple dots indicates three concrete types of Triplet: easy, semi-hard and hard.

As exhibited in Fig. 2, the holistic workflow can be presented as follows: i) first, given a part of clean examples as anchor primitive in the training dataset, we employ the commonly-used adversarial attack (e.g. PGD-$K$) to generate corresponding adversarial examples; ii) second, sample another set of clean examples belonging to the same categories into which the generated adversarial examples are misclassified as negative primitive, by which the generated Triplet is established; and iii) finally, send the Triplet examples to the target network $F$, and obtain its corresponding characteristics $< \text{feat}_{adv}, \text{feat}_{x}, \text{feat}_{F(adv)} >$. In consideration of coordinating the generated Triplet setting, we alternatively denote $x_{adv}$, $x$, $x_{F(adv)}$ as $x_a$, $x_p$, $x_n$ respectively. It is worth noting that, instead of directly calculating Triplet loss, we will first send the adversarial examples' features, which are extracted by the target model $F$, to the Generator $G$ for reshaping manipulation. Here the Generator $G$'s mission lies in creating the so-called semi-hard Triplet, i.e. decline the distance between the feature from misclassified class $\text{feat}_{F(adv)}$, and the feature from adversarial example $\text{feat}_{adv}$, and inversely enlarge the distance between the adversarial example feature $\text{feat}_{adv}$ and its corresponding clean sample feature $\text{feat}_{x}$. Then, calculate Triplet and classification loss. To guarantee the capability of reshaping of the Generator $G$, we also introduced a Discriminator $D$ w.r.t two main functions: i) ensure the quality of the generated Triplet after being reshaped by the Generator $G$; and ii) retain their labels of the clean and generated examples unchanged after the Generator $G$ reshapes the feature, which leashes the retraining model towards correct direction.

### C. Target Network F

The function of target network $F$ is to extract the input images and map them into a feature space, the outputs are their final transformed features (a logit vector or final label). As mentioned previously, we bring in two radical loss functions to tutor the generated Triplet adversarial training: classification and generated Triplet loss functions.

Firstly, the classification loss function is used to ensure the accuracy of the model's recognition for each categorical example, and it can be formulated as follow:

$$\begin{aligned}\mathcal{L}_{F,cls} = \mathcal{L}_{ce}(F(x_a), l_a) + \mathcal{L}_{ce}(F(x_p), l_p) \\ +\mathcal{L}_{ce}(F(x_n), l_n),\end{aligned} \tag{5}$$

where $\mathcal{L}_{ce}$ is the cross-entropy loss function, $F$ is the target network, $l$ is the corresponding label.

Secondly, a generated Triplet loss function is introduced to optimize and accommodate the intra-class distance and inter-class distance of the retraining data distribution. It is worth noting that, unlike the previous Triplet loss-oriented adversarial training method, we do not directly construct the Triples from the clean training examples or mix with adversarial examples, but rather send the constructed Triples first to the Generator $G$ with the endeavor of optimizing and reshaping, enforcing it more likely to be a semi-hard type as much as possible. Thus, we refer to this kind of reshape-targeted Triplet with Generator $G$ as generated Triplet Loss:

$$\begin{aligned}\mathcal{L}_{F,tri} = [d(G(F(x_a)), G(F(x_p))) \\ -d(G(F(x_a)), G(F(x_n))) + m]_+,\end{aligned} \tag{6}$$

where $d$ is a measurement of distance, $m$ is the margin.

Finally, with co-constraints on classification accuracy and three-tuple examples for adversarial training, the total loss function of Target Network $F$ can be expressed as:

$$\mathcal{L}_F = \mathcal{L}_{F,cls} + \alpha * \mathcal{L}_{F,tri}, \tag{7}$$

where $\alpha$ denotes the weight on Triplet-specific loss.

### D. Semi-Hard Triplet Generator G

For an input $x$, let us consider a semi-hard example Generator G that generates a new adversarial example $G(F(x)) \in \mathcal{R}^L$ by manipulating the feature (logit) representation $F(x)$ of an input $x$. Specifically, the Generator $G$ leashes the input examples as such that pushing the feature representation vectors from the same category apart, while inversely pulling the feature representation vectors from different categories close.

Formulaically, we can minimize the following adversarial generated Triplet loss to train the Generator $G$,

$$\begin{aligned}\mathcal{L}_{G,tri} = [d(G(F(x_a)), G(F(x_n))) \\ -d(G(F(x_a)), G(F(x_p))) + m]_+.\end{aligned} \tag{8}$$

Meantime, the Generator $G$ ought to preserve the de facto label of its input features, therefore, we need another constraint primitive on the Generator $G$, i.e. a Discriminator $D$. We expect that the our new designed Triplet, which is reshaped by the Generator $G$, is still more in line with the normal Triplet distribution, but not too outrageous.

Thus, we define the following loss function to enforce this label preservation assumption, it can be formulated as:

$$\begin{aligned}\mathcal{L}_{G,cls} = \frac{1}{3}(\mathcal{L}_{ce}(D(G(F(x_a))), l_a) \\ +\mathcal{L}_{ce}(D(G(F(p))), l_p) \\ +\mathcal{L}_{ce}(D(G(F(n))), l_n)).\end{aligned} \tag{9}$$

Clearly, these two loss implications $\mathcal{L}_{F,tri}$ and $\mathcal{L}_{G,tri}$ constitute a pair of adversarial loss functions. Compared with the previous adversarial training loss functions, $F$ is trained through semi-hard Triplets generated by Generator $G$ via pulling the positive pair closer and pushing the negative pair apart to meet the margin $m$. Through this semi-hard Triplet-specific adversarial training, the reborn target network $F$ will become resilient and general against various adversarial attacks. Thereby, the overall loss function for the Generator $G$ can be expressed as:

$$\mathcal{L}_G = \mathcal{L}_{G,cls} + \beta * \mathcal{L}_{G,tri}, \tag{10}$$

where $\beta$ is the weight of $\mathcal{L}_{G,tri}$.

### E. Multi-Category Discriminator D

Obviously, merely depending on the above adversarial mechanism is insufficient to train a generalized and attack resilient Generator $G$, this is because it can arbitrarily manipulate feature representation (logit output) mapped by Target Network $F$, without correct-guidance constrain. In an extreme case, for example, the Generator $G$ could simply output random vectors to achieve a lower value of $\mathcal{L}_{G,tri}$, which is completely useless for training a target network $F$. Thus, we need properly harness the Generator $G$ through resorting to other factors, e.g., the Discriminator $D$, along with the requirement that its logit output should not change the label of its input features $F(x)$.

Given the fact that some generated adversarial examples exist, the Discriminator $D$ ought to assign one more class, i.e. $(K+1)$, to categorize the adversarial examples' feature vectors, where the $K$ categories represent real classes of examples, and the last one denotes a fake class. For the three-tuple Triplet $(a,p,n)$ and the corresponding labels $(l_a,l_p,l_n)$, we have $l_a = l_p$ for the positive pair and $l_a \neq l_n$ for the negative pair. Then, we minimize the following loss function to train the Discriminator $D$

$$\mathcal{L}_D = \mathcal{L}_{D,real} + \gamma * \mathcal{L}_{D,fake}. \tag{11}$$

The first term enforces the Discriminator $D$ to correctly classify the feature vectors, which can be expressed as:

$$\begin{aligned}\mathcal{L}_{D,real} = \frac{1}{3}(&\mathcal{L}_{sm}(D(F(a)),l_a)\\ &+\mathcal{L}_{sm}(D(F(p)),l_p)\\ &+\mathcal{L}_{sm}(D(F(n)),l_n)).\end{aligned} \tag{12}$$

Meanwhile, the second term enables the Discriminator $D$ to distinguish generated features from the real ones, which can be expressed as:

$$\begin{aligned}\mathcal{L}_{D,fake} = \frac{1}{3}(&\mathcal{L}_{\text{sm}}\left(D(G(F(a))),l_{\text{fake}}\right)\\ &+\mathcal{L}_{sm}(D(G(F(p))),l_{fake})\\ &+\mathcal{L}_{sm}(D(G(F(n))),l_{fake})),\end{aligned} \tag{13}$$

where $l_{fake}$ represents the fake class.

## IV. Experiment Evaluation

### A. *Experiment settings*

**Datasets and Network Models.** We comparatively evaluate and analyze GTAT with other baselines on two popular datasets: MNIST and CIFAR-10. For MNIST dataset, we use LeNet and a variant as network models to launch white/black-box attacks. For CIFAR-10 dataset, we use VGG19 as source network model and ResNet as target network model to implement white/black-box attacks. It is worth noting that, on the one hand, due to the limitation of experimental resources, we only carried out experiments on network models with small capacity. On the other hand, this paper focuses on making fine-tune, not training from scratch, for the trained networks with normal classification ability. Therefore, experiments on network model with large capacity will be done in future work.

**Attack and Adversarial Training Methods.** For each dataset, we conducted a comprehensive performance evaluation to validate the effectiveness of our proposed GTAT both in black-box and white-box attack scenarios under a variety of representative adversarial attacks (FGSM, BIM, PGD-K, CW), and we also compared five state-of-the-art adversarial training methods (PGD, M-PGD, ALP, $AT^2L$, TLA). We

TABLE I: Adversarial training results under white-box and black-box attack on MNIST dataset. The best results of each column are in **bold and underline** and the second is **bold**.

| Adv. Train | Accuracy (%) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Clean | Attack | | | | | | | |
| | | FGSM | BIM | PGD-20 | PGD-40 | PGD-100 | PGD-500 | PGD-1000 | CW-200 |
| (a) White-box Attack | | | | | | | | | |
| No Train | **99.37** | 11.04 | 0.00 | 1.09 | 0.01 | 0.00 | 0.00 | 0.00 | 4.24 |
| PGD | 96.89 | 87.46 | 90.26 | 90.52 | 83.01 | 79.17 | 78.79 | 78.84 | 56.69 |
| M-PGD | 98.07 | 90.92 | 92.57 | 93.87 | 86.83 | 82.70 | 81.88 | 81.89 | 58.13 |
| ALP | 98.18 | 89.45 | 90.78 | 91.41 | 86.71 | 84.24 | 84.31 | 84.24 | 56.80 |
| $AT^2L$ | 97.35 | 92.36 | 93.70 | 94.01 | 89.70 | **87.25** | **86.91** | **86.95** | **59.57** |
| TLA | 96.95 | **92.90** | **93.90** | **95.09** | **89.76** | 86.45 | 85.90 | 85.89 | 52.15 |
| GTAT | **98.68** | **93.33** | **94.77** | **95.14** | **91.64** | **89.52** | **89.36** | **89.29** | **65.24** |
| (b) Black-box Attack | | | | | | | | | |
| No Train | **99.37** | 13.94 | 23.05 | 11.48 | 2.37 | 1.24 | 1.16 | 1.22 | 55.76 |
| PGD | 96.89 | 91.08 | 94.65 | 95.49 | 94.44 | 94.31 | 94.55 | 94.31 | 80.54 |
| M-PGD | 98.07 | 93.07 | 96.67 | **97.51** | 96.49 | 96.12 | **96.11** | **96.14** | 86.86 |
| ALP | 98.18 | 91.86 | 94.78 | 95.42 | 94.67 | 94.56 | 94.50 | 94.63 | **88.94** |
| $AT^2L$ | 97.35 | **93.88** | **96.69** | 97.45 | **96.73** | **96.63** | 95.62 | 95.59 | 88.83 |
| TLA | 96.95 | 93.28 | 96.24 | 97.00 | 95.77 | 96.37 | 94.75 | 95.02 | 87.19 |
| GTAT | **98.68** | **94.37** | **97.07** | **97.77** | **97.01** | **97.02** | **96.91** | **96.87** | **90.89** |

conduct all of our experiments using PyTorch v1.7.0 on a single Nvidia 2080 Ti and a CPU memory of 128GB with 24 cores. The size of mini-batch for each experiment is unified to 512 in the training phase and 1024 in the testing phase. We conduct 100 epochs of adversarial training for CIFAR-10 and 30 epochs for MNIST respectively, and then evaluate the defense effectiveness of each method under both white-box and black-box adversarial attacks. The learning rate is 0.01 for MNIST dataset and 0.1 for CIFAR-10 dataset, and for the epoch of 50 and 80, we reduce it to 0.01 and 0.001.

**Evaluation Metrics.** To evaluate the model's adversarial robustness, we straightly use the ratio of classification accuracy (RCA) to reflect the performance:

$$RCA = N_{correct}/N_{total}, \tag{14}$$

where $N_{total}$ is the total number of the adversarial examples in testing dataset, and $N_{correct}$ refers to the number of the adversarial examples which are classified correctly into their true-label classes by the target model.

Furthermore, we also introduce a distance matrix to microscopically evaluate the performance using the statistics data on the intra/inter-class distances of the examples.

$$Distance_{i,j} = \frac{1}{N}\sum_{n=1}^{N}||f(x_i^n) - f(x_j^{n'})||_2, \tag{15}$$

where $f$ is the feature extractor, $i$ and $j$ are the $i^{th}$ and $j^{th}$ class for clean and adversarial examples. $N$ is the number of sampled examples. $||*||_2$ is the $l_2$-norm for $*$.

### B. *Performance Evaluation*

*1) Under White-Box Adversarial Attacks:* We first exhibit the performance under white-box adversarial attacks, i.e. the adversary has the knowledge information on target model, such as its network structure, model parameters, loss function, etc. Table I (a) shows the robustness results with various adversarial training methods on the MNIST dataset under four white-box adversarial attacks (FGSM, BIM, PGD, CW) and one clean status, wherein the PGD is further assigned with

TABLE II: Adversarial training results under white-box and black-box attack on CIFAR-10 dataset. The best results of each column are in **bold and underline** and the second is **bold**.

| Adv. Train | Accuracy (%) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Clean | Attack | | | | | | |
| | | FGSM | BIM | PGD-5 | PGD-7 | PGD-20 | PGD-50 | CW-20 |
| a) White-box Attack | | | | | | | | |
| No Train | **95.40** | 49.73 | 0.32 | 1.04 | 0.01 | 0.01 | 0.01 | 3.61 |
| PGD | 85.60 | 56.09 | 46.72 | 54.75 | 50.65 | 46.56 | 46.21 | 54.60 |
| M-PGD | 87.95 | **59.01** | 48.02 | 57.37 | 51.53 | 46.45 | 45.78 | 56.00 |
| ALP | 86.44 | 58.65 | **48.48** | **56.59** | **52.04** | **47.25** | **47.05** | **56.17** |
| $AT^2L$ | 87.33 | 57.92 | 47.88 | 56.46 | 50.93 | 46.58 | 46.22 | 55.00 |
| TLA | 87.66 | 56.29 | 47.25 | 56.05 | 50.74 | 46.03 | 45.57 | 55.16 |
| GTAT | **89.10** | **62.20** | **52.68** | **60.29** | **56.10** | **54.61** | **54.24** | **62.46** |
| b) Black-box Attack | | | | | | | | |
| No Train | **95.40** | 59.61 | 43.76 | 46.62 | 52.23 | 46.58 | 47.41 | 80.67 |
| PGD | 85.60 | 84.46 | 85.17 | 85.06 | 85.28 | 85.35 | 85.04 | 85.86 |
| M-PGD | 87.95 | **86.13** | 86.42 | 86.22 | 87.13 | **86.68** | 86.04 | **87.69** |
| ALP | 86.44 | 84.72 | **86.47** | **86.58** | **86.51** | 86.61 | **86.54** | 87.18 |
| $AT^2L$ | 87.33 | 85.47 | 86.39 | 85.66 | 85.46 | 85.32 | 85.40 | 86.08 |
| TLA | 87.66 | 83.12 | 83.53 | 83.83 | 83.80 | 83.63 | 84.74 | 84.33 |
| GTAT | **89.10** | **86.16** | **89.48** | **88.60** | **88.64** | **88.46** | **88.37** | **89.62** |

different-level attack strengths, i.e. PGD-20/40/100/500/1000. From the experimental result, we can find several observations as follows.

**Adversarial attacks can completely defeat the network model that does not perform adversarial training.** For example, the accuracy severely deteriorates to 1.09%, 0.01%, even 0% under the PGD attacks at the steps of 20/40/100/500/1000. These experimental results witness the strength of these representative adversarial attacks, especially as the perturbation enlarges gradually in PGD, and a 200-iteration optimization in CW.

**Network models equipped with adversarial training strategy have achieved obvious robustness improvement.** In detail, PGD adversarial training can achieve the accuracy interval [96.89%, 56.69%] under the eight adversarial attacks, and M-PGD, ALP, $AT^2L$, TLA realize the accuracy intervals [58.13%, 98.07%], [56.80%, 98.18%], [59.57%, 97.35%], [52.15%, 96.95%] respectively. Compared to these baselines, our proposed GTAT can dramatically promote the overall accuracy intervals [65.24%, 98.68%].

**GTAT significantly outperforms other baselines.** Our experiment data have attested that when the PGD adversarial attack exceeds 100 iterations, its attack capability no longer increases. As is well known, the CW attack so far has stronger attack capability than others, and our experimental results also exhibit this fact in white-box scenarios. Against this powerful attack, our GTAT achieves an exciting accuracy of 65.24%, which is far superior to other baseline methods, and gains 5.27% improvement than the best baseline $AT^2L$. In addition, our proposed GTAT also behaves best given other attack cases. This is because of that our GTAT method better optimizes the intra-class and inter-class distance of the examples, and we will do a more detailed qualitative and quantitative analysis later.

Table II (a) shows the adversarial example recognition accuracy of each adversarial training method on the CIFAR-10 data set under different types and intensities of white box attacks. Compared with the MNIST dataset, we can see that the network model after adversarial training has an obvious reduction in the classification accuracy on clean examples. This is mainly due to the fact that the dataset of the CIFAR-10 has become more complex in terms of resolution and the number of channels compared to the MNIST dataset, which appeals higher requirements on the capabilities of the model. Nevertheless, $AT^2L$ and our GTAT have obtained significantly better classification accuracy compared to other baselines, and our GTAT still outperforms $AT^2L$ as the best defense against such attacks.

*2) Under Black-Box Adversarial Attacks:* Apart from the performance validation and analytics under white-box attack scenario, another group of experiments under black-box attacks is provided as well. Table I (b) and Table II (b) respectively show the classification accuracy in black-box attack scenario under different types of adversarial attacks on the two datasets, also providing that PGD possesses different-level intensities of perturbation to produce adversarial examples. Through the experimental results, we have the following observations.

**Compared to the white-box case, these adversarial attacks perform poorly in black-box scenario.** For the MNIST dataset, the adversarial training methods PGD and M-PGD carry out accuracy intervals [80.54%, 96.89%] and [86.86%, 98.07%], and ALP, $AT^2L$, TLA accomplish the accuracy intervals [88.94%, 98.07%], [88.83%, 97.35%] and [87.19%, 96.95%]. Even with poor performance of these adversarial attacks in this black-box scenario, our GTAT achieves accuracy interval [90.89%, 98.68%], which significantly outperforms other baselines against these adversarial attacks.

**Compared to the high attack success rate, the well-known strong CW attack achieves a very low attack success rate.** From the experimental results on CIFAR-10 dataset in Table I (b), our GTAT realizes best performance compared to other baselines, while followed by the baseline M-PGD with the second best results. This on the other hand indicates the migration of adversarial attacks between target models is not well. This is because the CW attack pursues the current white-box model to find a minimum disturbance that can make it misclassified. When switching to other models, the effect is often easily compromised.

To sum up, both white-box and black-box experiments verify that our proposed GTAT is a viable and attack-resilient adversarial training fashion to resist several representative attacks like FGSM, BIM, PGD and CW w.r.t. different-level attack strengths. We think the paramount reason lies in the appropriately-reshaping capability over multi-class boundaries through generating semi-hard Triplet constraint retraining examples in terms of anchor, positive and negative primitives.

## *C. Analysis on Intra/Inter-class Distance*

To further validate the effectiveness of our proposed GTAT, both qualitative analysis and quantitative analysis are provided.

**Qualitative Analysis.** Concretely, we respectively sample 500 clean examples from the class/label-1 and class/label-8,

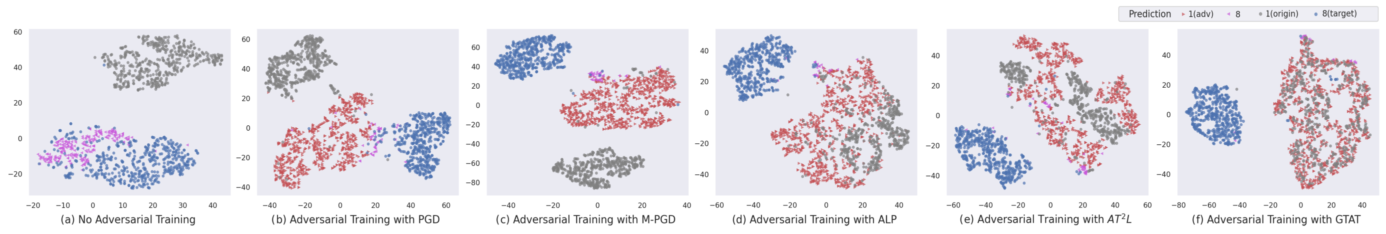

Fig. 3: t-SNE Visualization of adversarial images from the same class which are mistakenly classified to wrong classes (the gray and blue dots represent the randomly sampled clean examples from class-1 and class-8 respectively, the red triangles denote the generated adversarial examples based on gray dots, the left triangles with pink color represent the ones that be classified into class-8 by the target model, and the right triangles with red color represent the ones that be correctly identified back to class-1).

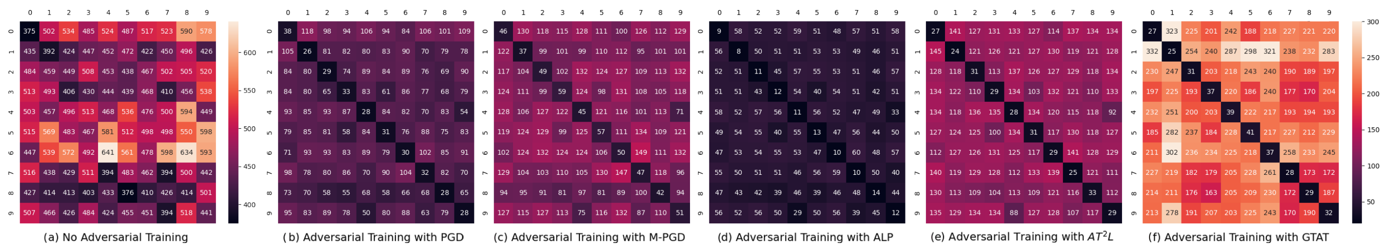

Fig. 4: Heatmap visualization of intra-class and inter-class distance under different training methods (The numbers on the ordinate axis represent clean examples of the corresponding category, and the numbers on the abscissa axis represent the adversarial examples of the corresponding category. The value on the diagonal represents the calculated intra-class distance, and the value on the off-diagonal represents the calculated inter-class distances).

and then sample 1000 adversarial examples from class/label-1 using PGD-100 adversarial attack for each adversarial training method. Finally, we use the popular t-SNE visualization technology to show the distribution of these 2000 examples. Fig. 3 exhibits the distribution of the clean and adversarial examples with different adversarial training methods. The experimental results show that our proposed GTAT can well-fit the adversarial examples and clean examples into a consistent distribution, which purports that our generated semi-hard Triplet constraint can indeed tightly pull the adversarial examples and clean examples with the identical label together, and simultaneously push other categories of examples away.

**Quantitative Analysis.** Fig. 4 shows the statistics of the intra-class distance and inter-class distance under various adversarial training methods. Different colors represent different distance values, the lighter the color is the larger the distance is, and conversely the darker the color is, the smaller the distance is. Observed from the subfigures, we can see that our GTAT yields a salient color-disparity compared to the baselines, this presents it can moderately reshape the boundaries across different classes through pulling the intra-class examples but pushing the inter-class examples, this renders a high attack-resilient space to refitting the classification function.

### *D. Effects on Adversarial Examples with Different Attack Strength*

We perform two groups of experiments, which use smaller learning rate and more steps, to showcase the adaptability to generate semi-hard Triplet for two opposite statuses of weak

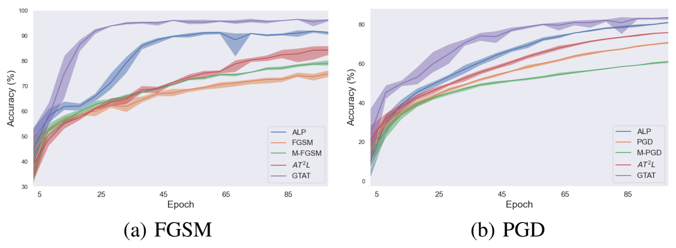

Fig. 5: Training-stage accuracy with adversarial examples.

examples and strong examples during the retraining course. Intuitively, the weak means the examples are generated by those adversarial attacks turned out to be weak, in contrast, the strong implies the examples are yielded by those adversarial attacks attested to be strong. Fig. 5 (a) and Fig. 5 (b) portray the recognition accuracy on the FGSM-generated weak sample and PGD-40-generated strong examples after epoch training. For the sake of clear description of experimental results, we aggregate each 5 epochs as a group and compute the average, maximum and minimum accuracies.

Seen from the two groups of experimental results, we obtain the following observations: i) for both weak examples in Fig. 5 (a) and strong examples in Fig. 5 (b), our proposed GTAT can achieve the best performance compared to other baselines. Also in line with the intuition, the accuracies on weak examples behave better than that on strong examples, arising from different-level strengths of FGSM and PGD-40 attacks; ii) for the weak examples in Fig. 5 (a), our GTAT can culminate the best accuracy swiftly at the 25th epoch, compared to the 55th epoch for ALP, and up to the 95th epoch for FGSM, M-FGSM and $AT^2L$; and iii) for the strong examples in Fig. 5 (b), the convergence speed to reach the stable states becomes slowly for GTAT and ALP, which, on the one hand, indicates the complexity of strong examples, but on the other hand, our GTAT can still escalate to a high-level accuracy as the epoch enlarges gradually.

Through the above analytics, we know that our proposed GTAT can significantly improve the robustness against whatever weak adversarial examples or strong adversarial examples. This also exhibits two insights of our work: i) our designed generated semi-hard Triplet can easily and correctively reshape the class boundaries on weak adversarial examples; and ii) although the complexity of strong adversarial examples slows down the convergence speed, our GTAT can still appropriately relearn the class boundaries by mapping the three-tuple (anchor, positive and negative) examples into a distance-constraint vector space.

## V. Conclusion

We figure out a generated Triplet-based adversarial training method against various adversarial attacks. Concretely, three remarkable contributions are provided. First, we analyze the inherent vulnerabilities that deteriorate the performance of various adversarial training methods from the perspectives of easy and strong adversarial examples. Second, we propose a classification boundary-reshaping idea through resorting to generated semi-hard Triplet constraint with the purposes of simplifying the refitting function and broadening the classification boundaries across different classes. Third, multi-facet experiments are produced to validate the effectiveness of our proposed GTAT. Besides that, we also provide a microscopic observation on the distribution variation of adversarial examples after executing adversarial training, and a fine-grained verification on the boundary-reshaping. These comprehensive experiments show our GTAT significantly outperforms the representative baselines against a serial of while-box and black-box attacks. We believe our work will shed a light on the study of robustness of the deep neural network models.

## References

[1] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International Conference on Machine Learning*. PMLR, 2018, pp. 274–283. 1

[2] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*. IEEE, 2017, pp. 39–57. 1

[3] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," 2010. 3

[4] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9185–9193. 1

[5] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of jpg compression on adversarial images," *arXiv preprint arXiv:1608.00853*, 2016. 1

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 1

[7] S. Jandial, P. Mangla, S. Varshney, and V. Balasubramanian, "Advgan++: Harnessing latent layers for adversary generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0. 1

[8] X. Jia, X. Wei, X. Cao, and H. Foroosh, "Comdefend: An efficient image compression model to defend adversarial examples," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6084–6092. 1

[9] H. Kannan, A. Kurakin, and I. Goodfellow, "Adversarial logit pairing," *arXiv preprint arXiv:1803.06373*, 2018. 2

[10] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017. 1

[11] P. Li, J. Yi, B. Zhou, and L. Zhang, "Improving the robustness of deep neural networks via adversarial training with triplet loss," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. 2

[12] X. Liu and C.-J. Hsieh, "Rob-gan: Generator, discriminator, and adversarial attacker," in *CVPR*, 2019, pp. 11 234–11 243. 1

[13] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017. 1, 2

[14] C. Mao, Z. Zhong, J. Yang, C. Vondrick, and B. Ray, "Metric learning for adversarial robustness," *arXiv preprint arXiv:1909.00900*, 2019. 2

[15] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436. 1

[16] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519. 1

[17] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 582–597. 1

[18] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie, "Generative adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4422–4431. 1

[19] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8093–8104. 1

[20] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823. 3

[21] S. Shen, G. Jin, K. Gao, and Y. Zhang, "Ape-gan: Adversarial perturbation elimination with gan," *arXiv preprint arXiv:1707.05474*, 2017. 1

[22] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013. 1, 2

[23] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *arXiv preprint arXiv:1705.07204*, 2017. 1

[24] B. Wang, X. Fan, Q. Jing, H. Tan, and J. Bi, "Advcgan: An elastic and covert adversarial examples generating framework," in *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*, 2021, pp. 1–8. 1

[25] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song, "Generating adversarial examples with adversarial networks," *arXiv preprint arXiv:1801.02610*, 2018. 1

[26] C. Xie, Y. Wu, L. v. d. Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 501–509. 1

[27] Y. Zhao, Z. Jin, G.-j. Qi, H. Lu, and X.-s. Hua, "An adversarial approach to hard triplet generation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 501–517. 3

[28] Y. Zhou, M. Han, L. Liu, J. He, and X. Gao, "The adversarial attacks threats on computer vision: a survey," in *MASSW*. IEEE, 2019, pp. 25–30. 1