AN OPERATOR NORM BASED PASSIVE FILTER PRUNING METHOD FOR EFFICIENT CNNS

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional neural networks (CNNs) have shown state-of-the-art performance in various applications. However, CNNs are resource-hungry due to their requirement of high computational complexity and memory storage. Recent efforts toward achieving computational efficiency in CNNs involve filter pruning methods that eliminate some of the filters in CNNs based on the "importance" of the filters. Existing passive filter pruning methods typically use the entry-wise norm of the filters to quantify filter importance, without considering how well the filter contributes in producing the node output. Under situations where the large number of filters are to be pruned from the network, the entry-wise norm methods always select high entry-wise norm filters as important, and ignore the diversity learned by the other filters that may result in degradation in the performance. To address this, we present a passive filter pruning method where the filters are pruned based on their contribution in producing output by implicitly considering the operator norm of the filters. The computational cost and memory requirement is reduced significantly by eliminating filters and their corresponding feature maps from the network. Accuracy similar to the original network is recovered by fine-tuning the pruned network. The proposed pruning method gives similar or better performance and recovers accuracy faster during the fine-tuning process than the entry-wise norm-based pruning methods. The efficacy of the proposed pruning method is evaluated on audio scene classification (e.g. TAU Urban Acoustic Scenes 2020) and image classification (MNIST handwritten digit classification).

1 INTRODUCTION

Convolutional neural networks (CNNs) have shown great success and exhibit state-of-the-art performance when compared to traditional hand-crafted methods in many domains (Gu et al. (2018)). Even though CNNs are highly efficient in solving non-linear complex tasks (Denton et al. (2014)), it may be challenging to deploy large-size CNNs on resource-constrained devices such as mobile phones or internet of things (IoT) devices, owing to high computational costs during inference and the memory requirement for CNNs (Simonyan & Zisserman (2015); Krizhevsky et al. (2012)). Thus, the issue of reducing the size and the computational cost of CNNs has drawn a significant amount of attention in the research community.

Recent efforts toward reducing the computational complexity of CNNs involve pruning methods where a set of parameters, such as weights or filters, are eliminated from the CNNs. These pruning methods are motivated by the existence of redundant parameters (Denil et al. (2013); Livni et al. (2014)) in CNNs that only yield extra computations without contributing much in performance (Frankle & Carbin (2019)). For example, Li et al. (2017) found that 64% of the parameters, contributing approximately 34% of the computation time, are redundant. Eliminating such redundant parameters from CNNs provides small CNNs that perform similar to the original CNNs while reducing the computations and the memory requirement compared to the original CNNs.

While eliminating weights from an unpruned CNN may result in a highly sparse network with few parameters, a pruned network obtained by eliminating individual weights is unstructured and may not be straightforward to run more efficiently. The practical acceleration in the unstructured sparse pruned networks is limited due to random connections despite high sparsity (Luo et al. (2017)). Moreover, the unstructured sparse networks can not be supported by off-the-shelf libraries and re-

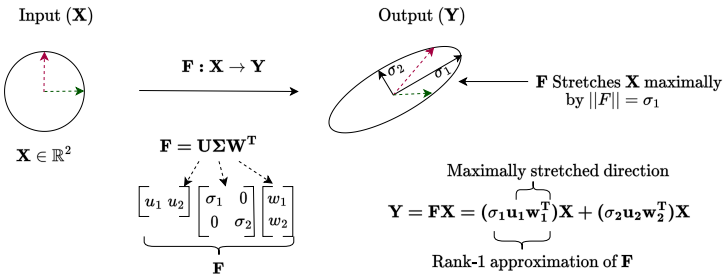


Figure 1: A geometrical view of output produced by a convolution operation, where input feature maps $\mathbf{X}$ in $\mathbb{R}^2$ are transformed to output feature maps $\mathbf{Y}$ in $\mathbb{R}^2$ using a transformation matrix $\mathbf{F}$. $\mathbf{F}$ is decomposed to a left singular matrix ($\mathbf{U}$), a right singular matrix ($\mathbf{W}$) and a diagonal matrix (Σ) using a singular value decomposition method. $\mathbf{U}$ and $\mathbf{W}$ are orthogonal matrices that cause rotation in the input, and σ_1 and σ_2 are singular values that scale the input. $\mathbf{F}$ stretches $\mathbf{X}$ maximally by $||\mathbf{F}|| = \sigma_1$ which is an operator norm of $\mathbf{F}$.

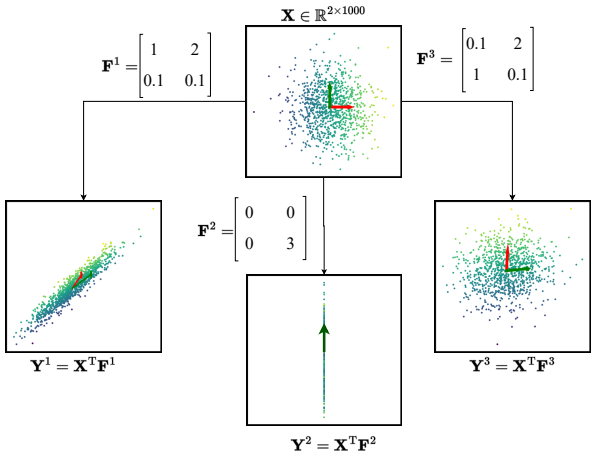
quire specialised software or hardware for speed-up (Wen et al. (2016); Han et al. (2016)). To address this unstructured pruning problem, several filter pruning methods have been proposed which eliminates whole filters, resulting in a structured pruned network (Luo et al. (2018)) that does not require additional resources for speed-up.

In these structured filter pruning methods, the "importance" of a filter, used to decide if a filter is retained or eliminated, is measured using either active or passive methods. Active filter pruning methods involve a dataset to compute the importance of filters. For example, (Luo & Wu (2017); Polyak & Wolf (2015); Hu et al. (2016)) use feature maps which are the outputs generated from CNN filters corresponding to a set of examples, and apply metrics such as entropy, variance and the average percentage of zeros on the feature maps to quantify the importance of the filters. On the other hand, passive filter pruning methods only use the parameters of the filters without involving any dataset to compute the importance of the filters.

A typical passive filter pruning method uses an entry-wise norm of the filters to measure their importance. For example, this might be a l_1-norm a sum of absolute values each entry of the filter or an l_2-norm square root of the squared sum of each entry of the filter. Li et al. (2017) eliminates filters having smaller entry-wise l_1-norm or l_2-norm as measured from the origin, and finds that eliminating filters based on the entry-wise l_2-norm of the filters gives similar performance to that of the entry-wise l_1-norm. He et al. (2019) eliminates the filters with smaller l_2-norm as measured from the geometric median of all filters. Both the previous methods assume that a filter with smaller entry-wise norm is less informative or less important, without considering how significantly a filter contribute in producing output. For example, an illustration of the contribution by the filters in producing output is shown in Figure 1, where filters $\mathbf{F}$ produces an output $\mathbf{Y}$ by maximally stretching the input $\mathbf{X}$ by a largest singular value σ_1 that represents an operator norm of the $\mathbf{F}$. However, the entry-wise norm methods do not consider any input-output relationship information and rely on each entry of the filter while computing the filter importance.

To illustrate the above further, we pictorially show in Figure 2(a) that two filters $\mathbf{F}^1$ and $\mathbf{F}^3$ having same entry-wise norm contribute differently and produce different output due to different operator norm of the each filter shown in Figure 2(b). Hence such filters should be given different importance. Moreover, when a few number of filters have to be retained in the CNN to yield a very small pruned CNN, selecting filters with only high entry-wise norm may ignore the smaller norm filters that may also contribute significantly in producing output (Ye et al. (2018)). This may degrade the accuracy of the pruned network significantly due to relying on the filters having high entry-wise norm, however low significance in producing output. For example as shown in Figure 2(b), $\mathbf{F}^2$ has σ_1 = 3, and it stretches $\mathbf{X}$ relatively larger than that of the $\mathbf{F}^1$ and the $\mathbf{F}^3$. However, the entry-wise norm of the $\mathbf{F}^2$ is the smallest among all three filters. Therefore $\mathbf{F}^2$ shall be ignored by the entry-wise norm methods despite a relatively high contribution than other filters.

To select the important filters based on their contribution in producing output, we propose a novel passive filter pruning framework by considering filters in a convolution layer as an operator that



(a) Outputs produced by applying three different filters

	$\mathbf{F}^1$	$\mathbf{F}^2$	$\mathbf{F}^3$
Entry-wise l_1-norm (normalized)	1	0.94	1
Operator norm (σ_1)	2.24	3	2

(b)

Figure 2: (a) An illustration of output produced in the convolution layer by three CNN filters, $\mathbf{F}^1$, $\mathbf{F}^2$ and $\mathbf{F}^3$ after applying a convolution operation on $\mathbf{X} \in \mathbb{R}^{2\times1000}$. (b) shows the entry-wise l_1-norm and the operator norm of the three filters.

transform input feature maps to output feature maps. To compute the importance of filters, we use the operator norm of the filters, which represents the maximum contribution of the filters in producing output rather than relying on the entry-wise norm of the filters. Utilising all filters in a convolutional layer, we use a Rank-1 approximation of the filters to obtain the maximally stretched direction associated with the operator norm in which the input gets stretched maximally by all filters. A filter in the convolutional layer is deemed important based on how well it is aligned along the maximally stretched direction represented by all filters in that convolutional layer.

The proposed passive filter pruning method is evaluated on pre-trained CNNs such as VGGish (Hershey et al. (2017)) and DCASE 2021 Task 1A baseline (Martín-Morató et al. (2021)) designed for audio scene classification (ASC), and the VGG-16 network (Liu & Deng (2015)) designed for image classification problem.

2 Methodology

Notation: Consider a CNN having L convolutional layers with indexes $\in \{1, 2, \cdots, l, \cdots, L\}$. A feature map in a convolutional layer denotes an output produced by a CNN filter. In the lth convolution layer, let n_l denote the number of input channels with each feature map of size $h_l \times w_l$. Let $\mathbf{X}$ of size $(n_{l-1} \times h_{l-1} \times w_{l-1})$ and $\mathbf{Y}$ of size $(n_l \times h_l \times w_l)$, denote the feature map matrices, produced by stacking all the respective feature maps in the $(l-1)^{\text{th}}$ and the lth layer respectively. A jth feature map in the lth layer is produced by a convolution operation using a jth filter $\mathbf{F}^{l,j}$ of size $(n_{l-1} \times k_l \times k_l)$ having n_{l-1} 2D kernels each of $(k_l \times k_l)$ on $\mathbf{X}$. All filters in the lth convolution layer constitute a kernel tensor $\mathbf{K}_l = [\mathbf{F}^{l,1}\ \mathbf{F}^{l,2}\ \ldots\ \mathbf{F}^{l,n_l}]$ of size $(n_{l-1} \times n_l \times k_l \times k_l)$. A pictorial illustration of two intermediate layers in the CNN is described in Figure 3.

2.1 Computing filter importance in the lth convolution layer

Given a kernel tensor $\mathbf{K}_l$ for the lth convolution layer, our aim is to compute the importance of each of the n_l filters. Without loss of generality, we transform the kernel tensor $\mathbf{K}_l$ to $\mathbf{K^v}_l$ of size $(n_{l-1} \times n_l \times {k_l}^v)$ by vectrozing the 2D kernels each of $(k_l \times k_l)$ to ${k_l}^v$.

To identify importance of each filter, we use $\mathbf{K^v}_l$ to obtain a maximally stretched direction corresponding to the operator norm of the filters along which the filters in the given convolutional layer stretches the input maximally. Since the output produced by the cth channel of a filter depends only on the cth channel of the input in CNNs, the maximally stretched direction is obtained for each channel of the filters independently.

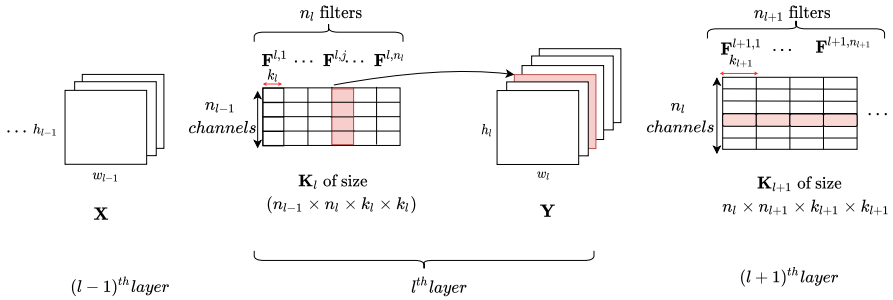


Figure 3: An illustration of the intermediate layer structure and filter pruning in CNN. In the lth layer, there are n_l filters each having $(k_l \times k_l)$ size and n_{l-1} channels. The n_l filters produce n_l feature maps. Pruning the jth filter, $\mathbf{F}^{l,j}$, in the lth convolution layer results in elimination of the feature map produced by the pruned filter and corresponding channel in the $(l+1)^{\text{th}}$ layer.

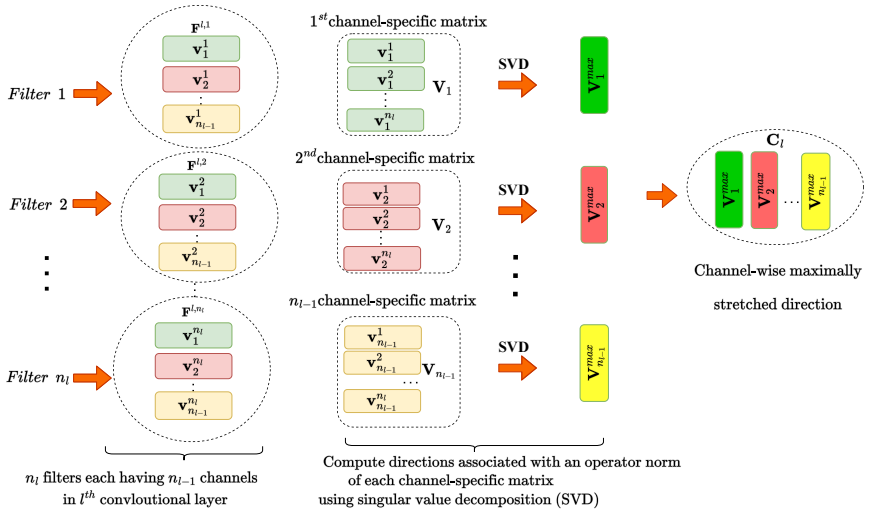


Figure 4: An illustration to obtain maximally stretched direction for each n_{l-1} channels across all filters in the lth convolutional layer.

A channel-specific matrix $\mathbf{V}_c$ of size $(n_l \times {k_l}^v)$ is constructed by taking c^{th} channel of all filters. $\mathbf{V}_c$ denotes the learned weights of the cth channel across the n_l filters. Next, singular value decomposition (SVD) is performed on $\mathbf{V}_c$ to compute a rank-1 approximation of the cth channel as $\mathbf{V}_c \approx \sigma_1 \mathbf{u}_1 \mathbf{w}_1^{\text{T}}$. Here, σ_1 denotes the maximum singular value that affects the corresponding channel of the input maximally and is equivalent to the operator norm of the $\mathbf{V}_c$, $\mathbf{u}_1$ is the first left singular vector, and $\mathbf{w}_1$ is the first right singular vector. A row of $\mathbf{V}_c$ normalized to unit norm is denoted as $\mathbf{V}_c^{\text{max}}$, and is considered as a maximally stretched direction for the cth channel. $\mathbf{V}_c^{\text{max}}$ provides a reference for measuring the significance the cth channel of the filter.

After obtaining $\mathbf{V}_c^{\text{max}}$ corresponding to the cth channel, we obtain maximally stretched directions for other channels. Finally, the $\mathbf{V}_c^{\text{max}}$ of all channels are stacked together to yield $\mathbf{C}_l$ of size $({k_l}^v \times n_{l-1})$ for a given convolutional layer. A pictorial representation to obtain $\mathbf{C}_l$ is shown in Figure 4.

Given $\mathbf{C}_l$, the importance for the jth filter is computed as, $\text{Trace}[(\mathbf{F}^{l,j})\mathbf{C}_l)]$. The filters are ranked as per their importance with a relatively high importance score indicates a high contribution of the filter in producing output. Algorithm 1 summarizes the process to compute the importance of various filters in a given convolutional layer.

After ranking the filters based on their importance for various convolutional layers, few unimportant filters are eliminated based on a user-defined pruning ratio for various convolutional layers of CNN, and a pruned network is obtained. Pruning a filter in the lth convolution layer also eliminates the

Algorithm 1 Filter Importance Calculation

input : $\mathbf{K^v}_l$ of size $(n_{l-1} \times n_l \times {k_l}^v)$, kernel matrix in l^{th} layer .
output: Score_norm, # normalized importance scores of filters, in l^{th} layer.
Initialization: $\mathbf{C}_l$ = [ ], Score = [ ], # importance score of filters.
... # Obtaining channel-wise maximally stretched direction associated with operator norm...
for $c \leftarrow 1$ **to** n_{l-1} **do**
  $\mathbf{V}_c$ =$\mathbf{K^v}_l$[c , : , :] # Take c^{th} channel of size $(n_l \times {k_l}^v)$ from all the filters,
  $\mathbf{u}, \Sigma, \mathbf{w}$ = SVD($\mathbf{V}_c$) # Perform SVD on $\mathbf{V}_c$,
  $\mathbf{C}_l$.append(($\mathbf{u}_1 \mathbf{w}_1^{\text{T}}$)[1,:]) #Take first left ($\mathbf{u}_1$) and first right ($\mathbf{w}_1$) singular vector associated with σ_1, take any row of $\mathbf{u}_1 \mathbf{w}_1^T$ and normalized to 1 and append the normalized row to $\mathbf{C}_l$.
end
.....# Importance-score calculation.....
for $j \leftarrow 1$ **to** n_l **do**
  $\mathbf{F}^{l,j}$ = $\mathbf{K^v}_l$[: , j , :] # Take j^{th} filter.
  $\bar{\mathbf{F}}^{l,j}$= Score.append([trace(($\mathbf{F}^{l,j}$)$\mathbf{C}_l$)]) #Compute importance.
end
α =[Score]
Score_norm = $\frac{\alpha^2}{\max(\alpha^2)}$ # Normalized importance.
return Score_norm

feature map produced by the pruned filter and the related kernel or channel in the next layer as shown in Figure 3, hence the computations is reduced in both the lth layer and the $(l+1)^{\text{th}}$ layer. In the end, a fine-tuning of the pruned network is performed to regain the performance loss due to elimination of filters from the original network.

3 EXPERIMENTS

We evaluate the proposed pruning method on CNNs designed for audio scene classification (ASC) and image classification. A brief summary of the unpruned CNNs used for experimentation is shown in Figure 5 and is described below,

Unpruned CNNs: We use two different unpruned networks for ASC, (a) VGGish_Net and (b) DCASE21_Net. We also use (c) VGG-16 network for image classification.

(a) **VGGish_Net:** The VGGish_Net is built using a publicly available pre-trained VGGish network (Hershey et al. (2017)) followed by a dense and a classification layer. We train the VGGish_Net on TUT Urban Acoustic Scenes 2018 development (we denote it as DCASE-18) training dataset (Mesaros et al. (2018)) to classify 10 different audio scenes using Adam optimizer with cross-entropy loss function for 200 epochs. The input to the VGGish_Net is a log-melspectrogram of size (96 $\times$ 64) computed corresponding to 960ms audio segment for a whole 10s audio scene. The VGGish_Net has 55,361,162 (approximately 55M) parameters, requiring 903M multiply-accumulate operations (MACs) during inference corresponding to an audio clip of 960ms and gives 64.69% accuracy on 10s audio scene for DCASE-18 development validation dataset. The VGGish_Net has six convolution layers (termed as C1 to C6).

(b) **DCASE21_Net:** DCASE21_Net is a publicly available pre-trained network designed for DCASE 2021 Task 1A that is trained using TAU Urban Acoustic Scenes 2020 Mobile development dataset (we denote "DCASE-20") to classify 10 different audio scenes (Martín-Morató et al. (2021)). The input to the network is a log-melspectrogram of size (40 $\times$ 500) corresponding to a 10s audio clip. The trained network has 46246 parameters, requiring approximately 287M MACs during inference corresponding to 10-second-length audio clip and gives 48.58% accuracy on the DCASE-20 development validation dataset. DCASE21_Net consists of three convolutional layers (termed as C1 to C3) and one fully connected layer.

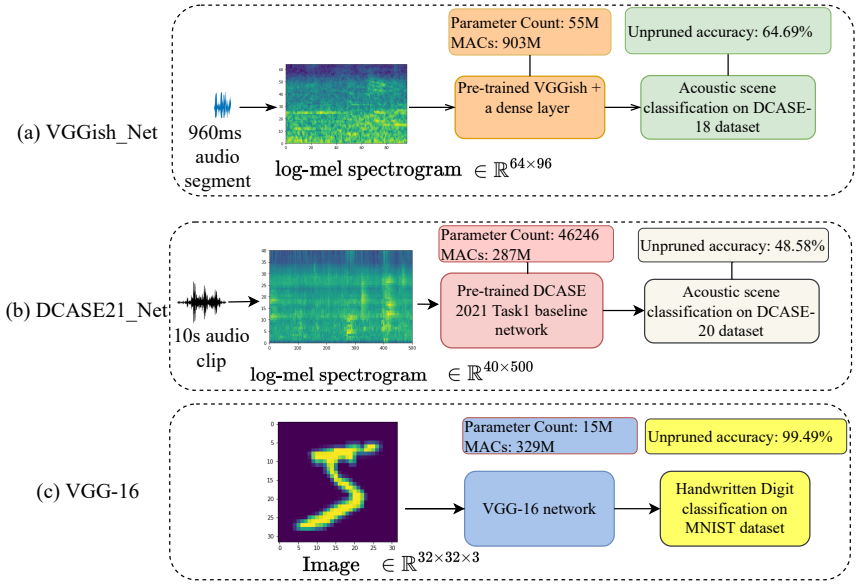


Figure 5: Unpruned CNNs used for experimentation; (a) VGGish_Net (b) DCASE21_Net and (c) VGG_Net.

(c) **VGG-16:** We use a VGG-16 network ((Liu & Deng (2015))) for handwritten digit classification using MNIST dataset[1](LeCun et al. (2010)). We train the VGG-16 from scratch for 200 epochs on the MNIST training dataset. The VGG-16 consists of 13 convolutional layer (termed as C1 to C13) and 2 dense layers. The VGG-16 has 15M parameters and requires 329M MACs per inference and the trained VGG-16 gives 99.49% accuracy for the MNIST testing dataset.

Fine-tuning of the pruned network: After computing importance of the filters using Algorithm 1 across various convolutional layers, we eliminate p percentage of unimportant filters from various convolutional layer, where $p \in \{25, 50, 75, 90\}$ denotes a pruning ratio. Once the pruned network is obtained, we re-train the network with similar conditions such as same optimizer as used in the training of the unpruned networks (a), (b) and (c) for 100 epochs. The codes for the proposed pruning framework, various pruned and unpruned networks can be found at the link [2].

Other methods for comparison: We compare the proposed operator norm based pruning method with that of the entry-wise norm based methods, (a) l_1-norm method that eliminates filters with smaller entry-wise l_1-norm (Li et al. (2017)) and (b) geometric median (GM) method that eliminates filters with smaller l_2-norm as measured from the geometric median of all filters (He et al. (2019)).

4 Results

We analyse accuracy, the number of MACs and the number of parameters in the pruned networks obtained after pruning various subsets of convolution layers at different pruning ratios from the unpruned network.

VGGish_Net and DCASE21_Net on ASC: As shown in Figure 6 (a,b), the accuracy obtained using the various pruned networks is similar to that of the unrpuned networks at $p = 25\%$. For VGGish_Net, the number of MACs are reduced by 40 percentage points and the parameters are reduced by 25 percentage points, when all convolutional layers (C1_6: C1 to C6) are pruned at $p = 25\%$

[1]The MNIST dataset is downloaded using Keras API which has a pre-defined training and testing set of grayscale images. Each grayscale image of size (28×28) is converted into three channels by depthwise stacking the grayscale image and then reshaping to ($32 \times 32 \times 3$) that is used as an input to VGG-16.

[2]`https://anonymous.4open.science/r/Operator_norm_passive_Filter_Pruning-125D`

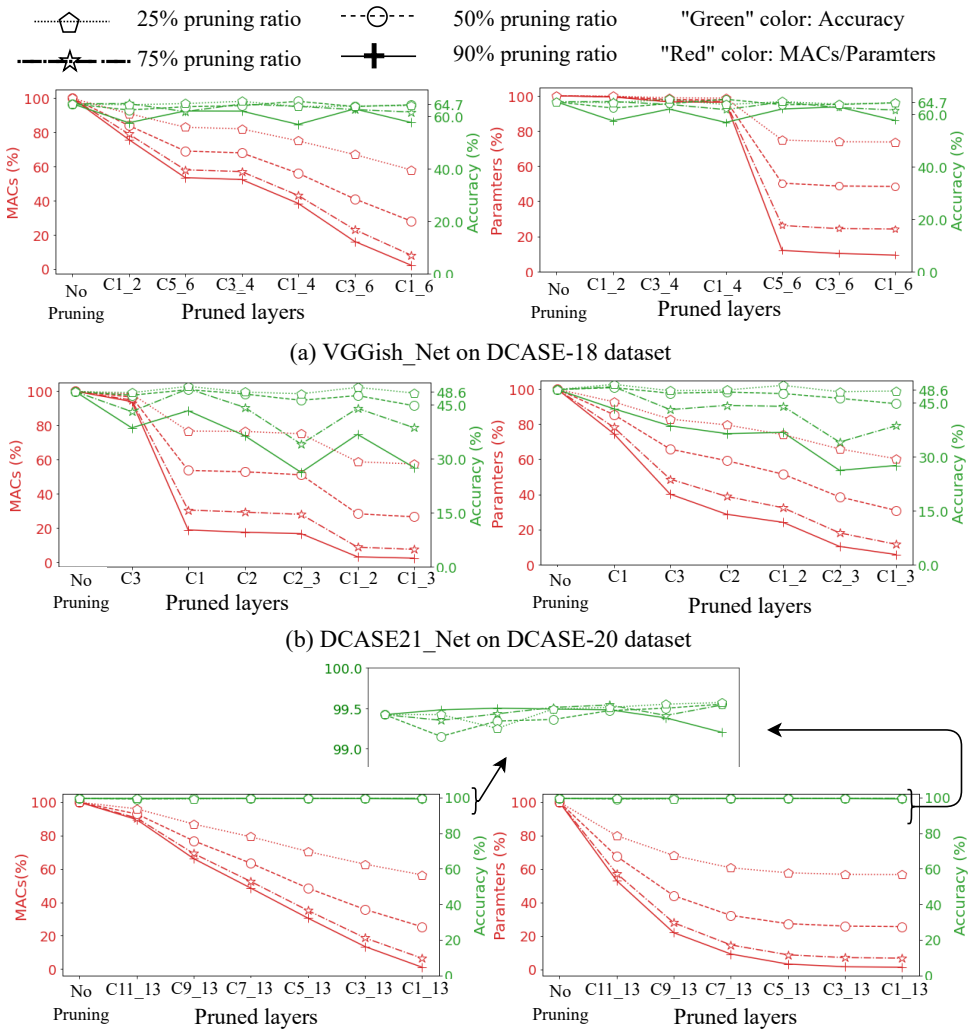


Figure 6: Accuracy, MACs and parameters across different pruned networks for (a) VGGish_Net, (b) DCASE21_Net and (c) VGG-16 network, when different subsets of convolutional layers are pruned at different pruning ratios. Here, "CA_B" as "pruned layers" means that convolution layers from A to B are pruned.

shown in Figure 6(a). For DCASE21_Net, both the MACs and the parameters are reduced by 40 percentage points when all convolutional layers (C1_3: C1 to C3) are pruned at $p = 25\%$ shown in Figure 6(b).

At $p = 50\%$, the accuracy drop across various pruned networks is less than 4 percentage points compared to that of the unpruned network for both VGGish_Net and DCASE21_Net as shown in Figure 6(a, b). For VGGish_Net, the number of MACs are reduced by 75 percentage points and the parameters are reduced by 55 percentage points, when all convolutional layers (C1_6: C1 to C6) are pruned at $p = 50\%$ shown in Figure 6(a). For DCASE21_Net, both the MACs and the parameters are reduced by 75 percentage points when all convolutional layers (C1_3: C1 to C3) are pruned at $p = 50\%$ shown in Figure 6(b).

At $p = 75\%$, the accuracy drop across various pruned networks is less than 5 percentage points and 10 percentage points compared to that of the unpruned network for both VGGish_Net and DCASE21_Net respectively as shown in Figure 6(a, b). The accuracy of the pruned networks degrades further at $p = 90\%$ and the drop in accuracy for VGGish_Net and DCASE21_Net is 10 percentage points and 20 percentage points respectively, when all layer are pruned at p = 90%.

Table 1: Comparison of accuracy, the MACs and the parameters among the (a) unpruned, (b) the pruned network obtained using the proposed pruning method and the same pruned network obtained in (b), however, trained from scratch.

Network	Accuracy (%)	MACs	Reduced MACs	Paramters	Reduced paramters
VGGish_Net	64.69	903M	-	55M	-
(a) Pruned VGGish_Net	64.30	253M	72%	26.82M	52%
(b) Pruned VGGish_Net-scratch	61.40				
DCASE21_Net	48.58	286M	-	46246	-
(a) Pruned DCASE21_Net	48.18	164M	43%	27906	40%
(b) Pruned DCASE21_Net-scratch	46.80				
VGGNet-MNIST	99.49	329M	-	15M	-
(a) Pruned VGGNet-MNIST	99.10	3.33M	99%	0.18M	99%
(b) Pruned VGGNet-MNIST-scratch	98.97				

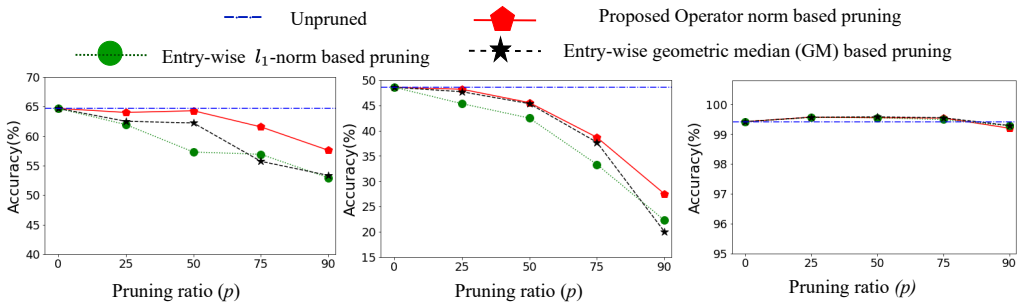
On the other hand, both the MACs and the parameters are reduced significantly by more than 75 percentage points when all convolutional layers of VGGish_Net and DCASE21_Net are pruned at $p = \{75\%, 90\%\}$ as shown in Figure 6(a, b).

In general, the MACs, the parameters and the accuracy decrease when various convolutional layers are pruned from 25% to 90% pruning ratio. The accuracy of the pruned DCASE21_Net reduces significantly from 0.2 percentage points to 20 percentage points compared to that of the unpruned network, when all layers are pruned with 25% to 90% pruning ratio. This might be due to the smaller network size of the DCASE21_Net, where eliminating large number of parameters at high pruning ratio results in under-fitting problem due to insufficient parameters.

VGG-16 for image classification : For VGG-16, the accuracy of the various pruned networks is reduced by less than 0.5% percentage points compared to that of the unpruned VGG-16 as shown in Figure 6(c). The number of MACs and the parameters are reduced from approximately 40 percentage points to 99 percentage points when pruning ratio across all convolutional layers (C1_13: C1 to C13) varies from 25% to 90% respectively.

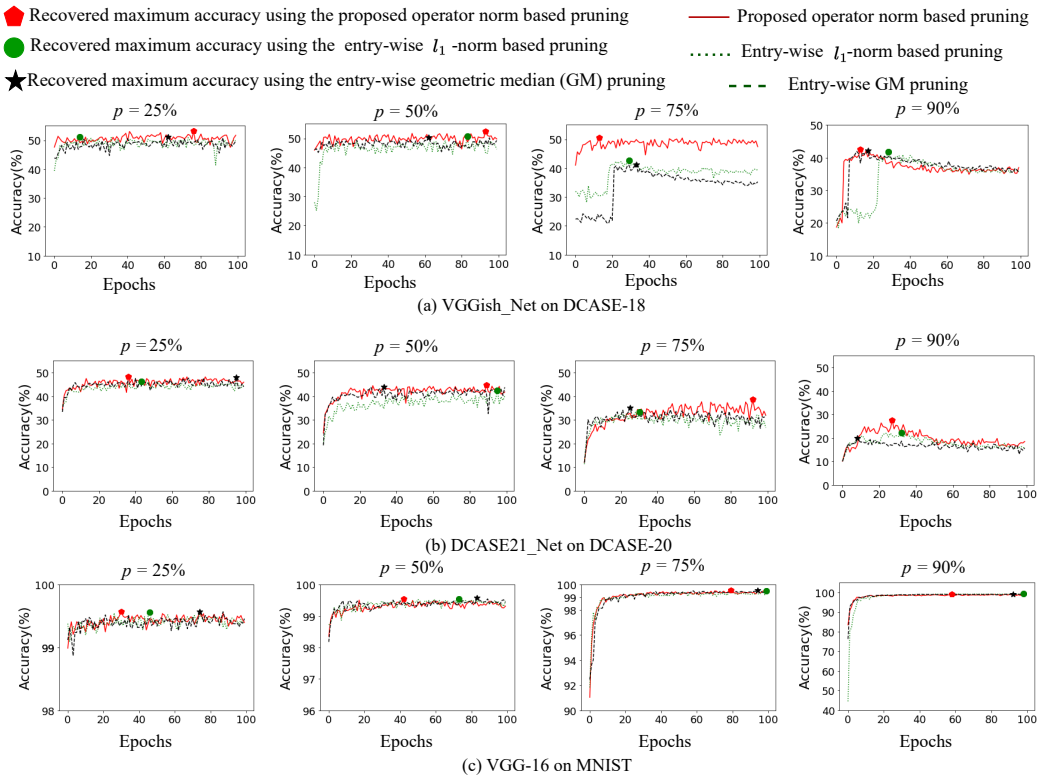
Next, Table 1 compares the accuracy, MACs and parameters in the unpruned network with that of (a) the pruned network having important filters obtained using the proposed pruning method as an initialisation before fine-tuning process, (b) the same pruned network as obtained in (a) and trained from scratch with random initialization. For comparison, we choose the pruned network in (a) which gives a maximum drop of 0.5 percentage point in accuracy compared to that of the unpruned network. For various networks, the parameters and the MACs are reduced significantly with a marginal drop in accuracy compared to that of the unpruned network. The accuracy of (a) the pruned network obtained from the proposed pruning method is better than (b) the same pruned network trained from scratch. This shows that it is beneficial to train a large network, and then perform pruning to reduce the parameters rather than training the same size pruned network from scratch.

Comparison with other methods: Figure 7 compares the accuracy of the proposed operator norm pruning method with that of the entry-wise l_1-norm and the geometrical median (GM) based methods, when all layers in the unpruned network are pruned at different pruning ratios. For VGGish_Net and DCASE21_Net, the accuracy obtained using the proposed operator norm based pruning method is better than that of the entry-wise norm methods particularly when a large number of filters ($p = 90\%$) are pruned from the network as shown in Figure 7(a, b). Moreover, the pruned network obtained using the proposed operator norm based pruning method recovers faster to regain the accuracy during the fine-tuning process compared to that of the other pruning methods as shown in Figure 8(a, b) at different pruning ratios. For VGG-16, the accuracy obtained using the proposed operator norm pruning method is similar to the other methods as shown in Figure 7(c) and recovers faster compared to that of the other pruning methods as shown in Figure 8(c) at different pruning ratios. This shows that selecting important filters based on the contribution of filters in producing output is advantageous compared to considering entry-wise norm of the filters due to similar or better performance and faster recovery of the performance by the pruned network after eliminating filters from the unpruned network.



(a) VGGish_Net on DCASE-18 (b) DCASE21_Net DCASE-20 (c) VGG-16 on MNIST

Figure 7: Accuracy obtained using the pruned network when all layers are pruned at different pruning ratio using entry-wise l_1-norm method, geometrical-median (GM) method and the proposed operator norm pruning method across various networks.



(a) VGGish_Net on DCASE-18

(b) DCASE21_Net on DCASE-20

(c) VGG-16 on MNIST

Figure 8: Convergence plots showing regain in the accuracy during fine-tuning of the pruned network when the pruned network is obtained by pruning all convolutional layers at different pruning ratios.

5 Conclusion

This paper presents a passive filter pruning method which reduces the computational complexity and the memory of the unpruned CNN significantly at marginal drop in accuracy. We observe that while selecting important filters based on their contribution in producing outputs improve the performance in terms of accuracy or recovery rate during fine-tuning of the pruned network compared to that the entry-wise norm of the filters particularly when large number of filters are pruned from the network.

REFERENCES

Misha Denil, Babak Shakibi, Laurent Dinh, M.A. Ranzato, and Nando De Freitas. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pp. 2148–2156, 2013.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pp. 1269–1277, 2014.

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

Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. In *Pattern Recognition*, volume 77, pp. 354–377. Elsevier, 2018.

Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: Efficient inference engine on compressed deep neural network. In *ACM SIGARCH Computer Architecture News*, volume 44, pp. 243–254. ACM New York, NY, USA, 2016.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, 2019.

Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. CNN Architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. In *arXiv preprint arXiv:1607.03250*, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient ConvNets. In *International Conference on Learning Representations*, 2017.

Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 730–734. IEEE, 2015.

Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems*, pp. 855–863, 2014.

Jian-Hao Luo and Jianxin Wu. An entropy-based pruning method for CNN compression. In *arXiv preprint arXiv:1706.05791*, 2017.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. ThiNet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5058–5066, 2017.

Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. ThiNet: pruning CNN filters for a thinner net. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 41, pp. 2525–2538, 2018.

Irene Martín-Morató, Toni Heittola, Annamaria Mesaros, and Tuomas Virtanen. Low-complexity acoustic scene classification for multi-device audio: Analysis of DCASE 2021 Challenge systems. *DCASE Workshop*, 2021.

Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. A multi-device dataset for urban acoustic scene classification. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop*, pp. 9–13, November 2018.

Adam Polyak and Lior Wolf. Channel-level acceleration of deep face representations. In *IEEE Access*, volume 3, pp. 2163–2175, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2074–2082, 2016.

Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *International Conference on Learning Representations*, 2018.