Improving Generalization of Adversarial Training via Robust Critical Fine-Tuning

Kaijie Zhu[1,2], **Jindong Wang**[3], **Xixu Hu**[4], **Xing Xie**[3], **Ge Yang**[1,2] *
[1]School of Artifical Intelligence, University of Chinese Academy of Sciences
[2]Institute of Automation, CAS [3] Microsoft Research [4] City University of Hong Kong

Abstract

*Deep neural networks are susceptible to adversarial examples, posing a significant security risk in critical applications. Adversarial Training (AT) is a well-established technique to enhance adversarial robustness, but it often comes at the cost of decreased generalization ability. This paper proposes Robustness Critical Fine-Tuning (**RiFT**), a novel approach to enhance generalization* without *compromising adversarial robustness. The core idea of RiFT is to exploit the redundant capacity for robustness by fine-tuning the adversarially trained model on its non-robust-critical module. To do so, we introduce* module robust criticality (MRC)*, a measure that evaluates the significance of a given module to model robustness under worst-case weight perturbations. Using this measure, we identify the module with the lowest MRC value as the non-robust-critical module and fine-tune its weights to obtain fine-tuned weights. Subsequently, we linearly interpolate between the adversarially trained weights and fine-tuned weights to derive the optimal fine-tuned model weights. We demonstrate the efficacy of RiFT on ResNet18, ResNet34, and WideResNet34-10 models trained on CIFAR10, CIFAR100, and Tiny-ImageNet datasets. Our experiments show that RiFT can significantly improve both generalization and out-of-distribution robustness by around* 1.5% *while maintaining or even slightly enhancing adversarial robustness. Code is available at* `https://github.com/microsoft/robustlearn`.

1. Introduction

The pursuit of accurate and trustworthy artificial intelligence systems is a fundamental objective in the deep learning community. Adversarial examples [43, 14], which perturbs input by a small, human imperceptible noise that can cause deep neural networks to make incorrect predictions, pose a significant threat to the security of AI systems. Notable experimental and theoretical progress has been made in defending against such adversarial examples [6, 4, 10, 18, 11, 15, 36]. Among various defense methods [49, 32, 54, 30, 8], adversarial training (AT) [28] has been shown to be one of the most promising approaches [4, 11] to enhance the adversarial robustness. However, compared to standard training, AT severely sacrifices generalization on in-distribution data [40, 44, 55, 35, 31] and is exceptionally vulnerable to certain out-of-distribution (OOD) examples [13, 50, 21] such as Contrast, Bright and Fog, resulting in unsatisfactory performance.

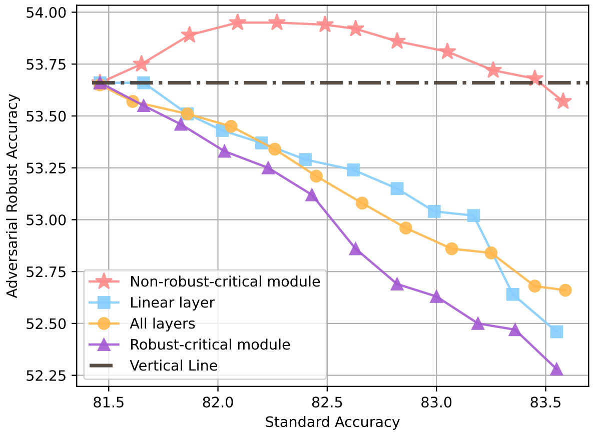

Figure 1. Interpolation results of fine-tuning on different modules of ResNet18 on CIFAR10 dataset. Dots denote different interpolation points between the final fine-tuned weights of RiFT and the initial adversarially trained weights. All fine-tuning methods improve the generalization ability, but only fine-tuning on the non-robust-critical module (`layer2.1.conv2`) can preserve robustness. Additionally, fine-tuning on robust-critical module (`layer4.1.conv1`) causes the worst trade-off between generalization and robustness. In the initial interpolation stage, fine-tuning on non-robust-critical modules enhances adversarial robustness by around 0.3%.

Prior arts tend to mitigate the trade-off between gener-

*Corresponding author: Ge Yang <ge.yang@ia.ac.cn>.

alization and adversarial robustness within the adversarial training procedure. For example, some approaches have explored reweighting instances [56], using unlabeled data [35], or redefining the robust loss function [55, 46, 48, 31]. In this paper, we take a different perspective to address such a trade-off by leveraging the *redundant capacity for robustness* of neural networks after adversarial training. Recent research has demonstrated that deep neural networks can exhibit *redundant capacity for generalization* due to their complex and opaque nature, where specific network modules can be deleted, permuted [45], or reset to their initial values [52, 9] with only minor degradation in generalization performance. Hence, it is intuitive to ask: *Do adversarially trained models have such redundant capacity? If so, how to leverage it to improve the generalization and OOD robustness* [1] *while maintaining adversarial robustness?*

Based on such motivation, we introduce a new concept called *Module Robust Criticality (MRC)* [2] to investigate the redundant capacity of adversarially trained models for robustness. MRC aims to quantify the *maximum increment of robustness loss* of a module's parameters under the *constrained weight perturbation*. As illustrated in Figure 3, we empirically find that certain modules do exhibit redundant characteristics under such perturbations, resulting in negligible drops in adversarial robustness. We refer to the modules with the lowest MRC value as the *non-robust-critical modules*. These findings further inspire us to propose a novel fine-tuning technique called **R**obust Cr**i**tical **F**ine-**T**uning (RiFT), which aims to leverage the redundant capacity of the non-robust-critical module to improve generalization while maintaining adversarial robustness. RiFT consists of three steps: (1) Module robust criticality characterization, which calculates the MRC value for each module and identifies the non-robust-critical module. (2) Non-robust-critical module fine-tuning, which exploits the redundant capacity of the non-robust-critical module via fine-tuning its weights with standard examples. (3) Mitigating robustness-generalization trade-off via interpolation, which interpolates between adversarially trained parameters and fine-tuned parameters to find the best weights that maximize the improvement in generalization while preserving adversarial robustness.

Experimental results demonstrate that RiFT significantly improves both the generalization performance and OOD robustness by around 2% while maintaining or even improving the adversarial robustness of the original models. Furthermore, we also incorporate RiFT to other adversarial training regimes such as TRADES [55], MART [46], AT-AWP [48], and SCORE [31], and show that such incorporation leads to further enhancements. More importantly, our experiments reveal several noteworthy insights. *First*, we found that fine-tuning on non-robust-critical modules can effectively mitigate the trade-off between adversarial robustness and generalization, showing that these two can both be improved (Section 5.3). As illustrated in Figure 1, adversarial robustness increases alongside the generalization in the initial interpolation procedure, indicating that the features learned by fine-tuning can benefit both generalization and adversarial robustness. This contradicts the previous claim [44] that the features learned by optimal standard and robust classifiers are fundamentally different. *Second*, the existence of non-robust-critical modules suggests that current adversarial training regimes do not fully utilize the capacity of DNNs (Section 5.2). This motivates future work to design more efficient adversarial training approaches using such capacity. *Third*, while previous study [24] reported that fine-tuning on *pre-train models* could distort the learned robust features and result in poor performance on OOD samples, we find that fine-tuning *adversarially trained models* do *NOT* lead to worse OOD performance (Section 5.3).

The contribution of this work is summarized as follows:

1. **Novel approach.** We propose the concept of module robust criticality and verify the existence of redundant capacity for robustness in adversarially trained models. We then propose RiFT to exploit such redundancy to improve the generalization of AT models.

2. **Superior experimental results.** Our approach improves both generalization and OOD robustness of AT models by around 2%. It can also be incorporated with previous AT methods to mitigate the trade-off between generalization and adversarial robustness.

3. **Interesting insights.** The findings of our experiments shed light on the intricate interplay between generalization, adversarial robustness, and OOD robustness. Our work emphasizes the potential of leveraging the redundant capacity in adversarially trained models to improve generalization and robustness further, which may inspire more efficient and effective training methods to fully utilize this redundancy.

2. Related Work

Trade-off between adversarial robustness and generalization The existence of such trade-off has been extensively debated in the adversarial learning community [40, 44, 55, 20, 35, 31]. Despite lingering controversies, the prevalent viewpoint is that this trade-off is inherent. Theoretical analyses [44, 35, 20] demonstrated that the trade-off provably exists even in simple cases, *e.g.*, binary classification and linear regression. To address this trade-off, various methods have been proposed during adversarial train-

[1] Here, generalization refers to generalization to in-distribution (ID) samples, and OOD robustness refers to generalization to OOD samples.

[2] In our paper, a module refers to a layer of the neural network.

ing, such as instance reweighting [56], robust self-training [35], incorporating unlabeled data [7, 18], and redefining the robust loss function [55, 46, 48, 31]. This paper presents a novel post-processing approach that exploits the excess capacity of the model after adversarial training to address such trade-off. Our RiFT can be used in conjunction with existing adversarial training techniques, providing a practical and effective way to mitigate the trade-off further.

Redundant Fitting Capacity The over-parameterized deep neural networks (DNNs) exhibit striking fitting power even for random labels [52, 3]. Recent studies have shown that not all modules contribute equally to the generalization ability of DNNs [45, 38, 53, 9], indicating the redundant fitting capacity for generalization. Veit *et al.* [45] found that some blocks can be deleted or permuted without degrading the test performance too much. Rosenfeld and Tsotsos [38] demonstrated that one could achieve comparable performance by training only a small fraction of network parameters. Further, recent studies have identified certain neural network modules, referred to as *robust modules* [53, 9], rewinding their parameters to initial values results in a negligible decline in generalization. Previous studies have proposed methods to reduce the computational and storage costs of deep neural networks by *removing* the redundant capacity for generalization while preserving comparable performance, such as compression [16] and distillation [19]. In contrast, our work focuses on the *redundant capacity for robustness* of adversarially trained models and tries to *exlpoit* such redundancy.

Fine-tuning Methods Pre-training on large scale datasets has been shown to be a powerful approach for developing high-performing deep learning models [5, 12, 34, 22]. Fine-tuning is a widely adopted approach to enhance the transferability of pre-trained models to downstream tasks and domain shifts. Typically, fine-tuning methods involve fine-tuning the last layer (linear probing) [1, 24] or all layers (fully fine-tuning) [1, 18, 29, 24]. Salman *et al.* [39] demonstrated that both fully fine-tuning and linear probing of adversarially trained models can improve the transfer performance on downstream tasks. Nevertheless, recent studies [2, 47, 24] have suggested that fine-tuning can degrade pre-trained features and underperformance on out-of-distribution (OOD) samples. To address this issue, different fine-tuning techniques are proposed such as WiSE-FT [47] and surgical fine-tuning [27] that either leveraged ensemble learning or selective fine-tuning for better OOD performance. Kumar *et al.* [24] suggested the two-step strategy of linear probing then full fine-tuning (LP-FT) combines the benefits of both fully fine-tuning and linear probing.

3. Module Robust Criticality

Improving the generalization of adversarially trained models requires a thorough understanding of DNNs, which, however, proves to be difficult due to the lack of explainability. Luckily, recent studies show that specific modules in neural networks, referred to as *critical modules* [53, 9], significantly impact model generalization if their parameters are rewound to initial values. In this work, we propose a metric called **Module Robust Criticality (MRC)** to evaluate the robustness contribution of each module explicitly.

3.1. Preliminaries

We denote a l-layered DNN as $f(\boldsymbol{\theta}) = \phi(\boldsymbol{x}^{(l)}; \boldsymbol{\theta}^{(l)}) \circ \ldots \circ \phi(\boldsymbol{x}^{(1)}; \boldsymbol{\theta}^{(1)})$, where $\boldsymbol{\theta}^{(i)}$ is the parameter of i-th layer and $\phi(\cdot)$ denotes the activation function. We use $\boldsymbol{\theta}_{AT}$ and $\boldsymbol{\theta}_{FT}$ to denote the weights of the adversarially trained and fine-tuned model, respectively. We use $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)\}$ to denote a dataset and $\mathcal{D}_{std}$ means a standard dataset such as CIFAR10. The cross-entropy loss is denoted by $\mathcal{L}$ and $\|\cdot\|_p$ is denoted as the ℓ_p norm.

Let $\Delta\boldsymbol{x} \in \mathcal{S}$ denote the adversarial perturbation applied to a clean input $\boldsymbol{x}$, where $\mathcal{S}$ represents the allowed range of input perturbations. Given a neural network $f(\boldsymbol{\theta})$ and a dataset $\mathcal{D}$, adversarial training aims to minimize the robust loss [28] as:

$$\begin{gathered}\arg\min_{\boldsymbol{\theta}} \mathcal{R}(f(\boldsymbol{\theta}), \mathcal{D}), \text{ where} \\ \mathcal{R}(f(\boldsymbol{\theta}), \mathcal{D}) = \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \max_{\Delta\boldsymbol{x}\in\mathcal{S}} \mathcal{L}(f(\boldsymbol{\theta}, \boldsymbol{x} + \Delta\boldsymbol{x}), y).\end{gathered} \tag{1}$$

Here, $\mathcal{R}(f(\boldsymbol{\theta}), \mathcal{D})$ is the robust loss to find the *worst-case input perturbation* that maximizes the cross-entropy classification error.

3.2. Module Robust Criticality

Definition 3.1 (Module Robust Criticality)**.** Given a weight perturbation scaling factor $\epsilon > 0$ and a neural network $f(\boldsymbol{\theta})$, the robust criticality of a module i is defined as

$$\begin{aligned}MRC(f, \boldsymbol{\theta}^{(i)}, \mathcal{D}, \epsilon) = \max_{\Delta\boldsymbol{\theta}\in\mathcal{C}_{\boldsymbol{\theta}}} & \mathcal{R}(f(\boldsymbol{\theta} + \Delta\boldsymbol{\theta}), \mathcal{D}) \\ & - \mathcal{R}(f(\boldsymbol{\theta}), \mathcal{D}),\end{aligned} \tag{2}$$

where $\Delta\boldsymbol{\theta} = \{\boldsymbol{0}, \ldots, \boldsymbol{0}, \Delta\boldsymbol{\theta}^{(i)}, \boldsymbol{0}, \ldots, \boldsymbol{0}\}$ denotes the weight perturbation with respect to the module weights $\boldsymbol{\theta}^{(i)}$, $\mathcal{C}_{\boldsymbol{\theta}} = \{\Delta\boldsymbol{\theta} \mid \|\Delta\boldsymbol{\theta}\|_p \leq \epsilon\|\boldsymbol{\theta}^{(i)}\|_p\}$, $\mathcal{R}(\cdot)$ is the robust loss defined in Eq. (1).

The MRC value for each module represents how they are critically contributing to model adversarial robustness. The module with the lowest MRC value is considered redundant, as changing its weights has a negligible effect on robustness degradation. We refer to this module as the non-robust-critical module. Intuitively, MRC serves as an upper bound for weight changing of a particular module, as demonstrated in Theorem 3.1. Since we do not know the optimization directions and how they might affect the model robustness to adversarial examples, we measure the extent to which *worst-case* weight perturbations affect the robustness, providing an upper bound loss for optimizing the weight. Further, the MRC for a module depicts the sharpness of robust loss landscape [48, 41] around the minima $\boldsymbol{\theta}^{(i)}$. If the MRC score is high, it means that the robust loss landscape with respect to $\boldsymbol{\theta}^{(i)}$ is sharp, and fine-tuning this module is likely to hurt the adversarial robustness.

Theorem 3.1. The MRC value for a module i serves as an upper bound for the robust loss increase when we optimize the module under constraint $\mathcal{C}_{\boldsymbol{\theta}}$:

$$\mathcal{R}(f(\boldsymbol{\theta}^*), \mathcal{D}) - \mathcal{R}(f(\boldsymbol{\theta}), \mathcal{D}) \leq MRC(f, \boldsymbol{\theta}^{(i)}, \mathcal{D}, \epsilon),$$
$$\text{where } \boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}', (\boldsymbol{\theta}'-\boldsymbol{\theta}) \in \mathcal{C}_{\boldsymbol{\theta}}}{\arg\min} \sum_{(\boldsymbol{x}, y) \in \mathcal{D}} \mathcal{L}(f(\boldsymbol{\theta}', x), y). \tag{3}$$

Proof. By the definition of MRC, for any weights $(\boldsymbol{\theta}' - \boldsymbol{\theta}) \in \mathcal{C}_{\boldsymbol{\theta}}$, we have:

$$\mathcal{R}(f(\boldsymbol{\theta}'), \mathcal{D}) - \mathcal{R}(f(\boldsymbol{\theta}), \mathcal{D}) \leq MRC(f, \boldsymbol{\theta}^{(i)}, \mathcal{D}, \epsilon). \tag{4}$$

Thus, for the optimized weights:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}', (\boldsymbol{\theta}'-\boldsymbol{\theta}) \in \mathcal{C}_{\boldsymbol{\theta}}}{\arg\min} \sum_{(\boldsymbol{x}, y) \in \mathcal{D}} \mathcal{L}(f(\boldsymbol{\theta}', x), y), \tag{5}$$

it satisfies

$$\mathcal{R}(f(\boldsymbol{\theta}^*), \mathcal{D}) - \mathcal{R}(f(\boldsymbol{\theta}), \mathcal{D}) \leq MRC(f, \boldsymbol{\theta}^{(i)}, \mathcal{D}, \epsilon). \tag{6}$$

Such that the proof ends. □

Remark: The definition of MRC is similar in spirit to the work of Zhang *et al.* [53] and Chatterji *et al.* [9]. However, MRC differs fundamentally from them in two aspects. First, MRC aims to capture the influence of a module on *adversarial robustness*, while Zhang *et al.* [53] and Chatterji *et al.* [9] focus on studying the impact of a module on *generalization*. Second, MRC investigates the robustness characteristics of module weights under *worst-case weight perturbations*, whereas Zhang *et al.* [53] and Chatterji *et al.* [9] analyzed the properties of a module by *rewinding its weights to their initial values*. Similar to [25, 41], we define the weight perturbation constraint $C_{\boldsymbol{\theta}}$ as a multiple of the ℓ_p norm of original parameters, which ensures the scale-invariant property and allows us to compare the robust criticality of modules across different layers, see Appendix A for a detailed proof.

Theorem 3.1 establishes a clear upper bound for fine-tuning particular modules. This theorem assures us that fine-tuning on non-robust-critical modules shouldn't harm the model robustness. However, it does not ascertain if fine-tuning the robust-critical module will lead to a significant decline in robust accuracy.

Algorithm 1 Module Robust Criticality Characterization

```
Input: neural network $f$, adversarially trained model weights $\boldsymbol{\theta}_{AT}$, desired module $i$'s weights $\boldsymbol{\theta}^{(i)}$, standard dataset $\mathcal{D}_{std}$, weight perturbation scaling factor $\epsilon$, optimization iteration steps $T$, learning rate $\gamma$.
Output: The module robust criticality of module $i$.
Initialize adversarial dataset: $\mathcal{D}_{adv} = \{\}$
for Batch $\mathcal{B}_k \in \mathcal{D}_{std}$ do    ▷ Generate adversarial dataset
    $\mathcal{B}_k^{adv}$ = PGD-10($\boldsymbol{\theta}_{AT}, \mathcal{B}_k$)
    $\mathcal{D}_{adv} = \mathcal{D}_{adv} \bigcup \mathcal{B}_k^{adv}$
end for
Freeze all parameters of $\boldsymbol{\theta}_{AT}$ except for $\boldsymbol{\theta}^{(i)}$
$\boldsymbol{\theta}_1 = \boldsymbol{\theta}_{AT}$
for $t = 1, \ldots, T$ do    ▷ Iterate $T$ epochs
    $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t$
    for Batch $\mathcal{B}_k^{adv} \in \mathcal{D}_{adv}$ do
        Calculate Loss: $\mathcal{L}(f, \boldsymbol{\theta}_t, \mathcal{B}_k^{adv}))$
        $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_{t+1} + \gamma \nabla_{\boldsymbol{\theta}_t}(\mathcal{L})$    ▷ Gradient Ascent
    end for
    $\Delta\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}_{t+1}^{(i)} - \boldsymbol{\theta}_{AT}^{(i)}$    ▷ Check perturb constraint
    if $\|\Delta\boldsymbol{\theta}^{(i)}\|_2 \geq \epsilon \|\boldsymbol{\theta}_{AT}^{(i)}\|_2$ then
        $\Delta\boldsymbol{\theta}^{(i)} = \epsilon \frac{\|\boldsymbol{\theta}_{AT}^{(i)}\|_2}{\|\Delta\boldsymbol{\theta}^{(i)}\|_2} \Delta\boldsymbol{\theta}^{(i)}$
        $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \Delta\boldsymbol{\theta}^{(i)}$
        break
    end if
end for
$MRC(\boldsymbol{\theta}^{(i)}) = \mathcal{L}(f, \boldsymbol{\theta}_T, \mathcal{D}_{adv}) - \mathcal{L}(f, \boldsymbol{\theta}_{AT}, \mathcal{D}_{adv})$
Return $MRC(\boldsymbol{\theta}^{(i)})$
```

3.3. Relaxation of MRC

Optimizing in Eq. (2) requires simultaneously finding worst-case weight perturbation $\Delta\boldsymbol{\theta}$ and worst-case input perturbation $\Delta\boldsymbol{x}$, which is time-consuming. Thus, we propose a relaxation version by fixing $\Delta\boldsymbol{x}$ at the initial optimizing phase. Concretely, we first calculate the adversarial examples $\Delta\boldsymbol{x}$ with respect to $\boldsymbol{\theta}_{AT}$. By fixing the adversarial examples unchanged during the optimization, we iteratively optimize the $\Delta\boldsymbol{\theta}$ by gradient ascent method to maximize the robust loss to find the optimal $\Delta\boldsymbol{\theta}$. We set a weight perturbation constraint and check it after each optimization step. If the constraint is violated, we project the perturbation onto the constraint set. The pseudo-code is described in Algorithm 1. In our experiments, if not specified, we set $\|\cdot\|_p = \|\cdot\|_2$ and $\epsilon = 0.1$ for $C_{\boldsymbol{\theta}}$, the iterative step for optimizing $\Delta\boldsymbol{\theta}$ is 10.

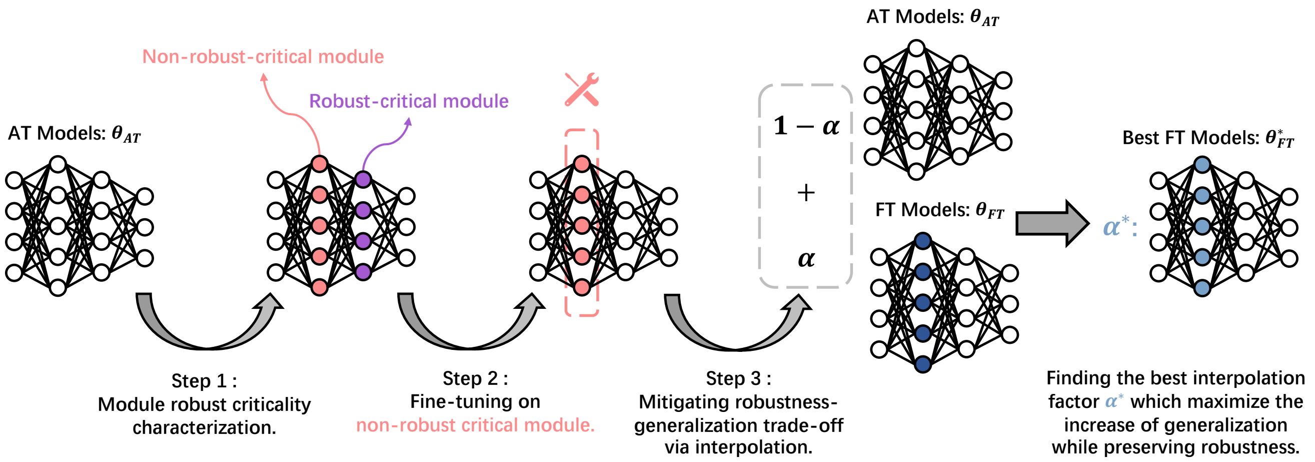
AT Models: $\boldsymbol{\theta}_{AT}$
Non-robust-critical module
Robust-critical module
AT Models: $\boldsymbol{\theta}_{AT}$
$1-\alpha$
$+$
α
FT Models: $\boldsymbol{\theta}_{FT}$
Best FT Models: $\boldsymbol{\theta}^{*}_{FT}$
α^{*}:
Step 1 :
Module robust criticality characterization.
Step 2 :
Fine-tuning on
non-robust critical module.
Step 3 :
Mitigating robustness-generalization trade-off via interpolation.
Finding the best interpolation factor α^{*} which maximize the increase of generalization while preserving robustness.

Figure 2. The pipeline of our proposed Robust Critical Fine-Tuning (RiFT).

4. RiFT: Robust Critical Fine-tuning

In this paper, we propose **RiFT**, a robust critical fine-tuning approach that leverages MRC to guide the fine-tuning of a deep neural network to improve both generalization and robustness. Let $\mathcal{P}_{adv}(x, y)$ and $\mathcal{P}_{std}(x, y)$ denote the distributions of adversarial and standard inputs, respectively. Then, applying an adversarially trained model on $\mathcal{P}_{adv}(x, y)$ to $\mathcal{P}_{std}(x, y)$ can be viewed as a *distributional shift* problem. Thus, it is natural for RiFT to exploit the redundant capacity to fine-tune adversarially trained models on the standard dataset.

Specifically, RiFT consists of three steps as shown in Figure 2. First, we calculate the MRC of each module and choose the module with the lowest MRC score as our non-robust-critical module. Second, we freeze the parameters of the adversarially trained model except for our chosen non-robust-critical module. Then we fine-tune the adversarially trained models on corresponding standard dataset $\mathcal{D}_{std}$. Third, we linearly interpolate the weights of the original adversarially trained model and fine-tuned model to identify the optimal interpolation point that maximizes generalization improvement while maintaining robustness.

Step 1: Module robust criticality characterization According to the Algorithm 1, we iteratively calculate the MRC value for each module $\boldsymbol{\theta}^{(i)} \in \boldsymbol{\theta}_{AT}$, then we choose the module with the lowest MRC value, denoted as $\tilde{\boldsymbol{\theta}}$:

$$\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(i)} \text{ where } i = \arg\min_{i} MRC(f, \boldsymbol{\theta}^{(i)}, \mathcal{D}, \epsilon). \quad (7)$$

Step 2: Fine-tuning on non-robust-critical modules Next, we freeze the rest of the parameters and fine-tune on desired parameters $\tilde{\boldsymbol{\theta}}$. We solve the following optimization problem by SGD with momentum [42]

$$\arg\min_{\tilde{\boldsymbol{\theta}}} \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \mathcal{L}(f(x, (\tilde{\boldsymbol{\theta}}; \boldsymbol{\theta} \setminus \tilde{\boldsymbol{\theta}})), y) + \lambda\|\tilde{\boldsymbol{\theta}}\|_2, \quad (8)$$

where λ is the ℓ_2 weight decay factor.

Step 3: Mitigating robustness-generalization trade-off via interpolation For a interpolation coefficient α, the interpolated weights is calculated as:

$$\boldsymbol{\theta}_{\alpha} = (1-\alpha)\boldsymbol{\theta}_{AT} + \alpha\boldsymbol{\theta}_{FT}, \quad (9)$$

where $\boldsymbol{\theta}_{AT}$ is the initial adversarially trained weights and $\boldsymbol{\theta}_{FT}$ is the fine-tuned weights obtained by Eq. (8). Since our goal is to improve the generalization while preserving adversarial robustness, thus the best interpolation point is chosen to be the point that most significantly improves the generalization while the corresponding adversarial robustness is no less than the original robustness by 0.1.

Remark: Theorem 3.1 establishes an upper bound on the possible drop in robustness loss that can be achieved through fine-tuning. It is expected that the second step of optimization would enforce the parameters to lie within the boundary $\mathcal{C}_{\boldsymbol{\theta}}$ in order to satisfy the theorem. However, here we do not employ constrained optimization but find the optimal point by first optimizing without constraints and then interpolating. This is because (1) the constraints are empirically given and may not always provide the optimal range for preserving robustness, and it is possible to fine-tune outside the constraint range and still ensure that there is not much loss of robustness. (2) the interpolation procedure serves as a weight-ensemble, which may benefit both robustness and generalization, as noted in WiSE-FT [47]. The complete algorithm of RiFT is shown in Appendix B.

5. Experiments

5.1. Experimental Setup

Datasets We adopt three popular image classification datasets: CIFAR10 [23], CIFAR100 [23], and Tiny-ImageNet [26]. CIFAR10 and CIFAR100 comprise 60,000 32×32 color images in 10 and 100 classes, respectively. Tiny-ImageNet is a subset of ImageNet and contains 200 classes, where each class contains 500 colorful images with size 64×64. We use three OOD datasets accordingly to evaluate the OOD robustness: CIFAR10-C, CIFAR100-C, and Tiny-ImageNet-C [18]. These datasets simulate 15 types of common visual corruptions and are grouped into four classes: Noise, Blur, Weather, and Digital.

Evaluation metrics We use the test set accuracy of each standard dataset to represent the generalization ability. For evaluating adversarial robustness, we adopt a common setting of PGD-10 [28] with constraint $\ell_\infty = 8/255$. We run PGD-10 with three times and select the *worst* robust accuracy as the final metric. The OOD robustness is evaluated by the accuracy of the test set of the corrupted dataset corresponding to the standard dataset.

Training details We use ResNet18 [17], ResNet34 [17], WideResNet34-10 (WRN34-10) [51] as backbones. ResNet18 and ResNet34 are 18-layer and 34-layer ResNet models, respectively. WideResNet34-10 is a 34-layer WideResNet model with a widening factor of 10. Similarly, we adopt PGD-10 [28] with constraint $\ell_\infty = 8/255$ for adversarial training. Following standard settings [37, 33], we train models with adversarial examples for 110 epochs. The learning rate starts from 0.1 and decays by a factor of 0.1 at epochs 100 and 105. We select the weights with the highest test robust accuracy as our adversarially trained models.

We fine-tune the adversarially trained models $\boldsymbol{\theta}_{AT}$ using SGD with momentum [42] for 10 epochs. The initial learning rate is set to 0.001.[3] We decay the learning rate by $1/10$ after fine-tuning for 5 epochs We choose the weights with the highest test accuracy as fine-tuned model weights, denoted as $\boldsymbol{\theta}_{FT}$. We then interpolate between initial adversarially trained model weights $\boldsymbol{\theta}_{AT}$ and $\boldsymbol{\theta}_{FT}$, the best interpolation point selected by Step 3 in Section 4 is denoted as $\boldsymbol{\theta}^*_{FT}$. We then compare the generalization, adversarial robustness, and OOD robustness of $\boldsymbol{\theta}^*_{FT}$ and $\boldsymbol{\theta}_{AT}$.

We report the average of three different seeds and omit the standard deviations of 3 runs as they are tiny ($< 0.20\%$), which hardly effect the results. Refer to Appendix C for more training details.

5.2. Empirical Analysis of MRC

Before delving into the main results of RiFT, we first empirically analyze our proposed MRC metric in Definition 3.1, which serves as the foundation of our RiFT approach. We present the MRC analysis on ResNet18 [17] on CIFAR-10 in Figure 3, where each column corresponds to the MRC value and its corresponding robust accuracy drop of a specific module.

Our analysis shows that the impact of worst-case weight perturbations on model robustness varies across different modules. Some modules exhibit minimal impact on robustness under perturbation, indicating the presence of *redundant capacity for robustness*. Conversely, for other modules, the worst-case weight perturbation shows a significant impact, resulting in a substantial decline in robustness. For example, in module `layer2.1.conv2`, worst-case weight perturbations only result in a meager addition of 0.09 robust loss. However, for `layer4.1.conv1`, the worst-case weight perturbations affect the model's robust loss by an additional 12.94, resulting in a substantial decline (53.03%) in robustness accuracy. Such robust-critical and non-robust-critical modules are verified to exist in various network architectures and datasets, as detailed in Appendix C.4. We also observe that as the network capacity decreases (*e.g.*, from WRN34-10 to ResNet18) and the task becomes more challenging (*e.g.*, from CIFAR10 to Tiny-ImageNet), the proportion of non-robust-critical modules increases, as less complex tasks require less capacity, leading to more non-robust-critical modules.

It is worthy noting that the decrease in robust accuracy doesn't directly correlate with MRC. For instance, both `layer4.0.conv2` and `layer4.1.conv1` have a robust accuracy drop of 53.05%, yet their MRC values differ. This discrepancy can be attributed to the different probability distributions of misclassified samples across modules, resulting in same accuracy declines but different losses.

5.3. Main Results

Table 1 summarizes the main results of our study, from which we have the following findings.

RiFT improves generalization First, RiFT effectively mitigates the trade-off between generalization and robustness raised by adversarial training. Across different datasets and network architectures, RiFT improves the generalization of adversarially trained models by approximately 2%. This result prompts us to rethink the trade-off, as it may be caused by inefficient adversarial training algorithm rather than the inherent limitation of DNNs. Furthermore, as demonstrated in Figure 1, both adversarial robustness and generalization increase simultaneously in the initial interpolation process, indicating that these two characteristics can

[3] The best learning rate for fine-tuning vary across architectures and datasets and is required to be carefully modified.

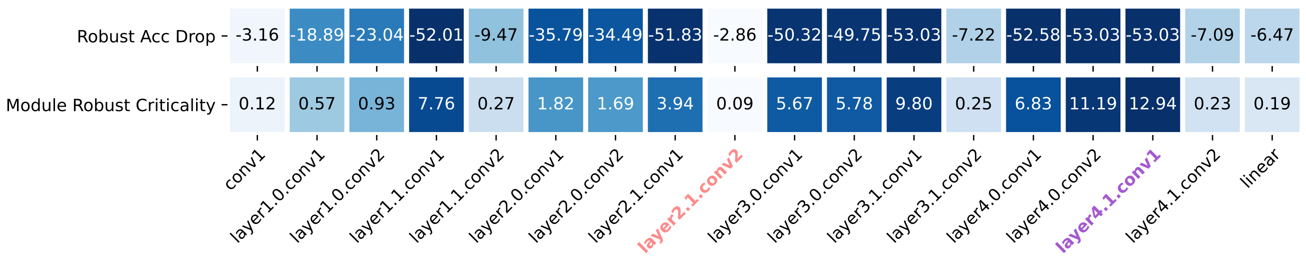

Figure 3. Example of module robust criticality (MRC) and its corresponding robust accuracy drop of ResNet18 trained on CIFAR10. Each column represents an individual module. The first row represents the corresponding robust accuracy drop and the second row represents the MRC value of each module. The higher the MRC value is, the more robust-critical the module is. Some modules are not critical to robustness, exhibiting redundant characteristics for contributing to robustness. However, some modules are critical to robustness. For example, the robust acc drop is only 2.86% for `layer2.1.conv2` while for `layer4.1.conv1` the robust acc drop is up to 53.03%.

Table 1. Results of RiFT on different datasets and backbones. *Std* means the standard test accuracy for in distribution generalization, *OOD* denotes the OOD robust accuracy of corresponding corruption dataset (*e.g.*, CIFAR10-C). *Adv* denotes the adversarial robust accuracy. In each column, we bold the entry with the higher accuracy. RiFT improves both generalization and OOD robustness across architectures and datasets while maintaining adversarial robustness.

Architecture	Method	CIFAR10			CIFAR100			Tiny-ImageNet		
		Std	*OOD*	*Adv*	*Std*	*OOD*	*Adv*	*Std*	*OOD*	*Adv*
ResNet18	AT	81.46	73.56	53.63	57.10	46.43	30.15	49.10	27.68	23.28
	AT+RiFT	**83.44**	**75.69**	**53.65**	**58.74**	**48.06**	**30.17**	**50.61**	**28.73**	**23.34**
	Δ	+1.98	+2.13	+0.02	+1.64	+1.63	+0.02	+1.51	+1.05	+0.06
ResNet34	AT	84.23	75.37	55.31	58.67	48.24	30.50	50.96	27.91	24.27
	AT+RiFT	**85.41**	**77.15**	**55.34**	**60.88**	**49.97**	**30.58**	**52.54**	**30.07**	**24.37**
	Δ	+1.18	+1.78	+0.03	+2.21	+1.73	+0.08	+1.58	+2.16	+0.10
WRN34-10	AT	87.41	78.75	55.40	62.35	50.61	**31.66**	52.78	31.81	26.07
	AT+RiFT	**87.89**	**79.31**	**55.41**	**64.56**	**52.69**	31.64	**55.31**	**33.86**	**26.17**
	Δ	+0.48	+0.56	+0.01	+2.21	+2.08	-0.02	+2.53	+2.05	+0.10
Avg	Δ	+1.21	+1.49	+0.02	+2.02	+1.81	+0.02	+1.87	+1.75	+0.08

be improved together. This trend is observed across different datasets and network architectures; see Appendix C.5 for more illustrations. This finding challenges the notion that the features of optimal standard and optimal robust classifiers are fundamentally different, as previously claimed by Tsipras *et al.* [44], as fine-tuning procedures can increase both robustness and generalization.

Fine-tuning improves OOD robustness Second, our study also investigated the out-of-distribution (OOD) robustness of the fine-tuned models and observed an improvement of approximately 2%. This observation is noteworthy because recent work [2, 24, 47] showed that fine-tuning pretrained models can distort learned features and result in underperformance in OOD samples. Furthermore, Yi *et al.* [50] demonstrated that adversarial training enhances OOD robustness, but it is unclear whether fine-tuning on adversarially trained models distorts robust features. Our results indicate that fine-tuning adversarially trained models does not distort the robust features learned by adversarial training and instead helps improve OOD robustness. We suggest fine-tuning adversarially trained models may be a promising avenue for further improving OOD robustness.

5.4. Incorporate RiFT to Other AT Methods

To further validate the effectiveness of RiFT, we conduct experiments on ResNet18 [17] trained on CIFAR10 and CIFAR100 [23] using four different adversarial training techniques: TRADES [55], MART [46], AWP [48], and SCORE [31], and then apply our RiFT to the resulting models. As shown in Table 2, our approach is compatible with various adversarial training methods and improves generalization and OOD robustness.

5.5. Ablation Study

Fine-tuning on different modules To evaluate the efficacy of fine-tuning the non-robust-critical module, we conducted further experiments by fine-tuning the adversarially trained model on different modules. Specifically, we used four fine-tuning methods: fully fine-tuning, linear probing

Table 2. Results of RiFT + other AT methods.

Method	CIFAR10			CIFAR100		
	Std	*OOD*	*Adv*	*Std*	*OOD*	*Adv*
TRADES	81.54	73.42	**53.31**	57.44	47.23	30.20
TRADES+RiFT	**81.87**	**74.09**	53.30	**57.78**	**47.52**	**30.22**
Δ	+0.33	+0.67	-0.01	+0.34	+0.29	+0.02
MART	76.77	68.62	56.90	51.46	42.07	31.47
MART+RiFT	**77.14**	**69.41**	**56.92**	**52.42**	**43.35**	**31.48**
Δ	+0.37	+0.79	+0.02	+0.96	+1.28	+0.01
AWP	78.40	70.48	53.83	52.85	43.10	31.00
AWP+RiFT	**78.79**	**71.12**	**53.84**	**54.89**	**45.08**	**31.05**
Δ	+ 0.39	+0.64	+0.01	+2.04	+1.98	+0.05
SCORE	84.20	75.82	54.59	54.83	45.39	29.49
SCORE+RiFT	**85.65**	**77.37**	**54.62**	**57.63**	**47.77**	**29.50**
Δ	+1.45	+1.55	+0.03	+2.80	+2.38	+0.01

Table 3. Results of fine-tuning on different modules.

Method	*Std*	*OOD*	*Adv*
All layers	83.56	75.48	52.66
Last layer	83.35	75.16	52.75
Robust-critical	83.36	75.42	52.48
Non-robust-critical	83.44	75.69	**53.65**

Table 4. Results of fine-tuning on multiple non-robust-critical modules.

Method	*Std*	*OOD*	*Adv*
Top 1	83.44	75.69	**53.65**
Top 2	83.41	75.61	52.47
Top 3	83.59	75.77	52.22
Top 5	83.70	75.82	52.35

(fine-tuning on the last layer), fine-tuning on the non-robust-critical module, and fine-tuning on the robust-critical module. The experiment was conducted using ResNet18 on CIFAR-10, and the results are presented in Figure 1 and Table 3. As described in Section 3.2, MRC is an upper bound for weight perturbation, indicating the criticality of a module in terms of model robustness. Fine-tuning on a non-robust-critical module can help preserve adversarial robustness but does not guarantee improvement in generalization. Similarly, fine-tuning on the robust-critical module does not necessarily hurt robustness. However, our experiments observed that all fine-tuning methods improved generalization ability, but only fine-tuning on non-robust-critical module preserved adversarial robustness. Moreover, fine-tuning on the robust-critical module exhibited the worst trade-off between generalization and robustness compared to fine-tuning on all layers.

More non-robust-critical modules, more useful? To investigate whether fine-tuning on more non-critical modules could further improve generalization, we additionally fine-tune on the top two, top three, and top five non-robust-critical modules. However, Table 3 reveals that generalization and OOD robustness did not surpass the results achieved by fine-tuning a singular non-robust-critical module. Notably, performance deteriorated when fine-tuning multiple non-critical modules compared to fine-tuning all layers. It's pivotal to note that this doesn't negate MRC's applicability to several modules. The MRC for module i is evaluated with other module parameters held constant, making it challenging to discern the impact of worst-case perturbations across multiple modules using the MRC of a single one. We posit that broadening MRC's definition to encompass multiple modules might address this problem.

Ablation on interpolation factor α^* The value of α^* is closely related to the fine-tuning learning rate. Specifically, a large learning rate can result in substantial weight updates that may push the fine-tuned weights $\boldsymbol{\theta}_{FT}$ away from their adversarially trained counterparts $\boldsymbol{\theta}_{AT}$. Our empirical results indicate that a fine-tuning learning rate of 0.001 is suitable for most cases and that the corresponding α^* value generally ranges between 0.6 to 0.9.

Factors related to the generalization gain of RiFT "Our results unveiled patterns and behaviors that offer insights into the determinants of the generalization gains observed with RiFT. First, the generalization gain of RiFT is a function of both the neural network's inherent capacity and the inherent difficulty posed by the classification task. Specifically, as the classification task becomes more challenging, the robust criticality of each module increases, which in turn decreases the generalization gain of RiFT. This effect can be mitigated by using a model with a larger capacity. For instance, we observe that the generalization gain of RiFT increases as we switch from ResNet18 to ResNet34 and to WRN34-10 when evaluating on CIFAR100 and Tiny-ImageNet. Further, We observed that the generalization gain of RiFT with WRN34-10 on CIFAR10 is notably lower, at approximately 0.5%, compared to 2% on other datasets. This might be attributed to the minimal generalization disparity between adversarially trained models and their standard-trained counterparts; specifically, while WRN34-10's standard test accuracy stands at around 95%, its adversarial counterpart registers at 87%. It is evident that fine-tuning on a single module may not yield significant improvements. Investigating these patterns further could offer strategies for enhancing the robustness and generalization capabilities of deep neural networks.

6. Conclusion

In this paper, we aim to exploit the redundant capacity of the adversarially trained models. Our proposed RiFT leverages the concept of module robust criticality (MRC) to guide the fine-tuning process, which leads to improved generalization and OOD robustness. The extensive experiments demonstrate the effectiveness of RiFT across various network architectures and datasets. Our findings shed light on the intricate relationship between generalization, adversarial robustness, and OOD robustness. RiFT is a primary exploration of fine-tuning the adversarially trained models. We believe that fine-tuning holds great promise, and we call for more theoretical and empirical analyses to advance our understanding of this important technique.

References

[1] Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*, 2021. 3

[2] Anders Andreassen, Yasaman Bahri, Behnam Neyshabur, and Rebecca Roelofs. The evolution of out-of-distribution robustness throughout fine-tuning. *arXiv preprint arXiv:2106.15831*, 2021. 3, 7

[3] Devansh Arpit, Stanislaw Jastrzkebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pages 233–242. PMLR, 2017. 3

[4] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 274–283. PMLR, 10–15 Jul 2018. 1

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020. 3

[6] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy*, pages 39–57, Los Alamitos, CA, USA, may 2017. IEEE Computer Society. 1

[7] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 3

[8] Alvin Chan, Yi Tay, Yew Soon Ong, and Jie Fu. Jacobian adversarially regularized networks for robustness. In *International Conference on Learning Representations*, 2020. 1

[9] Niladri Chatterji, Behnam Neyshabur, and Hanie Sedghi. The intriguing role of module criticality in the generalization of deep networks. In *International Conference on Learning Representations*, 2020. 2, 3, 4

[10] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1310–1320. PMLR, 09–15 Jun 2019. 1

[11] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, pages 2206–2216. PMLR, 2020. 1

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 3

[13] Justin Gilmer, Nicolas Ford, Nicholas Carlini, and Ekin Cubuk. Adversarial examples are a natural consequence of test error in noise. In *International Conference on Machine Learning*, pages 2280–2289. PMLR, 2019. 1

[14] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 1

[15] Sven Gowal, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann. Improving robustness using generated data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4218–4233. Curran Associates, Inc., 2021. 1

[16] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 3

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 6, 7

[18] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721. PMLR, 2019. 1, 3, 6

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[20] Adel Javanmard, Mahdi Soltanolkotabi, and Hamed Hassani. Precise tradeoffs in adversarial training for linear regression. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2034–2078. PMLR, 09–12 Jul 2020. 2

[21] Klim Kireev, Maksym Andriushchenko, and Nicolas Flammarion. On the effectiveness of adversarial training against

common corruptions. In *Uncertainty in Artificial Intelligence*, pages 1012–1021. PMLR, 2022. 1

[22] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 491–507. Springer, 2020. 3

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6, 7

[24] Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022. 2, 3, 7

[25] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5905–5914. PMLR, 18–24 Jul 2021. 4

[26] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 6

[27] Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *The Eleventh International Conference on Learning Representations*, 2023. 3

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 1, 3, 6

[29] John P Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7721–7735. PMLR, 18–24 Jul 2021. 3

[30] Aamir Mustafa, Salman Khan, Munawar Hayat, Roland Goecke, Jianbing Shen, and Ling Shao. Adversarial defense by restricting the hidden space of deep neural networks. In *The IEEE International Conference on Computer Vision*, October 2019. 1

[31] Tianyu Pang, Min Lin, Xiao Yang, Jun Zhu, and Shuicheng Yan. Robustness and accuracy could be reconcilable by (Proper) definition. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17258–17277. PMLR, 17–23 Jul 2022. 1, 2, 3, 7

[32] Tianyu Pang, Kun Xu, Yinpeng Dong, Chao Du, Ning Chen, and Jun Zhu. Rethinking softmax cross-entropy loss for adversarial robustness. In *International Conference on Learning Representations*, 2020. 1

[33] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In *International Conference on Learning Representations*, 2021. 6

[34] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 3

[35] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7909–7919. PMLR, 13–18 Jul 2020. 1, 2, 3

[36] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan A Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021. 1

[37] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8093–8104. PMLR, 13–18 Jul 2020. 6

[38] Amir Rosenfeld and John K Tsotsos. Intriguing properties of randomly weighted networks: Generalizing while learning next to nothing. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 9–16. IEEE, 2019. 3

[39] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020. 3

[40] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 1, 2

[41] David Stutz, Matthias Hein, and Bernt Schiele. Relating adversarially robust generalization to flat minima. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7807–7817, 2021. 4

[42] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147. PMLR, 2013. 5, 6

[43] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1

[44] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019. 1, 2, 7

[45] Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow net-

works. *Advances in Neural Information Processing Systems*, 29, 2016. 2, 3

[46] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020. 2, 3, 7

[47] Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7959–7971, June 2022. 3, 5, 7

[48] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2958–2969. Curran Associates, Inc., 2020. 2, 3, 4, 7

[49] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018. 1

[50] Mingyang Yi, Lu Hou, Jiacheng Sun, Lifeng Shang, Xin Jiang, Qun Liu, and Zhiming Ma. Improved ood generalization via adversarial training and pretraing. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11987–11997. PMLR, 18–24 Jul 2021. 1, 7

[51] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference*, pages 87.1–87.12. BMVA Press, September 2016. 6

[52] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017. 2, 3

[53] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal? *Journal of Machine Learning Research*, 23(67):1–28, 2022. 3, 4

[54] Haichao Zhang and Jianyu Wang. Defense against adversarial attacks using feature scattering-based adversarial training. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 1

[55] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7472–7482. PMLR, 09–15 Jun 2019. 1, 2, 3, 7

[56] Jingfeng Zhang, Jianing Zhu, Gang Niu, Bo Han, Masashi Sugiyama, and Mohan Kankanhalli. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations*, 2021. 2, 3

A. Proof of the scale-invariant property

Without loss of generality, assume a two layers neural network f and ϕ is a ReLU-based activation function.

$$f(\boldsymbol{\theta}_f, \boldsymbol{x}) = \boldsymbol{\theta}^{(2)}\phi(\boldsymbol{\theta}^{(1)}\boldsymbol{x}). \tag{10}$$

The corresponding scaled neural network g is:

$$g(\boldsymbol{\theta}_g, \boldsymbol{x}) = \frac{1}{\beta}\boldsymbol{\theta}^{(2)}\phi(\beta\boldsymbol{\theta}^{(1)}\boldsymbol{x}), \tag{11}$$

where the non-negative β is the scaling factor.

Suppose we calculate the MRC value of the first module $\boldsymbol{\theta}^{(1)}$ and $\frac{1}{\beta}\boldsymbol{\theta}^{(1)}$.

Theorem A.1. The rectified function $\phi(x) = \max(x, 0)$ is a homogeneous function where

$$\forall (z, \beta) \in \mathbb{R} \times \mathbb{R}^+,\ \phi(\beta z) = \beta\phi(z). \tag{12}$$

Proof.

$$\phi(\beta z) = \max(\beta z, 0) = \beta\max(z, 0) = \beta\phi(z). \tag{13}$$

□

Theorem A.2. $\forall \boldsymbol{x}, f(\boldsymbol{\theta}_f, \boldsymbol{x}) \equiv g(\boldsymbol{\theta}_g, \boldsymbol{x})$.

Proof.

$$\begin{aligned} g(\boldsymbol{\theta}_g, \boldsymbol{x}) &= \frac{1}{\beta}\boldsymbol{\theta}^{(2)}\phi(\beta\boldsymbol{\theta}^{(1)}\boldsymbol{x}) && (14)\\ &\equiv \frac{1}{\beta}\beta\boldsymbol{\theta}^{(2)}\phi(\boldsymbol{\theta}^{(1)}\boldsymbol{x}) && (15)\\ &\equiv \boldsymbol{\theta}^{(2)}\phi(\boldsymbol{\theta}^{(1)}\boldsymbol{x}) && (16)\\ &\equiv f(\boldsymbol{\theta}_f, \boldsymbol{x}) && (17) \end{aligned}$$

□

Theorem A.3. The robust losses of f and g are equal:

$$\mathcal{R}(f(\boldsymbol{\theta}_f), \mathcal{D}) \equiv \mathcal{R}(g(\boldsymbol{\theta}_g), \mathcal{D}). \tag{18}$$

Proof. According to Theorem A.2,

$$\forall \boldsymbol{x} + \Delta\boldsymbol{x}, f(\boldsymbol{\theta}_f, \boldsymbol{x} + \Delta\boldsymbol{x}) \equiv g(\boldsymbol{\theta}_g, \boldsymbol{x} + \Delta\boldsymbol{x}). \tag{19}$$

Thus,

$$\begin{aligned} &\max_{\Delta\boldsymbol{x}\in\mathcal{S}} \ell(f(\boldsymbol{\theta}_f, \boldsymbol{x} + \Delta\boldsymbol{x}), y) && (20)\\ \equiv &\max_{\Delta\boldsymbol{x}\in\mathcal{S}} \ell(g(\boldsymbol{\theta}_g, \boldsymbol{x} + \Delta\boldsymbol{x}), y). && (21) \end{aligned}$$

Thus,

$$\begin{aligned} \mathcal{R}(f(\boldsymbol{\theta}_f), \mathcal{D}) &= \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \max_{\Delta\boldsymbol{x}\in\mathcal{S}} \ell(f(\boldsymbol{\theta}_f, \boldsymbol{x} + \Delta\boldsymbol{x}), y) && (22)\\ &\equiv \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \max_{\Delta\boldsymbol{x}\in\mathcal{S}} \ell(g(\boldsymbol{\theta}_g, \boldsymbol{x} + \Delta\boldsymbol{x}), y) && (23)\\ &= \mathcal{R}(g(\boldsymbol{\theta}_g), \mathcal{D}) && (24) \end{aligned}$$

□

Theorem A.4. The Module Robustness Criticality (MRC) proposed in Definition 3.1 is invariant to the scaling of the parameters.

Proof. Let $\Delta\boldsymbol{\theta}_f = \{\Delta\boldsymbol{\theta}_f^{(1)}, \mathbf{0}\}, \Delta\boldsymbol{\theta}_g = \{\Delta\boldsymbol{\theta}_g^{(1)}, \mathbf{0}\}$ be the perturbation of the first layer for network f and g respectively. First, we prove

$$\begin{aligned} &\max_{\Delta\boldsymbol{\theta}_f\in\mathcal{C}_{\boldsymbol{\theta}_f}} \mathcal{R}(f(\boldsymbol{\theta}_f + \Delta\boldsymbol{\theta}_f), \mathcal{D}) && (25)\\ \leq &\max_{\Delta\boldsymbol{\theta}_g\in\mathcal{C}_{\boldsymbol{\theta}_g}} \mathcal{R}(g(\boldsymbol{\theta}_g + \Delta\boldsymbol{\theta}_g), \mathcal{D}). && (26) \end{aligned}$$

Let

$$\begin{aligned} \Delta\boldsymbol{\theta}_f^* &= \arg\max_{\Delta\boldsymbol{\theta}_f\in\mathcal{C}_{\boldsymbol{\theta}_f}} \mathcal{R}(f(\boldsymbol{\theta}_f + \Delta\boldsymbol{\theta}_f), \mathcal{D}), && (27)\\ \Delta\boldsymbol{\theta}_g^* &= \arg\max_{\Delta\boldsymbol{\theta}_g\in\mathcal{C}_{\boldsymbol{\theta}_g}} \mathcal{R}(g(\boldsymbol{\theta}_g + \Delta\boldsymbol{\theta}_g), \mathcal{D}). && (28) \end{aligned}$$

Consider the perturbation $\Delta\tilde{\boldsymbol{\theta}_g} = \beta\Delta\boldsymbol{\theta}_f^*$ for g, it is easy to show that $\Delta\tilde{\boldsymbol{\theta}_g} \in \mathcal{C}_{\theta_g}$,

$$\begin{aligned} \mathcal{C}_{\boldsymbol{\theta}_f} &= \{\Delta\boldsymbol{\theta}_f \mid \|\Delta\boldsymbol{\theta}_f\|_p \leq \epsilon\|\boldsymbol{\theta}_f^{(1)}\|_p\}, && (29)\\ \mathcal{C}_{\boldsymbol{\theta}_g} &= \{\Delta\boldsymbol{\theta}_g \mid \|\Delta\boldsymbol{\theta}_g\|_p \leq \epsilon\|\boldsymbol{\theta}_g^{(1)}\|_p\} && (30)\\ &= \{\Delta\boldsymbol{\theta}_g \mid \Delta\boldsymbol{\theta}_g = \beta\|\Delta\boldsymbol{\theta}_f\|_p \leq \epsilon\beta\|\boldsymbol{\theta}_f^{(1)}\|_p\}. && (31) \end{aligned}$$

Therefore,

$$\begin{aligned} \mathcal{R}(g(\boldsymbol{\theta}_g + \Delta\tilde{\boldsymbol{\theta}_g}), \mathcal{D}) &\leq \max_{\Delta\boldsymbol{\theta}_g\in\mathcal{C}_{\boldsymbol{\theta}_g}} \mathcal{R}(g(\boldsymbol{\theta}_g + \Delta\boldsymbol{\theta}_g), \mathcal{D}) && (32)\\ &= \mathcal{R}(g(\boldsymbol{\theta}_g + \Delta\boldsymbol{\theta}_g^*), \mathcal{D}). && (33) \end{aligned}$$

Repeat the same analysis as presented in Theorem A.2,

$$\begin{aligned} &g(\boldsymbol{\theta}_g + \Delta\tilde{\boldsymbol{\theta}_g}) && (34)\\ =&\frac{1}{\beta}\boldsymbol{\theta}^{(2)}\phi((\beta\boldsymbol{\theta}^{(1)} + \beta\Delta\boldsymbol{\theta}_f^*)\boldsymbol{x}) && (35)\\ =&\boldsymbol{\theta}^{(2)}\phi((\boldsymbol{\theta}^{(1)} + \Delta\boldsymbol{\theta}_f^*)\boldsymbol{x}) && (36)\\ \equiv&f(\boldsymbol{\theta}_f + \Delta\boldsymbol{\theta}_f^*). && (37) \end{aligned}$$

According to Theorem A.3,

$$\begin{aligned} \mathcal{R}(f(\boldsymbol{\theta}_f + \Delta\boldsymbol{\theta}_f^*), \mathcal{D}) &\equiv \mathcal{R}(g(\boldsymbol{\theta}_g + \Delta\tilde{\boldsymbol{\theta}_g}), \mathcal{D}) && (38)\\ &\leq \mathcal{R}(g(\boldsymbol{\theta}_g + \Delta\boldsymbol{\theta}_g^*), \mathcal{D}). && (39) \end{aligned}$$

Similarly, we can prove

$$\begin{aligned} &\max_{\Delta\boldsymbol{\theta}_g\in\mathcal{C}_{\boldsymbol{\theta}_g}} \mathcal{R}(g(\boldsymbol{\theta}_g + \Delta\boldsymbol{\theta}_g), \mathcal{D}) && (40)\\ \leq &\max_{\Delta\boldsymbol{\theta}_f\in\mathcal{C}_{\boldsymbol{\theta}_f}} \mathcal{R}(f(\boldsymbol{\theta}_f + \Delta\boldsymbol{\theta}_f), \mathcal{D}). && (41) \end{aligned}$$

Thus,

$$\begin{aligned} &\max_{\Delta\boldsymbol{\theta}_f\in\mathcal{C}_{\boldsymbol{\theta}_f}} \mathcal{R}(f(\theta_f + \Delta\boldsymbol{\theta}_f), \mathcal{D}) && (42)\\ = &\max_{\Delta\boldsymbol{\theta}_g\in\mathcal{C}_{\boldsymbol{\theta}_g}} \mathcal{R}(g(\theta_g + \Delta\boldsymbol{\theta}_g), \mathcal{D}). && (43) \end{aligned}$$

Such that the proof ends. □

B. Algorithm of RiFT

The complete algorithm of RiFT is presented in Algorithm 2.

Algorithm 2 Robust Critical Fine-Tuning

```
Input: adversarially trained model weights $\boldsymbol{\theta}_{AT}$, standard
    dataset $\mathcal{D}_{std}$, weight perturbation scaling factor $\alpha$, fine-
    tuning optimization iteration steps $T$ and learning rate
    $\gamma$, weight decay facotr $\lambda$.
Output: The fine-tuned model weights $\boldsymbol{\theta}^*_{AT}$.
Step 1: Calculate MRC for each module
for Module weight $\boldsymbol{\theta}^{(j)}$ do
    Calculate MRC value of $\boldsymbol{\theta}^{(j)}$ using Algorithm 1.
end for
Select the module with lowest MRC value, denote as
    non-robust critical module $\boldsymbol{\theta}^{(i)}$
Step 2: Fine-tuning on Non-robust critical module
$\boldsymbol{\theta}_1 = \boldsymbol{\theta}_{AT}$
for $t = 1, \dots, T$ do                          ▷ Fine-tuning $T$ epochs
    for Batch $\mathcal{B}_k \in \mathcal{D}_{std}$ do
        Calculate loss: $\mathcal{L}(f(\boldsymbol{\theta}_t), \mathcal{B}_k))$
        $\boldsymbol{\theta}^{(i)}_{t+1} = \boldsymbol{\theta}^{(i)}_{t+1} - \gamma \nabla_{\boldsymbol{\theta}_t}(\mathcal{L})$    ▷ Gradient Descent
    end for
    $\boldsymbol{\theta}_{FT} = \boldsymbol{\theta}_t$ if $\boldsymbol{\theta}_t$ obtain highest std test acc.
end for
Step 3: Interpolation
for $\alpha \in (0, 1, 0.05)$ do
    $\boldsymbol{\theta}_\alpha = (1 - \alpha)\boldsymbol{\theta}_{AT} + \alpha \boldsymbol{\theta}_{FT}$
    $\boldsymbol{\theta}^*_{FT} = \boldsymbol{\theta}_\alpha$ if it reaches best standard test acc while
    preserve the robustness as $\boldsymbol{\theta}_{AT}$.
end for
Return Fine-tuned model weights $\boldsymbol{\theta}^*_{FT}$
```

C. Training Details

C.1. Experiment Environment

All experiments are conducted on a workstation equipped with an NVIDIA GeForce RTX 3090 GPU with 24GB memory and NVIDIA A100 with 80GB memory. The PyTorch version is 1.11.0.

C.2. Adversarial Training Details

For vanilla adversarial training, We set the initial learning rate as 0.1, which decays at 100 and 105 epochs with factor 10. When generating adversarial examples, we set BN as train mode since it usually achieves higher robustness.

When incorporating RiFT with other adversarial training methods, the SCORE method is incorporated with TRADES. For the CIFAR100 training, we ran with three different learning rate and select the best model weights as the one with highest robust accuracy. The hyper-parameter settings are either based on their original paper or same as the vanilla AT, depends on which method achieves better robust accuracy.

C.3. Fine-tuning Details

The hyper-parameter that most affects fine-tuning is the initial learning rate. According to our experience, we find a small learning rate usually performs better. If the adversarial robustness of the final fine-tuned weights is still higher than the robustness of the initial adversarial training, we then increase the learning rate.

C.4. The MRC value of ResNet34 and WRN34-10

Figure C.1 and Figure C.2 shows the Module Robust Criticality (MRC) value of each module in ResNet34 trained on CIFAR100 and WideResNet34 trained on Tiny-ImageNet, respectively. It can be observed that both models exhibit redundant capacity. Additionally, Figure C.3 and Figure C.4 shows the MRC value of each module in ResNet18 trained on CIFAR100 and Tiny-ImageNet, respectively. As we discussed in Section 5.3 and Section 5.5, ResNet18 has a lower redundant capacity compared to ResNet34 and WideResNet34, and the redundant capacity decreases as the classification task becomes more complex.

C.5. More interpolation results

Figure C.5 shows the interpolation results of different modules of ResNet18 trained on CIFAR100 dataset. It can be observed that fine-tuning on robust-critical module can also help improve generalization and robustness. This does not mean that our MRC is wrong, as we claimed in Section 5.2, fine-tuning on robust-critical module does not necessarily hurt robustness. The MRC provides guidance on which module to fine-tune for optimal results, and still, fine-tuning on non-robust-critical module achieves the highest test accuracy while preserving robustness.

D. Analysis of the complexity of MRC algorithm

When identifying the most non-robust-critical module, it is required to iterate all modules of the model. Suppose a model with n modules, for each module, the calculation complexity depends on the iteration steps in Algorithm 1. Considering the different overheads for each iterative computation of the modules at different locations, for example, when calculating the last module's MRC value, it only requires forward-backward iteration of the last layer of parameters. Thus, the average total forward-backward iteration of each module is $n/2$. In our experiments, we set the learning rate as 1 and the iteration step as 10. Thus, in our experiments, the complexity of MRC algorithm cost $5n$ total forward-backward propagation.

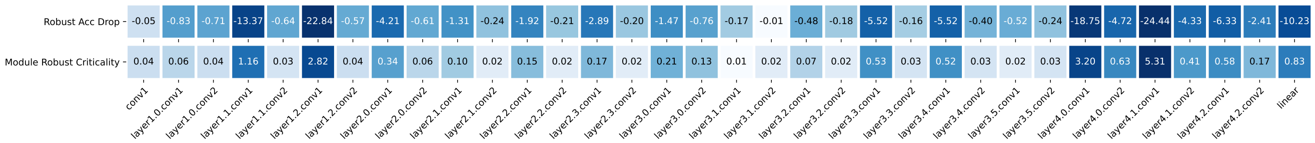

Figure C.1. Example of module robust criticality (MRC) and its corresponding robust accuracy drop of ResNet34 trained on CIFAR100.

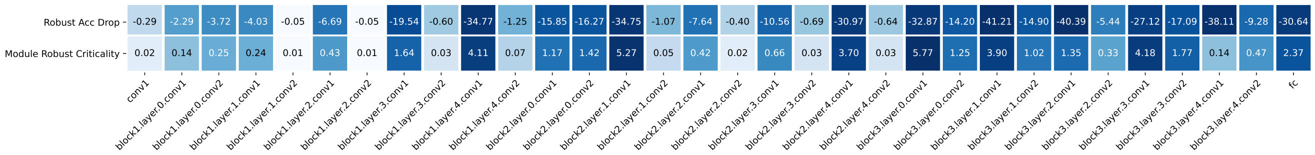

Figure C.2. Example of module robust criticality (MRC) and its corresponding robust accuracy drop of WideResNet34 trained on Tiny-ImageNet.

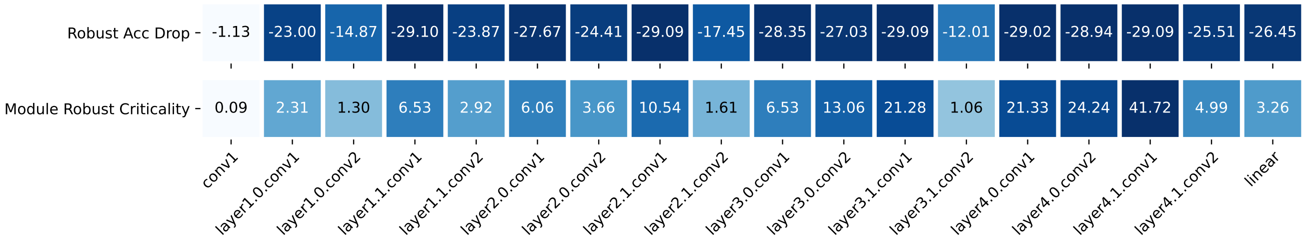

Figure C.3. Example of module robust criticality (MRC) and its corresponding robust accuracy drop of ResNet18 trained on CIFAR100.

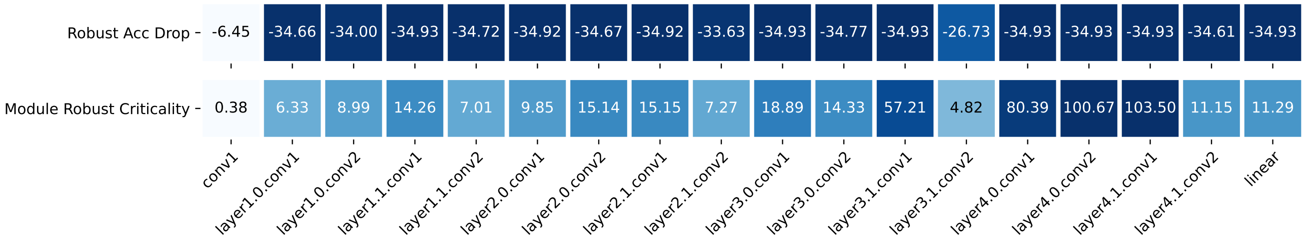

Figure C.4. Example of module robust criticality (MRC) and its corresponding robust accuracy drop of ResNet18 trained on Tiny-ImageNet.

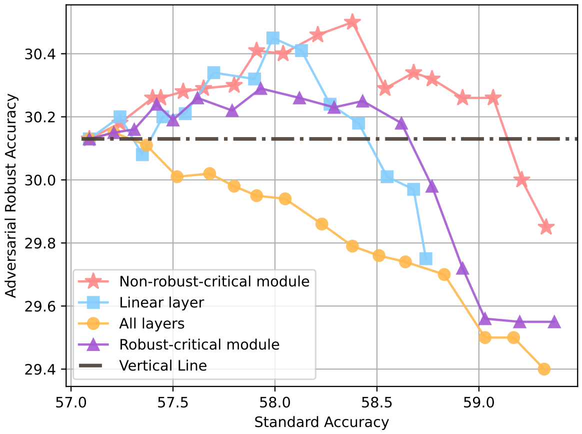

Figure C.5. Interpolation results of fine-tuning on different modules of ResNet18 on CIFAR100 dataset. Dots denote different interpolation points between the final fine-tuned weights of RiFT and the initial adversarially trained weights.