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

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

Let

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

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

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

Therefore,

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

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,

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

Similarly, we can prove

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

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 $\boldsymbol{\theta}^*_{AT}$.
 1: Step 1: Calculate MRC for each module
 2: for Module weight $\boldsymbol{\theta}^{(j)}$ do
 3:     Calculate MRC value of $\boldsymbol{\theta}^{(j)}$ using Algorithm 1.
 4: end for
 5: Select the module with lowest MRC value, denote as non-robust critical module $\boldsymbol{\theta}^{(i)}$
 6: Step 2: Fine-tuning on Non-robust critical module
 7: $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_{AT}$
 8: for $t = 1, \ldots, T$ do                         ▷ Fine-tuning $T$ epochs
 9:     for Batch $\mathcal{B}_k \in \mathcal{D}_{std}$ do
10:         Calculate loss: $\mathcal{L}(f(\boldsymbol{\theta}_t), \mathcal{B}_k))$
11:         $\boldsymbol{\theta}^{(i)}_{t+1} = \boldsymbol{\theta}^{(i)}_{t+1} - \gamma \nabla_{\boldsymbol{\theta}_t}(\mathcal{L})$   ▷ Gradient Descent
12:     end for
13:     $\boldsymbol{\theta}_{FT} = \boldsymbol{\theta}_t$ if $\boldsymbol{\theta}_t$ obtain highest std test acc.
14: end for
15: Step 3: Interpolation
16: for $\alpha \in (0, 1, 0.05)$ do
17:     $\boldsymbol{\theta}_\alpha = (1-\alpha)\boldsymbol{\theta}_{AT} + \alpha\boldsymbol{\theta}_{FT}$
18:     $\boldsymbol{\theta}^*_{FT} = \boldsymbol{\theta}_\alpha$ if it reaches best standard test acc while preserve the robustness as $\boldsymbol{\theta}_{AT}$.
19: end for
20: 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.

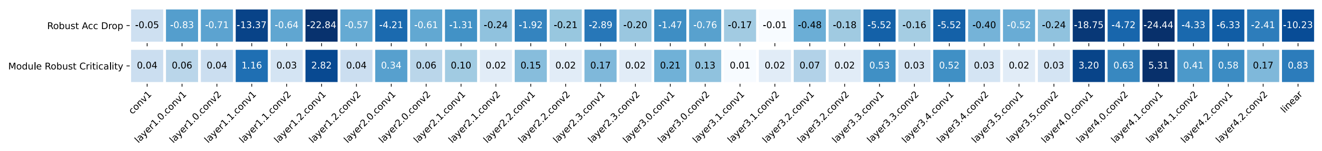

Figure C.1. Example of module robust criticality (MRC) and its corresponding robust accuracy drop of ResNet34 trained on CIFAR100.

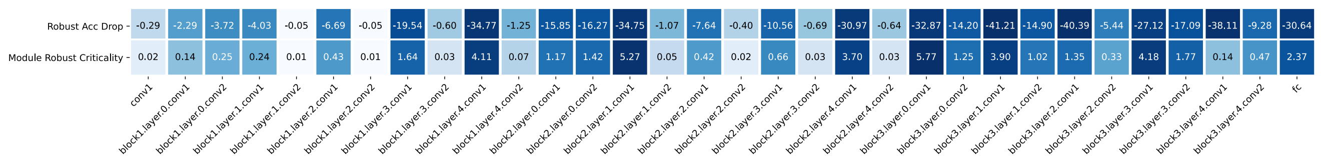

Figure C.2. Example of module robust criticality (MRC) and its corresponding robust accuracy drop of WideResNet34 trained on Tiny-ImageNet.

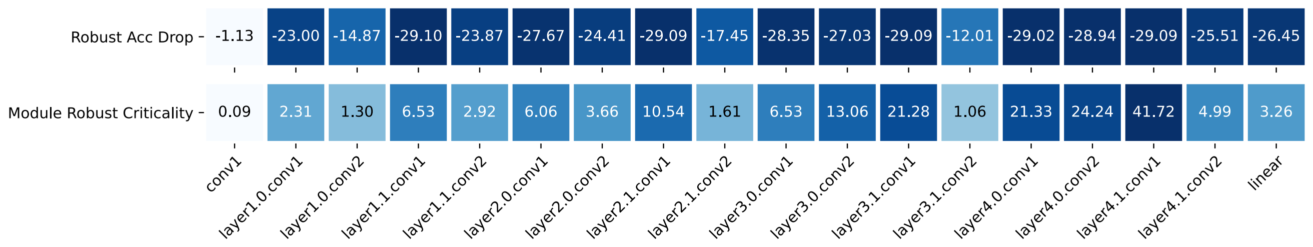

Figure C.3. Example of module robust criticality (MRC) and its corresponding robust accuracy drop of ResNet18 trained on CIFAR100.

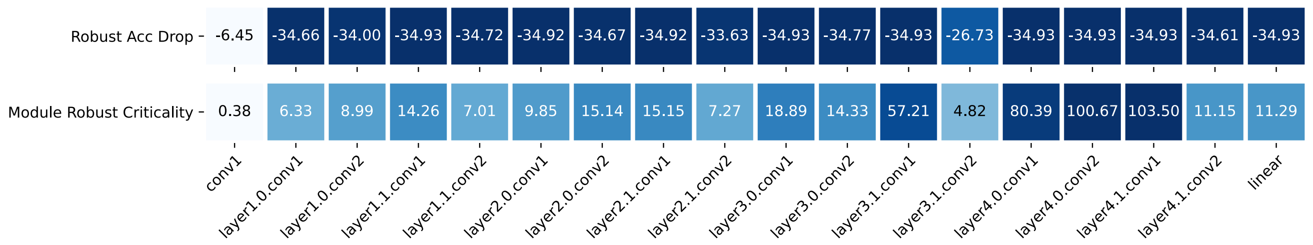

Figure C.4. Example of module robust criticality (MRC) and its corresponding robust accuracy drop of ResNet18 trained on Tiny-ImageNet.

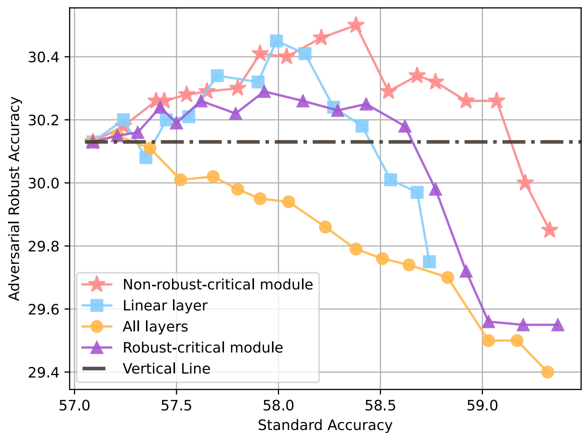

Figure C.5. Interpolation results of fine-tuning on different modules of ResNet18 on CIFAR100 dataset. Dots denote different interpolation points between the final fine-tuned weights of RiFT and the initial adversarially trained weights.