IEICE TRANS. INF. & SYST., VOL.E102–D, NO.12 DECEMBER 2019

LETTER

A Spectral Clustering Based Filter-Level Pruning Method for Convolutional Neural Networks

Lianqiang LI[†], ***Student Member***, **Jie ZHU**[†a)], ***and*** **Ming-Ting SUN**[††b)], ***Nonmembers***

SUMMARY Convolutional Neural Networks (CNNs) usually have millions or even billions of parameters, which make them hard to be deployed into mobile devices. In this work, we present a novel filter-level pruning method to alleviate this issue. More concretely, we first construct an undirected fully connected graph to represent a pre-trained CNN model. Then, we employ the spectral clustering algorithm to divide the graph into some subgraphs, which is equivalent to clustering the similar filters of the CNN into the same groups. After gaining the grouping relationships among the filters, we finally keep one filter for one group and retrain the pruned model. Compared with previous pruning methods that identify the redundant filters by heuristic ways, the proposed method can select the pruning candidates more reasonably and precisely. Experimental results also show that our proposed pruning method has significant improvements over the state-of-the-arts.

key words: *convolutional neural network, spectral clustering, filter-level, pruning*

1. Introduction

CNNs have played central roles in a lot of artificial intelligence applications [1]–[3]. However, in order to gain satisfying performance, CNNs are usually set to have millions or even billions of parameters, which make them hard to be deployed into mobile devices. As larger CNNs with more parameters are considered, lessening the complexity of CNNs is imperative.

Several works have been done trying to solve this issue. Among them, network pruning [4]–[7] are the promising ways to effectively reduce the redundancy of CNNs. They can be roughly classified into connection-level pruning and filter-level pruning. The connection-level pruning method [4] would incur irregularly sparse structures which need the support of specialized libraries and devices to accelerate the pruned CNNs. In contrast, filter-level pruning methods [5]–[7] can not only remove more parameters in one time but also result in regularly sparse structures which can be easily implemented using existing software and hardware. Although the existing filter-level pruning methods have gained achievements to some extent, there are still some drawbacks as we will describe in Sect. 2.

Manuscript received June 5, 2019.
Manuscript revised August 23, 2019.
Manuscript publicized September 17, 2019.
[†]The authors are with the Dept. of Electronic Engineering, Shanghai Jiao Tong University (SJTU), Shanghai, 200240, China.
[††]The author is with the Dept. of Electrical and Computer Engineering, University of Washington. Seattle, 98105, USA.
a) E-mail: zhujie@sjtu.edu.cn (Corresponding author)
b) E-mail: mts@uw.edu (Corresponding author)
DOI: 10.1587/transinf.2019EDL8118

In this work, we propose a novel filter-level pruning method to compress and accelerate the CNN models. Our proposed method consists of three main modules. First of all, we construct an undirected fully connected graph to represent a pre-trained CNN model. The vertices in the graph stand for the filters while the edges in the graph are labeled with the "similarity" among the filters. Secondly, we employ the spectral clustering algorithm [8] to divide the graph into some subgraphs, which is equivalent to clustering the similar filters of the CNN into the same groups. After gaining the grouping information among the filters, we retain one filter for each group, and retrain the pruned model to make up for the accuracy loss due to the pruning. Experimental results show that our proposed pruning method has significant improvements over the state-of-the-arts [5]–[7].

2. Related Works

In this section, we review some representative filter-level pruning methods.

Li et al. [5] proposed that the filters with lower absolute weight sum should have less "importance" and can be pruned away. This method is straightforward and can be easily implemented. However, its pruning strategy may be not reasonable. For example, we cannot consider that the following Sobel kernel is less important than the Laplacian kernel, in spite of the absolute weight sum in the Sobel kernel is smaller than that of the Laplacian kernel.

$$Sobel = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, Laplacian = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (1)$$

In fact, every filter should have its own function in capturing features no matter what the scale of its absolute weight sum is. Moreover, Ye et al. [9] have empirically demonstrated that the filters with small absolute weight sum also have an important influence on neural network performance.

Abbasi-Asl et al. [6] introduced the Classification Accuracy Reduction (CAR) to guide pruning process. Specifically, they first removed one filter for one time and recorded the final performance without the filter. Then they used the degeneration of performance to assess the "importance" of the filter. At last, they removed the inessential ones according to CAR. This method seems to be more convincing. However, such a greedy filter-level pruning method needs a lot of computation. Besides, identifying which filter has

a greater impact on large networks is very difficult as many filters have similar or even the same CAR.

Instead of evaluating the filters in an "importance" approach, our previous work [7] employed the clustering algorithm, i.e., K-means++ [10] to explore the "similarity" among the filters and abandon the homogeneous ones. We supposed that the filters with similar functions in extracting features could be clustered into the same groups, and pruning the similar filters would not hurt the performance of CNNs. This method works well in small networks. However, the filters in large networks may be linear inseparable in the Euclidean Space employed by K-means++ as the filters usually have thousands of attributes, which would bring a negative influence on the performance of the pruned CNN models.

3. The Proposed Method

In trying to develop a better filter-level pruning strategy, we recognize that defining an "importance" measure for filters is meaningless because each filter has its own feature extraction preference. Furthermore, the existing "similarity" based methods are too naive to distinguish the similar filters in large networks. As a result, we put forward to a spectral clustering based filter-level pruning method in this paper. The intuition is that the filters with similar representations would introduce redundancy to the CNN models, and the spectral clustering algorithm is more reliable to identify the "similarity" among the filters.

Taking an example of layer L, the parameters of it are denoted as W. W consists of N 3-D filters $W_i \in R^{C\times h_L\times w_L}$ $(1 \le i \le N)$, where C is the number of input channels, and h_L and w_L stand for the spatial height and width of the filters, respectively. We first project the N filters $(W_1, \ldots, W_i, \ldots, W_j, \ldots, W_N)$ into a new feature space where the filters could be more spatially distinct. In practice, we employ the following Gaussian kernel function.

$$A_{ij} = exp(-\frac{\|W_i - W_j\|^2}{2\sigma^2}) \tag{2}$$

where A_{ij} can be interpreted as the degree of "similarity" between W_i and W_j, and σ is the size of the Gaussian kernel. Therefore, $A \in R^{N\times N}$ is an affinity matrix. After that, an undirected fully connected graph $G = (V, E)$ is constructed. Here, V represents the filters while E depicts the similar relationships among the filters, i.e., A_{ij}.

Next, we cut G into K subgraphs $(G_1, \ldots, G_m, \ldots, G_K)$. The goal is to partition the filters into different groups to achieve that the filters in the same groups are similar to each other while the filters in different groups are dissimilar to each other. As a result, the objective is minimizing the following equation.

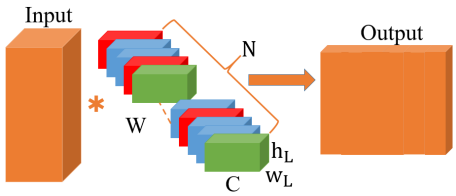
$$Cut(G_1, \ldots, G_m, \ldots, G_K) = \sum_{m=1}^{K} \sum_{W_i\in G_m, W_j\in \overline{G_m}} A_{ij} \tag{3}$$

where $\overline{G_m}$ is the complement of G_m. In addition, we do not know the distribution of the number of filters with respect to the function in capturing features. It is natural to assume that every kind of filter in capturing features owns similar amounts. Therefore, Eq. (3) is rewritten as follows.

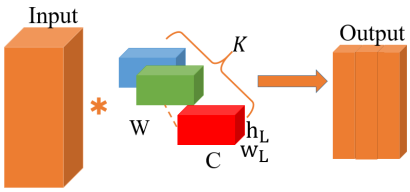
$$Ncut(G_1, \ldots, G_m, \ldots, G_K) = \sum_{m=1}^{K} \frac{\sum_{W_i\in G_m, W_j\in \overline{G_m}} A_{ij}}{\sum_{W_i\in G_m} D_i} \tag{4}$$

where $D_i = \sum_{j=1}^{N} A_{ij}$, and $D = diag(D_1, \ldots, D_i, \ldots, D_N)$. In fact, minimizing Eq. (4) is a problem of normalized graph cut in graph theory, and several works [8], [11] have proven that it is equivalent to carrying out the following three steps. 1) compute the first p eigenvectors of U, where $U = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, 2) form a matrix $F \in R^{N\times p}$ from the eigenvectors and normalize it by rows to get $F_n \in R^{N\times p}$, and 3) employ K-means to divide the F_n into K groups by rows. It is worth noting that p and K are different numbers in theory. However, in practice, people usually use $p = K$, and we follow this setting in our work.

After gaining the grouping information among the filters, we keep the filter nearest to the centroid in each group and prune out the others, as shown in Fig. 1. To compensate for the possible accuracy drop from the pruning process, we retrain the pruned model as the training phase.



(a) After clustering, similar filters are clustered into the same groups(makred by the same colors).



(b) After pruning and retraining, only the filter closest to the centroid in each group is kept.

Fig. 1 The proposed process in a convolutional layer.

4. Experiments and Results

In this section, we validate the effectiveness of our proposed method with three state-of-the-arts from the literature [5]–[7] by pruning VGG-16 [12] on CIFAR-10. The overall architecture of VGG-16 is shown in Fig. 2. The accuracy baseline of VGG-16 on CIFAR-10 is 92.25%. In order to explore the impact of σ in Eq. (2), we set three different sizes, i.e., 0.1, 1, and 10. For fair comparisons, all experiments[†] are carried out in Caffe [13].

[†]More details are in https://github.com/shiyuetianqiang

Fig. 2 The overall architecture of VGG-16. VGG-16 has five convolutional blocks, the convolutional layers in each block have the same number of filters, i.e., conv1_x have 64 filters, conv2_x have 128 filters, conv3_x have 256 filters, conv4_x have 512 filters, conv5_x have 512 filters.

Table 1 Accuracy performance of VGG-16 on CIFAR-10 under different pruning methods.

Methods	Accuracy(%)		
	25% PR	50% PR	75% PR
Weight sum [5]	91.64	90.10	85.37
CAR [6]	91.70	90.14	85.49
K-means++ [7]	91.83	90.21	86.02
Ours(σ=0.1)	91.83	90.17	85.93
Ours(σ=1)	92.15	90.57	86.74
Ours(σ=10)	92.23	90.89	87.45

Table 1 presents the accuracy performance of the pruning methods under different pruning ratios (PRs) after retraining. PR is defined as the ratio of the number of removed filters to the number of original filters, which is shown in Eq. (5).

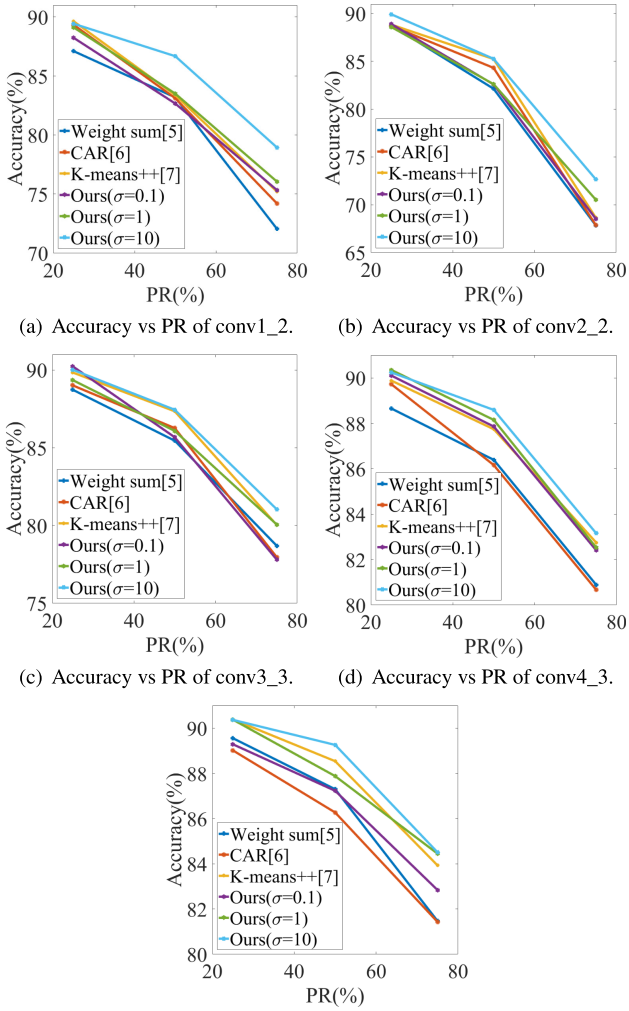
$$PR = \frac{N - K}{N} \tag{5}$$

It is possible to analyze the pruning sensitivity of each layer and decide the PR for them separately. However, for not introducing too many hyper-parameters, we set the same PR for all layers.

We can see from Table 1 that there is a trade-off between the final accuracy performance and the PR. We can also observe that our proposed method is superior to the others in general. Especially for the proposed method with σ of 10, its final accuracy performance is always the best. The final accuracy performance of [7] is close to that in our method with small σ. As for [5] and [6], the accuracy performance of them is a bit of disappointing.

The performance is attributed to pruning strategies. For learning the "similarity" among filters, our proposed method utilizes the spectral clustering algorithm to map the filters into a new feature space where those filters are more different to each other and are easy to be distinguished. Therefore, compared with [7], the proposed method gains better accuracy performance. As for [6], it prunes the filters starting from these with the least impact on the resultant accuracy. However, the CAR of the remaining filters may change dramatically from the initial values after some filters have been removed due to the complex dependence among them. Consequently, its performance is unsatisfying. Although the criterion in [5] is straightforward, the ignorance of the functionalities of the filters with small absolute weight sum leads to the worst performance.

A good pruning method should not only have better final accuracy performance after retraining but also deserves a better accuracy performance without retraining. As there are 13 convolutional layers in VGG-16, we only choose some representative layers to validate the effectiveness of our proposed method. Figure 3 presents the trade-off curves between the accuracy and the PR under different pruning methods before retraining.



(a) Accuracy vs PR of conv1_2. (b) Accuracy vs PR of conv2_2.

(c) Accuracy vs PR of conv3_3. (d) Accuracy vs PR of conv4_3.

(e) Accuracy vs PR of conv5_3.

Fig. 3 Trade-off curves for accuracy and PR of some representative convolutional layers on VGG-16 under different pruning methods before retraining.

It is obvious that the performance of our proposed method is still outstanding under different PRs before retraining. Furthermore, the proposed method with the σ of 10 owns the highest accuracy. We can also find that the performance of [7] is sometimes comparable to that in our method with small σ. As far as the [5] and [6] are concerned, their performance is relatively lower.

These can be explained as follows. Our pruning method manages to partition the filters into groups so that the filters of a group are similar to each other and dissimilar to others outside of the group. Discarding the filters in one group just enforces the CNN model to abandon the

Table 2 Clustering evaluation results of layer conv5_3 of VGG-16.

Methods	Types	Average Distance		
		25% PR	50% PR	75% PR
Ours (σ=0.1)	Intra-difference	0.0481	0.0530	0.1485
	Inter-difference	0.5948	0.6634	0.7259
Ours (σ=1)	Intra-difference	0.0144	0.0508	0.1227
	Inter-difference	0.5965	0.6678	0.7467
Ours (σ=10)	Intra-difference	0.0144	0.0469	0.1216
	Inter-difference	0.5982	0.6707	0.7439

filters with similar functionality in extracting features and would not bring serious damages to the accuracy performance even without retraining. The K-means++ algorithm in [7] is not good at clustering the filters with high dimensions, it may cluster the filters with different functionalities into one group. Consequently, the performance of [7] before retraining ranks the second class. As for [5] and [6], they fail to analyze the "similarity" among the filters, and their pruning strategies may reduce the diversity of feature-extraction functions provided by original CNN model. As a result, their poor accuracy performance without retraining is inevitable.

We also conducted clustering evaluation experiments to check the quality of the resultant clusters. We first computed the average distance within a cluster, and then computed the average value of the above distance among different clusters to denote the intra-difference. We also computed the average distance among different centroids to denote the inter-difference. Table 2 presents the clustering evaluation results of conv5_3 as an example.

According to Table 2, we can observe that the intra-difference decreases and the inter-difference increases when the proposed method with larger σ, which means the clusters are denser and more well separated. Besides, we can find that both the intra-difference and the inter-difference gradually rise with the increase of PR for different σ. It is because that more distinct filters are forced into the same clusters with the increase of PR which naturally increases the intra and inter differences. The clustering evaluation results are consistent with the results in Table 1 and Fig 3, and also demonstrate that the proposed method works well.

Besides, we checked the number of filters in each cluster. The results show that most clusters contain the similar number of filters as desired. Also, a few clusters have a large number of filters, which indicates that our algorithm could effectively put the similar filters (more redundant filters) into the same clusters.

5. Conclusion

In this paper, we present a novel spectral clustering based filter-level pruning method to lighten the complexity of CNNs. The proposed method can select the pruning candidates more reasonably and precisely. Besides, as it belongs to the filter-level pruning methods, it results in regularly sparse structures which can be easily implemented using existing software and hardware. Experimental results demonstrate that the proposed pruning method has apparent improvements over the state-of-the-arts.

For future work, we would like to explore the possibility of further improving the "similarity" measure and investigate the combination of different pruning methods to achieve the optimal pruning results for lessening CNN models. We would also like to do more research on the effect of pruning on other applications besides image classification.

Acknowledgments

This work is supported by the National Key Research Project of China under Grant No. 2017YFF0210903, the National Natural Science Foundation of China under Grant Nos. 61371147 and 11433002 and China Scholarship Council. Thanks to Kevin Anderson for helping with English language editing and checking.

References

[1] C.S. Wicramasinghe, K. Amarasinghe, and M. Manic, "Deep self-organizing maps for unsupervised image classification," IEEE Transactions on Industrial Informatics, 2019.
[2] J. Gao, T. Zhang, X. Yang, and C. Xu, "P2t: Part-to-target tracking via deep regression learning," IEEE Transactions on Image Processing, vol.27, no.6, pp.3074–3086, 2018.
[3] Y. Zhang, Z. Zhang, D. Miao, and J. Wang, "Three-way enhanced convolutional neural networks for sentence-level sentiment classification," Information Sciences, vol.477, pp.55–64, 2019.
[4] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," Advances in Neural Information Processing Systems, pp.1135–1143, 2015.
[5] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H.P. Graf, "Pruning filters for efficient convnets," ICLR, 2017.
[6] R. Abbasi-Asl and B. Yu, "Structural compression of convolutional neural networks based on greedy filter pruning," CoRR, vol.abs/1705.07356, 2017.
[7] L. Li, Y. Xu, and J. Zhu, "Filter level pruning based on similar feature extraction for convolutional neural networks," IEICE Transactions on Information and Systems, vol.E101-D, no.4, pp.1203–1206, 2018.
[8] A.Y. Ng, M.I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," Advances in Neural Information Processing Systems, pp.849–856, 2002.
[9] J. Ye, X. Lu, Z. Lin, and J.Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," ICLR, 2018.
[10] J. MacQueen et al., "Some methods for classification and analysis of multivariate observations," Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, pp.281–297, Oakland, CA, USA, 1967.
[11] U. Von Luxburg, "A tutorial on spectral clustering," Statistics and computing, vol.17, no.4, pp.395–416, 2007.
[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," Proceedings of the 22nd ACM international conference on Multimedia, pp.675–678, ACM, 2014.