# ZIPPER: DECOUPLING THE TRADEOFF BETWEEN ROBUSTNESS AND ACCURACY

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Deep neural networks obtained by standard training have been constantly plagued by adversarial examples. Although adversarial training demonstrates its capability to defend against adversarial examples, unfortunately, it leads to an inevitable drop in the natural generalization. To address the issue, we decouple the natural generalization and the robust generalization from joint training and formulate different training strategies for each one. Specifically, instead of minimizing a global loss on the expectation over these two generalization errors, we propose a bi-expert framework called *Zipper* where we simultaneously train base learners with task-aware strategies so that they can specialize in their own fields. The parameters of base learners are collected and combined to form a global learner at intervals during the training process, which is then distributed to base learners as initialized parameters for continued training. Theoretically, we show that the risks of Zipper will get lower once the base learners are well trained. Extensive experiments verify the applicability of Zipper to achieve high clean accuracy in the natural setting while keeping considerably robust to the adversarial setting.

## 1 INTRODUCTION

Modern deep learning techniques have achieved remarkable success in many fields, including computer vision (Krizhevsky et al., 2012; He et al., 2016), natural language processing (Vaswani et al., 2017; Devlin et al., 2019), and speech recognition (Sak et al., 2015). Yet, deep neural networks (DNNs) suffer a catastrophic performance degradation by adversarial perturbations where wrong predictions are made with extremely high confidence (Szegedy et al., 2014; Goodfellow et al., 2015). The vulnerability of DNNs has led to the proposal of various defense approaches (Xu et al., 2018; Liao et al., 2018; Qin et al., 2020; Papernot et al., 2016; Ross & Doshi-Velez, 2018) for protecting DNNs from adversarial attacks. One of those representative techniques is adversarial training (AT) (Madry et al., 2018), which dynamically injects perturbed examples that deceive the current model but preserve the right label into the training set. Adversarial training has been demonstrated to be the most effective method to improve adversarially robust generalization (Athalye et al., 2018).

Despite these successes, such attempts of adversarial training have found a tradeoff between natural and robust accuracy, *i.e.*, there exists an undesirable increase in the error on unperturbed images when the error on the worst-case perturbed images decreases. Prior works (Fawzi et al., 2018; Tsipras et al., 2019; Zhang et al., 2019) even argue that natural and robust accuracy are fundamentally at odds, which indicates that a robust classifier can be achieved only when compromising the natural generalization. However, the following works found that the tradeoff may be settled in a roundabout way, such as incorporating additional labeled/unlabeled data (Alayrac et al., 2019; Najafi et al., 2019; Carmon et al., 2019; Raghunathan et al., 2019; 2020) or relaxing the magnitude of perturbations to generate suitable adversarial examples for better optimization (Lamb et al., 2019; Zhang et al., 2020; Lee et al., 2020). These works all focus on the data used for training while we propose to tackle the tradeoff problem from the perspective of the training paradigm in this paper.

Inspired by the spirit of the divide-and-conquer method, we decouple the objective function of adversarial training into two sub-tasks: one is used for natural example classification while the other one is used for adversarial example classification. Specifically, for each sub-task, we train a base learner on natural or adversarial datasets with the task specific configuration while sharing the same model architecture. The parameters of base learners are collected and combined to form a global learner

at intervals during the training process, which is then distributed to base learners as initialized parameters for continued training. We name the framework as *Zipper* and its proof-of-concept pipeline is shown in Figure 1. Different from the traditional joint training framework for natural and robust generalization, our proposed Zipper fully leverages task-specific information to individually train the base learners, which makes each sub-task be solved better. Theoretically, we show that if the base learners are well trained, the final global learner is guaranteed to have a lower risk. The proposed Zipper is the first to mitigate the tradeoff issue by task-aware training strategies to achieve high clean accuracy in the natural setting, while also being considerably robust to the adversarial setting. In summary, the main contributions are as follows:

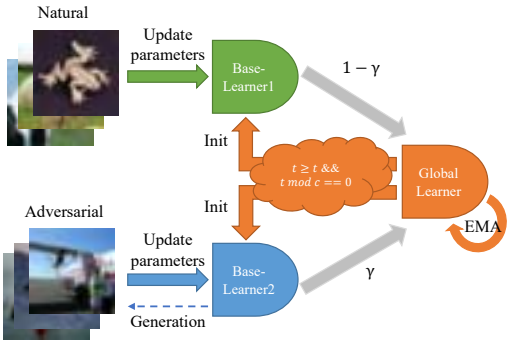



Figure 1: A pipeline for the proposed Zipper.

- For the tradeoff between natural and robust generalization, previous methods are hard to find a sweet point to meet both goals in the joint training framework. Here, we propose a novel Zipper paradigm, which constructs multiple task-aware base learners to respectively achieve the generalization goal on natural and adversarial datasets.
- For each task, rather than being constricted in a stiff manner, every detail of training strategies (*e.g.*, optimization scheme) can be totally customized, thus each base learner can better explore the optimal trajectory in its field and the global learner can fully leverage the merits of all base learners.
- We conduct comprehensive experiments in common settings against extensive adversarial attacks to verify the applicability of our approach. Results demonstrate that both clean and robust accuracy have been greatly improved on benchmark datasets compared to relevant techniques.

## 2 Preliminaries and Related Work

In this section, we briefly introduce some relevant background knowledge and terminology about adversarial training and meta-learning.

**Notations.** Consider an image classification task with input space $\mathcal{X}$ and output space $\mathcal{Y}$. Let $x \in \mathcal{X} \subseteq \mathbb{R}^d$ denote a natural image and $y \in \mathcal{Y} = \{1, 2, \ldots, K\}$ denote the corresponding ground-truth label. The natural and adversarial datasets $\mathcal{X} \times \mathcal{Y} = \{(x_i, y_i)\}_{i=1}^n$ and $\mathcal{X}' \times \mathcal{Y} = \{(x_i', y_i)\}_{i=1}^n$ are sampled from a distribution $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. We denote a DNN model as $f_\theta : \mathcal{X} \to \mathbb{R}^K$ whose parameters are $\boldsymbol{\theta} \in \Theta$, which should classify any input image into one of $K$ classes. The objective functions $\ell_1$ and $\ell_2$ for the natural and adversarial setting can be defined as: $\ell_1 \overset{def}{=} \mathcal{D}_1 \times \Theta \to [0, \infty)$ and $\ell_2 \overset{def}{=} \mathcal{D}_2 \times \Theta \to [0, \infty)$, which are usually positive, bounded, and upper-semi continuous (Blanchet & Murthy, 2019; Villani, 2003; Bartlett & Mendelson, 2001).

### 2.1 Standard Adversarial Training

The goal of the adversary is to generate a malignant example $x'$ by adding an imperceptible perturbation $\varepsilon \in \mathbb{R}^d$ to $x$. And the generated adversarial example $x'$ should be in the vicinity of $x$ so that it looks visually similar to the original one. This neighbor region $\mathbb{B}_\varepsilon(x)$ anchored at $x$ with apothem $\varepsilon$ can be defined as $\mathbb{B}_\varepsilon(x) = \{(x', y) \in \mathcal{D}_2 \mid \|x - x'\| \leq \varepsilon\}$. For adversarial training, it first generates adversarial examples and then updates the parameters over these samples. The iteration process of adversarial training can be summed up as:

$$\begin{cases} x'^{(t+1)} = \Pi_{\mathbb{B}(x,\epsilon)} \left( x'^{(t)} + \alpha \operatorname{sign} \left( \nabla_{x'} \ell_2 \left( x'^{(t)}, y; \boldsymbol{\theta}^t \right) \right) \right) \\ \boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \tau \nabla_{\boldsymbol{\theta}} \mathbb{E}[\ell_1(x, y; \boldsymbol{\theta}^t) + \beta \mathcal{R}(x', x, y; \boldsymbol{\theta}^t)], \end{cases} \tag{1}$$

where $\Pi_{\mathbb{B}(x,\epsilon)}$ is the projection operator, $\alpha$ is the step size, $\tau$ is the learning rate, and $\mathcal{R}(\cdot)$ is the loss difference of $\ell_2(x', y; \boldsymbol{\theta}^t) - \ell_1(x, y; \boldsymbol{\theta}^t)$. The tradeoff factor $\beta$ balances the importance of natural and robust errors. Various adversarial training methods can be derived from Eq. 1. For instance, when $\beta = 1$, it is equivalent to the vanilla PGD training (Madry et al., 2018), and when $\beta = 1/2$, it is transformed into the half-half loss in Goodfellow et al. (2015). The formulation degenerates to standard natural training as $\beta = 0$. Besides, we can get the formulation in TRADES (Zhang et al., 2019) when replacing $\mathcal{R}(\cdot)$ with the KL-divergence.

### 2.2 Multi-Task Learning and Meta-Initialization

**Multi-Task Learning.** Multi-Task Learning (MTL) is to improve performance across tasks through joint training of different models (Bilen & Vedaldi, 2016; Lu et al., 2017; Yang & Hospedales, 2017). Consider a set of assignments containing data distribution and loss function defined as $\mathcal{A} = \{\mathcal{D}, \ell\}$ with corresponding models $\{\mathcal{M}_a\}_{a=1}^{|\mathcal{A}|}$ parameterized by trainable tensors $\boldsymbol{\theta}_{\mathcal{M}_a}$. In MTL, these sets have non-trivial pairwise intersections, and are trained in a joint model to find optimal parameters $\boldsymbol{\theta}^\star_{\mathcal{M}_a}$ for each task:

$$\bigcup_{a=1}^{|\mathcal{A}|} \boldsymbol{\theta}^\star_{\mathcal{M}_a} = \underset{\cup_{a=1}^{|\mathcal{A}|}\boldsymbol{\theta}_{\mathcal{M}_a}}{\operatorname{argmin}} \mathbb{E}_{\mathcal{A}}\mathbb{E}_{\mathcal{D}}\, \ell_a\left(\mathcal{D}_a; \boldsymbol{\theta}_{\mathcal{M}_a}\right), \tag{2}$$

where $\ell_a(\mathcal{D}_a; \boldsymbol{\theta}_{\mathcal{M}_a})$ measures the performance of a model trained using $\boldsymbol{\theta}_{\mathcal{M}_a}$ on dataset $\mathcal{D}_a$. Our approach Zipper is directly related to MTL at first glance because both of them tend to learn a specific predictive model for different sources. However, Zipper differs significantly from MTL, *i.e.*, multiple tasks are still learned *jointly* under a unified form in MTL while each assignment can be optimized by heterogeneous strategies in Zipper.

**Meta-Learning.** Meta-learning is to train a model that can quickly adapt to a new task. Suppose $\mathcal{A}$ is divided into non-overlapping splits $\mathcal{V}$ and $\mathcal{W}$, the model is first trained on the training sets and then guided by a small validation set on a set of tasks to make the trained model can be well adapted to new tasks:

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathbb{E}_{\mathcal{V}}\mathbb{E}_{\mathcal{D}_\mathcal{V}}\, \ell_\mathcal{V}\left(\mathcal{D}_\mathcal{V}; \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathbb{E}_{\mathcal{W}}\mathbb{E}_{\mathcal{D}_\mathcal{W}}\, \ell_\mathcal{W}\left(\mathcal{D}_\mathcal{W}; \boldsymbol{\theta}\right)\right), \tag{3}$$

meta-learning (Finn et al., 2017; Nichol et al., 2018) is often designed to generalize across unseen tasks, whereas the goal of MTL is to tackle a series of known tasks. Nonetheless, our approach Zipper uses the technique of meta-learning to set good initializations for base learners to transfer knowledge between tasks.

## 3 The Proposed New Framework: Zipper

Similar to a physical-world zipper, our proposed Zipper also consists of two rows of protruding teeth (*i.e.*, base learners). Numbers of teeth are paved along both halves of a zipper to link the rows (*i.e.*, checkpoints of base learners through training). A Y-shaped slider (*i.e.*, the global learner) moves along the rows of teeth from a starting point and meshes together with the opposing rows of teeth.

### 3.1 Overview

The overall procedure of our proposed algorithm is shown in Algorithm 1, which mainly comprises two steps: optimizing parameters of the base learner $\boldsymbol{\theta}_a$ in its assigned data distribution $\mathcal{D}_a$ and distributing parameters of the global learner $\boldsymbol{\theta}_g$ to all base learners. Base learners and the global learner share the same architecture, *i.e.*, $\mathcal{M}_1 = \mathcal{M}_2 = \cdots = \mathcal{M}_{|\mathcal{A}|}$. Since we only focus on recognizing natural examples and adversarial ones in our setting, the total number of tasks $\mathcal{W}$ is set to two.

### 3.2 Task-aware base learners

Given a global data distribution $\mathcal{D}$ for the tradeoff problem, as denoted in Section 2, $\mathcal{D}_1, \mathcal{D}_2$ are subject to the distribution of training data $\mathcal{D}_\mathcal{W}$. And natural images $(x, y) \sim \mathcal{D}_1$ while adversarial

**Algorithm 1** Zipper: Leverage the learning trajectory with respect to task-aware base learners

**Input:** A DNN classifier $f(\cdot)$ with initial learnable parameters $\boldsymbol{\theta}_g$ for the global learner and $\boldsymbol{\theta}_n, \boldsymbol{\theta}_r$ for each base learner with objective function $\ell_1, \ell_2$; number of iterations $T$; number of adversarial attack steps $K$; magnitude of perturbation $\varepsilon$; step size $\kappa$; learning rate $\tau_1, \tau_2$; exponential decay rates for ensembling $\alpha' = 0.999$; mixing ratio $\gamma$; starting point and frequency of communication $t', c$.
Initialize $\boldsymbol{\theta}_g, \boldsymbol{\theta}_n, \boldsymbol{\theta}_r$ in $\Theta$ space.
**for** t $\leftarrow 1, 2, \cdots, T$ **do**
  Sample a minibatch $(x, y)$ from data distribution $\mathcal{D}_1$
  */* Parallel-1: Update parameters of base learner-1 over $\mathcal{D}_1$ */*
  (Optional) Performing model ensembling, data augmentation or label smoothing, etc.
  $\boldsymbol{\theta}_n \leftarrow \mathcal{Z}_n \left[\mathbb{E}_{(x,y)}(\nabla_{\boldsymbol{\theta}} \ell_1(x, y; \boldsymbol{\theta}_n)), \tau_n\right]$
  */* Parallel-2: Update parameters of base learner-2 over $\mathcal{D}_2$ */*
  $x'_0 \leftarrow x + \varepsilon$, $\varepsilon \sim \mathrm{Uniform}(-\varepsilon, \varepsilon)$.
  **for** k $\leftarrow 1, 2, \cdots, K$ **do**
    $x'_k \leftarrow \Pi_{x'_k \in \mathbb{B}_\varepsilon(x)} \left(\kappa \operatorname{sign}\left(x'_{k-1} + \nabla_{x'_{k-1}} \ell_2(x'_{k-1}, y; \boldsymbol{\theta}_r)\right)\right)$
  **end for**
  (Optional) Performing model ensembling, data augmentation or label smoothing, etc.
  $\boldsymbol{\theta}_r \leftarrow \mathcal{Z}_r \left[\mathbb{E}_{(x',y)}(\nabla_{\boldsymbol{\theta}} \ell_2(x'_K, y; \boldsymbol{\theta}_r)), \tau_r\right]$
  */* For the global learner*/*
  $\boldsymbol{\theta}_g \leftarrow \alpha' \boldsymbol{\theta}_g + (1 - \alpha')(\gamma \boldsymbol{\theta}_r + (1 - \gamma)\boldsymbol{\theta}_n)$
  **if** $t \geq t'$ and $t \mod c == 0$ **then**
    $\boldsymbol{\theta}_r, \boldsymbol{\theta}_n \leftarrow \boldsymbol{\theta}_g$
  **end if**
**end for**
**Return** Parameters of the global learner $\boldsymbol{\theta}_g$

examples $(x', y) \sim \mathcal{D}_2$ generated by Eq. 1. So the training process of base learners is to solve the inner minimization of Eq. 3 over different distributions in a distributed manner:

$$\{\boldsymbol{\theta}_n^\star, \boldsymbol{\theta}_r^\star\} = \underset{\bigcup_{\mathcal{W}=1}^{2} \boldsymbol{\theta}_{\mathcal{W}}}{\operatorname{argmin}} \mathbb{E}_{\mathcal{D}_\mathcal{W}} \, \ell_\mathcal{W} \left(\mathcal{D}_\mathcal{W}; \boldsymbol{\theta}_\mathcal{W}\right). \tag{4}$$

Specifically, during the process, base learners $f_{\boldsymbol{\theta}_n}$ and $f_{\boldsymbol{\theta}_r}$ are assigned different subproblems that only requires accessing their own data distribution, respectively. Note that two base learners work in a complementary manner, meaning the update of parameters is independent among base learners and the global learner always collects parameters of both base learners. So the subproblem for each base learner is defined as:

$$\boldsymbol{\theta}_\mathcal{W}^\star = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{Z}_\mathcal{W}^T \left[\mathbb{E}_\mathcal{W}(\nabla_{\boldsymbol{\theta}} \ell_\mathcal{W}(\mathcal{D}_\mathcal{W}; \boldsymbol{\theta}_\mathcal{W})), \tau_\mathcal{W}\right], \tag{5}$$

where the task-aware optimizer $\mathcal{Z}_\mathcal{W}^T(\cdot, \cdot)$ search the optimal parameter states $\boldsymbol{\theta}_\mathcal{W}^\star$ over the subproblem $\mathcal{W}$ in $T$ rounds. Loss functions can also be task-specific and applied to each base learner separately. It is natural to consider minimizing the 0-1 loss in the natural and robust errors, however, solving the optimization problem is NP-hard thus computationally intractable. In practice, we select cross-entropy as the surrogate loss for both $\ell_1$ and $\ell_2$ since it is simple but good enough.

### 3.3 Initialization from the Global Learner

During the initial training periods, base learners are less instrumental since they are not adequately learned. Directly initializing parameters of base learners may mislead the training procedure and further accumulate bias when mixing them. Therefore, we set aside $t'$ epochs from the beginning for fully training base learners and just aggregates states on the searching trajectory of base learners through optimization by exponential moving average (EMA), computed as: $\boldsymbol{\theta}_g \leftarrow \alpha' \boldsymbol{\theta}_g + (1 - \alpha')(\gamma \boldsymbol{\theta}_r + (1 - \gamma)\boldsymbol{\theta}_t)$, where $\alpha'$ is the exponential decay rates for EMA and $\gamma$ is the mixing ratio for base learners. They then learn an initialization from parameters of the global learner every $c$ epochs when each base learner is well trained in its field. Thus, the optimization of each base learner for every interlude can be expressed in Eq. 6:

$$\boldsymbol{\theta}_\mathcal{W}^\star = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{Z}_\mathcal{W}^c \left[\mathbb{E}_\mathcal{W}(\nabla_{\boldsymbol{\theta}} \ell_\mathcal{W}(\mathcal{D}_\mathcal{W}; \boldsymbol{\theta}_g)), \tau_\mathcal{W}\right]. \tag{6}$$

Note that $\boldsymbol{\theta}_g$ contains both $\boldsymbol{\theta}_n$ and $\boldsymbol{\theta}_r$, meaning there always exists a term updated by gradient information of distribution different from the current subproblem. This mechanism enables fast

learning within a given assignment and improves generalization, and the acceleration is applicable to the given assignment for its corresponding base learner only (proof in Appendix B.1).

With all discussed above, the learning progress of Zipper can be constructed by decending the gradient of $\boldsymbol{\theta}_r, \boldsymbol{\theta}_n$ and mixing both of them. The calculating steps in Algorithm 1 can be summarized in Eq. 7.

$$\begin{cases} \boldsymbol{\theta}_n^t = \mathcal{Z}_n \left[ \mathbb{E}_{(x,y)\sim\mathcal{D}_1} (\nabla_{\boldsymbol{\theta}_n} \ell_1(x, y; \boldsymbol{\theta}_n^{t-1})), \tau_1 \right] \\ \boldsymbol{\theta}_r^t = \mathcal{Z}_r \left[ (\mathbb{E}_{(x',y)\sim\mathcal{D}_2} \nabla_{\boldsymbol{\theta}_r} \ell_2(x', y; \boldsymbol{\theta}_r^{t-1})), \tau_2 \right] \\ \boldsymbol{\theta}_g^{t+1} = \alpha' \boldsymbol{\theta}_g^{t-1} + (1-\alpha')[\gamma \boldsymbol{\theta}_r^t + (1-\gamma)\boldsymbol{\theta}_n^t] \\ \boldsymbol{\theta}_n^t = \mathcal{B}(t, t', c)\boldsymbol{\theta}_g^{t+1} + (1 - \mathcal{B}(t, t', c))\boldsymbol{\theta}_n^t \\ \boldsymbol{\theta}_r^t = \mathcal{B}(t, t', c)\boldsymbol{\theta}_g^{t+1} + (1 - \mathcal{B}(t, t', c))\boldsymbol{\theta}_r^t, \end{cases} \tag{7}$$

where $\mathcal{B}(t, t', c)$ is a Boolean function that returns one only when both $t \geq t'$ and $t \mod c == 0$, otherwise it returns zero. $\mathcal{Z}_n$ and $\mathcal{Z}_r$ are optimizers for natural training and adversarial training assignments.

### 3.4 Theoretical Analysis

In this part, we theoretically analyze how base learners help global learner in Zipper. For brevity, we omit the expectation notation over samples from each distribution without losing generalization.

**Definition 1.** ***(Tradeoff Regret with Mixed Strategies)*** *For the natural training assignment $a_1$ and adversarial training assignment $a_2$, consider an algorithm generates the trajectory of states $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ for two base learners, the regret of both base learners on its corresponding loss function $\ell_1$, $\ell_2$ is*

$$\mathbf{R}_T = \frac{1}{2} \sum_{a=1}^{2} \left( \sum_{t=1}^{T} \ell_a \left( \boldsymbol{\theta}_a^t \right) - \inf_{\boldsymbol{\theta}_a^t \in \Theta} \sum_{t=1}^{T} \ell_a \left( \boldsymbol{\theta}_a^\star \right) \right). \tag{8}$$

The oracle state $\boldsymbol{\theta}_a^\star$ represents the theoretically optimal parameters for each task $a$. $\mathbf{R}_T$ is the sum of the difference between the parameters of each base learner and the theoretically optimal parameters for each task. Based on the definition, we can give the following upper bound on the expected error of classifier trained by Zipper with respect to $\mathbf{R}_T$ as:

**Theorem 1.** *(Proof in Appendix B.2) Consider an algorithm with regret bound $R_T$ that generates the trajectory of states for two base learners, for any parameter state $\boldsymbol{\theta} \in \Theta$, given a sequence of convex surrogate evaluation functions $\ell : \Theta \mapsto [0, 1]_{a\in\mathcal{A}}$ drawn i.i.d. from some distribution $\mathcal{L}$, the expected error of the global learner $\boldsymbol{\theta}_g$ on both tasks over the test set can be bounded with probability at least $1 - \delta$:*

$$\mathop{\mathbb{E}}_{\ell\sim\mathcal{L}} \ell \left( \boldsymbol{\theta}_g \right) \leq \mathop{\mathbb{E}}_{\ell\sim\mathcal{L}} \ell \left( \boldsymbol{\theta} \right) + \frac{\mathbf{R}_T}{T} + 2\sqrt{\frac{2}{T} \log \frac{1}{\delta}}. \tag{9}$$

So the above inequality indicates that any strategy beneficial to reducing the error of each task that makes $\mathbf{R}_T$ smaller will decrease the error bound of the global learner. Considering Zipper divides the tradeoff problem into two independent tasks, Theorem 1 guarantees the upper bound of the risks given by the global learner trained by Zipper will get lower once the error for each task becomes lower. In practice, we can apply customized learning rate strategies, optimizers, and weight averaging to guarantee the error reduction of each base learner.

## 4 Experiments

We conduct a series of experiments on ResNet-18 (He et al., 2016) and WRN-32-10 (Zagoruyko & Komodakis, 2016) on benchmark datasets MNIST, SVHN, CIFAR-10, CIFAR-100, and TinyImageNet. For simplicity, we only report the results based on $L_\infty$ norm for the non-targeted attack (results against the $L_2$ adversary are in Appendix A.5).

**Baselines.** We select six approaches to compare with: AT using PGD ($\beta = 1$ in Eq. 1) (Madry et al., 2018), AT using the half-half loss ($\beta = 1/2$ in Eq. 1) (Goodfellow et al., 2015), TRADES with different $\lambda$, Friendly Adversarial Training (FAT) (Zhang et al., 2020), Interpolated Adversarial Training (IAT) (Lamb et al., 2019), and Robust Self Training (RST) (Raghunathan et al., 2020) used labeled data for fair comparison. For Zipper, we set $t' = 75$ and the optimal mixing strategy will be discussed in Section 4.2.1.

Table 1: Comparison of our algorithm with different training methods using ResNet-18 and WRN-32-10 on CIFAR-10. The maximum perturbation is $\varepsilon = 8/255$. The best checkpoint is selected based on the tradeoff between clean accuracy and robust accuracy against PGD20 on the test set. We highlight the top two results on each task. We omit standard deviations of Zipper as they are very small ($< 0.5\%$). Average accuracy rates (in %) have shown that the proposed Zipper method greatly mitigate the tradeoff of the model.

(a) The evaluation results based on ResNet-18.

| Method | NAT | PGD20 | PGD100 | MIM | CW | $\mathrm{APGD}_{ce}$ | $\mathrm{APGD}_{dlr}$ | $\mathrm{APGD}_{t}$ | $\mathrm{FAT}_{t}$ | Square | AA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NT | **93.04** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| AT ($\beta = 1$) | 84.32 | 48.29 | 48.12 | 47.95 | 49.57 | **47.47** | 48.57 | 45.14 | 46.17 | 54.21 | 44.37 |
| AT ($\beta = 1/2$) | 87.84 | 44.51 | 44.53 | 47.30 | 44.93 | 40.58 | 42.55 | 40.20 | 44.56 | 50.76 | 40.06 |
| TRADES ($\lambda = 6$) | 83.91 | **54.25** | **52.21** | **55.65** | **52.22** | **53.47** | **50.89** | **47.93** | **48.53** | **55.75** | **48.20** |
| TRADES ($\lambda = 1$) | 87.88 | 45.58 | 45.60 | 47.91 | 45.05 | 42.95 | 42.49 | 40.32 | 43.89 | 53.49 | 40.38 |
| FAT | 87.72 | 46.69 | 46.81 | 47.03 | 49.66 | 46.20 | 47.51 | 44.88 | 45.76 | 52.98 | 43.14 |
| IAT | 84.60 | 40.83 | 40.87 | 43.07 | 39.57 | 37.56 | 37.95 | 35.13 | 36.06 | 49.30 | 35.14 |
| RST | 84.71 | 44.23 | 44.31 | 45.33 | 42.82 | 41.25 | 42.01 | 40.41 | 46.54 | 50.49 | 37.68 |
| Zipper | **89.09** | **50.01** | **50.00** | **52.19** | **50.04** | 46.53 | **48.70** | **45.37** | **47.32** | **56.68** | **46.07** |

(b) The evaluation results based on WRN-32-10.

| Method | NAT | PGD20 | PGD100 | MIM | CW | $\mathrm{APGD}_{ce}$ | $\mathrm{APGD}_{dlr}$ | $\mathrm{APGD}_{t}$ | $\mathrm{FAT}_{t}$ | Square | AA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NT | **93.30** | 0.01 | 0.02 | 0.05 | 0.00 | 0.00 | 0.00 | 0.00 | 0.87 | 0.28 | 0.00 |
| AT ($\beta = 1$) | 87.32 | 49.01 | 48.83 | 48.25 | 52.80 | 48.83 | 49.00 | 46.34 | 46.11 | 54.26 | 48.17 |
| AT ($\beta = 1/2$) | 89.27 | 48.95 | 48.86 | 51.35 | 49.56 | 45.98 | 47.66 | 44.89 | 46.42 | 56.83 | 44.81 |
| TRADES ($\lambda = 6$) | 85.11 | **54.58** | **54.82** | **55.67** | **54.91** | **54.89** | **55.5** | **52.71** | **52.61** | **57.62** | **52.19** |
| TRADES ($\lambda = 1$) | 87.20 | 51.33 | 51.65 | 52.47 | 53.19 | 51.60 | 51.88 | 49.97 | 50.01 | 54.83 | 49.81 |
| FAT | 89.65 | 48.74 | 48.69 | 48.24 | 52.11 | 48.50 | 48.81 | 46.17 | 44.73 | 51.51 | 46.70 |
| IAT | 87.93 | 50.55 | 50.72 | 52.37 | 48.71 | 47.71 | 46.55 | 43.84 | 45.78 | 56.52 | 43.85 |
| RST | 87.27 | 46.55 | 46.76 | 47.02 | 45.99 | 45.73 | 46.58 | 41.52 | 43.18 | 52.44 | 45.78 |
| Zipper | **91.03** | **56.88** | **56.92** | **58.87** | **57.23** | **53.94** | **55.80** | **53.00** | **53.65** | **63.10** | **52.91** |

**Evaluation.** To evaluate the robustness of the proposed method, we apply several adversarial attacks including PGD (Madry et al., 2018), MIM (Dong et al., 2018), CW (Carlini & Wagner, 2017), AutoAttack (AA) (Croce & Hein, 2020) and all its components ($\mathrm{APGD}_{ce}$, $\mathrm{APGD}_{dlr}$, $\mathrm{APGD}_{t}$, $\mathrm{FAB}_{t}$ and Square).

### 4.1 The Tradeoff Performance on Benchmark Datasets

To comprehensively manifest the power of our Zipper method, we present the results of both ResNet-18 and WRN-32-10 on CIFAR-10 in Table 1. In Table 1(a), Zipper consistently improves standard test error relative to models trained by several robust methods, while maintaining adversarial robustness at the same level. More specifically, Zipper achieves the second highest standard accuracy of 89.09% (only lower than 93.04% obtained by natural training (NT)), while meantime robust accuracy against AA is 46.07%, hanging on to 48.2% from TRADES. If we force TRADES to meet the same level of clean accuracy as Zipper (89%), the robustness of TRADES against APGD will drop to 30% (see TRADES in Appendix A.6), which is significantly worse than Zipper. That means it is hard to obtain acceptable robustness but maintain clean accuracy above 89% in the joint training framework even if it is equipped with an advanced loss function, while the improvement of Zipper is notable since we only use the naive cross-entropy loss. Contrary to FAT managing the tradeoff through adaptively decreasing the step size of PGD, which still hurts robustness a lot, Zipper is the only method with clean accuracy above 89% and robust accuracy against AA above 46%. We should emphasize the final obtained model of Zipper is the *same size* as other trained models are. For the training time, Zipper does perform both NT and naive AT but the cost of NT is negligible, so the overhead of Zipper is smaller than TRADES, and whatever serial and parallel versions of Zipper are even *faster* than TRADES (see Appendix A.6).

Things become more obvious when it comes to WRN-32-10. In Table 1(b), the gap between test natural accuracy of Zipper and NT is reduced to 2.27%, a relative decrease of 3.65% in standard test error as compared to the second highest natural accuracy (except NT) achieved by FAT. It is also remarkable that the boost of accuracy does not hurt the robustness of Zipper, instead, Zipper even outperforms TRADES across multiple types of adversarial attacks. In particular, we find that Zipper has a standard test error of 6.7% while TRADES with $\lambda = 6$ has a standard test error of 14.89% only. And the improved robustness of Zipper among PGD20/100, MIM, CW, $\mathrm{FAT}_{t}$ and Square is conspicuous. Besides, the best performance on AA, which is an ensemble of different attacks and the most powerful adaptive adversarial attack so far, demonstrates the reliability of Zipper. Likewise, only Zipper attains robust accuracy of AA higher than 52% along with clean accuracy higher than 90%. It should be emphasized that these features confirm the practicability of Zipper. In short, Zipper

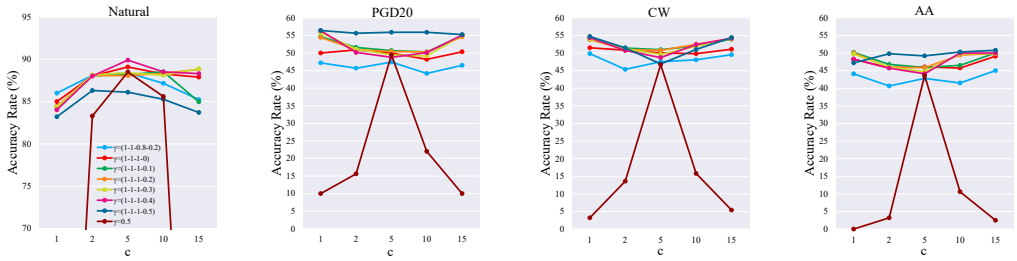



Figure 2: Zipper with different mixing ratio strategies and various values of frequency on CIFAR-10. We evaluate both natural accuracy and robustness against PGD20, C&W and AA attacks using ResNet-18.

has consistently improved robustness without loss of natural accuracy. For more results on benchmark datasets like MNIST, SVHN, CIFAR-100, and TinyImageNet, please refer to the results in Appendix A.2 - A.4.

### 4.2 Comprehensive Understanding of Zipper

We run a number of ablations to analyze the Zipper framework in this part. If not specified otherwise, the experiments are conducted on CIFAR-10 using ResNet-18.

#### 4.2.1 Selection of Communication Frequency and Mixing Strategies

As illustrated in Algorithm 1, two factors control the tradeoff between accuracy and robustness of the global learner: *frequency of communication* $c$ and *mixing ratio* $\gamma$. So there must exist an optimal tradeoff between communication frequency $c$ and mixing rate $\gamma$. In this part, we empirically investigate how these factors affect the performance of Zipper. For $\gamma$, we do not explicitly assign a fixed value to it. In practice, we set several breakpoints and compute $\gamma$ by the piecewise linear function of these points. As for $c$, with the fixed mixing ratio strategy, we sweep over the frequency of communication from 1 to 15.

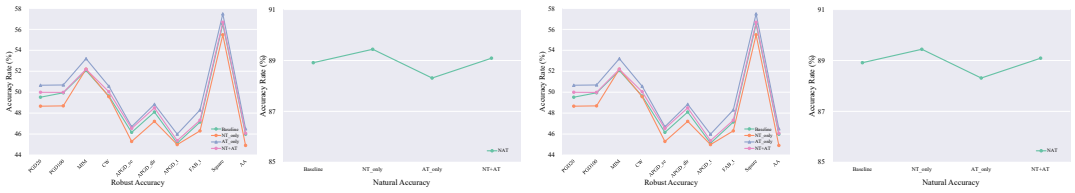
Results are shown in Figure 2, and we have the following observations: Intuitively, a larger $c$ means base learners communicate with the global learner less frequently to get the initialization, so they barely have the opportunity to move alternately towards two optimal solution manifolds. But specifically, the natural accuracy falls back down after reaching the peak while the robust accuracy in different adversarial settings roughly shows a trough. Such observation manifests that too much/little communication has a negative influence on standard accuracy but results in relatively higher robustness. It captures a tradeoff between natural and robust errors with respect to $c$. We also deeply investigate the independence of different base learners when communication occurs, please refer to Appendix C.1. Similarly, $\gamma$ controls the tradeoff via balancing the contribution of individuals to the global learner when base learners are gradually well trained. Note that $\gamma$ is a scalar but we dynamically adjust the value along the training process using a piecewise linear function to decrease (the setting of increasing $\gamma$ is in Appendix A.9). The numbers in brackets are the values at the 0/40/80/120-th epoch. If $\gamma$ gets smaller, the base learner in charge of natural classification has a pronounced influence on the global learner. Among all configurations, the best one is to apply $\gamma = (1, 1, 1, 0)$ and $c = 5$ to the global learner after the 75th epoch. When compared to strategies that $\gamma$ decays during late periods, $\gamma = (1, 1, 0.8, 0.2)$ shows lower standard and robust accuracy, confirming that more sophisticated initialization could be useful for both accuracy and robustness. With the increase of the last breakpoint of dynamical strategies, the robust accuracy gradually increases; while the standard accuracy decreases by a small margin. We also investigate the static/dynamic strategy for $\gamma$. By observing $\gamma = 0.5$ and $\gamma = (1, 1, 1, 0.5)$, the scheduled mixing strategy makes Zipper more robust to various attacks.

#### 4.2.2 Customized Policies for Individuals

As stressed in the preceding paragraphs, one of the major advantages of Zipper in comparison with the standard joint training framework is that each base learner enables to customize the corresponding strategy for their own tasks freely rather than using the same strategy for all tasks. In this part, we investigate whether Zipper performs better when cooperating with diverse techniques.

**Weight Averaging.** Recent works (Rebuffi et al., 2021; Izmailov et al., 2018; Wang & Wang, 2022) have shown that weight averaging (WA) greatly improves both natural and robust generalization. The average parameters of all history model snapshot through the training process to build an ensemble model on the fly. However, such technique cannot benefit both accuracy and robustness in the joint training framework. Therefore, we introduce WA into base learners separately. Results are shown in Figure 3 (a). We employ WA in either NT (NT_only) or AT (AT_only) or both of them (NT+AT). We also evaluate Zipper without EMA in Appendix A.8. Overall, the results confirm that the performance of the global learner can be further improved after both base learners exploit WA. But unfortunately, an obvious tradeoff happens if only one of the base learner is equipped with WA. For instance, the standard test accuracy of NT_only continues to increase at the expense of the drop in the ability to defend attacks. A likely reason is that WA implicitly controls the learning speed of base learners. Actually, the base learner with WA becomes an expert much faster than the one without WA in its sub-task, meaning the fast one is not in accordance with the slow one. This result is important because it not only illustrates the potential of Zipper comes from its base learners but also identifies a key challenge of tradeoff for future improvement.

**Different Optimizers.** We also investigate the effect of optimizers designed for different tasks. We choose AT ($\beta = 1$) using SGD with momentum and Adam for piecewise learning rate schedule optimized by joint training as the baseline. The initial learning rate for Adam is 0.0001. We alternately apply these two optimizers in each subproblem. The comparison of the results is shown in Figure 3 (b). We can see that the gap of robust accuracy between models adversarially trained by Adam and the ones trained by SGD is significant. All three schemes equipped with Adam, namely NT (Adam)+AT (Adam), NT (SGD)+AT (Adam), and Baseline (Adam), perform worse than the ones using SGD when evaluated by adversarial attacks. But on the other hand, by comparing the results of Baseline (Adam) and NT (Adam)+AT (SGD), it confirms a proper optimization scheme with respect to data distribution can effectively benefit the corresponding performance without overlooking the other. That not only demonstrates the necessity of Zipper to decouple task-aware assignments from joint training but also indicates using Adam may not be the principal reason for robustness drop. It is just ill-suited for the outer and inner optimization in AT. Besides, though the best results still come from using SGD, the learning rate for different tasks can be customized which is not feasible in the joint framework, as shown in Appendix A.7.




(a) Weight Averaging (b) Different Optimizers

Figure 3: (a) We apply weight averaging to one of the base learner or both of them. Results demonstrate that using weight averaging through training can bring performance boost in its corresponding sub-task, and thus has an effect on predictions of the global learner. (b) Base learners of Zipper optimized by different optimizers. The optimal selection is using Adam for the natural classification task but maintaining SGD for the adversarial one.

### 4.2.3 Where does Zipper Step Forward?

Considering the proposed method achieves impressive clean accuracy without a harsh drop in robustness, it is naturally curious about what improvements Zipper has secured in comparison with robust methods in detail. Thus, we further investigate the predictions that robust classifiers are prone to make. As shown in Figure 4, we provide two perspectives to analyze the differences that classifiers trained by different AT methods.

To broadly study the case, we perform experiments on NT, TRADES with different $\lambda$, FAT and Zipper, then plot the distribution of the correct predictions of all methods for each class in Figure 4(a). As evident at first glance, we note that animals are more frequently misclassified, especially cats/dogs in the natural scenario and cats/deers in the adversarial scenario. In addition, the classifier trained by standard natural training does not always outperform the ones adversarially trained. Actually, they


(a) (b)

Figure 4: Analyses of predilections that different robust classifiers have on CIFAR-10 using ResNet-18. (a) Distribution of the correct predictions of different training methods for each class. We separate out results on natural examples from adversarial ones (AA). Note that results of 'NT Adv' does not appear in the figure just because they are literally zero. (b) Visualization of samples that other methods misclassify while Zipper makes right predictions.

are equally skilled at most categories and the outcome is decided by specific categories (e.g. birds, cats and dogs). Zipper keeps pace with NT in the natural task, and meanwhile promotes the higher improvements in difficult items (e.g. cats and deers) against AA attack.

In Figure 4(b), we display specific samples in the testing dataset that are misclassified by robust classifiers (TRADES and FAT) but recognized by our proposed method, including both natural examples (the first two rows) and adversarial examples (the last two rows). We also provide typical failure cases of Zipper while TRADES or FAT correctly classifies in Appendix C.2. Here, images shown in the first row are *easy* ones where the foreground objects stand out from the clear backgrounds, while *hard* samples are referred to those having confused objects with messy backgrounds. It is worth noting that TRADES delivers poor performances not only on hard examples with complex backgrounds or obscured objects but also on simple ones. For example, each image in the first row is typically plain and regular, however, TRADES fails in categorizing them into the right class. A plausible explanation for the issue is that TRADES lacks in a set of support measures specially devised for the natural classification task unlike Zipper does, highlighting design differentiation for sub-tasks is necessary.

Another interesting finding is that though both TRADES and FAT can build a robust classifier, they still rely on spurious background information and thus are easily deceived when encountering images with similar backgrounds but different objects. This phenomenon can be verified from the misclassification of the fourth and fifth images in the first row (taking white/blue backgrounds as evidence), and the fifth image in the fourth row (confused by the green background). But Zipper has the ability to sift the invariant feature of the foreground object while ignoring the background information spuriously correlated with the categories in both natural and adversarial settings. On the whole, Zipper demonstrates its strength to differentiate difficult samples close to the decision boundary and its potential to learn a background-invariant classifier.

## 5 Conclusion

In this paper, we proposed a bi-expert framework for improving the tradeoff issue between natural and robust generalization, named Zipper, which trains two base learners responsible for complementary fields and collects their parameters to construct a global learner. By decoupling from the joint training paradigm, each base learner can wield customized strategies based on data distribution. We also provide the theoretical analysis to justify the effectiveness of task-aware strategies and extensive experiments show that Zipper better mitigates the tradeoff of accuracy and robustness.

## REFERENCES


Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? In *NeurIPS*, 2019.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19:357–367, 1967.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. In *COLT*, 2001.

Hakan Bilen and Andrea Vedaldi. Integrated perception with recurrent multi-task neural networks. In *NeurIPS*, 2016.

Jose H. Blanchet and Karthyek R. A. Murthy. Quantifying distributional model risk via optimal transport. *Math. Oper. Res.*, 44(2):565–600, 2019.

Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *SP*, 2017.

Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C. Duchi, and Percy Liang. Unlabeled data improves adversarial robustness. In *NeurIPS*, 2019.

Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Trans. Inf. Theory*, 50(9):2050–2057, 2004.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *CVPR*, 2018.

Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Mach. Learn.*, 107(3):481–508, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Alex Lamb, Vikas Verma, Juho Kannala, and Yoshua Bengio. Interpolated adversarial training: Achieving robust neural networks without sacrificing too much accuracy. In *ACM AISec Workshop*, 2019.

Saehyung Lee, Hyungyu Lee, and Sungroh Yoon. Adversarial vertex mixup: Toward better adversarially robust generalization. In *CVPR*, 2020.

Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Defense against adversarial attacks using high-level representation guided denoiser. In *CVPR*, 2018.

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogério Schmidt Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *CVPR*, 2017.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Amir Najafi, Shin-ichi Maeda, Masanori Koyama, and Takeru Miyato. Robustness to adversarial perturbations in learning from incomplete data. In *NeurIPS*, 2019.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018.

Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *SP*, 2016.

Yao Qin, Nicholas Frosst, Sara Sabour, Colin Raffel, Garrison W. Cottrell, and Geoffrey E. Hinton. Detecting and diagnosing adversarial images with class-conditional capsule reconstructions. In *ICLR*, 2020.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Adversarial training can hurt generalization. *CoRR*, abs/1906.06032, 2019.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C. Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. In *ICML*, 2020.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A. Calian, Florian Stimberg, Olivia Wiles, and Timothy A. Mann. Fixing data augmentation to improve adversarial robustness. *CoRR*, abs/2103.01946, 2021.

Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *AAAI*, 2018.

Hasim Sak, Andrew W. Senior, Kanishka Rao, and Françoise Beaufays. Fast and accurate recurrent neural network acoustic models for speech recognition. In *INTERSPEECH*, 2015.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *ICLR*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

C. Villani. Topics in optimal transportation. 2003.

Hongjun Wang and Yisen Wang. Self-ensemble adversarial training for improved robustness. In *ICLR*, 2022.

Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *NDSS*, 2018.

Yongxin Yang and Timothy M. Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *ICLR*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019.

Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan S. Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *ICML*, 2020.

# A ADDITIONAL EXPERIMENTS

## A.1 DETAILED CONFIGURATIONS

All images are normalized into $[0, 1]$. We train ResNet-18 using SGD with 0.9 momentum for 120 epochs (200 epochs for CIFAR-100) and the weight decay factor is set to $3.5e^{-3}$ for ResNet-18 and $7e^{-4}$ for WRN-32-10. We use the piecewise linear learning rate strategy for performing weight averaging in base learners. For the base learner of AT, the initial learning rate for ResNet-18 is set to 0.01 and 0.1 for WRN-32-10 till Epoch 40 and then linearly reduced by 10 at Epoch 60 and 120, respectively. The magnitude of maximum perturbation at each pixel is $\varepsilon = 8/255$ with step size $\kappa = 2/255$ and the PGD steps number in the inner maximization is 10. For the base learner of NT, we fix the initial learning rate as 0.1 and the weight decay is $5e^{-4}$ for both ResNet-18 and WRN-32-10.

## A.2 EXPERIMENTS ON MNIST/SVHN

We conducted experiments on MNIST ($\varepsilon = 0.3$) and SVHN using ResNet-18 with the same setup in Sec. A.1. We ran 5 individual trials and results with standard deviations are shown Table 2.

Table 2: Comparison of our algorithm with different training methods using ResNet-18 on MNIST and SVHN. The maximum perturbation is $\varepsilon = 8/255$. The best checkpoint is selected based on the tradeoff between clean accuracy and robust accuracy against PGD20 on the test set. We highlight the top two results on each task. Average accuracy rates (in %) have shown that the proposed Zipper method greatly mitigate the tradeoff of the model.

| | MNIST | | | SVHN | | |
|---|---|---|---|---|---|---|
| Methods | NAT | PGD20 | AA | NAT | PGD20 | AA |
| TRADES | 99.07 ±0.13 | 94.45 ±0.07 | 92.17 ±0.21 | 93.1 ±0.25 | **55.38 ±0.71** | **45.52 ±0.37** |
| FAT | 99.18 ±0.03 | 93.54 ±0.1 | 90.04 ±0.68 | 93.87 ±0.4 | 53.61 ±0.88 | 40.92 ±0.29 |
| Zipper | **99.24 ±0.07** | **96.14 ±0.15** | **92.3 ±0.3** | **94.11 ±0.27** | **55.29 ±0.23** | **45.41 ±0.26** |

## A.3 EXPERIMENTS ON CIFAR-100

To further demonstrate our proposal achieve a better tradeoff between accuracy and robustness, we also conduct experiments on CIFAR-100 datasets. Here we still use ResNet-18 as the backbone model with the same configurations as claimed in Sec. A.1. We report the results of natural accuracy and several advanced adversarial attack methods in Table 3. Note that we do not design a specialized strategy for Zipper on CIFAR-100 but Zipper still achieves a gratifying tradeoff, so it still has the potential to perform better.

Table 3: Comparison of our algorithm with different training methods using ResNet-18 on CIFAR-100. The maximum perturbation is $\varepsilon = 8/255$. The best checkpoint is selected based on the tradeoff between clean accuracy and robust accuracy against PGD20 on the test set. We highlight the top two results on each task. Average accuracy rates (in %) have shown that the proposed Zipper method greatly mitigate the tradeoff of the model.

| Method | NAT | PGD20 | PGD100 | MIM | CW | $\mathrm{APGD}_{ce}$ | $\mathrm{APGD}_{dlr}$ | $\mathrm{APGD}_{t}$ | $\mathrm{FAT}_{t}$ | Square | AA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NT | **65.74** | 0.02 | 0.01 | 0.02 | 0.01 | 0.00 | 0.00 | 0.00 | 0.07 | 0.37 | 0.00 |
| AT ($\beta = 1$) | 60.10 | 28.22 | 28.27 | 28.31 | 24.87 | 26.63 | 24.13 | 21.98 | 23.87 | 27.93 | 23.91 |
| AT ($\beta = 1/2$) | 60.84 | 22.64 | 22.61 | 23.86 | 22.28 | 20.66 | 21.67 | 19.17 | 20.09 | 25.36 | 19.2 |
| TRADES ($\lambda = 6$) | 59.93 | **29.90** | **29.88** | **29.55** | **26.14** | **27.93** | **25.43** | **23.72** | **25.16** | **30.03** | **24.72** |
| TRADES ($\lambda = 1$) | 60.18 | 28.93 | 28.91 | 29.12 | 25.79 | 27.07 | 25.00 | 23.22 | 24.31 | 28.76 | 23.65 |
| FAT | 61.71 | 22.93 | 22.87 | 22.64 | 23.45 | 24.78 | 24.91 | 20.56 | 23.16 | 26.37 | 20.01 |
| IAT | 57.04 | 21.40 | 21.39 | 22.37 | 19.18 | 19.63 | 18.92 | 15.50 | 16.63 | 23.26 | 15.50 |
| RST | 60.30 | 23.56 | 23.61 | 23.71 | 22.40 | 24.69 | 24.18 | 21.66 | 23.82 | 27.05 | 21.18 |
| Zipper | **62.97** | **29.48** | **29.49** | **30.35** | **27.77** | **27.45** | **27.42** | **23.96** | **25.54** | **31.41** | **24.04** |

## A.4 EXPERIMENTS ON TINYIMAGENET & TINYIMAGENET-C

We also perform empirical evaluation on more complex datasets. We adversarially train models on TinyImageNet and add the robustness evaluation on TinyImageNet-C instead. We still use ResNet18

as the baseline model to run experiments and the hyperparameters, such as learning rate and size of attacks, are kept the same as for CIFAR datasets. We denote NAT as the classification accuracy on the original clean testing images. Following the common setting in Hendrycks & Dietterich (2019), we define natural robustness (NROB) as the average classification accuracy over all 15 corruptions, to evaluate the robustness of trained models on TinyImageNet-C. Besides, we also report adversarial robustness against various attacks. Results are shown in Table 4. Note that we only report the results of TRADES ($\lambda = 1$), for it outperforms other versions of TRADES on the tradeoff performance which our proposed method focuses on. From Table 4, we see that our method improves robust accuracy against natural corruptions by a considerable margin compared with AT ($\beta = 1$) and TRADES ($\lambda = 1$) and maintains excellent performance on clean and adversarial samples.

Table 4: Clean and robust accuracy (%) on TinyImageNet dataset with ResNet18.

| | NAT | NROB | PGD20 | PGD100 | MIM | CW | $\mathrm{APGD}_{ce}$ | $\mathrm{APGD}_{dlr}$ | $\mathrm{APGD}_{t}$ | $\mathrm{FAB}_{t}$ | Square | AA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NT | 57.31 | 20.17 | 0.00 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| AT ($\beta = 1$) | 41.10 | 21.93 | 17.68 | 17.22 | 17.39 | 15.65 | 13.19 | 15.28 | 14.12 | 15.28 | 23.1 | 15.44 |
| TRADES ($\lambda = 1$) | 43.28 | 22.45 | 23.21 | 23.24 | 23.01 | 22.89 | 15.99 | 17.17 | 17.01 | 17.68 | 24.55 | 17.76 |
| Zipper | **48.1** | **24.88** | **24.57** | **23.24** | **23.43** | **23.27** | **16.2** | **17.53** | **17.16** | **18.01** | **26.11** | **19.03** |

### A.5 Results for training with $l_2$ attacks

We adversarially train models using AT ($\beta = 1$), TRADES ($\lambda = 1$) and Zipper against the $l_2$ adversary on CIFAR-10. For $l_2$ threat model, we fix magnitude of perturbation $\varepsilon = 128/255$; step size $\kappa = 15/255$, which is a standard setting for adversarial training in Madry et al. (2018). We report the results in Table 5. Our Zipper still shows its effectiveness when facing the $l_2$ threat model.

Table 5: Clean and robust accuracy (%) on CIFAR-10 dataset using ResNet18 against the $l_2$ adversary.

| | NAT | PGD20 | PGD100 | MIM | CW | $\mathrm{APGD}_{ce}$ | $\mathrm{APGD}_{dlr}$ | $\mathrm{APGD}_{t}$ | $\mathrm{FAB}_{t}$ | Square | AA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AT ($\beta = 1$) | 88.27 | 64.43 | 64.35 | 63.10 | 60.77 | 62.19 | 62.03 | 62.51 | 62.81 | 72.91 | 62.71 |
| TRADES ($\lambda = 1$) | 89.06 | 68.3 | 68.41 | 67.99 | 67.03 | 66.18 | 66.23 | 66.37 | 66.58 | 77.91 | 66.26 |
| Zipper | **91.39** | **75.91** | **75.99** | **75.33** | **74.18** | **74.54** | **72.69** | **72.48** | **72.77** | **82.01** | **72.1** |

### A.6 Computational Cost of Zipper

We compute the actual training time of TRADES and Zipper (serial/parallel version) using ResNet18 on RTX 3090 GPU in Table 6. We also report the standard deviations over 5 runs to show the sensitivity of Zipper. Neither version of Zipper is slower than TRADES. Zipper does perform both NT and naive AT, but the cost of NT is negligible so the overhead (NT+AT) is smaller than TRADES.

Table 6: Evaluation of time complexity of different training methods using ResNet-18 on CIFAR-10.

| Method | NAT | PGD100 | APGD | Training Time (mins) |
|---|---|---|---|---|
| TRADES | 89.91 $\pm$0.69 | 34.25 $\pm$0.56 | 30.20 $\pm$0.81 | 414 |
| Zipper (Serial) | 89.11 $\pm$0.23 | 50.12 $\pm$0.12 | 46.12 $\pm$0.11 | 397 |
| Zipper (Parallel) | 89.09 $\pm$0.34 | 50.00 $\pm$0.44 | 46.53 $\pm$0.3 | **342** |

### A.7 Influence of Learning Rate

In this part, we also study the influence of the learning rate for different distribution-aware tasks. For simplicity, we set $t'$, $\gamma$ and $c$ as their best options according to the main body of the paper. We search the most grid of learning rate configurations in the range of 0.1, 0.01, 0.001 for both natural training and adversarial training.

Table 7: Comparison of different learning rate for base learners. The optimal strategy for natural accuracy occurs when NT=0.1 and AT=0.001, while the most robust model is trained using NT=0.01 and AT=0.01. This is not surprising because either of them is the optimal learning rate configuration in its own domain, but each of them obtains performance boost in its complementary task due to communication mechanism in our Zipper framework.

| | NAT | AA |
|---|---|---|
| NT=0.1, AT=0.01 | 89.09 | 46.07 |
| NT=0.1, AT=0.1 | 90.12 | 41.86 |
| NT=0.1, AT=0.001 | **90.45** | 43.55 |
| NT=0.01, AT=0.01 | 88.4 | **48.03** |
| NT=0.01, AT=0.1 | 88.25 | 42.98 |

### A.8 Influence of EMA

We investigate the performance of Zipper without using the EMA technique on the global learner but keep other settings unchanged (denoted as Zipper w/o EMA). Results are shown in Table 8. When compared with the version of using the EMA technique on the global learner, removing EMA from the global learner indeed brings performance decay. In conclusion, we recommend equipping Zipper with the EMA technique for the global learner since it does not bring too much computational cost.

Table 8: Clean and robust accuracy (%) on CIFAR-10 dataset using ResNet18 with or without EMA.

| | NAT | PGD20 | PGD100 | MIM | CW | $\mathrm{APGD}_{ce}$ | $\mathrm{APGD}_{dlr}$ | $\mathrm{APGD}_{t}$ | $\mathrm{FAB}_{t}$ | Square | AA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| TRADES ($\lambda = 1$) | 87.88 | 45.58 | 45.60 | 47.91 | 45.05 | 42.95 | 42.49 | 40.32 | 43.89 | 53.49 | 40.38 |
| Zipper w/o EMA | **88.33** | **49.13** | **49.14** | **51.68** | **49.80** | **45.36** | **47.88** | **44.25** | **45.24** | **55.89** | 44.91 |
| Zipper with EMA | **89.09** | **50.01** | **50.00** | **52.19** | **50.04** | **46.53** | **48.70** | **45.37** | **47.32** | **56.68** | **46.07** |

### A.9 Influence of different strategies of $\gamma$

We apply two kinds of increasing strategies for $\gamma$ by using a piecewise linear function, including $\gamma = (0.25, 0.5, 0.75, 1)$ and $\gamma = (0.75, 0.75, 0.75, 1)$. Results are shown in Table 9. Because $\gamma$ controls the contribution of the base learner for the adversarial task, gradually increasing $\gamma$ will lead the global learner to better robustness than the baseline Zipper model. Higher $\gamma$ in the later stage of training means that adversarial learner brings more robustness to the global learner. Likewise, when $\gamma$ becomes smaller and smaller, clean accuracy of the global learner would be improved with relatively high robustness. Overall, increasing strategy leads to relatively better robustness while the decreasing one brings better tradeoff performance.

Table 9: Clean and robust accuracy (%) on CIFAR-10 dataset using ResNet18 when $\gamma$ decreases/increases by using a piecewise linear function.

| | NAT | PGD20 | PGD100 | MIM | CW | $\mathrm{APGD}_{ce}$ | $\mathrm{APGD}_{dlr}$ | $\mathrm{APGD}_{t}$ | $\mathrm{FAB}_{t}$ | Square | AA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Zipper (Basline, $\gamma = (1, 1, 1, 0.5)$) | **89.09** | 50.01 | 50.00 | 52.19 | 50.04 | 46.53 | 48.70 | 45.37 | 47.32 | 56.68 | 46.07 |
| Zipper ($\gamma = (0.25, 0.5, 0.75, 1)$) | 85.33 | 52.48 | 52.46 | 54.26 | 52.16 | **49.64** | 50.76 | **47.85** | 48.49 | 57.49 | **47.58** |
| Zipper ($\gamma = (0.75, 0.75, 0.75, 1)$) | 86.01 | **52.56** | **52.50** | **54.54** | **52.32** | 49.56 | **50.98** | 47.80 | **48.73** | **57.62** | 47.15 |

## B Proofs of Theoretical Results

### B.1 Proof of Claim in Section 3.3

*Proof.* At epoch $t$, the parameters of the global learner are distributed to the experts and each expert train from this initialization with $c$ steps by calculating the gradients (e.g. using SGD optimizer). Following Nichol et al. (2018), we approximate the update performed by the initialization based on

the Taylor expansion:

$$\begin{aligned} g^{t+c} = \ell'\left(\boldsymbol{\theta}^{t+c}\right) &= \ell'\left(\boldsymbol{\theta}^{t}\right) + \ell''\left(\boldsymbol{\theta}^{t}\right)\left(\boldsymbol{\theta}^{t+c} - \boldsymbol{\theta}^{t}\right) + O\left(\left\|\boldsymbol{\theta}^{t+c} - \boldsymbol{\theta}^{t}\right\|^2\right) \\ &= \bar{g}^t + \bar{H}^t\left(\boldsymbol{\theta}^{t+c} - \boldsymbol{\theta}^{t}\right) + O\left(\tau^2\right)\big) \\ &= \bar{g}^t - \tau\bar{H}^t\sum_{j=t}^{t+c} g^j + O\left(\tau^2\right) \\ &= \bar{g}^t - \tau\bar{H}^t\sum_{j=t}^{t+c} \bar{g}^j + O\left(\tau^2\right). \end{aligned} \tag{10}$$

Recalling that $\mathcal{Z}^i$ represents an optimizer that updates the parameter vector at the $t$-th step: $\mathcal{Z}^i(\boldsymbol{\theta}, \tau) = \boldsymbol{\theta} - \tau\ell'(\boldsymbol{\theta})$. For each base learner, we approximate the gradient at intervals:

$$\begin{aligned} g_{val} = \frac{\partial}{\partial\boldsymbol{\theta}^t}\ell\left(\boldsymbol{\theta}^{t+c}\right) &= \frac{\partial}{\partial\boldsymbol{\theta}^t}\ell\left(\mathcal{Z}^{t+c-1}\left(\mathcal{Z}^{t+c-2}\left(\ldots\left(\mathcal{Z}^{t}\left(\boldsymbol{\theta}^{t}\right)\right)\right)\right)\right) \\ &= \mathcal{Z}'^{t}\left(\boldsymbol{\theta}^{t}\right)\cdots\mathcal{Z}'^{t+c-1}\left(\boldsymbol{\theta}^{t+c-1}\right)\ell'\left(\boldsymbol{\theta}^{t+c}\right) \\ &= \left(I - \tau\ell''\left(\boldsymbol{\theta}^{t}\right)\right)\cdots\left(I - \tau\ell''\left(\boldsymbol{\theta}^{t+c-1}\right)\right)\ell'\left(\boldsymbol{\theta}^{t+c}\right) \\ &= \left(\prod_{j=t}^{t+c-1}\left(I - \tau\ell''\left(\boldsymbol{\theta}^{j}\right)\right)\right) g^{t+c}. \end{aligned} \tag{11}$$

Replacing $\ell''\left(\boldsymbol{\theta}^{j}\right)$ with $\bar{H}^j$ and substituting $g^{t+c}$ for Eq. 10, we expand to leading order:

$$\begin{aligned} g_{val} &= \left(\prod_{j=t}^{t+c-1}\left(I - \tau\bar{H}^{j}\right)\right)\left(\bar{g}^{t+c} - \tau\bar{H}^{t+c}\sum_{j=t}^{t+c-1}\bar{g}^{j}\right) + O\left(\tau^2\right) \\ &= \left(I - \tau\sum_{j=t}^{t+c-1}\bar{H}^{j}\right)\left(\bar{g}^{t+c} - \tau\bar{H}^{t+c}\sum_{j=t}^{t+c-1}\bar{g}^{j}\right) + O\left(\tau^2\right) \\ &= \bar{g}^{t+c} - \tau\sum_{j=t}^{t+c-1}\bar{H}^{j}\bar{g}^{t+c} - \tau\bar{H}^{t+c}\sum_{j=t}^{t+c-1}\bar{g}^{j} + O\left(\tau^2\right) \end{aligned} \tag{12}$$

Therefore, we take the expectation of $g_{val}$ over steps, and obtain:

$$\mathbb{E}\left[g_{val}\right] = \mathbb{E}\left[\bar{g}^{t+c}\right] - \tau\mathbb{E}\left[\sum_{j=t}^{t+c-1}\bar{H}^{j}\bar{g}^{t+c} - \bar{H}^{t+c}\sum_{j=t}^{t+c-1}\bar{g}^{j}\right] + \mathbb{E}\left[O\left(\tau^2\right)\right] \tag{13}$$

Recalling that $\boldsymbol{\theta}_g$ is mixed by $\boldsymbol{\theta}_n$ and $\boldsymbol{\theta}_r$. For simplicity of exposition, we use $p$ and $q$ to stand for the scalar factors, meaning $\boldsymbol{\theta}_g = p\boldsymbol{\theta}_n + q\boldsymbol{\theta}_r$. Ignoring the higher order terms, for each expert initialized by the global learner (e.g. $\boldsymbol{\theta}_n$), we have:

$$\begin{aligned} \boldsymbol{\theta}_n = \boldsymbol{\theta}_g - \mathbb{E}_n\left[g_{val}\right] &= p\boldsymbol{\theta}_n + q\boldsymbol{\theta}_r - [\mathbb{E}\left[\bar{g}_n^{t+c}\right] + \tau_n\mathbb{E}\left[\sum_{j=t}^{t+c-1}\bar{H}^{j}\bar{g}_n^{t+c} - \bar{H}^{t+c}\sum_{j=t}^{t+c-1}\bar{g}_n^{j}\right]] \\ &= [p\boldsymbol{\theta}_n - \mathbb{E}\left[\bar{g}_n^{t+c}\right]] + [q\boldsymbol{\theta}_r - \tau_n\mathbb{E}\left[\bar{H}^{t+c}\sum_{j=t}^{t+c-1}\bar{g}_n^{j} - \sum_{j=t}^{t+c-1}\bar{H}^{j}\bar{g}_n^{t+c}\right]] \\ &= [p\boldsymbol{\theta}_n - \sum_{i=t}^{t+c-1}\bar{g}^{i}] + [q\boldsymbol{\theta}_r - \tau_n\sum_{i=t}^{t+c-1}\sum_{j=1}^{i-1}\bar{H}^{i}\bar{g}^{j}] \qquad (\text{for } c \geq 2). \end{aligned} \tag{14}$$

The first term pushes $\theta_n$ to move forward the minimum of its assigned loss over its data distribution; while the second one improves generalization by increasing the inner product between gradients of different minibatches and update the parameters from the other task. □

### B.2 PROOF OF THEOREM 1

Before we present the proof of the Theorem we present useful intermediate results which we require in our proof.

**Proposition 1.** *Consider a sequence of loss functions $\ell_a : \Theta \mapsto [0,1]_{a\in\mathcal{A}}$ drawn i.i.d. from some distribution $\mathcal{L}$ is given to an algorithm that generates a sequence of hypotheses $\{\boldsymbol{\theta}_a \in \Theta\}_{a\in\mathcal{A}}$ then the following inequality each hold w.p. $1-\delta$:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{\ell\sim D}\ell\left(\boldsymbol{\theta}^t\right) \leq \frac{1}{T}\sum_{t=1}^{T}\ell^t\left(\boldsymbol{\theta}^t\right) + \sqrt{\frac{2}{T}\log\frac{1}{\delta}}. \tag{15}$$

*Proof.* The proof of the Proposition can be directly derived from the Proposition 1 in Cesa-Bianchi et al. (2004). □

Then we could immediately obtain the below inequality by the symmetric version of the Azuma-Hoeffding inequality Azuma (1967)

**Remark 1.**

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\boldsymbol{\theta}^t\right) \geq \frac{1}{T}\sum_{t=1}^{T}\ell^t\left(\boldsymbol{\theta}^t\right) - \sqrt{\frac{2}{T}\log\frac{1}{\delta}}. \tag{16}$$

Finally, we give the definition of the regret of minimizing any subproblem:

**Definition 2.** ***(Subproblem Regret)*** *Consider an algorithm generates the trajectory of states $\{\boldsymbol{\theta}^t \in \Theta\}_{t\in[T]}$, the regret of such an algorithm on loss function $\{\ell^t\}_{t\in[T]}$ is:*

$$\bar{\mathbf{R}} = \sum_{t=1}^{T}\ell^t\left(\boldsymbol{\theta}^t\right) - \inf_{\boldsymbol{\theta}^\star\in\Theta}\sum_{t=1}^{T}\ell^t(\boldsymbol{\theta}). \tag{17}$$

**Theorem 2.** *(Restated) Consider an algorithm with regret bound $R_T$ that generates the trajectory of states for two base learners, for any parameter state $\boldsymbol{\theta} \in \Theta$, given a sequence of convex surrogate evaluation functions $\ell : \Theta \mapsto [0,1]_{a\in\mathcal{A}}$ drawn i.i.d. from some distribution $\mathcal{L}$, the expected error of the global learner $\boldsymbol{\theta}_g$ on both tasks over the test set can be bounded with probability at least $1-\delta$:*

$$\mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\boldsymbol{\theta}_g\right) \leq \mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\boldsymbol{\theta}\right) + \frac{\mathbf{R}_T}{T} + 2\sqrt{\frac{2}{T}\log\frac{1}{\delta}}. \tag{18}$$

*Proof.* From Theorem 1 and Remark 1, we obtain that

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\boldsymbol{\theta}^t\right) \leq \frac{1}{T}\sum_{t=1}^{T}\ell^t\left(\boldsymbol{\theta}\right) + \frac{\bar{\mathbf{R}}}{T} + \sqrt{\frac{2}{T}\log\frac{1}{\delta}} \leq \mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\boldsymbol{\theta}\right) + \frac{\bar{\mathbf{R}}}{T} + 2\sqrt{\frac{2}{T}\log\frac{1}{\delta}}. \tag{19}$$

It is obvious that:

$$\frac{\bar{\mathbf{R}}}{T} + \sqrt{\frac{2}{T}\log\frac{1}{\delta}} \leq \frac{\mathbf{R}_T}{T} + \sqrt{\frac{2}{T}\log\frac{1}{\delta}} \quad \text{and} \quad \frac{\bar{\mathbf{R}}}{T} + 2\sqrt{\frac{2}{T}\log\frac{1}{\delta}} \leq \frac{\mathbf{R}_T}{T} + 2\sqrt{\frac{2}{T}\log\frac{1}{\delta}}. \tag{20}$$

So we obtain:

$$\frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\boldsymbol{\theta}^t\right) \leq \frac{1}{T}\sum_{t=1}^{T}\ell^t\left(\boldsymbol{\theta}\right) + \frac{\mathbf{R}_T}{T} + \sqrt{\frac{2}{T}\log\frac{1}{\delta}} \leq \mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\boldsymbol{\theta}\right) + \frac{\mathbf{R}_T}{T} + 2\sqrt{\frac{2}{T}\log\frac{1}{\delta}}. \tag{21}$$

Recalling that in Section 3.3, $\boldsymbol{\theta}_g$ can be expressed by the linear combination of $\boldsymbol{\theta}_n$ and $\boldsymbol{\theta}_r$ through $t = 1, \cdots, T$ since $\boldsymbol{\theta}_g$ is aggregateed by EMA, so the above inequality can be further derived by the Jensen's inequality (convex surrogate functions could be selected to evaluate the test errors instead of the 0-1 loss):

$$\begin{aligned}\mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\boldsymbol{\theta}_g\right) = \mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\sum_{t=1}^{T}\boldsymbol{\theta}^t\right) \leq \frac{1}{T}\sum_{t=1}^{T}\mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\boldsymbol{\theta}^t\right) &\leq \frac{1}{T}\sum_{t=1}^{T}\ell^t\left(\boldsymbol{\theta}\right) + \frac{\mathbf{R}_T}{T} + \sqrt{\frac{2}{T}\log\frac{1}{\delta}} \\ &\leq \mathop{\mathbb{E}}_{\ell\sim\mathcal{L}}\ell\left(\boldsymbol{\theta}\right) + \frac{\mathbf{R}_T}{T} + 2\sqrt{\frac{2}{T}\log\frac{1}{\delta}}.\end{aligned} \tag{22}$$

Note that this inequality also holds when applying weight averaging technique to the base learner, because weight averaging is still the linear combination of all history states. □

# C VISUALIZATION

## C.1 PROPERTIES OF BASE LEARNERS AS TRAINING PROGRESSES

We show clean/adversarial accuracy of base learners as training progresses in Figure 5. From the curve of adversarial accuracy of two clean learners (the yellow and the red one), it can be inferred that adversarial examples generated from different base learners are not the same.

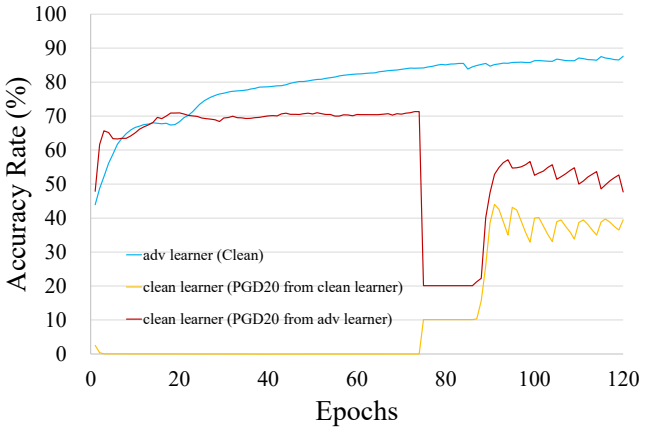



Figure 5: We plot clean accuracy of adversarial learner and robust accuracy of clean learner as training progresses. Adversarial examples for the evaluation of clean learner are generated from both clean learner and adversarial learner using PGD20.

We also apply t-SNE to visualize adversarial images of the clean learner and adversarial learner as training progresses in Figure 6. Note that the mixture happens every 5 epochs when the training epoch $\geq 75$. It is clear that adversarial images of two learners are disparate and easy to separate before epoch=90 even though they are periodically re-initialized using the same parameters of the global learner from epoch=75. Specifically, when compared with the base learners before being re-initialized (e.g. epoch=79, 84) and after being re-initialized (e.g. epoch=80, 85), the distribution of the adversarial images generated by different base learners is totally different. However, we should admit that the optimizer with a small learning rate is not able to greatly change the weight state after each re-initialization since the learning rate becomes smaller and smaller as training progresses, so the boundary of adversarial examples from different base learners is not clear at a later stage (e.g. epoch$\geq 94$). But the best tradeoff performance mainly comes from $90 \leq$epoch$\leq 100$ rather than at the very end of training and it is still obvious that a majority of adversarial images of two learners are not the same at that time, which reflects the validity of our theory from the side.

## C.2 FAILURE SAMPLES OF ZIPPER

We visualize typical failure cases of Zipper while TRADES or FAT correctly classifies them in Figure 7. For natural samples, we notice that most cases that Zipper fails in also make TRADES misclassify and the wrong predictions of TRADES seem irrelevant in semantics. Considering TRADES is not as good as Zipper on clean accuracy, it is natural to see TRADES cannot recognize samples not successfully predicted by Zipper. As for adversarial ones, we also observe that samples wrongly classified by Zipper will also be misclassified by FAT in most cases. Similarly, this is attributed to the poorer robustness of FAT than that of our proposed method. Because the robustness of the model trained by TRADES is better than Zipper with ResNet18, TRADES ($\lambda = 6$) will correctly classify more adversarial examples than our method. We also provide two cases that TRADES, FAT and our Zipper all misclassify. We believe that these cases contain some confusing information or texture (e.g. horse body hair in the third sample) that other categories may have, leading to the misclassification. From the above qualitative analysis, we believe that there is still some room for Zipper to improve recognition of foreground objects with confusing body shapes (e.g. cat/dog or horse/deer).

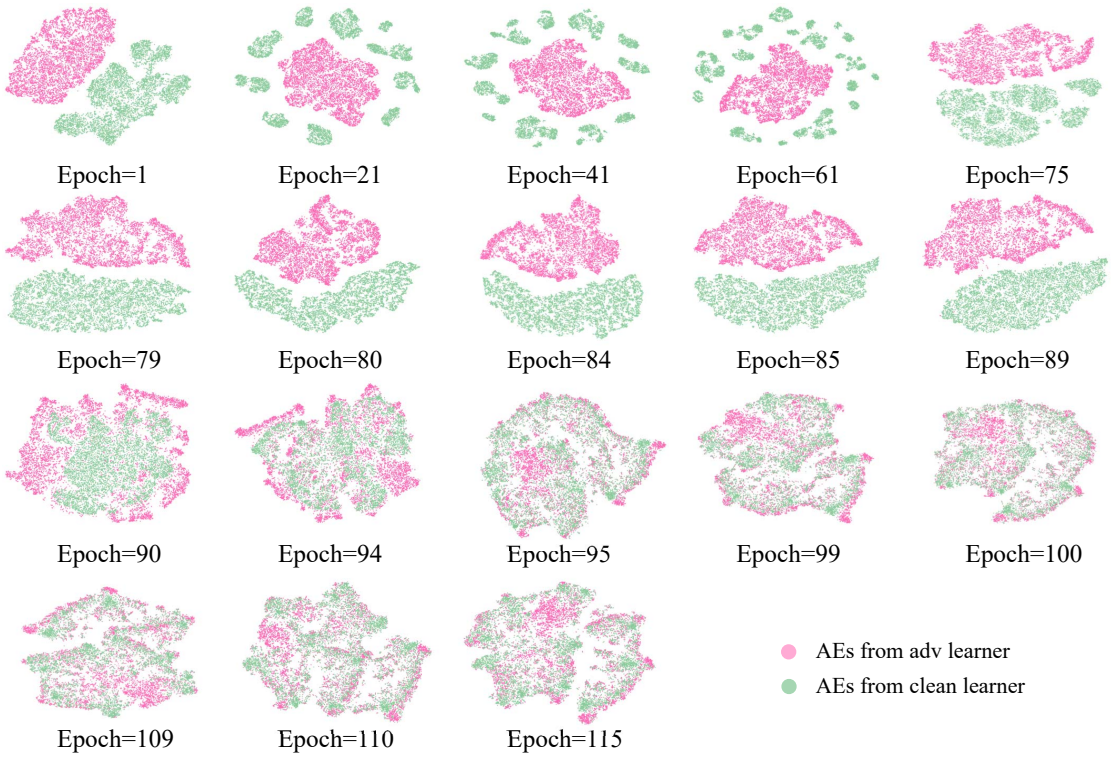



Figure 6: We choose natural images to generate corresponding adversarial examples of the clean learner and adversarial learner and visualize these samples by t-SNE as training progresses. Note that the mixture happens every 5 epochs when the training epoch $\geq 75$.


Figure 7: Typical failure cases of our method. We also provide the corresponding predictions of TRADES and FAT.