The Bonsai Hypothesis: An Efficient Network Pruning Technique

Yasuaki Ito, Koji Nakano, Akihiko Kasagi

The Bonsai Hypothesis: an Efficient Network Pruning Technique

Yasuaki Ito[1], Koji Nakano[1], and Akihiko Kasagi[2]

[1] Graduate School of Advanced Science and Engineering,
Hiroshima University, Japan
{yasuaki,nakano}@cs.hiroshima-u.ac.jp
[2] Fujitsu Ltd., Japan
kasagi.akihiko@fujitsu.com

Abstract. Machine learning technology has made it possible to solve a variety of previously unfeasible problems. Accordingly, the size of network models has been increasing. Thus, research on model compression by network pruning has been conducted. Network pruning is usually performed on already-trained network models. However, it is often difficult to remove a large number of weights from a network while maintaining accuracy. We suppose that the reason for this is that well-trained networks have many weight parameters with complicated correlations among them, and such parameters make pruning difficult. Based on this supposition, in this paper, we state *the bonsai hypothesis*: pruning can be more effective when starting from untrained models than already trained models. To support the hypothesis, we present a simple and efficient channel pruning algorithm. We performed pruning of untrained and trained models using the proposed algorithm for VGG16 on CIFAR-10 and CIFAR-100. As a result, we found that the untrained models tend to reduce more channels than the already-trained models.

Keywords: Model compression · Channel pruning · Machine learning · Convolutional neural network.

1 Introduction

Over the past decade, the development of machine learning technologies, starting with deep learning, has led to rapid progress in AI capabilities. At the same time, the demand for computational power and memory usage is getting higher as networks have become larger and more complicated to adopt practical applications. This high demand often impedes the execution of such applications in computing environments that do not have high performance processors and/or high memory capacities, such as smartphones and IoT devices. As a solution to this problem, a method called *pruning* has been proposed to reduce the number of parameters and the amount of computation by removing some of the weights of the network [1, 10, 17].

Many pruning methods are derived from [5], in which removing weights is performed as follows.

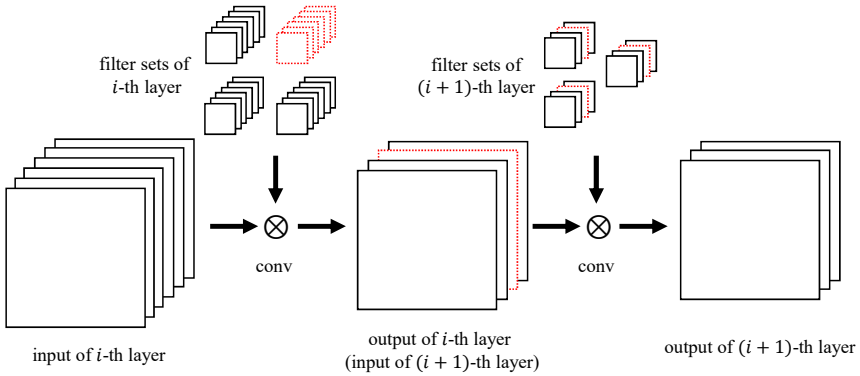


Fig. 1. Channel pruning for a convolutional layer

Step 1: Training the network model to be pruned
Step 2: Pruning unimportant weights of the network
Step 3: Re-training the pruned network
Step 4: Repeat Steps 2 and 3

In Step 1, the network model to be pruned is trained. However, in practice, this step is not necessary to perform when pruning since the target network model has already been trained. For the trained model, in Step 2, we perform the pruning. According to the granularity of pruning, pruning is mainly classified as weight pruning, kernel pruning, channel pruning, and layer pruning. In weight pruning and kernel pruning, since pruning is performed at fine granularity, a large number of weights can be removed, maintaining the accuracy [9, 12]. However, the data structure storing weights becomes irregular weight matrices that prevent efficient parallel computation. Therefore, GPU implementations to accelerate the computation for such models have been proposed [4, 18]. On the other hand, in channel pruning and layer pruning, models obtained by pruning have a regular structure that can be efficiently computed in parallel [13]. Hence, in this paper, we adopt channel pruning [6, 9, 11, 19] for efficient computation on the GPU. In channel pruning for a convolutional neural network, output channels of a layer and filters related to the channels are removed as illustrated in Figure 1. For example, when pruning the k-th output channel of the i-th layer, the k-th filter set of the i-th layer and the k-th filter in each filter set of the $(i+1)$-th layer are removed together. After pruning, in Step 3, since the accuracy drops significantly, we retrain the model to recover it. We repeat Steps 2 and 3 until a sufficient number of weight parameters are eliminated and/or a certain number of iterations are performed.

Typically, network pruning is performed on already trained network models as shown above. However, it is often difficult to remove a large number of weights from a network while maintaining accuracy. We suppose that the reason for this is that well-trained networks have many weight parameters with complicated correlations among them, and such parameters make pruning difficult.

In other words, network pruning should be performed before the network has been sufficiently trained. Based on this idea, in this paper, we state *the bonsai hypothesis*.

The Bonsai Hypothesis. *Pruning can be more effective when starting from untrained models than already trained models.*

This hypothesis is analogous to *bonsai*, which is the art of producing a small tree in a small pot that mimics the shape of a real tree. We correspond network models to bonsai trees, untrained networks to saplings, network training to tree growth, and network pruning to tree pruning. Also, well-pruned networks as the goal of network pruning correspond to well-shaped trees called *ideal bonsai trees*. Applying the above hypothesis to the bonsai making process, we can say that it is difficult to make an ideal bonsai by pruning from a fully grown tree, while it is easy to get close to an ideal bonsai by pruning during the growth process from a sapling.

Based on the bonsai hypothesis, the proposed channel pruning is performed from an untrained network initialized randomly. In addition, we propose a simple and efficient network channel pruning technique. Our new idea of the technique includes (i) *early growing of acceptance accuracy*, (ii) *pruning channels using the exponential search*, and (iii) *slightly shaking*. The details are described as follows.

Early growing of acceptance accuracy: In the pruning process of a trained network, retraining is generally performed to recover the accuracy decreased by pruning to the target accuracy. However, in the initial stage of training, since the accuracy of the network is not sufficient compared to the target accuracy, it is not easy to directly use the target accuracy in retraining. Therefore, in the proposed method, the target accuracy at the initial stage of training is obtained from the pre-training. More specifically, we train the network without pruning and record the change in the accuracy during the training. When pruning from an untrained network, we use this history as the target accuracy at the early stage of training and pruning. In terms of bonsai, we observe how a tree grows when it is grown from a sapling without pruning. This corresponds to using the growth process to raise the tree while pruning it again from a sapling to make it an ideal bonsai. While it is not possible to turn a mature tree back into a sapling in actual bonsai, we use the fact that it is possible in network training.

Pruning channels using the exponential search: In the general pruning approach, channels to be pruned are determined according to some metric, and then the channels are removed. For example, the L_1-norm of weights for each channel is computed, and a certain number or percentage of smaller channels are pruned. However, it is not easy to prune the appropriate number of channels. In this study, to determine the number of pruning channels, we adopt *the exponential search*, which is a method to efficiently find the number of nodes in radio networks with a small number of trials when the unknown number of nodes [14]. This method is employed to obtain the channels to be pruned efficiently. More specifically, by doubling the number of pruned channels while the acceptance

threshold is satisfied, a large number of channels are removed in a small number of epochs.

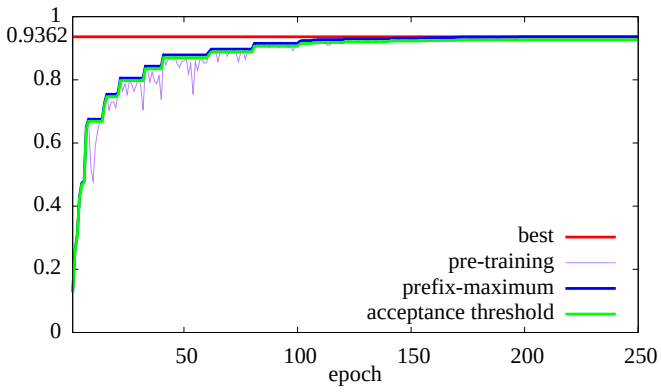
Slightly shaking: Channel pruning generally involves deleting channels layer by layer and terminating when no more channels can be deleted in any layer. However, in many cases, the pruning is finished before enough channels are removed. Therefore, in the proposed technique, when no more channels in any layer can be removed, the network is trained for a certain number of epochs without pruning. This causes a slight change in the value of the weight parameters at random, which may allow for further pruning. In addition, the proposed method combines dropout [16] to facilitate pruning.

In order to evaluate the performance of the above proposed method, we have performed channel pruning. We have evaluated the proposed approach for image classification by applying to VGG-16 [15] on CIFAR-10 and CIFAR-100 data sets [7]. We performed pruning of untrained and trained models using the proposed method. As a result, for CIFAR-10, we successfully reduced the size of the untrained and trained models to 2.1% and 3.1% of the original model, respectively. On the other hand, for CIFAR-100, we reduced the models to 16.0% and 89.0%, respectively. Also, starting the pruning from an untrained model tends to reduce more weights in the early stages of the iterative pruning, which indicates that the computation time of the pruning decreases more. This result well-relates to the lottery ticket hypothesis [2] and its subsequent results [3, 8]. In these studies, pruning is used to find essential sub-structures in the network to achieve good performance, called winning tickets. More specifically, the same initial values as before are assigned to the network compacted by pruning, and further pruning is performed. By repeating this process, the model is pruned while looking for a winning lottery ticket. On the other hand, the proposed approach based on the bonsai hypothesis shows that the pruning depends largely on the initial weights, not on the structure of the network model.

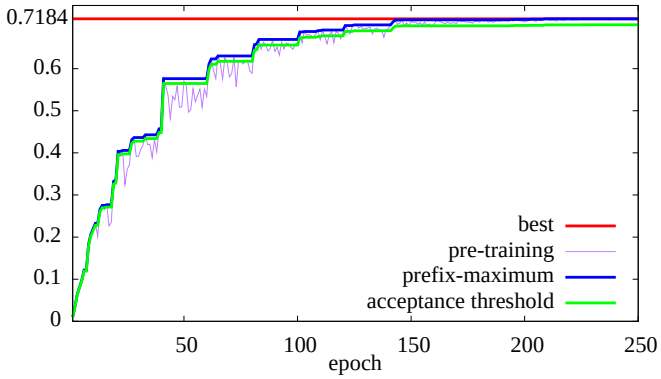
This paper is organized as follows. In Section 2, we propose a channel pruning approach based on the bonsai hypothesis. Section 3 shows the results of the channel pruning by the proposed approach. We evaluated the models with untrained models and already-trained models. Section 4 concludes our work.

2 Channel Pruning on the Bonsai Hypothesis

In this section, we propose the channel pruning algorithm on the bonsai hypothesis. For an input network model already trained, called *a target model*, this algorithm prunes the weights of the model. In the algorithm, the transition of the accuracy during the training of the target model is used to determine the acceptance threshold. More specifically, we record the value of the test accuracy at each epoch in the pre-training as shown in Figure 2. After that, the prefix-maximums of the test accuracy p_i $(i = 1, 2, \ldots)$ at epoch i are computed such that $p_i = \max_{1 \leq j \leq i} a_j$, where a_j denotes the test accuracy at epoch j obtained in the pre-training of the non-pruning model. In the following pruning algorithm, we use rp_i as the acceptance threshold values that are the lower limits of the test



(a) CIFAR-10



(b) CIFAR-100

Fig. 2. The test accuracy in pre-training with no pruning

accuracy at epoch i in training, where r ($r > 0$) is the acceptance coefficient. The algorithm prunes the model so that the test accuracy of the pruning model maintains at least rp_i. The proposed algorithm performs the following steps.

Step 1: Initialize weights of the model
Step 2: Prune channels by the exponential search for each layer
Step 3: If no channel is pruned in Step 2, retrain the model with no pruning
Step 4: Repeat Steps 2 and 3

The detail of each step is explained as follows.

According to our experiments, the pruning results depend on the initial values of weights. Therefore, it is necessary to run this algorithm several times to remove a sufficient number of weights. Hence, in Step 1, weights are initialized to have different initial values for each execution of this algorithm.

In Step 2, we perform training and pruning on the model for each layer. Algorithm 1 shows the details of this step. To determine the number of pruning

channels, we adopt *the exponential search*, which is a method to efficiently find the number of nodes in radio networks with a small number of trials when the unknown number of nodes [14]. This method is employed to obtain the channels to be pruned efficiently. Specifically, while the test accuracy of the pruned model satisfies the threshold, we repeat pruning by doubling the number of removed channels. After each pruning, since the test accuracy drops significantly, we retrain the model for a specific epoch to recover it. If the test accuracy fails to satisfy the threshold after retraining, the pruning is discarded and the model is restored to that before pruning.

Algorithm 1 Channel pruning using the exponential search for layer l

```
pruned ← 0
pruning ← 1
loop
    Backup the model
    k ← pruning − pruned
    Prune k channels with small L1-norm in layer l
    Retrain the pruned model for a certain epochs
    if the test accuracy can be accepted then
        pruned ← pruning
        pruning ← pruning × 2
    else
        Restore the model to discard the last pruning
        return
    end if
end loop
```

Step 3 retrains the model with no pruning for several epochs if no channel is pruned in Step 2. The aim of this step is to improve the accuracy of the model and to allow the model to be pruned by slightly changing the values of the parameters. We repeat Steps 2 and 3 until a certain amount of time elapses or a certain number of epochs of training is performed.

3 Experimental Results

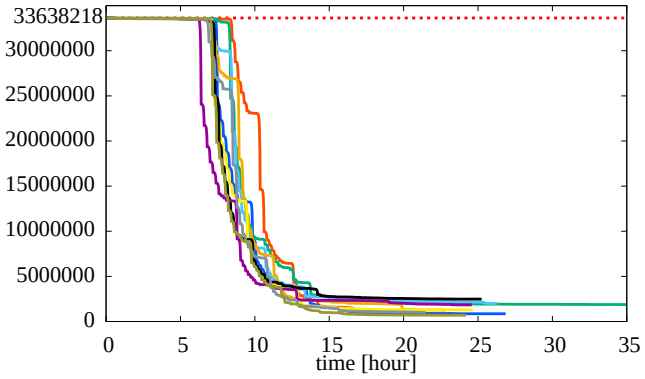
In this section, we show the experimental results to evaluate the proposed channel pruning approach for VGG-16[15]. VGG-16 is one of the popular deep convolutional neural networks that is mainly used image classification problems. The structure of VGG-16 consists of 13 convolutional layers and 3 fully-connected layers. This network is widely used as a benchmark model for machine learning because it has the typical structure of a convolutional neural network. We have trained models using two image classification problems, the CIFAR-10 and CIFAR-100 datasets [7]. The CIFAR-10 dataset consists of 60000 32×32 color images in 10 classes, with 6000 images per class. There are 50000 training images

and 10000 test images. On the other hand, the CIFAR-100 dataset consists of 100 classes containing 600 images each. There are 500 training images and 100 testing images per class.

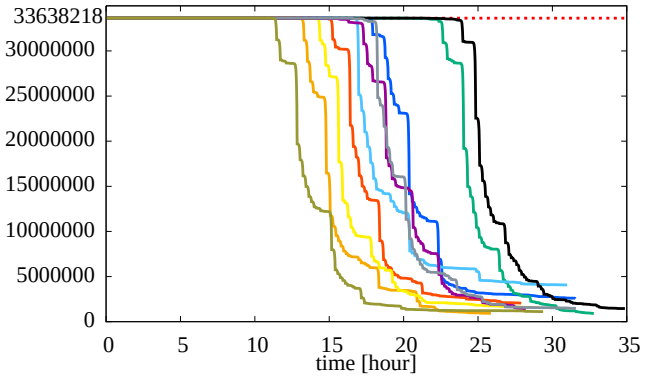
We have trained the VGG-16 models on CIFAR-10 and CIFAR-100 without pruning as pre-training to obtain the transition of the test accuracy. The networks have been trained for a sufficient number of epochs until no improvement in test accuracy is observed. In our experiment, we have trained the models for 250 epochs. Figure 2 shows the transition of the test accuracy in the pre-training. From the results, the test accuracy of the models on CIFAR-10 and CIFAR-100 achieves 0.936 and 0.718, respectively. In our experiments, for the first 250 epochs, we used 0.99 and 0.98 times of the prefix-maximums of test accuracy shown in Figure 2 as the threshold of acceptable accuracy for pruning. After 250 epochs, we also used the same values at 250 epochs as a threshold. The network models have been trained using momentum-accelerated mini-batch SGD with a batch size of 128 and momentum set to 0.9. Also, the learning rate has been set to 0.1 and decayed to 1/2 every 20 epochs.

For the purpose of confirming the bonsai hypothesis, we have pruned untrained models and already-trained models, called *scratch models* and *trained models*, respectively. Scratch models are initialized randomly and trained models are obtained by training in the above pre-training. For each execution, pruning is performed until the models have trained 2000 epochs. Note that if the test accuracy does not exceed the threshold in Step 2 and the pruning is discarded, the number of epochs at that time is not counted in the above 2000 epochs. For both models, we perform the proposed channel pruning, shown in Section 2, 10 times each. The settings for training are basically the same as for the pre-training above, except that the learning rate is fixed after 250 epochs. For the trained models, we start pruning after the pre-training. Specifically, the pruning is performed for a total of 2000 epochs from the 251st epoch to the 2250th epoch. For the recovery of the test accuracy after pruning a channel in Step 2, we have trained the models for 5 epochs. Also, in Step 3, we have performed the training with no pruning for 5 epochs when no layer is pruned in Step 2.

Figures 3 and 4 shows graphs of the total weights over time for 2000 epoch pruning on CIFAR-10 and CIFAR-100, respectively. From the graph of CIFAR-10, we can see that the scratch models can be pruned faster than the trained models, with less fluctuation and more stability. On CIFAR-100, the weights could not be pruned much for the trained models. On the other hand, for the scratch models, a large number of weights were pruned compared to the trained models, including some cases where the weights could not be reduced much. From the curves for successful pruning, a small number of weights are removed in the early stages of pruning. After that, each graph has a sudden exponential downward curve. This is due to the exponential search, which doubles the number of channels to be removed in the case of successful pruning. Since no such curve is observed in the graph where pruning was not successful, we can say that pruning is not successful without this exponential downward curve. Furthermore, in the scratch models, such a rapid descent starts at a small number of epochs. Thus,



(a) scratch models

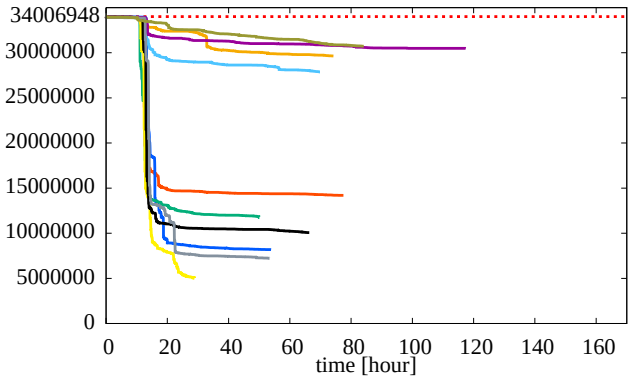


(b) trained models

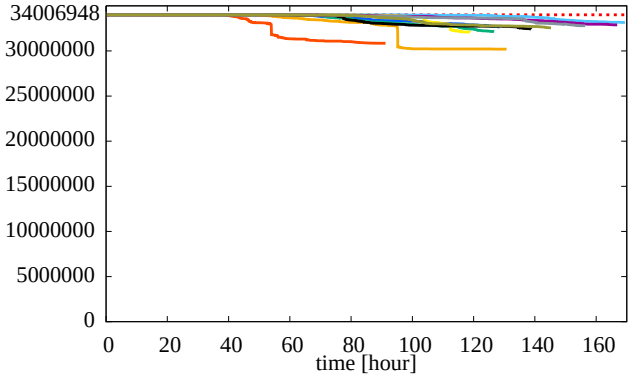
Fig. 3. The total number of weights of the model pruned from scratch and trained models for CIFAR-10

by starting pruning from the scratch model, the pruning is more successful and more effective in fewer epochs. Table 1 shows the total number of weights for the ten executions. According to this table, pruning from the scratch models is more efficient than that from the trained models both on CIFAR-10 and CIFAR-100. In addition, for CIFAR-100, it can be seen that when pruning is successfully performed, more weights can be reduced by pruning from the scratch models. Thus, these results support the bonsai hypothesis, i.e., model pruning should start from scratch models rather than trained models.

Tables 2 and 3 show the resulting number of channels and weights for each layer of VGG16 on CIFAR-10 and CIFAR-100, respectively. These tables show the results of the model with the most weights reduced when pruned from the scratch models. From the tables, we can see that for both CIFAR-10 and CIFAR-



(a) scratch models



(b) trained models

Fig. 4. The total number of weights of the model pruned from scratch and trained models for CIFAR-100

100, we can reduce more channels in the layer close to the output. In particular, more channels in the second layer of the fully-connected layers are pruned.

Next, we evaluate the computation time of the pruned models. Since the computation time depends on the execution environment, in this paper, the computation time is estimated by evaluating the number of multiplications required to evaluate the model. More specifically, we count the number of multiplications necessary to compute the convolutional layers and the fully-connected layers, not including operations except for multiplications and memory latency. From the pruned results in Tables 2 and 3, the number of multiplications is reduced 3.32×10^8 to 7.13×10^7 for CIFAR-10, while 3.32×10^8 to 1.80×10^8 for CIFAR-100. Thus, we can estimate that by the proposed method, the computing time of evaluating the pruned models is reduced to 21.1% and 54.1%, respectively.

Table 1. The resulting total number of weights of the pruned models for VGG-16 on CIFAR-10 and CIFAR-100

dataset	initial model	minimum	average	maximum
CIFAR-10	scratch	700,967 (2.1%)	1,738,962 (5.2%)	2,990,007 (8.9%)
	trained	1,033,042 (3.1%)	2,238,188 (6.7%)	5,951,868 (17.7%)
CIFAR-100	scratch	5,445,432 (16.0%)	17,896,110 (52.6%)	30,994,145 (91.1%)
	trained	30,260,638 (89.0%)	32,280,803 (95.0%)	33,203,464 (97.6%)

Table 2. The detailed results of the pruning on VGG16 for CIFAR-10

layer	size of	baseline model		pruned model	
type	feature map	#channels	#weights	#channels	#weights
conv 1	32×32	64	1.8×10^3	41	1.1×10^3 (64.06%)
conv 2	32×32	64	3.7×10^4	45	1.7×10^4 (45.09%)
conv 3	16×16	128	7.4×10^4	79	3.2×10^4 (43.43%)
conv 4	16×16	128	1.5×10^5	97	6.9×10^4 (46.80%)
conv 5	8×8	256	3.0×10^5	148	1.3×10^5 (43.82%)
conv 6	8×8	256	5.9×10^5	133	1.8×10^5 (30.04%)
conv 7	8×8	256	5.9×10^5	81	9.7×10^4 (16.44%)
conv 8	4×4	512	1.2×10^6	42	3.1×10^4 (2.60%)
conv 9	4×4	512	2.4×10^6	42	1.6×10^4 (0.67%)
conv 10	4×4	512	2.4×10^6	54	2.0×10^4 (0.87%)
conv 11	2×2	512	2.4×10^6	52	2.5×10^4 (1.07%)
conv 12	2×2	512	2.4×10^6	72	3.4×10^4 (1.43%)
conv 13	2×2	512	2.4×10^6	57	3.7×10^4 (1.57%)
linear 1	1	4096	2.1×10^6	22	1.1×10^4 (0.54%)
linear 2	1	4096	1.7×10^7	119	2.7×10^3 (0.02%)
linear 3	1	10	4.1×10^4	10	1.2×10^3 (2.93%)
total	—	—	3.4×10^7	—	7.0×10^5 (2.08%)

Thus far, several studies of channel pruning have been devoted. Table 4 shows the performance comparison between our work and the existing approaches [6, 9, 11, 19]. Unfortunately, since the parameter settings of the training and pruning differ, we cannot directly compare the performance with them. From the table, however, the proposed method achieves high compression with almost the same test accuracy. Thus, the bonsai hypothesis, pruning from scratch models, may be applied to existing methods to achieve further pruning.

4 Conclusions

In this paper, we have stated *the bonsai hypothesis* which is a hypothesis about network pruning that is analogous to bonsai: pruning can be more effective when starting from untrained models than already trained models. To support the hypothesis, we have presented a simple and efficient channel pruning algorithm. We performed pruning of untrained and trained models using the proposed algorithm for VGG16 on CIFAR-10 and CIFAR-100. As a result, we found that

Table 3. The detailed results of the pruning on VGG16 for CIFAR-100

layer type	size of feature map	baseline model		pruned model	
		#channels	#weights	#channels	#weights
conv 1	32×32	64	1.8×10^3	59	1.7×10^3 (92.19%)
conv 2	32×32	64	3.7×10^4	59	3.1×10^4 (85.00%)
conv 3	16×16	128	7.4×10^4	113	6.0×10^4 (81.40%)
conv 4	16×16	128	1.5×10^5	113	1.2×10^5 (77.94%)
conv 5	8×8	256	3.0×10^5	216	2.2×10^5 (74.50%)
conv 6	8×8	256	5.9×10^5	184	3.6×10^5 (60.65%)
conv 7	8×8	256	5.9×10^5	195	3.2×10^5 (54.76%)
conv 8	4×4	512	1.2×10^6	368	6.5×10^5 (54.76%)
conv 9	4×4	512	2.4×10^6	331	1.1×10^6 (46.47%)
conv 10	4×4	512	2.4×10^6	228	6.8×10^5 (28.79%)
conv 11	2×2	512	2.4×10^6	275	5.6×10^5 (23.92%)
conv 12	2×2	512	2.4×10^6	177	4.4×10^5 (18.57%)
conv 13	2×2	512	2.4×10^6	93	1.5×10^5 (6.28%)
linear 1	1	4096	2.1×10^6	1003	5.1×10^5 (24.49%)
linear 2	1	4096	1.7×10^7	225	2.3×10^5 (1.35%)
linear 3	1	100	4.1×10^5	100	2.3×10^4 (5.52%)
total	—	—	3.4×10^7	—	5.4×10^6 (16.01%)

Table 4. Performance comparison of different channel pruning methods to prune the VGG16 models

	CIFAR-10		CIFAR-100	
	test accuracy	remaining weights	test accuracy	remaining weights
ref. [6]	92.7%	16.0%	72.0%	35.2%
ref. [9]	93.1%	11.5%	71.6%	24.0%
ref. [11]	93.8%	11.5%	73.5%	24.9%
ref. [19]	93.8%	26.6%	73.3%	62.1%
this work	92.6%	2.1%	70.3%	16.0%

the untrained models tend to reduce more channels than the already-trained models.

References

1. Dhouibi, M., Ben Salem, A.K., Saidi, A., Ben Saoud, S.: Accelerating deep neural networks implementation: A survey. IET Computers & Digital Techniques **15**(2), 79–96 (2021). https://doi.org/10.1049/cdt2.12016
2. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: Proc. of International Conference on Learning Representations (2019)
3. Frankle, J., Dziugaite, G.K., Roy, D.M., Carbin, M.: Stabilizing the lottery ticket hypothesis. arXiv preprint arXiv:1903.01611 (2019)
4. Gale, T., Zaharia, M., Young, C., Elsen, E.: Sparse GPU kernels for deep learning. In: Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis (2020)

5. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. In: Proc. of the 28th International Conference on Neural Information Processing Systems. vol. 1, pp. 1135–1143 (2015)
6. Hu, Y., Sun, S., Li, J., Wang, X., Gu, Q.: A novel channel pruning method for deep neural network compression. ArXiv **abs/1805.11394** (2018)
7. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep. (2009)
8. Lange, R.T.: The lottery ticket hypothesis: A survey (2020), https://roberttlange.github.io/posts/2020/06/lottery-ticket-hypothesis/
9. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient ConvNets. In: 5th International Conference on Learning Representations. OpenReview.net (2017), https://openreview.net/forum?id=rJqFGTslg
10. Liang, T., Glossner, J., Wang, L., Shi, S.: Pruning and quantization for deep neural network acceleration: A survey. CoRR **abs/2101.09671** (2021), https://arxiv.org/abs/2101.09671
11. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: Proc. of the IEEE international conference on computer vision. pp. 2736–2744 (2017)
12. Luo, J.H., Wu, J., Lin, W.: ThiNet: A filter level pruning method for deep neural network compression. In: Proc. of the IEEE international conference on computer vision. pp. 5058–5066 (2017)
13. Matsumura, N., Ito, Y., Nakano, K., Kasagi, A., Tabaru, T.: A novel structured sparse fully connected layer in convolutional neural networks. Concurrency and Computation: Practice and Experience p. e6213. https://doi.org/10.1002/cpe.6213
14. Nakano, K., Olariu, S.: Uniform leader election protocols for radio networks. IEEE Transactions on Parallel and Distributed Systems **13**(5), 516–526 (2002). https://doi.org/10.1109/TPDS.2002.1003864
15. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations (2015), http://arxiv.org/abs/1409.1556
16. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
17. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE **105**(12), 2295–2329 (2017). https://doi.org/10.1109/JPROC.2017.2761740
18. Wang, Z.: SparseRT: Accelerating unstructured sparsity on GPUs for deep learning inference. In: Proc. of the ACM International Conference on Parallel Architectures and Compilation Techniques. pp. 31–42 (2020)
19. Zhao, C., Ni, B., Zhang, J., Zhao, Q., Zhang, W., Tian, Q.: Variational convolutional neural network pruning. In: Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2780–2789 (2019)