Single-Shot Pruning for Pre-trained Models: Rethinking the Importance of Magnitude Pruning

Hirokazu Kohama
Chubu University
tuna0724@mprg.cs.chubu.ac.jp

Hiroaki Minoura
Chubu University
himi1208@mprg.cs.chubu.ac.jp

Tsubasa Hirakawa
Chubu University
hirakawa@mprg.cs.chubu.ac.jp

Takayoshi Yamashita
Chubu University
takayoshi@isc.chubu.ac.jp

Hironobu Fujiyoshi
Chubu University
fujiyoshi@isc.chubu.ac.jp

Abstract

Transformer models with large-scale pre-training have performed excellently in various computer vision tasks. However, such models are huge and difficult to apply to mobile devices with limited computational resources. Moreover, the computational cost of fine-tuning is high when the model is optimized for a downstream task. Therefore, our goal is to compress the large pre-trained models with minimal performance degradation before fine-tuning. In this paper, we first present the preliminary experimental results on the parameter change by using pre-trained or scratch models when training in a downstream task. We found that the parameter magnitudes of pre-trained models remained largely unchanged before and after training compared with scratch models. With this in mind, we propose an unstructured pruning method for pre-trained models. Our method evaluates the parameters without training and prunes in a single shot to obtain sparse models. Our experiment results show that the sparse model pruned by our method has higher accuracy is more than previous methods on the CIFAR-10, CIFAR-100, and ImageNet classification tasks.

1. Introduction

The performance of a deep neural network improves in proportion to the number of parameters [7]. However, while a deep neural network with many parameters can be highly effective for generalization, it uses more memory, so the computational cost is high [25]. Therefore, pruning a dense network with many parameters is a significant technique for deploying deep neural networks in real-time processes and devices with limited computational resources.

MetaFormer [24] is a deep neural network structure that has recently attracted attention in various fields. This structure consists of a token mixer, which mixes information between tokens, and a channel mixer, which mixes information within tokens. The structure of MetaFormer is represented by Transformer [19] in natural language processing, and a Vision Transformer (ViT) [2], MLP-Mixer (Mixer) [18], and PoolFormer (Pool) [24] for image recognition have been proposed. However, pre-training with large datasets is required because these models have many parameters.

The purpose of this study is to compress the pre-trained large models, such as Vision Transformers, with minimal performance degradation before fine-tuning. To develop suitable pruning methods for pre-trained models, we investigate the difference in how parameters are trained by using pre-trained or scratch models when training in a downstream task (Section 3). This investigation reveals that pre-trained models tend to have almost no change in parameter magnitude before and after training in downstream tasks. This means that parameters obtained by pre-trained models are sufficient for downstream tasks. Considering this tendency, we demonstrate the importance of parameters obtained by pre-trained models. Then we show that our approach can outperform previous methods by highly evaluating parameters obtained from pre-training and pruning (Section 5). The contributions of our approach are summarized below:

- By pruning parameters before fine-tuning, the computational cost of fine-tuning is reduced for a pre-trained model.
- Our approach evaluates the parameters without training and can be applied as a single shot.
- Since the selection of redundant parameters does not depend on the network structure, our method can be applied to various models.

2. Related work

In this section, we review the literature on neural network pruning including the lottery tickets hypothesis [3].

2.1. Network Pruning

Network pruning has been widely studied since it was proposed by LeCun et al. [10]. The simplest method of network pruning is Magnitude Pruning [4]. Magnitude Pruning removes the parameters in the network from lower absolute values. Earlier methods compressed the model size by iterating pruning and fine-tuning. However, subsequent pruning methods have aimed to find a subnetwork that is as accurate as the original network and is more accurate faster after it has been fine-tuned once. The lottery tickets hypothesis was proposed by Frankle et al. [15] because subnetworks found by rewinding weights are more accurate than those found by repeating pruning and fine-tuning, the pruning algorithm for the lottery hypothesis includes iterative pruning, in which a small number of parameters is repeatedly pruned, and a single shot pruning, in which many parameters are pruned in a single shot. However, these algorithms incur a high computational cost because the parameter evaluation requires training. To solve this problem, pruning methods that can be applied before training have been proposed by Lee et al. [11] and Wang et al. [21]. Details of these methods are presented in Section 4.

2.2. Acceleration of Transformers

Although models with a MetaFormer structure is more accurate, they are computationally costly and slow. Thus, there have been several attempts to accelerate Transformers. For example, the Adaptive Vison Transformer (A-ViT) [23] and Dynamic Vision Transformer (DViT) [22] have been proposed to accelerate ViT. Whereas A-ViT is accelerated by reducing the number of tokens that are forward propagated during inference, DViT is accelerated by configuring an appropriate number of tokens for each individual image automatically. However, these approaches cannot fundamentally solve the memory usage issues because they do not reduce the model size. As an approach to pruning, we introduce several methods for reducing head in multi-head attention [19]. Shim et al. [16] proposed a head pruning method in which the gating parameters are attached to each head and regularized with L0 loss. Voita et al. [20] removed the heads by evaluating their attention confidence. However, these pruning methods only remove part of the token mixer. If the dimensionality of the multi-layer perceptron (MLP) is four times greater than the embedding dimensionality d, then the number of parameters for MLP and multi-head attention are $8d^2$ and $4d^2$, respectively (excluding their biases). Therefore, to compress the model size sufficiently, the channel mixer must also be removed.

3. Limitations of previous methods

In this section, we introduce gradient-based pruning methods and then discuss the evaluation of these methods on pre-trained models.

3.1. Revisiting previous methods applied prior to training

Single-Shot Network Pruning

Single-Shot Network Pruning (SNIP) [11] is the first algorithm to not require training for a pruning process. This algorithm quantitatively evaluates how a change in a single parameter θ_q affects the loss of the network:

$$\begin{aligned}\Delta\mathcal{L} &= |\mathcal{L}(\theta_q) - \mathcal{L}(\delta\theta_q)| \\ &= |\mathcal{L}(\theta_q) - \mathcal{L}(\theta_q) - \frac{\partial\mathcal{L}}{\partial\theta_q}(\delta\theta_q - \theta_q) - \mathcal{O}(||\delta\theta_q||^2)| \\ &= |\frac{\partial\mathcal{L}}{\partial\theta_q}\theta_q| \qquad (1)\end{aligned}$$

where θ_q is the q_{th} element of all parameters, δ is a perturbation to θ_q, $\Delta\mathcal{L}$ is the variation of loss $\mathcal{L}$, and $\mathcal{O}(||\delta\theta_q||^2)$ notation is terms of higher order than the 2th degree. The parameters where $\Delta\mathcal{L}$ become small can be considered to have a small impact on the loss even if they are removed. Thus, we can efficiently prune the parameters by removing those that have small $\Delta\mathcal{L}$.

Gradient Signal Preservation

Gradient Signal Preservation (GraSP) [21] is a pruning algorithm (like SNIP) that does not require training for a pruning process. SNIP evaluates each parameter independently, whereas GraSP evaluates the relationships between the parameters. Specifically, the objective of GraSP is to maintain or increase gradient flow after pruning. When the initial parameters $\boldsymbol{\theta}_0$ are perturbed $\boldsymbol{\delta}$, the gradient flow after pruning changes as follows:

$$\begin{aligned}\boldsymbol{S}(\boldsymbol{\delta}) &= \Delta\mathcal{L}(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) - \Delta\mathcal{L}(\boldsymbol{\theta}_0) \\ &= 2\boldsymbol{\delta}^T\mathbf{Hg} + \mathcal{O}(||\boldsymbol{\delta}||_2^2) \qquad (2)\end{aligned}$$

where $\mathbf{Hg}$ is a Hessian-vector product. Therefore, the score of each parameter is defined as follows:

$$\boldsymbol{S}(-\boldsymbol{\theta}) = -\boldsymbol{\theta} \odot \mathbf{Hg}. \qquad (3)$$

If the value of $\boldsymbol{S}(\boldsymbol{\theta})$ is negative, it means that the gradient flow decreases. Thus, the parameters with large $\boldsymbol{S}(\boldsymbol{\theta})$ are removed in priority.

3.2. Differences in parameters with and without pre-training

We investigate how the parameters change before and after training with and without pre-training to identify these

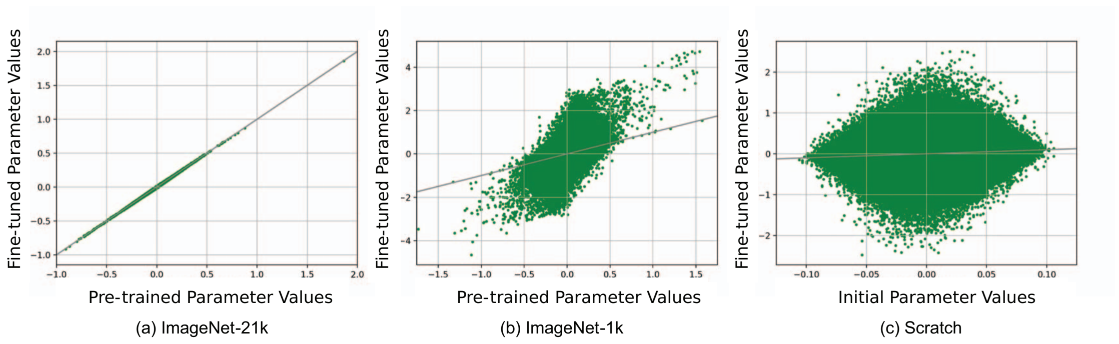

Figure 1. Distributions of ViT-B/16. The gray line indicates that the parameters are the same value before and after the training. The parameters are in good agreement with those before training in proportion to the dataset size used for pre-training.

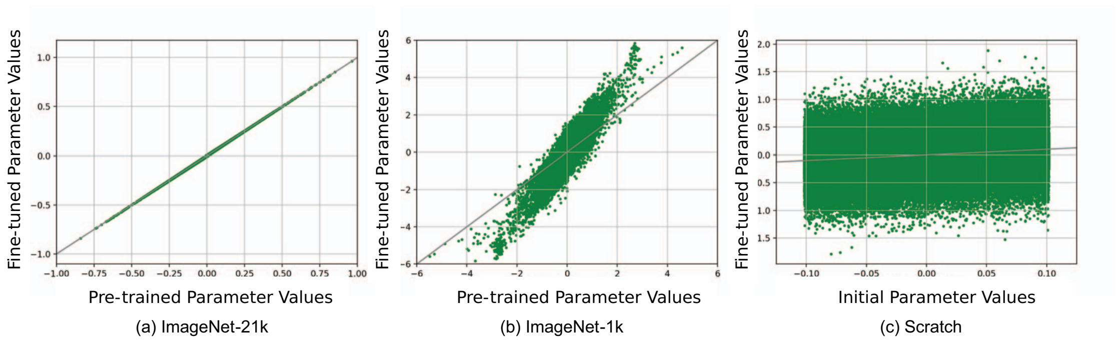

Figure 2. Distributions of Mixer-B/16. The gray line indicates that the parameters are the same value before and after training. The tendency is the same as in Figure 1. Mixer-B/16 additionally shows less variation in values than ViT-B/16 when pre-trained on ImageNet-1k.

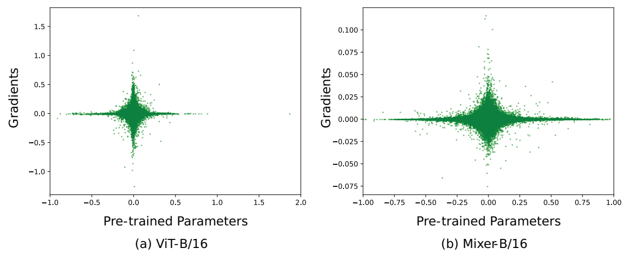

Figure 3. Correspondence relationship between parameters and gradients. To compute the gradient, we randomly select 128 images from CIFAR-10 as a mini-batch.

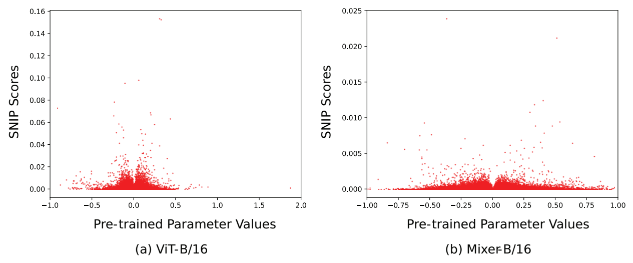

Figure 4. Scores of pre-trained parameters by SNIP. We compute the SNIP scores using the same gradient as in Figure 3. The evaluation is concentrated on parameters near 0.

differences. To compare the differences, we use ViT-B/16 and Mixer-B/16, which are pre-trained on ImageNet-1k or ImageNet-21k [1] and fine-tuned on CIFAR-10 [9]. Pre-training on ImageNet-21k or ImageNet-1k is done using AdamW [13] for 300 epochs.

Figure 1 shows the parameters of ViT-B/16, which is pre-trained, and the scratch models. Likewise, Figure 2 shows that of Mixer-B/16. The vertical axis is the trained parameters, and the horizontal axis is the parameters before training. Both results show that the parameters are in good agreement with that before training (initial values) in proportion to the dataset size used for pre-training. In particular, we observe that well-trained parameters with a large magnitude changed little before and after fine-tuning, while pre-trained parameters with small magnitude changed chaotically before and after fine-tuning. These results mean that the knowledge obtained by pre-training is very useful for downstream tasks such as CIFAR-10.

Previous methods such as SNIP and GraSP limit the score evaluation by Magnitude Pruning to a smaller range. This is because Magnitude Pruning considers the magnitude of the parameters, whereas SNIP and GraSP consider the magnitude and gradient or Hessian of the parameters simultaneously. Although limiting the evaluation range en-

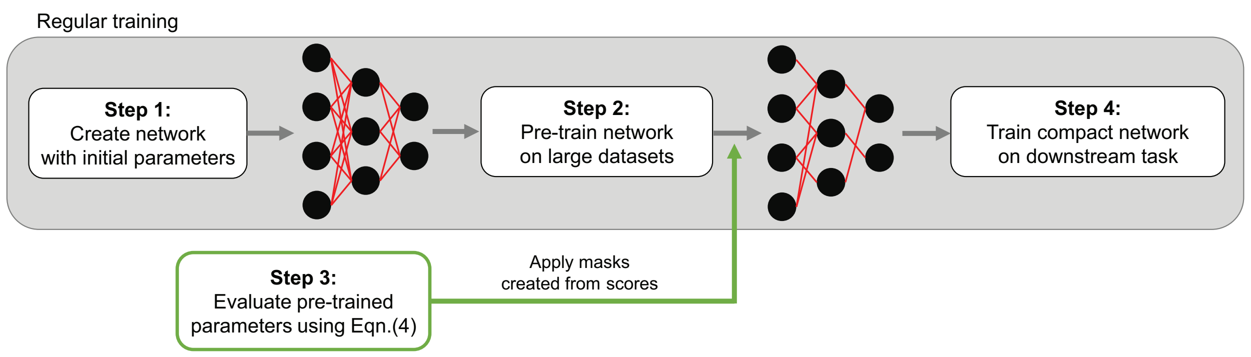
Regular training
Step 1: Create network with initial parameters
Step 2: Pre-train network on large datasets
Step 4: Train compact network on downstream task
Step 3: Evaluate pre-trained parameters using Eqn.(4)
Apply masks created from scores

Figure 5. Procedure of pruning for pre-trained models. First, we prepare an initial network (Step 1) and pre-train it (Step 2). Second, our pruning algorithm is applied once between Steps 2 and 4. Finally, the pruned network is trained on downstream task (Step 4).

ables more rigorous definition for redundant parameters, it can lead to errors in scoring parameters that change little. Specifically, when experiments are conducted under the assumption of pre-training, such as the results in Figures 1 and 2, the parameters of the pre-trained model vary less because the converges in the downstream task after less training than in the case of the scratch models. Visualization of the gradients of the pre-trained models (ViT-B/16 and Mixer-B/16) shows large gradients for the parameters near 0 and nearly 0 for the other parameters (in Figure 3). In such extreme gradient distributions, SNIP evaluations are concentrated on parameters near 0 (in Figure 4). Are these methods, which evaluate based on the magnitude of the gradient, adequately evaluating the parameters obtained by pre-training? We insist on that the parameters with nearly 0 gradients and large scales must also be evaluated simultaneously. This is because parameters obtained in the pre-training that are useful for the downstream task may remain largely unchanged and maintain their magnitude when trained in the downstream task. In addition, it is known that the variation in the parameters due to training is very small when the number of network dimensions is extremely large [6]. For these reasons, the magnitude of the gradient may be extremely small, which is a problem when pruning methods that use gradients, such as SNIP or GraSP, cannot adequately evaluate parameters with small variability and a large scale.

4. Pruning considering pre-training

The results in Section 3 show that the parameters obtained by pre-training are in good agreement with the initial values. However, it is not preferable to evaluate parameters before training by considering only their gradient-based methods because their scores are concentrated on parameters near 0. Therefore, we propose a method that can simultaneously evaluate the magnitudes of the parameters obtained by pre-training and connection sensitivity.

4.1. Proposed method

A prospective pruning method for the pre-trained model must consider both (a) the useful parameters obtained by pre-training and (b) those optimized in downstream tasks during fine-tuning. Here, we define (a) as parameters with a large value that changes little before and after fine-tuning and (b) as parameters with a small value that changes chaotically before and after fine-tuning. SNIP can be evaluated for (b) by capturing changes due to optimization of downstream tasks, but not for parameters with small change, such as (a). Therefore, we directly evaluate parameters with small variations by magnitude, and the others by SNIP:

$$S(\theta_q) = |\frac{\partial \mathcal{L}}{\partial \theta_q}\theta_q| + \alpha\theta_q^2 \tag{4}$$

where α is the scaling parameter, and $S(\theta_q)$ is the score of θ_q.

Other aspects of proposed method

Eqn.(4) contains the term of $\alpha\theta_q^2$. Therefore, our proposed method is similar to Eqn.(1) approximated by a second-order Taylor expansion:

$$\begin{aligned}\mathcal{L}(\delta\theta_q) =& \mathcal{L}(\theta_q) + \frac{\partial \mathcal{L}}{\partial \theta_q}(\delta\theta_q - \theta_q) + \frac{1}{2}\frac{\partial^2 \mathcal{L}}{\partial \theta_q^2}(\delta\theta_q - \theta_q)^2 \\ &+ \mathcal{O}(||\delta\theta_q||^3).\end{aligned} \tag{5}$$

In other words, our proposed method seems to be an approximation of second and third terms of Eqn. (1) to the second order, with the Hessian replaced by constant α.

Pruning Algorithm

Figure 3 shows the procedure of our proposed method for pre-training models. First, we prepare an initial network in Step 1 and pre-train it in Step 2. Next, we evaluate pre-trained parameters by using Eqn. (4). Then we create masks from the scores and apply them to the network. Here, we show the specific evaluation algorithm in Algorithm 1.

Algorithm 1 Pruning considering pre-training

```
Require: Sparsity $k$, training data $\mathcal{D}$, pre-trained parameters $\boldsymbol{\theta}$, hyper parameter $\alpha$
1: $\mathcal{D}_b = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{b} \sim \mathcal{D}$        ▷ Sample a mini-batch of training examples
2: for $\theta_q$ in $\boldsymbol{\theta}$ do
3:     $S \leftarrow |\frac{\partial \mathcal{L}(\mathcal{D}_b)}{\partial \theta_q} \theta_q| + \alpha \theta_q^2$        ▷ Compute the score of each parameter
4: end for
5: Compute $k_{\mathrm{th}}$ percentile of $\boldsymbol{S}$ as $\kappa$
6: $\boldsymbol{m} \leftarrow \boldsymbol{S} < \kappa$        ▷ Prune to the parameters with the smallest scores
7: Train the network with masked parameters $\boldsymbol{m} \odot \boldsymbol{\theta}$ on $\mathcal{D}$ until convergence.
```

Table 1. Test accuracy of Vision Transformers pruned to 98% (pre-trained on ImageNet-1k).

Method	Dataset		Ours					Random
α	CIFAR-10	CIFAR-100	0.001	0.0005	0.0001	0.00005	0.00001	-
ViT-B/16 [2]	✓	-	**90.48**	89.86	89.14	88.88	88.69	81.96
	-	✓	**67.16**	66.59	64.36	64.18	64.23	55.66
ViT-L/16 [2]	✓	-	**93.94**	92.81	92.44	92.26	92.13	83.67
	-	✓	**74.85**	73.47	71.74	70.57	70.99	56.49
Mixer-B/16 [18]	✓	-	86.36	88.10	**90.02**	89.89	89.69	79.03
	-	✓	64.39	65.96	**69.15**	68.00	67.13	54.35
Mixer-L/16 [18]	✓	-	**88.59**	88.28	87.79	87.78	88.29	78.27
	-	✓	**66.09**	65.58	64.59	64.75	65.31	52.81
Pool-M36 [24]	✓	-	**92.85**	**92.85**	92.55	92.92	92.75	68.47
	-	✓	74.27	73.89	73.90	**74.69**	74.31	41.10
Pool-M48 [24]	✓	-	93.61	93.56	93.71	93.45	**93.75**	65.89
	-	✓	**77.27**	76.71	76.11	76.10	76.25	39.30

Given pruning sparsity k, the pruning mask is computed by the score of the target parameter and the parameters are removed from the bottom at the ratio of k. Finally, we apply regular training in downstream tasks to the pruned network. This sequence of our pruning procedure is performed in a single shot.

Why do we not evaluate the parameters iteratively?

Intuitively, iterative pruning is thought to be more likely to yield higher performance than single-shot pruning because it does less damage to the model. However, Ma et al. [14] showed that single-shot pruning is more accuracy for models with smooth landscapes containing residual connections [5] than iterative pruning. Thus, model compression by single-shot pruning is applied in this paper because the models with the MetaFormer structure have residual connections.

5. Experiments

In this section, we present the experiments we conducted to validate the effectiveness of our proposed method. First, we compare the test accuracies of the models pruned by our proposed method using various scaling parameters α. Second, we compare our proposed method with three previous singe-shot pruning methods: magnitude pruning, SNIP, and GraSP. Finally, we compare our proposed method with SNIP in detail.

5.1. Experiment setup

To evaluate the effectiveness of our proposed method, we test it on two image classification datasets, CIFAR-10 and CIFAR-100, with six types of Vision Transformers. For the experiments on CIFAR-10 and CIFAR-100, we pre-train the networks on ImageNet-1k or ImageNet-21k. Pre-training on ImageNet-21k or ImageNet-1k is done using AdamW for 300 epochs. The targets to be pruned are the token mixer and channel mixer in all layers, and the pruning is applied non-structurally (Note that pool networks are only applied to the channel mixer because they do not have a parametric token mixer). The pruned network is trained using Adam [8] for 200 epochs for CIFAR-10, and 250 epochs for CIFAR-100, with an initial learning rate of 0.0001 and batch size of 128. The training datasets are also augmented by Mixup [26].

5.2. Proposed method and random pruning

We compare our proposed method with random pruning, which randomly generates the mask for a given sparsity. The test accuracy is reported in Table 1. Here, the networks pre-trained on ImageNet-1k are pruned to 98%.

Table 2. Test accuracy of Vision Transformers pruned to 98%. The models are trained on CIFAR-100 (pre-trained on ImageNet-1k).

Method	Ours	Magnitude	SNIP	GraSP
ViT-B/16	**67.16**	57.56	63.84	60.62
ViT-L/16	**74.85**	65.47	70.62	65.11
Mixer-B/16	64.39	62.09	**67.21**	61.22
Mixer-L/16	**66.09**	62.44	64.53	59.20
Pool-M36	**74.27**	63.10	73.47	62.19
Pool-M48	**77.27**	65.85	76.02	63.16

Table 3. Test accuracy of Vision Transformers pruned to 98%. The models are trained on CIFAR-10 (pre-trained on ImageNet-1k).

Method	Ours	Magnitude	SNIP	GraSP
ViT-B/16	**90.48**	83.81	89.41	85.74
ViT-L/16	**93.94**	90.32	92.07	87.92
Mixer-B/16	86.36	83.97	**89.15**	84.57
Mixer-L/16	**88.59**	86.77	88.46	85.63
Pool-M36	**92.85**	75.23	92.66	85.82
Pool-M48	93.61	87.99	**93.68**	86.53
Swin-B [12]	**96.10**	95.17	95.13	89.03

We found that our proposed method is more accurate than random pruning in all comparisons. We also found that for most networks, the highest accuracy is obtained when α is equal to 0.001. However, Mixer-B/16, scratch Pool-M36, and pre-trained Pool-M48 are most accurate for smaller α.

5.3. Proposed method and previous methods

The proposed and previous methods are compared in Tables 2, 3, and 4. The networks in Tables 2 and 3 are pre-trained on ImageNet-1k and fine-tuned on CIFAR-100 and CIFAR-10, respectively. The networks in Table 4 are pre-trained on ImageNet-21k and fine-tuned on ImageNet-1k. The test accuracy in Tables 2, 3, and 4 are obtained from the networks pruned to 98%. Here, our proposed method with scaling parameter 0.001 is applied to the networks. As shown in Table 2, high accuracy was obtained for all networks except Mixer-B/16. However, Mixer-B/16 is made more accurate than SNIP by tuning the scaling parameter (see Table 1). The network pruned by magnitude pruning is also effective, and we can observe the accuracy exceeds those of GraSP with ViT-L/16, Mixer-B/16, Mixer-L/16, and Pool-M48. This result shows that the magnitude

Table 4. Test accuracy of Vision Transformers pruned to 98%. The models are trained on ImageNet-1k (pre-trained on ImageNet-21k).

Method	Ours	Magnitude	SNIP	GraSP
ViT-B/16	**79.07**	78.62	**79.07**	77.23
ViT-L/16	**80.79**	80.60	80.61	78.53

of parameters obtained by pre-training is important. In Table 3, we can similarly observe high accuracy for all networks except Mixer-B/16 and Pool-M48. Mixer-B/16 and Pool-M48 can also be made more accurate than SNIP by tuning the scaling parameter (see Table 1). In Table 4, the networks pruned by the proposed method obtain the highest accuracy, and the difference is especially noticeable for ViT-L/16. Meanwhile, SNIP and Magnitude have almost the same accuracy.

Table 5 compares the effectiveness of the proposed method and SNIP in further detail. The networks in Tables 5 is retrained on CIFAR-10. Most test accuracy in Tables 5, including the scratch networks, are higher when pruned by our proposed method. We also observe a monotonic degradation of accuracy in proportion to sparsity. This result supports the hypothesis that no subnetwork can outperform the original network of Ma et al. [14].

5.4. Detailed comparison with Magnitude

Tables 6 and 7 shows the overlap ratio for the parameters pruned by Magnitude and those pruned by the other methods. The overlap ratio is computed using the remaining 2% parameters of the model compressed to 98%. The models used to calculate the overlap ratio are pre-trained on ImageNet-21k. The scaling parameter of the proposed method is 0.001.

We consistently observe that the proposed method has the highest overlap ratio in comparison with all methods. This shows that the proposed method is able to reflect the magnitude of the initial values more strongly in the evaluation of the parameters. In addition, we found that the models such as Mixer and Pool, which consist of pure fully-connected layers, have high overlap ratio. We intuitively understand that this trend is due to the computational complexity of the attention mechanism included in ViTs, but more experiments are needed to prove it.

5.5. Layer Collapse

In single-shot pruning, layer collapse is caused by removing many parameters [17]. Here, in layer collapse, all of the parameters of an entire layer are removed, resulting in a severe loss of accuracy. In this section, we experimentally show that layer collapse is less likely to occur in our proposed method than in SNIP.

We qualitatively investigate where the layers have been pruned. In Figures 6, 7 and 8, we compare the sparsities at each layer of the Vision Transformers globally pruned to 95%. The scaling parameter of the proposed method is 0.001. These models are pre-trained on ImageNet-1k and retrained on CIFAR-10. The light red bars and the deep red bars represent the token mixer. Note that the token mixer in ViT-B/16 is divided into scaled dot-product attention, which includes query, key, and value weights, and one linear layer.

Table 5. Test accuracy of Vision Transformers pruned by our proposed method and SNIP on CIFAR-10.

Method	Pretrain		Ours ($\alpha = 0.001$)			SNIP		
Sparsity[%]	ImageNet-1k	ImageNet-21k	90.0	95.0	98.0	90.0	95.0	98.0
ViT-B/16	✓	-	**95.98**	**94.71**	**90.48**	95.46	93.37	89.41
	-	✓	**95.92**	94.37	**90.92**	95.49	**93.46**	88.50
	-	-	78.27	**78.48**	**76.42**	**79.03**	77.96	76.21
ViT-L/16	✓	-	**96.40**	**95.83**	**93.94**	96.33	95.04	92.07
	-	✓	**96.54**	**95.82**	**93.70**	96.13	94.91	92.49
	-	-	77.70	78.13	77.78	**77.76**	**78.32**	**77.04**
Mixer-B/16	✓	-	**95.20**	92.52	86.36	94.55	**92.81**	**89.15**
	-	✓	95.17	92.34	87.05	**95.58**	**92.99**	**89.57**
	-	-	79.97	75.85	65.77	**81.97**	**80.68**	**77.92**
Mixer-L/16	✓	-	**94.02**	**91.13**	**88.59**	91.64	90.53	88.46
	-	✓	**93.93**	**90.86**	**88.82**	91.70	90.72	88.22
	-	-	**81.33**	**79.89**	**75.21**	74.58	68.71	66.03
Pool-M36	✓	-	**96.08**	**94.99**	**92.85**	95.70	94.50	92.66
	-	✓	**96.29**	94.30	91.51	96.17	**94.94**	**92.49**
	-	-	**75.01**	**69.21**	**62.36**	72.91	39.81	39.55
Pool-M48	✓	-	**96.43**	**95.46**	93.61	96.02	95.38	**93.68**
	-	✓	**96.72**	95.10	93.41	96.54	**95.65**	**93.46**
	-	-	**76.23**	**76.23**	**70.26**	72.80	66.72	39.59

Table 6. Overlap ratio for the parameters pruned by Magnitude and those pruned by the other methods which used CIFAR-10 to calculate the scores [%].

	ViT-B/16	ViT-L/16	Mixer-B/16	Mixer-L/16	Pool-M36	Pool-M48
Ours	**11.91**	**13.41**	**99.97**	**98.06**	**77.48**	**81.60**
SNIP	11.78	12.24	1.69	1.99	11.48	11.62
GraSP	7.53	6.73	1.69	1.99	8.54	9.06

Table 7. Overlap ratio for the parameters pruned by Magnitude and those pruned by the other methods which used CIFAR-100 to calculate the scores [%].

	ViT-B/16	ViT-L/16	Mixer-B/16	Mixer-L/16	Pool-M36	Pool-M48
Ours	**12.14**	**12.14**	**99.99**	**99.92**	**79.81**	**82.23**
SNIP	12.03	13.06	1.73	2.20	12.38	11.93
GraSP	7.23	6.75	1.71	2.19	9.56	7.73

The light blue bars and the deep red bars also represent the channel mixer.

Comparing these figures, we observe that each layer is most uniformly pruned when using magnitude pruning. In contrast, some layers are hardly pruned when using SNIP. Our proposed method also prunes with bias due to the influence of SNIP as shown in Figures 6, 7 and 8. However, our proposed method is less biased than that of simple SNIP because it includes an evaluation of magnitude pruning.

6. Conclusion

In this paper, we focused on how parameters are in good agreement before training (initial values) in proportion to the dataset size used for pre-training. On the basis of this characteristic, we proposed a pruning method suitable for pre-trained models. Our proposed method prunes the pre-trained model for a given task in a single shot prior to re-training. We empirically showed that the proposed method exceeds the test accuracy of previous methods in most cases and is able to remove parameters more evenly than single-shot network pruning (SNIP). This method facilitates the deployment of networks that require pre-training in environments with limited computational resources. In future work, we plan to determine the scaling parameter automatically. Our code is available at: `https://github.com/tuna0724/Pruning`.

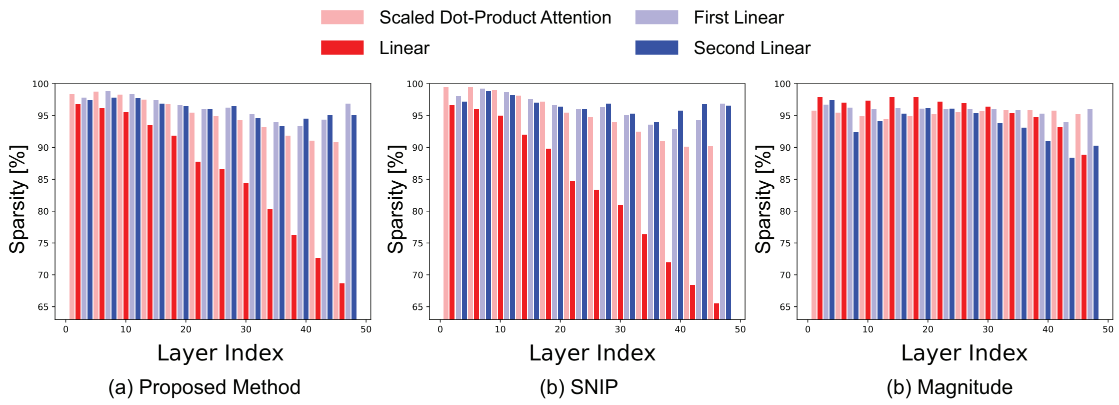

Figure 6. Sparsity at each layer of ViT-B/16. Our proposed method and SNIP have the same tendency that the linear layers in the token mixer are minimally pruned. However, our proposed method prunes more uniformly than SNIP. Also, each layer is most uniformly pruned.

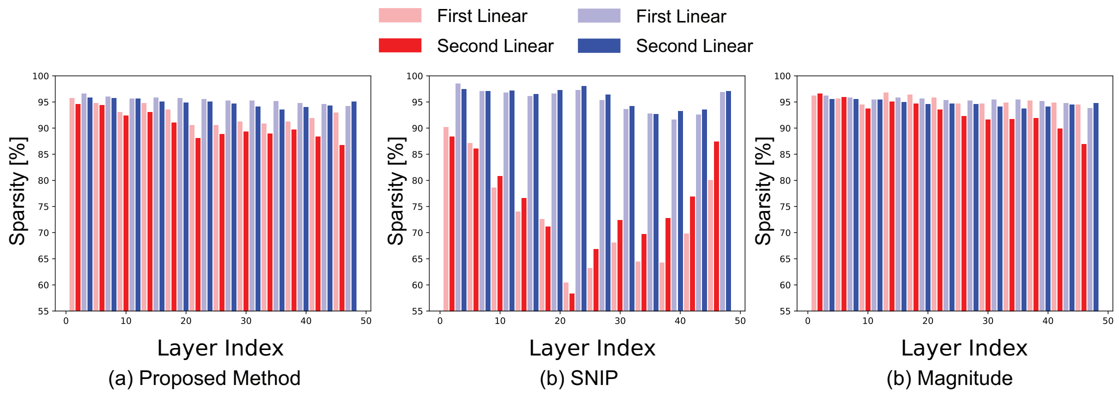

Figure 7. Sparsity at each layer of Mixer-B/16. SNIP biasedly prunes the parameters of Token Mixer. Our proposed method is more uniformly applied to entire layers than SNIP.

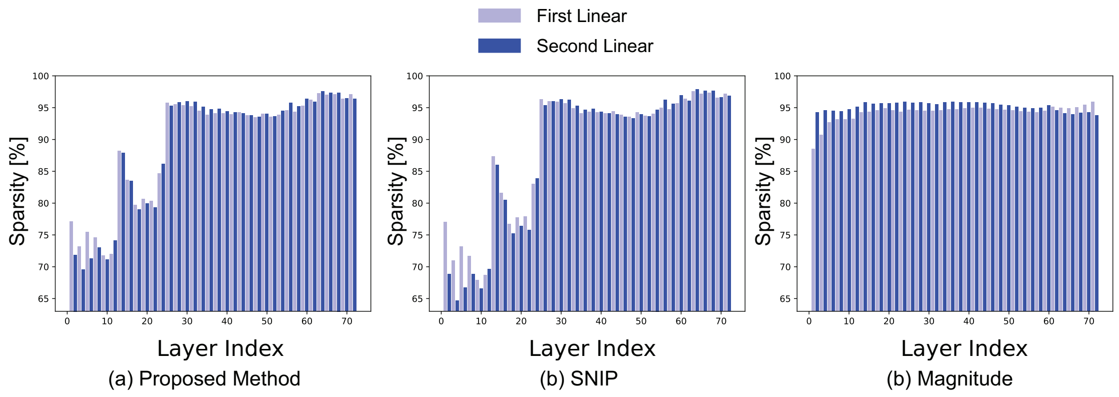

Figure 8. Sparsity at each layer of Pool-M36. Our proposed method and SNIP exhibit the same tendency as in Figure 4. Between our proposed method and SNIP, our proposed method prunes more uniformly.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[3] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[4] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1135–1143, Cambridge, MA, USA, 2015. MIT Press.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, June 2016.

[6] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[7] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[9] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[10] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

[11] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY. In *International Conference on Learning Representations*, 2019.

[12] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[14] Xiaolong Ma, Geng Yuan, Xuan Shen, Tianlong Chen, Xuxi Chen, Xiaohan Chen, Ning Liu, Minghai Qin, Sijia Liu, Zhangyang Wang, and Yanzhi Wang. Sanity checks for lottery tickets: Does your winning ticket really win the jackpot? In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[15] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020.

[16] Kyuhong Shim, Iksoo Choi, Wonyong Sung, and Jungwook Choi. Layer-wise pruning of transformer attention heads for efficient language modeling. *CoRR*, abs/2110.03252, 2021.

[17] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6377–6389. Curran Associates, Inc., 2020.

[18] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Peter Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. MLP-mixer: An all-MLP architecture for vision. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[20] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics.

[21] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020.

[22] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[23] Hongxu Yin, Arash Vahdat, Jose Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[24] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer

is actually what you need for vision. *CoRR*, abs/2111.11418, 2021.

[25] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[26] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.