CARTL: Cooperative Adversarially-Robust Transfer Learning

Dian Chen [1] **Hongxin Hu** [2] **Qian Wang** [1] **Yinli Li** [1] **Cong Wang** [3] **Chao Shen** [4] **Qi Li** [5]

Abstract

Transfer learning eases the burden of training a well-performed model from scratch, especially when training data is scarce and computation power is limited. In deep learning, a typical strategy for transfer learning is to freeze the early layers of a pre-trained model and fine-tune the rest of its layers on the target domain. Previous work focuses on the accuracy of the transferred model but neglects the transfer of adversarial robustness. In this work, we first show that transfer learning improves the accuracy on the target domain but degrades the inherited robustness of the target model. To address such a problem, we propose a novel cooperative adversarially-robust transfer learning (CARTL) by pre-training the model via *feature distance minimization* and fine-tuning the pre-trained model with *non-expansive fine-tuning* for target domain tasks. Empirical results show that CARTL improves the inherited robustness by about 28% at most compared with the baseline with the same degree of accuracy. Furthermore, we study the relationship between the batch normalization (BN) layers and the robustness in the context of transfer learning, and we reveal that freezing BN layers can further boost the robustness transfer.

1. Introduction

The immense progress of deep neural networks (DNNs) leads interactions with machines to a new era. In many fields, DNNs achieve high performance, even better than humans. However, training such a model requires a well-designed network architecture, massive high-quality training data, and extensive computational resources. Obviously, it is impractical for small-scale scenarios due to limited GPUs or insufficient training datasets.

When further implementing DNNs, we are facing more problems. Numerous research efforts have revealed the brittle robustness of DNNs, which hinders their applications in many security-critical scenarios. Previous work on adversarial examples (Szegedy et al., 2014; Papernot et al., 2016a; Moosavi-Dezfooli et al., 2016; Carlini & Wagner, 2017; Kurakin et al., 2017) demonstrated that DNNs can be deceived when given the input with a carefully-crafted perturbation. To solve this problem, *adversarial training* (Goodfellow et al., 2015; Madry et al., 2018; Kannan et al., 2018) has been considered as a promising defensive approach for improving the adversarial robustness of DNNs. The key idea of these approaches is to generate adversarial examples during model training and add them to training datasets. An extra computational burden, however, is introduced to the model training process.

Regarding the above prerequisites of model training, prior work (Pan & Yang, 2010; Bengio, 2012; Yosinski et al., 2014) proposed *transfer learning* to obtain high-performance DNN models with significantly reduced efforts. It can greatly ease the burden in the (adversarially) training process, especially for those with limited capabilities. Thus, it has been considered as a promising machine learning as a service (MLaaS) technique in the industry (Liakhovich & Mbemba, 2017; Li & Li, 2018). The idea of transfer learning is similar to the knowledge transfer in the human world, where the knowledge obtained from the *source domain* is applied to the *target domain* for improving model performance. For DNNs, the "knowledge" is included in the weights of models. We call the model trained on the source domain the *source model* and the one for the target domain the *target model* (Utrera et al., 2021).

So far, most research efforts have mainly been devoted to improving the accuracy of the target model (Kornblith et al., 2019; Utrera et al., 2021; Salman et al., 2020), but neglecting its robustness. The most recent work from Shafahi *et al.* (2020) discussed how the robustness transfers in transfer learning and pointed out that the target model can in-

[1]School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, Hubei, China [2]Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260, USA [3]Department of Computer Science, City University of Hong Kong, HK SAR, China [4]School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, Shanxi, China [5]Institute for Network Sciences and Cyberspace & BNRist, Tsinghua University, Beijing 100084, China. Correspondence to: Qian Wang <qianwang@whu.edu.cn>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

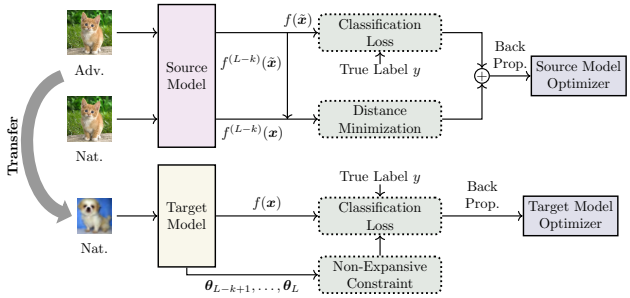

Figure 1: Overview of CARTL.

herit the robustness from an adversarially pre-trained source model. However, it suggested only fine-tuning the *last* fully-connected layer for inheriting robustness, which fails to cover more general scenarios where the target model requires to fine-tune *multiple* layers in transfer learning (Wang et al., 2018; Utrera et al., 2021).

In this work, we first provide a complete evaluation for the robustness transfer and demonstrate that the robustness transfer is highly affected by the transfer strategy, namely, the number of fine-tuned layers during transfer learning. Specifically, we transfer a robust source model, which is adversarially trained on the source domain, to the target domain while freezing its first few layers. Our evaluation indicates that as the number of fine-tuned layers increases, the target model's accuracy also improves. However, its robustness only improves at the beginning but soon starts to decrease afterward.

Based on the above observation, we find that training more layers enables the target model to adapt to the target domain, which is good for improving accuracy on the target domain. At the same time, modifying more early layers of the source model, which can be seen as a robust feature extractor, reduces the robustness of the target model. To improve this trade-off, we present a new approach, cooperative adversarially-robust transfer learning (CARTL), as illustrated in Figure 1. In CARTL, we consider training a robust feature extractor of the source model, which outputs similar features when given natural inputs and corresponding adversarial inputs, and we call this *feature distance minimization*. In addition, to further reduce the negative effects caused by the feature differences between natural and adversarial inputs, we propose *non-expansive fine-tuning*, which is used to control the Lipschitz constant of the network during fine-tuning. We call our method *cooperative* because it consists of an adjusted adversarial training on the source model side and a constrained fine-tuning on the target model side. We emphasize that CARTL requires *no adversarial examples* during fine-tuning, which is far more efficient than adversarial training.

Besides, we conduct a study, for the first time to our best knowledge, on the relationship between the batch normalization (BN) layers and the robustness in the context of transfer learning. Our results reveal that selectively freezing BN layers' parameters helps boost the robustness transfer. We conduct extensive experiments on several transfer learning scenarios and observe that the target model freezing affine parameters of BN layers obtains higher robustness with negligible loss of accuracy. We also show that though BN layers' statistics play a crucial role in the robustness transfer, it will degrade the target model's accuracy.

We summarize our main contributions as follows:

- Through experimental analysis, we reveal that there is a trade-off between accuracy and robustness during transfer learning, which has been overlooked by prior work. Specifically, the target model obtains higher accuracy on the target domain as it fine-tunes more layers. However, as the number of fine-tuned layers increases, the target model's robustness is greatly affected and eventually severely degraded.
- We propose a new transfer learning strategy, CARTL, for improving the accuracy-robustness trade-off of the target model. Our experimental evaluations on broadly-used datasets show that our design improves the target model's inherited robustness while gaining competitive accuracy on the target domain.
- We also conduct extensive experiments on several transfer learning scenarios to demonstrate that selectively freezing the BN layers can further boost the robustness transfer.

2. Related Work

Various techniques focusing on defending adversarial examples have been proposed (Papernot et al., 2016b; Papernot & McDaniel, 2017; Xie et al., 2019; Song et al., 2018). However, many defenses were still proven to be vulnerable to stronger attacks (Carlini & Wagner, 2017; Athalye et al., 2018; Tramer et al., 2020). Despite the fails of many defenses, adversarial training (Szegedy et al., 2014; Kurakin et al., 2017; Madry et al., 2018; Kannan et al., 2018) is still widely regarded as a promising defense for protecting trained models from adversarial examples and has been extensively discussed (Tramèr et al., 2018; Schmidt et al., 2018; Tsipras et al., 2019; Tramer & Boneh, 2019; Zhang et al., 2019b).

Goodfellow *et al.* (Goodfellow et al., 2015) firstly observed that adding adversarial examples into the training datasets improves adversarial robustness, and the strategy is called adversarial training. Madry *et al.* (Madry et al., 2018) used a strong attack method to generate adversarial examples during training, which demonstrated that the trained model is

robust to single-step attacks as well as multi-step attacks. So far, compared with the standard training, the main drawback of adversarial training is the degraded efficiency of training, which is introduced by the generation of adversarial examples. Meanwhile, some literature (Shafahi et al., 2019; Zhang et al., 2019a; Wong et al., 2020) works on efficiency optimization for adversarial training. Very recently, Shafahi *et al.* (Shafahi et al., 2020) gave an evaluation of adversarial robustness in transfer learning. They found that the target model can efficiently inherit the adversarial robustness from an adversarially pre-trained model. However, they only considered the target model that fine-tunes the *last* layer of the source model while ignoring fine-tuning more layers of the source model.

Similar to adversarial training, other work (Cisse et al., 2017; Qian & Wegman, 2019; Lin et al., 2019) improves the robustness of the model during the model-training stage with a different idea. The seminal work of Szegedy *et al.* (Szegedy et al., 2014) attributed the vulnerability of adversarial examples to the instability of the model, which can be mitigated by controlling the Lipschitz constant of the model. Based on this idea, Cisse *et al.* (Cisse et al., 2017) proposed to add a regularization term during model training to constrain the Lipschitz constant of the entire model. The Lipschitz constant of the entire model approximates to one, which makes the model's final prediction less sensitive to little perturbations. The following work (Qian & Wegman, 2019) relaxed the training limitations of Cisse *et al.* (Cisse et al., 2017), which forces the weight matrix of each layer to be orthogonal, providing more freedom for training.

We emphasize that our work is orthogonal to prior work on domain adaptation (Shu et al., 2018). In this work, we focus on transferring the adversarial robustness from the source domain to the target domain, while domain adaptation refers to leveraging the source model's knowledge to improve the accuracy of the target model. Besides, recent work (Utrera et al., 2021; Salman et al., 2020) also reveals that an adversarially pre-trained model tends to improve the target model's accuracy. We leave the studies of the connection between the source domain robustness and the target domain accuracy for our future work.

3. Preliminary

We consider DNN-based classification tasks and define an L-layer feed-forward DNN model:

$$f(\cdot;\boldsymbol{\theta}) := \left(f^{L}_{\boldsymbol{\theta}_L} \circ f^{L-1}_{\boldsymbol{\theta}_{L-1}} \circ \cdots f^{1}_{\boldsymbol{\theta}_1}\right)(\cdot), \tag{1}$$

which is parameterized by $\boldsymbol{\theta} := \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L\}$. We use f^k to represent the kth layer of the model f and use $f^{(k_1..k_2)}$ to represent layers ranging from k_1 to k_2, *i.e.*, $f^{(k_1..k_2)} := f^{k_2} \circ \cdots \circ f^{k_1}$. We also denote the first k layers as $f^{(k)}$ for shorthand.

To train the model, given a proper loss function $\mathcal{L}$, *e.g.*, cross-entropy loss, we want to find the optimal parameters $\boldsymbol{\theta}^*$ that minimizes the risk:

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\mathcal{L}(f(\boldsymbol{x};\boldsymbol{\theta}),y)\right], \tag{2}$$

where $\mathcal{D}$ is the data distribution of image-label pair $(\boldsymbol{x}, y)$. We define data $\boldsymbol{x} \in [0,1]^d$, where d is the input dimension, and define label $y \in \{0, 1, \ldots, C-1\}$, where C is the number of class labels.

3.1. Adversarial Examples & Adversarial Training

Most adversarial example attacks consider an ℓ_p-norm constrained optimization problem that can be generalized as:

$$\arg\max_{\boldsymbol{\delta}} \mathcal{L}\left(f(\boldsymbol{x}+\boldsymbol{\delta};\boldsymbol{\theta}),y\right) \quad s.t. \parallel \boldsymbol{\delta} \parallel_p \leq \epsilon. \tag{3}$$

The hyper-parameter ϵ guarantees that the perturbation $\boldsymbol{\delta}$ is imperceptible. In our work, we consider ℓ_∞-norm-based attacks (Madry et al., 2018) and let $\epsilon = 8/255$.

We follow (Madry et al., 2018) defining the adversarial training as a saddle-point problem that aims to minimize a variant of the training risk:

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\max_{\|\boldsymbol{\delta}\|_p\leq\epsilon} \mathcal{L}\left(f\left(\boldsymbol{x}+\boldsymbol{\delta};\boldsymbol{\theta}\right),y\right)\right]. \tag{4}$$

The inner maximization problem can be approximated by an iterative version of Eq. (3), known as the projected gradient descent (PGD). The method can be summarized as:

$$\boldsymbol{\delta}^{i+1} := \Pi\left(\boldsymbol{\delta}^i + \alpha \cdot \text{sign}\left(\nabla_{\boldsymbol{\delta}}\mathcal{L}\left(f\left(\boldsymbol{x}+\boldsymbol{\delta}^{\boldsymbol{i}}\right),y\right)\right)\right), \tag{5}$$

where α is the step size. For ℓ_∞-based perturbations, the projection Π clips the noise $\boldsymbol{\delta}$ to the interval $[-\epsilon, \epsilon]$. Compared with single-step attacks, PGD achieves higher error rates since it tends to find the global maxima of Eq. (3). In the following sections, we represent the N-step PGD attack as PGD-N.

3.2. Lipschitz Constant

The Lipschitz constant defines an upper bound of the function's slope. If we can find such an upper bound, we call the function Lipschitz continuous, and formally, it can be described as

$$\parallel f(x) - f(x') \parallel_2 \leq \Lambda \cdot \parallel x - x' \parallel_2, \tag{6}$$

where Λ is the Lipschitz constant. Specifically, we call a function *non-expansive* when $\Lambda \leq 1$. Intuitively, if a function is non-expansive, the deviation of its output is no more than the perturbation on its input. Recall Eq. (1), a

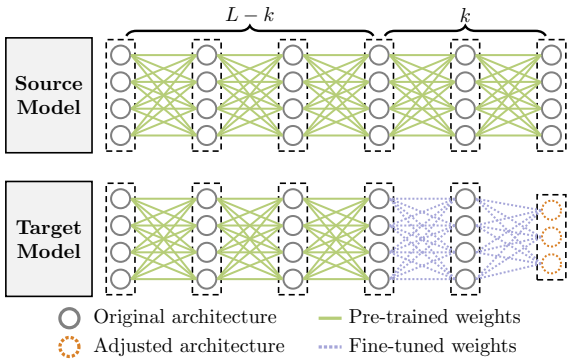

Figure 2: Illustration of transfer learning.

DNN model is stacked in a layer-wise manner. From the Lipschitz continuity perspective, we have

$$\| f(\boldsymbol{x}) - f(\boldsymbol{x}') \|_2 \leq \Lambda_L \cdot \Lambda_{L-1} \cdots \Lambda_1 \| \boldsymbol{x} - \boldsymbol{x}' \|_2, \quad (7)$$

where Λ_i is the Lipschitz constant of the ith layer. We can observe that the Lipschitz constant of the whole model is a product of each layer's Lipschitz constant. When the layers' Lipschitz constants are more than one, a little change of the input may be amplified during forward propagation and result in misclassification (Lin et al., 2019). The above inequality also implies that we can mitigate the vulnerability of adversarial examples by constraining the Lipschitz constant of each layer no more than one (Cisse et al., 2017; Qian & Wegman, 2019; Lin et al., 2019).

3.3. Transfer Learning

The main idea of transfer learning is to transfer "knowledge" from a pre-trained source model to a target model for solving target domain tasks. In deep learning, a widely-adopted method for transfer learning is that the target domain tasks copies the whole *pre-trained* model from the source model, refines the architecture, typically adjusting the last fully-connected layer, and fine-tunes the last $k \in \{1, \ldots, L\}$ layers (Wang et al., 2018; Utrera et al., 2021). Formally, it can be formulated as

$$\arg\min_{\bar{\boldsymbol{\theta}}} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{L} \left(f_{\bar{\boldsymbol{\theta}}}^{(L-k+1..L)} \left(f^{(L-k)}(\boldsymbol{x}) \right), y \right) \right], \quad (8)$$

where $\bar{\boldsymbol{\theta}} := \{\boldsymbol{\theta}_{L-k+1}, \ldots, \boldsymbol{\theta}_L\}$. An illustration of transfer learning is depicted in Figure 2. Intuitively, we can view the output of the first frozen $L-k$ layers as the extracted features of the input $\boldsymbol{x}$, while the fine-tuned part of the target model is a sub-model that directly takes as input those features. During transfer learning, we reuse the powerful feature extractor of the source model (green solid lines) and adapt the sub-model (blue dashed lines) for the target domain task.

4. Problem Statement

The previous work of Shafahi *et al.* (2020) gave an empirical analysis of robustness transfer. However, their exploratory experiments put less attention on the accuracy. Besides, they mainly considered a strategy that all layers but the *last* fully-connected layer are frozen, which does not cover more general cases. In this work, we study how both the robustness and accuracy transfer while the last k layers are fine-tuned.

To see the effect of fine-tuning on the robustness and accuracy, we adversarially train a Wide-ResNet (WRN) 34-10 (Zagoruyko & Komodakis, 2017) on CIFAR-100 and a WRN 28-4 on CIFAR-10 as source models, then transfer them to CIFAR-10 and SVHN, respectively. The source models are trained with PGD-7, and the perturbation is constrained in an ℓ_∞ ball with a radius of $\epsilon = 8/255$. During transferring, we break the source models into blocks and fine-tune them in the unit of blocks (*e.g.*, two layers at once for a WRN block). Then we report the adversarial robustness of the target models against the PGD-100 attack. We emphasize that our settings are different from the exploration in (Shafahi et al., 2020), where the last k blocks were instead fine-tuned on the source domain.

Figure 3 illustrates how both the accuracy and robustness are affected during transfer learning. As can be seen, only retraining the last fully-connected layer fails to guarantee high accuracy for target domain tasks. Besides, the insufficient accuracy also results in lower robustness. If we further fine-tune the last few layers, the model accuracy is increased together with the increased robustness. We attribute this phenomenon to the increment of the accuracy of natural inputs. The accuracy is continuously increased while we fine-tune more layers, but the robustness quickly drops and ends with negligible. The results demonstrate there is a trade-off between the target model's accuracy on the target domain and its robustness inherited from the source model. Besides, simply fine-tuning the last few layers does help the target model inherit the accuracy and robustness in a low cost.

Furthermore, besides the above observations, it is natural to raise another question:

Can the target model obtain high accuracy while inheriting more robustness from the source model?

To answer with this question, we propose a novel strategy for transfer learning, which further improves the accuracy-robustness trade-off during transfer learning.

5. Our Design: CARTL

In this section, to cope with the question raised in Section 4, we propose a new approach, cooperative adversarially-

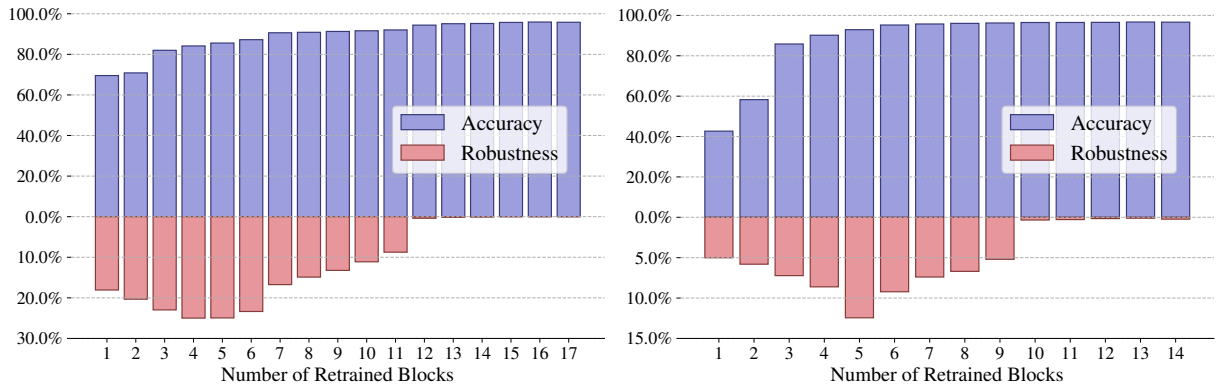

Figure 3: Accuracy and robustness of target models transferred from robust source models. *Left*: A WRN 34-10 model is transferred from CIFAR-100 to CIFAR-10. *Right*: A WRN 28-4 model is transferred from CIFAR-10 to SVHN.

robust transfer learning (CARTL), to improve both the robustness and accuracy of the target model. We divide layers of a robust source model into two parts during transfer learning according to whether they will be retrained. We take the frozen part as a feature extractor like (Utrera et al., 2021), and we propose that it outputs similar features if given natural examples and corresponding adversarial examples. As for the trainable part fine-tuned on the target domain, we aim to reduce the classification error caused by the differences between the features extracted from natural and adversarial inputs. Without loss of generality, we assume that the target model fine-tunes the last k layers and freezes the first $L-k$ layers during transfer learning. We introduce our scheme, starting with training the source model.

5.1. Feature Distance Minimization

Our intuition is that if a specific layer of a model extracts similar features from two inputs, the subsequent layers tend to classify the inputs into identical classes, even they have different labels (Wang et al., 2018). Hence, we consider making the frozen part of the source model a robust feature extractor that can output similar features given natural examples and the corresponding adversarial examples. To do so, we propose *feature distance minimization* (FDM) to reduce the dissimilarity of the extracted features. Specifically, for the first $L-k$ layers that take as input $\boldsymbol{x}$ and output intermediate features $f^{(L-k)}(\boldsymbol{x})$, FDM adds a penalty term to the training loss for two different inputs $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$:

$$\mathcal{L}_{AT} + \lambda \cdot D\left(f^{(L-k)}(\boldsymbol{x}), f^{(L-k)}(\tilde{\boldsymbol{x}})\right). \tag{9}$$

Here, λ is the hyper-parameter controlling the strength of the FDM penalty term, and D is a distance metric measuring the dissimilarity between two features. $\mathcal{L}_{AT}$ is the loss function used during adversarial training, and $\tilde{\boldsymbol{x}}$ is the adversarial example corresponding to the natural example $\boldsymbol{x}$.

Intuitively, we can view extracted features as points in a subspace of $\mathbb{R}^d$, where d is the feature's dimension. For a source model that can extract similar intermediate features, features of natural examples and adversarial examples should be close enough. Thus, we propose to use the Euclidean distance between the natural feature and adversarial feature as the penalty term. Specifically, we adjust the original training loss Eq. (4) to

$$\mathcal{L}_{AT} + \frac{\lambda}{\sqrt{d}} \cdot \sum \parallel f^{(L-k)}(\boldsymbol{x}) - f^{(L-k)}(\tilde{\boldsymbol{x}}) \parallel_2 . \tag{10}$$

5.2. Non-Expansive Fine-tuning

The source model trained with FDM outputs similar features if both natural and adversarial examples are given. However, considering the previous discussion in Eq. (7), the little dissimilarity between the features may be still amplified during propagation in the rest of the network, leading to misclassification. To improve the target model's robustness, we propose to suppress such the amplification effect via controlling the network's Lipschitz constant.

We start with the basic linear layer that can be expressed as $\boldsymbol{x}^{l+1} = \boldsymbol{W}^l\boldsymbol{x}^l + \boldsymbol{b}^l$, where $\boldsymbol{W}^l \in \mathbb{R}^{d_{out}\times d_{in}}$ and $\boldsymbol{b}^l \in \mathbb{R}^{d_{out}}$. It is straightforward that the bias $\boldsymbol{b}^l$ does not affect the Lipschitz constant. Thus, the Lipschitz constant of the linear layer is determined by $\boldsymbol{W}^l$. Since the Lipschitz constant is upper bounded by the spectrum norm of $\boldsymbol{W}^l$, *i.e.*, its maximum singular value (Szegedy et al., 2014), similar to the spectrum normalization (Miyato et al., 2018), we divide weights $\boldsymbol{W}^l$ of last k layers by their spectrum norm $\sigma(\boldsymbol{W}^l)$ as

$$\boldsymbol{W}^l_* := \beta \cdot \frac{\boldsymbol{W}^l}{\sigma(\boldsymbol{W}^l)}, \quad (l = L-k+1, \ldots, L). \tag{11}$$

We also emphasize that different from the naive spectrum normalization, we add a hyper-parameter $\beta \in (0, 1]$ for

Table 1: Effect of selectively freezing BN layers in various scenarios. The third and fourth columns are results of target models freezing affine parameters of the feature extractor. The fifth and sixth rows are results of target models freezing *all* parameters of the feature extractor. For each transfer learning scenario, the first rows are results of target models fine-tuning *all* parameters of the sub-model, and the second rows are those that freeze affine parameters.

		$\boldsymbol{W}, \boldsymbol{b}$		$\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{W}, \boldsymbol{b}$	
		Acc.(%)	Rob.(%)	Acc.(%)	Rob.(%)
CIFAR-100 → CIFAR-10 ($k = 8$)	-	<u>91.17</u>	14.36	90.86	14.89
	$\boldsymbol{W}, \boldsymbol{b}$	90.70	17.41	90.84	**18.54**
CIFAR-10 → GTSRB ($k = 6$)	-	<u>93.02</u>	30.22	89.29	32.22
	$\boldsymbol{W}, \boldsymbol{b}$	92.13	32.22	88.94	**34.53**
CIFAR-10 → SVHN ($k = 6$)	-	<u>95.29</u>	3.88	95.24	9.22
	$\boldsymbol{W}, \boldsymbol{b}$	95.16	4.90	94.86	**11.52**
CIFAR-10 → SVHN ($k = 5$)	-	<u>93.47</u>	4.71	92.92	12.45
	$\boldsymbol{W}, \boldsymbol{b}$	93.41	5.64	92.10	**14.16**

further scaling the Lipschitz constant of the fine-tuned part. The idea comes from the observation that parameters trained with FDM tend to have a smaller (< 1) Lipschitz constant. For the details of $\sigma(\cdot)$, please refer to Appendix A.

For the convolutional layer, we flatten each filter into a vector with $C_{in} \cdot K \cdot K$ dimensions, where C_{in} is the number of input channels, and K is the size of the convolution kernel. We further stack the vectors forming a C_{out}-row matrix, where C_{out} is the number of output channels. Hence, we transform the weight matrix as $\boldsymbol{W} \in \mathbb{R}^{C_{out} \times (K^2 \cdot C_{in})}$. As for the aggregation layer in the residual network (He et al., 2016), which adds the predecessor layer's output with the shortcut connection's output, we instead modify it to a convex combination of its inputs (Cisse et al., 2017) and manually set the weights be $1/n$, where n is the number of its inputs.

6. Rethinking Fine-tuning BN Layers

During our evaluation, we find a strong connection between the BN layer and the transferred robustness. Before presenting experimental evaluations of CARTL, we first investigate how the BN layer affects the target models' robustness in this section. Specifically, we find that selectively freezing the BN layers of source models improves the transferred robustness of target models and generally brings little negative impact on their accuracy.

We first simply recap the basis of the BN layer. In current implementations of the BN layer[1], it usually consists of four parameters, including two running statistics $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, and two affine parameters $\boldsymbol{W}$ and $\boldsymbol{b}$. Typically, a BN layer can be expressed as

$$BN(\boldsymbol{x}) := \boldsymbol{W} \cdot \frac{\boldsymbol{x} - mean(\boldsymbol{x})}{\sqrt{var(\boldsymbol{x}) + \varepsilon}} + \boldsymbol{b}. \tag{12}$$

During training, both the running statistics $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are updated with a momentum based on batch's statistics (*i.e.*, $mean$ and var), while $\boldsymbol{W}$ and $\boldsymbol{b}$ are updated via the gradient descent. During inference, BN layers normalize the activation with running statistics instead of batch's statistics.

To see the effect of BN layers in transfer learning, we divide all BN layers of a source model into two sets according to whether they are in the frozen feature extractor or the fine-tunable sub-model (see Section 3.3). In total, we consider four cases regarding the BN layers, including updating or freezing source model's running statistics (*i.e.*, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$) in the frozen feature extractor, and fine-tuning or freezing source model's affine weights (*i.e.*, $\boldsymbol{W}$ and $\boldsymbol{b}$) in the sub-model. For other cases, we note that both $\boldsymbol{W}$ and $\boldsymbol{b}$ in the feature extractor are naturally frozen in transfer learning, and we also find that freezing running statistics $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ in the fine-tunable layers makes the target model hard to converge.

We conduct experiments on several transfer learning scenarios where robust models are *naively* transferred to target domains, and we present the results in Table 1 (more setups are presented in Appendix C). It is shown that though it slightly degrades accuracy, freezing all affine parameters (*i.e.*, $\boldsymbol{W}$ and $\boldsymbol{b}$) of BN layers can further improve the transferred robustness. For example, the target model's robustness is increased from 14.36% to 17.41%, while the accuracy is decreased by 0.47% if we transfer a robust source model from CIFAR-100 to CIFAR-10. Unlike the analysis in Section 4, where we show that reducing the number of fine-tunable layers harms the accuracy of the target model, we can see

[1] *E.g.*, PyTorch, TensorFlow.

Table 2: Accuracy and robustness of target models transferred to CIFAR-10 under different choices of hyper-parameters, including λ for FDM, β for NEFT and the number of fine-tuned layers.

		NEFT $\beta = 1.0$		NEFT $\beta = 0.6$		NEFT $\beta = 0.4$	
		Acc.(%)	Rob.(%)	Acc.(%)	Rob.(%)	Acc.(%)	Rob.(%)
Case-4	$\lambda = 0.01$	86.09	25.73	86.08	27.17	85.64	28.40
	$\lambda = 0.005$	85.41	25.75	85.47	27.14	85.51	28.47
Case-6	$\lambda = 0.01$	87.78	25.58	87.92	27.27	87.96	29.60
	$\lambda = 0.005$	87.66	25.97	88.07	27.64	87.79	30.94
Case-8	$\lambda = 0.01$	91.85	16.36	91.63	19.22	91.55	27.47
	$\lambda = 0.005$	91.71	17.62	91.10	21.60	91.30	29.34

that merely freezing the BN layers' affine parameters does not aggressively decrease the accuracy while improving the robustness. In addition, reusing the running statistics of the BN layers in the frozen feature extractor plays a crucial role in robustness transfer, *e.g.*, transferring from CIFAR-10 to SVHN. However, it tends to bring more negative effects to the accuracy, especially when transferring from CIFAR-10 to GTSRB. We note that our findings corroborate the recent studies (Xie & Yuille, 2019; Xie et al., 2020), which argue that the BN layers highly relate to robustness.

7. CARTL Evaluation

In this section, we present the experimental results of CARTL compared with the vanilla method and Shafahi's work (2020). We conduct a detailed experimental analysis for the scenario of transferring from CIFAR-100 to CIFAR-10. Besides, more scenarios are also tested to demonstrate the generality of CARTL. We report the robustness of target models under the PGD-100 attack. For more details about experiment settings, please refer to Appendix C, and our codes are available on GitHub[2].

7.1. Improved Robustness-Accuracy Trade-off

Shafahi *et al.* (2020) noticed that merely retraining the last layer maintains the robustness of the source model but results in low accuracy on the target domain. To cope with this problem, they proposed an end-to-end fine-tuning method with learning without forgetting (LwF)[3]. In addition to LwF, we also consider the vanilla method, which refers to simply fine-tune the last few layers of a robust source model on the target domain without additional techniques. Source models for both LwF and the vanilla method are adversarially trained with (Madry et al., 2018).

Figure 4 qualitatively illustrates the robustness-accuracy trade-off achieved by CARTL in comparison with LwF and the vanilla method. It is shown that to obtain high accuracy, LwF significantly degrades the target model's robustness (close to 0%). Moreover, LwF improves the robustness of the target model but aggressively harms its accuracy on the target domain. On the other hand, both the vanilla method and CARTL maintain higher robustness in the case of an equivalent level of accuracy, demonstrating a better robustness-accuracy trade-off. Moreover, our method CARTL further improves the accuracy-robustness trade-off on the target domain during transfer learning. Specifically, it improves the robustness by about 28% compared with LwF when the accuracy is about 90%.

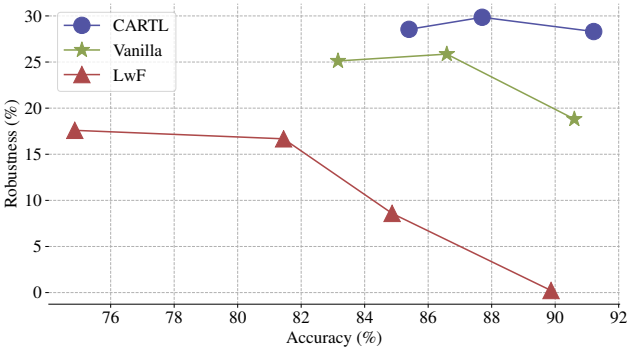

Figure 4: The robustness-accuracy trade-off of target models transferred with CARTL in comparisons with LwF and the vanilla method when robust models transfer from CIFAR-100 to CIFAR-10.

We also notice that a peak point appears in both curves of CARTL and Vanilla. This phenomenon is in line with the observations in Section 4 that the target model's robustness increases when the last few layers are fine-tuned. It implies that there may exist a potential optimal configuration of k for transfer learning, and we leave the corresponding searching strategies for our future work.

[2]https://github.com/NISP-official/CARTL

[3]We give detailed introduction of LwF in Appendix B.

Table 3: Ablation studies of CARTL in the scenario of CIFAR-100 → CIFAR-10.

Method		Case-4		Case-6		Case-8	
Source	Transfer	Acc.(%)	Rob.(%)	Acc.(%)	Rob.(%)	Acc.(%)	Rob.(%)
AT	TL	83.22	25.23	86.92	25.38	90.82	18.54
AT	NEFT	83.72	26.29	86.87	27.95	90.92	29.97
AT + FDM	NEFT	85.51	28.47	87.79	30.94	91.30	29.34

Table 4: Comparison of CARTL with LwF and the vanilla method in multiple scenarios.

Source	Target	Arch.	LwF		Vanilla		CARTL	
			Acc.(%)	Rob.(%)	Acc.(%)	Rob.(%)	Acc.(%)	Rob.(%)
CIFAR-100	SVHN	WRN 34-10 (k=6)	85.90	6.67	92.83	17.64	93.96	**22.21**
CIFAR-100	GTSRB	WRN 34-10 (k=6)	70.34	15.85	80.40	30.25	83.07	**47.34**
CIFAR-10	SVHN	WRN 28-4 (k=6)	94.32	4.68	94.86	11.52	94.76	**21.65**
GTSRB	SVHN	WRN 28-4 (k=6)	81.80	1.08	93.91	6.08	94.07	**15.26**

7.2. Selections for Hyper-parameters

In this subsection, we evaluate how hyper-parameters of CARTL affect the accuracy and robustness of the target model.

First, we test the effect of increasing the number of retrained layers. We report both the target model's accuracy and robustness when fine-tuning the layers of the last 4, 6, and 8 blocks of WRN 34-10, which are denoted as Case-k, and $k = 4, 6, 8$. When we increase the number of retrained blocks, the accuracy generally rises from $\approx 86\%$ to $\approx 91\%$. As for the robustness, CARTL exhibits similar trends to the vanilla method, achieving higher robustness at Case-6.

We further investigate how the hyper-parameter λ affects both the accuracy and robustness of the target models. We observe that for all cases, a smaller λ helps robustness transfer, especially for the Case-8. On the other hand, we can see that reducing λ cast slight negative impacts to the accuracy of target models.

Finally, we evaluate the best choices for the hyper-parameter β. Recall that in Eq. (11), we divide weights of fine-tuned layers by their largest singular value while multiplying a scalar β for further scaling their Lipschitz constants. For example, if we let $\beta = 0.4$, the Lipschitz constants of all fine-tuned layers are reduced to about $\beta^2 = 0.16$. We can see that reducing Lipschitz constants significantly improves the target models' robustness from 17.62% to 29.34% but brings a negligible negative impact to the accuracy.

7.3. Ablation Studies

To test the effect of each component of CARTL on the accuracy and robustness of the target model, we replace part of the components of CARTL with the vanilla method while training the source models and fine-tuning the target models. In Table 3, the first row presents the results using the vanilla method, which is used as the baseline. Recall that the vanilla method is consisted of simply adversarially training the source model with (Madry et al., 2018), denoted as AT, and directly fine-tuning the source model on the target domain, denoted as TL. For the second row, we transfer an adversarially trained model with NEFT, and the last row presents the results achieved by CARTL.

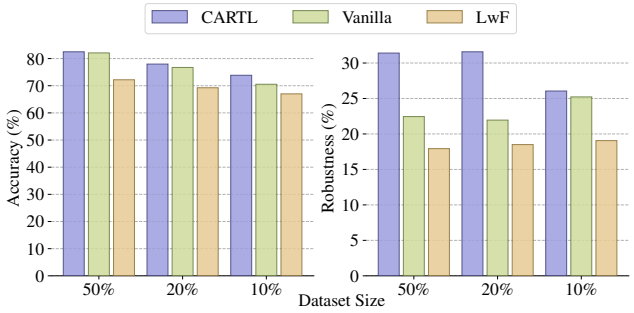

Figure 5: The accuracy and robustness of target models fine-tuned on CIFAR-10 of different sizes. For comparison, we also train target models with LwF and the vanilla method.

We can see that fine-tuning the target model with NEFT significantly increases its robustness by observing the first and second rows. Besides, FDM further improves the robustness except for the Case-8. We attribute this to the reason that with the increasing number of fine-tuned layers, the effect of constraining the features' distance (*i.e.*, FDM) is reduced. On the other hand, the Lipschitz constraint (*i.e.*, NEFT) plays a more important role when we fine-tune more layers. As can be seen from the last column, the robustness increases from 18.54% to 29.97%. Finally, we find that by using FDM, the target model's accuracy slightly rises in all

cases.

7.4. Transfer to Smaller Dataset

In this subsection, we do additional experiments to test the effect of data size on the target model's performance. For the target domain, CIFAR-10, we fine-tune target models on subsets with the sizes of 50%, 20%, and 10%. To mitigate bias, the number of training data for each label is equal. We summarize and plot the results in Figure 5. Fine-tuning the last six blocks with CARTL maintains better accuracy in all cases, and CARTL provides outstanding robustness except on the extremely small training set. However, CARTL still slightly outperforms the vanilla method in that case. In comparison, LwF relatively results in the lowest accuracy and robustness in all cases. Generally speaking, CARTL provides a better accuracy-robustness trade-off for small datasets.

7.5. Studies on Extra Scenarios

To demonstrate the generality of CARTL, we conduct experiments in scenarios including transferring from CIFAR-100 to SVHN, from CIFAR-100 to GTSRB, from CIFAR-10 to SVHN, and from GTSRB to SVHN. According to Table 4, we can see that the target model fine-tuned with CARTL inherits superior robustness from the source model. Besides, CARTL provides comparable accuracy against the vanilla method. Table 4 demonstrates that CARTL can universally improve the accuracy-robustness trade-off. As for LwF, it leads to lower accuracy on the natural inputs and fails to guarantee the robustness transfer. In all scenarios, LwF obtains the lowest robustness.

8. Conclusion & Future Work

In this work, we have revealed the trade-off between the robustness and the accuracy of the target model transferred from an adversarially-trained source model. From our observation, we have proposed CARTL, which consists of feature distance minimization and non-expansive fine-tuning, to help the target model inherit more robustness from the source model while maintaining high accuracy. We have also found that freezing all batch normalization layers' affine parameters can further improve the transferred robustness. We hope our work brings insights to enable the following researchers to build a more robust and accurate model in the transfer learning scenario.

As our future work, we would like to solve the limitation that FDM requires pre-training the source model with an explicitly defined k. Other interesting directions include improving the effect of robustness transfer further and considering more security threats against DNNs.

Acknowledgements

This work was partially supported by the National Key R&D Program of China (2020AAA0107701), the NSFC under Grants U20B2049, 61822207, 61822309, 61773310 and U1736205, RGC HK under GRF projects CityU 11217819 and 11217620, RIF project R6021-20, and BNRist under Grant BNR2020RC01013.

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *Proc. of ICML*, pp. 274–283, July 2018.

Bengio, Y. Deep Learning of Representations for Unsupervised and Transfer Learning. In *Proc. of ICML*, pp. 17–36, June 2012.

Carlini, N. and Wagner, D. Towards Evaluating the Robustness of Neural Networks. In *Proc. of IEEE S&P*, pp. 39–57, May 2017.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *Proc. of ICML*, pp. 854–863, August 2017.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and Harnessing Adversarial Examples. In *Proc. of ICLR*, March 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*, pp. 770–778, 2016.

Kannan, H., Kurakin, A., and Goodfellow, I. Adversarial Logit Pairing. *arXiv preprint arXiv:1803.06373*, March 2018.

Kornblith, S., Shlens, J., and Le, Q. V. Do Better ImageNet Models Transfer Better? In *Proc. of CVPR*, pp. 2661–2671, 2019.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial Machine Learning at Scale. In *Proc. of ICLR*, February 2017.

Li, F. and Li, J. Cloud automl: Making ai accessible to every business. https://blog.google/products/google-cloud/cloud-automl-making-ai-accessible-every-business/, 2018.

Liakhovich, O. and Mbemba, C. Food classification with custom vision service. https://devblogs.microsoft.com/cse/2017/05/12/food-classification-custom-vision-service/, 2017.

Lin, J., Gan, C., and Han, S. Defensive Quantization: When Efficiency Meets Robustness. In *Proc. of ICLR*, April 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. In *Proc. of ICLR*, 2018.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral Normalization for Generative Adversarial Networks. In *Proc. of ICLR*, February 2018.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *Proc. of CVPR*, pp. 2574–2582, June 2016.

Pan, S. J. and Yang, Q. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

Papernot, N. and McDaniel, P. Extending Defensive Distillation. *arXiv preprint arXiv:1705.05264*, May 2017.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. The Limitations of Deep Learning in Adversarial Settings. In *Proc. of EuroS&P*, pp. 372–387, March 2016a.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In *Proc. of IEEE S&P*, pp. 582–597, May 2016b.

Qian, H. and Wegman, M. N. L2-Nonexpansive Neural Networks. In *Proc. of ICLR*, February 2019.

Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., and Madry, A. Do Adversarially Robust ImageNet Models Transfer Better? In *Proc. of NeurIPS*, volume 33, pp. 3533–3545, 2020.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially Robust Generalization Requires More Data. In *Proc. of NeurIPS*, pp. 5014–5026, 2018.

Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! In *Proc. of NeurIPS*, pp. 3358–3369, 2019.

Shafahi, A., Saadatpanah, P., Zhu, C., Ghiasi, A., Studer, C., Jacobs, D., and Goldstein, T. Adversarially robust transfer learning. In *Proc. of ICLR*, February 2020.

Shu, R., Bui, H. H., Narui, H., and Ermon, S. A DIRT-T Approach to Unsupervised Domain Adaptation. In *Proc. of ICLR*, 2018.

Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples. In *Proc. of ICLR*, May 2018.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *Proc. of ICLR*, February 2014.

Tramer, F. and Boneh, D. Adversarial Training and Robustness for Multiple Perturbations. In *Proc. of NeurIPS*, pp. 5866–5876, 2019.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble Adversarial Training: Attacks and Defenses. In *Proc. of ICLR*, 2018.

Tramer, F., Carlini, N., Brendel, W., and Madry, A. On Adaptive Attacks to Adversarial Example Defenses. In *Proc. of NeurIPS*, volume 33, pp. 1633–1645, 2020.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness May Be at Odds with Accuracy. In *Proc. of ICLR*, September 2019.

Utrera, F., Kravitz, E., Erichson, N. B., Khanna, R., and Mahoney, M. W. Adversarially-Trained Deep Nets Transfer Better: Illustration on Image Classification. In *Proc. of ICLR*, 2021.

Wang, B., Yao, Y., Viswanath, B., Zheng, H., and Zhao, B. Y. With Great Training Comes Great Vulnerability: Practical Attacks against Transfer Learning. In *Proc. of USENIX Security*, pp. 1281–1297, 2018.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In *Proc. of ICLR*, January 2020.

Xie, C. and Yuille, A. Intriguing properties of adversarial training at scale. In *Proc. of ICLR*, December 2019.

Xie, C., Wu, Y., van der Maaten, L., Yuille, A. L., and He, K. Feature Denoising for Improving Adversarial Robustness. In *Proc. of CVPR*, pp. 501–509, 2019.

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A. L., and Le, Q. V. Adversarial Examples Improve Image Recognition. In *Proc. of CVPR*, pp. 819–828, 2020.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Proc. of NeurIPS*, pp. 3320–3328, 2014.

Zagoruyko, S. and Komodakis, N. Wide Residual Networks. *arXiv preprint arXiv:1605.07146*, June 2017.

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. You Only Propagate Once: Accelerating Adversarial Training via Maximal Principle. In *Proc. of NeurIPS*, pp. 227–238, 2019a.

Zhang, H., Yu, Y., Jiao, J., Xing, E., Ghaoui, L. E., and Jordan, M. Theoretically Principled Trade-off between Robustness and Accuracy. In *Proc. of ICML*, pp. 7472–7482, May 2019b.

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