Structured Pruning for Deep Neural Networks with Adaptive Pruning Rate Derivation Based on Connection Sensitivity and Loss Function

Yasufumi Sakai
Fujitsu Research, Fujitsu Limited, Kawasaki, Japan
Email: sakaiyasufumi@fujitsu.com

Yu Eto and Yuta Teranishi
QNET Group, Fujitsu Limited, Fukuoka, Japan
Email: {eto.yu, teranishi.yuta}@fujitsu.com

***Abstract*—Structured pruning for deep neural networks has been proposed for network model compression. Because earlier structured pruning methods assign pruning rate manually, finding appropriate pruning rate to suppress the degradation of pruned model accuracy is difficult. In this paper, we propose a structured pruning method by deriving pruning rate for each layer adaptively based on gradient and loss function. The proposed method first calculates a threshold of L1-norm of the pruned weight for each layer using loss function and gradient per layer. The threshold guarantees no degradation of the loss function by pruning when the L1-norm of pruned weight is less than that threshold. Then, by comparing the L1-norm of the pruned weight and the calculated threshold while changing the pruning rate, we derive the pruning rate for each layer, which does not degrade the loss function. By applying the derived pruning rate per layer, the accuracy degradation of the pruned model is suppressed. We evaluate the proposed method on CIFAR-10 task with VGG-16 and ResNet in iterative pruning method. For example, our proposed method reduces model parameters of ResNet-56 by 66.3% with 93.71% accuracy.**

***Index Terms*—neural networks, structured pruning, automatic pruning rate search**

I. Introduction

Deep Neural Networks (DNNs) have shown remarkable performance on various tasks such as classification task [1] and semantic segmentation [2]. However, along with improved network performance, DNN models have become deeper and more complex. This trend has limited DNNs deployment on resource-constrained devices such as embedded systems and mobile phones.

Network pruning methods have been proposed to reduce the DNN model size and computational complexity. Earlier work [3] removes the elements of small value weight included in the weight tensor below a threshold determined by a given pruning rate. Because these methods introduce sparse connections, the method is called unstructured pruning. Structured pruning, which prunes with higher granularity like channels and neurons than unstructured pruning, has been studied to reduce computation costs. The presented work [4] removes channels based on the value rank of L1-norm of each filter. Other work [5] derives automatically that channels should be pruned using group LASSO, and prunes filters so that the error of feature maps connected to the output of the pruning target filters is minimized. [6] has proposed a method using gradient for a pruning metric to consider pruning effects for each layer on whole model accuracy. These existing methods prune weight tensor based on the pruning importance like the weight value and a given pruning rate. However, when the wrong pruning rate is set, the accuracy of the pruned model is degraded considerably. In practical cases, many practical applications use various neural networks. Therefore, manual determination of a pruning rate that can suppress pruning-related accuracy degradation for all cases is inefficient.

In this paper, we propose a structured pruning method for deriving the pruning rate for each layer adaptively based on gradient and loss function. Fig. 1 shows an overview of our proposed method. The proposed method first calculates a threshold of L1-norm of pruned weight, i.e. pruning error, for each layer using a loss function and the gradient per layer. The threshold guarantees no degradation of the loss function by pruning when the pruning error is less than that threshold. By comparing both the calculated pruning error and the threshold while changing pruning rate, we derive the pruning rate which does not degrade loss function for each layer. By applying the derived pruning rate per layer, the accuracy degradation of the pruned model is suppressed. We evaluate the proposed method on CIFAR-10 [7] with VGG-16 including batch normalization layer [8] and ResNet [9] in iterative pruning method. The proposed method reduces the model parameters of VGG-16 by 94.4% with 93.43% accuracy. Moreover, our pruned ResNet-32, ResNet-56, and ResNet-100 with 66.9%, 66.3%, and 84.4% reduction of parameters respectively show the accuracy of 92.93%, 93.71%, and 93.60%.

Manuscript received November 8, 2021; revised April 20, 2022.

doi: 10.12720/jait.13.3.295-300

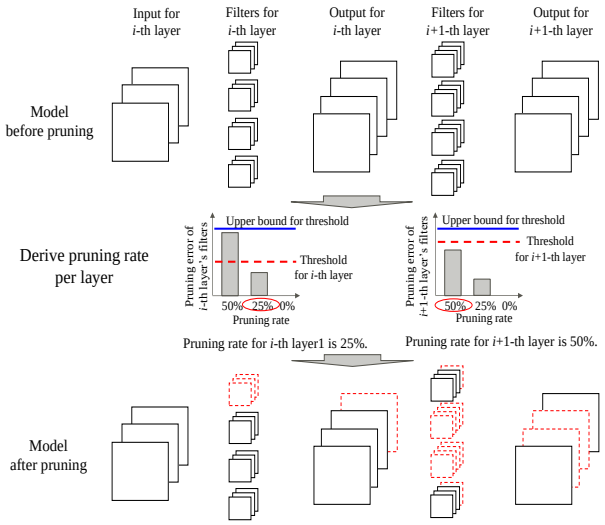


Figure 1. Overview of proposed structured pruning. The proposed method first calculates a threshold of L1-norm of pruned weight, i.e. pruning error, per layer. The threshold guarantees no degradation of the loss function by pruning, when the pruning error is less than the threshold. By comparing that calculated pruning error and the threshold while changing the pruning rate, the pruning rate per layer which does not degrade the loss function is derived.

Our contributions for structured network pruning are the following.

- We propose a structured pruning method deriving the pruning rate per layer adaptively to compress the network by removing whole model channels and neurons.
- We demonstrate that our proposed method can reduce model parameters and number of Floating Point Operations (FLOPs) on CIFAR-10 with VGG-16, ResNet-32, ResNet-56, and ResNet-110 with no marked accuracy degradation.

II. Related Works

A. Importance for Pruning

Structured pruning generally calculates the importance in unit of higher granularity like channels. Then, the channels with low importance below a threshold which is determined by pruning rate are pruned. The weight value is used widely for importance [3], [4]. Feature maps error [5] and scaling factor of batch normalization layer [10] are also often used for importance. However, these importance do not consider the network accuracy effects. Although [6] uses sensitivity of the loss function like gradient for importance, using sensitivity for pruning shows lower performance than that obtained by using weight value [11].

B. Local Pruning and Global Pruning

Two methods are available for setting the importance and pruning rate. Local pruning sets the rank of importance and the pruning rate per layer [3]. Because the combinations of pruning rates are extremely numerous in local pruning, finding a combination that suppresses accuracy degradation is difficult. Another method, global pruning, sets only one pruning rate for the whole model and calculates the rank of importance for the whole model [12]. Therefore, the pruning rate per layer is generally different in global pruning.

Algorithm 1: Proposed pruning method

Input: Pre-trained model
Pruning rate candidates: $Rate_{cand}$ = [20%, 10%, 0%]
Loss margin control parameter L_m
Accuracy control parameter Acc_{ctrl}
Output: Derived pruning rate

1: Calculate Acc_b using the pre-trained model.
2: Calculate loss function and gradient by pre-trained model.
3: *Target model* ← pre-trained model
4: Initialize upper bound U_b and upper bound limit $U_{b,limit}$
5: **while** All pruning rates for each layer are 0%. **do**
6: Calculate threshold $T_{h,k}$ by Eq. (6)
7: **if** L2-norm of $T_{h,k,all} > U_b$ **then** # Limit $T_{h,k}$
8: Scale all $T_{h,k}$ so that L2-norm $T_{h,k,all}$ equals U_b.
9: **end if**
10: Derive pruning rate per layer by Algorithm 2.
11: Prune *Target model* with derived pruning rate.
12: Fine-tune pruned *Target model*.
13: Calculate Acc_p by fine-tuned model.
14: **if** $Acc_p + Acc_{ctrl} \geq Acc_b$ **then**
15: *Target model* ← fine-tuned model
16: Calculate loss function and gradient by the *Target model*.
17: $U_b = \min(2 \times U_b , U_{b,limit})$
18: **else**
19: $U_b = 0.5 \times U_b$
20: **end if**
21: **end while**
22: **return** Derived pruning rate

C. Pruning Scheduling

To suppress accuracy degradation due to pruning, the pruned model is generally fine-tuned. One-shot pruning method prunes the model by a high pruning rate only once; it fine-tunes the pruned model [13]. Iterative pruning method iteratively executes the procedure of pruning the model with a low pruning rate and fine-tuning [4], [14]. Earlier work [12] reports that the accuracy degradation of a pruned model using iterative pruning is less than that of one-shot pruning.

Our proposed method derives the pruning rate for each layer like local pruning, whereas the final pruning rate per layer differs similarly to global pruning. That is true because the proposed method derives the pruning rate for each layer adaptively depending on the gradient of each layer. The proposed method uses the gradient for deriving the pruning rate to consider the effects of pruning for loss function. To suppress the accuracy degradation by pruning, the proposed method uses weight value for pruning importance with the iterative pruning method.

III. Methodology

The procedure used for our proposed pruning method is shown in Algorithm 1. The proposed method prunes the pre-trained model. First, proposed method derives the pruning rate and prunes the model with the derived pruning rate. After fine-tuning the pruned model, the proposed method is decided whether or not to adopt the derived pruning rate, by comparing the accuracy of the fine-tuned model and the pre-trained model. The procedure of pruning rate derivation, model pruning, fine-tuning, and

accuracy comparison is iteratively executed until the pruning rate of all layers are chosen to 0%.

Algorithm 2: Pruning rate derivation

Input: Pruning target tensor W
Threshold T_h
Pruning rate candidates: $Rate_{cand}$ = [20%, 10%, 0%]
Output: Derived pruning rate

```
i = 0
Rate_tmp = Rate_cand[i] # Set maximum pruning rate of Rate_cand.
Calculate the pruning error ΔW with Rate_tmp.
while i ≤ number of Rate_cand do
  if ΔW ≤ T_h then
    break
  else
    i = i + 1
    Rate_tmp = Rate_cand[i] # Set pruning rate as one lower
                              than last iteration.
    Calculate pruning error ΔW with Rate_tmp.
  end if
end while
return Rate_tmp
```

A. Pruning Rate Derivation

The ideal pruning rate prevents accuracy degradation due to pruning. If the ideal pruning rate is obtained, then it is expected that a loss function of the model pruned by the ideal pruning rate is also not degraded. We set the following constraint from this ideal situation. *The loss function derived from the pruned model is less than or equal to the loss function derived from the model before pruning.* When only one element including the pruning target tensor is removed, the constraint above is presented in (1).

$$L(w_1 + \Delta w_1, w_2, \cdots, w_n) \leq L_b \tag{1}$$

where w_i is the i-th element of a pruning target tensor. i =1,2,…,n are indexes of the elements. n is the number of elements of that tensor. $\Delta w_i = -w_i$ is the pruning error, i.e. the difference between weight value before and after pruning of the i-th element. $L(w_1 + \Delta w_1, w_2, \cdots, w_n)$ is a loss function of the pruned model and L_b is a loss function of the model before pruning. To apply first-order Taylor expansion to the loss function of the pruned model, we can derive (2).

$$\begin{aligned} &L(w_1 + \Delta w_1, w_2, \cdots, w_n) \\ &\approx L(W) + \partial L(W)/\partial w_1 \cdot \Delta w_1 \\ &\leq L(W) + |\partial L(W)/\partial w_1| \cdot |\Delta w_1| \leq L_b \end{aligned} \tag{2}$$

In this expression, $W = [w_1, w_2, \cdots, w_n]^{\mathrm{T}}$. Because the value of $L(W)$ is the same as that of L_b, (2) can not be satisfied. To satisfy (2), we relax the constraint by introducing a margin of loss function. *The loss function derived from the pruned model is less than or equal to the loss function derived from the model before pruning "with a margin".* We modify (2) according to the relaxed constraint.

$$L(W) + |\partial L(W)/\partial w_1| \cdot |\Delta w_1| \leq L_b + L_b \cdot L_m \tag{3}$$

where $L_b \cdot L_m$ is a margin of the loss function introduced by the relaxed constraint and L_m is a control parameter of the loss function margin. Although L_m is the hyperparameter, since the pruning rate derived by proposed method is finally determined by accuracy comparison, the final derived pruning rate is robust against L_m. We see the robustness against L_m in experiments (Sec. 4). Transforming (3) derives the upper bound of the pruning error Δw_1, which satisfies the constraint.

$$|\Delta w_1| \leq \frac{L_b + L_b \cdot L_m - L(W)}{|\partial L(W)/\partial w_1|} \tag{4}$$

Using (4), we can decide that w_1 can be pruned or not in terms of degradation of loss function. We expand (4) to (5) for larger granularity pruning such as channel pruning.

$$\begin{aligned} \Delta W &= \frac{1}{n} \cdot \sum_{i=1}^{n} |\Delta w_i| \\ &\leq \frac{L_b + L_b \cdot L_m - L(W)}{n} \cdot \sum_{i=1}^{n} \frac{1}{|\partial L(W)/\partial w_i|} = T_h \end{aligned} \tag{5}$$

The left side of (5), ΔW, is the L1-norm of pruned weights normalized by the number of elements, i.e. ΔW is a pruning error calculated with high granularity. When pruning a tensor, as the number of pruned weights is determined by the pruning rate, the value of L1-norm with pruned weights depends on the pruning rate. The right side of (5), which is upper bound of pruning error, i.e. threshold T_h, is independent of the pruning rate. Moreover, since gradients are derived independently per layer, threshold T_h is also derived independently per layer. Therefore, using (5), a pruning rate that suppresses the loss function degradation can be found independently for each layer by setting up a pruning rate candidate $Rate_{cand}$ and by finding a pruning rate from that candidate such that pruning error ΔW is less than or equal to threshold T_h.

Iterative pruning is one method for suppressing accuracy degradation due to pruning. The proposed method iteratively executes model pruning with the derived pruning rate by (5) and the pruned model fine-tuning. Equation (6) shows the threshold at k-times during iterative pruning: $T_{h,k}$.

$$T_{h,k} = \frac{L_b + L_b \cdot L_m - L(W_k)}{n} \cdot \sum_{i=1}^{n} \frac{1}{|\partial L(W_k)/\partial w_i|} \tag{6}$$

where W_k is the weight tensor of the pruned model after searching for pruning rate k times. When k=0, because the pruning rate have not been sought yet, W_0 is same as the weight tensor of the model before pruning. In the proposed pruning rate derivation method shown in Algorithm 2, the pruning rate for structured pruning can be derived by calculating the pruning error and threshold at filter and neuron granularity. Furthermore, the proposed method can derive the pruning rate for unstructured pruning by calculation at element granularity.

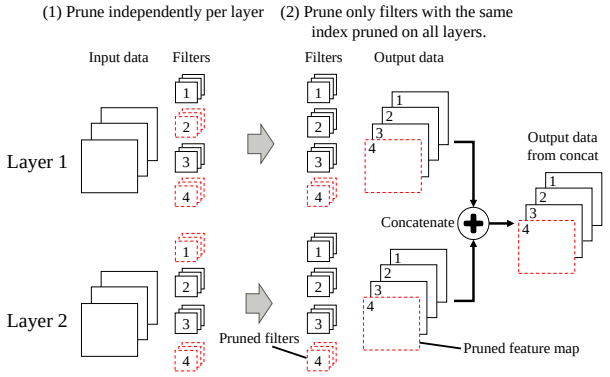


Figure 2. Pruning for shortcut connection.

Because the threshold $T_{h,k}$ includes approximation error derived from Taylor expansion, the possibility exists that the pruning rate derived by (6) is incorrect. To reduce approximation error effects, we introduce the trust region method, which is an optimization method using the parameter including error. When using the trust region method, to avoid incorrect optimization by the error, the upper bound, called trust radius, limit the value of the parameter including error. In our proposed method, thresholds for all layers are limited by U_b shown in (7).

$$\left\| T_{h,k,all} \right\|_2 \leq U_b \tag{7}$$

here, $T_{h,k,all} = [T_{h,k,1}, T_{h,k,2},\ldots, T_{h,k,m}]$ is a vector of the threshold. m represents the number of the pruning target tensor. $T_{h,k,i}$ (i=1,2,…,m) is the threshold for each pruning target tensor.

B. Model Pruning and Fine-Tuning

After deriving the pruning rate, the pruning target model is pruned by the derived pruning rate. The pruning importance is L1-norm of weight value. Weights with low importance below a threshold determined by a given pruning rate are removed. In fine-tuning, the pruned model reuses the weights of the latest fine-tuned model. In the case of first search, the pruned model reuses the weights of the pre-trained model.

C. Accuracy Comparison

To suppress the accuracy degradation of the pruned model and achieve the desired accuracy, the pruned model is fine-tuned until the accuracy of pruned model Acc_p plus the accuracy control parameter Acc_{ctrl} exceeds the accuracy of pre-trained model Acc_b. When Acc_p+Acc_{ctrl} exceeds Acc_b, the derived pruning rates are adopted. When Acc_p+Acc_{ctrl} does not exceed Acc_b, the derived pruning rates are discarded.

When executing the trust region method iteratively, the upper bound, which limits the value of parameter including error, is updated every iteration depending on the optimization result because the upper bound affects the convergence of optimization. In the proposed method, the upper bound U_b is increased when Acc_p+Acc_{ctrl} exceeds Acc_b, whereas U_b is decreased when Acc_p+Acc_{ctrl} does not exceed Acc_b.

D. Pruning Search Ternimation Condition

The proposed method iteratively executes pruning rate derivation, model pruning, fine-tuning, and accuracy comparison until the pruning rates of all layers are chosen as 0%.

E. Setting for the Upper Bound of Threshold

When the initial value of upper bound U_b is small, the threshold becomes set to a small value at the first search. Thus, the possibility exists that the pruning rate of all layers becomes 0%. The pruning rate search might be terminated at the first search. To avoid termination at the first search, the initial value of U_b is set as L2-norm of the pruning error of all layers with the maximum pruning rate of pruning rate candidates $Rate_{cand}$. Because the degree of freedom for pruning rate selection in all layers is guaranteed when U_b is set to that value of L2-norm, from equation (7).

The proposed method updates the upper bound U_b to a larger value depending on the accuracy comparison result. However, when the U_b value is extremely large, convergence of the optimization takes a long time. We set a initial value of U_b to the upper limit of upper bound $U_{b,limit}$.

F. Pruning for Shortcut Connection

A shortcut connection with concatenate block is often used for DNN such as ResNet. The number of multiple input feature maps must be equal for a concatenate block. To align the number of input feature maps, we prune only those filters with the same index pruned on all layers, as presented in Fig. 2.

IV. Experiments

We evaluated the proposed pruning method on CIFAR-10 with VGG-16 including batch normalization layer, ResNet-32, ResNet-56, and ResNet-110. We used pre-trained model by Kim's model [15] for VGG-16 and by Idelbayev's model [16] for ResNet. The pruning rate candidates $Rate_{cand}$ were 10% and 0%. The evaluated models were retrained by momentum SGD with L2 weight decay. A mini-batch size was 128. The momentum coefficient was 0.9. The coefficient of weight decay was 10^{-4}. The retraining duration was 300 epochs. The initial learning rate was 0.1. The learning rate was multiplied by 0.1 at epoch 100, 150, and 200. We implemented the proposed method on Pytorch. Our code is available in https://github.com/FujitsuLaboratories/CAC/tree/main/cac/pruning.

A. Performance for Hyperparameters

Fig. 3(a) shows the relation between pruned model performance and loss margin control parameter L_m with VGG-16. When L_m is zero, the compression ratio becomes 0%. The initial threshold for each layer also becomes zero. Therefore, all searched pruning rates at the first search become 0%. When L_m is greater than zero, because the choice of L_m does not much affect the compression ratio, which indicates the robustness of the L_m choice. We used L_m = 0.1 for the following experiments.

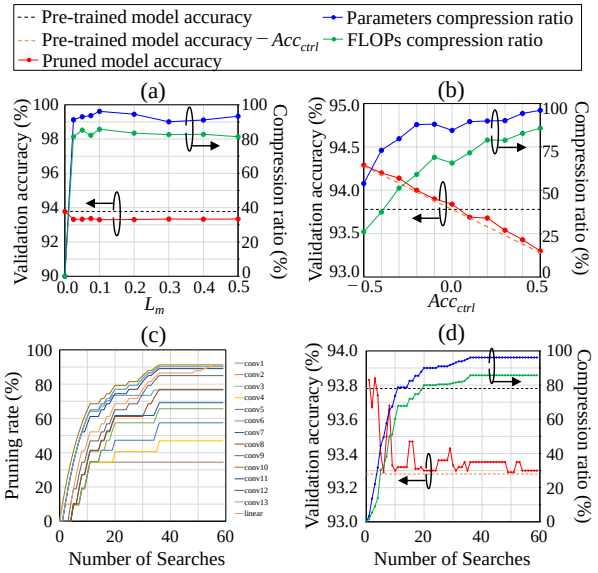


Figure 3. Performance of pruned model on CIFAR-10 with VGG-16: (a) performance on L_m (Acc_{ctrl} = 0.5%) and (b) performance on Acc_{ctrl} (L_m = 0.1). (c) Pruning rate per layer during pruning rate search and (d) validation accuracy and compression ratio for parameters and FLOPs during pruning rate search. In (c) and (d), L_m =0.1, Acc_{ctrl}=0.5%, respectively.

Fig. 3(b) shows the relation between pruned model performance and accuracy control parameter Acc_{ctrl} with VGG-16. Because the pruned model accuracies exceed the value of the pre-trained model accuracy subtracted by Acc_{ctrl} for all values of Acc_{ctrl}, the proposed method can control the pruned model accuracy by Acc_{ctrl}. The result, which is that the compression ratio increases as the accuracy is reduced, is consistent with intuition and earlier reports [12], [14].

B. Performance during Pruning Rate Search

Fig. 3(c) shows the pruning rate per layer during the pruning rate search with VGG-16. We set Acc_{ctrl} as 0.5%. In the proposed method, because the pruning rate for each layer is derived adaptively using the threshold and pruning error calculated independently per layer, the final pruning rate for each layer is set to different value similarly to global pruning.

The compression ratio during pruning rate search, as shown in Fig. 3(d), is increased immediately during initial iterations. Then the compression ratio converges gradually until 0% is set as the pruning rate for each layer. It seems that the compression ratio increases immediately at the beginning of a search and then slows is the model has many redundant parts at the beginning of the search, but only few redundant parts in the latter part of the search.

The pruned model accuracy is maintained as determined by Acc_{ctrl} during the pruning rate search. Therefore, when we set the target model size by terminating the search when the pruned model size reaches the target, we can get the pruned model, which guaranteed the accuracy and model size in a short time.

C. Performance Comparisons

Table I presents comparisons with earlier works. Results showed that our method achieves the best compression rate of parameters and FLOPs under the similar pruned model accuracy for all evaluated models. Moreover, our pruned ResNet-56 and ResNet-110 achieve the remarkable level of pruned model accuracy of 94.03% and 94.60% under a similar compression ratio of parameters to those used for earlier studies.

TABLE I. COMPARISON OF THE PRUNED NETWORKS WITH DIFFERENT METHODS ON CIFAR-10. 'BASELINE ACC.' AND 'PRUNED ACC.' DENOTE THE VALIDATION ACCURACY OF THE PRE-TRAINED AND PRUNED NETWORKS. 'PARAMS. COMPRESS.' AND 'FLOPS COMPRESS.' ARE COMPRESSION RATIO OF PARAMETERS AND FLOPS OF NETWORKS.

Model	Method	Baseline acc. (%)	Pruned acc. (%)	Params. compress. (%)	FLOPs compress. (%)
VGG-16	COP (2019) [13]	93.56	93.31	92.8	73.5
	PaG (2019) [4]	93.25	93.41	86.1	N/A
	SNIP (2019) [6]	93.18	92.73	95.0	N/A
	NM (2020) [14]	93.70	93.16	63.7	N/A
	CCP (2021) [10]	93.80	93.39	94.9	73.7
	Ours (Acc_{ctrl}=0.5)	93.78	93.30	**96.2**	**85.7**
	Ours (Acc_{ctrl}=0.4)	93.78	**93.43**	94.4	82.9
ResNet-32	COP (2019) [13]	92.64	91.97	57.5	53.9
	SCOP (2020) [17]	92.66	92.13	56.2	55.8
	CCP (2021) [10]	93.26	92.57	34.8	57.6
	Ours (Acc_{ctrl}=−0.3)	92.63	92.93	**66.9**	**60.3**
	Ours (Acc_{ctrl}=−0.4)	92.63	**93.05**	55.4	53.8
ResNet-56	PaG (2019) [4]	93.04	93.12	47.9	N/A
	NM (2020) [14]	93.70	93.64	56.3	56.0
	CCP (2021) [10]	93.59	93.04	58.8	73.5
	Ours (Acc_{ctrl}=−0.3)	93.39	93.71	**66.3**	**65.4**
	Ours (Acc_{ctrl}=−0.6)	93.39	**94.03**	63.4	59.1
ResNet-110	PaG (2019) [4]	93.53	93.56	53.1	N/A
	CCP (2021) [10]	94.11	93.36	64.2	68.0
	Ours (Acc_{ctrl}=0.1)	93.68	93.60	**84.4**	**79.5**
	Ours (Acc_{ctrl}=−0.9)	93.68	**94.60**	67.6	68.6

V. CONCLUSION

In this paper, we proposed a structured pruning method with adaptive pruning rate derivation. To suppress accuracy degradation by pruning, the pruning rate for each layer is derived based on the gradient per layer and whole the model loss function. By applying the proposed pruning method to VGG-16, ResNet-32, ResNet-56, and ResNet-110, parameters and FLOPs of the pruned models can be reduced considerably with no marked accuracy degradation. In our future work, we will apply the proposed method to other tasks and models such as ImageNet task [17], transformer [18], and BERT [19].

CONFLICT OF INTEREST

The authors declare no conflicts of interest associated with this manuscript.

AUTHOR CONTRIBUTIONS

Yasufumi Sakai mainly conducted this research; Yasufumi Sakai, Yu Eto, and Yuta Teranishi implemented the proposed algorithm; Yasufumi Sakai carried out the experiment and wrote the paper; all authors had approved the final version.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097-1105, 2012.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431-3440.

[3] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.

[4] A. Salama, *et al.*, "Pruning at a glance: Global neural pruning for model compression," arXiv preprint arXiv:1912.00200, 2019.

[5] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. the IEEE International Conference on Computer Vision*, 2017. pp. 1389-1397.

[6] N. Lee, T. Ajanthan, and P. H. S. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," arXiv preprint arXiv:1810.02340, 2018.

[7] A. Krizhevsky and G. E. Hinton. (2019). Learning multiple layers of features from tiny images. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.9220&rep=rep1&type=pdf

[8] H. Li, *et al.*, "Pruning filters for efficient convnets," arXiv preprint arXiv:1608.08710, 2016.

[9] K. He, *et al.*, "Deep residual learning for image recognition," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 770-778.

[10] Y. Chen, *et al.*, "CCPrune: Collaborative channel pruning for learning compact convolutional networks," *Neurocomputing*, vol. 451, pp. 35-45, 2021.

[11] D. Blalock, *et al.*, "What is the state of neural network pruning?" arXiv preprint arXiv:2003.03033, 2020.

[12] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," arXiv preprint arXiv:1803.03635, 2018.

[13] W. Wang, *et al.*, "COP: Customized deep model compression via regularized correlation-based filter-level pruning," arXiv preprint arXiv:1906.10337, 2019.

[14] W. Kim, *et al.*, "Neuron merging: Compensating for pruned neurons," arXiv preprint arXiv:2010.13160, 2020.

[15] Pruning Filters for Efficient ConvNets. [Online]. Available: https://github.com/tyui592/Pruning_filters_for_efficient_convnets

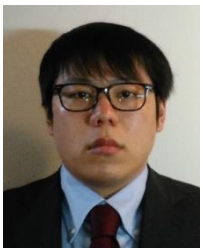
[16] Proper ResNet Implementation for CIFAR10/CIFAR100 in Pytorch. [Online]. Available: https://github.com/akamaster/pytorch_resnet_cifar10

[17] Y. Tang, *et al.*, "SCOP: Scientific control for reliable neural network pruning," arXiv preprint arXiv:2010.10732, 2020.

[18] A. Vaswani, *et al.*, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998-6008, 2017.

[19] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv: 1810.04805, 2018.

Yasufumi Sakai received the B.E. and M.E. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 2005 and 2007 respectively. In 2007, he joined Fujitsu Laboratories Limited, Kawasaki, Japan, where he engaged in the research and design of CMOS analog, RF circuits, and high-speed CMOS interconnect circuits until 2017. From 2017 to 2018, he was a Visiting Scholar with the University of California, San Diego, La Jolla CA, where he researched low-power neuromorphic computer system. In 2018, he was back to Fujitsu Laboratories, and engaged in research of efficient training methods for neural networks. In 2021, he moved to Fujitsu Limited, Kawasaki, Japan, and he has been engaged in research and development of efficient training and inference methods for neural networks.



Yu Eto received the B.S. degree from Kyushu University, Fukuoka, Japan in 2010 and the M.E. degree from Osaka University, Suita, Japan in 2014. In 2014, he joined Fujitsu Kyushu Network Technologies Limited, Fukuoka Japan, where he has been engaged in research and development of optical networks, robotics and machine learning. In 2021, he transferred to Fujitsu Limited, Fukuoka, Japan. His current work is data analysis for factories.

Yuta Teranishi received the B.S. and M.S. degrees in statistics from Kyushu University, Fukuoka, Japan, in 2009 and 2011 respectively. In 2011, he joined Fujitsu Kyushu Network Technologies Limited, Fukuoka, Japan. Since then, he has belonged to Fujitsu Laboratories Limited, Fujitsu Kyushu Network Technologies Limited, and now to Fujitsu Limited, Fukuoka, Japan. He has been involved in various tasks such as research and development of energy management systems, development of wireless base stations, and research and development of explainable AI technologies. Currently, he is engaged in research and development of pruning of neural networks technologies, and support for the conversion of factories into smart factories.