RIFLE: Backpropagation in Depth for Deep Transfer Learning through Re-Initializing the Fully-connected LayEr

Xingjian Li [* 1 2]   Haoyi Xiong [* 1]   Haozhe An [1]   Chengzhong Xu [2 3]   Dejing Dou [1]

Abstract

Fine-tuning the deep convolution neural network (CNN) using a pre–trained model helps transfer knowledge learned from larger datasets to the target task. While the accuracy could be largely improved even when the training dataset is small, the transfer learning outcome is usually constrained by the pre-trained model with close CNN weights (Liu et al., 2019), as the backpropagation here brings smaller updates to deeper CNN layers. In this work, we propose **RIFLE**– a simple yet effective strategy that deepens backpropagation in transfer learning settings, through periodically Re-Initializing the Fully-connected LayEr with random scratch during the fine-tuning procedure. **RIFLE** brings meaningful updates to the weights of deep CNN layers and improves low-level feature learning, while the effects of randomization can be easily converged throughout the overall learning procedure. The experiments show that the use of **RIFLE** significantly improves deep transfer learning accuracy on a wide range of datasets, outperforming known tricks for the similar purpose, such as Dropout, DropConnect, Stochastic Depth, Disturb Label and Cyclic Learning Rate, under the same settings with 0.5%–2% higher testing accuracy. Empirical cases and ablation studies further indicate **RIFLE** brings meaningful updates to deep CNN layers with accuracy improved.

---

[*]Equal contribution [1]Big Data Lab, Baidu Research, Beijing, China [2]Faculty of Science and Technology, University of Macau, Macau SAR, China [3]State Key Lab of IOTSC, Department of Computer Science, University of Macau, Macau SAR, China. Correspondence to: Xingjian Li, Haoyi Xiong <{lixingjian,xionghaoyi}@baidu.com>.

1. Introduction

Blessed by tons of labeled datasets that are open to the public, deep learning (LeCun et al., 2015) has shown enormous success for image classification and object detection tasks in the past few years. In addition to training a deep convolution neural network (CNN) from scratch with empty or random initialization, a more effective way might be fine-tuning a deep CNN using the weights of a pre–trained model, e.g., ResNet-50 trained using ImageNet, as the starting point (Kornblith et al., 2019), since the learning dynamics would converge to a local minima of loss that is close to its starting point (Neyshabur et al., 2015; Soudry et al., 2018; Liu et al., 2019). To the end, such deep transfer learning paradigms reuse convolutional filters of well-trained models (Yosinski et al., 2014; Long et al., 2015), and are capable of extracting useful features from the target dataset for better classification after a short fine-tuning procedure.

While above fine-tuning practice greatly enhances the accuracy of image classification even when the training dataset is small (Bengio, 2012; Kornblith et al., 2019), the room for further enhancement still exists. Due to the use of well-trained filters, the fine-tuning procedure frequently converges very quickly, while most of CNN weights have not been updated sufficiently. Although the fully-connected (FC) layer would be fine tuned with superior performance to classify the training datasets (Zhang et al., 2017), the features learned for classifications might still be largely based on the source datasets of transfer learning rather than the target one (Yosinski et al., 2014; Long et al., 2015). From the error backpropagation's perspectives (Rumelhart et al.; LeCun et al., 1988), the fast convergence of the FC layer leads to trivial weight updates to CNN (Liu et al., 2019), while the updates would become smaller and smaller with the depth of CNN layers (Srivastava et al., 2015). In such way, compared to the CNN weights learned from the target training dataset directly, the fine-tuning procedure seldom changes or improves the deep CNN layers (Liu et al., 2019). Our research hereby assumes that the in-depth backpropagation could further improve the performance of transfer learning based on pre–trained models.

To achieve the goal, tons of tricks, including Dropout (Gal & Ghahramani, 2016), Dropconnect (Wan et al., 2013), Stochastic depth (Huang et al., 2016), Cyclic learning rate (Smith, 2017), Disturb label (Xie et al., 2016) and etc., have been proposed in supervised learning settings, especially to train very deep neural networks (Srivastava et al., 2015). Our research, however, finds these efforts still might not work in-depth for transfer learning, as the FC layer converges so fast and overfits to the target training dataset with features extracted by the pre–trained CNN filers (Li et al., 2019). We thus aim at proposing a novel backpropagation strategy that appropriately updates deep CNN layers, in transfer learning settings, against the fast convergence of FC.

In this way, we propose a simple yet effective strategy, namely **RIFLE**, which makes backpropagation in-depth for deep transfer learning through *Re-Initializing the Fully-connected LayEr*. More specifically, we equally divide the whole fine-tuning epochs into several periods, and re-initialize weights of the fully connected layer with random weights at the beginning of each period. Furthermore, cyclic learning rate in the same pace has been used to update the FC layer to ensure the final convergence. To verify the performance of **RIFLE**, we carry out extensive experiments based on various popular image classification datasets with diverse types of visual objects. **RIFLE** stably improve the fine-tuning procedures and outperforms the aforementioned strategies under the same settings. In summary, our contributions are as follows:

- This work is the first attempt to improve deep transfer learning through pushing meaningful updates to the deep CNN layers, through the error backpropagation with the target training datasets. In a micro perspective, **RIFLE** re-initializes the FC layer with random weights and enlarges the fitting error for deeper backpropagation. From a macro perspective, **RIFLE** periodically incorporates non-trivial perturbation to the fine-tuning procedures and helps training procedure to escape the "attraction" of pre–trained CNN weights (Zhou et al., 2019). Furthermore, **RIFLE** is designed to randomly re-initialized the FC layer only and might not hurt the convergence of overall CNN.

- The existing regularizers for deep transfer learning, such as L^2 (weight decay) (Bengio, 2012), L^2-SP (Li et al., 2018)), and DELTA (Li et al., 2019), intend to improve the knowledge transfer through ensuring the consistency between the pre–trained model and the target one from weights and/or parameter perspectives. Our research, however, assumes such regularization reduces the effect of error backpropagation and might bring less meaningful modifications to the network, especially the deep CNN layers. On the other

hand, when such regularization for knowledge transfer is disabled or weakened, the deep transfer learning algorithm might overfit to the training dataset with poor generalization performance. To balance the these two issues, **RIFLE** is proposed. **RIFLE** avoids overfitting through incorporating additional noise by random re-initialization while improving the knowledge transfer through backpropagation in-depth.

- Through extensive experiments covering various transfer learning datasets for image classification, we demonstrate that, on top of two deep transfer learning paradigms (i.e., the vanilla fine-tuning and the one with L^2-SP (Li et al., 2018)), **RIFLE** could achieve stable accuracy improvement with 0.5%-2% higher testing accuracy. It also outperforms existing strategies, including Dropout (Gal & Ghahramani, 2016) (on both CNN layers and FC layers), Dropconnect (Wan et al., 2013), Stochastic depth (Huang et al., 2016), Cyclic learning rate (Smith, 2017) and Disturb label (Xie et al., 2016), which have been proposed for similar purposes, under the same settings. We also conducted extensive empirical studies and an ablation study to show that (1) **RIFLE** can make backpropagation in-depth and bring significant modifications to the weights of deeper layer; and (2) the weight modification brought by **RIFLE** benefits to the generalization performance through enhancing the capacity of lower-level feature learning.

In addition to above contributions, to the best of our knowledge, **RIFLE** is also yet the first algorithmic regularization (Allen-Zhu et al., 2019) designed for deep transfer learning purposes through weights randomization. **RIFLE** can work together with an explicit deep transfer learning regularizer, such as (Li et al., 2018), and makes a complementary contribution by further improving the generalization performance.

2. Related Work

In this section, we first introduce the related work of deep transfer learning with the most relevant work to our study.

2.1. Deep Transfer Learning in General

Transfer learning refers to a type of machine learning paradigms that aim at transferring knowledge obtained in the source task to a (maybe irrelevant) target task (Pan et al., 2010; Caruana, 1997), where the source and target tasks can share either the same or different label spaces. In our research, we primarily consider the inductive transfer learning with a different target label space for deep neural networks. As early as in 2014, authors in (Donahue et al., 2014) reported their observation of significant performance

improvement through directly reusing weights of the pre-trained source network to the target task, when training a large CNN with a tremendous number of filters and parameters.

However, in the interim, while reusing all pre-trained weights, the target network might be overloaded by learning tons of inappropriate features (that cannot be used for classification in the target task), while the key features of the target task have been probably ignored. In this way, Yosinki *et al.* (Yosinski et al., 2014) proposed to understand whether a feature can be transferred to the target network, through quantifying the "transferability" of features from each layer considering the performance gain. Furthermore, Huh *et al.* (Huh et al., 2016) made an empirical study on analyzing features that CNN learned from the ImageNet dataset to other computer vision tasks, so as to detail the factors affecting deep transfer learning accuracy. (Ge & Yu, 2017) use a framework of multi-task learning by incorporating examples from the source domain which are similar to examples of the target task. In recent days, this line of research has been further developed with an increasing number of algorithms and tools that can improve the performance of deep transfer learning, including subset selection (Ge & Yu, 2017; Cui et al., 2018), sparse transfer (Liu et al., 2017), filter distribution constraining (Aygun et al., 2017), and parameter transfer (Zhang et al., 2018).

2.2. Regularization for Deep Transfer Learning

Here, we review the knowledge transfer techniques that reuse pre-trained weights through the regularization. The square Euclidean distance between the weights of source and target networks is frequently used as the regularizer for deep transfer learning (Li et al., 2018). Specifically, (Li et al., 2018) studied how to accelerate deep transfer learning while preventing fine-tuning from over-fitting, using a simple L^2-norm regularization on top of the "Starting Point as a Reference" optimization. Such method, namely L^2-SP, can significantly outperform a wide range of deep transfer learning algorithms, such as the standard L^2-norm regularization. Yet another way to regularize the deep transfer learning is "knowledge distillation" (Hinton et al., 2015; Romero et al., 2014). In terms of methodologies, the knowledge distillation was originally proposed to compress deep neural networks (Hinton et al., 2015; Romero et al., 2014) through teacher-student network training, where the teacher and student networks are usually based on the same task (Hinton et al., 2015). In terms of inductive transfer learning, authors in (Yim et al., 2017) were first to investigate the possibility of using the distance of intermediate results (e.g., feature maps generated by the same layers) of source and target networks as the regularization term. Further, (Zagoruyko & Komodakis, 2016) proposed to use the regularization of the divergence between activation maps

for "attention transfer".

In addition to the above explicit regularization, the implicit regularization or algorithmic regularization has been frequently discussed in deep learning settings (Soudry et al., 2018). Generating and controlling the noise in the stochastic learning process, such as Dropout, cyclic learning rate and so on (Gal & Ghahramani, 2016; Huang et al., 2016; Smith, 2017), has been considered as a way to improve deep learning under the over-parameterization through implicit/algorithmic regularization. However, the performance improvement caused by these implicit/algorithmic regularizers has not been discussed in the context of deep transfer learning.

2.3. Connection to Our Work

Compared to above work and other transfer learning studies, our work aims at providing a *in-depth backpropagation* strategy that improves the performance of deep transfer learning. The intuition of **RIFLE** is to periodically re-initialize the FC layer of CNN with random weights while ensuring the final convergence of the overall fine-tuning procedure with cyclical learning rate (Smith, 2017) applied on the FC layer. In our work, we demonstrated the capacity of **RIFLE** working with two deep transfer learning paradigms — i.e., vanilla fine-tuning and the one with L^2-SP (Li et al., 2018) regularization, using a wide range of transfer learning tasks. The performance boosts with **RIFLE** in all cases of experiments suggests that **RIFLE** improves deep transfer learning with higher accuracy.

In terms of methodologies, the most relevant studies to our work are (Hinton et al., 2012; Wan et al., 2013; Huang et al., 2016; Smith, 2017), where all these methods intend to deepen the backpropagation through incorporating perturbation in the supervised learning procedure. More specific, Dropout (Hinton et al., 2012) randomly omits parts of neurons of input or hidden layers during learning to regularize the backpropagation. Furthermore, DropConnect (Wan et al., 2013) generalizes the idea of Dropout by randomly dropping some connections between neurons. Stochastic depth (Huang et al., 2016) incorporates layer-wise random shortcuts to train short networks with sufficient backpropagation while inferring over the complete one. Disturb label (Xie et al., 2016) randomly perturbs labels of the training batch in certain proportions and involves randomness in the loss layer. Furthermore, the use of cyclical learning rates (Smith, 2017) incorporates randomness over the training procedure to escape shallow and sharp local minima while achieving decent convergence. Our methodology follows this line of research, where periodical random re-initialization of the FC layer makes backpropagation in depth while ensuring convergence of the overall fine-tuning procedure.

3. Deep Transfer Learning with RIFLE

In this section, we first introduced the common algorithms used for deep transfer learning, then present the design of **RIFLE** based on randomized regularization.

3.1. Deep Transfer Learning with Regularization

The general deep transfer learning problem based on the pre–trained model is usually formulated as follow.

Definition 1 (Deep Transfer Learning) *First of all, let's denote the training dataset for the desired task as* $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3) \ldots, (\mathbf{x}_n, y_n)\}$, *where totally* n *tuples are offered and each tuple* $(\mathbf{x}_i, y_i)$ *refers to the input image and its label in the dataset. We then denote* $\omega \in \mathbb{R}^d$ *be the* d-*dimensional parameter vector containing all* d *parameters of the target model. Further, given a pretrained network with parameter* $\omega_\mathbf{s}$ *based on an extremely large dataset as the source, one can estimate the parameter of target network through the transfer learning paradigms. The optimization object with based deep transfer learning is to obtain the minimizer of* $\mathcal{L}(\omega)$

$$\min_w \mathcal{L}(\omega) = \left\{ \frac{1}{n} \sum_{i=1}^{n} L(z(\mathbf{x}_i, \omega), y_i) + \lambda \cdot \Omega(\omega, \omega_s) \right\} \tag{1}$$

where (i) the first term $\sum_{i=1}^{n} L(z(\mathbf{x}_i, \omega), y_i)$ *refers to the empirical loss of data fitting while (ii) the second term* $\Omega(\omega, \omega_\mathbf{s})$ *characterizes the differences between the parameters of target and source network. The tuning parameter* $\lambda > 0$ *balances the trade-off between the empirical loss and the regularization term.*

As was mentioned, two common deep transfer learning algorithms studied in this paper are vanilla fine-tuning (Kornblith et al., 2019) and L^2-SP (Li et al., 2018). Specifically, these two algorithms can be implemented with the general based deep transfer learning with different regularizers

3.1.1. REGULARIZATION FOR FINE-TUNING

The vanilla fine-tuning procedure incorporates a simple L^2-norm regularization (weight decay) of the weights to ensure the sparsity of weights, such that

$$\Omega(\omega, \omega_s) = \|\omega\|_2^2. \tag{2}$$

Note that such L^2 regularization is indeed independent with ω_s. Fine-tuning only adopts the weights of the pre-trained model ω_s as the starting point of optimization, so as to transfer knowledge learned from source datasets.

3.1.2. REGULARIZATION OF L2-SP

In terms of the regularizer, this algorithm(Li et al., 2018) uses the squared-Euclidean distance between the target

weights (i.e., optimization objective ω) and the pre-trained weights $\omega_\mathbf{s}$ of source network (listed in Eq 3) to constrain the learning procedure.

$$\Omega(\omega, \omega_s) = \|\omega - \omega_s\|_2^2 \tag{3}$$

In terms of optimization procedure, L^2-SP makes the learning procedure start from the pre-trained weights (i.e., using $\omega_\mathbf{s}$ to initialize the learning procedure).

In the rest of this work, we presented a strategy **RIFLE** to improve the general form of deep transfer learning shown in Eq. 1, then evaluated and compared **RIFLE** using the above two regularizers with common deep transfer learning benchmarks.

Algorithm 1 RIFLE with SGD Optimizer
1: **procedure RIFLE**($\mathcal{L}, \mathbf{D}, \omega_\mathbf{s}, P, T, \eta_{\max}$)
2:    **for** $t = 1, 2, 3 \ldots, T$ **do**
3:        $\tau \leftarrow t \bmod P$
4:        **if** $\tau = 0$ **then**
5:            /*Randomized Re-Initialization of FC*/
6:            $\Omega_{\text{FC}} \sim \mathcal{N}(0, \delta^2 I)$
7:            $\Omega_t' \leftarrow \text{resetFC}(\Omega_t, \Omega_{\text{FC}})$
8:        **else**
9:            $\Omega_t' \leftarrow \Omega_t$
10:        **end if**
11:        /*Cyclical Learning Rates*/
12:        $\eta_t \leftarrow \frac{1}{2} \cdot \eta_{\max} \cdot \text{cosine}(2\pi \cdot \tau / P) + \frac{1}{2} \cdot \eta_{\max}$
13:        $\mathbf{B}_t \leftarrow$ **mini-batch sampling from D**
14:        /*Backpropagation with SGD Optimizer*/
15:        $\Omega_{t+1} \leftarrow \Omega_t - \eta_t \cdot \frac{1}{|\mathbf{B}_t|} \sum_{(x,y) \in \mathbf{B}_t} \nabla \mathcal{L}_{(x,y)}(\Omega_t')$
16:    **end for**
17:    **return** Ω_T
18: **end procedure**

3.2. RIFLE Algorithm

Given the regularized loss function $\mathcal{L}(\omega)$, weights of the pre-trained model Ω_s, the target training dataset $\mathbf{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, the number of iterations P for the period of random re-initialization, the overall number of iterations for training T, and the maximal learning rate $\eta_{\max}$, we propose to use Algorithm 1 to train a deep neural network with **RIFLE** with SGD Optimizer. Note that the optimizers (line 15) used for backpropagation are interchangeable with Adam, Momentum and etc., all based on the same settings of **RIFLE**. The operator resetFC($\Omega_t, \Omega_{\text{FC}}$) in line 7 refers to the operation that replaces the weights of the fully-connected layer in Ω_t with the random weights Ω_{FC}.

Specifically, for each (e.g., the tth) iteration of learning procedure, **RIFLE** first checks whether a new period for random re-initialization of FC layer starts (i.e., whether

$t \bmod P = 0$). When a new period starts, **RIFLE** draws a new random vector from the Gaussian distribution $\mathcal{N}(0, \delta^2 I)$ as the weights of randomized FC layer Ω_{FC}, then reset the weights of the FC layer in the current model Ω_t with Ω_{FC} to obtain Ω_t'. Please refer to lines 4–8 for details. Later, **RIFLE** adopts the simple cosine curve to adapt the cyclical learning rates (Smith, 2017) then performs backpropagation based on Ω_t' and η_t accordingly. Note that in Algorithm. 1, we present the design to incorporate the SGD optimizer for training. It would be appropriate to use other optimizers such Adam, Momentum and etc. that also incorporate Ω_t' and η_t as inputs.

3.2.1. DISCUSSION

Note that **RIFLE** strategy is derived from the stochastic gradient estimator used in stochastic gradient based learning algorithms, such as SGD, Momentum, conditioned SGD, Adam and so on. We consider **RIFLE** as an alternative approach for descent direction estimation, where one can use a natural gradient-alike method to condition the descent direction or to adopt Momentum-alike acceleration methods on top of **RIFLE**. We are not intending to compare **RIFLE** with any gradient-based learning algorithms, as the contributions are complementary. One can freely use **RIFLE** to improve any gradient-based optimization algorithms (if applicable) with new descend directions.

4. Experiment

In this section, we present the experiments to evaluate **RIFLE** with comparison to the existing methods.

4.1. Datasets

Eight popular transfer learning datasets are used to evaluate the effect of our algorithm.

Stanford Dogs. The Stanford Dogs dataset is used for the task of fine-grained image classification, containing images of 120 breeds of dogs. Each category contains 100 training examples. It provides bounding box annotations for the further purpose of vision specific algorithms. We use only classification labels during training.

Indoors 67. Indoors 67 is a scene classification task containing 67 indoor scene categories, each of which consists of 80 images for training and 20 for testing. A major property unique to object recognition tasks is that, both spatial properties and object characters are expected to be extracted to obtain discriminative features.

Caltech-UCSD Birds-200-2011. CUB-200-2011 contains 11,788 images of 200 bird species from around the world. Each species is associated with a Wikipedia article and is organized by scientific classification. This task is challenging because birds are small and they exhibit pose variations. Each image is annotated with bounding box, part location, and attribute labels though we do not use this information.

Food-101. Food-101 is a large scale data set consisting of 101 different kinds of foods. It contains more than 100k images. Each category contains 750 training examples in total. In this paper, we select two subsets for transfer learning evaluation, each containing 30 and 150 training examples per category, named Food-101-30 and Food-101-150 respectively.

Flower-102. Flower-102 consists of 102 flower categories. 1020 images are used for training, 1020 for validation, and 6149 images for testing. Compared with other datasets, it is relative small since only 10 samples are provided for each class during training.

Stanford Cars. The Stanford Cars dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split. Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe.

FGVC-Aircraft. FGVC-Aircraft is composed of 102 different types of aircraft models, each of which contains 37 images for training and 37 for testing on average. It is a benchmark dataset for the fine grained visual categorization tasks. Bounding box annotations are not used for evaluation.

Describable Textures Dataset. Describable Textures Dataset (DTD) is a texture database, consisting of 5640 images. They are organized by a list of 47 categories, according to different perceptual properties of textures.

4.2. Settings

All experiments are implemented on the popular ResNet-50 architecture pre-trained with the ImageNet classification task. We use a batch size of 32. SGD with the momentum of 0.9 is used for optimizing all models. The initial learning rate is set to 0.01, using a cosine annealing learning rate policy. We train all models for 40 epochs. Normal data augmentation operations of random mirror and random crop are used for better performance in all experiments. Specifically, we first resize input images with the shorter edge being 256, in order to keep the original aspect ratio, following with normal data augmentation operations of random mirror and random crop to 224*224. Then inputs are normalized to zero mean for each channel.

We compare several important regularization algorithms in transfer learning scenarios. *1)* Dropout and DropConnect on fully connected layers. For each method, we test the drop probability of 0.1, 0.2, 0.5 and report the best one.

Table 1. Comparison of regularization methods. "Original" refers to the performance of using the standard fine-tuning algorithm. "Drop-Conn." refers to DropConnection. "CNN-DrO." refers to Dropout performed on CNN layers. "StochasticD." refers to Stochastic Depth. "CycleLR." refers to Cyclic Learning Rate. "DisturbL." refers to Disturb Label.

		Algorithm regularizers + L^2 (Bengio, 2012)						
Datasets	Original	Dropout	DropConn.	CNN-DrO.	StochasticD.	CyclicLR.	DisturbL.	**RIFLE**
CUB-200-2011	80.45±0.35	80.43±0.38	79.82±0.24	65.08±0.19	66.48±0.70	80.06±0.24	80.61±0.29	**81.13±0.23**
FGVC-Aircraft	76.58±0.36	76.52±0.58	75.76±0.71	74.83±0.37	60.94±0.74	75.63±0.43	76.65±0.33	**77.82±0.41**
Flower-102	92.53±0.31	92.82±0.21	91.99±0.36	89.36±0.26	67.30±1.28	92.33±0.26	91.98±0.41	**93.00±0.08**
DTD	64.08±0.44	64.09±0.70	62.62±0.88	63.24±0.83	56.33±0.85	63.64±0.43	**66.71±0.33**	64.94±0.37
Indoor-67	75.09±0.39	74.63±0.59	74.30±0.65	73.15±0.26	60.36±2.11	74.48±0.53	74.60±0.62	**76.15±0.52**
Stanford Cars	89.12±0.13	89.34±0.07	88.82±0.33	88.20±0.07	75.40±0.54	88.97±0.14	89.52±0.08	**90.08±0.14**
Stanford Dogs	79.21±0.18	79.24±0.15	78.92±0.32	78.07±0.26	67.97±1.04	78.74±0.15	79.76±0.10	**80.10±0.23**
Food-101-30	60.38±0.15	60.09±0.23	58.85±0.28	57.41±0.32	42.40±0.79	60.57±0.21	58.02±0.15	**62.31±0.27**
Food-101-150	75.81±0.14	75.71±0.19	75.45±0.15	74.53±0.29	64.11±0.33	75.31±0.05	75.35±0.08	**76.36±0.16**
		Algorithmic regularizers + L^2-SP (Li et al., 2018)						
Datasets	Original	Dropout	DropConn.	CNN-DrO.	StochasticD.	CyclicLR.	DisturbL.	**RIFLE**
CUB-200-2011	80.65±0.10	81.17±0.28	80.35±0.27	65.41±0.40	66.47±0.90	80.74±0.13	80.71±0.08	**81.73±0.20**
FGVC-Aircraft	76.31±0.42	76.44±0.32	75.79±0.47	74.26±0.21	60.31±1.13	76.08±0.37	77.04±0.32	**78.00±0.58**
Flower-102	91.98±0.34	92.10±0.25	91.86±0.32	89.09±0.43	68.95±0.93	91.91±0.16	91.79±0.48	**92.46±0.18**
DTD	69.41±0.47	69.21±0.25	68.92±0.45	63.40±0.62	54.73±1.01	68.90±0.38	66.61±0.50	**70.71±0.36**
Indoor-67	75.99±0.33	75.38±0.57	75.44±0.68	73.00±0.52	62.23±0.65	75.21±0.71	74.63±0.19	**77.11±0.44**
Stanford Cars	89.12±0.21	89.48±0.26	88.91±0.09	88.43±0.15	75.45±0.99	89.15±0.15	89.50±0.11	**90.00±0.14**
Stanford Dogs	88.32±0.14	88.73±0.14	87.93±0.14	78.47±0.16	67.94±0.65	88.07±0.13	79.96±0.21	**88.96±0.06**
Food-101-30	61.25±0.23	61.01±0.24	59.66±0.54	57.73±0.27	42.10±1.15	61.06±0.15	57.98±0.32	**62.84±0.25**
Food-101-150	77.02±0.15	77.04±0.16	76.62±0.18	73.98±0.15	65.28±0.61	76.52±0.11	75.35±0.17	**77.36±0.04**

2) Dropout on convolutional layers. We also test typical choices of 0.1, 0.2, 0.5 and report the best one. *3)* Stochastic Depth. We use linear decay of the probability of skipping with stochastic depth for $p_0 = 1$ and $p_L = 0.5$, following the best setting of (Huang et al., 2016). Conceptually the first block which is the closest to the input is always active, and the last block which is the closest to the output is skipped with a probability of 0.5. *4)* Cyclic learning rate. We divide the training procedure into N cycles. We test choices of 2, 3, 4 and report the best one. *5)* Disturb label. We test the perturbation probability of 0.05, 0.1, 0.2 recommended by (Xie et al., 2016) and report the best result. We run each experiment five times and report the average top-1 accuracy.

4.3. Overall Comparisons

We test our deep transfer learning algorithm with two common fine-tuning algorithms (regularization), which are standard fine-tuning (L^2) and using the starting point as reference (L^2-SP) in all cases. Through exhaustive experiments, we observe that:

- **Comparison to the explicit regularizers for transfer learning.** **RIFLE** stably and significantly improves the performances of two common fine-tuning algorithms, including L^2 and L^2-SP, under all the cases. It demonstrates the feasibility of using algorithmic regularization **RIFLE** to improve deep transfer learning on top of existing explicit regularizers.

- **Comparison to the algorithmic+explicit regularizers.** **RIFLE** outperforms the other state-of-the-art algorithms with similar purposes. These algorithms includes Dropout (Gal & Ghahramani, 2016), Drop Connect (Wan et al., 2013), Stochastic Depth (Huang et al., 2016), Cyclic Learning Rate (Smith, 2017) and Disturb Label (Xie et al., 2016). For both two fine-tuning algorithms, **RIFLE** shows an improvement over all the existing methods on most datasets. We notice that CNN-Dropout and Stochastic Depth show poor performances on two fine-tuning algorithms. Among the compared algorithms, Dropout and Disturb label show relatively better performances, while performing less well than **RIFLE** under most tasks.

We can readily conclude that when **RIFLE**+L^2 and **RIFLE**+L^2-SP significantly outperforms the vanilla explicit regularizers for deep transfer learning, while the simple combination of L^2 or L^2-SP with other algorithmic regularizers, such as Dropout, Drop-connection and so on, could not perform as well as **RIFLE** for deep transfer learning. The comparison demonstrates the advancement achieved by **RIFLE**.

4.4. Ablation Study

In order to investigate which is the most critical part in our algorithm, we do ablation study to show how each component affects the accuracy. First we define two modified versions of **RIFLE**, named **RIFLE**-A and **RIFLE**-B respec-

Table 2. Comparison for Ablation Studies.

	L^2+**RIFLE** and variants		
Datasets	**RIFLE**	**RIFLE-A**	**RIFLE-B**
CUB-200-2011	81.13±0.23	81.07±0.29	79.84±0.28
FGVC-Aircraft	77.82±0.41	77.70±0.45	75.95±0.50
Flower-102	93.00±0.08	92.17±0.19	92.44±0.16
DTD	64.94±0.37	65.12±0.53	64.11±0.84
Indoor-67	76.15±0.52	75.94±0.11	74.02±0.38
Stanford Cars	90.08±0.14	89.88±0.20	88.65±0.19
Stanford Dogs	80.10±0.23	80.67±0.10	78.79±0.20
Food-101-30	62.31±0.27	61.91±0.25	60.30±0.22
Food-101-150	76.36±0.16	76.66±0.11	75.59±0.12
	L^2-SP+**RIFLE** and variants		
Datasets	**RIFLE**	**RIFLE-A**	**RIFLE-B**
CUB-200-2011	81.73±0.20	81.73±0.18	80.99±0.31
FGVC-Aircraft	78.00±0.58	77.91±0.38	75.89±0.49
Flower-102	92.46±0.18	91.68±0.28	91.53±0.21
DTD	70.71±0.36	70.22±0.44	68.84±0.52
Indoor-67	77.11±0.44	76.71±0.33	75.44±0.59
Stanford Cars	90.00±0.14	89.82±0.16	89.08±0.12
Stanford Dogs	88.96±0.06	88.90±0.11	88.00±0.07
Food-101-30	62.84±0.25	62.48±0.09	61.19±0.24
Food-101-150	77.36±0.04	77.96±0.08	76.44±0.11

(a) L^2 (Train)

(b) L^2 (Test)

(c) L^2+CyclicLR (Train)

(d) L^2+CyclicLR (Test)

(e) L^2+**RIFLE**-A (Train)

(f) L^2+**RIFLE**-A (Test)

Figure 1. Comparisons of Learning Curves.

tively. For **RIFLE**-A, we remove the cyclic learning rate component from our **RIFLE** implementation and uses random re-initialization only. Therefore, the influence of different learning rates is cleared away in **RIFLE**-A. For **RIFLE**-B, we remove the re-initialization component while using cyclic learning rates only. Thus, the only difference of **RIFLE**-B compared with vanilla L^2 is the cyclic learning rate applied on the FC layer. We evaluate **RIFLE**-A and **RIFLE**-B on all above tasks with both L^2 and L^2-SP regularization. As observed in Table 2, **RIFLE**-A performs marginally worse than **RIFLE** on most tasks, and is even superior to **RIFLE** on particular datasets such as Food-101-150. **RIFLE**-A outperforms than **RIFLE**-Band vanilla L^2 and L^2-SP. We can conclude that re-initializing the FC layer rather than cyclic learning rate is the founding brick of our algorithm.

4.5. Comparisons of Learning Curves

We further analyze the effect of **RIFLE** by analysis of the learning curves, through the comparison with vanilla L^2 algorithm and the one based on L^2 with cyclic learning rate. As showed in Figure 1a, most tasks completely fit the training set after about half of the total number of epochs. For L^2 with cyclic learning rate, we find extremely sharp drops of the accuracy at the beginning of each period, due to the learning rate resetting. We notice in Figure 1c that, the magnitude of the decline diminishes as the number of training epochs increases. While for test set in Figure 1d, the accuracy always decreases severely at the first few epochs after learning rate resetting for most tasks. The training curve of **RIFLE**-A in Figure 1e is similar to
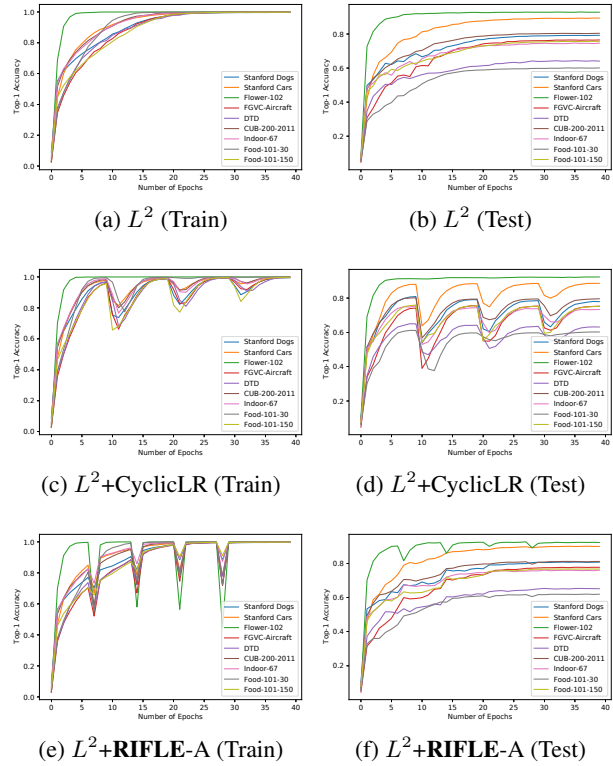
the cyclic learning rate. However, compared to training curves in Figure 1c, the magnitude of the accuracy decline doesn't diminish in later training epochs for the one with cyclic learning rates. On the test set, **RIFLE**-A obtains much smoother accuracy curves compared to the one with the cyclic learning rate, which implies more stable generalization capacity. We infer that it is because the feature extractor part is perturbed by meaningful backpropagation brought by the re-initialized FC layer.

We can conclude that, compared to cyclic learning rate or vanilla training procedure for deep transfer learning, **RIFLE** brings more modifications to the weights during the learning procedure as the training curve is more unstable. Furthermore, such modifications could improve the generalization performance of deep transfer learning, as the testing curve dominates in the comparison.

4.6. Empirical Studies

In this section, we intend to testify (1) Whether random re-initialization of the FC layer makes the backpropagation in-depth and updates the weights of deeper layers under transfer learning settings; (2) In which way, the random re-initialization of FC layer would improve the generalization performance of deep transfer learning.
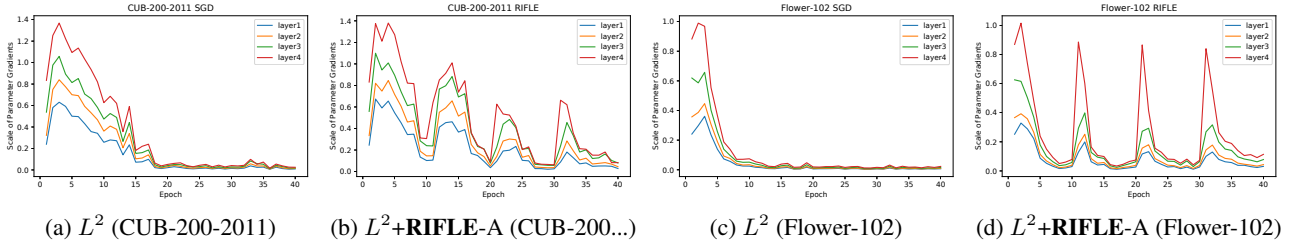
Figure 2. Scale of gradients corresponding to parameters of different layers.

4.6.1. **RIFLE** MAKES BACKPROPAGATION DEEPER

Our study finds **RIFLE** makes backpropagation deeper and brings more modifications ton deeper layers. In order to observe the phenomena, we measure the scale of gradients corresponding to different CNN layers during fine-tuning. To avoid the effects of cyclic learning rate, we perform the experiments using **RIFLE**-A with 4 times of re-initialization. We select the last 3×3 convolutional module in each of the four layers in ResNet-50. The Frobenius norm of the gradient of each parameter is calculated at the beginning of each training epoch. As illustrated in Figure 2, we can observe that scales of gradients in vanilla L^2 decrease rapidly during training. For example, gradients of the CUB-200-2011 task almost vanish after 20 epochs when training with vanilla L^2 as showed in Figure 2a. Flower-102 is even easier to convergence since gradients vanish after 10 epochs as showed in Figure 2c. While for our proposed **RIFLE**-A, gradients are periodicity reactivated as demonstrated in Figure 2b and 2d, even though we use the exact same learning rate. We notice that parameters of layers close to the input (e.g. layer1) have smaller scales of gradients, which is consistent between vanilla L^2 and **RIFLE**-A.

4.6.2. **RIFLE** LEARNS BETTER LOW-LEVEL FEATURES

Our study finds the low level features learned by **RIFLE** are better than the vanilla deep transfer learning, through an empirical asymptotic analysis as follows.

MLP Oracles. We proposed to use two MLPs as the oracles to generate training datasets for source and target tasks respectively. The MLPs are both with two layers and with ReLU as activation. To make sure the knowledge transfer between two tasks, we make these two MLPs share the same first layer, such that $h_1(X) = W_2^\top \mathrm{ReLU}(W_1^\top X)$ and $h_2(X) = W_3^\top \mathrm{ReLU}(W_1^\top X)$. The input of networks X is an 100-dimension vector (i.e., X is a 100×1 matrix), while the dimension of the hidden layer is 50 (i.e., W_1 is a 100×50 random matrix and W_2 and W_3 are two 50×1 random matrices).

Source/Target Tasks. We collect the datasets for source and target tasks using $h_1(X)$ and $h_2(X)$ respectively, each

of which is with 1000 samples drawn from standard Gaussian distributions. With X_i randomly drawn from a 200-dimension Gaussian distribution as a training sample, we obtain $h_1(X_i) + \mathcal{N}(0, 0.01)$ as the label of x_i for the source task. With a similar approach, we also collect the datasets for the target task using $h_2(X)$.

Transfer Learning. We first use the generated source dataset to train an MLP with the same architecture from scratch through vanilla SGD via regression loss, and we obtain the teacher network model $h_1'(X) = W_2'^\top \mathrm{ReLU}(W_1'^\top X)$. Note that the weights W_2' and W_1' are so different from W_2 and W_1 as multiple solutions exist. With the weights W_2' and W_1' and the generated target dataset, we use L^2 and L^2+**RIFLE** for transfer learning to obtain two networks $h_2'(X) = W_3'^\top \mathrm{ReLU}(W_1'^\top X)$ and $h_2^*(X) = W_3^{*\top} \mathrm{ReLU}(W_1^{*\top} X)$ respectively.

Comparisons. We find both $h_2'(X)$ and $h_2^*(X)$ achieves very good accuracy given the testing dataset generated by $h_2(X)$, while the testing loss of $h_2^*(X)$ (with MSE 3.98×10^{-3}) is significantly lower than $h_2'(X)$ (with MSE 1.16×10^{-2}). Furthermore, we also observe that $\mathrm{OT}(W_1^*, W_1) = 0.1198 \leq \mathrm{OT}(W_1', W_1) = 0.1397$, where $\mathrm{OT}(\cdot, \cdot)$ refers to the optimal transport distance (Flamary & Courty, 2017) between the inputs. As the weights of deep layers are closer to the MLP oracles, we conclude that **RIFLE** learns better low-level features.

5. Conclusion

In this work, we propose **RIFLE**– a simple yet effective strategy that enables in-depth backpropagation in transfer learning settings, through periodically <u>R</u>e-<u>I</u>nitializing the <u>F</u>ully-connected <u>LayE</u>r with random scratch during the fine-tuning procedure. **RIFLE** brings significant meaningful modifications to the deep layers of DNN and improves the low-level feature learning. The experiments show that the use of **RIFLE** significantly improves the accuracy of deep transfer learning algorithms (based on explicit regularization) on a wide range of datasets. It outperforms known algorithmic regularization tricks such as dropout and so on, under the same settings. To the best of our knowledge, **RIFLE** is yet the first algorithmic regulariza-

tion method to improve the deep transfer learning algorithms based on explicit regularization such as (Li et al., 2018). The contribution made in this work is complementary with the previous work.

6. Acknowledgements

References

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pp. 6155–6166, 2019.

Aygun, M., Aytar, Y., and Ekenel, H. K. Exploiting convolution filter patterns for transfer learning. In *ICCV Workshops*, pp. 2674–2680, 2017.

Bengio, Y. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 17–36, 2012.

Caruana, R. Multitask learning. *Machine learning*, 28(1): 41–75, 1997.

Cui, Y., Song, Y., Sun, C., Howard, A., and Belongie, S. Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4109–4118, 2018.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pp. 647–655, 2014.

Flamary, R. and Courty, N. Pot python optimal transport library, 2017. URL https://github.com/rflamary/POT.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Ge, W. and Yu, Y. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 10–19, 2017.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *European conference on computer vision*, pp. 646–661. Springer, 2016.

Huh, M., Agrawal, P., and Efros, A. A. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

Kornblith, S., Shlens, J., and Le, Q. V. Do better imagenet models transfer better? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2661–2671, 2019.

LeCun, Y., Touresky, D., Hinton, G., and Sejnowski, T. A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school*, volume 1, pp. 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Li, X., Grandvalet, Y., and Davoine, F. Explicit inductive bias for transfer learning with convolutional networks. *arXiv preprint arXiv:1802.01483*, 2018.

Li, X., Xiong, H., Wang, H., Rao, Y., Liu, L., and Huan, J. Delta: Deep learning transfer using feature map with attention for convolutional networks. *arXiv preprint arXiv:1901.09229*, 2019.

Liu, H., Long, M., Wang, J., and Jordan, M. I. Towards understanding the transferability of deep representations. *arXiv preprint arXiv:1909.12031*, 2019.

Liu, J., Wang, Y., and Qiao, Y. Sparse deep transfer learning for convolutional neural network. In *AAAI*, pp. 2245–2251, 2017.

Long, M., Cao, Y., Wang, J., and Jordan, M. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pp. 97–105, 2015.

Neyshabur, B., Salakhutdinov, R. R., and Srebro, N. Path-sgd: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2422–2430, 2015.

Pan, S. J., Yang, Q., et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Smith, L. N. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472. IEEE, 2017.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Training very deep networks. In *Advances in neural information processing systems*, pp. 2377–2385, 2015.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066, 2013.

Xie, L., Wang, J., Wei, Z., Wang, M., and Tian, Q. Disturblabel: Regularizing cnn on the loss layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4753–4762, 2016.

Yim, J., Joo, D., Bae, J., and Kim, J. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

Zagoruyko, S. and Komodakis, N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

Zhang, C.-L., Luo, J.-H., Wei, X.-S., and Wu, J. In defense of fully connected layers in visual representation transfer. In *Pacific Rim Conference on Multimedia*, pp. 807–817. Springer, 2017.

Zhang, Y., Zhang, Y., and Yang, Q. Parameter transfer unit for deep neural networks. *arXiv preprint arXiv:1804.08613*, 2018.

Zhou, M., Liu, T., Li, Y., Lin, D., Zhou, E., and Zhao, T. Toward understanding the importance of noise in training neural networks. In *International Conference on Machine Learning*, pp. 7594–7602, 2019.