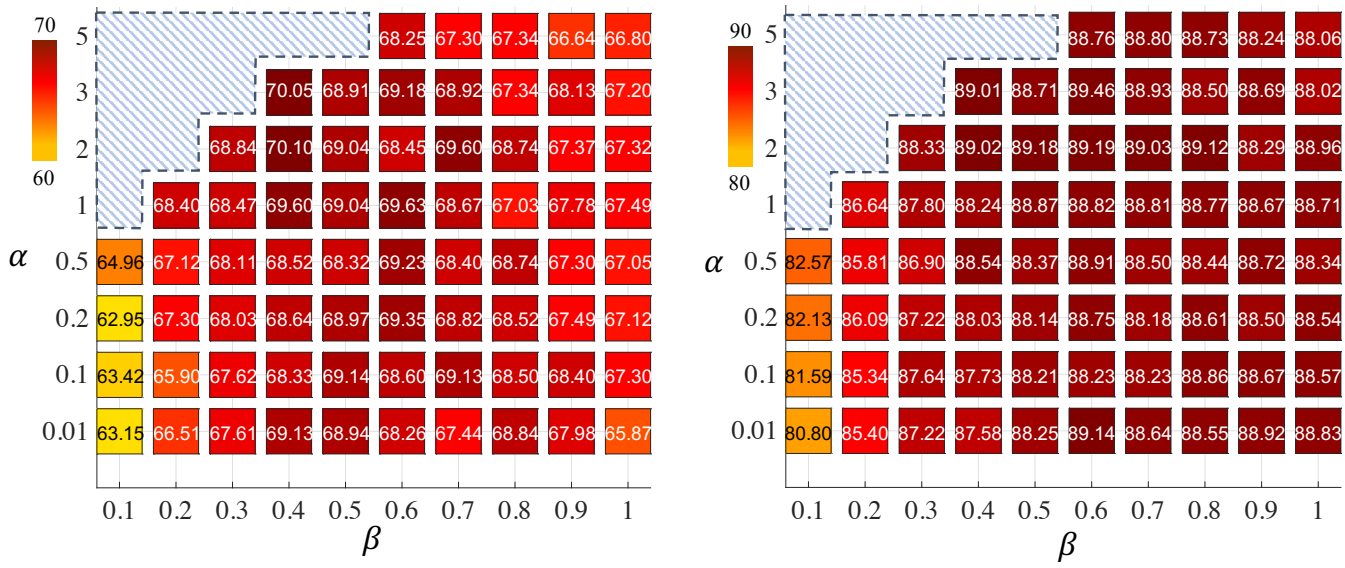
[Supplementary Material]
AutoLR: Layer-wise Pruning and Auto-tuning of Learning Rates in Fine-tuning of Deep Networks

Youngmin Ro[1,2] Jin Young Choi[1]
youngmin.ro@samsung.com, jychoi@snu.ac.kr
[1]Department of ECE, ASRI, Seoul National University, Korea
[2]Samsung SDS, Korea

This supplement contains additional experimental results that are omitted from the main paper due to space constraints.

A. Hyper-parameters α, β guidance for AutoLR

In the main paper, we have found that the tuning of layer-wise learning rates (LRs) yields much improved performance. To manually tune the best layer-wise LRs, there are too much combinations of layer-wise LRs in a multi-dimensional continuous real space even for the given upper bound and lower bound over the range of LRs. But our AutoLR can easily find the layer-wise LRs with only using the lower and upper bounds that are denoted by $\alpha \times \min_k v_0^{(k)}$ and $\beta \times \max_k v_0^{(k)}$, respectively, where α and β are hyper-parameters and $v_0^{(k)}$ is the initial weight variation of k-th layer obtained by single LR. In this section, we provide the investigation results for the hyper-parameters α and β on the CUB-200 [8] and Cars-196 [3] datasets. As shown in Figure 1-(a), the best performance on CUB-200 dataset is achieved when the α and β are set to values around 2 and 0.4, respectively. We selected the best values of α and β for main paper experiments. In the case of Cars-196 dataset, the setting α and β to 2 and 0.4 provides relatively high performance as shown in Figure 1-(b) but the setting α and β to 3 and 0.6 provides the best performance. Note that the hatched area of the Figure 1 is the unavailable area where the lower bound becomes larger than the upper bound.



(a) CUB-200 (b) Cars-196

Figure 1: Recall@1 results according to the hyper-parameters α and β on CUB-200 and Cars-196

B. Convergence Issues in AutoLR's Update Rule

The update rule Eq. (16) for AutoLR is derived by the assumption that $|||\nabla\mathcal{L}_{acc}(w_t^k)|||$ does not vary much. Before we show that the assumption is experimentally valid, we first derive an equation for the difference between the target weight variation and the estimated weight variation obtained by the updated LR. Letting $\tilde{v}_t^k$ be the estimated weight variation by LR updated by Eq. (16), whereas $\bar{v}_t^k$ be the target weight variation at t-th epoch. The convergence of $\tilde{v}_t^k$ to $\bar{v}_t^k$ is shown as

$$
\begin{aligned}
|\tilde{v}_t^k - \bar{v}_t^k| &= \left| \frac{1}{n_k} ||\tilde{w}_t^k|| - \bar{v}_t^k \right|; \quad \text{from} \quad \text{Eq. (1)} \\
&= \left| \frac{1}{n_k} \tilde{\eta_t}^k ||\nabla\mathcal{L}_{acc}(\tilde{w}_t^k)|| - \bar{v}_t^k \right|; \quad \text{from} \quad \text{Eq. (7)} \\
&= \left| \frac{1}{n_k} \left(\eta_t^k \frac{\bar{v_t}^k - v_t^k}{v_t^k} + \eta_t^k \right) ||\nabla\mathcal{L}_{acc}(\tilde{w}_t^k)|| - \bar{v}_t^k \right|; \quad \text{from} \quad \text{Eq. (16)} \\
&= \bar{v_t}^k \left| \frac{\eta_t^k}{n_k v_t^k} ||\nabla\mathcal{L}_{acc}(\tilde{w}_t^k)|| - 1 \right| \\
&= \bar{v_t}^k \left| \frac{||\nabla\mathcal{L}_{acc}(\tilde{w}_t^k)||}{||\nabla\mathcal{L}_{acc}(w_t^k)||} - 1 \right|; \quad \text{from} \quad \text{Eq. (8)} \\
&\approx 0; \text{ because } \gamma := \frac{||\nabla\mathcal{L}_{acc}(\tilde{w}_t^k)||}{||\nabla\mathcal{L}_{acc}(w_t^k)||} \approx 1
\end{aligned}
$$

Here $|||\nabla\mathcal{L}_{acc}(w_t^k)|||$ is determined by the momentum ρ and the gradients as shown in Eq. (5) in the manuscript. In general, the gradient statistically tends to depend on slope of error surface at a weight point. Hence the sum of the gradients in Eq. (5) not explicitly depends on the LR, which is validated in our investigation as shown in Table 1.

Table 1: The ratio γ according to the different LR condition

Layer	1	2	3	4	5	6	7	8	9	10	11	12	13	14
η^k ($\times 10^{-3}$)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$\tilde{\eta}^k$ ($\times 10^{-3}$)	0.16	0.29	0.33	0.76	0.80	0.78	1.05	2.06	1.91	3.05	3.29	2.78	1.73	4.33
$\|\|\|\nabla\mathcal{L}_{acc}(w_t^k)\|\|\|$	66.04	56.63	53.57	87.88	101.17	110.24	87.22	165.47	213.56	140.41	136.67	169.35	285.15	418.59
$\|\|\|\nabla\mathcal{L}_{acc}(\tilde{w}_t^k)\|\|\|$	62.41	51.25	50.56	80.15	86.36	96.92	77.90	142.70	161.59	107.43	107.87	127.23	196.16	333.57
$\gamma = \frac{\|\|\nabla\mathcal{L}_{acc}(\tilde{w}_t^k)\|\|}{\|\|\nabla\mathcal{L}_{acc}(w_t^k)\|\|}$	0.95	0.91	0.94	0.91	0.85	0.88	0.89	0.86	0.76	0.77	0.79	0.75	0.69	0.80

C. Applicability of AutoLR to the Proxy-Anchor Loss

We provide experiments that applied our AutoLR to the Proxy-anchor loss function [2]. The Proxy-anchor loss was designed to use an AdamW [5] optimizer, while our AutoLR was designed to use a stochastic gradient descent optimizer (weight decay and nestrov applied). Thus, we modified Proxy-anchor loss to use a stochastic gradient descent optimizer for applying the AutoLR scheduling to the Proxy-anchor loss function directly. Both official and modified versions used the same ResNet-50 network and the same input size of 224×224.

As shown in the second row of Table 2, the modified version (optimizer change) has a lot of performance degradation compared to the official version. However, applying the proposed pruning method to the modified version yields better performance than the official version in CUB-200. In addition, applying pruning and AutoLR to the modified version at the same time achieves considerable performance gains on both the CUB-200 and Cars-196 datasets. Through these results, we verify that the proposed layer-wise pruning and AutoLR method is applicable to other algorithms and is an effective way that provides additional performance improvements.

D. Generality of AutoLR in Fine-grained Visual Classification Tasks

Since our proposed AutoLR algorithm is designed to improve the general fine-tuning method, in this section we show the generalization of our algorithm for the fine-grained classification tasks. To this end, we employ three fine-grained visual classification (FGVC) datasets FGVC-CUB [8], FGVC-Cars [3], and FGVC-Aircraft [6]. The datasets consist of 200, 196,

Table 2: Recall@K score of proposed method applied to Proxy-anchor loss

Method	CUB-200				Cars-196			
	1	2	4	8	1	2	4	8
Proxy-Anchor + AdamW (Official version)	69.7	80.0	87.0	92.4	87.7	92.9	95.8	97.9
Proxy-Anchor + SGD (Modified version)	61.41	72.81	81.90	88.32	73.09	82.00	87.65	92.29
Proxy-Anchor + SGD + Pruning 15,16	72.57	81.97	88.76	93.64	86.73	91.74	94.84	96.80
Proxy-Anchor + SGD + Pruning 15,16 + AutoLR	**73.46**	**82.48**	**89.30**	**93.52**	**88.55**	**93.06**	**96.00**	**97.49**

and 100 classes, respectively. FGVC-CUB and FGVC-Cars are the same as CUB-200 and Cars-196 in the main paper, but the criteria for dividing train and test datasets are different. The experiments were conducted with training for 40 epochs by the ResNet-50 network and the image size is 224×224.

Table 3 shows the results of applying the proposed method to fine-grained visual classification tasks. Pruning 16 provides the increased performance from none pruned model in FGVC-CUB and FGVC-Cars. Pruning 15 provides further increased performance from Pruning 16 in FGVC-Cars. Additionally, AutoLR provides considerable performance improvements for both non-pruned and pruning model in every datasets. In the case of the FGVC-Aircraft, interestingly AutoLR gives better performance gain when it is applied to the none-pruned model than the pruning model. In the case of the FGVC-Aircraft, AutoLR applied to Pruning 16 model provides the best performance.

The experimental results in classification tasks show that the proposed algorithm is also applicable to the general tasks beyond image retrieval tasks.

Table 3: The classification accuracy for applying Layer-wise pruning and AutoLR to Fine-grained classification task

Model	FGVC-CUB	FGVC-Cars	FGVC-Aircraft
None pruned	78.53	84.65	77.83
Pruning 16	79.13	85.40	77.77
Pruning 15,16	78.74	85.96	77.95
None pruned + AutoLR	79.62	87.89	**82.18**
Pruning 16 + AutoLR	**80.34**	88.12	81.28
Pruning 15,16 + AutoLR	79.93	**88.30**	82.00

E. Compare with Other Learning rate Scheduling Methods

Here we provide the whole results for the 'Comparison with other LR setting' section of the main paper. The compared methods are 'Step-decay' [1], 'Cyclic' [7], and 'SGDR' [4]. Step-decay is the most widely used method in fine-tuning. It conducts LR decay by multiplying to all layers by a value γ at a drop timing t_d as shown in Figure 2. The initial LR of Step-decay is set to l_{max}. Cyclic adjusts the LR between the max value l_{max} and min value l_{min} with periodic triangular waveform as shown in Figure 2. The number of cycle is a hyper-parameter. Similarly, SGDR adjusts the LR to decrease exponentially and LR is reset to its initial value l_{max}. These resets are repeated multiple times with a hyper-parameter n_{reset} as shown in Figure 2. For fair comparison, all methods were applied to the equally pruned network (pruning 15, 16), and all experiments were conducted equally for 50 epochs with 224×224 images. There are two cases according to the setting of l_{max} and l_{min}. The first is the case where l_{max} and l_{min} were set to 10^{-3} and 10^{-4}, respectively. The second is the case where l_{max} and l_{min} were set to 10^{-2} and 10^{-3}, respectively.

Table 4 shows the Recall@K score for each hyper-parameter condition. The step-decay shows higher performance when l_{max} and l_{min} are set to 10^{-3} and 10^{-4} than set to 10^{-2} and 10^{-3}. The highest performance is provided when LR is reduced by 0.5 times at 40 epochs for a total of 50 epoch training. Cyclic and SGDR generally show higher performance when l_{max} and l_{min} are set to 10^{-2} and 10^{-4}. Cyclic shows the highest performance when repeated seven cycles, SGDR shows the highest performance when conducting eight resets. However, all of the other methods are ways to obtain performance improvement by applying single LR, while the AutoLR that applies LR according to the role for each layer outperforms the other LR scheduling methods consistently.

Table 4: Comparison of our AutoLR with the various learning rate setting method for the CUB-200 dataset with the equally pruned ResNet-50 (Recall@K Score)

Method	hyper-parameters	$l_{max}: 10^{-3}, l_{min}: 10^{-4}$ R@1	R@2	R@4	R@8	$l_{max}: 10^{-2}, l_{min}: 10^{-3}$ R@1	R@2	R@4	R@8
Step-decay	$t_d: 30, \gamma: 0.1$	67.00	78.49	86.85	90.26	66.95	78.04	86.41	91.63
	$t_d: 30, \gamma: 0.2$	67.39	79.59	87.51	92.35	67.10	77.92	86.06	91.61
	$t_d: 30, \gamma: 0.5$	68.26	79.84	87.54	92.72	66.63	77.25	85.48	91.31
	$t_d: 40, \gamma: 0.1$	67.91	79.30	87.36	92.39	67.17	78.19	86.19	91.81
	$t_d: 40, \gamma: 0.2$	67.86	79.20	87.20	92.42	67.35	78.14	86.04	91.64
	$t_d: 40, \gamma: 0.5$	68.75	79.74	87.96	92.76	66.64	77.30	85.63	91.73
Cyclic	*cycle* : 1	67.66	78.38	86.44	91.66	66.73	77.70	85.79	91.61
	cycle : 3	67.12	78.09	86.31	92.13	67.84	78.60	86.29	91.73
	cycle : 5	67.06	78.35	86.63	92.05	67.54	78.70	86.73	91.98
	cycle : 7	67.33	78.83	86.65	91.91	68.55	79.17	86.70	91.76
	cycle : 9	67.14	78.69	86.69	91.97	67.51	78.22	86.01	91.46
	cycle : 15	67.55	78.83	86.63	92.02	67.79	79.25	86.85	91.83
SGDR	n_{reset}: 0	67.15	78.75	86.93	92.38	66.42	77.41	86.44	92.20
	n_{reset}: 2	66.61	78.59	86.88	92.34	67.78	78.53	86.19	91.61
	n_{reset}: 4	66.62	78.53	86.99	92.33	67.81	79.15	87.04	92.47
	n_{reset}: 6	66.54	78.63	86.88	92.26	67.79	78.73	86.75	92.29
	n_{reset}: 8	67.29	78.36	86.31	91.85	68.18	79.34	86.92	92.27
	n_{reset}: 14	67.37	78.93	86.63	92.45	68.16	78.98	86.50	91.91
AutoLR	$\alpha: 2, \beta: 0.4$			R@1 **70.10**	R@2 **80.62**	R@4 **88.08**	R@8 **92.98**		

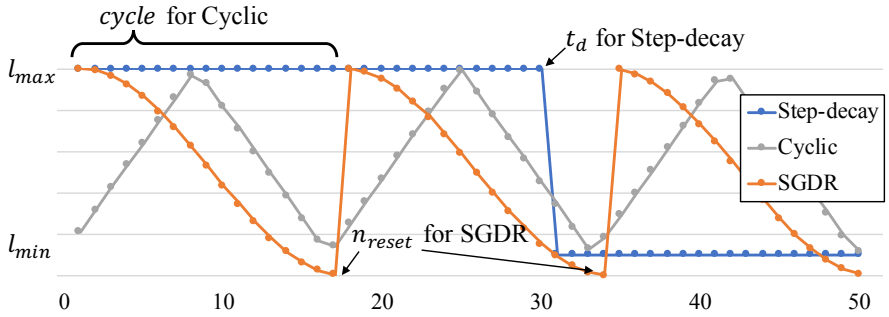


Figure 2: Illustration of the learning rate scheduling methods. (Three cycle for Cyclic; Twice reset for SGDR; 30 epoch dropping for Step-decay)

F. Class Activation Map for AutoLR

Here, we provide class activation map of all layers when applied by the proposed AutoLR. As shown in Figure 3, the activation areas are focused on the bird rather than on the background as the sorting quality grows. This patterns are observed consistently across all layers.

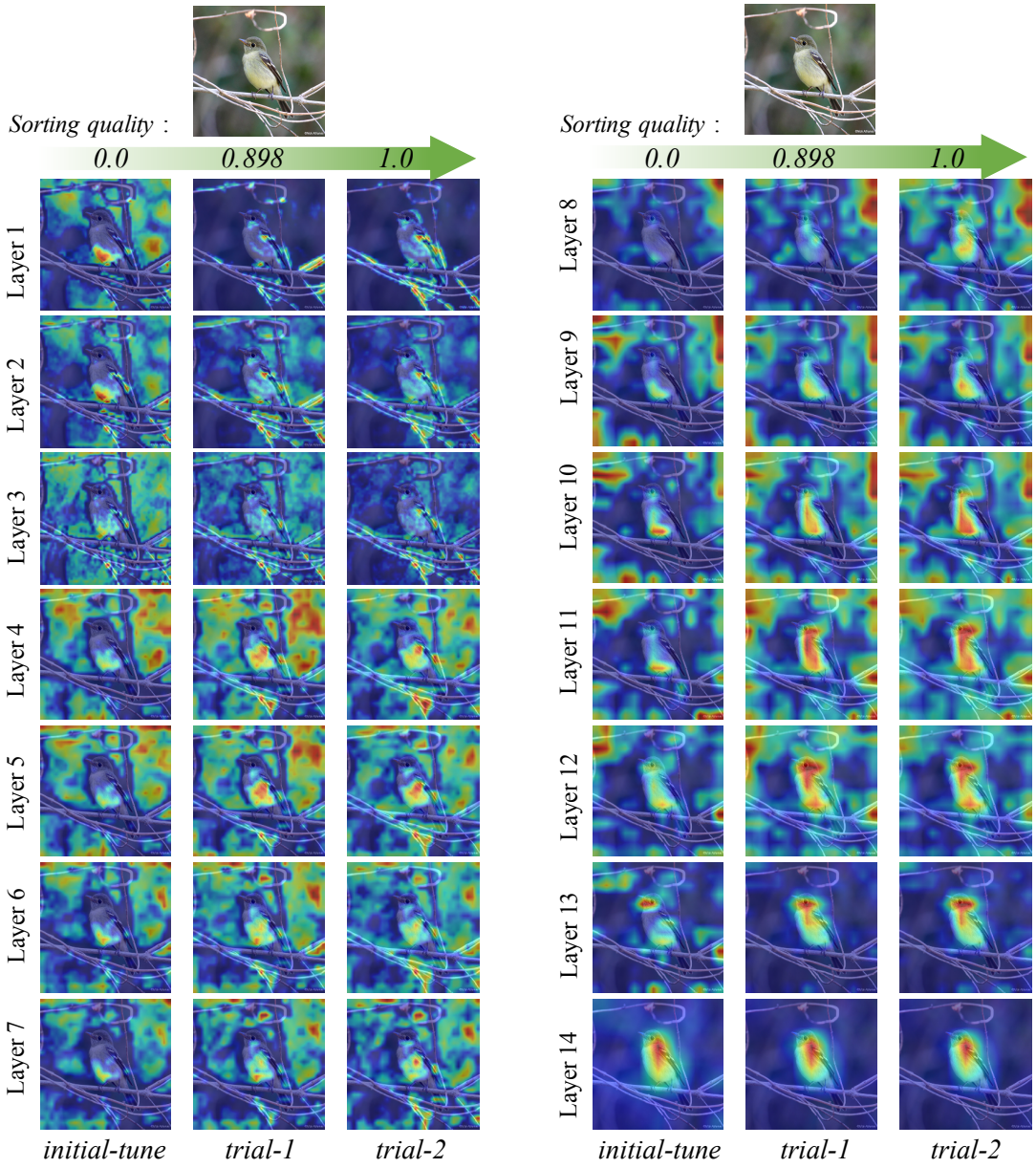


Figure 3: The class activation map (CAM) visualization of all layers according to the sorting quality. *initial-tune* is done by the conventional fine-tuning with single LR

References

[1] R. Ge, S. M. Kakade, R. Kidambi, and P. Netrapalli. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. In *Advances in Neural Information Processing Systems*, pages 14977–14988, 2019.

[2] S. Kim, D. Kim, M. Cho, and S. Kwak. Proxy anchor loss for deep metric learning. In *CVPR*, 2020.

[3] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[4] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[5] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[6] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[7] L. N. Smith. Cyclical learning rates for training neural networks. In *WACV*, 2017.

[8] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.