AI-ML SYSTEMS

SECOND INTERNATIONAL CONFERENCE ON AI-ML SYSTEMS

HYBRID, BANGALORE, INDIA

12-15 OCTOBER 2022

AN INITIATIVE OF THE COMSNETS ASSOCIATION

In association with

acm In-Cooperation

acm SIGAI KDD ACM SIGMOD

Technical Co-Sponsorship

IEEE Computational Intelligence Society

MeitY Government of India

Accurate and Efficient Channel pruning via Orthogonal Matching Pursuit

Kiran Purohit,
Anurag Parvathgari, Soumi Das and
Sourangshu Bhattacharya

Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur
Kharagpur- 721302, West Bengal, India

Outline

1. Introduction
2. Related Work
3. Motivation
4. Problem Definition
5. Contribution
   - 3.1. Identifying Multiple Channels for Pruning
   - 3.2 Weight compensation for multiple channel pruning
   - 3.3. Optimal filter search
6. Results and Analysis
7. Conclusion
8. References

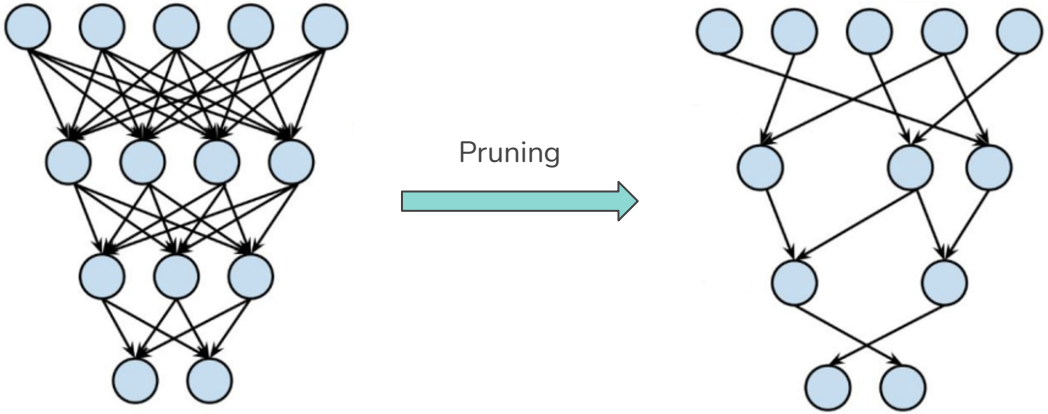
Introduction

Burden of CNNs ——ResNet-152	Filter Pruning ——Benefits
60.2 million parameters and 231MB storage spaces;	reduces the storage usage
380MB memory footprint	decreases the memory footprint
11.3 billion float point operations (FLOPs).	accelerates the inference

Introduction

Network Pruning

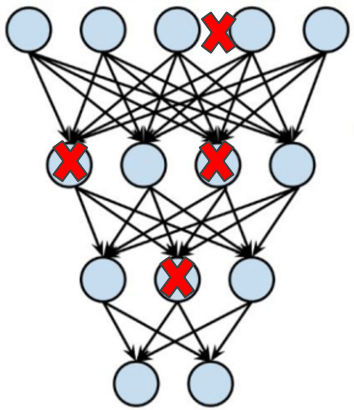
Given a pre-trained network Φ(.), the goal is to compress the network while maintaining the high performance as much as possible by removing the unnecessary parameters.



Pre-trained original network Φ(.)

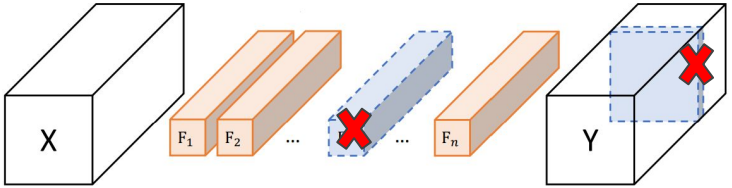
Final pruned network Φ'(.)

Related Work



Weights or Node Pruning

- Pruning applied to early DNN
- Check the importance of each weight or node
- Practical acceleration could not be achieved

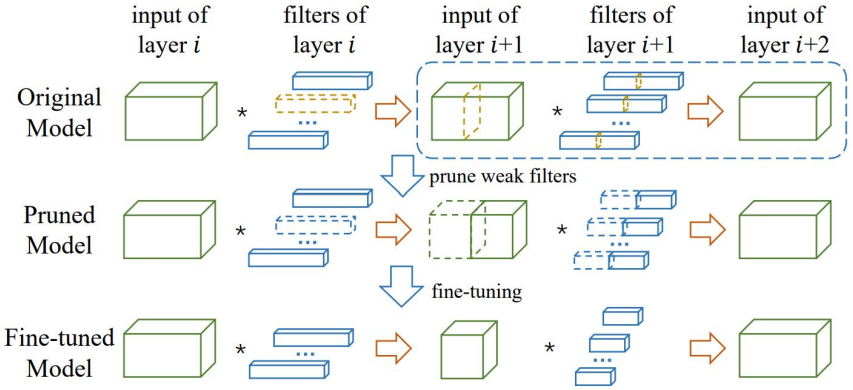


Filter or Channel Pruning

- Widely used for modern CNNs
- Remove the entire filter or channel at once
- Helps the practical acceleration of the network

Related Work

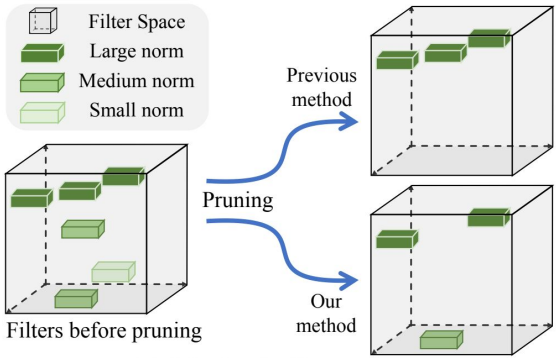
Thinet **“ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression” ICCV 2017**



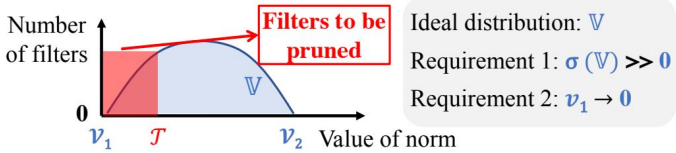
- Removes each channel one-by-one, and record the output feature map difference at each step.
- Selects the channel with the lowest difference.
- Greedy way of selecting channel, and layer wise pruning.

Related Work

FPGM **“Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration” CVPR 2019**



(a) Criterion for filter pruning

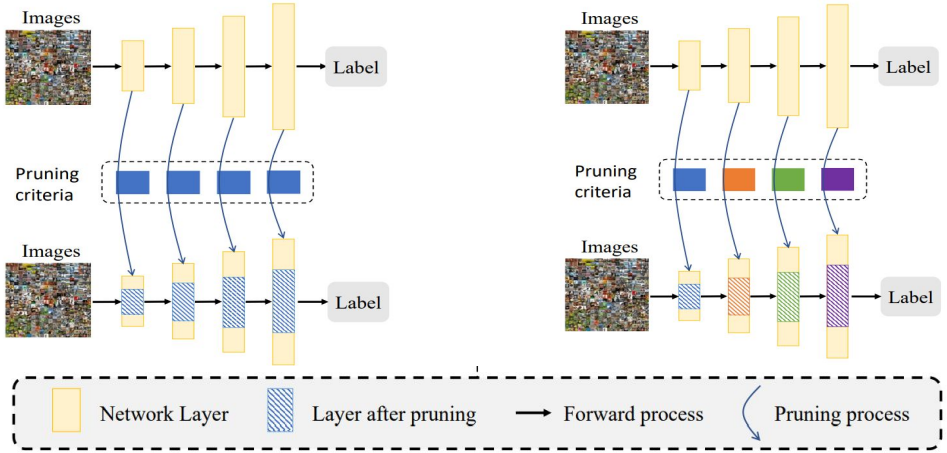


(b) Requirements for norm-based criterion

- FPGM presents a new view that medium norm is less important, departing from the existing view that small norm is less important.
- Remove the filter with the medium norm (norm close to the geometric median).
- Specific theoretical background is not sufficient.

Related Work

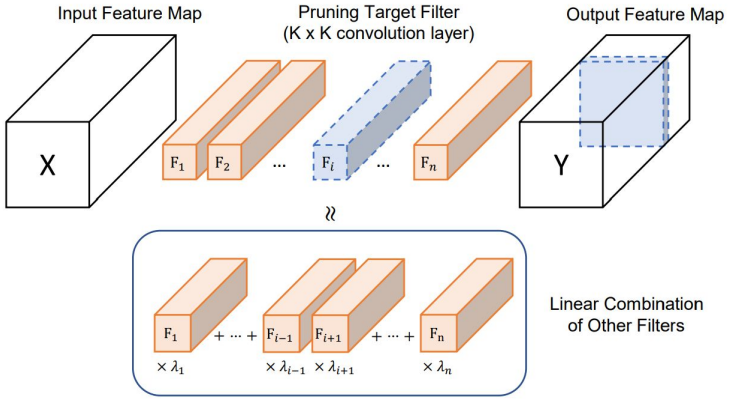
LFPC **“Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration” CVPR 2020**



- Existing methods usually utilize pre-defined pruning criteria, such as ℓp-norm, to prune unimportant filters.
- Filters of different layers have various distributions.
- Layer-by-layer pruning methods fails to consider that all the layers in the network work collaboratively.
- LFPC adaptively select the appropriate pruning criteria for different functional layers.

Related Work

LRF **“Linearly Replaceable Filters for Deep Network Channel Pruning” AAAI 2021**



- LRF suggests that *a filter that can be approximated by the linear combination of other filters is replaceable*

Related Work

- In a layer, each filter has the same dimension. We can approximate each filter as a linear combination of the other filters

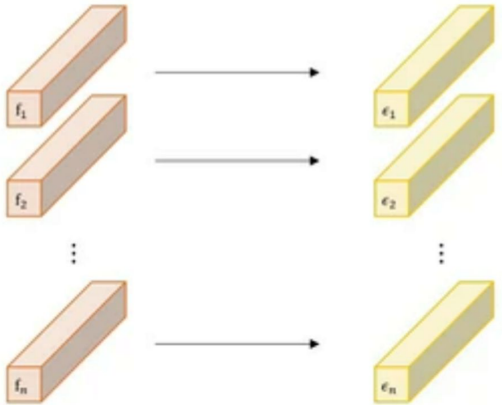
$$f_{:,j} = \sum_{l \neq j} \lambda_{j,l} f_{:,l} + \epsilon_j \qquad (1)$$

Here, ϵ = approximation error and $\lambda_{j,l}$ = weight coefficient of the respective filters

- Each $\lambda_{j,l}$ can be found easily by solving following minimization problem

$$\min ||f_{:,j} - \sum_{l \neq j} \lambda_{j,l} f_{:,l}||^2 \qquad (2)$$

- Each filter is approximated by linear combination of other filters
- Corresponding approximation error ϵ_i are computed
- Select the i-th filter with the smallest $||\epsilon_i||$
- Remove the filter and every connections together



Limitations of LRF

- The filter with minimum approximation error ($argmin_j||\epsilon_j||$) gets pruned at the end of each turn.
- However, for pruning filters by a fraction of $\boldsymbol{\beta}$ using LRF algorithm, one needs to prune the layer $\boldsymbol{\beta}$ of $\boldsymbol{N_c}$ times where N_c is the total number of filters in layer c.
- It is followed by fine-tuning the network each time after removing a filter.
- Hence, this approach is both slow and sub-optimal.
- In order to speed up the pruning method, we develop an algorithm that prunes the filters of a layer for a given fraction together, followed by fine tuning, thus reducing the pruning time by a reasonable margin.

Motivation

- The deeper and wider architectures of recent convolutional neural networks are responsible for superior performance in computer vision tasks.
- However, they also come with an enormous model size and heavy computational cost.
- Filter pruning is one of the methods applied to CNNs for compression and acceleration.
- Various techniques have been recently proposed for filter pruning.
- We address the limitation of the existing state-of-the-art method and motivate our setup.

Problem Definition

- To reduce the model size and heavy computational cost.
- To develop a novel method (FP-OMP) for filter selection using sparse approximation of filter weights.
- To address the problem of removal of uniform number of filters from all the layers of a network.
- To propose FP-OMP Search which solves the above problem.

Contribution

- Identifying Multiple Channels for Pruning
- Weight compensation for multiple channel pruning
- Optimal filter search

Identifying Multiple Channels for Pruning

We develop our algorithm for pruning multiple channels together. We develop an Orthogonal Matching Pursuit (OMP) based algorithm for filter pruning, Filter Pruning-OMP **(FP-OMP)** that addresses the limitations of LRF. Starting from Equation 2, we arrive at our formulation for sparse approximation as given in equation 3:

$$S^*, \lambda^* = \min_{|S| \leq (N_c - \beta * N_c), \lambda} \sum_{j \in \{1,2,..,N_c\}} \|f_{:,j} - \sum_{l \in S} \lambda_{j,l} f_{:,l}\|^2 \quad , \forall j \in \{1, 2, \ldots, N_c\} \qquad (3)$$

where **S** is the set of the selected/retained filters of layer **l**, $\mathbf{N_c}$ is the total number of filter in layer **c**, and $\boldsymbol{\beta}$ is the pruning fraction of filters in layer **c**.

We can *approximate the pruned filters in terms of retained filters*. Therefore, we can rewrite the above equation as given in equation 4.

$$f_{:,j} = \sum_{l \in S} \lambda_{j,l} f_{:,l} + \epsilon_j \quad , \forall j \in \{1, 2, .., N_c\} \qquad (4)$$

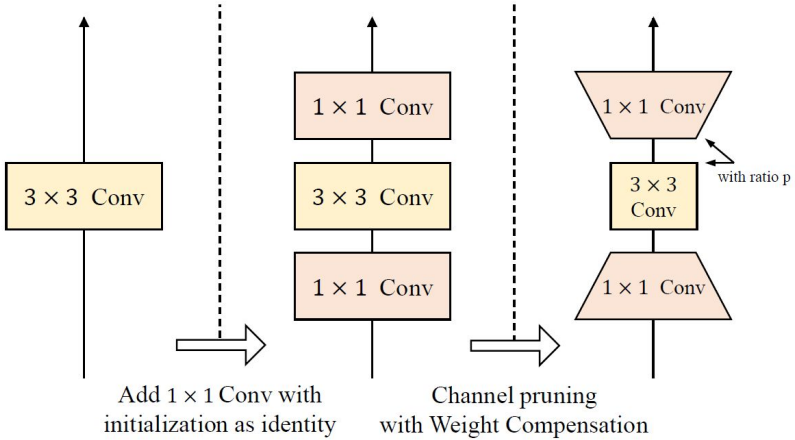
Identifying Multiple Channels for Pruning

FP-OMP describes our approach of selecting filters into **S** from layer **l**, that are to be retained. We can hence obtain the filters that are to be pruned from {1,2,...,n} \ S. The key idea behind the algorithm is to

(a) develop a sparse approximation approach that is faster than the adopted matrix computation approach in LRF.

(b) prune the network, for a given layer, all at once for a given fraction, compared to the one-at-a-time filter pruning approach in LRF, thus making the pruning method efficient.

Weight Compensation



- Before we prune a layer, we add 1x1 convolutional layer at the top and bottom.
- When we prune a filter, we modify the weight value of 1x1 convolutional layer appropriately.
- Then the loss change can be further reduced.

Weight compensation for multiple channel pruning

LRF had proposed the weight compensation module, for a single filter pruning, for two purposes:

- The difference in output feature map of the pruned and unpruned layer will get adjusted by the updation of weights of the $1x1$ convolution.
- Usage of $1x1$ convolution enables the pruning of any network, regardless of its architecture.

We adopt this module and derive the compensated weights as per our framework for multiple channel pruning.

For the output channel pruning,

$$g'_{l,:} = g_{l,:} + \sum_{j \in S^c} \lambda_{j,l} * g_{j,:} \quad , \forall l \in S$$

For the input channel pruning,

$$g'_{:,l} = g_{:,l} + \sum_{j \in S^c} \lambda_{j,l} * g_{:,j} \quad , \forall l \in S$$

Optimal filter search

- One other caveat of the work of LRF was sub-optimality.
- To find the optimal set of filters across the entire network, we discard the necessity of removal of uniform number of filters from each layer.
- **FP-OMP Search** proceeds by pruning a batch of filters from each layer followed by assessing the performance of the model in the pruned stage, and then putting back the removed filters.
- We keep a running record of the performance metric (acc) across all the layers.
- Eventually, the filters from a particular layer, on removing which gives the maximum accuracy compared to removing filters from other layers, are finally pruned from the network.
- This continues till we achieve the pruning criteria.

Experimental Results

Experimental Setting

Model Used

- ResNet-32, Resnet-34, Resnet-56
- Optimizer Used- SGD, batch size- 128, loss function- cross entropy loss, 1 epoch fine tune after pruning one layer and 300 after complete pruning
- Activation function- Relu

Performance Metric

Pruned Accuracy, Accuracy Drop, Parameters Drop, FLOPs Drop, Total Time taken for Pruning and Fine Tuning

Database Description

- CIFAR-10 - 10 Classes, 50k training images, 10k test images
- CIFAR-100 - 100 Classes, 50k training images, 10k test images
- Tiny Imagenet - 200 Classes, 0.1M images

Results and Analysis

Models	Method	Baseline Acc	Pruned Acc	Acc ↓	Param ↓	FLOPs ↓
ResNet-32	SFP [12]	92.63%	92.08%	0.55%	-	41.5%
	LFPC [11]	92.63%	92.12%	0.51%	-	52.6%
	FPGM [13]	92.63%	91.93%	0.70%	-	53.2%
	LRF [19]	92.63%	92.66%	-0.03%	63.3%	62.55%
	FP-OMP	**92.63%**	**92.79%**	**-0.16%**	**63.3%**	**62.55%**
	FP-OMP Search	**92.63%**	**92.81%**	**-0.18%**	**58.58%**	**44.9%**
ResNet-56	DCP [34]	93.80%	93.79%	0.01%	70.3%	47.1%
	HRank [22]	93.80%	93.17%	0.63%	42.4%	50.0%
	SFP [12]	93.80%	93.26%	0.54%	-	52.6%
	FPGM [13]	93.80%	93.49%	0.31%	-	52.6%
	LFPC [11]	93.80%	93.24%	0.56%	-	52.9%
	GBN [32]	93.80%	93.43%	0.37%	42.5%	55.1%
	LRF [19]	93.80%	93.85%	-0.05%	63.3%	62.55%
	FP-OMP	**93.80%**	**94.03%**	**-0.23%**	**63.3%**	**62.55%**
	FP-OMP Search	**93.80%**	**94.08%**	**-0.28%**	**56.50%**	**43.32%**

Table 1: Performance comparison of FP-OMP and FP-OMP Search for ResNet-32 and ResNet-56 on CIFAR-10 for 50% pruned filters of the network.

- We can clearly see that FP-OMP and FP-OMP Search outperform other pruning algorithms.
- They not only increase the accuracy but drop in params and flops is equivalent or more compared to other methods

Results and Analysis (Cont.)

Models	Method	Baseline Acc	Pruned Acc	Acc ↓	Param ↓	FLOPs ↓
ResNet-32	LRF [19]	68.78%	68.78%	-0.07%	62.5%	62.54%
	FP-OMP	**68.78%**	**69.05%**	**-0.27%**	**62.5%**	**62.54%**
	FP-OMP Search	**68.78%**	**69.11%**	**-0.33%**	**50.72%**	**53.18%**
ResNet-56	LRF [19]	69.98%	70.07%	-0.09%	63%	62.92%
	FP-OMP	**69.98%**	**70.39%**	**-0.41%**	**63%**	**62.92%**
	FP-OMP Search	**69.98%**	**70.43%**	**-0.45%**	**50.72%**	**53.18%**

Table 2: Performance comparison of FP-OMP and FP-OMP Search for ResNet-32 and ResNet-56 on CIFAR100 for 50% pruned filters of the network.

Models	Method	Baseline Acc	Pruned Acc	Acc ↓	Param ↓	FLOPs ↓
ResNet-34	LRF [19]	64.18%	62.86%	1.32%	62.79%	60.76%
	FP-OMP	**64.18%**	**65.68%**	**-1.50%**	**62.79%**	**60.76%**
	FP-OMP Search	**64.18%**	**65.75%**	**-1.57%**	**51.67%**	**55.73%**

Table 3: Performance comparison of FP-OMP and FP-OMP Search for ResNet-34 on TinyImagenet for 50% pruned filters of the network.

Results and Analysis (Cont.)

Models	Method	Pruning Time (hr)	Fine Tuning Time (hr)	Total Time (hr)
CIFAR10				
ResNet-32	**LRF**	0.58	3.43	4.01
	FP-OMP	**0.54**	**3.41**	**3.95**
	FP-OMP Search	13.63	3.45	17.08
ResNet-56	**LRF**	1.80	4.27	6.07
	FP-OMP	**1.55**	**4.25**	**5.8**
	FP-OMP Search	58.84	4.33	63.17
CIFAR100				
ResNet-32	**LRF**	0.60	3.38	3.98
	FP-OMP	**0.48**	**3.37**	**3.85**
	FP-OMP Search	12.97	3.39	16.36
ResNet-56	**LRF**	1.61	4.09	5.70
	FP-OMP	**1.57**	**4.07**	**5.64**
	FP-OMP Search	54.01	4.11	58.12

Table 4: Time comparison of different methods on ResNet for channel pruning on CIFAR-10 and CIFAR-100 dataset.

Results and Analysis (Cont.)

Output Channel											
Block 1	Layers	1	2	3	4	5	6	7	8	9	10
	Before/After	16/1	16/1	16/11	16/11	16/11	16/11	16/16	16/11	16/1	16/1
	Percent removed	**93.75**	**93.75**	31.25	31.25	31.25	31.25	0	31.25	**93.75**	**93.75**
Block 2	Layers	11	12	13	14	15	16	17	18	19	20
	Before/After	32/22	32/22	32/2	32/2	32/27	32/27	32/2	32/2	32/27	32/17
	Percent removed	31.25	31.25	**93.75**	**93.75**	15.62	15.62	**93.75**	**93.75**	15.62	46.87
Block 3	Layers	21	22	23	24	25	26	27	28	29	30
	Before/After	64/64	64/54	64/14	64/14	64/9	64/4	64/34	64/29	64/64	64/54
	Percent Removed	0	15.62	**78.12**	**78.12**	**85.93**	**93.75**	46.87	54.68	0	15.62
Input Channel											
Block 1	Layers	1	2	3	4	5	6	7	8	9	10
	Before/After	16/1	16/1	16/11	16/6	16/11	16/11	16/16	16/16	16/1	16/1
	Percent removed	**93.75**	**93.75**	31.25	62.5	31.25	31.25	0	0	**93.75**	**93.75**
Block 2	Layers	11	12	13	14	15	16	17	18	19	20
	Before/After	16/16	32/17	32/2	32/2	32/37	32/27	32/7	32/2	32/17	32/17
	Percent removed	0	46.87	**93.75**	**93.75**	0	15.62	**78.12**	**93.75**	46.87	46.87
Block 3	Layers	21	22	23	24	25	26	27	28	29	30
	Before/After	32/32	64/59	64/9	64/14	64/9	64/4	64/39	64/24	64/64	64/59
	Percent Removed	0	7.81	**85.93**	**78.12**	**85.93**	**93.75**	39.06	62.5	0	7.81

Table 5: Percentage removal of filters from each layer of ResNet-32 on CIFAR-100 dataset using FP-OMP Search method with overall 50% removal of filters from the ResNet32.

Results and Analysis (Cont.)

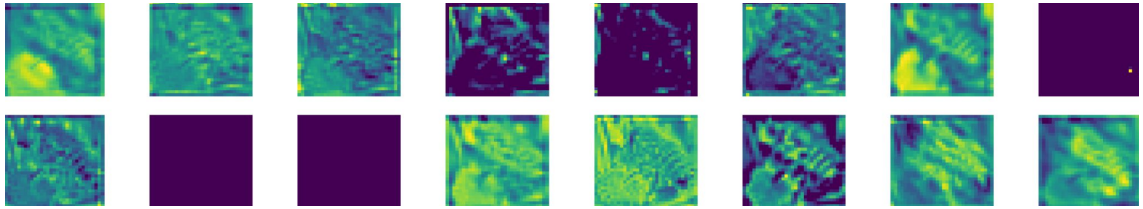


Figure: Visualisation of output feature map of ResNet-32 4^{th} layer on CIFAR-100

- Feature map of Layer 4 has a diverse set of filter outputs, indicates its usefulness in capturing different features of the inputs. Our FP-OMP Search prunes only 31.25% of its filters.

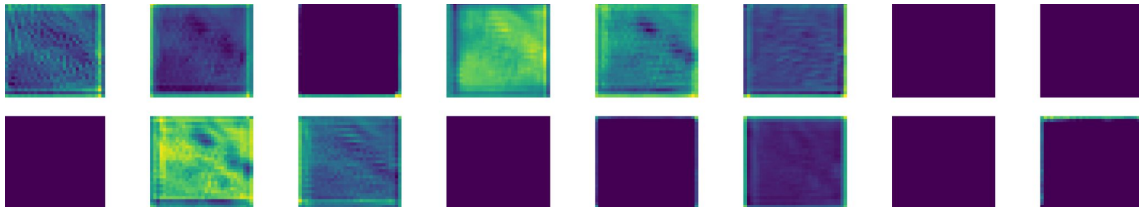


Figure: Visualisation of output feature map of ResNet-32 10^{th} layer on CIFAR-100

- Feature map outputs from Layer 10 looks very similar, denoting its redundancy in filter outputs. 93.75% of its filters are removed by FP-OMP Search.

Conclusion

- We proposed the FP-OMP and FP-OMP Search algorithms, a fresh and efficient channel pruning technique.
- It is a novel pruning criterion that chooses the channel using a sparse approximation method.
- Regardless of kernel size, block type, or even architectures, it shows a good performance across all of them.
- Extensive experiments on the 3 datasets with 2 different architectures prove our hypothesis.

References

1. Xuanyi Dong and Yi Yang. 2019. Network pruning via transformable architecture search. Advances in Neural Information Processing Systems 32 (2019).
2. Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. 2020. Learning filter pruning criteria for deep convolutional neural networks acceleration. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2009–2018.
3. Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. 2019. Filter pruning via geometric median for deep convolutional neural networks acceleration. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 4340–4349.
4. Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE international conference on computer vision. 5058–5066
5. Donggyu Joo, Eojindl Yi, Sunghyun Baek, and Junmo Kim. 2021. Linearly replaceable filters for deep network channel pruning. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. 8021–8029.

THANK YOU
FOR
YOUR ATTENTION!!!

https://github.com/kiranpurohit/

@kiranpurohit08