Rethinking Group Fisher Pruning for Efficient Label-Free Network Compression

Jong-Ryul Lee[1]
jongryul.lee@etri.re.kr
Yong-Hyuk Moon[†1,2]
yhmoon@etri.re.kr

[1] Artificial Intelligence Research Laboratory
Electronics and Telecommunications Research Institute (ETRI)
Daejeon, Korea

[2] University of Science and Technology (UST)
Daejeon, Korea

Abstract

Group Fisher Pruning is a powerful gradient-based channel pruning method for convolutional neural networks. Even though it provides excellent convenience for allocating sparsity over layers with strong effectiveness, the cost of its pruning process is significantly expensive for large neural networks. In addition, it was proposed to handle neural networks having residual connections, but it still cannot handle concatenation-type connections like DenseNet. These drawbacks make Group Fisher Pruning hard to be utilized for somewhat large or complex neural networks. Motivated by them, we propose an improved method based on Group Fisher Pruning for efficiency and applicability. For efficiency, we parameterize the number of pruned channels at each pruning step and demonstrate that it can be a much larger value than one. We devise a formal algorithm for applicability to prune DenseNet-style neural networks. In addition, we devise a knowledge distillation-based channel importance scoring scheme that can work for label-free channel pruning, which is a crucial task for exploiting unlabeled data from edge devices. To demonstrate the superiority of our method, we conduct extensive experiments dealing with label-free channel pruning. Our method prunes neural networks at most two orders of magnitude faster than Group Fisher Pruning with comparable accuracy. It should be noticed that our method does not require any label for pruning and retraining while Group Fisher Pruning does.

1 Introduction

Finding efficient neural networks while minimizing accuracy loss has been one of the fundamental topics in deep learning. This topic is especially essential to deploy neural network models into various edge devices with limited resources, and there are many recent works for handling it [4, 9, 12, 15, 16, 18, 26, 28].

Group Fisher Pruning is an effective channel pruning method in this line of research [12]. It selects channels to prune based on their gradients with training data and normalization

† Corresponding author.

in terms of FLOPs or memory reduction. Due to this feature, we can compare channels globally across layers so that we do not need to struggle to additionally find the optimal layer-wise sparsity. In addition, Group Fisher Pruning is the first method to be designed to handle neural networks containing residual connections like ResNet programmatically. Despite these merits, Group Fisher Pruning suffers from expensive pruning costs because it prunes a channel one by one with accumulated gradients through several training iterations. Thus, it would not be preferred if we wanted to compress relatively large neural networks.

This paper mainly focuses on handling label-free channel pruning, which is an essential task for on-device AI because most data gathered from edge devices are unlabeled. There are many existing works [13, 23, 27] exploiting such unlabeled data to improve the accuracy of a model. Similarly, we want to exploit unlabeled data for pruned models and see how much they affect the accuracy.

Contributions. This paper aims to improve Group Fisher Pruning in terms of efficiency and applicability for label-free channel pruning. The contributions are as follows.

- We propose an algorithm for handling DenseNet-style skip connections for channel pruning. Especially, the algorithm correctly finds convolutional layers sharing coupled output channels through residual connections and DenseNet-style skip connections. The supplemental material provides the formal algorithm description for space limitation.

- We first introduce a simple parameterization technique for reducing the pruning costs of Group Fisher Pruning. We formulate how to exploit knowledge distillation for Group Fisher Pruning for effective label-free channel pruning. In addition, we propose an effective selection method for positions where distillation occurs.

- We conduct extensive experiments with large and small image classification tasks and neural network models, each of which has residual or skip connections. The proposed method outperforms Group Fisher Pruning in terms of pruning costs with comparable compression performance, even if the ground truth labels are unavailable.

2 Related Work

Channel Pruning. Channel pruning is one of the powerful concepts to achieve a lightweight neural network model from a large and effective model. Most existing works for channel pruning did not focus on adequately handling coupled output channels by residual connections [14, 18, 19, 24, 31]. Luo and Wu [15] discovered the importance of pruning coupled channels by residual connections and proposed an effective way of pruning them. However, they did not propose how to find such coupled channels in a programmatic way. Liu et al. [17] proposed a grouping method to find layers sharing coupled output channels. In addition to it, they proposed an effective method to prune coupled channels with gradient-based channel importance. Since these two methods are designed to handle residual connections, we will compare our method with them in the experiments.

Few Sample Distillation. For label-free pruning and retraining, block-level knowledge distillation methods were proposed [10, 20]. These methods deal with blocks, each of which contains multiple layers for label-free pruning like ours, but they do not deal with how to make blocks from a model. In addition, even if they can be used for retraining a compressed model by our method, we did not use them because we wanted to focus attentively on the effectiveness of the layer-level topology.

3 Algorithm

3.1 Background

Our pruning method consists of three steps, as depicted in Figure 1. Given a neural network model, the method first adds a gate layer having channel-wise gates after each convolutional layer. If the i-th masking value of a gate layer becomes 0, the values on the i-th output channel of its corresponding convolutional layer are suppressed to zero together. Note that the gate layer after the batch normalization (BN) is the same as that of the convolutional layer (Conv). The gate layer for BN is needed to prevent non-zero values on masked channels occurring due to the shift variables of BN. Next, our method calculates a masking vector for every gate layer as channel pruning. Then, the method retrains the model with gate layers. Note that gates are not trained in retraining because they are not trainable. Finally, the method prunes the model with the masking vectors.

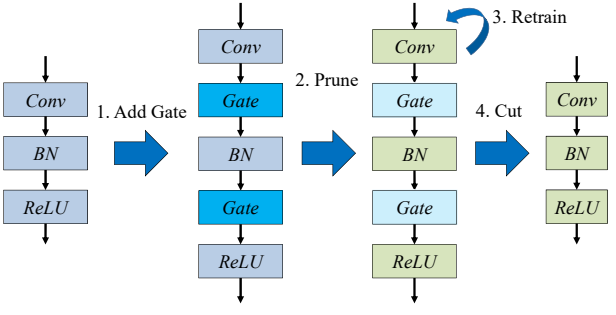


Figure 1: Our Pruning Procedure with Gate Layers.

Handling Skip Connections. Our strategy for handling skip connections is to find convolutional layers which share coupled output channels and to manage them through their gate layers. Once we find such convolutional layers and their gate layers, it is easy to make them share the same gating vector. Liu et al. proposed an algorithm to find such layers in [12] for Group Fisher Pruning, but it cannot handle DenseNet-type skip connections. For example, in Figure 2, the output channels of C_1 are coupled with those of C_2 due to the addition operation. On the other hand, the output channels of C_1 are not coupled with those of C_3 even if they are merged into a single operation named concatenation. The finding algorithm of Liu et al. cannot distinguish these two cases.

To resolve this issue, we propose an algorithm to handle them in a formal way based on the layer-level topology of a neural network. The detailed algorithm is omitted for space limitation. Instead, it is provided in the supplemental material.

3.2 Review on Group Fisher Pruning

Let us review the channel pruning method, Group Fisher Pruning, proposed in [12]. Given N training samples and a single convolutional layer C having d out-channels, we define a masking (gating) vector $\mathbf{m} \in \mathbb{R}^d$. We can emulate channel pruning by multiplying the masking vector to the output of C. Initially, $\mathbf{m}$ is the all-ones vector. For the i-th output

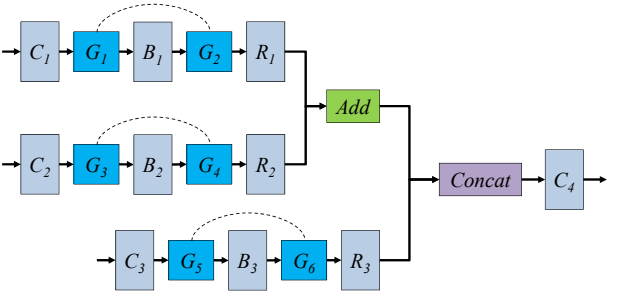


Figure 2: Example of Skip Connections. (C_i represents a convolutional layer, G_i represents a channel-wise gate layer, B_i represents batch normalization, and R_i represents a ReLU layer. The gate layers connected with a dotted line are actually the same gate layer.)

channel of C, the channel importance score s_i is defined as,

$$s_i = \mathcal{L}(\mathbf{m} - \mathbf{e}_i) - \mathcal{L}(\mathbf{m}) \tag{1}$$

$$\approx \frac{1}{2N} \sum_{n=1}^{N} \left(\frac{\partial \mathcal{L}_n}{\partial \mathbf{m}_i} \right)^2 \propto \sum_{n=1}^{N} \left(\frac{\partial \mathcal{L}_n}{\partial \mathbf{m}_i} \right)^2, \tag{2}$$

where n is the n-th example, $\mathbf{m}_i$ is the i-th element of $\mathbf{m}$, $\mathbf{e}_i \in \mathbb{R}^d$ is the one-hot vector whose the i-element is one, and $\mathcal{L}$ is the training loss with the mask computed with $\mathbf{m}$ and $\mathbf{e}_i$. The gradient-based channel importance score is derived by approximating the loss change, which is caused by pruning a channel, with Taylor expansion and Fisher information. The detailed derivation is provided in [12].

Then, let us discuss how to prune coupled output channels with a group $\mathbf{G}$ of convolutional layers. In this case, suppose that the masking vector $\mathbf{m}$ is shared by the convolutional layers. For the i-th output channel shared by $\mathbf{G}$, the importance score s_i is calculated as,

$$s_i \propto \sum_{n=1}^{N} \left(\frac{\partial \mathcal{L}_n}{\partial \mathbf{m}_i} \right)^2 = \sum_{n=1}^{N} \left(\sum_{G \in \mathbf{G}} \frac{\partial \mathcal{L}_n}{\partial \mathbf{m}_i^G} \right)^2, \tag{3}$$

where $\frac{\partial \mathcal{L}_n}{\partial \mathbf{m}_i^G}$ is the gradient on $\mathbf{m}_i$ toward each layer $G \in \mathbf{G}$. Based on this importance scoring method, the pruning method in [12] collects gradients for several iterations and then prunes the least important channel by assigning 0 to its associated masking vector. That is, $\mathbf{m}$ is iteratively updated as $\mathbf{m} := \mathbf{m} - \mathbf{e}_i$ after pruning the i-th channel. This procedure is repeated until achieving a target sparsity.

Normalization. Group Fisher Pruning includes a normalization technique on the importance score function with two factors: FLOPs and memory reduction. In this work, we use memory reduction-based normalization.

Applicability to Connection Types. The importance score function of Group Fisher Pruning can be used for neural network models having residual connections or concatenation-type connections. On the other hand, it is not applicable to sparse convolution skip connections proposed in [21] without any change. For handling such neuron-level skip connections, we may modify the score function to work with gradients on weights, but it is out of scope in this paper.

Make It Efficient. We use the same importance score s_i of [12] without any change. For efficiency, we modify the pruning procedure described in [12]. For each pruning step, instead of removing a single channel one by one, the top-k least important channels are pruned together (i.e., $k = 1$ in [12]). With this simple parameterization, the method still has good pruning performance while providing sweet efficiency.

3.3 Toward Effective Label-Free Channel Pruning

It should be noticed that the gradient-based importance scoring method can take any loss function. That is, we can consider it as a framework to connect the network (task) loss and the importance of each output channel of a layer in context of pruning channels. We expect that if an effective loss is given as the network loss, the pruning performance can be improved through the framework.

Motivated by this, we propose a new loss function as the network loss. Let us consider the original network M, and we are applying Group Fisher Pruning to M at the j-th iteration. We denote the pruned neural network of M after the j-th iteration by M_j. The idea is that the loss function is defined for the difference between the outputs of intermediate convolutional layers at the same positions of M and M_j. Suppose that we have a set $\mathbf{L}$ of specific convolutional layers in M and the output tensor of a layer L with the i-th example is denoted by $\mathcal{X}_i^L$. We call such layers anchor layers. Then, the loss for pruning is defined as,

$$\mathcal{L}_{\text{KL}} = \frac{1}{N}\sum_{n=1}^{N} -\mathbf{p}_i \log \frac{\mathbf{p}_i}{\mathbf{q}_i}, \tag{4}$$

$$\mathcal{L}_{\text{prune}} = \mathcal{L}_{\text{KL}} + \frac{1}{N}\sum_{n=1}^{N}\sum_{L\in\mathbf{L}} \left(\mathcal{X}_i^L - \mathcal{X}_i^{L_j}\right)^2, \tag{5}$$

where $\mathbf{p}_i$ (or $\mathbf{q}_i$) is the output distribution of M (or M_j) and L_j is the pruned layer in M_j corresponding to L in M.

The rationale for this approach comes from the effectiveness of knowledge distillation. We expect that the pruning loss ($\mathcal{L}_{\text{prune}}$) is helpful to compute channel-wise importance scores with information about the network's internal dynamics as knowledge distillation does. In addition, since computing $\mathcal{L}_{\text{prune}}$ does not require any label, we can prune channels without labels. Note that our method uses the same distillation loss for retraining a pruned model, so the retraining process also can be done without labels.

Good Anchor Layers. To get good anchor layers, we introduce the following topological concept for formulation. For a layer l in a neural network, we define a node ordering method denoted by *t-rank*(l) as follows.

$$\textit{t-rank}(l) = \begin{cases} \max_{n\in\mathbf{N}(l)} \textit{t-rank}(n) + 1 & \text{if } |\mathbf{N}(l)| \geq 1 \\ 1 & \text{otherwise} \end{cases}, \tag{6}$$

where $\mathbf{N}(l)$ is a set of inbound layers of l. For a layer l, the inbound layers of l are defined as layers directly connected to l. That is, the inputs of l correspond to the outputs of its inbound layers. Thus, *t-rank* is consistent with the order of calculation of layers during model inference. An algorithm for calculating *t-rank* is provided in the supplemental material.

Based on the concept of *t-rank*, we propose a simple anchor selection method named First Common Descendant (FCD). FCD and its comparison methods are as follows.

- **First Common Descendant (FCD):** This method first finds all groups of convolutional layers sharing coupled output channels and sorts them by the least *t-rank* of each group. Then, it uniformly divides the sorted groups to d partitions. For each partition p, it selects the activation layer of the first common descendant convolution layer of the last group in p as an anchor layer. In our experiments, d is fixed to 5.

- **All Activations (ALL):** This method selects all activation layers as anchor layers.

- **Heuristics (HEU):** This method selects the last layer of each block as an anchor layer. Note that blocks we consider for HEU are defined by the authors of models. Such blocks are used by many existing works for exploiting a network's internal feature maps for knowledge distillation.

Note that due to the concept of *t-rank*, the anchor layers found by FCD are evenly distributed in the order of calculation no matter what type of skip connection the model has. The rationale of FCD is based on the assumption that the gradients from the pruning loss derived with evenly distributed anchor layers can be more stably propagated through layers in the model.

4 Experiments

4.1 Implementation Detail

Our implementation is based on TensorFlow's Keras, because we can access and transform the topology of a neural network with it.

Datasets. We use four datasets: ImageNet [2], CIFAR100 [8], Caltech-UCSD Birds 200 [25], and The Oxford-IIIT Pet dataset [17]. The last two datasets are rather small, and they were used for evaluating pruning performance with limited data as in [15].

Competitors. We call our pruning method BTS (**B**ridging **T**-*rank* and channel-wise importance **S**core). We have three competitors as follows. CURL is the Kullback-Leibler divergence-based pruning method in [15]. This method does not require labels to evaluate channel-wise importance scores like BTS. Since we compare BTS with CURL in terms of the effectiveness of selecting unimportant channels, we do not use the approach of combining the knowledge distillation method and MixUp specified in [15] for fairness. HRank is a channel pruning method based on the rank of the feature map associated with each channel [11]. GF is Group Fisher Pruning proposed in [12]. Since HRank does not provide a scheme assigning sparsity to each layer, we make HRank work with the same layer-wise sparsity of GF.

Models. We use famous convolutional neural networks having residual or skip connections: EfficientNet-B0 (ENetB0) [22], ResNet-50 (RNet50) [3], and DenseNet-121 (DNet121) [5]. Note that we use models provided in Keras [1] which are pretrained for ImageNet.

For the transfer learning, we use Adam [7] as an optimizer, and the initial learning rate is set to be 1e-3 or 2e-3. This setting is also used for computing channel-wise importance scores based on Group Fisher Pruning. Note that we did not struggle to achieve state-of-the-art performance so hyper-parameter optimization was not used. Nevertheless, we used CutMix [29] and MixUp [30] due to their simplicity when computing channel importance scores, fine-tuning, and transfer learning.

Table 1: ImageNet and CIFAR-100 Results

Method	ImageNet			CIFAR-100		
	Top-1	#P (M)	#F (B)	Top-1	#P (M)	#F (B)
ENetB0	77.19	5.33	0.40	86.46	4.18	0.40
GF	70.02	2.47	0.22	81.49	1.28	0.22
CURL_D	69.37	1.58	0.23	**82.47**	1.20	0.22
HRank_D	**71.37**	2.47	0.22	80.08	1.28	0.22
BTS_HEU	68.66	2.12	0.22	78.25	1.13	0.22
BTS_ALL	68.10	2.09	0.21	80.55	1.18	0.21
BTS_FCD	**71.89**	2.62	0.22	**81.79**	1.40	0.22
RNet50	74.89	25.64	3.88	82.66	23.79	3.88
GF	66.41	4.76	2.04	80.27	7.19	1.85
CURL_D	68.94	6.71	2.05	80.76	8.32	2.10
HRank_D	**70.15**	7.16	1.82	**81.36**	9.82	2.45
BTS_FCD	**69.33**	5.74	2.04	**81.01**	7.53	2.10
DNet121	74.76	8.06	2.85	84.39	7.14	2.85
GF	67.13	3.29	1.67	**83.11**	2.91	1.68
CURL_D	69.45	3.52	1.69	82.13	3.11	1.65
HRank_D	**69.59**	3.77	1.74	81.37	3.12	1.65
BTS_FCD	**70.52**	3.71	1.69	**82.91**	3.43	1.65

Table 2: Channel Pruning with Limited Data

Method	CUB 200			Oxford-IIIT Pets		
	Top-1	#P (M)	#F (B)	Top-1	#P (M)	#F (B)
ENetB0	80.32	4.31	0.40	92.12	4.10	0.40
GF	76.66	1.25	0.22	87.11	0.97	0.22
CURL_D	79.00	1.38	0.23	89.65	1.32	0.22
HRank_D	78.81	1.25	0.22	87.89	0.99	0.22
BTS_HEU	78.81	1.34	0.22	90.43	1.00	0.22
BTS_ALL	**79.20**	1.50	0.21	**90.82**	1.12	0.21
BTS_FCD	**79.20**	1.58	0.22	**90.62**	1.31	0.22

4.2 Results

Overall Results. The overall results are presented in Table 1 and Table 2. For this experiment, compressed models are controlled to have similar FLOPs. Each compressed model is retrained with about 15 or 30 epochs after pruning it. #P is the number of parameters, and #F is the number of FLOPs. CURL and HRank are presented with 'D' because they were retrained based on our distillation method without labels.

It should be noticed that only GF is set to work with ground truth labels, while the others are not. The overall results say that BTS effectively prunes unimportant channels even without labels. BTS_FCD is always included in the top-2 results, even if the ranks of the methods vary over datasets and models. In addition, it is similar to GF in all cases in terms of accuracy, while BTS_ALL and BTS_HEU do not. This result shows that FCD plays a crucial role in making our distillation method effective.

One can worry about why the accuracy of pruned models by GF is usually lower than that by other methods. This may be because the knowledge distillation is more effective than expected. In addition, even though we controlled the pruning methods in terms of the number of FLOPs, the importance score function of GF is normalized by memory reduction.

Table 3: Cost Analysis (min) for Pruning (**ENetB0**)

Datasets	GF	CURL	HRank	BTS_{FCD}
ImageNet	1,319	32	5	**3**
CIFAR-100	1,457	35	5	**4**
CUB 200	1,727	36	4	**5**
Oxford-IIIT Pets	1,766	39	4	**5**

Table 4: Comparing with Differentiable Pruning (CIFAR-100)

Method	Top-1	#P (M)	#F (B)	Cost (min)
ENetB0	86.46	4.18	0.40	-
DPruning	80.93	1.50	0.23	67
BTS_{FCD}	**81.79**	1.40	0.22	**4**
RNet50	82.66	23.79	3.88	-
DPruning	80.06	9.77	2.09	53
BTS_{FCD}	**81.01**	7.53	2.10	**3**

That is, the experimental setup is somewhat unfair to GF. Due to this limitation, we do not claim that BTS_{FCD} is more effective than GF. Nevertheless, we can still say that BTS_{FCD} has comparable pruning performance to GF.

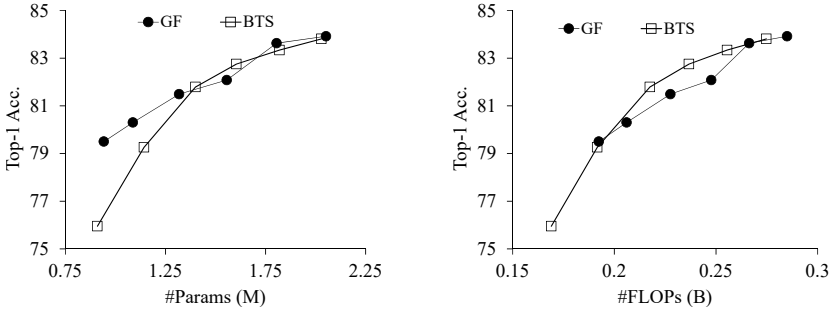
Pruning Cost. Table 3 presents the pruning cost for each method. Even if BTS is a gradient-based method like GF, its pruning cost is two orders of magnitude lower than that of GF. It is also much lower than that of CURL and similar to that of HRank. Because HRank does not provide sparsity information per layer, we used the same sparsity distribution of GF. Thus, the cost of determining layer-wise sparsity is not included in the pruning cost for HRank reported in Table 3. It is well-known that such a cost is not that cheap because it usually requires a kind of optimization based on many repetitions. Thus, even if HRank has comparable compression performance to our pruning method, the actual cost for HRank is likely to be much larger than that for ours.

Comparing with Differentiable Pruning. In addition to the competitors, we compared BTS_{FCD} with a differentiable channel pruning method [5]. This pruning method determines which channels could be removed by training channel-wise gates. Since it works with given layer-wise sparsity information like HRank, we used the layer-wise sparsity of GF for this. In addition, we used our knowledge distillation for pruning and retraining with this method.

Table 4 presents the result of comparing the differentiable method and ours. DPruning denotes the pruning method of [5]. In this table, the pruning cost of BTS_{FCD} is an order of magnitude smaller than that of DPruning. Since DPruning requires a training process with multiple epochs to train gates for channels, its pruning cost is somewhat expensive. Note that the pruning cost for DPruning does not include the cost of calculating the layer-wise sparsity. With such an efficiency, BTS_{FCD} is more effective than DPruning in terms of pruning performance for ENetB0 and ResNe50.

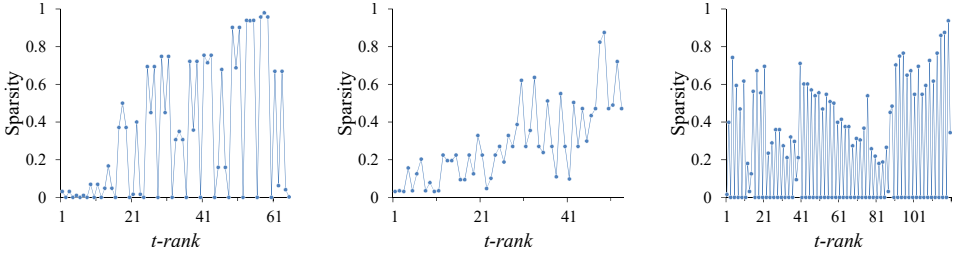
#FLOPs (#Params) vs. Accuracy Test. We conducted experiments to compare BTS_{FCD} and GF in terms of pruning performance over the number of parameters and FLOPs. Figure 3 presents their results with ENetB0 for CIFAR-100. The results show that BTS_{FCD} and GF have similar pruning performance regarding the number of parameters and FLOPs.

Ablation Study for Knowledge Distillation. We present the result of an ablation study in Table 5 for the loss derived with the anchor layers. For this study, we used 10 epochs for retraining. 'Anchor Loss' represents the loss with the anchor layers selected by FCD. Table 5 says that using $\mathcal{L}_{KL}$ and the anchor loss together is more effective than using $\mathcal{L}_{KL}$ solely.



(a) #Params vs. Accuracy (b) #FLOPs vs. Accuracy

Figure 3: #Params and #FLOPs vs. Accuracy



(a) ENetB0 (b) RNet50 (c) DNet121

Figure 4: Sparsity Distribution over Different Models (BTS_{FCD}, CIFAR-100)

Table 5: Ablation study for anchor-based loss (ImageNet, Acc., 10 Epochs)

Methods	ENetB0	RNet50	DNet121
$\mathcal{L}_{\text{KL}}$	70.96	67.27	67.89
$\mathcal{L}_{\text{KL}}$+Anchor Loss	**71.45**	**67.70**	**68.96**

Layer-wise Sparisity. Figure 4 presents layer-wise sparsity values ordered by *t-rank*. For ENetB0 and RNet50, the channels of layers close to inputs are less pruned, while those of layers close to outputs are heavily pruned. For DNet121, it has a unique sparsity distribution due to concatenation-based skip connections. Figure 5 presents layer-wise sparsity values over different pruning methods. It is interesting that BTS_{FCD} and CURL produce similar

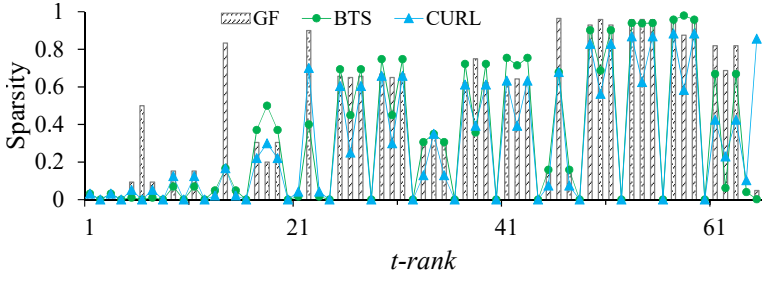


Figure 5: Sparsity Distribution over Different Methods (ENetB0, CIFAR-100)

sparsity distributions, while GF produces a relatively different distribution. There are some layers having high sparsity in the distribution produced by GF even if they are close to the inputs. Since the sparsity distribution of BTS_{FCD} does not have such a tendency, it is interesting to investigate further why GF has a different distribution.

5 Conclusions and Future Works

This paper dealt with improving and modifying Group Fisher Pruning for efficiency and applicability. We proposed the formal method for handling DenseNet-style skip connections in channel pruning. We also proposed the technique to make Group Fisher Pruning much faster in terms of pruning time. Then, we proposed the effective channel-wise importance scoring method based on feature-level knowledge distillation. Since knowledge distillation does not require labels, our technique can be easily applied to label-free channel pruning. Finally, we conducted extensive experiments for demonstration. The results showed that the proposed method has similar and comparable pruning performance to Group Fisher Pruning, with lower pruning costs for two orders of magnitude.

In the future, we will analyze the effect of anchor layers in our pruning methods on the final accuracy of a pruned model. We will devise a more effective algorithm for selecting anchor layers based on the analysis. In addition, we will improve the algorithm for finding convolutional layers sharing coupled output channels to support very complex neural networks such as NASNet.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2021-0-00907, Development of Adaptive and Lightweight Edge-Collaborative Analysis Technology for Enabling Proactively Immediate Response and Rapid Learning).

References

[1] Francois Chollet et al. Keras, 2015. URL https://github.com/fchollet/keras.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *ECCV*, pages 630–645, 2016.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[5] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269, 2017.

[6] Jaedeok Kim, Chiyoun Park, Hyun-Joo Jung, and Yoonsuck Choe. Plug-in, trainable gate for streamlining arbitrary neural networks. In *AAAI*, pages 4452–4459, 2020.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[8] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research).

[9] Changsik Lee, Seungwoo Hong, Sungback Hong, and Taeyeon Kim. Performance analysis of local exit for distributed deep neural networks over cloud and edge computing. *ETRI Journal*, 42(5):658–668, 2020.

[10] Tianhong Li, Jianguo Li, Zhuang Liu, and Changshui Zhang. Few sample knowledge distillation for efficient network compression. In *CVPR*, 2020.

[11] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao. Hrank: Filter pruning using high-rank feature map. In *CVPR*, pages 1526–1535, 2020.

[12] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In *ICML*, volume 139, pages 7021–7032, 2021.

[13] Nan Lu, Zhao Wang, Xiaoxiao Li, Gang Niu, Qi Dou, and Masashi Sugiyama. Federated learning from only unlabeled data with class-conditional-sharing clients. In *International Conference on Learning Representations*, 2022.

[14] Ekdeep Singh Lubana and Robert Dick. A gradient flow framework for analyzing network pruning. In *ICLR*, 2021.

[15] Jian-Hao Luo and Jianxin Wu. Neural network pruning with residual-connections and limited-data. In *CVPR*, 2020.

[16] Sejun Park, Jaeho Lee, Sangwoo Mo, and Jinwoo Shin. Lookahead: A far-sighted alternative of magnitude-based pruning. In *ICLR*, 2020.

[17] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, pages 3498–3505, 2012.

[18] Hanyu Peng, Jiaxiang Wu, Shifeng Chen, and Junzhou Huang. Collaborative channel pruning for deep networks. In *ICML*, volume 97, pages 5113–5122, 2019.

[19] Xin Qian and Diego Klabjan. A probabilistic approach to neural network pruning. In *ICML*, volume 139, pages 8640–8649, 2021.

[20] Chengchao Shen, Xinchao Wang, Youtan Yin, Jie Song, Sihui Luo, and Mingli Song. Progressive network grafting for few-shot knowledge distillation. In *AAAI*, 2021.

[21] Arvind Subramaniam and Avinash Sharma. N2nskip: Learning highly sparse networks using neuron-to-neuron skip connections. In *BMVC*, 2020.

[22] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*, volume 97, pages 6105–6114, 2019.

[23] Jialiang Tang, Xiaoyan Yang, Xin Cheng, Ning Jiang, Wenxin Yu, and Peng Zhang. Data-free knowledge distillation with positive-unlabeled learning. In Teddy Mantoro, Minho Lee, Media Anugerah Ayu, Kok Wai Wong, and Achmad Nizar Hidayanto, editors, *Neural Information Processing*, 2021.

[24] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Neural pruning via growing regularization. In *ICLR*, 2021.

[25] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[26] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *CVPR*, pages 7130–7138, 2017.

[27] Jaemin Yoo, Minyong Cho, Taebum Kim, and U Kang. Knowledge extraction with no observable data. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[28] Sixing Yu, Arya Mazaheri, and Ali Jannesari. Auto graph encoder-decoder for neural network pruning. In *ICCV*, pages 6362–6372, 2021.

[29] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *CoRR*, abs/1905.04899, 2019.

[30] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

[31] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *NIPS*, page 883–894, 2018.