Adaptive Learning Rate Adjustment with Short-Term Pre-Training in Data-Parallel Deep Learning

Kazuki Yamada, Haruki Mori, Tetsuya Youkawa, Yuki Miyauchi, Shintaro Izumi,
Masahiko Yoshimoto, and Hiroshi Kawaguchi
Graduate School of System Informatics, Kobe University, Kobe, Japan
yamada.kazuki@cs28.cs.kobe-u.ac.jp

***Abstract*— This paper introduces a method to adaptively choose a learning rate (LR) with short-term pre-training (STPT). This is useful for quick model prototyping in data-parallel deep learning. For unknown models, it is necessary to tune numerous hyperparameters. The proposed method reduces computational time and increases efficiency in finding an appropriate LR; multiple LRs are evaluated by STPT in data-parallel deep learning. STPT means training only with the beginning iterations in an epoch. When eight LRs are evaluated using eight parallel workers, the proposed method can easily reduce the computational time by 87.5% in comparison with the conventional method. The accuracy is also improved by 4.8% in comparison with the conventional method with a reference LR of 0.1; thus, no deterioration in accuracy is observed. For an unknown model, this method shows a better training curve trend than other cases with fixed LRs.**

Keywords- Deep learning, learning rate, hyperparameter, data-parallel

I. Introduction

The impact of deep learning has been expanded from an object recognition competition, ImageNet Large-Scale Visual Recognition Competition (ILSVRC) 2012 [1], with a dramatic accuracy advantage; deep learning using convolutional neural networks (CNNs) achieved a steep error rate improvement of 9.4 % in comparison with conventional methods [2]. These days, deep learning is recognized as a significant algorithm. In January 2016, AlphaGo developed by DeepMind Ltd. competed with the world's Go champion. Deep learning proved that, even for the most complex table game, the algorithm is able to acquire a better solution than a human [3]. Moreover, deep learning is expected to bring singularity in various fields. Researchers all over the world exploit deep learning not only in computer science but also for applications in medicine, finance, agriculture, energy management, and other fields [4]. Deep learning represents a breakthrough in the field of artificial intelligence.

The CNN model imitates the visual cortex in the human cerebrum. It is an extension of a multi-layer perceptron that became most successful in image recognition [5]. Various algorithms for deep learning using convolutional networks have been proposed in the object recognition field [6] [7] [8]. Actually, a CNN algorithm is already expected to be put into practical use for security cameras and safety driving assistant applications [9]. The CNN algorithm achieves far superior recognition performance in comparison to other algorithms [10].

Still, there are complex problems to be addressed for various practical applications of CNNs in the future. One of the most noteworthy issues is that a training model requires a huge number of hyperparameters to achieve a high-accuracy recognition rate. Generally, a learning (training) process requires a large amount of training data and a deeper network to ensure good recognition performance. Therefore, training a network model requires enormous computational time even if state-of-the-art processors are used. AlexNet takes about six days to learn 90 epochs of ImageNet data using two NVIDIA GTX 580 GPUs [11]. Also, ResNet with 200 layers (ResNet-200) takes three weeks for ImageNet learning, even using eight parallel GPGPUs [12].

It is well known that ResNet shows better performance with greater numbers of layers; accuracy improvement becomes better even for networks with 1,000 or more layers [11]. Training such a deep network consumes much more time than before. This long training time is a major barrier for the practical application of deep learning. Parallel processing in deep learning is necessary to shorten the training time. Parallel processing may be divided into the following two types [13]:

- Data parallel: Every worker has a replica of the model, and each worker processes different data and learns from the data. The data parallel has a synchronous mechanism in which parameter updating is synchronized among all workers.

- Model parallel: One model is divided into parts, and each worker is associated with a single part. As a result, each worker has less memory capacitance and less memory bandwidth than those in the data parallel. However, it is more difficult to implement because each worker has a different part of the model.

A deeper network requires enormous dimensions of hyperparameters. In this paper, we specifically focus on the learning rate (LR). It is not automatically determined by a neural network; thus, a designer has to select an appropriate LR for its network structure in advance. Although the LR has the greatest influence on accuracy among the hyperparameters, it is currently determined experimentally and empirically in many cases. The problem here is that it is difficult for a trial of parameter tuning to achieve a sufficient result in a single learning process; it is necessary to repeat multiple trials of learning processes.

As a prior work, the Hyperband [14], SMAC [15], and TPE [16] are introduced for the automatic adjustment of hyperparameters. The Hyperband formulates the hyper-parameter optimization scheme as a pure-exploration non-stochastic infinite-armed bandit problem. The predefined resources such as iteration, data samples, and features are allocated to randomly sampled configurations [14]. The tree-structured Parzen estimator (TPE) models $p(x|y)$ by transforming a graph-structured generative process and replacing the distributions of the configuration prior to a non-parametric density[16].

Among them, Methods to algorithmically determine the LR have been proposed. For the methods that adaptively determine the LR, such as Adam [17], AdaGrad [18] and AdaDelta [19], there are advantages and disadvantages; we currently have no optimal method for every problem. To make matters worse, the adaptive training methods increase the number of hyperparameters. After all, it is necessary to try various methods for a new network and to select an LR based on the experimental results. In this paper, we propose an adaptive LR adjustment method with STPT for quick model prototyping.

The remainder of this paper is organized as follows. Section II presents a method for adaptive LR adjustment with STPT. Software implementation of the proposed method and its performance are explained in Section III. The final section suggests directions for future work.

II. Adaptive LR Adjustment with STPT

A. Algorithm

Even in the conventional method, training is performed while the LR is changed, but it is changed every epoch. Then, as mentioned in the previous section, it is necessary to carry out various patterns, which is very time consuming. To solve these problems, we propose an adaptive LR selection method that uses synchronous data parallelism. An overview diagram of the proposed method is shown in **Fig. 1**. A major feature of the proposed method is that it uses multiple LRs and evaluates their accuracy at the end of several iterations (within 1 epoch) in training. This process is described in detail below.

In the proposed method, training for one epoch is divided into two steps: *Pre-train* and *Main-train*. The first step is *Pre-train*. In *Pre-train*, training is performed independently with m ($\leq n$) kinds of LRs using n workers (in **Fig. 1**, $m = n$). The initial values of parameters, such as the model at the time of starting *Pre-train*, are unified for each worker. The important point is that *Pre-train* trains only α iterations. We set α so that $\alpha + \beta \leq 1$ epoch, where β is the number of iterations of *Main-train*. Only the α iteration is trained using a different LR for each worker. The best LR is selected as the *bestLR* based on the accuracy at the time of the α iteration as the evaluation value. The second step is *Main-train*. In *Main-train*, β iterations are trained using the *bestLR* selected during the *Pre-train* stage and the model trained at the α iteration. At this time, synchronous data-parallel of n parallel is used.

The above is the training flow for 1 epoch ($= \alpha + \beta$ iretarions) in the proposed method. Training for the next epoch uses this model for the start parameters of *Pre-train*. In this way, training progresses. As a concept, **Fig. 2** shows training courses obtained when the proposed algorithm was executed using three candidate LRs (*LR set*), 3.0, 1.0, and 0.5. The three short lines at the beginning of each epoch represent the accuracy at the *Pre-train* and the black line represents *Main-train*'s accuracy. Focusing on the trend of accuracy, although the beginning of each epoch greatly rises and falls, it seems that it quickly settles

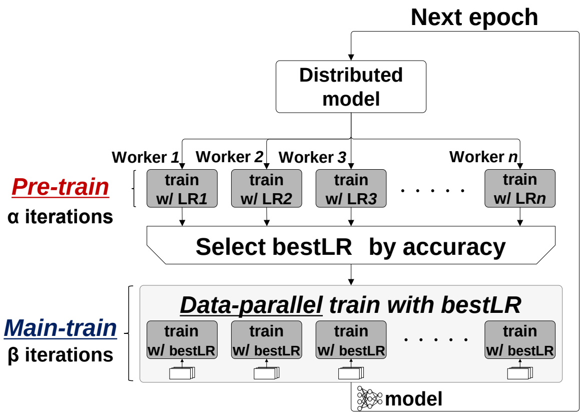

Fig. 1 Overview of the proposed LR selection with STPT algorithm.

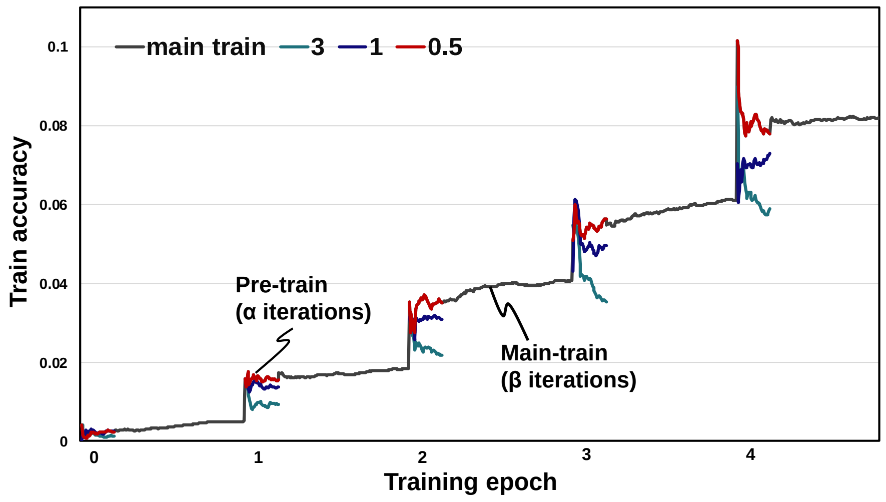

Fig. 2 Concept of the proposed STPT algorithm for LR selection: The best LR is chosen by comparison between the *LR set*: 3.0, 1.0, and 0.5.

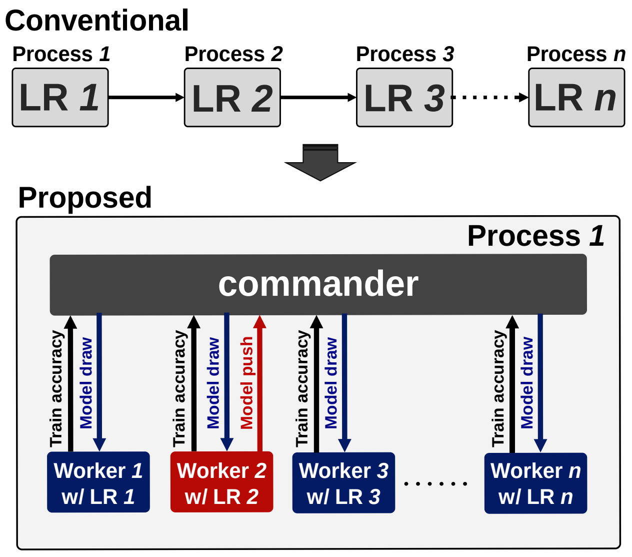

Fig. 3 Comparison of conventional and proposed methods

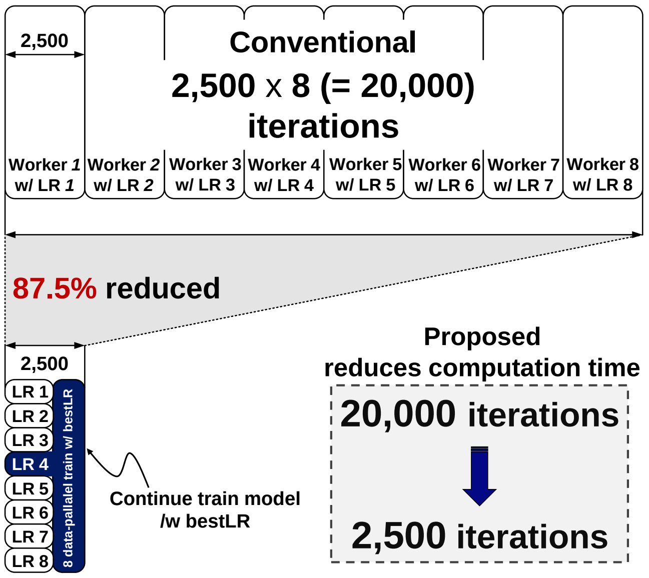

Fig. 4 In comparison with the conventional method, the proposed algorithm can reduce the number of iterations by 87.5%.

and slowly improves. In the proposed method, we use this phenomenon to learn only a number of iterations until the accuracy of each epoch settles. Then, we evaluate each LR based on the accuracy during the settled period; the number of iterations can be reduced. In fact, *Main-train* traces the line of best accuracy in *Pre-train*, which shows that training progresses when the *bestLR* is chosen.

Fig. 3 compares the processes of the conventional and proposed methods. The experimental results are obtained using one LR for one process, but in the proposed method, training is conducting using multiple LRs in one process. Therefore, it is possible to drastically reduce the time spent on experiments. For an experiment with eight LRs and eight workers, **Fig. 4** compares the number of iterations when each LR is used for learning for 1 epoch and using the proposed method setting α as 200 iterations. By using the proposed method, it is possible to reduce the number of iterations by 87.5 % in comparison with the conventional method.

III. Experiment Results

In this work, we conducted an image recognition experiment to evaluate the performance of the proposed method.

A. *Implementation*

Data set used in ILSVRC2012 of ImageNet [20] was used as the data set. In this data set, RGB images of 1,000 categories of general objects having different sizes were prepared with 1,280,000 images of learning data, 50,000 images of verification data, and 100,000 images of test data. We conducted the experiment with a consistent image size of 256 × 256. We used ResNet [11]. ResNet learns including the residuals, and it becomes possible to avoid the gradient elimination problem; thus, it is possible to deepen the layer. Momentum SGD [21] was used for the optimization function, and the LR was multiplied for each epoch. In early epoch training, we used a method called Warmup [11] [22], in which a few epochs are used for training with a small LR at the beginning of learning. As a reference, we set the initial value of the LR to 0.1 and divide it by 10 when the accuracy saturates according to a paper on ResNet [11]. Also, when Warmup is applied to the reference, accuracy deteriorates, so reference don't use Warmup.

TABLE 1 shows the parameters used in this experiment. In this experiment, a software simulation was performed using synchronous eight-parallel data parallelism in one processor. The pseudo code for implementing the proposed algorithm is shown in Algorithm 1. Also, as seen in Algorithm 1, for *Pre-train*, simulation was performed using eight-parallel data parallelism. Chainer [23] was used as a framework for implementation.

TABLE 1 Parameters

Network	ResNet50
Number of workers	8
Batch size (per worker)	512 (64)
Training max epoch	40
Pre-train (α) iterations	200, 600, 1,250, 2,500
Weight initializations	He's initialization [18]
Momentum cofficient	0.9
Initial LR	0.1
LR set	TABLE 2, TABLE 3
Warmup epoch	4
Warmup LR	0.005

B. *α iterations in Pre-train*

In this section, we present an experiment that was conducted with α iterations in Pre-train. Since the batch size was set at 512 (= 64 * 8) in this experiment, 2,500 iterations were required to learn 12.8 million training data for one turn. Therefore, the maximum value of α was 2,500 iterations. For implementation

Algorithm 1 SOFTWARE IMPLEMENTATION OF THE PROPOSED METHOD

```
for epoch in range(max_epoch + warmup_epoch)
    if epoch < warmup_epoch
        run warmup train with warmup_LR using data parallel
    else
        #initialization
        best_accuracy = 0.0
        bestLR = 0.0
##---------------------------------------Pre-train--------------------------------------##
        for LR_index in range( len( LR_set ) )
            new_accuracy = run pre_train with LR_set[ LR_index ] using data parallel
            if new_accuracy > best_accuracy
                bestLR = LR_set[ LR_index ]
                best_accuracy = new_accuracy
##---------------------------------------Main-train-------------------------------------##
        run main_train with bestLR using data parallel
```

TABLE 2 *LR SET* FOR PRE-TRAIN LENGTH SET UP

LR set	**#1**	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**
LRs	10.0	5.0	3.0	1.5	1.0	0.9	0.5	0.4

reasons, the minimum value was 200 iterations. Therefore, *α* had a value in the range of 200 to 2,500 iterations. The larger *α* is better for the more, the more reliability is, but the computation time increases. Conversely, as *α* decreases, the computation time becomes shorter, but the evaluation of the LR at the stage where learning cannot be sufficiently done may reduce the reliability. In this experiment, we examined how the *α* value affects accuracy. The experiment was conducted at four points of 200, 625, 1,250 and 2,500. Also, the *LR set* values used for experiment are shown in **TABLE 2**.

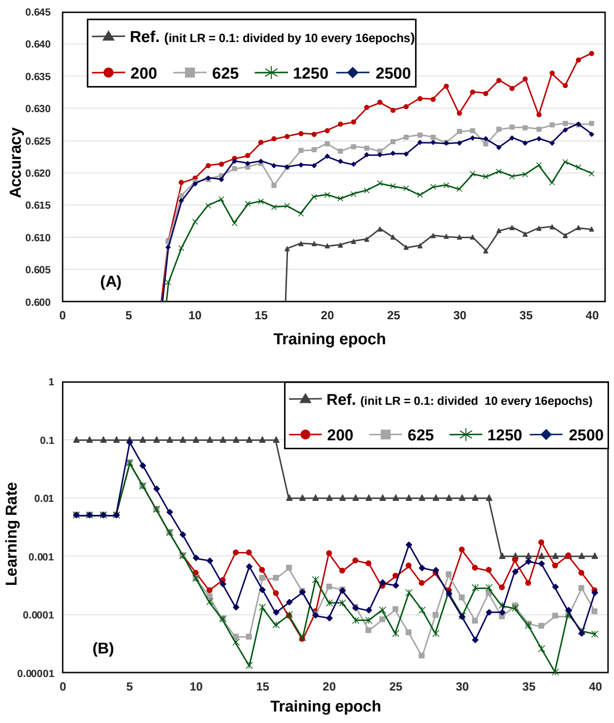

Fig. 5 Reference and proposed method when *α* is 200, 625, 1,250 and 2,500 (A) accuracy and (B) LR.

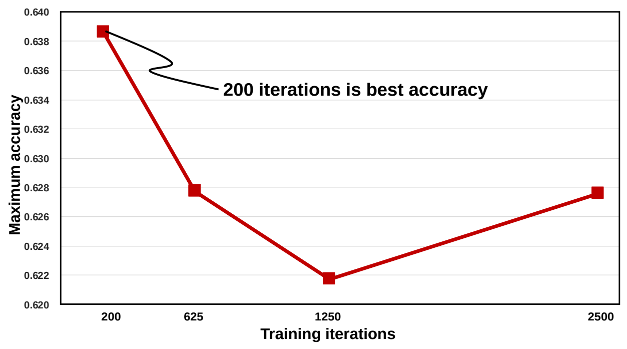

Fig. 6 Maximum accuracy when *α* is 200, 625, 1,250, and 2,500

Fig. **5** shows the transition of accuracy and LR when 40 epoch training was performed using each *α* value. Also, **Fig. 6** shows maximum accuracies for various *α* values. Accuracies achieved with 200 iterations and 1,250 iterations shows that the difference is 1.6 %, which indicates that the number of iterations does not significantly affect the accuracy. Therefore, in this work *α* was set to 200 iterations.

The best accuracy of 0.639 is observed when *α* is 200. When *α* is 625 or 1,250, a smaller accuracy of 0.628 or 0.622 is exhibited; in this case, the accuracies among the LR candidates become closer and averaged out due to the larger iteration. So, they are susceptible to random noises. When *α* is much larger such as 2,500, the impact of the random noise is mitigated, but the distribution of the accuracies among the LR candidates is narrowed. Namely, the smaller value of *α* (α = 200) is the best;. it is effective to take the suitable LR value for training the network model better.

TABLE 3 LR SETS

Values	**(a) Narrow set**	**(b) Middle set**	**(c) Wide set**
#1	1.25	2.50	5.00
#2	1.17	2.00	3.67
#3	1.08	1.50	2.33
#4	1.00	1.00	1.00
#5	0.95	0.85	0.80
#6	0.90	0.70	0.60
#7	0.85	0.55	0.40
#8	0.80	0.40	0.20

TABLE 4 SCHEDULE TO CHANGE LR SET

Epoch	**1–4**	**5–19**	**20–29**	**30–40**
LR set	Warmup	(a) Narrow	(b) Middle	(c) Wide

C. *LR set*

A LR set with multiple values is an important point in the proposed algorithm. In this section, we will examine the method of determining the LR set. Three sets of (a) Narrow set, (b) Middle set, and (c) Wide set are prepared for convergence comparison, as shown in **TABLE 3**.

The number LRs of each *LR set* is eight. The maximum value and the minimum value are determined for each, and the others are set to be evenly spaced around 1.0. Other parameters are the same as those in **TABLE 1**. Fig. **7** and Fig. **8** show the accuracy and LR transition results of 40-epoch learning using (a), (b), and (c).

(a) **Narrow set:** Accuracy is 65.0 %. Although the final accuracy is relatively good, the accuracy slightly declines after epoch 25. As seen in Fig. **8**, there are variations in the transition of the LR after epoch 25. This is due to the fact that the difference between the LRs becomes small and it loses the noise generated at the time of GPGPU computing. As a result, this variation adversely affects the accuracy.

(b) **Middle set:** Accuracy is 63.4 %. The final accuracy is second, but learning has saturated.

(c) **Wide set:** Accuracy is 61.4 %. The final accuracy is the lowest. However, although training is slow, it has advanced even at epoch 40. Fig. **8** suggests that the reason is that the LR is too small at the beginning. However, since the difference between epochs is large, LR can be selected rationally without influence noise even in later epochs.

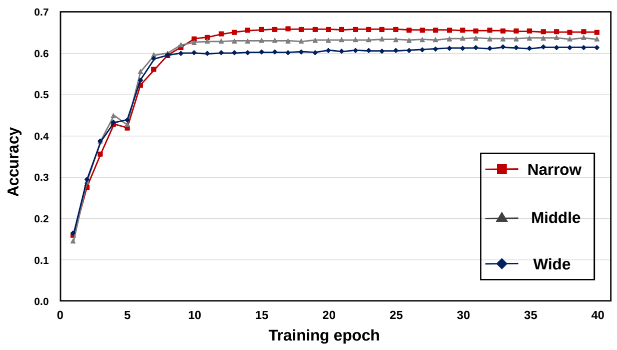

Fig. 7 Accuracy transition using (a) Narrow set, (b) Middle set, and (c) Wide set

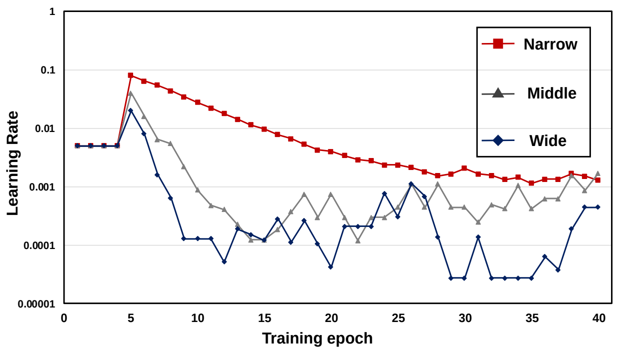

Fig. 8 LR transition using (a) Narrow set, (b) Middle set, and (c) Wide set.

Based on the above results, we combine each LR set and execute with the schedule shown in **TABLE 4**. We call this LR set "Mix". When executed according to the schedule in **TABLE 4**, the shaded area in **Fig. 9** becomes the search area of the LR. The upper limit to 1.0 or less is necessary because the simulation will not operate if the LR becomes too large. **Fig. 10** and **Fig. 11** show the transitions of the accuracy and LR of the execution result. Mix improves by 1.2 % in comparison with (a), which was the most accurate in the LR sets. In (a), deterioration of accuracy was seen after 25 epochs, but Mix was able to prevent it. By changing the LR set in this schedule, we were able to compensate for each disadvantage. Also, Mix is 4.8% better than the reference. Therefore, the proposed adaptive LR adjustment with STPT can improve the accuracy over the conventional method although the computational time of 87.5 % is reduced.

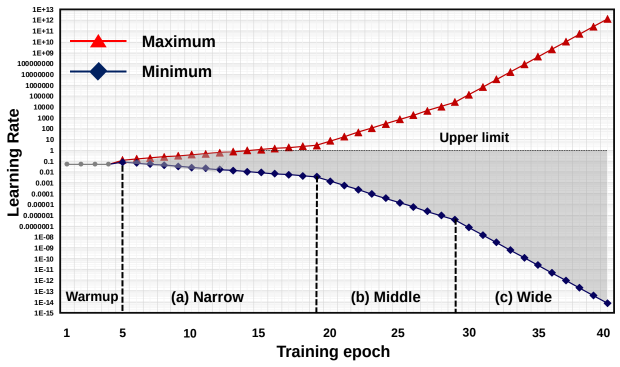

Fig. 9 Search area of the LR when executed according to the schedule in **TABLE 4**.

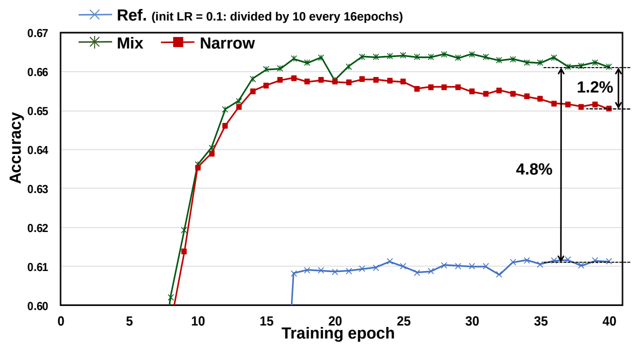

Fig. 10 Accuracy transition of reference and proposed method using Narrow set

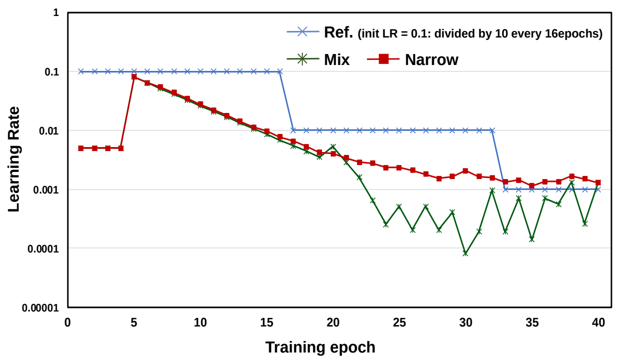

Fig. 11 LR transition of reference and proposed method using Narrow set

Those results show our proposed LR adaptation method is suitable for unknown problems that has been taking much computational time.

IV. Future Work

In future work, we believe that the proposed method can be combined with others methods, such as Adam and AdaGrad. Furthermore, we would like to adapt our algorithm to LARS [21], which make it possible to conduct data parallelism even with a huge batch size.

Acknowledgement

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, L. Fei-Fei and A. C. Berg, “ImageNet Large Scale Visual Recognition Challenge, ” IJCV, vol. 115, pp. 211–252, Dec. 2015.

[2] A. Krizhevsky, I. Sutskerver and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks, ” Neural Information Processing Systems Conference, pp. 1097–1105, Dec. 2012.

[3] “Mastering the game of Go with deep neural networks and tree search, ” *Nature,* vol. 529, pp. 484–489, Jan. 2016.

[4] L. Yann, Y. Bengio and G. Hinton, “Deep learning”, nature, no. 512, pp. 436–444, May 2015.

[5] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV, pp. 818–833, Nov. 2014.

[6] D. G. Lowe, “Distinctive Image Features, ” *International Journal of Computer Vision,* vol. 60, Nov. 2004.

[7] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean and A. Y. Ng, “Building High-level Features, ” arXiv: 1112.6209v5, Jul. 2012.

[8] H. Lee, R. Grosse, R. Ranganth , A. Y. Ng, “Convolutional Deep Belief Networks, ” ICML, pp. 609–616, June 2009.

[9] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Fleep, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang , J. Zhao, “End to End Learning for Self-Driving Cars,” arXiv: 1604.07316v1, Apr. 2016.

[10] T. Okatani, “Deep Learning for Image Recognition, ” Journal of Japanese Artificial Intelligence Society, vol. 28, no. 6, pp. 962–974, Nov. 2013.

[11] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image recognition," Microsoft Research, CVPR, pp. 770–778, June 2015.

[12] K. He, X. Zhang, S. Ren and J. Sun, “Identity Mappings in Deep Residual Networks, ” arXiv: 1603.05027, July 2016.

[13] H. Mori, T. Youkawa, S. Izumi, M. Yoshimoto, H. Kawaguchi, A. Inoue, “A LAYER-BLOCK-WISE PIPELINE, ” IEEE International Workshop on Machine Learing for Signal Processing, pp. 1–6, Dec. 2017.

[14] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh and A. Talwalkar, “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization,” arXiv: 1603.06560, 18 Jun. 2018 .

[15] F. Hutter, H. H. Hoos and K. Leyton-Brown, “Sequential Model-Based Optimization for General Algorithm Configuration,” LION, vol. 6683, pp. 507-523, 2011.

[16] J. Bergstra, R. Bardenet, Y. Bengio and B. Kegl, “Algorithms for hyper-parameter optimization,” Neural Information Processing Systems Conference, Dec. 2011.

[17] D. Kingma , J. Ba, “Adam A Method for Stochastic Optimization,” ICLR, Dec. 2014.

[18] J. Duchi, E. Hazan , Y. Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,”, JMLR, pp2121–2159, July 2011.

[19] M. D.Zeoler, “AdaDelta: An Adaptive Learning Rate Method,”, arXiv:1212.5701, Dec. 2012.

[20] NVIDIA, “cuDNN, ” [Online]. Available: https://developer.nvidia.com/cudnn.

[21] “IMAGENET, ” [Online]. Available: http://www.image-net.org/.

[22] N. Qian, “On the momentum term in gradient descent learning algorithms, ” Neural Networks : The Official Journal of the International Neural Network Society, vol. 12, no. 1, p. 145–151, Jan. 1999.

[23] P. Goyal, P. Dollar, R. Girshick , P. Noordhuis, “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour,” arXiv:1706.02677, June 2017.

[24] P. Networks, “Chainer, ” [Online]. Available: https://chainer.org/.

[25] Y. Yang, G. Igor and G. Boris, “LARGE BATCH TRAINING OF CONVOLUTIONAL NETWORKS, ” arXiv:1708.03888, Aug. 2017.