A. Spectrum Normalization

In this section, we briefly recap spectrum normalization (Miyato et al., 2018) for NEFT.

Without loss of generality, we define a linear function $f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{W} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^{m\times 1}$. The Lipschitz constant of f is upper bounded by its largest singular value $\sigma(\boldsymbol{W})$, *i.e.*, the largest eigenvalue of $\boldsymbol{W}^T\boldsymbol{W}$. Because directly calculating $\sigma(\boldsymbol{W})$ is costly, we can use an iterative method to approximate it. To do so, we define two vectors, $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^n$, and iteratively calculate $\sigma(\boldsymbol{W})$ by

$$\begin{aligned} \sigma(\boldsymbol{W}) &= \boldsymbol{u}^T\boldsymbol{W}\boldsymbol{v}, \\ \boldsymbol{v}_{t+1} &= \boldsymbol{W}^T\boldsymbol{u}_t / \parallel \boldsymbol{W}^T\boldsymbol{u}_t \parallel_2, \\ \boldsymbol{u}_{t+1} &= \boldsymbol{W}\boldsymbol{v}_{t+1} / \parallel \boldsymbol{W}\boldsymbol{v}_{t+1} \parallel_2 . \end{aligned} \tag{13}$$

To constrain the Lipschitz constant of f to be one, we divide $\boldsymbol{W}$ by $\sigma(\boldsymbol{W})$.

B. Details of LwF (Shafahi et al., 2020)

This section introduces LwF (Shafahi et al., 2020), which is considered as one of our baselines. In LwF, Shafahi *et al.* (Shafahi et al., 2020) fine-tune all layers of the source model while reducing the difference between the penultimate layer outputs of the target model and the source model. The loss of LwF can be expressed as

$$\begin{aligned} \mathcal{L}_{\text{LwF}} := \mathcal{L}_{\text{CE}}(f(\boldsymbol{x};\boldsymbol{\theta}), y) \\ + \lambda_d \cdot \parallel f^{(L-1)}(\boldsymbol{x};\boldsymbol{\theta}) - f^{(L-1)}(\boldsymbol{x};\boldsymbol{\theta}_0) \parallel_2, \end{aligned} \tag{14}$$

where $\boldsymbol{\theta}_0$ is the original parameters of the source model, and $\boldsymbol{\theta}$ is the target models' parameter. The hyper-parameter λ_d controls the trade-off between the target domain accuracy and the inherited robustness. In Figure 4, we follow the settings in (Shafahi et al., 2020) where $\lambda_d = 0.1$, 0.01, 0.005, and 0.001, and we provide detailed results in Table 5. For other results of LwF (*i.e.*, Table 4, Figure 5), we let $\lambda_d = 0.1$, where it achieves best transferred robustness.

Table 5: Accuracy and robustness of target models transferred from CIFAR-100 to CIFAR-10 using LwF.

λ_d	Acc.(%)	Rob.(%)
0.1	74.87	17.59
0.01	81.45	16.67
0.005	84.86	8.59
0.001	89.87	0.22

C. Experiment Settings

In this section, we provide the experiment settings for our evaluations.

- We adopt SGD with a momentum of 0.9 as the optimizer for training all models. The learning rate is initialized as 0.1 and decays at epoch 40, 70, and 90 by a rate of 0.2. We train models with a batch size of 128 and set the training epoch as 100. Similar settings are also applied during transfer learning.
- For adversarial training, we utilize PGD-7 to generate adversarial examples during training source models. The ℓ_∞-norm constraint (*i.e.*, ϵ) is $8/255$, and the step-size is $2/255$.
- We set the number of iterations for the spectrum normalization to be one to avoid introducing extra computational overhead, and the experimental results demonstrate a perfect approximation.
- In the model evaluation, we report both the accuracy and robustness of the trained model on the entire test set. Specifically, for the robustness evaluation, we adopt the PGD-100 attack implemented by Foolbox[4], an adversarial attack framework, with the step-size of $2/255$ and $\epsilon = 8/255$.
- When we transfer robust source models to the target domain with CARTL and the vanilla method, we freeze all BN layers' affine parameters, including the feature extractor and the sub-model.
- The network architectures used in Section 6 are detailed in Table 6.

Table 6: Network architecture configuration of experiments for investigating BN layers' effect.

Source	Target	Arch.
CIFAR-100	CIFAR-10	WRN 34-10
CIFAR-10	GTSRB	WRN 28-4
CIFAR-10	SVHN	WRN 28-4

[4]https://github.com/bethgelab/foolbox