A. Friendly Adversarial Training

For completeness, besides the learning objective by loss value, we also give the learning objective of FAT by class probability. Then, based on the learning objective by loss value, we give the proof of Theorem 1, which theoretically justifies FAT.

A.1. Learning Objective

Case 1 (by loss value, restated). The outer minimization still follows Eq. (3). However, instead of generating $\tilde{x}_i$ via inner maximization, we generate $\tilde{x}_i$ as follows:

$$\tilde{x}_i = \arg\min_{\tilde{x}\in\mathcal{B}_\epsilon[x_i]} \ell(f(\tilde{x}), y_i)$$
$$\text{s.t. } \ell(f(\tilde{x}), y_i) - \min_{y\in\mathcal{Y}} \ell(f(\tilde{x}), y) \geq \rho.$$

Note that the operator $\arg\max$ in Eq. (4) is replaced with $\arg\min$ here, and there is a constraint on the margin of loss values (i.e., the mis-classification confidence).

The constraint firstly ensures $y_i \neq \arg\min_{y\in\mathcal{Y}} \ell(f(\tilde{x}), y)$ or $\tilde{x}$ is mis-classified, and secondly ensures for $\tilde{x}$ the wrong prediction is better than the desired prediction y_i by at least ρ in terms of the loss value. Among all such $\tilde{x}$ satisfying the constraint, we select the one minimizing $\ell(f(\tilde{x}), y_i)$. Namely, we minimize the adversarial loss given that a confident adversarial data has been found. This $\tilde{x}_i$ could be regarded as a friend among the adversaries, which is termed as friendly adversarial data.

Case 2 (by class probability). We redefine the above objective from the loss point of view (above) to the class probability point of view. The objective is still Eq. (3), in which $\ell(f(\tilde{x}), y) = \ell_{\mathrm{B}}(\ell_{\mathrm{L}}(f(\tilde{x})), y)$. Hence, $\ell_{\mathrm{L}}(f(\tilde{x}))$ is an estimate of the class-posterior probability $p(y \mid \tilde{x})$, and for convenience, denote by $p_f(y \mid \tilde{x})$ the y-th element of the vector $\ell_{\mathrm{L}}(f(\tilde{x}))$.

$$\tilde{x}_i = \arg\max_{\tilde{x}\in\mathcal{B}_\epsilon[x_i]} p_f(y_i \mid \tilde{x})$$
$$\text{s.t. } \max_{y\in\mathcal{Y}} p_f(y \mid \tilde{x}) - p_f(y_i \mid \tilde{x}) \geq \rho.$$

The constraint ensures $\tilde{x}$ is misclassified by at least ρ, but here the margin ρ is applied to the class probability instead of the loss value. Hence, $\tilde{x}_i$ should usually be different from the one according to the loss value.

A.2. Proofs

We derive a tight upper bound on adversarially robust risk (adversarial risk), and provide our theoretical analysis for the adversarial risk minimization. X and Y represent random variables. Adversarial risk $\mathcal{R}_{\mathrm{rob}}(f) := \mathbb{E}_{(X,Y)\sim\mathcal{D}}\mathbb{1}\{\exists X' \in \mathcal{B}_\epsilon[X] : f(X') \neq Y\}$. $\mathcal{R}_{\mathrm{rob}}(f)$ can be decomposed, i.e., $\mathcal{R}_{\mathrm{rob}}(f) = \mathcal{R}_{nat}(f) + \mathcal{R}_{\mathrm{bdy}}(f)$, where natural risk $\mathcal{R}_{\mathrm{nat}}(f) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}\mathbb{1}\{f(X) \neq Y\}$ and boundary risk $\mathcal{R}_{\mathrm{bdy}}(f) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}\mathbb{1}\{X \in \mathcal{B}_\epsilon[DB(f)], f(X) = Y\}$. Note that $\mathcal{B}_\epsilon[DB(f)]$ is the set denoting the decision boundary of f, i.e., $\{x \in \mathcal{X} : \exists x' \in \mathcal{B}_\epsilon[x] \quad \text{s.t.} \quad f(x) \neq f(x')\}$.

Lemma 1. For any classifier $f : \mathcal{X} \to \mathcal{Y}$, any probability distribution $\mathcal{D}$ on $\mathcal{X} \times \mathcal{Y}$, we have

$$\mathcal{R}_{\mathrm{rob}}(f) = \mathcal{R}_{\mathrm{nat}}(f) + \mathbb{E}_{(X,Y)\sim\mathcal{D}}\mathbb{1}\{\exists X' \in \mathcal{B}_\epsilon[X] : f(X) \neq f(X')\} \cdot \mathbb{1}\{f(X) = Y\}.$$

Proof. By the equation $\mathcal{R}_{\mathrm{rob}}(f) = \mathcal{R}_{\mathrm{nat}}(f) + \mathcal{R}_{\mathrm{bdy}}(f)$,

$$\begin{aligned}
\mathcal{R}_{\mathrm{rob}}(f) &= \mathcal{R}_{\mathrm{nat}}(f) + \mathcal{R}_{\mathrm{bdy}}(f) \\
&= \mathcal{R}_{\mathrm{nat}}(f) + \mathbb{E}_{(X,Y)\sim\mathcal{D}}\mathbb{1}\{X' \in \mathcal{B}_\epsilon[DB(f)], f(X) = Y\} \\
&= \mathcal{R}_{\mathrm{nat}}(f) + \Pr[X' \in \mathcal{B}_\epsilon[DB(f)], f(X) = Y] \\
&= \mathcal{R}_{\mathrm{nat}}(f) + \Pr[f(X) \neq f(X'), f(X) = Y] \\
&= \mathcal{R}_{\mathrm{nat}}(f) + \mathbb{E}_{(X,Y)\sim\mathcal{D}}\mathbb{1}\{\exists X' \in \mathcal{B}_\epsilon[X] : f(X) \neq f(X'), f(X) = Y)\} \\
&= \mathcal{R}_{\mathrm{nat}}(f) + \mathbb{E}_{(X,Y)\sim\mathcal{D}}\mathbb{1}\{\exists X' \in \mathcal{B}_\epsilon[X] : f(X) \neq f(X')\} \cdot \mathbb{1}\{f(X) = Y\}
\end{aligned}$$

The fourth equality comes from the definition of decision boundary of f, i.e., $\mathcal{B}_\epsilon[DB(f)] = \{x \in \mathcal{X} : \exists x' \in \mathcal{B}_\epsilon[x] \quad \text{s.t.} \quad f(x) \neq f(x')\}$. □

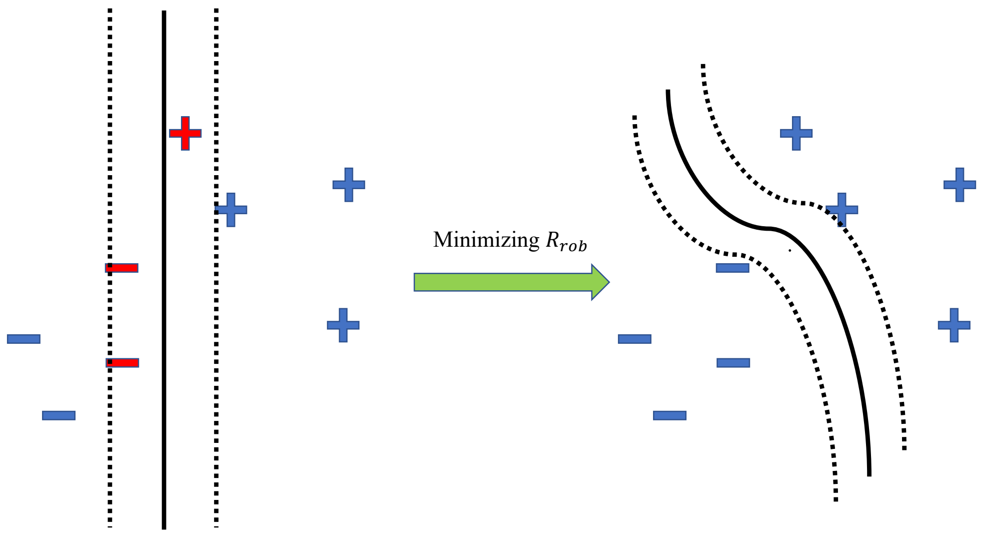


Figure 8. Solid line is the classifier. The area between dashed line is decision boundary of the classifier. Minimizing robust risk R_{rob} is to find a classifier, where data is less likely located within decision boundary of the classifier.

Figure 8 illustrates the message of Lemma 1. Minimizing robust risk R_{rob} encourages the learning algorithm to find a classifier whose decision boundary contains less training data. Meanwhile, the classifier should correctly separate data from different classes. Finding such a classifier is hard. As we can see in Figure 8, the hypothesis set of hyperplanes is enough for minimizing the natural risk (the left figure). However, a robust classifier (the right figure) has much curvatures, which is more complicated. Nakkiran (2019) states that robust classification needs more complex classifiers (exponentially more complex, in some examples). This implies that in order to learn a robust classifier, our learning algorithm needs (a) setting a large hypothesis set and (b) fine-tuning the decision boundary.

Theorem 1 (restated). For any classifier f, any non-negative surrogate loss function ℓ which upper bounds $0/1$ loss, and any probability distribution $\mathcal{D}$, we have

$$\begin{aligned}\mathcal{R}_{\mathrm{rob}}(f) \leq\ & \underbrace{\mathbb{E}_{(X,Y)\sim\mathcal{D}}\ell(f(X),Y)}_{\text{For standard test accuracy}} \\ & + \underbrace{\mathbb{E}_{(X,Y)\sim\mathcal{D},X'\in\mathcal{B}_\epsilon[X,\epsilon]}\ell^*(f(X'),Y)}_{\text{For robust test accuracy}},\end{aligned}$$

where

$$\ell^* = \begin{cases}\min \ell(f(X'),Y)+\rho, & \text{if } f(X')\neq Y;\\ \max \ell(f(X'),Y), & \text{if } f(X') = Y.\end{cases}$$

ρ is the small constant.

Proof.

$$
\begin{aligned}
\mathcal{R}_{\mathrm{rob}}(f) &= \mathcal{R}_{\mathrm{nat}}(f) + \mathcal{R}_{\mathrm{bdy}}(f) \\
&\leq \mathbb{E}_{(X,Y)\sim\mathcal{D}}\ell(f(X),Y) + \mathcal{R}_{\mathrm{bdy}}(f) \\
&= \mathbb{E}_{(X,Y)\sim\mathcal{D}}\ell(f(X),Y) + \mathbb{E}_{(X,Y)\sim\mathcal{D}}\mathbb{1}\{X \in \mathcal{B}_\epsilon[DB(f)], f(X)=Y\} \\
&= \mathbb{E}_{(X,Y)\sim\mathcal{D}}\ell(f(X),Y) + \Pr[X \in \mathcal{B}_\epsilon[DB(f)], f(X)=Y] \\
&= \mathbb{E}_{(X,Y)\sim\mathcal{D}}\ell(f(X),Y) + \Pr[f(X) \neq f(X'), f(X)=Y] \\
&\leq \mathbb{E}_{(X,Y)\sim\mathcal{D}}\ell(f(X),Y) + \Pr[f(X') \neq Y] \\
&= \mathbb{E}_{(X,Y)\sim\mathcal{D}}\ell(f(X),Y) + \mathbb{E}_{(X,Y)\sim\mathcal{D}}\mathbb{1}\{\exists X' \in \mathcal{B}_\epsilon[X] : f(X') \neq Y\} \\
&\leq \mathbb{E}_{(X,Y)\sim\mathcal{D}}\ell(f(X),Y) + \mathbb{E}_{(X,Y)\sim\mathcal{D}, X' \in \mathcal{B}_\epsilon[X,\epsilon]}\ell^*(f(X'),Y)
\end{aligned}
$$

The first inequality comes from the assumption that surrogate loss function ℓ upper bound $0/1$ loss function. The second inequality comes from the fact that there exists misclassified natural data within the decision boundary set. Therefore, $\Pr[f(X) \neq f(X'), f(X)=Y] \cup \Pr[f(X) \neq f(X'), f(X) \neq Y] = \Pr[f(X') \neq Y]$. The third inequality comes from the assumption that surrogate loss function ℓ upper bound $0/1$ loss function, i.e., in Figure 2, the adversarial data X' (purple triangle) is on line of logistic loss (blue line), which is always above the 0/1 loss (yellow line). □

Our Theorem 1 informs our strategy to fine tune the decision boundary. To fine-tune the decision boundary, the data "near" the classifier plays an important role. Those data are easily wrongly predicted with small perturbations. As we show in the Figure 2, when adversarial data are wrongly predicted, our adversarial data (purple triangle) increases to minimize the loss by a violation of a small constant ρ. Thus, our adversarial data can help fine-tune the decision boundary "bit by bit" over the training.

B. Alternative Adversarial Data Searching Algorithm

In this section, we give an alternative adversarial data searching algorithm to generate friendly adversarial data via modifying the method to update $\tilde{x}$ in Algorithm 1. We remove the constraint of ϵ-ball projection in Eq. (5), i.e.,

$$
x^{(t+1)} = x^{(t)} + \alpha\, \mathrm{sign}(\nabla_{x^{(t)}}\ell(f_\theta(x^{(t)}), y)), \forall t \geq 0 \tag{6}
$$

where x^0 is a natural data and $\alpha > 0$ is step size.

We employ Small CNN and ResNet-18 to show the performance of deep models against FGSM, PGD-20, PGD-100 and C&W$_\infty$ in Figure 9 and Figure 10. Deep models are trained using SGD with 0.9 momentum for 80 epochs with the initial learning rate 0.01 divided by 10 at 60 epoch. We compare FAT combined with the alternative adversarial data searching algorithm ($\tau = 0, 1, 3$) and standard adversarial training (Madry) with different step size α, i.e., $\alpha \in [0.003, 0.015]$. The maximum PGD step K is fixed to 10. All the testing settings are the same as those are stated in Section 6.1.

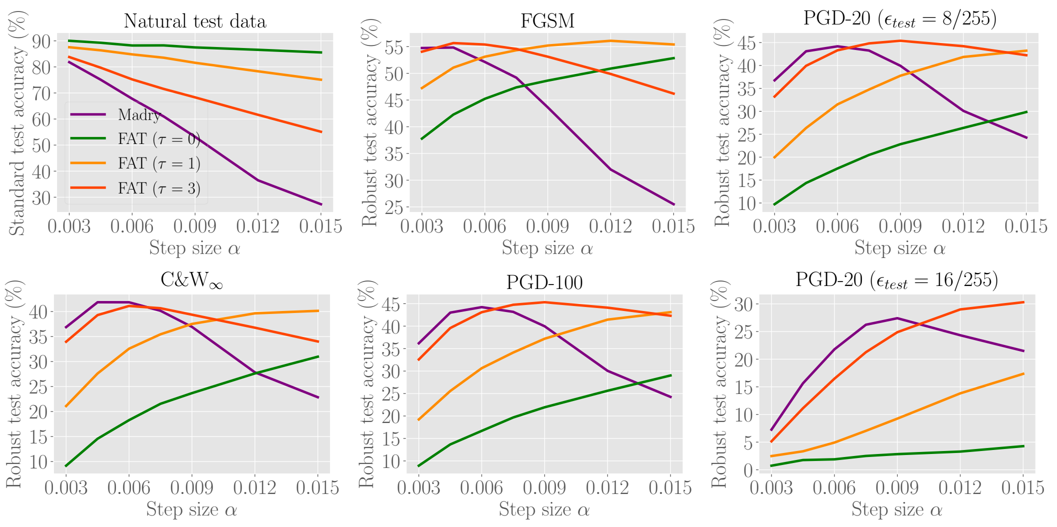


Figure 9. Test accuracy of Small CNN trained under different step size α on CIFAR-10

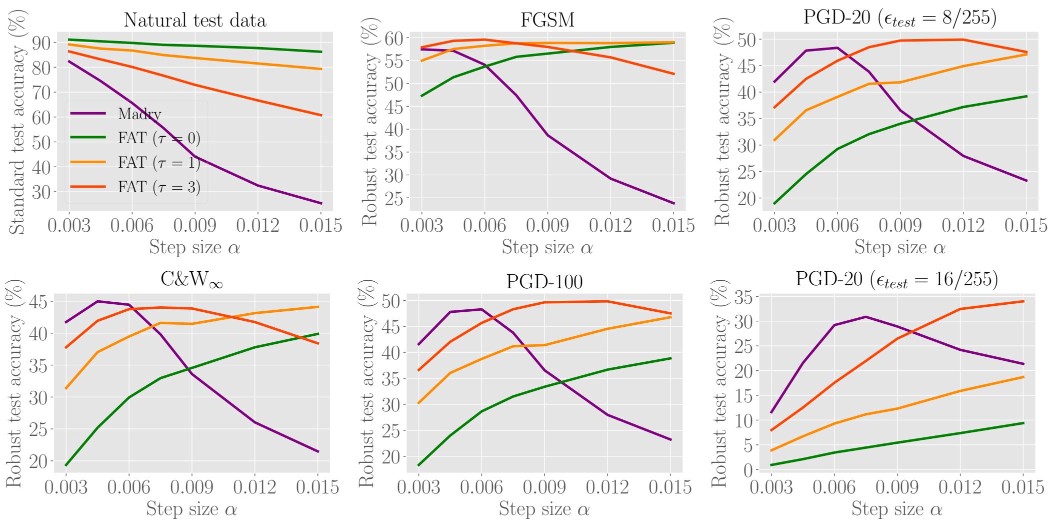


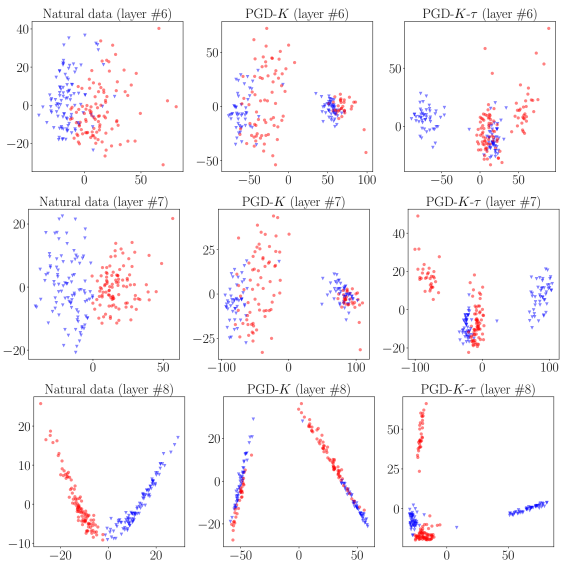
Figure 10. Test accuracy of ResNet-18 trained under different step size α on CIFAR-10

C. Mixture Alleviation

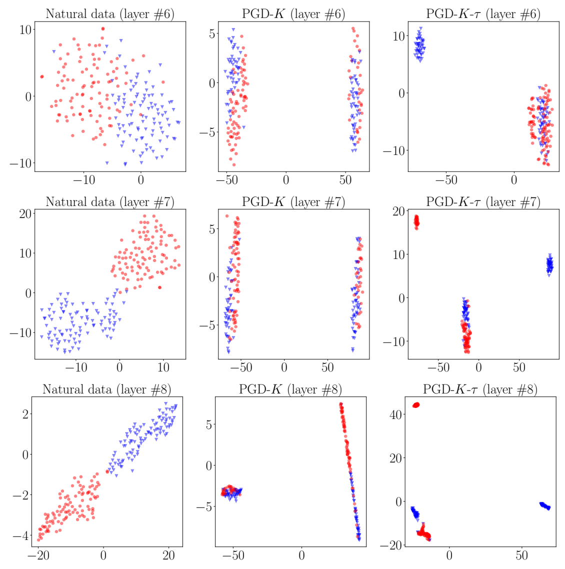
C.1. Output Distributions of Small CNN's Intermediate Layers

In Figure 3 in Section 4.2, we only visualize layer #7s output distribution by Small CNN (8-layer convolutional neural network with 6 convolutional layers and 2 fully connected layers). For completeness, we visualize the output distributions of layers #7 and #8.

We conduct warm-up training using natural training data of two randomly selected classes (bird and deer) in CIFAR-10, then involve its adversarial variants generated by PGD-20 with step size $\alpha = 0.007$ and maximum perturbation $\epsilon = 0.031$. We show output distributions of layer #6, #7 and #8 by PCA in Figure 11(a) and t-SNE in Figure 11(b).



(a) Output distributions visualized by PCA

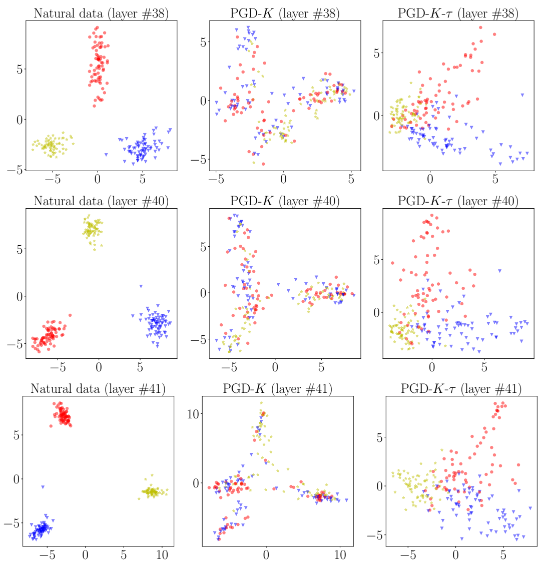


(b) Output distributions visualized by t-SNE

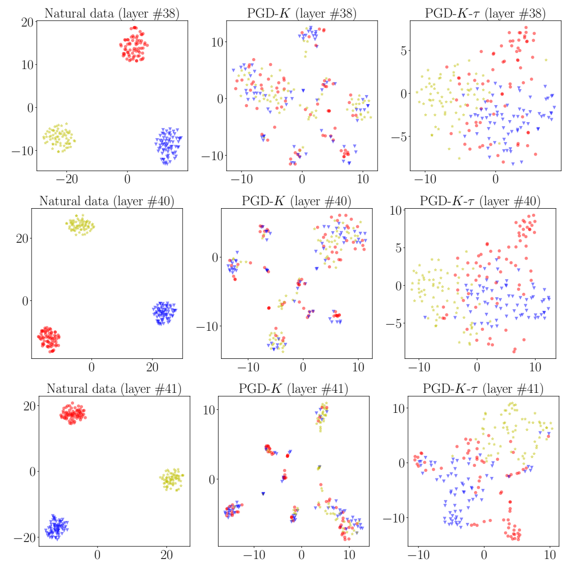
Figure 11. Output distributions of Small CNN's intermediate layers. Left column: Intermediate layers' output distributions on natural data (not mixed). Middle column: Intermediate layers' output distributions on adversarial data generated by PGD-20 (significantly mixed). Right column: Intermediate layers' output distributions on friendly adversarial data generated by PGD-20-0 (no significantly mixed).

C.2. Output distributions of WRN-40-4's intermediate layers

We train a Wide ResNet (WRN-40-4, totally 41 layers) using natural data on 10 classes in CIFAR-10 and then include adversarial variants. We randomly select 3 classes (deer, horse and truck) for illustrating output distributions of WRN-40-4's intermediate layers. Adversarial data are generated by PGD-20 with step size $\alpha = 0.007$ and maximum perturbation $\epsilon = 0.031$ on WRN-40-4. We show output distributions by layer #38, #40 and #41 by PCA in Figure 12(a) and t-SNE in Figure 12(b).



(a) Output distributions visualized by PCA



(b) Output distributions visualized by t-SNE

Figure 12. Output distributions of WRN-40-4's intermediate layers. Left column: Intermediate layers' output distributions on natural data (not mixed). Middle column: Intermediate layers' output distributions on adversarial data generated by PGD-20 (significantly mixed). Right column: Intermediate layers' output distributions on friendly adversarial data generated by PGD-20-0 (no significantly mixed).

D. FAT for TRADES

D.1. Learning Objective of TRADES

Besides the standard adversarial training, TRADES is another effective adversarial training method (Zhang et al., 2019b), which trains on both natural data x and adversarial data $\tilde{x}$.

Similar to virtual adversarial training (VAT) adding a regularization term to the loss function (Miyato et al., 2016), which regularizes the output distribution by its local sensitivity of the output w.r.t. input, the objective of TRADES is

$$\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\left\{\ell(f(x_i),y_i)+\beta\ell_{KL}(f(\tilde{x}_i),f(x_i))\right\}, \tag{7}$$

where $\beta > 0$ is a regularization parameter, which controls the trade-off between standard accuracy and robustness accuracy, i.e., as β increases, standard accuracy will decease while robustness accuracy will increase, and vice visa. Meanwhile, $\tilde{x}_i$ in TRADES is dynamically generated by

$$\tilde{x}_i = \arg\max\nolimits_{\tilde{x}\in\mathcal{B}_\epsilon[x_i]} \ell_{KL}(f(\tilde{x}),f(x)), \tag{8}$$

and ℓ_{KL} is Kullback-Leibler loss that is calculated by

$$\ell_{KL}(f(\tilde{x}),f(x)) = \sum_{i=1}^{C}\ell_{\mathrm{L}}^{i}(f(x))\log\left(\frac{\ell_{\mathrm{L}}^{i}(f(x))}{\ell_{\mathrm{L}}^{i}(f(\tilde{x}))}\right).$$

D.2. FAT for TRADES - Realization

Algorithm 3 PGD-K-τ (Early Stopped PGD for TRADES)

```
Input: data $x \in \mathcal{X}$, label $y \in \mathcal{Y}$, model $f$, loss function $\ell_{KL}$, maximum PGD step $K$, step $\tau$, perturbation bound $\epsilon$, step size $\alpha$
Output: $\tilde{x}$
$\tilde{x} \leftarrow x + \xi\mathcal{N}(\mathbf{0},\mathbf{I})$
while $K > 0$ do
  if $\arg\max_i f(\tilde{x}) \neq y$ and $\tau = 0$ then
    break
  else if $\arg\max_i f(\tilde{x}) \neq y$ then
    $\tau \leftarrow \tau - 1$
  end if
  $\tilde{x} \leftarrow \Pi_{\mathcal{B}[x,\epsilon]}\big(\alpha\,\mathrm{sign}(\nabla_{\tilde{x}}\ell_{KL}(f(\tilde{x}),f(x)) + \tilde{x}\big)$
  $K \leftarrow K - 1$
end while
```

In Algorithm 3, $\mathcal{N}(\mathbf{0},\mathbf{I})$ generates a random unit vector of d dimension. ξ is a small constant. ℓ_{KL} is Kullback-Leibler loss.

Given a dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathcal{R}^d$ and $y_i \in \{0, 1, ..., C-1\}$, adversarial training (TRADES) with early stopped PGD-K-τ returns a classifier θ^*:

$$\theta^* = \arg\min_{\theta}\sum_{i=1}^{n}\left\{\ell_{CE}(f_\theta(x_i),y_i)+\beta\ell_{KL}(f_\theta(\tilde{x}_i),f_\theta(x_i))\right\} \tag{9}$$

where $f_\theta : \mathcal{R}^d \to \mathcal{R}^C$ is DNN classification function, $f_\theta(\cdot)$ outputs predicted probability over C classes, The adversarial data $\tilde{x}_i$ of x_i is dynamically generated according to Algorithm 3, $\beta > 0$ is a regularization parameter, ℓ_{CE} is cross-entropy loss, ℓ_{KL} is Kullback-Leibler loss.

Based on our early stopped PGD-K-τ for TRADES in Algorithm 3, our friendly adversarial training for TRADES (FAT for TRADES) is

Algorithm 4 Friendly Adversarial Training for TRADES (FAT for TRADES)

```
Input: network $f_\theta$, training dataset $S = \{(x_i, y_i)\}_{i=1}^n$, learning rate $\eta$, number of epochs $T$, batch size $m$, number of batches $M$
Output: adversarially robust network $f_\theta$
for epoch $= 1, \ldots, T$ do
    for mini-batch $= 1, \ldots, M$ do
        Sample a mini-batch $\{(x_i, y_i)\}_{i=1}^m$ from $S$
        for $i = 1, \ldots, m$ (in parallel) do
            Obtain adversarial data $\tilde{x}_i$of $x_i$ by Algorithm 3
        end for
        $\theta \leftarrow \theta - \eta \frac{1}{m} \sum_{i-1}^m \nabla_\theta \big[\ell_{CE}(f_\theta(\tilde{x}_i), y_i) + \beta \ell_{KL}(f_\theta(\tilde{x}_i), f_\theta(x_i))\big]$
    end for
end for
```

E. FAT for MART

E.1. Learning Objective of MART

MART (Wang et al., 2020) emphasizes the importance of misclassified natural data on the adversarial robustness. Wang et al. (2020) propose a regularized adversarial learning objective which contains an explicit differentiation of misclassified data as the regularizer. The learning objective of MART is

$$\min_{f\in\mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left\{ \ell_{BCE}(f(\tilde{x}_i), y_i) + \beta \cdot \ell_{KL}(f(\tilde{x}_i), f(x_i)) \cdot (1 - \ell_L^{y_i}(f(x_i))) \right\}, \tag{10}$$

where $\beta > 0$ is a regularization parameter which balances the two parts of the final loss. ℓ_{KL} is Kullback-Leibler loss. ℓ_L^k stands for the k-th element of the soft-max output and $\tilde{x}_i$ in MART is dynamically generated according to Eq. 4 that is realized by PGD-K. The first part ℓ_{BCE} is the proposed BCE loss in MART that is calculated by

$$\ell_{BCE}(f(\tilde{x}_i), y_i) = -\log(\ell_L^{y_i}(f(\tilde{x}_i))) - \log(1 - \max_{k\neq y_i} \ell_L^k(f(\tilde{x}_i)))$$

where the first term $-\log(\ell_L^{y_i}(f(\tilde{x}_i)))$ is the cross-entropy loss and the second term $-\log(1 - \max_{k\neq y_i} \ell_L^k(f(\tilde{x}_i)))$ is a margin term used to increase the distance between $\ell_L^{y_i}(f(\tilde{x}_i))$ and $\max_{k\neq y_i} \ell_L^k(f(\tilde{x}_i))$. This is a similar to C&W (Carlini & Wagner, 2017) attack that is to improve attack strength. For the second part, they combine Kullback-Leibler loss (Zhang et al., 2019b) and emphases on misclassified examples. This part of loss will be large for misclassified examples and small for correctly classified examples.

E.2. FAT for MART - Realization

Given a dataset $S = \{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathcal{R}^d$ and $y_i \in \{0, 1, ..., C-1\}$, FAT for MART returns a classifier θ^*:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{n} \left\{ \ell_{BCE}(f_\theta(\tilde{x}_i), y_i) + \beta \cdot \ell_{KL}(f_\theta(\tilde{x}_i), f_\theta(x_i)) \cdot (1 - \ell_L^{y_i}(f_\theta(x_i))) \right\} \tag{11}$$

where $f_\theta : \mathcal{R}^d \to \mathcal{R}^C$ is DNN classification function, $f_\theta(\cdot)$ outputs predicted probability over C classes. The adversarial data $\tilde{x}_i$ of x_i is dynamically generated according to Algorithm 1, ℓ_L is the soft-max activation, $\beta > 0$ is a regularization parameter, ℓ_{BCE} is the proposed BCE loss in MART and ℓ_{KL} is Kullback-Leibler loss. Based on our early stopped PGD-K-τ in Algorithm 1, FAT for MART Algorithm 5.

F. Experimental Setup

F.1. Selection of Step τ

Figure 4 presents empirical results on CIFAR-10 via our FAT algorithm,where we train 8-layer convolutional neural network (Small CNN, blue line) and 18-layer residual neural network (ResNet-18, red line) (He et al., 2016). The maximum step

Algorithm 5 Friendly Adversarial Training for MART (FAT for MART)

```
Input: network $f_\theta$, training dataset $S = \{(x_i, y_i)\}_{i=1}^n$, learning rate $\eta$, number of epochs $T$, batch size $m$, number of batches $M$
Output: adversarially robust network $f_\theta$
for epoch $= 1, \ldots, T$ do
  for mini-batch $= 1, \ldots, M$ do
    Sample a mini-batch $\{(x_i, y_i)\}_{i=1}^m$ from $S$
    for $i = 1, \ldots, m$ (in parallel) do
      Obtain adversarial data $\tilde{x}_i$of $x_i$ by Algorithm 1
    end for
    $\theta \leftarrow \theta - \eta \frac{1}{m} \sum_{i-1}^{m} \nabla_\theta \left[ \ell_{BCE}(f_\theta(\tilde{x}_i), y_i) + \beta \cdot \ell_{KL}(f_\theta(\tilde{x}_i), f_\theta(x_i)) \cdot (1 - \ell_L^{y_i}(f_\theta(x_i))) \right]$
  end for
end for
```

$K = 10$, $\epsilon_{train} = 8/255$, step size $\alpha = 0.007$, and step $\tau \in \{0, 1, \ldots, 10\}$. We train deep networks for 80 epochs using SGD with 0.9 momentum, where learning rate starts at 0.1 and divided by 10 at 60 epoch.

For each τ, we take five trials, where each trial will obtain standard test accuracy evaluated on natural test data and robust test accuracy evaluated on adversarial test data that are generated by attacks FGSM (Goodfellow et al., 2015), PGD-10 and PGD-20, PGD-100 (Madry et al., 2018) and C&W attack (Carlini & Wagner, 2017) respectively. All those attacks are white box attacks, which are constrained by the same perturbation bound $\epsilon_{test} = 8/255$. Following Zhang et al. (2019b), all attacks have the random start, and the step size α in PGD-10, PGD-20, PGD-100 and C&W is fixed to 0.003.

G. Supplementary Experiments - FAT Enabling Larger ϵ_{train}

In this section, we provide extensive experimental results. The test settings are the same as those are stated in Section 6.1. In Section G.1, instead of using ResNet-18, we conduct adversarial training on the deep model of Small CNN. In Section G.2, instead of applying FAT, we compare our FAT for TRADES and TRADE (Zhang et al., 2019b) under different values of perturbation bound ϵ_{train} on the deep models ResNet-18 and Small CNN. In Section G.3, we set maximum PGD steps $K = 20$ and report results of FAT and FAT for TRADES over existing methods with larger perturbation bound ϵ_{train}. To sum up, all those extensive results verify that FAT and FAT for TRADES can enable deep models trained under larger values of perturbation bound ϵ_{train}.

G.1. A Different Deep Model - Small CNN

We train Small CNN on CIFAR-10 and SVHN using the same settings as those stated in Section 6.1. We show standard and robust test accuracy of deep model (Small CNN) on CIFAR-10 dataset (Figure 13) and SVHN dataset (Figure 14).

G.2. FAT for TRADES

We apply FAT for TRADES(Algorithm 4) to Small CNN and ResNet-18 on CIFAR-10 dataset. All training settings are the same as those are stated in Section 6.1. Regularization parameter $\beta = 6$. We present standard and robust test results of Small CNN (Figure 15) and ResNet-18 (Figure 16).

G.3. Maximum PGD Step $K = 20$

By setting maximum PGD step $K = 20$, we conduct more experiments on Small CNN and ResNet-18 using FAT and FAT for TRADES. Except maximum PGD steps $K = 20$, training settings are the same as those are stated in Section 6.1. Test results of robust deep models are shown in Figures 17, 18, 19 and 20.

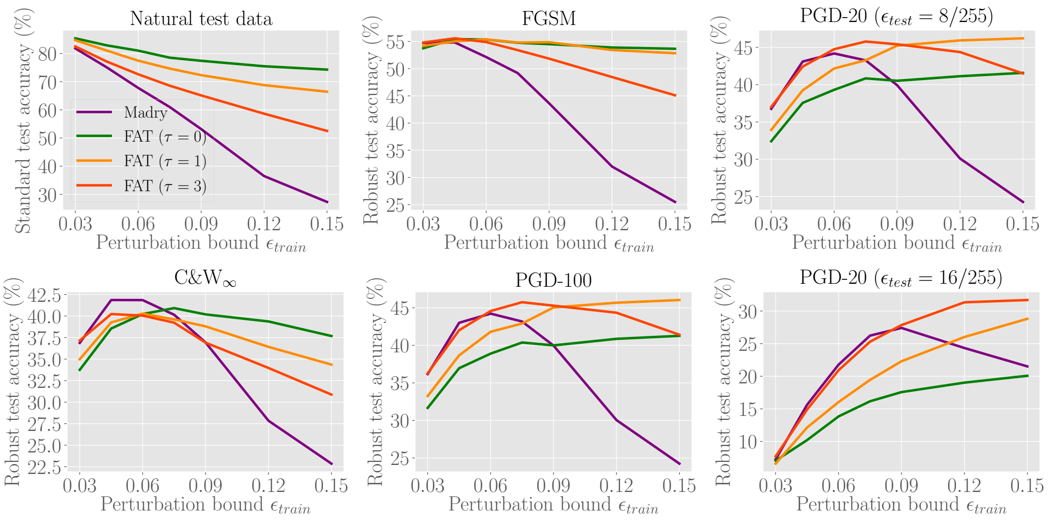


Figure 13. Test accuracy of Small CNN trained under different values of ϵ_{train} on CIFAR-10 dataset.

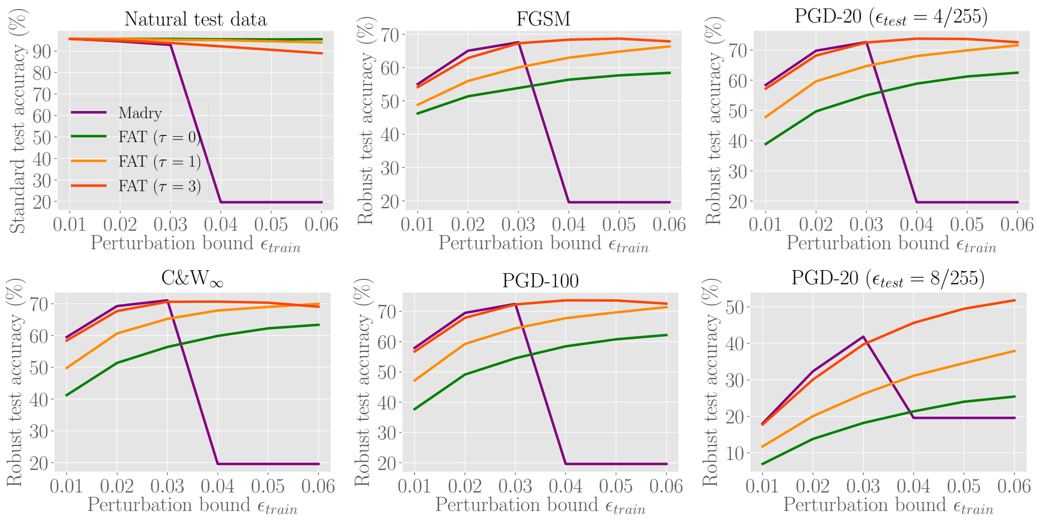


Figure 14. Test accuracy of Small CNN trained under different values of ϵ_{train} on SVHN dataset.

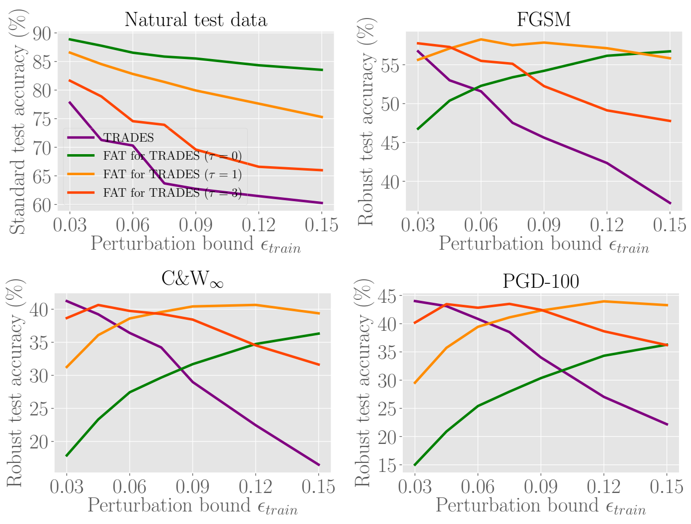


Figure 15. Test accuracy of Small CNN trained by FAT for TRADES ($\tau = 0, 1, 3$) and TRADES under different values of ϵ_{train} on CIFAR-10 dataset.

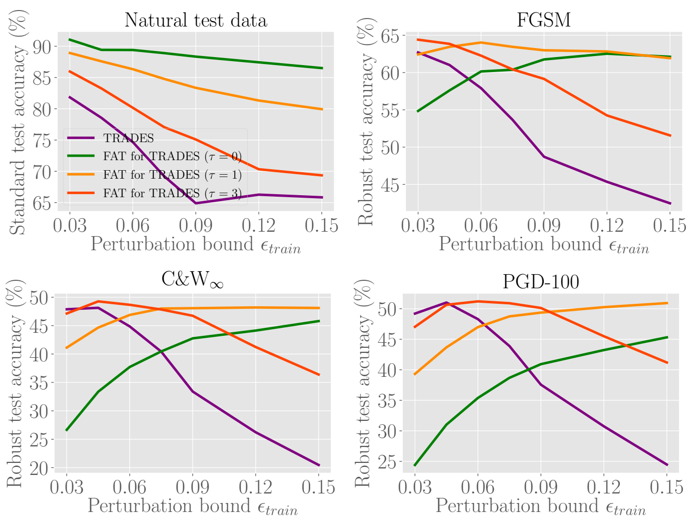


Figure 16. Test accuracy of ResNet-18 trained by FAT for TRADES ($\tau = 0, 1, 3$) and TRADES under different values of ϵ_{train} on CIFAR-10 dataset.

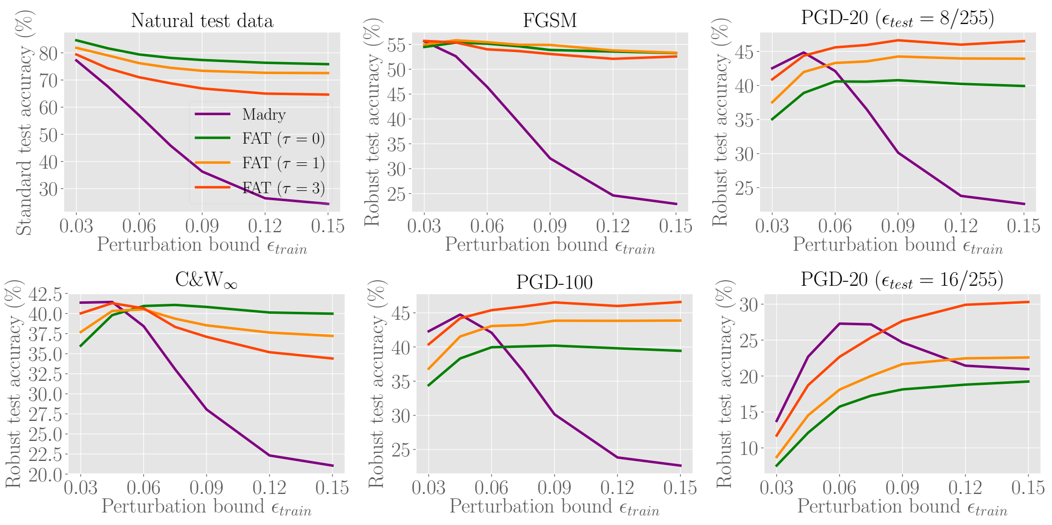


Figure 17. Test accuracy of Small CNN trained by FAT and standard adversarial training (Madry) with maximum PGD step $K = 20$ under different values of ϵ_{train} on CIFAR-10 dataset.

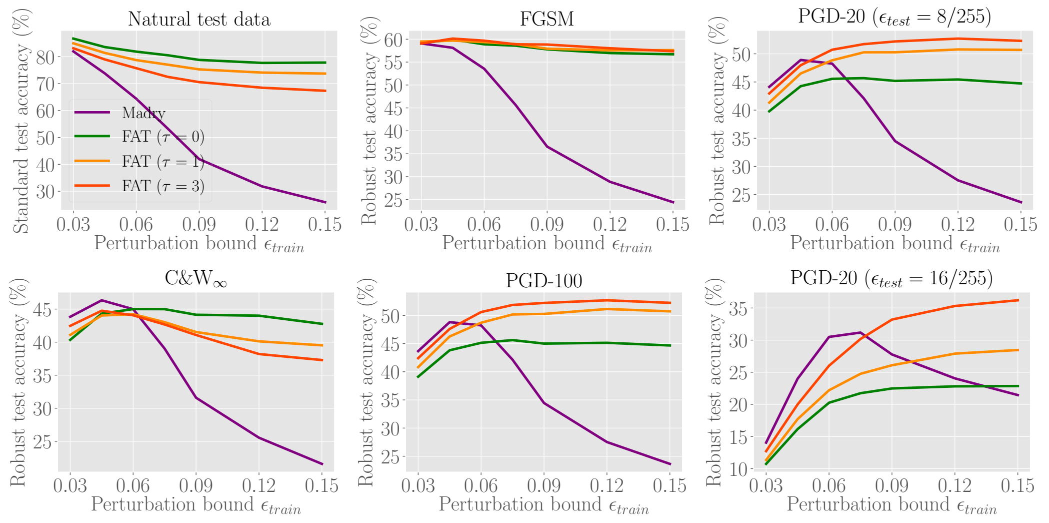


Figure 18. Test accuracy of ResNet-18 trained by FAT standard adversarial training (Madry) with maximum PGD step $K = 20$ under different values of ϵ_{train} on CIFAR-10 dataset.

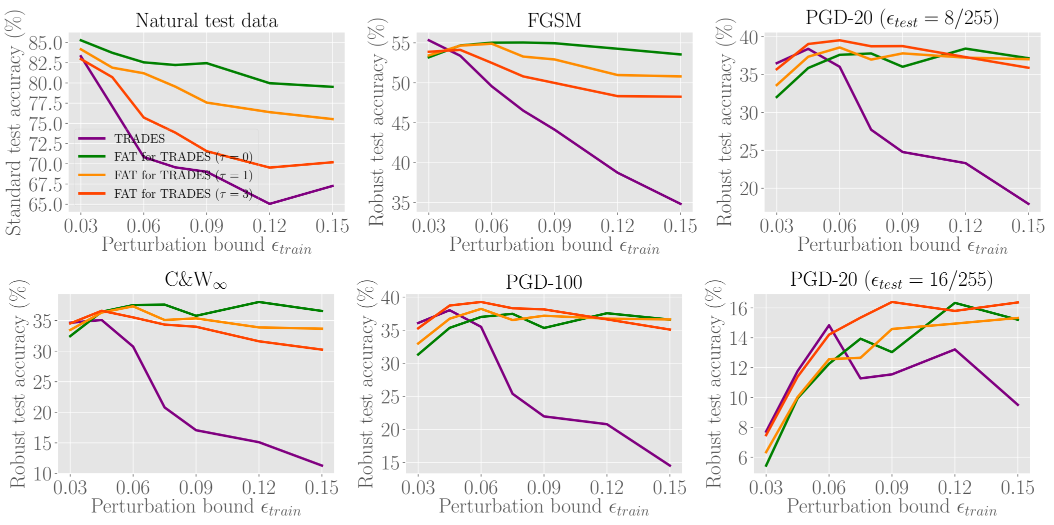


Figure 19. Test accuracy of Small CNN trained by FAT for TRADES and TRADES with maximum PGD step $K = 20$ under different values of ϵ_{train} on CIFAR-10 dataset.

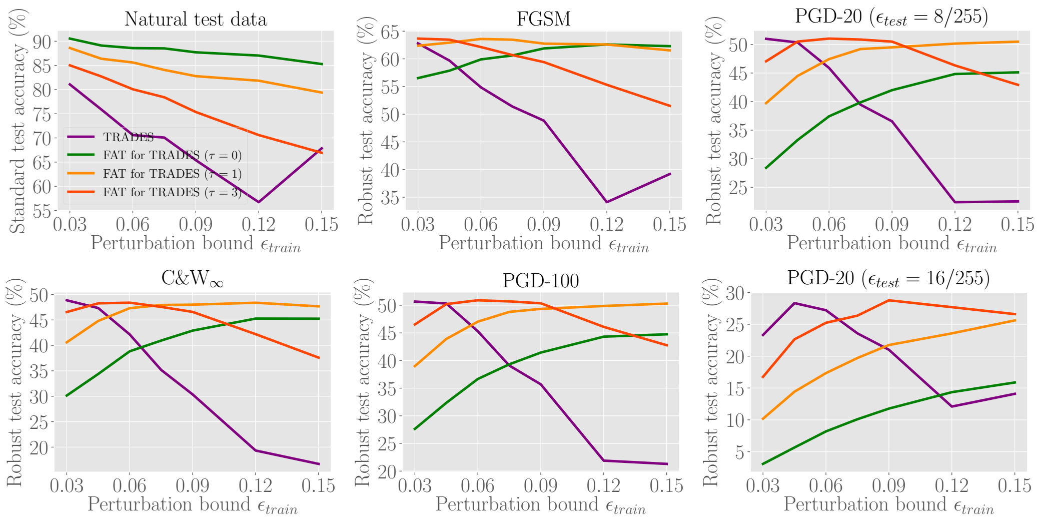


Figure 20. Test accuracy of ResNet-18 trained by FAT for TRADES and TRADES with maximum PGD step $K = 20$ under different values of ϵ_{train} on CIFAR-10 dataset.

G.4. C&W Attack Analysis

As is shown in Figure 6 along with Figure 13 in Section G.1, both standard adversarial training and friendly adversarial training do not perform well under C&W (Carlini & Wagner, 2017) attack with larger ϵ_{train} (e.g., $\epsilon_{train} > 0.075$). We discuss the reasons for these phenomena.

C&W attack. Given x, we choose a target class t and then search for adversarial data $\tilde{x}$ under C&W attack in the L_p metric by solving

$$\text{minimize} \quad \|\tilde{x} - x\|_p + c \cdot h(\tilde{x})$$

with h defined as

$$h(\tilde{x}) = \max(\max_{i \neq t} f(\tilde{x})_i - f(\tilde{x})_t, -\kappa).$$

The parameter $c > 0$ balances two parts of loss. $\kappa > 0$ encourages the solver to find adversarial data $\tilde{x}$ that will be classified as class t with high confidence. Note that this paper follows the implementation of C&W$_\infty$ attack in (Cai et al., 2018)[1] and (Wang et al., 2019)[2] where they replace the cross-entropy loss with $h(\tilde{x})$ in PGD, i.e.,

$$\tilde{x}_i = \arg\max_{\tilde{x} \in \mathcal{B}_\epsilon[x_i]} (\max_{i \neq y_i} f(\tilde{x})_i - f(\tilde{x})_{y_i} - \kappa). \tag{12}$$

Analysis. In Figure 6, with larger ϵ_{train}, the performance evaluated by PGD attacks increases, while performance evaluated by C&W attack decreases. The reason is that C&W and PGD have different ways of generating adversarial data according to Eq. (12) and Eq. (4) respectively. The two interactive methods search adversarial data in different directions due to gradients w.r.t. different loss. Therefore, the distributions of C&W and PGD adversarial data are inconsistent. As perturbation bound ϵ_{train} increases, there are more PGD adversarial data generated within ϵ_{train}-ball. A DNN learned from more PGD adversarial data becomes more defensive to PGD attacks, but this deep model may not effectively defend C&W adversarial data.

[1] curriculum adversarial training GitHub
[2] dynamic adversarial training GitHub

H. Extensive State-of-the-art Results on Wide ResNet

H.1. Training Details of FAT on WRN-32-10

In Table 1, we compare our FAT with standard adversarial training (Madry), CAT (Cai et al., 2018) and DAT (Wang et al., 2019).

We use FAT ($\epsilon_{train} = 8/255$ and $16/255$ respectively) to train WRN-32-10 for 120 epochs using SGD with 0.9 momentum, and weight decay is 0.0002. Maximum PGD step is 10 and step size is fixed to 0.007. The initial learning rate is 0.1 reduced to 0.01, 0.001 and 0.0005 at epoch 60, 90 and 110. We set step $\tau = 0$ initially and increase τ by one at epoch 50 and 90 respectively. The maximum step $K = 10$. We report performance of the deep model at the last epoch. For fair comparison, in Table 1 we use the same test settings as those in DAT (Wang et al., 2019). Performance of robust deep model is evaluated standard test accuracy for natural data and robust test accuracy for adversarial data, that are generated by FGSM, PGD-20 (20-steps PGD with random start), PGD-100 and C&W$_\infty$(L$_\infty$ version of C&W optimized by PGD-30).

All attacks have the same perturbation bound $\epsilon_{test} = 0.031$ and step size in PGD is $\alpha = \epsilon_{test}/4$. The same as DAT (Wang et al., 2019), there is random start in PGD attack, i.e., uniformly random perturbations ($[-\epsilon_{test}, +\epsilon_{test}]$) added to natural data before PGD perturbations. We report the median test accuracy and its standard deviation over 5 repeated trails of adversarial training in Table 1.

H.2. Training details of FAT for TRADES on Wide ResNet

In Table 2, we use FAT for TRADES ($\epsilon_{train} = 8/255$ and $16/255$ respectively) train WRN-34-10 by FAT for TRADES for 85 epochs using SGD with 0.9 momentum and 0.0002 weight decay. Maximum PGD step $K = 10$ and step size $\alpha = 0.007$. The initial learning rate is 0.1 and divided 10 at epoch 75. We set step $\tau = 0$ initially and increased by one at epoch 30, 50 and 70. Since TRADES has a trade-off parameter β, for fair comparison, our FAT for TRADES use the same β. In Table 2, we set $\beta = 1$ and 6 separately, which are endorsed by (Zhang et al., 2019b).

For fair comparison, we use the same test settings as those are stated in TRADES (Zhang et al., 2019b). All attacks have the same perturbation bound $\epsilon_{test} = 0.031$ (without random start), and step size $\alpha = 0.003$, which is the same as stated in the paper (Zhang et al., 2019b). Performance of robust deep model is evaluated standard test accuracy for natural data and robust test accuracy for adversarial data, that are generated by FGSM, PGD-20, PGD-100 and C&W$_\infty$(L$_\infty$ version of C&W optimized by PGD-30). We report the median test accuracy and its standard deviation of the deep model at the last epoch over 3 repeated trials of adversarial training in Table 2.

Fair comparison based on TRADES's experimental setting. However, in TRADES's experimental testing[3], they use random start before PGD perturbation that is deviated from the statements in the paper (Zhang et al., 2019b). For fair comparison, we also retest the robust deep models under PGD attacks with random start. We evaluate their publicly released robust deep model[4] WRN-34-10 and compare it with ours trained by FAT for TRADES. The test results are reported in Table 3.

FAT for TRADES on larger WRN-58-10. We employ Wide ResNet with larger capacity, i.e., WRN-58-10 to show our superior performance achieved by FAT for TRADES in Table 3. All the training settings are the same as details on WRN-34-10 in this section. The regularization parameter β is fixed to 6.0. All attacks have the same perturbation bound $\epsilon_{test} = 0.031$ and step size $\alpha = 0.003$, which is the same as TRADES's experimental setting. Robustness against FGSM, PGD-20(20-steps PGD with random start) and C&W$_\infty$ is reported in Table 3.

H.3. FAT for MART on Wide ResNet

We train WRN-34-10 by FAT for MART ($\epsilon_{train} = 8/255$ and $16/255$ respectively) using SGD with 0.9 momentum and 0.0002 weight decay. Maximum PGD step $K = 10$ and step size $\alpha = 0.007$. The initial learning rate is 0.1 and divided 10 at epoch 60 and 90 respectively. We set step $\tau = 0$ initially and increase τ by one at epoch 20, 40, 60 and 80. The regularization parameter β is fixed to 6.0. The maximum step size $K = 10$.

[3]TRADES GitHub
[4]TRADES's pre-trained model

Table 3. Robustness (test accuracy) of deep models on CIFAR-10 dataset (evaluated with random start)

Model	Defense	Natural	FGSM	PGD-20	C&W$_\infty$
WRN-34-10	TRADES ($\beta = 6.0$)	84.92	67.00	57.18	54.72
	FAT for TRADES ($\epsilon_{train} = 8/255$)	86.38 $\pm$ 0.548	67.64 $\pm$ 0.572	56.65 $\pm$ 0.262	54.51 $\pm$ 0.299
	FAT for TRADES ($\epsilon_{train} = 16/255$)	84.39 $\pm$ 0.030	67.38 $\pm$ 0.370	57.67 $\pm$ 0.198	54.62 $\pm$ 0.140
WRN-58-10	FAT for TRADES ($\epsilon_{train} = 8/255$)	**87.09**	**68.7**	57.17	55.43
	FAT for TRADES ($\epsilon_{train} = 16/255$)	85.28	68.08	**58.39**	**55.89**

Fair comparison based on MART's experimental setting. For fair comparison, all attacks have the same perturbation bound $\epsilon_{test} = 8/255$ and step size $\alpha = \epsilon_{test}/10$, which is the same setting in MART (Wang et al., 2020). White-box robustness of the deep model against attacks such as FGSM, PGD-20 (20-steps PGD with random start) and C&W$_\infty$ (L$_\infty$ version of C&W optimized by PGD-30) is reported. We evaluate Wang et al. (2020) publicly released robust deep model[5] WRN-34-10 and compare it with ours trained by FAT for MART. In Table 4, we report the median test accuracy and its standard deviation over 3 repeated trails of FAT for MART on WRN-34-10.

FAT for MART on larger WRN-58-10. In Table 4, we also employ WRN-58-10 to show the performance achieved by FAT for MART. All the training and testing settings are the same as those on WRN-34-10.

Table 4. Robustness (test accuracy) of deep models on CIFAR-10 dataset

Model	Defense	Natural	FGSM	PGD-20	C&W$_\infty$
WRN-34-10	MART ($\beta = 6.0$)	83.62	67.38	58.24	**53.67**
	FAT for MART ($\epsilon_{train} = 8/255$)	86.40 $\pm$ 0.071	68.94 $\pm$ 0.195	57.89 $\pm$ 0.144	52.28 $\pm$ 0.110
	FAT for MART ($\epsilon_{train} = 16/255$)	84.39 $\pm$ 0.390	68.52 $\pm$ 0.297	59.13 $\pm$ 0.180	52.85 $\pm$ 0.459
WRN-58-10	FAT for MART ($\epsilon_{train} = 8/255$)	**87.10**	**69.52**	58.57	52.73
	FAT for MART ($\epsilon_{train} = 16/255$)	85.19	69.00	**59.82**	53.01

[5]MART's pre-trained model