Structure Level Pruning of Efficient Convolutional Neural Networks with Sparse Group LASSO

Aakash Kumar, Baoqun Yin, Ajeet Kumar Bhatia, Aneel Kumar Bhatia, and Avinash Rohra

***Abstract*—The rapid progress of convolutional neural networks (CNNs) in multiple applications of practical implementation is generally hindered by an upsurge in network size and computational complexity. Currently, engineers focus on reducing these problems through compressing the CNNs by pruning filters and their weights. In this paper, we present a fresh and easy-to-use pruning approach that reduces the model size by eliminating complete filters and filter weights based on the sparse group LASSO (Least Absolute Shrinkage and Selection Operator) method across the convolutional layers. More precisely, it regulates the sparsity at the feature level and the group level. During the process of pruning, the unnecessary filters with their weights eliminate directly without sacrificing accuracy in the test, resulting in much compact and slimmer architectures. We experimentally compute the effectiveness of our methodology with various state-of-art CNN models on various benchmark data sets. Mainly, CIFAR-10 data sets applied on VGG-16 model and reduce the parameters approx. 96.1% and saved approx. 83.55% float-point-operations (FLOPs) without sacrificing accuracy and have obtained development in state-of-art.**

***Index Terms*—Convolutional neural networks, filter pruning, FLOPs, sparse group LASSO.**

I. Introduction

Deep convolutional neural networks (CNNs) have been effective in several computer vision problems including image generation [1], object detection [2], [3], image segmentation [4], [5], natural language processing [6], image processing [3], and robotic control [7] due to the effectiveness of graphics processing units (GPUs) in the last few years. CNN's have a wide and deep structure; therefore, it requires a huge parameter storage memory and computational cost. Thus, a process of reducing the size of CNN's is needed to embed CNNs into embedded hardware.

Amongst several approaches of compressing CNN's including knowledge distilling [8], matrix decomposition [9], [10], weight quantization [11], and pruning [12], [13]. Currently, the pruning approach that selects and removes redundant parameters without considerably corrupting the model performance has been progressively researched. Early approaches for pruning are generally for fully connected layers (FC), for instance, second-order derivatives [14] and optimal brain damage [15]. The Second Order Derivatives presented applying the second derivative as a process for calculating the significance of units in the FC layer. Mariet [16] presented an approach of the searching subset of distinct units that do not require to be fine-turned and outcome in shrinkage in model redundancy. The main deficiency of the above-mentioned approach is that pruning units do not significantly reduce the computation time, as we know that most of the unnecessary units are not from deeper layers where the cost of computation is reasonably high.

Pruning approaches can be sorted out into two groups: filter pruning [17] and weight pruning [18]. The filter pruning approach chooses unnecessary filters and deletes them entirely, on the other hand, the weight pruning approach straightly deletes redundant weights of the layer. However, the weight level pruning approach suffers the non-structured sparsity in the weight matrix, which is not saving memory and computational cost, because it needed separate Basic Linear Algebra Subprograms (BLAS) libraries. In contrast, the filter pruning permits the model without damaging structured sparsity and vital to save memory than weight pruning, thus, it does not require separate BLAS libraries or hardware. As pruning often guides the network to be sparse, it is normal to include penalty term on the objective functions, and it is known as penalty induced sparsity [19]. The mathematics behind LASSO (the least absolute shrinkage and select in operator) is to build an L1-penalty norm for achieving a sparse network, where many parameters are forced to be 0 for sake of feature selection [20], [21]. Past methods [22] have highlighted L1-norm-based methods for model pruning. In contrast with other approaches for structure pruning, the penalty-induced method can induce the number of filters towards zero and then prunes filters with zero values naturally [23]. Considering group LASSO as a selection function, the author [24] considers filters or channels in each convolutional layer as separate ‘’group’’, which have to be penalized. Better to mention, L2-norm is generally accepted for regularization on vectors of group weight. While the network approaches converge, it is broadly adopted that the methods like LASSO should penalize many filters to keep penalizing until they approach zero and not letting their values increase.

In the last few years, the hype of using sparse group lasso (SGL) has been increased. The sparse group lasso method is presented by [25]. The sparse group LASSO is pool of regularization approaches, combination of the group LASSO and the LASSO. The sparse group LASSO penalty produces a results that obtains the between- and within- group sparsity concurrently. It allows to powers sparsity at the features level and groups level concurrently. The platform is given by the

Manuscript received November 2021; revised April 4, 2022.

Aakash Kumar and Baoqun Yin are with the University of Science and Technology of China, Hefei 2300026, P.R. China (e-mail: akb@mail.ustc.edu.cn, bqyin@ustc.edu.cn).

Ajeet Kumar Bhatia and Avinash Rohra are with Nanjing University of Aeronautics and Astronautics, Nanjing, P.R. China (e-mail: ajeet@nuaa.edu.cn, avinashrohra5@gmail.com).

Aneel Kumar Bhatia is with the University of Sindh, Jamshoro, Pakistan (e-mail: akumar.bhatia@usindh.edu.pk).

SGL, as in group lasso and lasso, always have the number of sparse predictor weights, however, several parameters in the solution are precisely zero. The fact is that SGL is more advantageous over lasso when the predictor weights are grouped, as lasso regularizes all the parameters of the solution evenly. On the other hand, SGL decides amongst groups and, it can also calculate the sparsity within each group, unlike group lasso. It can be considered, that SGL can contribute a vital part in covering up the problem of model compression in CNNs, where one filter can be considered as one group, and weights of the filter as group members.

Furthermore, sparsity methods are proven to be very significant in feature selection, regression learning, and classification, also for ensemble pruning [26]. In this article, we present a sparse group lasso-based approach which is a combination of group lasso and lasso [27] The general idea of the method is illustrated in the Fig. 1. This approach aids to achieve filter selection as a group as well as produce a solution that obtains between filters and within filters sparsity concurrently. This method is advantageous for the filter selection of CNN models and each filter highlighted with a collection of scores is considered as a group. Moreover, our work is also concerned with recognizing vital filter groups along with vital scores within the chosen filters.

The article is structured as: Section II contains the current related work considering structure pruning, section III gives the deep insight of the methodology used in the article, in section IV, the complete experiment is described with all the settings been used in experiments and the pruned models and their results shown with graphs. Finally, the last section concludes the article with future work in detail.

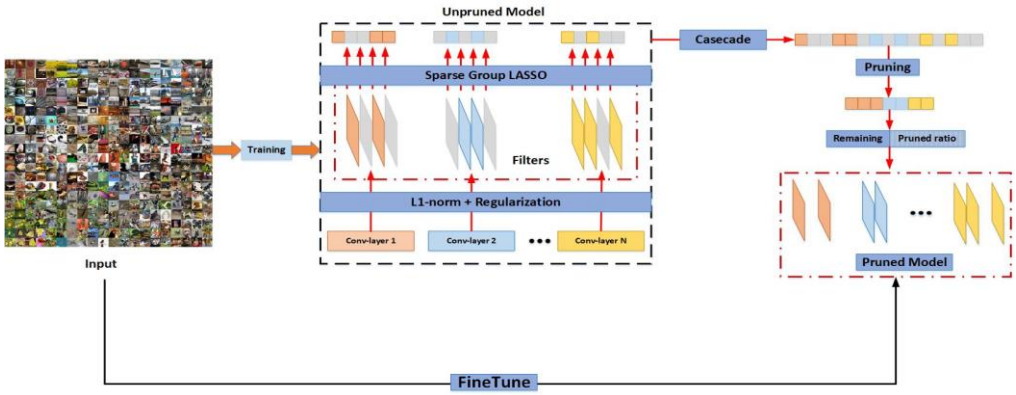


Fig. 1. A general idea of the introduced approach: in the initial step, the L1-norm is computed of each filter of each layer, after that the sparse group LASSO is applied on selected filters and weights of the filters, next all the filter values cascaded from all the layers. Finally, we prune weights of the filters according to the values of λ_1 and λ_2

II. Related Work

Pruning based on weight removes needless weight connections with a low score between neural network layers. Numerous methods based on weight pruning have been presented to prune unnecessary connections. Presently, the author in [28] proposed a method of pruning to eliminate connections whose entire scores are smaller than the predefined par score. The par score is measured by applying the standard deviation of weights of the layer. The model is, afterward, fine-tuned to overcome the accuracy loss. Furthermore, a compression approach is introduced by Louizos [29], which uses L-0 sparsity regularization. However, due to the uneven model structure of these approaches after pruning unable to achieve real speed gain without dedicated libraries or hardware.

Meanwhile, several filter level pruning approaches for eliminating the complete filter have been presented to maximize the real speed of CNN models. The initial filter level pruning is based on norm approaches to remove the filters with a small value of norm. He et al. [30] presented an approach to choose a filter with an L2 norm function with softly pruning the chosen filters, while Li et al. [17] presented an approach for pruning filter with a small value of L1 norm. In [31], the author presented an approach that measures the filter value in the batch normalization layer by the L1 norm criterion. Molchanov [32] calculated the significance of the feature maps by applying them to perform the square of the multiplication on the weight value and calculate the filter gradients as the vital value and measured global filter pruning in the descending order using First-Order Taylor Expansion. Aketi [19] calculated the feature-related value, then perform backpropagation on it, and finally, apply filter pruning globally with minimum related values.

In [33], the sparse deep neural network was introduced. This proposed method induced sparsity-induced constraints to penalize sparsity in network parameters while the network being trained. After the training, the basic model is pruned after applying some set of par scores to analyze insignificant weight values. Unfortunately, the compression results in unstructured pruning, and therefore despite good reported results are not applicable as mentioned earlier. On the other hand, group LASSO is applied for learning structured sparsity in convolutional neural networks, surprisingly, outcomes are suggested that they significantly obtain network speed up for the duration of inference without any alternative convention algorithms [24]. In a similar kind of work [22], sparse group sparsity was introduced. The work is the same in contrast with [33], which is, penalizing unstructured pruning among network parameters. Once more, the complete outcome suffers from unstructured sparsity and therefore, leads to the above-mentioned disadvantages. Furthermore, in [34], the LASSO penalizing is applied for CNN channel selection and then consequent pruning. This

method produced slim architecture with speedup inference and compacted model size.

In this article, we emphasized a modified version of the LASSO penalty, also known as group LASSO penalty in the linear regression task [35], which is significantly applicable to this end. A formula of group LASSO can be applied to enforce sparsity on a group level, for instance, all the members in a group are either all set to 0, or none of them are set. An additional modification, known as the sparse group LASSO, can also be applied to enforce further sparsity on the group of non-sparse variables [36]. In this article, we use this method by assuming a single feature map as a single group and weights in a feature map as group members. In this way, the optimization method can be used to delete the complete feature map as well as selected weights in the feature map. Depending on the particular feature map, we achieve different outcomes, such as feature selection when deleting a feature map and pruning when deleting features. The method of group L1 norm in machine learning is fairly well-known including convex loss function, multi-kernel, and multitask problems. However, as far as we know, this kind of general formula was never backed in the CNNs literature, excluding few particular cases. For instance, the author in [37], applied a group sparse approach to choosing groups of features co-occurring in a robotic control problem.

III. Method

The representation of our study is to produce a simple structure to achieve filter pruning. Initially, we provide a method to measure the feature maps and weights of feature maps. After that, we introduce tactics for feature map selection with filter pruning. Recently, filter pruning has been in the attention of researchers [13] that calculates the importance of the filter by capped L1- norm, scale factors, or Shannon entropy. The L1- norm regularization assists two tasks: 1. It penalizes sparsity for filter selection, 2. It controls the overfitting as well. Additionally, the L1-regularization can be applied directly in most current software libraries, and as compared to the classical weight decay method, it does not grow the computational cost.

Generally, CNN is considered a feedforward architecture made of several convolutional layers. We apply F_l and M_l to illustrate the number of filters and channels for $l-th$ convolution layer weights $W^{(l)} \in \mathbb{R}^{M_l \times F_l \times K \times K}$, where W shows the set of all weights in the CNN ($W = \{W^1, W^2, \dots, W^L\}$), and K represent the kernel size in the architecture. Furthermore, L is the number of CNN layers. For suitability, $W_{i,j}^l$ denotes a 2D kernel in the $i-th$ channel of the $j-th$ filter for the $l-th$ convolutional layer. Finally, The $i-th$ layer of architecture W^l should be denoted for channels as $\{W_{i,:}^l : 1 \le i \le M_l\}$, and for filters as $\{W_{:i}^l : 1 \le i \le F_l\}$. When a pruning filter is taking place, its related feature maps are pruned too, and compression is done in the $i-th$ layer. The filters in the coming convolutional layer are pruned too due to the kernels has used in pruned feature maps in the last layer, and it also saves an extra computation operation in the $i+1-th$ layer.

As we are aware that the LASSO cannot accept the group information and chooses a subgroup of features from all groups. Further, the Correlated variables are selected by the elastic net. On the other hand, the subset of the groups is selected by group LASSO. In filters, F_{ij} is defined by values of s. Each filter F_i have d feature maps. Hence, the feature maps of the filter shape have a natural group structure. Every feature map links to a group and every group has s sub-feature maps. While we want to select feature maps for a filter, it is necessary to handle each feature map with s sub-feature maps as a unit when picking an significant filter. And we desire to delete some redundant feature maps in each filter.

According to the above-mentioned explanation, we have chosen sparse group LASSO to solve the filter selection problem. If we consider the weights $w = \{w_1, w_2, \dots, w_d\}$ as feature maps of filter d, and w_i have s values. Then we have $w_i = \{w_{i1}, w_{i2}, \dots, w_{is}\}$. Hence, sparse group LASSO can be described as:

$$L = \min_w \frac{1}{2} \| Xw - Y \|_2^2 + \lambda_1 \| w \|_1 + \lambda_2 \sum_{i=1}^{d} \|w_i\|_2 \quad \text{s.t. } \lambda_1 > 0, \lambda_2 > 0 \tag{1}$$

Whereas L is represented to be the squared loss, and X, Y shows train input and output, while w is trainable parameters. The second expression regulates the sparsity in the feature level, finally, the last expression regulates the sparsity in the group level. For instance, if the value of λ_1 is set to zero, then the expression should be called group LASSO, and when the value of λ_2 is set to zero, the expression should be known as LASSO.

Afterward the weights w of feature maps and sub-feature maps are achieved, the most naive approach is that only feature maps and sub-feature maps with non-zero weights are picked. Now we apply $\widehat{w}$ to define that the feature is picked or not. Further, the size of $\widehat{w}$ is the similar as that of w. $\widehat{w}$ achieved by:

$$\widehat{w}_{ij} = \begin{cases} 0 & \text{if } w_{ij} = 0 \\ 1 & \text{others} \end{cases} \tag{2}$$

whereas $i = 1,2, \dots, d$ and $j = 1,2, \dots, s$. Once the sum of absolute of w_i is equivalent to 0, then the weights of the $i-th$ feature map are completely 0. Thus, the $i-th$ feature has to be detached. On the other hand, some sub-feature maps approach zero weights should also be detached in each filter. after selection of feature map, the feature maps X are converted into $X\widehat{w}$.

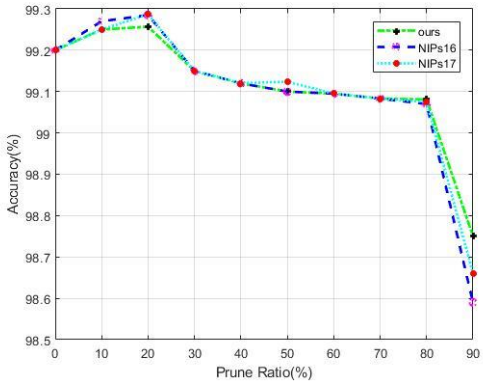
We describe two terms; the number of feature maps with ratio as follows:

$$FNum = d - \sum_{i=1}^{d} ceil\left(\frac{\sum_{j=1}^{s} \widehat{w_{ij}}}{s}\right)$$
$$ratio = \frac{\sum_{i=1}^{d} \sum_{j=1}^{s} \widehat{w_{ij}}}{d * s} \tag{3}$$

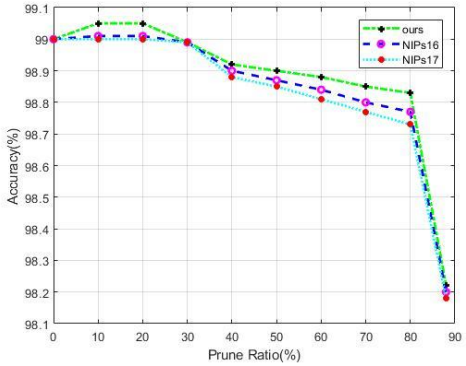
whereas $ceil(x)$ pushes the members of x to the near integers to infinity. Next, *FNum* illustrates the related number of feature maps in the filter. At that point, *FNum* shows the degree of sparsity between filters. The higher *FNum* is, the higher the related number is, and the lower the sparsity degree between filters is. The ratio represents the

degree of sparsity in the filters. The minimum ratio is, higher than the sparsity degree in the filters. For feature map selection, we want to obtain better accuracy with less ratio and *FNum*.

In equation 1, the optimal problem of group LASSO, we can observe that the optimization problem is the sum of convex functions [38]. The first term is considered as squared loss which is smooth. Further, the last two terms show the non-smooth regularized. This algorithm is based on the sub-gradient method. It calculates the gradient update iteratively. The first-order black-box approach is applied at each iteration. Therefore, only the gradient and function value is required to estimate and the rate of convergence is optimal for smooth convex optimization. Moreover, the projection of Euclidean can be evaluated either linear time or analytically. Thus, this method can be used on big CNN architecture such as LeNet, AlexNet, VGGNet, and ResNet. These features of the algorithm make it reliable to choose the feature maps of the CNN models.



(a) Accuracy loss comparison with same sparsity in the first convolutional layers



(b) Accuracy loss comparison with same sparsity in the second convolutional layer

Fig. 2. Comparision of loss of accuracy with similar sparsity for each convolutional layer of LeNet-5 Model.

TABLE I: ILLUSTRATING OUTCOMES FOR THE LENET-5 ARCHITECTURE ON THE MNIST DATASET

Approach Used	Number of Filters in each layer	Error%	FLOPs saved	Pruned Ratio
Baseline	Layer1: 20, Layer2: 50	0.83	4.40×10^6	-
NIPS'16 [24]	Layer1: 5, Layer2: 19	0.80	5.97×10^5	86.42%
NIPS'17[39]	-	0.86	2.89×10^5	90.47%
Singh[40]	Layer1: 3, Layer2: 8	0.92	2.14×10^5	95.14%
Ours	**Layer1: 2, Layer2: 6**	**0.95**	**2.13×10^5**	**96.15%**

IV. EXPERIMENT SETTINGS

We performed our technique on several standard architectures and datasets. We have applied CIFAR family datasets (CIFAR10) on VGGNet (VGG-16) and ImageNet on AlexNet models, on the other hand, we have used MNIST datasets (handwritten digits) on the LeNet-5 model. Datasets are pre-split, for the CIFAR family, datasets are distributed for 50,000 for train images and 10,000 for test images with 10 classes, and the MNIST dataset is split into 60,000 examples for the train set and 10,000 examples for the test set with 10 different classes of handwritten numbers. The NVIDIA GTX TITAN Xp GPU is used for experiments with a Python framework known as Pytroch. The initial models are trained from scratch for sake of calculating baseline accuracies in the test set. The data augmentation is used all through the training time, which cropped every image arbitrarily into a shape of 32 by 32 with padding set to four and applies a horizontal flip. Further, we fix the mini-batch size to 100 for the training set and 1000 for the test set for the VGGNet model. The LeNet model comparatively is small and takes input 28 by 28 image size, therefore, we cropped every image randomly into the shape of 28 by 28, with padding of two and a flip through data augmentation method. The Stochastic Gradient Descent (SGD) is used for both models during the trained and fine-tuned process for about 150 iterations. During this process, the initial learning rate is set to 0.001 to 0.1 for all number of iterations. Furthermore, we have also used weight initialization and different appropriate scores for λ_1 and λ_2 in equation 2 are applied to penalize the required sparsity regularization and grouping influence respectively in the network. Lastly, we set all extra parameters the same as used in baseline training.

A. LeNet-5 on MNIST dataset

This section describes the usefulness of our approach on traditional CNN architecture, known as LeNet-5 using MNIST datasets. The LeNet-5, here number five shows the number of layers in the network, the input layer with two convolutional layers and two fully connected (FC) layers with a total of 431K parameters. This network has a baseline accuracy of 99.13% on the MNIST datasets. We apply our sparse group lasso method that measures the filter and filter weight importance layer-wise and apply fine-tuning process iteratively. We adjust the value of $\lambda_1 = 0.0001$ in equation D(2) to compute the filter importance in each iteration of pruning. The initial learning rate is set to 0.001 to 0.1 for all number of iterations for this experiment. Comparing with the past methods given in Table I, we have a considerably greater pruned ratio with Flop's compression is the higher and negligible loss in the accuracy. This shows the greatness of our introduced approach for measuring filter importance against the past approaches. It could be seen in the Fig. 2(a) that the accuracy of the three approaches would decrease with the rise in pruning ratio in the first convolutional layer. Additionally, there is no deceptive dissimilarity among these algorithms. It can be seen that the proposed method`s

accuracy has minor enhancement when the pruning ratio is dropped illustrated in Fig. 2(b). This might be due to our method neglect over-fitting to some degree under that kind of situation.

TABLE II: PRUNING RESULTS OF THE ALEXNET MODEL PERFORMED ON IMAGENET.

Method	Accuracy	Param-baseline	Parameters	Pruned	FLOPs Saved
Weight sum[17]	54.99%	6.0 x 10^6	5.4 x 10^6	73.3%	43.8%
Slimming[31]	53.87%	-	-	70.5%	46.9%
Group Lasso[24]	54.31%	6.0 ×10^6	3.5 ×10^5	67.4%	51.4%
ThiNet[41]	53.67%	-	-	75.6%	55.9%
Jiang[42]	54.63%	-		76.1%	63.7%
Ours	**54.91%**	6.0 ×10^6	3.1 x 10^5	**75.9%**	**67.8%**

B. AlexNet on ImageNet Dataset

The ImageNet dataset splits into 1.2 million training sets and 50 thousand test sets, respectively, with 1000 different classes. Moreover, AlexNet contains around 61 million parameters that are distributed into five convolutional layers with 3 fully connected (FC) layers and a softmax layer. We set the $\lambda_1 = 5 \times 10^{-6}$ and $\lambda_2 = 10^{-4}$ and trained the model for 80 iterations. Table II shows the result of compressing AlexNet with our proposed method. It is clear that according to obtained accuracy, our method pruned 75.9% of parameters with no loss of accuracy, comparing to the group LASSO which achieved around 66.7% parameters pruned and Jiang et al [42] has pruned 63.7% parameters and suffer a loss in accuracy of 0.17% and 0.28% respectively. It is confirmed that if more compression rates applied using EGL or group LASSO, it may lead to greater loss in performance. Fig. 3 illustrates the accuracy of different classification problems over pruned FLOPs, achieved by distinguishing the structured regularization. Our Sparse Group Lasso is an effective structure regularization that generally does well than other state-of-art methods.

C. VGG16 on CIFAR10 dataset

The VGG16 model, here 16 shows the number of layers, in which it has 1 input layer, 13 convolutional layers with 2 fully connected (FC) layers, there are about 138 million parameters across these layers. We use our method on VGG16 for 180 iterations applying the $\lambda_1 = 1 \times 10^{-5}$ and $\lambda_2 = 10^{-6}$. Detailed outcomes are presented in table. III for VGG16 pruning through our model. Our approach has performed well on parameters pruning contrary to work presented in table. II by pruning 96.1% parameters, whereas in [17] it only prunes 64.0% of parameters. Moreover, we have achieved FLOPs reduction of about 83.55% against the work in [17] achieved 34.2% of FLOPs reduction. Moreover, Fig. 4 illustrates the results of filter pruning with SGL pruning rate starts from 10 to 100%. The pruning outcomes show that data distribution shares among different convolutional layers. With some convolutional layers, pruning only 10% of information can achieve over 70% reduction in filters.

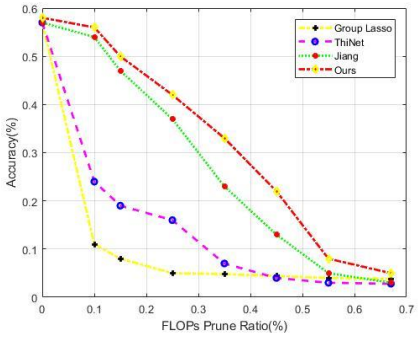


Fig. 3. Comparison between Sparse Group Lasso (ours) and other state-of-art methods.

TABLE III: PRUNING RESULTS OF THE VGG-16 MODEL PERFORMED ON CIFAR-10

Method	Baseline	Accuracy	Param-baseline	Parameters	Pruned
Li [41]	93.25%	93.30%	1.5 x 10^7	5.4 x 10^6	64.0%
Slimming[31]	93.66%	93.80%	-	-	88.5%
Entropy[43]	93.72%	93.97%	1.5 ×10^7	3.5 ×10^5	76.4%
Aketi[19]	93.75%	93.80%	-	-	90.5%
Kumar[13]	93.77%	93.81%	1.5 x 10^7	3.0 x 10^5	92.7%
Ours	**93.76%**	**93.80%**	-	**3.1 x 10^5**	**96.1%**

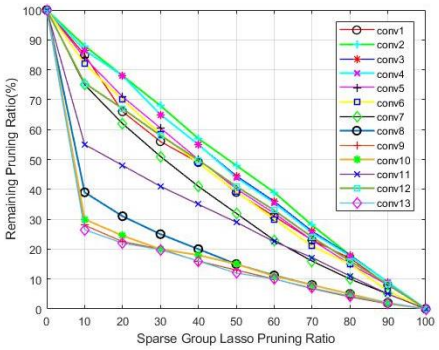


Fig. 4. The VGG16 pruning results on the CIFAR-10 dataset with a sparse group lasso pruning rate from 10 to 100%.

V. CONCLUSION

In this article, we presented a fresh new methodology to measure the impact of the filters, which computes the valuation of filters and weight of the filters based on the effectiveness driven by these filters. We proposed a sparse group lasso approach for filter level pruning, by assuming a single feature map as a single group and weights in a feature map as group members. In this way, the optimization method can be used to delete the complete feature map as well as select weights in the feature map according to importance valuation. Hence, generated to formulate the method of

pruning. Moreover, the effectiveness of the filters in each layer of CNNs is also mentioned and the results illustrated that in various layers. The large range of filters have a slight impact on the performance of the model therefore, it is necessary to remove those filters from the layer. Considerable study shows the benefit of our proposed approach by comparing the present methods. Particularly, VGG-16 architecture used on CIFAR-10 datasets, our introduced methodology can effectively prune 96.1% parameters with FLOPs saved approx. 83.55%, and a slight accuracy gain, these results show that our method has the upper hand against some benchmark approaches. In future, we will try to implement this method to more complex architectures such as ResNet and GoogLeNet with huge datasets like ImageNet.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Aakash Kumar devised the project, the main conceptual ideas and proof outline. Baoqun YIN worked out almost all of the technical details, and performed the numerical calculations for the suggested experiment. Both Ajeet Kumar Bhatia and Aneel Kumar Bhatia contributed to the final version of the manuscript. And Avinash Rohra wrote the article.

REFERENCES

[1] M. Park, "JGAN: A joint formulation of GAN for synthesizing images and labels," *IEEE Access*, vol. 8, pp. 188883–188888, 2020, doi: 10.1109/ACCESS.2020.3031292.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[3] M. J. Norval, Z. Wang, and Y. Sun, "Evaluation of image processing technologies for pulmonary tuberculosis detection based on deep learning convolutional neural networks," *JAIT*, vol. 12, no. 3, 2021, doi: 10.12720/jait.12.3.253-259.

[4] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with ATROUS separable convolution for semantic image segmentation," in *Proc. Computer Vision – ECCV 2018*, vol. 11211.

[5] S. Bunrit, N. Kerdprasop, and K. Kerdprasop, "Improving the representation of CNN based features by Autoencoder for a task of construction material image classification," *Journal of Advances in Information Technology*, vol. 11, pp. 192–199, Jan. 2020, doi: 10.12720/jait.11.4.192-199.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *arXiv:1409.3215 [cs]*, Dec. 2014.

[7] V. Mnih *et al.*, "Playing ATARI with deep reinforcement learning," *arXiv:1312.5602 [cs]*, Dec. 2013.

[8] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv:1503.02531 [cs, stat]*, Mar. 2015.

[9] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *arXiv:1505.06798 [cs]*, Nov. 2015.

[10] M. Ali, B. Yin, A. Kumar, A. M. Sheikh, and H. Bilal, "Reduction of Multiplications in Convolutional Neural Networks," in *Proc. 2020 39th Chinese Control Conference (CCC)*, Jul. 2020, pp. 7406–7411, doi: 10.23919/CCC50068.2020.9188843.

[11] M. Nagel, M. V. Baalen, T. Blankevoort, and M. Welling, "Data-Free Quantization Through Weight Equalization and Bias Correction," in *Proc. 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), Oct. 2019, pp. 1325–1334, doi: 10.1109/ICCV.2019.00141.

[12] Y. Li *et al.*, "Weight-dependent Gates for Differentiable Neural Network Pruning," *arXiv:2007.02066 [cs]*, Aug. 2020.

[13] A. Kumar, A. M. Shaikh, Y. Li, H. Bilal, and B. Yin, "Pruning filters with L1-norm and capped L1-norm for CNN compression," *Appl Intell*, vol. 51, no. 2, pp. 1152–1160, Feb. 2021, doi: 10.1007/s10489-020-01894-y.

[14] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal Brain Surgeon," *Advances in Neural Information Processing Systems*, vol. 5, 1992.

[15] [15] Y. LeCun, J. S. Denker, and S. A. Solla, *Optimal Brain Damage*, p. 8.

[16] Z. Mariet and S. Sra, "Diversity networks: neural network compression using determinantal point processes," *arXiv:1511.05077 [cs]*, Apr. 2017.

[17] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," *arXiv:1608.08710 [cs]*, Mar. 2017.

[18] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv:1510.00149 [cs]*, Feb. 2016.

[19] S. A. Aketi, S. Roy, A. Raghunathan, and K. Roy, "Gradual channel pruning while training using feature relevance scores for convolutional neural networks," *IEEE Access*, vol. 8, pp. 171924–171932, 2020.

[20] L. Jacob, G. Obozinski, and J.-P. Vert, "Group lasso with overlap and graph lasso," in *Proc. the 26th Annual International Conference on Machine Learning*, New York, NY, USA, Jun. 2009, pp. 433–440.

[21] J. A. Villaruz, "Deep convolutional neural network feature extraction for berry trees classification," *JAIT*, vol. 12, no. 3, 2021, doi: 10.12720/jait.12.3.226-233.

[22] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81–89, Jun. 2017, doi: 10.1016/j.neucom.2017.02.029.

[23] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," *arXiv:1611.06321 [cs]*, Oct. 2018.

[24] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning Structured Sparsity in Deep Neural Networks," *arXiv:1608.03665 [cs, stat]*, Oct. 2016, Accessed: Jul. 15, 2021.

[25] J. Friedman, T. Hastie, and R. Tibshirani, "A note on the group lasso and a sparse group lasso," *arXiv:1001.0736 [math, stat]*, Jan. 2010.

[26] A. K. Fletcher, S. Rangan, and V. K. Goyal, "Necessary and sufficient conditions on sparsity pattern recovery," *IEEE Trans. Inform. Theory*, vol. 55, no. 12, Art. no. 12, Dec. 2009, doi: 10.1109/TIT.2009.2032726.

[27] J. Peng *et al.*, "Regularized multivariate regression for identifying master predictors with application to integrative genomics study of breast cancer," *Ann Appl Stat*, vol. 4, no. 1, Art. no. 1, Mar. 2010, doi: 10.1214/09-AOAS271SUPP.

[28] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both Weights And Connections For Efficient Neural Networks," *arXiv:1506.02626 [cs]*, Oct. 2015.

[29] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through L_0 regularization," *arXiv:1712.01312 [cs, stat]*, Jun. 2018.

[30] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," *arXiv:1808.06866 [cs]*, Aug. 2018, Accessed: May 26, 2021.

[31] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," *arXiv:1708.06519 [cs]*, Aug. 2017.

[32] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," *arXiv:1906.10771 [cs, stat]*, Jun. 2019.

[33] B. Y. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Penksy, "Sparse convolutional neural networks," in *Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 806–814.

[34] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," *arXiv:1707.06168 [cs]*, Aug. 2017.

[35] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J Royal Statistical Soc B*, vol. 68, no. 1, pp. 49–67, Feb. 2006.

[36] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *Journal of Computational and Graphical Statistics*, vol. 22, no. 2, pp. 231–245, Apr. 2013, doi: 10.1080/10618600.2012.681250.

[37] W. Zhu *et al.*, "Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks," *AAAI*, vol. 30, no. 1, Mar. 2016.

[38] *Convex Analysis*. (1997). [Online]. Available: https://press.princeton.edu/books/paperback/9780691015866/convex-analysis

[39] K. Neklyudov, D. Molchanov, A. Ashukha, and D. P. Vetrov, "Structured Bayesian pruning via log-normal multiplicative noise," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[40] P. Singh, V. S. R. Kadi, N. Verma, and V. P. Namboodiri, "Stability based filter pruning for accelerating Deep CNNs," *arXiv:1811.08321 [cs]*, Nov. 2018.
[41] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," *arXiv:1707.06342 [cs]*, Jul. 2017.
[42] C. Jiang, G. Li, C. Qian, and K. Tang, "Efficient DNN neuron pruning by minimizing layer-wise nonlinear reconstruction error," in *Proc. the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, Jul. 2018, pp. 2298–2304.
[43] Y. Li, L. Wang, S. Peng, A. Kumar, and B. Yin, "Using feature entropy to guide filter pruning for efficient convolutional networks," in *Proc. Artificial Neural Networks and Machine Learning – ICANN 2019: Deep Learning*, Cham, 2019, pp. 263–274.

Aakash Kumar received the B.S. degree in electronic engineering from the University of Sindh, Jamshoro, Pakistan, in 2011, and the M.S degree in control science and engineering from the University of Science and Technology of China (USTC), China, in 2017, where he is currently pursuing the Ph.D. degree with the Department of Automation. His main interests include deep learning, deep compression models, and reduction of multiplications for CNNs.