Layer-wise Pruning and Auto-tuning of Layer-wise Learning Rates in Fine-tuning of Deep Networks

Youngmin Ro, Jin Young Choi
Seoul National University, Korea
{treeoflife, jychoi}@snu.ac.kr

Abstract

Existing fine-tuning methods use a single learning rate over all layers. In this paper, first, we discuss that trends of layer-wise weight variations by fine-tuning using a single learning rate do not match the well-known notion that lower-level layers extract general features and higher-level layers extract specific features. Based on our discussion, we propose an algorithm that improves fine-tuning performance and reduces network complexity through layer-wise pruning and auto-tuning of layer-wise learning rates. Through in-depth experiments on image retrieval (CUB-200-2011, Stanford online products, and Inshop) and fine-grained classification (Stanford cars, Aircraft) datasets, the effectiveness of the proposed algorithm is verified.

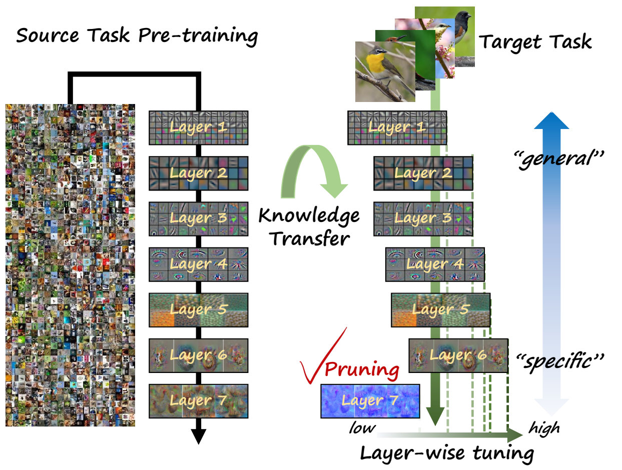

Figure 1. The conceptual figure of the proposed algorithm: Conducting layer-wise pruning and auto-tuning of layer-wise learning rates on the target task according to role of each layer.

1. Introduction

The ability to collect large amounts of data has allowed deep learning to advance dramatically over the last decade. In particular, deep neural network architectures have evolved by targeting large datasets such as ImageNet [4]. However, in actual computer vision applications, large amounts of data such as those contained ImageNet, cannot be easily obtained. Thus, for applications such as image retrieval or fine-grained classification, for which only small datasets are available, the use of pre-trained networks has become essential [5, 34, 22, 6, 3, 26]. To utilize the pre-trained network for a target task, we use a fine-tuning scheme that initially takes pre-trained weights and adjusts them in whole or in part.

The performance of the fine-tuning depends highly on the following factors: similarity between the source and target tasks [1, 2], choice of deep network model [9], and tuning strategy [24, 6, 19]. From among these factors, our paper focuses on enhancement of tuning strategy. Many efforts have been made to enhance the capability of tuning the network for a given target task. [24] has proposed a fine-tuning method that tunes only the higher-level layers. Weight-reverting methods were proposed in [6, 19], where the tuned weights are reverted to the initial (pre-trained) weights during fine-tuning. In addition, some researchers adjusted the learning rate (LR) periodically to ensure efficient learning by adjusting it to follow a triangular shape [21], and even proposing an exponentially decaying type of LR [14]. However, most existing fine-tuning methods adopt a single LR regardless of layers.

In this paper, instead of using a single LR, we propose an algorithm for layer-wise auto-tuning of LRs where the LR in each layer is automatically tuned according to its role. In addition, we propose a layer-wise pruning algorithm that removes layers according to the usefulness of each pre-trained layer to the new task as illustrated in Figure 1. Our work is inspired by our observation that the trends of layer-wise weight variations by conventional fine-tuning using a single LR, contradict previous studies [31, 32], which claim that low-level layers extract general features while high-level layers extract specific features. Based on our observation, we establish two hypotheses and validate them empirically via preliminary experiments. Following the validation of the hypotheses, we develop algorithms for layer-wise pruning and auto-tuning of layer-wise LRs.

The results of the experiment on image retrieval and fine-grained classification tasks show that our methods outperform the existing fine-tuning methods. Furthermore, with regard to image retrieval tasks, our method, which allies only fine-tuning, outperforms the state-of-the-art methods using various kinds of techniques. On the fine-grained classification tasks, our method shows a performance comparable to those of the state-of-the-art methods using add-on techniques.

2. Related works

Fine-tuning is a kind of transfer learning and is used for tuning of pre-trained parametric model. In fine-tuning, the similarity between the source task and the target task is also important as in transfer learning. Regarding the studies on the similarity, Azizpour *et al*. [1] has suggested factors to transfer knowledge well considering the similarity between target and source tasks. And Cui *et al*. [2] proposed a method to promote knowledge transfer using similarity between multiple tasks. In the computer vision fields, ImageNet [4] dataset has been widely used for a source task. Hence lots of target tasks have been handled by using deep network models pre-trained by using ImageNet. In Kornblith *et al*. [9], it has been claimed that there is a positive correlation between ImageNet and most target tasks regardless of deep networks. This implies that ImageNet includes huge amounts of image data covering most image-related tasks. Hence in our paper, we will adopt ResNet-50 pre-trained using ImageNet to validate our algorithm.

In our paper, we are motivated from the role of each layer in fine-tuning a deep network. Regarding the studies on the roles of hidden layers, Yosinski *et al*. [31] conducted empirical study to quantify the degree of generality/specificity of each layer in deep networks. And Zeiler *et al*. [32] visualized features in hidden layers to analyze generality/specificity in the layers. Through the studies [31, 32], they claim that the low-level layers extract general features and the high-level layers extract specific features in a deep network. This claim provides insight into our algorithm.

There have been similar works to ours that exploits the role of each layer. In [24], Tajbakhsh *et al*. have shown that tuning only a few high-level layers is more effective than tuning all layers. Guo *et al*. [6] proposed an auxiliary policy network that decides whether to use the pre-trained weights or fine-tune them in layer-wise manner for each instance. Ro *et al*. [19] proposed a rollback scheme that returns a part of weights to their pre-trained state to improve performance. However, these works adopt existing learning settings unlike our method.

Similarly to ours, there are studies that adjust the learning rate periodically during learning process. Smith [21] suggested a way to adjust the learning rate in a triangular form that linearly reduces learning rate and then grows it again. Similarly Loshchilov *et al*. [14] has also proposed an exponentially decaying and restarting the learning rate for a certain period of time. However, the existing methods for on-line tuning of the learning rate use single learning rate over all the layers. In our work, we aims to develop an algorithm to auto-tune the layer-wise learning rates depending on the role of each layer.

3. Proposed Method

The purpose of this paper is to devise an algorithm that automatically sets the appropriate learning rate (LR) for each layer after removing layers that do not contribute to the learning results in the fine-tuning process. We first discuss our findings observed in the conventional fine-tuning process and then analyze the effects of the layer-wise pruning or layer-wise tuning of LRs through preliminary experiments (Section 3.1). Then, we derive an approximate relationship between the weight variation of each layer and the LR (Section 3.2). Finally, using the results in Sections 3.1 and 3.2, we propose an algorithm for the layer-wise pruning and auto-tuning of LR in each layer (Section 3.3).

3.1. Layer-wise Weight Variations in Fine-tuning

This paper is inspired by previous studies [31, 32, 1]. These studies have demonstrated empirically and qualitatively that lower-level layers extract general features and higher-level layers extract specific features. Based on these results, we build two hypotheses:

Hypothesis 1: *The pre-trained high-level layers may not be helpful to a new target task because they are specific to the source task.*

Hypothesis 2: *The weight variations of pre-trained low-level layers would be small because they are generally valid for most tasks, whereas those in high-level layers would be large because they should adopt themselves to a new task specifically.*

To clarify these hypotheses and support our idea, we define the weight variation of k-th layers as

$$v_t^k = \frac{1}{n_k}\|\Delta w_t^k\|, \tag{1}$$

where $\Delta w_t^k = w_t^k - w_{t-1}^k$ denotes the amount of weight changes in the k-th layer during t-th epoch, and n_k is the number of weights in the k-th layer.

First, we investigate the trends of layer-wise weight variations in the conventional fine-tuning process. The investigation has been conducted on a retrieval dataset (CUB-200-2011 [25]) with ResNet-50 [7]. Figure 2 shows the trends of weight variation between two consecutive epochs. ‘Layer’

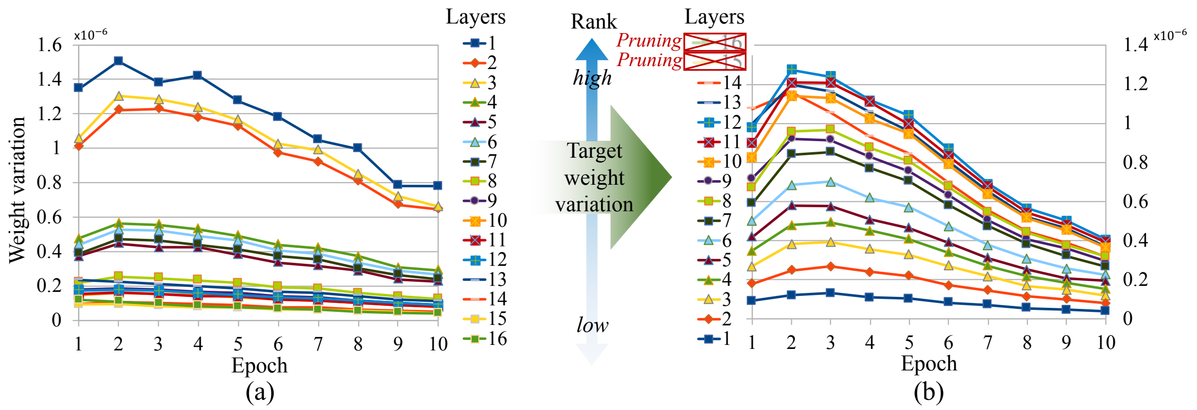

Figure 2. The trend of the weight variation between two adjacent epochs. In the result of the conventional fine-tuning (a), it is observed that the variations of the higher-level layers are small and the lower-level layers are large. (b) The trend after pruning two highest-level layers and tuning layer-wise LRs by trial-and-error. After pruning and layer-wise LR tuning, the performance improved significantly (In Table 1).

denotes a residual block in Resnet-50, where 'Layer 1' indicates the lowest layer to the input layer and thus 'Layer 16' becomes the highest layer. From the results shown in Figure 2-(a), which shows the trends of layer-wise weight variations by the conventional fine-tuning, we have observed two interesting points described in the following two paragraphs. These observations support the key ideas of our paper.

The first point is that the weight variations in high-level layers are too small from fine-tuning, as shown in Figure 2-(a). Based on Hypothesis 1, the high-level layer with small weight variation may not contribute to the new task learning. To support this claim, we have conducted a layer-wise pruning experiments in which the high-level layers were removed one by one from the highest layer. The performance variations by the pruning are shown in Table 1. Interestingly, the pruning until Layer 15 improves the performance, which implies that Layers 15 and 16 might not be helpful to the target task. The pruning of layers from Layer 14 does degrades the performance, which means the layers below Layer 14 are helpful to the new task. This preliminary result supports Hypothesis 1 and can be motivation for our simple but efficient layer-wise pruning scheme.

The second point is that the trends in weight variations by traditional fine-tuning do not match Hypothesis 2, as shown in Figure 2-(a). This implies the traditional fine-tuning destroys the general features of the pre-trained network by large variations in low-level layers and cannot promote the high-level layer to adapt itself to the new task. We believe this result is due to the LR assigned by a single value regardless of layers. To verify this, we have conducted a preliminary experiment by adjusting layer-wise LRs using the empirical trial-and-error approach. The layer-wise adjustment of LRs gives a significant improvement, as shown in rows 5, 6, and 7 of Table 1. Figure 2-(b) shows the order of layer-wise weight variations for row 7 (the best result). Interestingly, we can see the order for row 7 meets Hypothesis 2. This result validates Hypothesis 2 that can be utilized to design an auto-tuning scheme for layer-wise LRs.

Table 1. The Recall@K score results of preliminary experiment for layer-wise pruning and layer-wise adjusting of learning rates. In †, the learning rates in Layers 1,2,3 were reduced to 1/10. In ‡, the learning rates in Layer 13,14 were increased by 10 times. In ⋆, the learning rates in all layers were empirically adjusted to meet Hypothesis 2 by trial and error way.

	R@1	R@2	R@4	R@8
Original	63.49	75.03	84.00	90.58
Prune until 16	66.95	77.63	86.39	92.00
Prune until 15	<u>67.00</u>	<u>78.49</u>	<u>86.85</u>	<u>92.07</u>
Prune until 14	51.00	64.16	75.44	85.94
Prune until 15†	67.15	78.61	86.83	92.08
Prune until 15‡	67.29	79.15	87.36	92.66
Prune until 15*	**68.33**	**79.88**	**87.69**	**92.71**

Based on Hypotheses 1 and 2, we aim to develop an algorithm for the layer-wise pruning and auto tuning of LR (AutoLR). In particular, the layer-wise AutoLR is designed to achieve the order of layer-wise weight variations that meets Hypothesis 2, that is,

$$v_t^1 \leq v_t^2 \leq \cdots \leq v_t^K. \tag{2}$$

3.2. Weight Variation and LR

Our key idea is to control the weight variation of each layer by tuning the LR of each layer. In this section, we derive a relationship between the LR and the weight variation, which will be used in the auto-tuning scheme for LRs according to the layers.

The weights in the k-th layer are updated during l-th iteration for a randomly chosen mini-batch by the stochastic gradient descent rule with the momentum as follows:

$$\Delta w_{t,l}^k = \rho \Delta w_{t,l-1}^k - \eta^k \nabla \mathcal{L}(w_{t,l-1}^k), \tag{3}$$

where $\Delta w_{t,l}^k = w_{t,l}^k - w_{t,l-1}^k$, $\mathcal{L}(w_{t,l-1}^k)$ denotes the loss for $w_{t,l-1}^k$, ρ is a momentum coefficient, and η^k is LR of k-th layer. By applying (3) recursively, we can obtain

$$\begin{aligned}\Delta w_{t,l}^k = &-\eta^k[\nabla \mathcal{L}(w_{t,l-1}^k) + \rho \nabla \mathcal{L}(w_{t,l-2}^k) \\ &+\rho^2 \nabla \mathcal{L}(w_{t,l-3}^k)) + \rho^3 \nabla \mathcal{L}(w_{t,l-4}^k) + \cdots].\end{aligned} \tag{4}$$

Define

$$\begin{aligned}\nabla \mathcal{L}_{acc}\&(w_t^k) = \sum_{l=1}^{L} [\nabla \mathcal{L}(w_{t,l-1}^k) + \rho \nabla \mathcal{L}(w_{t,l-2}^k) \\ + \rho^2 \nabla \mathcal{L}(w_{t,l-3}^k)) + \rho^3 \nabla \mathcal{L}(w_{t,l-4}^k) + \cdots],\end{aligned} \tag{5}$$

where L is the number of mini-batches and $\nabla \mathcal{L}(w_{t,l}^k) = 0$ for $l \leq 0$. Then

$$\Delta w_t^k = \sum_{l=1}^{L} \Delta w_{t,l}^k = -\eta^k \nabla \mathcal{L}_{acc}(w_t^k), \tag{6}$$

which leads to

$$\|\Delta w_t^k\| = \eta^k \|\nabla \mathcal{L}_{acc}(w_t^k)\| \tag{7}$$

From (1) and (7),

$$v_t^k = \frac{\eta^k}{n_k} \|\nabla \mathcal{L}_{acc}(w_t^k)\|. \tag{8}$$

In the next section, this equation is used for the AutoLR scheme. Actually, $\|\nabla \mathcal{L}_{acc}(w_t^k)\|$ is a function of η_k; however, they do not depend on η_k as much, which arises from the vanishing of the high order of η_k in the terms. In AutoLR scheme, we ignore the variation of $\|\nabla \mathcal{L}_{acc}(w_t^k)\|$ between consecutive epochs. Instead, we apply this equation to the AutoLR scheme repeatedly until we get the goal in (2).

3.3. Layer-wise Pruning and Auto-tuning of LR

Based on the results analyzed above, we propose an algorithm to prune layers that are not helpful to the target task. Then we also propose an algorithm (AutoLR) to automatically adjust the LR of each layer so that the order of the weight variation size of each layer is consistent with (2). AutoLR is divided into two parts: setting the target weight variation and tuning by adjusting the LR accordingly.

3.3.1 Layer-wise pruning rule

Before applying layer-wise AutoLR, low-contributed high-level layers are pruned by the procedure in Algorithm 1. In the pruning procedure, we fine-tune a network on a target task using the traditional fine-tuning scheme with a layer-wise fixed LR.

Algorithm 1 Layer-wise Pruning

```
Notation:
    $\eta$ : learning rate of a network
    *network* : network with weight parameters
    *score* : performance of current network
    *best score* : best performance of previous networks
Pruning:
 1: Set $\eta$ with a constant
 2: *network* ← *pre-trained network*
 3: *best score*=0
 4: Fine-tune *network*
 5: *score* ← performance of *network*
 6: **while** *score* $\geq$ *best score* **do**
 7:     *best score* ← *score*
 8:     *network* ← *pre-trained network*
 9:     Prune the highest-level layer of *network*
10:     Fine-tune *network*
11:     *score* ← performance of *network*
12: **end while**
```

3.3.2 Setting target weight variations

We need to renew the target weight variations that satisfy the goal in (2) in each epoch of the learning process. We design two formulas to set the renewed target weight variation depending on *sorting quality* that denotes how well the current weight variations satisfy the goal in (2). To this end, we measure the *sorting quality* defined as follows:

$$sorting\ quality = 1 - \frac{2}{K^2} \sum_{k=1}^{K} |k - \sigma(v_t^k)|, \tag{9}$$

where $\sigma(v_t^k)$ is a function that maps v_t^k to its ranking in ascending order, and $\frac{2}{K^2}$ is the scale factor that enforces the sorting quality to be scaled within 0 and 1.

If the *sorting quality* is above a pre-defined threshold (τ_s), sorting can be considered successful. If the *sorting quality* is below a pre-defined threshold (τ_r), the target weight variation $\bar{v}_t^{(k)}$ is completely reset based on the distribution of the current weight variation as follows:

$$d_t = \frac{1}{K-1}\left(\alpha \max_{1 \leq k \leq K} v_t^{(k)} - \beta \min_{1 \leq k \leq K} v_t^{(k)}\right) \tag{10}$$

$$\bar{v}_t^{(k)} \leftarrow \min_{1 \leq i \leq K} v_t^{(k)} + (k-1)d_t,\ k = 1, \cdots, K. \tag{11}$$

where the α and β are the hyper-parameter to set a range of weight variation.

If the *sorting quality* is greater than τ_r, the target weight variation requires only partial modifications. To prevent the renewed target sorting from shifting to one side, the modification starts from the center ($\check{k}$) and moves to both ends as follows:

For $k = \check{k}, \check{k}+1, \cdots, K$

$$\bar{v}_t^{(k)} \leftarrow \begin{cases} v_t^{(k)} & k = \check{k} \text{ or } \bar{v}_t^{(k-1)} \leq v_t^{(k)} \\ \bar{v}_t^{(k-1)} + d_t & \text{otherwise,} \end{cases} \quad (12)$$

For $k = \check{k}-1, \check{k}-2, \cdots, 1$

$$\bar{v}_t^{(k)} \leftarrow \begin{cases} v_t^{(k)} & \bar{v}_t^{(k+1)} \geq v_t^{(k)} \\ \bar{v}_t^{(k+1)} - d_t & \text{otherwise.} \end{cases} \quad (13)$$

3.3.3 Renewing LR

To get a new LR $\bar{\eta}_t^k$ which produces the renewed target weight variance $\bar{v}_t^{(k)}$, we utilize (8). As mentioned in 3.2, assuming that $\|\nabla\mathcal{L}_{acc}(w_t^k)\|$ does not vary so much between epochs, we can set $\bar{v}_t^k \approx \frac{\bar{\eta}^k}{n_k}\|\nabla\mathcal{L}_{acc}(w_t^k)\|$, which, along with (8), leads to

$$\bar{v}_t^k - v_t^k \approx \frac{\bar{\eta}_t^k - \eta_t^k}{n_k}\|\nabla\mathcal{L}_{acc}(w_t^k)\| = \frac{\bar{\eta}_t^k - \eta_t^k}{\eta^k} v_t^k. \quad (14)$$

Finally we can get the renewed LR as

$$\bar{\eta}_t^k \approx \frac{\eta_t^k(\bar{v}_t^k - v_t^k)}{v_t^k} + \eta_t^k. \quad (15)$$

In actual AutoLR scheme, convergence to the goals of each epoch cannot be done at once, so we adopt an iteration policy as follows:

$$\bar{\eta}_t^k \leftarrow \frac{\eta_t^{k(i)}(\bar{v}_t^k - v_t^{k(i)})}{v_t^{k(i)}} + \eta_t^{k(i)}, \quad \eta_t^{k(0)} = \eta_t^k, \quad (16)$$

where i is the iteration index. This renew procedure is repeated until the goal (2) is achieved in each epoch. The overall flow of AutoLR is given in Algorithm 2.

Algorithm 2 AutoLR: Auto-tuning of learning rates

```
Notation:
    $\eta^k$ : learning rate of $k$th block
    $\bar{\eta}^k$ : target learning rate of $k$th block
    $v_t^k$ : weight variation of $k$th block in $t$th epoch
    $\bar{v}_t^k$ : target weight variation of $k$th block in $t$th epoch
    network : network with tuned weights
    trial-network : trial network for tuning of $\eta^k$
Auto-tuning:
 1: Initialize $\{\eta_k\}$ with $\{\eta_k^0\}$
 2: Initialize network with pre-trained network
 3: for epoch $\leftarrow 1$ to $T$ do
 4:     trial-network $\leftarrow$ network
 5:     Set sorting quality $= 0$.
 6:     while sorting quality $\leq \tau_s$ do
 7:         Fine-tune trial-network for target dataset
 8:         Calculate weight variation in (1)
 9:         Calculate sorting quality in (9)
10:         if sorting quality $> \tau_s$ then
11:             network $\leftarrow$ trial-network
12:         else
13:             if sorting quality $< \tau_r$ then
14:                 Renew $\bar{v}_t^k$ by (11)
15:             else
16:                 Renew $\bar{v}_t^k$ by (12) and (13)
17:             end if
18:             Renew $\bar{\eta}^k$ by (15)
19:             $\eta^k \leftarrow \bar{\eta}^k$
20:         end if
21:     end while
22:     epoch++
23: end for
```

4. Experiments

This section shows our experimental results. We begin by describing the target datasets and metric (Section 4.1). Then, the experimental details are presented (Section 4.2). We then describe the ablation study (Section 4.3) and visualization results (Section 4.4) of our algorithm. Finally, we show the comparison results with existing methods in the remaining sections.

4.1. Datasets and metric

Image retrieval datasets To verify our method, we conducted experiments on benchmark datasets for image retrieval tasks and fine-grained classification tasks. Three datasets for image retrieval tasks are given in the following. *CUB-200-2011 [25]* dataset consists of 200 different species of birds. The first 100 classes are used for training and the other 100 classes for testing. Especially this dataset can be also utilized for fine-grained classification. This dataset is denoted by '*CUB-Retrieval*' for retrieval and '*CUB-C*' for classification. We do not use bounding box annotations. *Stanford Online Products (SOP) [17]* has 120,053 images of 22,634 categories of products. 11,318 and 11,316 classes are used for training and testing, respectively. *Inshop [13]* has 54,642 images of 11,735 categories of clothing items. 3,997 classes are used for training and the other 3,985 classes for testing. In the case of Inshop, the test dataset is divided into query and gallery.

Fine-grained classification datasets In order to verify generality of our method, we conducted experiments on fine-grained classification tasks. Including *CUB-C* above, the following two datasets are used for fine-grained classification tasks. *Stanford Cars [10]* dataset has 196 categories of car images and its total number is 16,185. *Aircraft [15]* dataset has 100 categories of aircraft images and its total number of images is 10,000.

Evaluation metric The Recall@K metric [17] is employed for evaluating compared methods in image retrieval task. For the classification task, we employ a top-1 accuracy.

Table 2. Recall@K score of proposed method on image retrieval dataset for the ablation study

Ablation variants	CUB-Retrieval				Stanford online products				InShop			
	1	2	4	8	1	10	10^2	10^3	1	10	20	30
(1): baseline	63.88	75.14	83.86	90.26	80.02	91.21	96.20	98.64	87.64	97.12	98.04	98.40
(2): (1) + layer pruning	67.00	78.49	86.85	92.07	83.31	92.90	96.68	98.62	88.12	97.10	97.92	98.35
(3): (2) + auto-tuning	**70.41**	**80.89**	**88.22**	**93.05**	**84.10**	**93.31**	**97.01**	**98.88**	**91.97**	**98.12**	**98.64**	**98.87**

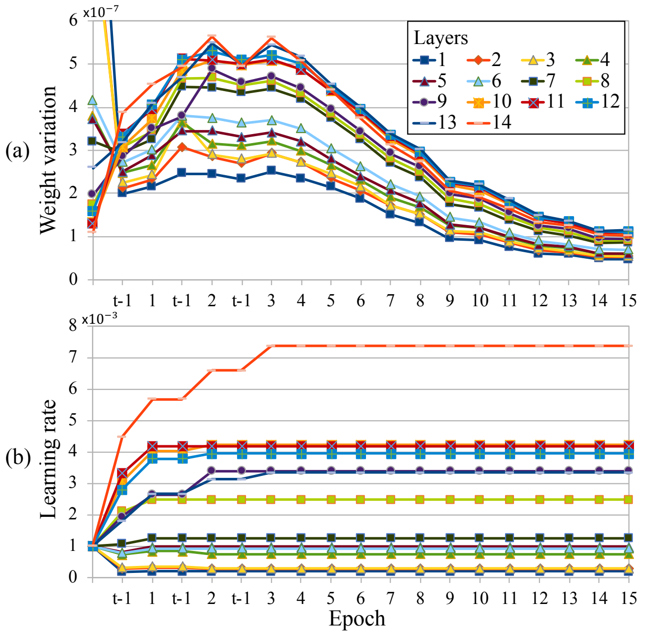

Figure 3. (a) Layer-wise weight variations and (b) layer-wise learning rate adaptations by our AutoLR algorithm

4.2. Implementation detail

We utilized ResNet-50 [7] pre-trained by using ImageNet [4] for our base network. The input size was set to 224×224 for image retrieval tasks and 448×448 for fine-grained classification tasks. The batch size was set to 40 for retrieval, and 16 for classification. The hyper-parameters α and β for setting a range of weight variation were empirically set to 0.4 and 2 for CUB-Retrieval and classification tasks. In Stanford online products and Inshop, α and β were set to 0.5 and 2. The threshold parameters τ_s τ_r were set to 0.94 and 0.5, respectively. We used cross-entropy soft-max as a loss function. The stochastic gradient descent (SGD) optimizer was used along with Nesterov momentum [16]. Initial momentum rate and weight decay coefficient were set to 0.9 and 5e-4, respectively.

4.3. Ablation Study

We conducted ablation studies to validate the components of the proposed algorithm. We consider three ablation variants: (1) use conventional fine-tuning with the same LR over all layers (baseline), (2) apply layer-wise pruning to the baseline, and (3) apply layer-wise AutoLR to (2). The ablation studies were conducted on CUB-Retrieval, SOP, and Inshop with the pre-trained ResNet-50. Layer-wise pruning was done using our proposed rule described in Section 3.3.1. Then, for all the target tasks (CUB-Retrieval, SOP, and Inshop), the layer-wise pruning until Layer 15 showed the best performance. The second row of Table 2 shows a consistent performance improvement over the two target datasets: CUB-Retrieval and SOP. But for the Inshop dataset, the performance does not improve, but the pruning of two layers contributes to a reduction in the network complexity while maintaining comparable performance. The results show that our simple layer-wise pruning is an effective way to both improve performance and reduce network complexity.

The results for variant (3) are shown in the third row of Table 2. There are meaningful performance improvements for the three target tasks. Figure 3 shows the layer-wise trends of weight variations and LRs by our algorithm for layer-wise AutoLR. The learning was done up to 30 epochs, but after the convergence process was omitted after 15 epochs. t-i represents the i-th automatic tuning trial in each learning epoch. Figure 3-(a) shows that our AutoLR algorithm achieves the goal that the magnitudes of the weight variations are sorted in ascending order from low-level to high-level layers. In Figure 3-(a), Layer 14 is observed to be unsorted after four epochs. This is because the parameter τ_s to determine the successful sorting quality is not set to 1 (perfect sorting). Figure 3-(b) shows how the layer-wise LRs are adjusted from the initial LR of 0.001 for all layers and converge to layer-wise constants. The trial tuning iteration was performed once or twice before three epoch, and thereafter the first tuning was successful without further trial tuning. Hence the additional overhead for trial tuning iterations required by the proposed method is negligible. According to the LR trends in the highest layer 14, its change in each tuning was the largest. This result supports our Hypothesis 2 that the high-level layer is specific to the target task and thus its weight variations should be large to adapt itself to the new target task. To meet the goal, the LR in Layer 14 converges to a large value promptly. Our AutoLR algorithm also is valid for other layers, as shown in Figure 3-(b). In conclusion, the ablation study illustrates that the proposed layer-wise pruning and AutoLR algorithm

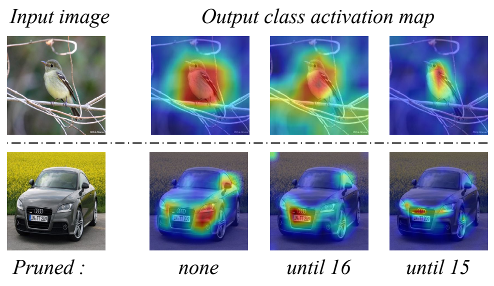

Figure 4. The class activation map (CAM) visualization of the last layers by applying different layers pruning

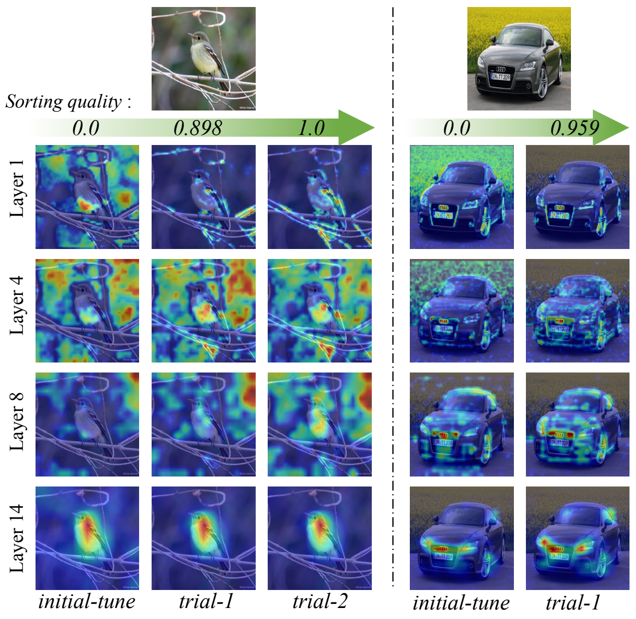

Figure 5. The class activation map (CAM) visualization of several layers (1, 4, 8, 14) according to the sorting quality. *initial-tune* is done by the conventional fine-tuning with single LR

is an effective and promising ways io improve performance and reduce network complexity.

4.4. Visualization on Effect to Layer-wise Features

To understand how the sorting quality of layer-wise weight variations affects the responses in the layers, we investigated the class activation map (CAM) in each layer using a visualization technique (Grad-CAM) [20].

Figure 4 shows the CAM at the last layer of each pruned model. In the case of *none* without pruning, activation gives attention to a relatively large area. However, as high-level layers are pruned one by one, activation has more attention to the object or specific area, although the receptive field of each pruned one is the same. This supports our Hypothesis 1 that the there may be useless high-layers of a pre-trained network in a new task.

Figure 5 shows the CAM at each layer at the first epoch, where the sorting quality increases by AutoLR via one or two trial iterations. The *initial-tune* in Figure 5 is done using the conventional fine-tuning with fixed LR and the remaining trials are done using our AutoLR algorithm. As shown in Figure 5, the CAM at each layer tends to have more attention to the object as the sorting quality increases by our AutoLR. In Layer 1, as the sorting quality increases, unnecessary areas are deactivated and essential activation is formed in the target object area. This is because the AutoLR does not corrupt general features on the unnecessary area while the existing fine-tuning learns excessively the unnecessary area as a specific feature of the new target. In Layer 4, the CAM does not vary on unnecessary area; the CAM on the target area tends to be more attentive as the sorting quality increases. In Layer 8, the CAM on the target area tends to be more attentive as the sorting quality increases. However, the CAM also be attentive on unnecessary areas for the bird image due to the high LR tuned by our algorithm. In Layer 14, due to the pruning Layers 15 and 16 not being well-tuned by the existing fine-tuning, as shown in Figure 4, Layer 14 already has a good attention to the target. However, the activation is more attentive to the target as the sorting quality increases.

Table 3. Comparison of our AutoLR with the various learning rate setting method for the CUB-Retrieval dataset with the equally pruned ResNet-50 (Recall@K Score)

Method	hyper-parameters	R@1	R@2	R@4
Step-decay[9]	$t_d : 20, \gamma : 0.1$	67.22	78.53	86.88
	$t_d : 20, \gamma : 0.5$	66.37	78.73	<u>87.17</u>
Cyclic [21]	$cycle : 3$	66.93	77.55	86.29
	$cycle : 4$	66.19	77.72	85.85
Warm restart [14]	$T_0 : 30/9 , T_{mult} : 1$	67.69	<u>79.41</u>	86.53
	$T_0 : 30/15, T_{mult} : 1$	<u>68.50</u>	79.16	86.93
	$T_0 : 30/18, T_{mult} : 1$	67.40	78.58	86.77
AutoLR	$\alpha : 0.2, \beta : 1$	**70.54**	**80.89**	**88.22**

4.5. Comparison with Other LR Settings

Our AutoLR algorithm belongs to a category for the setting of LRs. Here, we verify the superiority of our algorithm by a comparative study on the setting of LRs. The compared methods are ‘Step-decay’ [9], ‘Cyclic’ [21], and ‘Warm restart’ [14]. Step-decay is the most widely used method in fine-tuning [9, 23, 19]. It conducts LR decay by multiplying to all layers by a value γ at a drop timing t_d. The initial LR of Step-decay is set to 10^{-3}. Cyclic [21] triangularly adjusts the LR between the max value l_{max} and min value l_{min}, where starting at an initial LR, it decreases to l_{min} linearly and then increases to l_{max} linearly. In their experiment, l_{max} and l_{min} were set to 10^{-2} and 10^{-3}, respectively. In Warm restart [14], the LR decays exponen-

Table 4. Comparison with state-of-the-art methods on image retrieval datasets (Recall@K Score)

Method	Network	CUB-Retrieval 1	2	4	8	Stanford online products 1	10	10^2	10^3	Inshop 1	10	20	30
A-BIER [18]	Inception-v1	57.5	68.7	78.3	86.2	74.2	86.9	94.0	97.8	83.1	95.1	96.9	97.5
DREML [29]	Inception-v3	58.9	69.6	78.4	85.6	-	-	-	-	78.4	93.7	95.8	96.7
ABE-8 [8]	Inception-v1	60.6	71.5	79.8	87.4	76.3	88.4	94.8	98.2	87.3	96.7	97.9	98.2
NormSoft [33]	ResNet-50	61.3	73.9	83.5	90.0	79.5	91.5	96.7	-	89.4	97.8	**98.7**	**99.0**
Margin [28]	ResNet-50	63.6	74.4	83.1	90.0	72.7	86.2	93.8	98.0	-	-	-	-
MS [26]	Inception-v1	65.7	77.0	86.4	91.2	78.2	90.5	96.0	98.7	89.7	97.9	98.5	98.8
SCHM [22]	Inception-v1	66.2	76.3	84.1	90.1	77.6	89.1	94.7	-	91.9	98.0	**98.7**	**99.0**
BFE [3]	ResNet-50	-	-	-	-	83.0	**93.3**	**97.3**	**99.2**	89.1	96.3	97.6	98.5
Pruning only	ResNet-50	67.0	78.5	86.9	92.1	81.7	91.9	96.2	98.6	87.7	96.9	97.7	98.2
Proposed	ResNet-50	**70.4**	**80.9**	**88.2**	**93.1**	**84.1**	**93.3**	97.0	98.9	**92.0**	**98.1**	98.6	98.9

Table 5. The comparison results (Top-1 Accuracy) with existing methods on fine-grained classification dataset, † indicates the case using additional network module.

Method	Network	ST-Car	Air	CUB-C
RA-CNN [5]	VGG-19†	92.5	88.2	85.3
MA-CNN [34]	VGG-19†	92.8	89.9	86.5
DT-RAM [12]	ResNet-50†	93.1	-	86.0
DFL-CNN[27]	ResNet-50†	93.1	<u>91.7</u>	87.4
iSort-Cov [11]	ResNet-101	93.3	91.4	**88.7**
NTS-Net [30]	ResNet-50†	**93.9**	91.4	<u>87.5</u>
Spot-tune[6]	ResNet-50†	92.4	-	84.0
Pruning only	ResNet-50	93.0	89.5	84.6
Proposed	ResNet-50	<u>93.8</u>	**91.8**	84.9

tially and then is reset to its initial value. This reset procedures are repeated multiple times. The i-th reset duration is defined by T_i. T_{mult} denotes the factor that controls T_i. The initial leaning rate was set to 10^{-2}.

For fair comparison, all methods were applied to the equally pruned network, and all experiments were conducted equally for 30 epochs. For the hyper-parameters of each method, we tested the performance in a range guided in their methods and selected the values demonstrating the best performance. In Table 3, the tuned hyper-parameters are listed for all the methods. As shown in Table 3, our AutoLR algorithm outperforms the other methods consistently.

4.6. Validity on Fine-grained Classification

To verify that our method is valid for other tasks, we conducted experiments on the fine-grained classification task. As shown in Table 5, our method outperforms Spot-tune [6], which is a similar method that performs layer-wise tuning with the same network ResNet-50. Spot-tune requires additional weights to determine layer-wise tuning, while our method reduces weights through layer-wise pruning. Nevertheless, our method performs better performance than Spot-tune. Hence, this result is quite meaningful. In addition, as shown in Table 5, the proposed method using only fine-tuning achieves comparable performance to the state-of-the-art methods that use various kind of add-on techniques.

4.7. Comparison with Others in Image Retrieval

Table 4 shows that the Recall@*1* score of our **Layer-wise Pruning and AutoLR** achieves the best performance on all three retrieval datasets even though our method uses only the fine-tuning method with no add-on techniques. In particular, for the CUB-Retrieval datasets, the performance of the proposed method outperforms the latest SOTA SCHM [22] with margins of more than 4%. Another impressive one is that our method with only a pruning scheme outperforms the current SOTA even though it is a simple pruning method that removes just a couple of high-level layers assessed to be useless to a new task.

5. Conclusion

In this paper, we proposed a novel algorithm for layer-wise pruning and auto-tuning of layer-wise LRs. The pruning algorithm uses a simple technique to prune a couple of high-level layers that are not helpful to a new task. The auto-tuning algorithm automatically adjusts the LRs depending on the role of each layer so that they contribute to performance improvement. The advantages of the proposed algorithm are not only simple for implementation, but also effective in improving performance and reducing network complexity. The effectiveness and efficiency of the proposed algorithm has been validated by the experiments on image retrieval and fine-grained classification tasks. Hence the proposed automatic pruning and tuning algorithm will be able to contribute to the advances for automated and efficient machine learning.

References

[1] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1790–1802, 2015. 1, 2

[2] Yin Cui, Yang Song, Chen Sun, Andrew Howard, and Serge Belongie. Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4109–4118, 2018. 1, 2

[3] Zuozhuo Dai, Mingqiang Chen, Xiaodong Gu, Siyu Zhu, and Ping Tan. Batch dropblock network for person re-identification and beyond. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1, 8

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*. Ieee, 2009. 1, 2, 6

[5] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, volume 2, page 3, 2017. 1, 8

[6] Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: transfer learning through adaptive fine-tuning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4805–4814, 2019. 1, 2, 8

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 6

[8] Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning. In *The European Conference on Computer Vision (ECCV)*, September 2018. 8

[9] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018. 1, 2, 7

[10] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. 6

[11] Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 947–955, 2018. 8

[12] Zhichao Li, Yi Yang, Xiao Liu, Feng Zhou, Shilei Wen, and Wei Xu. Dynamic computational time for visual attention. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1199–1209, 2017. 8

[13] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016. 5

[14] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 1, 2, 7, 8

[15] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 6

[16] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, 1983. 6

[17] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016. 5, 6

[18] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Bier-boosting independent embeddings robustly. In *ICCV*, pages 5189–5198, 2017. 8

[19] Youngmin Ro, Jongwon Choi, Dae Ung Jo, Byeongho Heo, Jongin Lim, and Jin Young Choi. Backbone can not be trained at once: Rolling back to pre-trained network for person re-identification. In *AAAI*, 2019. 1, 2, 7

[20] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017. 7

[21] Leslie N Smith. Cyclical learning rates for training neural networks. In *WACV*, 2017. 1, 2, 7, 8

[22] Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *CVPR*, June 2019. 1, 8

[23] Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang. Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline). In *The European Conference on Computer Vision (ECCV)*, September 2018. 7

[24] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016. 1, 2

[25] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2, 5

[26] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *CVPR*, June 2019. 1, 8

[27] Yaming Wang, Vlad I Morariu, and Larry S Davis. Learning a discriminative filter bank within a cnn for fine-grained recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4148–4157, 2018. 8

[28] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017. 8

[29] Hong Xuan, Richard Souvenir, and Robert Pless. Deep randomized ensembles for metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 723–734, 2018. 8

[30] Ze Yang, Tiange Luo, Dong Wang, Zhiqiang Hu, Jun Gao, and Liwei Wang. Learning to navigate for fine-grained classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 420–435, 2018. 8

[31] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014. 1, 2

[32] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 1, 2

[33] Andrew Zhai and Hao-Yu Wu. Making classification competitive for deep metric learning. *arXiv preprint arXiv:1811.12649*, 2018. 8

[34] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *Int. Conf. on Computer Vision*, volume 6, 2017. 1, 8