SCIREA Journal of Information Science
and Systems Science
http://www.scirea.org/journal/ISSS
December 21, 2021
Volume 5, Issue 6, December 2021
https://doi.org/10.54647/isss12188

Accelerating SegNet-Based Semantic Segmentation Using a Model Post-Pruning Strategy

Wei Liu

School of Business Administration, Shandong University of Finance and Economics, Jinan 250014, China.

Correspondence should be addressed to Wei Liu; vivian.liu@sdufe.edu.cn

Abstract

Accelerating deep convolutional networks has recently attracted a great deal of attention due to the demand of real-time applications. SegNet is a typical deep convolution network in the field of semantic segmentation, and also is a smaller and more memory, time efficient model. In this paper, we focus on accelerating the SegNet-based semantic segmentation by using a model post-pruning strategy. Despite the fact that several methods have been proposed for accelerating deep models including pruning and compressing the weights of each layer, these methods may cause a certain loss of segmentation accuracy by irregular pruning. To address this issue, we propose a post-pruning strategy for deep model compression, which is commonly used to essentially deal with the over-fitting problem of decision tree. Different from some existing methods that employ the irregular pruning strategy, the proposed method can significantly improve the generalization ability of the compressed model. Inspired by the post-pruning method originally used in decision tree, our method minimizes a channel pruning loss function. The compressed model is then retrained to further improve its performance of semantic segmentation. Experimental results on two segmentation datasets

show that our method obtain competitive results compared with other existing methods in terms of reducing the computational burden and improving the generalization ability of the SegNet model.

Keywords: Deep learning, Convolutional neural networks, Semantic segmentation, Model pruning

1. Introduction

Semantic segmentation of natural images has attracted a tremendous interest in the computer vision and image processing community, which is widely used in some tasks such as scene understanding, autonomous driving, medical image analysis, and so on. Deep convolutional neural network (DCNN) has achieved state of the art breakthroughs in image semantic segmentation task thanks to its high ability of feature extraction. In recent years, there have been many deep models for semantic segmentation, including FCN [1], SegNet [2], U-Net [3], DeepLab [4], RefineNet [5], etc. Among them, the SegNet model is more suitable than others for real-time applications due to its low memory occupancy and computational time during inference. SegNet is composed of a symmetric structure of the encoder and decoder. Compared with the above mentioned network models, the key component of SegNet is the position index recorded by the maximum pooling layer of the encoder, which effectively improves calculation efficiency and decoding accuracy. The pooling indices with location information is transferred to the decoder and then used in the upsampling process. The location information transmission is illustrated as dotted lines in Figure 1. Despite the fact that SegNet shows faster performance than other semantic segmentation models, a shortcoming of SegNet is the information discarded by the maximum pooling layer of the encoder, causing a loss of network segmentation accuracy. However, in high-precision real-time semantic segmentation tasks such as autonomous driving, scene understanding and inferring support-relationships among objects, it is quite important to improve segmentation accuracy as well as speed.

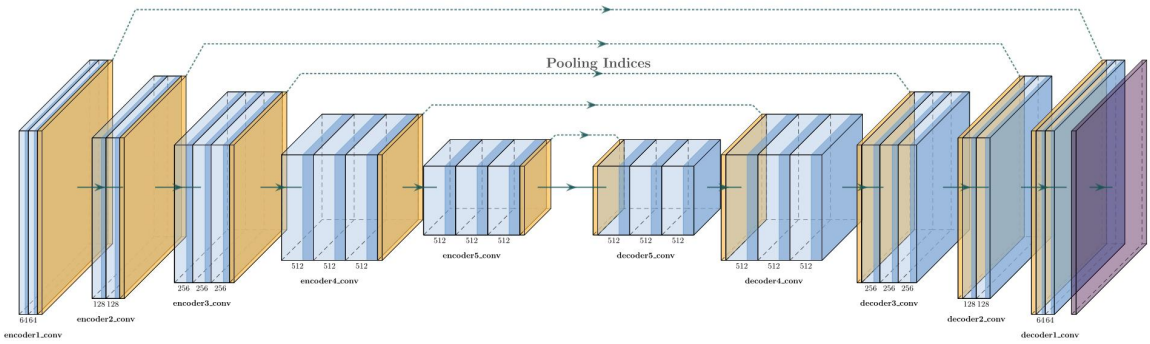


Figure 1: The structure diagram of SegNet. The orange layers correspond to pooling layers and up-sampling layers; the light blue layers denote convolutional layers; the dark blue layers mean batch normalization layers and relu layers; the purple ones are the classification layers; and the dotted lines represent pooling indices.

Generally speaking, the computational complexity of DCNNs is determined by the number of the parameters of convolutional layers. Therefore, most network acceleration methods focus on reducing the computational burden of convolutional layers to improve the computational efficiency of deep models [1, 8]. At present, the commonly used strategies for model acceleration can be divided into three categories: tensor factorization [9, 16, 17], sparse connection [10], and channel pruning [6, 12]. Tensor factorization of convolutional layers aims to replace convolutional layers in the original deep convolutional network with multiple low-rank subtensors, and model parameters in the resulting new deep network can be effectively compressed compared with the original one. However, the width of convolutional layers after tensor decomposition has not changed, and it even brings additional computational costs during the decomposition process [14, 15]. Unlike tensor factorization, sparse connection is to remove redundant connection weights and obtain the smallest possible model structure. However, due to the irregular channel shape after weight sparsification, it brings some challenges to further accelerate DCNNs by other possible techniques, especially for parallel implementations. Channel pruning is a specific structured case of sparse connection, which performs model compression by removing redundant the channels with small contributions in the network. Specifically, after pruning a layer filter, the number of input channels of the next layer filter is also reduced relatively [7, 11]. Therefore, channel pruning method has received extensive attention from researchers due to its effectiveness. The key issue of channel pruning is how to select the pruning channel to maintain the network segmentation accuracy.

In this paper, we employ a post-pruning strategy to accelerate the SegNet, a state-of-the-art network in fast semantic segmentation. Post-pruning is a widely used strategy for pruning decision tree to improve the overall performance of decision tree. We apply it to prune the SegNet channel for accelerating the SegNet model. Moreover, we retrain the model to update weights of the pruned model, which can reduce the computational cost and improve the generalization ability and accuracy of the SegNet model.

The main contributions of this paper are as follows.

1) We propose a post-pruning strategy for the SegNet acceleration, which minimizes a loss function and introduces LASSO regression to achieve pruning channel selection. Our channel pruning method not only effectively compresses the width of convolutional layers, reducing the computational cost, but also improves the segmentation accuracy and generalization ability of the SegNet.

2) We retraining the model to update weights after pruning. Through the global adjustment of convolution weights of the SegNet model, it can effectively improve the discriminativeness of the model after channel pruning.

3) We construct a footpath scene dataset called "EyeCan" for the footpath segmentation task, which focuses on image segmentation of footpaths and obstacles. In addition, we verify the presented post-pruning method on the CamVid road scene segmentation dataset and the EyeCan footpath scene dataset. Our method shows better generalization ability, as well as the improvement of segmentation speed and segmentation accuracy.

2. Related Work

This paper mainly focuses on accelerating the SegNet model by adopting a post-pruning strategy used in decision tree, which can effectively sparsify and speed up a deep model. In this section, we review some related works to accelerate deep model, i.e., model sparsification and post-pruning decision tree methods.

2.1 Model Sparsification

Recently, a variety of acceleration methods by sparsifying model structure have been proposed in the literature. Most related works attempt to transform the channel pruning into a sparse optimization problem [9-11]. One prominent method for accelerating deep model is the structured sparse learning method (SSL) [9], which relies on the group sparsity on

convolution weights to generate a sparse structure. In the training process, SSL exploits the group LASSO to achieve channel pruning by sparsifying the filter weights. Another similar method uses group sparsity to sparsify the model by introducing regularization sparsification into the training loss function [10]. It is worth noting that this method automatically determines the number of neurons in each layer of a deep network while learning the network parameters. Instead of explicitly solving LASSO, the above methods integrate sparsity regularization into the training loss. He et al. [11] applied the scale factor of the normalization layer to directly evaluate the importance of filters, using a two-steps iterative pruning strategy, which includes LASSO regression-based channel selection and least squares method to reconstruct convolution weights. This strategy does not introduce additional selection factors. In practice, an independent two-steps solution has been introduced to reduce the computing time for solving the iterative pruning problem [11]. But it makes the selection of pruning channels unstable, which leads to a loss of accuracy.

Although the above-mentioned methods for channel pruning can improve the calculation speed of DCNNs, they may reduce the segmentation accuracy to some extent. However, for real-time semantic segmentation tasks, improving model accuracy is important as well as accelerating the model. The main goal of this paper is to accelerate the SegNet model by channel pruning to obtain a more sparse structure. After the model channel pruning is conducted, if we update weights based on the feature maps of the pre-trained model, the discrimination of the pruned model will depend on the pre-trained model and ignore the discrimination of the new network. Most of existing pruning methods perform layer-by-layer by minimizing a loss function, which can only ensure local optimization of each layer. In this paper, an alternative method is given to retraining the model, which can make the model achieve the effect of global optimization.

2.2 Post-pruning Using in Decision Tree

Intuitively, the more complex the decision tree, the better the classification effect of the training data. However, the higher the complexity of the decision tree, the more likely to cause a over-fitting problem. Post-pruning is an important strategy to solve over-fitting of decision trees. Commonly-used pruning strategies include reduced-error pruning, pesimistic-error pruning, cost-complexity pruning, and error-based pruning. The work most closely related to ours and actually inspires us is the CART tree [18]. The CART tree algorithm, a post-pruning algorithm based on cost complexity, shows that decision trees are often pruned by minimizing an overall loss function or cost function. Based on the cost complexity

algorithm to guide the SegNet post-pruning process, conveniently, we construct the indicator vector β to mark the pruning channel. Briefly, in the cost complexity algorithm, if the leaf node before pruning does not improve the accuracy very well but brings more complexity, we can consider cutting this leaf node. In this way, the validation set data can be used to selectively prune the decision tree, effectively preventing the decision tree from over-fitting. This means that the post-pruning method can more safely remove redundant information and keep key information in the decision tree. Therefore, we explore a post-pruning method for SegNet compression, which can effectively improve the generalization ability and the speed of SegNet.

In this paper, we employ the idea of post-pruning to accelerate the SegNet model, and propose a method for selecting pruning channels by minimizing a loss function and introducing the LASSO regression. Furthermore, we present a pruning and retraining once strategy to update the weights and optimize the model globally during the retraining process. The proposed strategy can effectively accelerate model, improving the global accuracy and generalization ability of the compressed SegNet model, which has been verified in subsequent experiments.

3. Post-Pruning Method

We focus mainly on a post-pruning method to prune channels of the SegNet. In Figure 2, we highlight the channel input feature maps of the next layer convolutional filters that will be reduced with orange dotted boxes, where the SegNet structure is effectively sparsified by post-pruning. More importantly, the key strategy of channel pruning is how to select pruning channels. Inspired by a post-pruning strategy used in decision tree, we select pruning channels which maintain the verification set accuracy after pruning and minimize the pruning loss function.

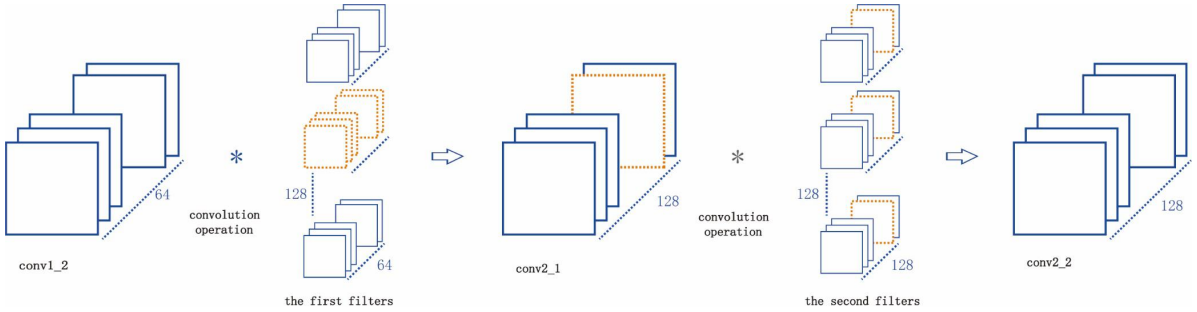


Figure 2: Illustrate of post-pruning method on the SegNet second encoder. The orange layer is the pruned channel and feature maps. If the channel group of the first layer is pruned, the corresponding feature maps of the second layer will be removed.

3.1 Overview

The acceleration strategy of the SegNet model is implemented in the following three steps.

1) Single-layer pruning. We conduct a single-layer pruning experiment for a pre-trained SegNet model. According to the sensitivity of the convolutional layer to channel pruning, the rough range of channel pruning of each layer is empirically determined.

2) Layer-by-layer channel pruning. The input maps vector of a channel in the next convolutional layer is generated by the channel corresponding to the current convolutional layer. We use statistical information of the next convolutional layer and the pruning rate determined by the single-layer experiment to guide the pruning of the current convolutional layer with ensuring the global accuracy of network pruning.

3) After channel pruning, the model segmentation error is minimized through retraining, which improves the segmentation accuracy and generalization ability of the SegNet model.

Figure 2 illustrates the process of single-layer pruning on conv2_1, where we highlight with orange dotted boxes that the channel input feature maps of the conv2_2 will be reduced after a part of conv2_1 channels are pruned. Note also that single-layer experiments verify the sensitivity and applicability of our channel pruning method on the SegNet model.

3.2 A Post-pruning Strategy of the SegNet

Decision trees usually use pruning redundant information to mitigate the over-fitting problem, which has been proved to be an efficient and effective tool. The major challenge in SegNet training is over-fitting. Considering that the over-fitting problem of both SegNet and decision tree are essentially caused by redundant information, we apply the post-pruning method on SegNet to select pruned channels. Thanks to the simplicity and effectiveness of the post-pruning method, we not only decrease the complexity of the model to compress and

accelerate the model, but also prevent the model from over-fitting, resulting in the improvement of the generalization ability and segmentation accuracy of the model.

The key issue of the post-pruning strategy used in decision tree is to minimize a loss function. When this strategy is applied on the channel pruning of DCNNs, the pruned deep model is determined by minimizing the loss function of convolutional filters layer by layer. Formally, given a sequence of each layer for DCNN channels, i.e., $c_i, i = 1, \ldots, n$, the task of DCNN pruning is to generate directive sequence, $\beta_i, i = 1, \ldots, n$, where $\beta_i \in \{0,1\}$ denotes whether the *i*-th channel of the convolutional layer is pruned.

Our goal is to minimize the loss of the feature maps of the current layer and those of the next layer after pruning. We set β as the pruning channel selection vector and define the loss function for the first layer channel pruning of the model as follows:

$$C_\lambda(\beta_i) = C(\beta_i) + \lambda \mid \beta_i \mid,$$

where $C(\beta_i)$ is the prediction error of the *i*-th convolutional layer of the model on the training data, which is the sum of the prediction errors of all channels; λ is the regularization parameter that is used to balance the accuracy and complexity of the SegNet. The larger the λ, the greater the extent of network compression and the more pruning, and vice versa.

The pruning channel selection strategy depends on the minimization of the loss function $C_\lambda(\beta_i)$ of the pruning process. In practice, minimizing this loss function quickly is a challenging task. In [11], an alternative way to solve this problem is to transform the problem into a two-steps iterative problem, which firstly fixes *W* and solves β for channel selection, and secondly fixes β and solves *W* to minimize the loss function. But in practical applications, the two-steps iteration is time-consuming. Considering further reduction of the computational cost as well as improvement of the global segmentation accuracy of the pruned SegNet, instead of iterative computation, our way to solve this problem is to exploit LASSO regression to compute β layer by layer, and perform a pruning and retraining once strategy to update W. LASSO is more easily to obtain a higher sparse solution, which is widely used for model selection [20, 21].

$$C_\lambda(\beta) = \arg\min_\beta \left\| Y - \sum_{i=1}^{c} \beta_i Y_i^{'} \right\|_F^2 + \lambda \left\| \beta_i \right\|_1 \quad s.t. \quad \left\| \beta \right\|_0 \le c^{'},$$

where $Y_i^{'} = X_i W_i^T$. Y is the feature maps of the pre-trained model of the next layer of the pruning layer, $Y^{'}$ is the feature maps of the next layer after channel pruning. $X_i, i = 1, \ldots, n$ is a matrix sliced from *i*-th channel of input volumes X . W_i are convolutional filter weights sliced from *i*-th channel of W . Notice that, if $\beta_i = 0$, X_i will be no longer useful, which could be pruned from feature maps. W_i can also be removed. $\|\cdot\|_F$ is Frobenius norm, $c^{'}$ is the number of channels to be retained in the current layer, and its value is determined during single-layer pruning.

After the channel pruning being completed, we use a pruning and retraining once strategy to prune away the redundancy of the network and improve the global accuracy. We retrain the model to update weights for a short period of time, which is less than the original training time. In the following experiments, the results show that the discriminative performance of the SegNet model by using retraining for updating weights has been significantly improved.

4. Experiments

In this section, we evaluate the proposed approach for the SegNet [2] on the CamVid [13] and EyeCan datasets. Our method is implemented in MATLAB and executed on a desktop with an Intel Core E5-2643 v4 @ 3.4 GHz and 256 GB RAM.

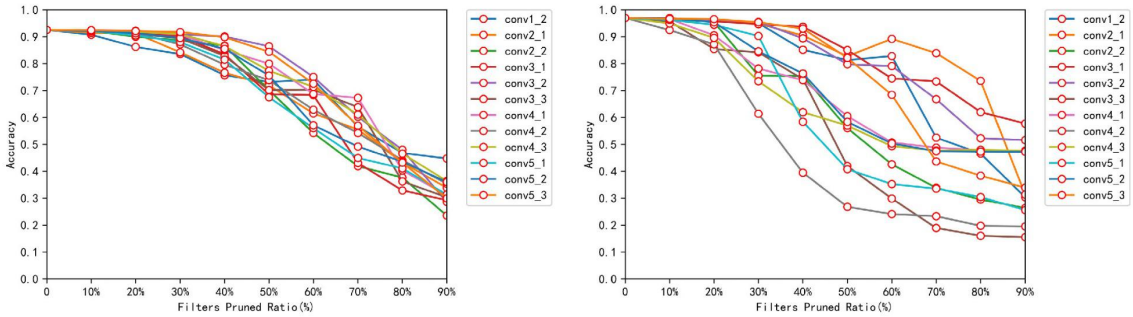
To determine the range of pruning channels, we first perform a single-layer sensitivity experiment. According to the guidance of the proposed channel selection strategy and single-layer experimental results, we prune convolutional filters layer by layer. Finally, we retrain the pruned SegNet model to minimize the segmentation error and update the convolutional layer parameters to improve the segmentation accuracy. Moreover, we conduct comparative experiments with other pruned methods on the CamVid and EyeCan datasets. Experimental results clearly show the benefits of the channel pruning strategy by the post-pruning method and verify that the segmentation accuracy and generalization ability of the accelerated SegNet model have been improved by our method.

4.1 Datasets

To evaluate the performance of the accelerated SegNet model, we employ the CamVid road scene segmentation dataset [13] which contains 701 natural images and correspondingly labelled images with 960*720 pixels, and the EyeCan footpath scene dataset which contains

112 natural and corresponding labelled images with 540*960 pixels. The former dataset is mainly used to test semantic segmentation performance of the model in the field of autonomous pilot. In addition, guiding the blind is an important application in the field of scene semantic segmentation, while there is currently no public dataset for the outer scenes of blind people in our knowledge. Therefore, we constructed the EyeCan dataset to test the semantic segmentation performance of the model in the field of blind footpath scenes. We use handheld devices to capture footpath scene images, including crosswalks, highways, and green belts at three time periods in the morning, noon and afternoon, respectively. In order to reflect the completeness of the EyeCan dataset, the dataset contains abnormal weathers such as backlight and rainy. The pixel annotation of the dataset is labelled by the Image Labeler Module of MATLAB. Figure 3 shows several typical images of CamVid dataset and EyeCan dataset.

Figure 3: Experimental datasets. The top row shows images from CamVid, and the bottom row shows images from EyeCan.



(a) CamVid (b) EyeCan

Figure 4: Single-layer sensitive results on different datasets.

4.2 Implementation Details

We make a unified processing of datasets avoid the negative impact of different initializations on the accuracy of experiments. Each input image is first resized to 360*480. In order to prevent the segmentation accuracy decreasing owing to unbalanced data classes, we use class weights to balance the number of pixels in each class in the dataset. To enrich datasets, we

use data argument by using random reflection of left/right directions and translation of +/-10 pixels. In training process, we empirically set learning rate to 0.001 and batchsize to 4, which usually works well. Moreover, SegNet is initialized with VGG-16 weights. The optimization algorithm used for training is stochastic gradient descent with 0.9 momentum. We fix these parameters in the following experiments.

4.3 Single-layer Sensitivity Analysis

We first conduct a single-layer experiment to evaluate the sensitivity of each layer of the SegNet, which is helpful for comprehensive understanding the model robustness to exactly prune channel. Then, we use single-layer sensitivity and experimental verification to roughly determine the range of the number of pruning channels in each layer.

We prune each layer of the SegNet on the CamVid dataset and the EyeCan dataset, and use the same filters pruned away ratio for all layers in the same stage. As shown in Figure 4, for each sensitive pruning layers (conv3_3, con4_2, conv2_2 and conv5_1) of the EyeCan dataset, we can prune the ratio of 20%-30% without losing accuracy. Specially, the sensitivity of each layer is approximately similar in CamVid dataset, but the convolutional layer with 512 channels is more sensitive. We determine the sensitivity of the pruning channel based on the single-layer pruning experiment, and set the pruning complexity empirically based on the sensitivity and a large number of test results. Tables 1 and 2 show the parameter settings for each layer of SegNet, including the size of the convolutional layer, the number of channels, and the pruning ratio. It is worth noting that, due to higher pruning sensitivity, we indispose the first convolutional layer in the SegNet network.

Table 1: Parameter settings of SegNet pruned model on CamVid.

Prunable layer	Convolutional Layer size	Channel numbers	Filters pruned away
Conv1_2	3*3*64	64	80%
Conv2_1	3*3*64	128	80%
Conv2_2	3*3*128	128	80%
Conv3_1	3*3*128	256	80%
Conv3_2	3*3*256	256	80%
Conv3_3	3*3*256	256	80%
Conv4_1	3*3*256	512	75%
Conv4_2	3*3*512	512	75%

Conv4_3	3*3*512	512	75%
Conv5_1	3*3*512	512	75%
Conv5_2	3*3*512	512	75%
Conv5_3	3*3*512	512	75%

Table 2: Parameter settings of SegNet pruned model on EyeCan.

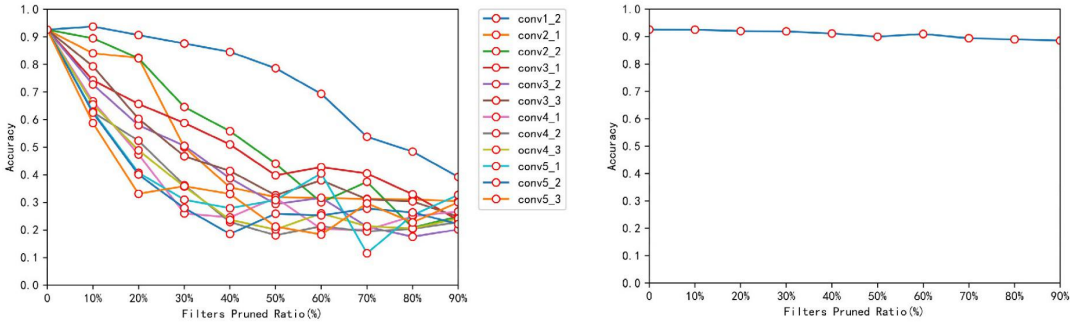
Prunable layer	Convolutional Layer size	Channel numbers	Filters pruned away
Conv1_2	3*3*64	64	80%
Conv2_1	3*3*64	128	80%
Conv2_2	3*3*128	128	25%
Conv3_1	3*3*128	256	80%
Conv3_2	3*3*256	256	80%
Conv3_3	3*3*256	256	25%
Conv4_1	3*3*256	512	60%
Conv4_2	3*3*512	512	25%
Conv4_3	3*3*512	512	60%
Conv5_1	3*3*512	512	25%
Conv5_2	3*3*512	512	60%
Conv5_3	3*3*512	512	60%

4.4 Pruning Performance Analysis

According to the range of pruning channel numbers determined by sensitivity experiment and post-pruning method, we prune the SegNet model layer by layer. After pruning the SegNet with different pruning rates, we obtain a new model with a few channels. The remaining parameters in the pruned convolutional layer and the unaffected layer are copied to the new model. Note also that if a channel is cut off, the weight of the subsequent batch normalization layer is also removed. Then, we improve the segmentation accuracy of the SegNet model by retraining the pruned model to update weights of convolutional layers and reconstruct parameters of batch normalization layers. In the retraining process, we adopt the same pre-processing methods and hyperparameter values as the SegNet pre-training model. Finally, we obtain a SegNet compression model based on the post-pruning method.

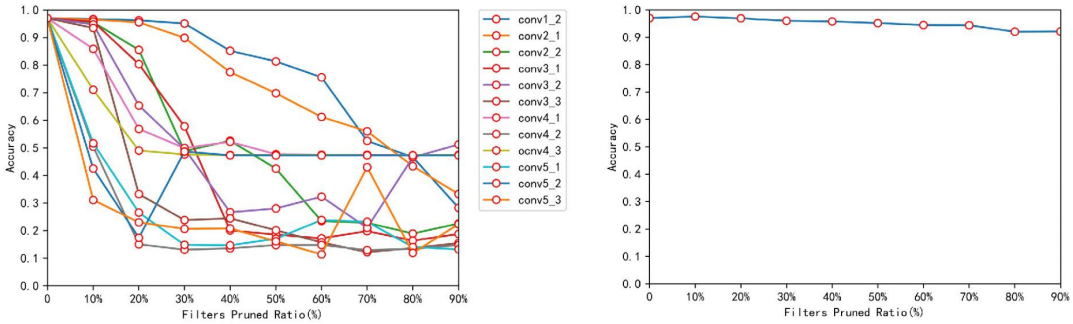
Figure 5a shows the complete pruning experiment results on CamVid that the convolutional layer with 512 channels (conv5_1, conv5_3, conv5_2) are sensitive with pruning and the first

three layers are more suitable for pruning. The accuracy of the model drops quickly when we prune more strongly with the all layers except the first three layers. The reason may be that the significant information loss from previous layers hurts the accuracy. Figure 5b shows that through retraining, almost 90% of the convolutional layer channels can be safely removed. Similar experiments were performed on the EyeCan dataset, and the results are shown in Figure 6a. Similar observations are also found for the EyeCan dataset. Without retraining, the first two convolutional layers may prune easily, which can be pruned by 25% without loss of accuracy. Figure 6b demonstrates that through retraining, almost 90% of the channels of these layers can be safely removed, so retraining the network may compensate the loss of accuracy caused by pruning.



(a) Model accuracy after pruning (b) Model accuracy after retraining

Figure 5: Pruning SegNet filters with post-pruning method on CamVid.



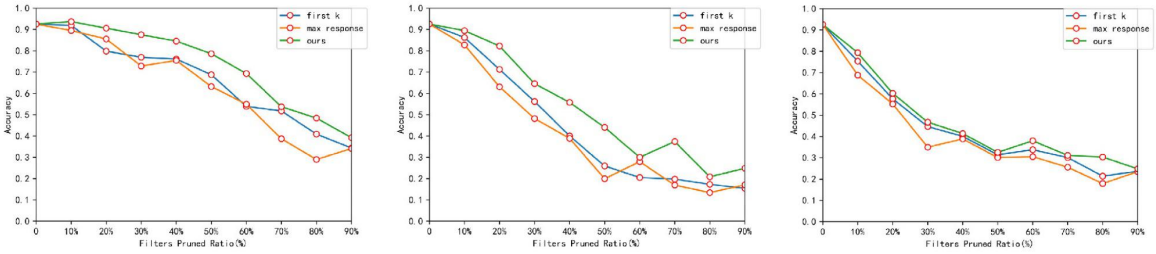
(a) Model accuracy after pruning (b) Model accuracy after retraining

Figure 6: Pruning SegNet filters with post-pruning method on EyeCan.

4.5 Comparison with Fisrt *k* and Max Response Methods

In the following experiments, we compare the key pruning channel selection strategy of post-pruning with max response and frist *k* pruning methods. First *k* directly selects the first *k* feature maps. Max response selects channels based on absolute sum of corresponding weights filter [19]. In order to make a fair comparison, all methods are pruned using the same SegNet model on the same computer.

As expected, the accuracy decreases as filters pruned ratio increases. As shown in Figure 7, our method is consistently better than other approaches in different convolutional layers under different filters pruned ratio. Figure 8 also demonstrates that our method has higher segmentation accuracy at the footpath edge. Figures 7 and 8 show that sometimes max response is even worse than first k. We argue that filters in our method are regularized and converge to a smoother and more natural pattern. The maximum response ignores the correlation between different filters that is why max response does not work better than first k.



(a) conv1_2 (b) conv2_2 (c) conv3_2

Figure 7: Single-layer sensitive analysis results of different pruning methods on the CamVid dataset (without retraining). To verify the importance of channel selection, we compare with two baselines such as first *k* and max response methods.

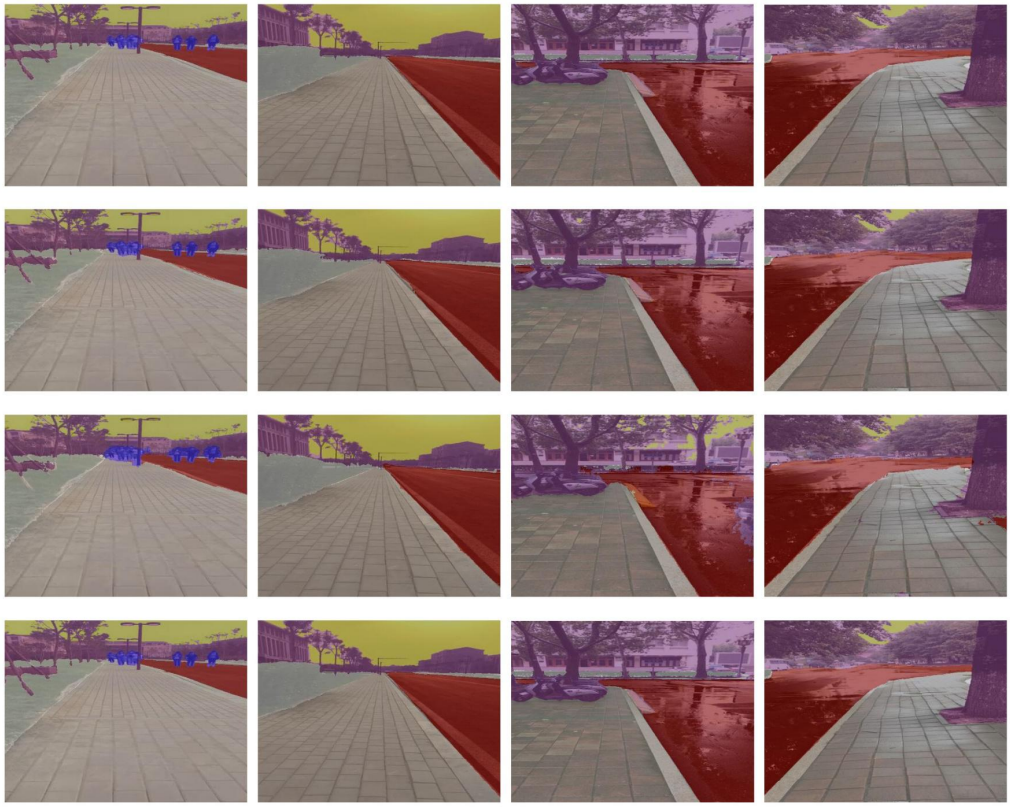


Figure 8: Semantic segmentation results of different SegNet variants. From top rows to bottom rows: SegNet baseline, Pruned SegNet by the first k method, Pruned Segnet by max response, Pruned SegNet by the proposed post-pruning.

Table 3: Performance of accelerated SegNet on the CamVid dataset.

Pruning ratio	Accuracy	CPU(s)	Acceleration ratio
0%	92.565%	44.495	-
30%	91.287%	41.465	6.8%
30%	90.544%	39.765	10.63%
50%	89.469%	39.683	11.18%
70%	87.758%	39. 519	10.81%

5. Conclusion

Channel pruning is a commonly-used method for compressing DCNNs which could structurally sparsify network models. Previous works mainly focused on reducing model size and faster training, but paid little attention to the improvement of model generalization ability and segmentation accuracy. In addition, these works have caused a slightly performance degradation. We propose a post-pruning model acceleration method, which achieves a significant speed improvement with almost no performance loss. However, due to SegNet being a more compact and less redundancy, it is more challenging to prune a large amount of filters of the SegNet model. In spite of this, our method can still obtain a 11.18% acceleration with 89.469% accuracy in Table 3, which proves the feasibility of post-pruning method for accelerating tasks.

Our method is not incompatible with other methods. We demonstrate competitive acceleration results on CamVid and EyeCan datasets. Therefore, SegNet has the ability to handle high-precision real-time semantic segmentation tasks in terms of speed and accuracy. In the future, we plan to further extend our method to accelerate other DCNNs.

Data Availability

The data used to support the findings of this study are available from the author upon request.

Funding Statement

This research was funded by National Natural Science Foundation of China (Grant No. 61472220), the Science and Technology Innovation Program for Distinguished Young

Scholars of Shandong Province Higher Education Institutions (Grant No. 2019KJN045), and the Humanities and Social Sciences in Universities of Shandong Province (Grant No. J16WJ04).

References

[1] J. Long, E. Shelhamer, T. Darrell, "Fully convolutional networks for semantic segmentation," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[2] V. Badrinarayanan, A. Kendall, R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[3] O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, 2015.

[4] L.-C. Chen, Y. Zhu, G. Papandreou, et al., "Encoder-decoder with atrous separable convolution for semantic image segmentation," In *Proceedings of the European Conference on Computer Vision*, pp. 801–818, 2018.

[5] G. Lin, A. Milan, C. Shen, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1925–1934, 2017.

[6] S. Anwar, K. Hwang, W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 3, pp. 1–18, 2017.

[7] M. Jaderberg, A. Vedaldi, A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," In *Proceedings of the British Machine Vision Conference*, pp. 1–12, 2014.

[8] S. Han, J. Pool, J. Tran, et al., "Learning both weights and connections for efficient neural network," In *Advances in Neural Information Processing Systems*, pp. 1135–1143, 2015.

[9] W. Wen, C. Wu, Y. Wang, et al., "Learning structured sparsity in deep neural networks," In *Advances in Neural Information Processing Systems*, pp. 2074–2082, 2016.

[10] J.M. Alvarez, M. Salzmann, "Learning the number of neurons in deep networks," In *Advances in Neural Information Processing Systems*, pp. 2270–2278, 2016.

[11] Y. He, X. Zhang, J. Sun, “Channel pruning for accelerating very deep neural networks,” In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

[12] Z. Huang, N. Wang, “Data-driven sparse structure selection for deep neural networks,” In *Proceedings of the European Conference on Computer Vision*, pp. 304–320, 2018.

[13] G.J. Brostow, J. Shotton, J. Fauqueur, “Segmentation and recognition using structure from motion point clouds,” In *Proceedings of the European Conference on Computer Vision*, pp. 44–57, 2008.

[14] M. Denil, B. Shakibi, L. Dinh, “Predicting parameters in deep learning,” In *Advances in Neural Information Processing Systems*, pp. 2148–2156, 2013.

[15] E.L. Denton, W. Zaremba, J. Bruna, “Exploiting linear structure within convolutional networks for efficient evaluation,” In *Advances in Neural Information Processing Systems*, pp. 1269–1277, 2014.

[16] V. Lebedev, Y. Ganin, M. Rakhuba, “Speeding-up convolutional neural networks using fine-tuned cp-decomposition,” In *Proceedings of the International Conference on Learning Representations*, pp. 1–11, 2015.

[17] X. Zhang, J. Zou, X. Ming, “Efficient and accurate approximations of nonlinear convolutional networks,” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1984–1992, 2015.

[18] L. Breiman, J. Friedman, C.J. Stone, Olshen, R.A. *CART*: *Classification and Regression Trees*, Sprinter, 1984.

[19] H. Li, A. Kadav, I. Durdanovic, et al., “Pruning filters for efficient convnets,” In Proceedings of the International Conference on Learning Representations, pp. 1–11, 2016.

[20] S. Anwar, W. Sung, “Compact deep convolutional neural networks with coarse pruning,” In *Proceedings of the International Conference on Learning Representations*, pp. 1–10, 2016.

[21] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society: Series* , vol. 58, no. 1, pp. 267–288, 1996.