Scalable and Accurate Channel pruning

CNeRG talk series

Kiran Purohit (20CS91R09)

Advisor: **Prof. Sourangshu Bhattacharya**

Dept. of Computer Science & Engineering
IIT Kharagpur

Outline

1. Introduction
2. Related Work
3. **Accurate and Efficient Channel pruning via Orthogonal Matching Pursuit** *(AIMLSystems 2022)*
    a. Contribution
        i. Proposed **FP-OMP** for pruning multiple Channels
        ii. Proposed **FP-OMP-Search** for non-uniform pruning
    b. Results and Analysis
    c. Conclusion
4. **A Hierarchical Approach to Non-Uniform Filter-Pruning for highly-efficient CNNs** *(AAAI 2023 submitted)*
    a. Contribution
        i. Hierarchical approach for non-uniform pruning
        ii. Proposed **HBGS** and **HBGTS** for *layer selection*
    b. Results and Analysis
    c. Conclusion

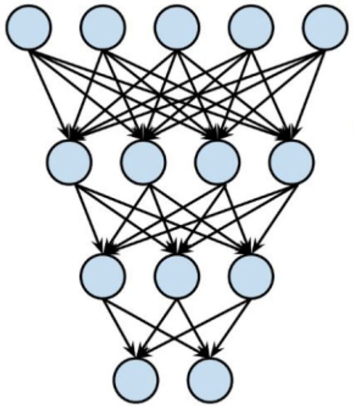
Introduction

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

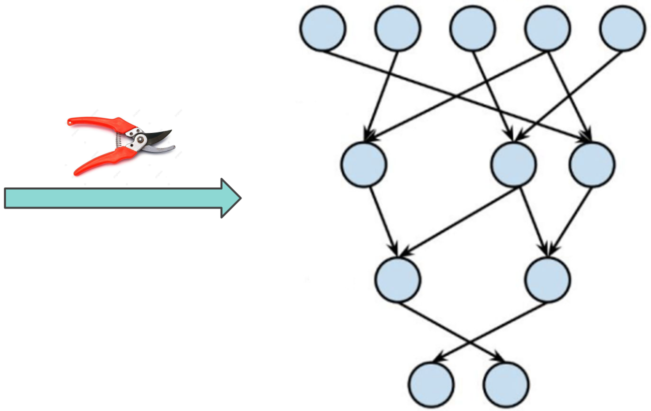
Introduction

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.

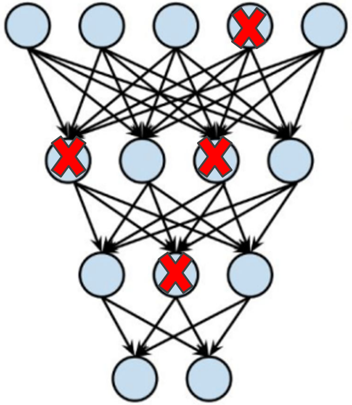


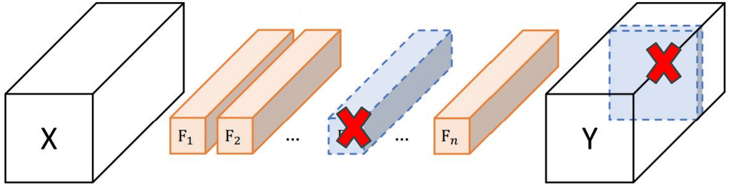
Pre-trained original network Φ(.)



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

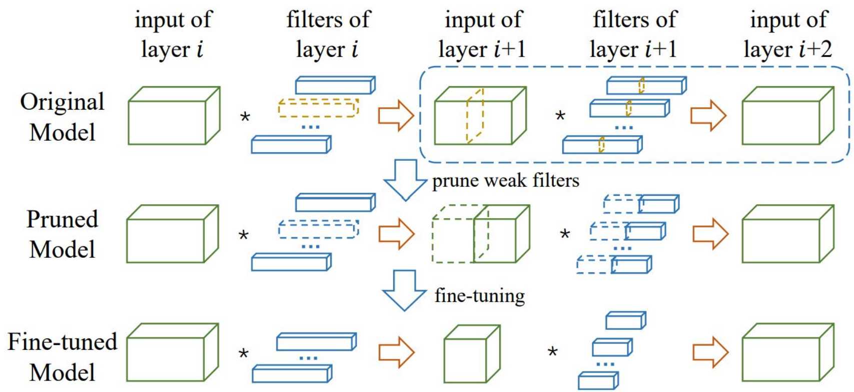
Filter or Channel Pruning

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

Related Work

Thinet

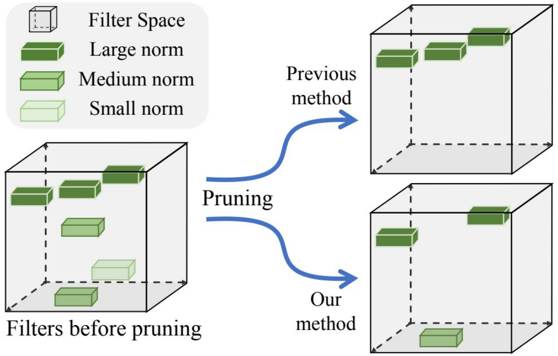
"ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression" ICCV 2017



- ❖ 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.

Related Work

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

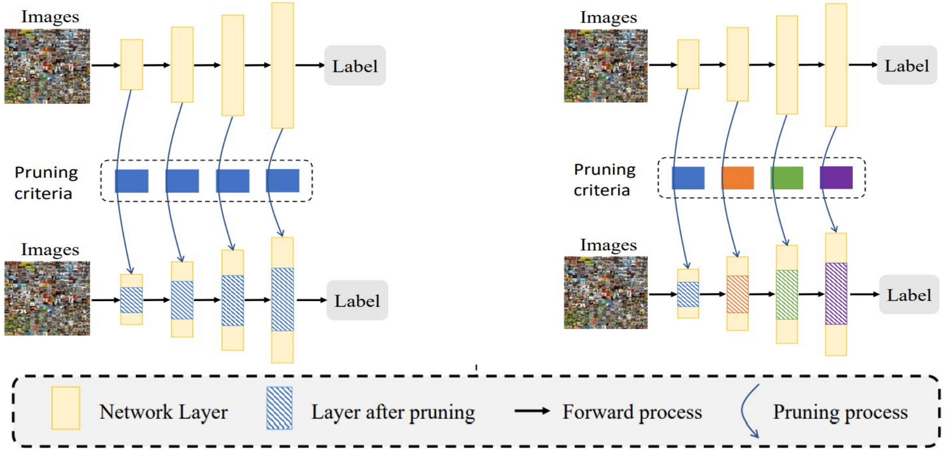


(a) Criterion for filter pruning

- ❖ Traditionally the filter weight, with small norm was regarded as less important filter.
- ❖ FPGM presents that filter with median norm is less important and can be removed

Related Work

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

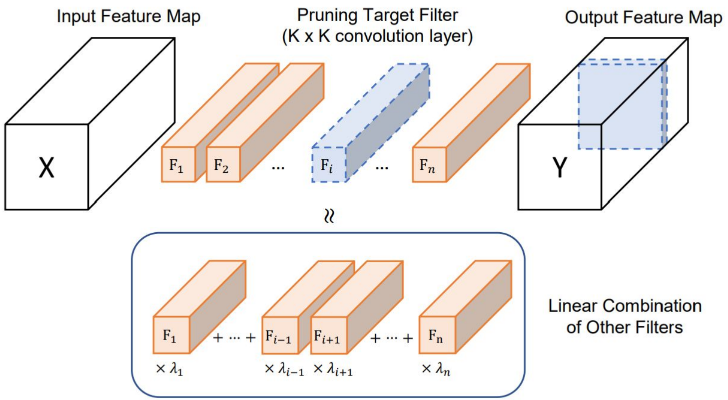


- ❖ Existing methods usually utilize pre-defined pruning criteria, such as ℓp-norm, to prune unimportant filters from each layer.
- ❖ LFPC adaptively select the appropriate pruning criteria for different layers.

Related Work

LRF

"Linearly Replaceable Filters for Deep Network Channel Pruning" AAAI 2021



❖ LRF suggests that *we can replace the filter that can be approximated by the linear combination of other filters*

Related Work

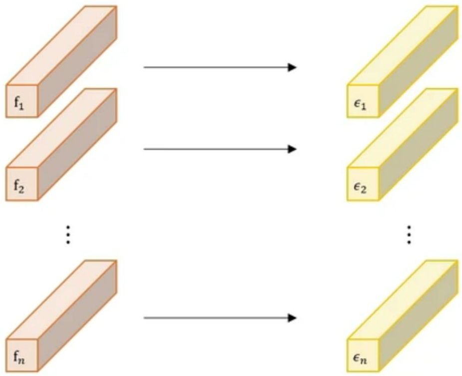
❖ In a layer, we can approximate each filter as a linear combination of the other filters

$$f_{:,j} = \sum_{l \neq j} \lambda_{j,l} f_{:,l} + \epsilon_j$$

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

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

$$\min_{\lambda_{j,:}} ||f_{:,j} - \sum_{l \neq j} \lambda_{j,l} f_{:,l}||^2$$



Remove the i^{th} filter with the smallest $||\epsilon_i||$

Limitations of LRF

- For pruning a layer by a given fraction $\boldsymbol{\beta}$, one needs to prune $\boldsymbol{\beta}$ of the total number of filters in that layer. LRF does 1 epoch of SGD update each time after removing a filter. Hence, this approach is **slow.**

- LRF does not provide any optimality guarantee for the removed filter. Hence, this approach is **sub-optimal**.

- In order **to speed up** the pruning method:
  - We prune the filters of a layer for a given fraction together
  - Followed by just 1 epoch of SGD update

Accurate and Efficient Channel pruning via Orthogonal Matching Pursuit

(AIMLSystems 2022)

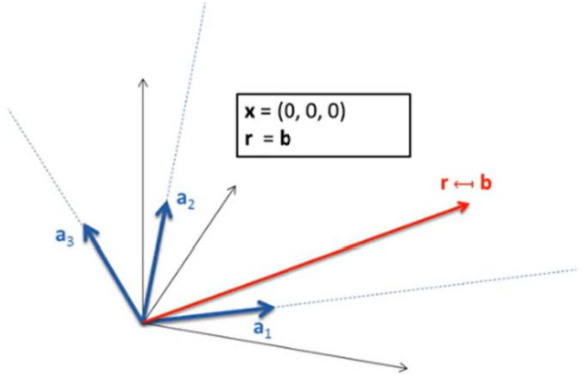
Contribution

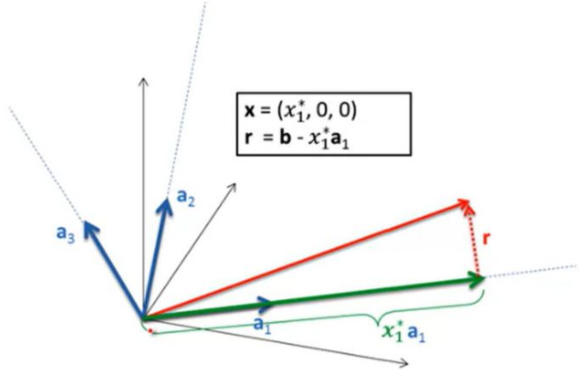
- Proposed **FP-OMP** for pruning multiple Channels
- Proposed **FP-OMP-Search** for non-uniform pruning

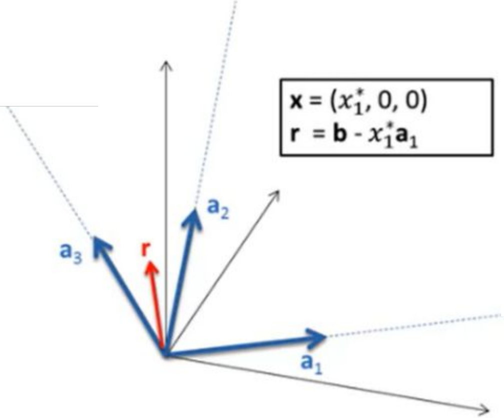
Orthogonal Matching Pursuit

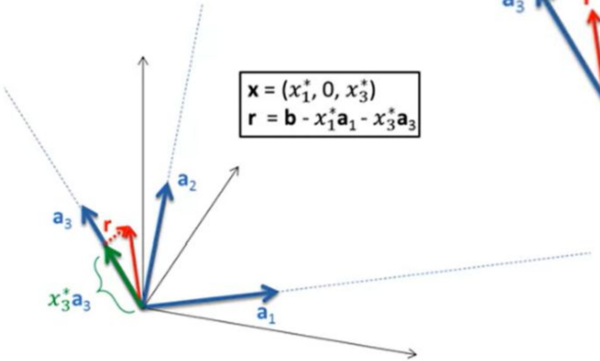
$$\boxed{\begin{array}{c} \underset{\mathbf{x}}{min} \, \|A\mathbf{x} - \mathbf{b}\|_2 \\ subject\, to \; \|\mathbf{x}\|_0 \le S \end{array}}$$

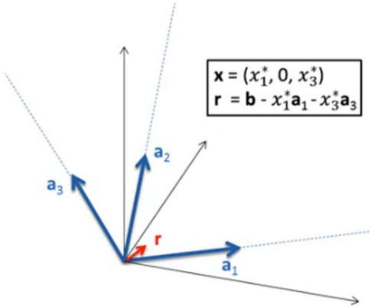
Input: A (with unit norm columns), $\mathbf{b}$, and S.
Initialize $\mathbf{r} = \mathbf{b}$ and $\Omega = \emptyset$.
While $\|\mathbf{x}\|_0 < S$
compute $x_j = \mathbf{a}_j^T \mathbf{r}$ for all $j \notin \Omega$
$i = \underset{j \notin \Omega}{argmax} \, |x_j|$
$\Omega \longleftarrow \Omega \cup \{i\}$
$\mathbf{x}_\Omega^* = \underset{\mathbf{x}}{argmin} \|A_\Omega \mathbf{x} - \mathbf{b}\|_2^2$
$\mathbf{r} \longleftarrow \mathbf{b} - A_\Omega \mathbf{x}_\Omega^*$











FP-OMP for Pruning Multiple Channels

We develop an Orthogonal Matching Pursuit (OMP) based algorithm for selecting retained filters of a layer into S. Hence filters that are to be pruned are {1,2,...n} \ S.

We can *approximate the pruned filters in terms of retained filters.*

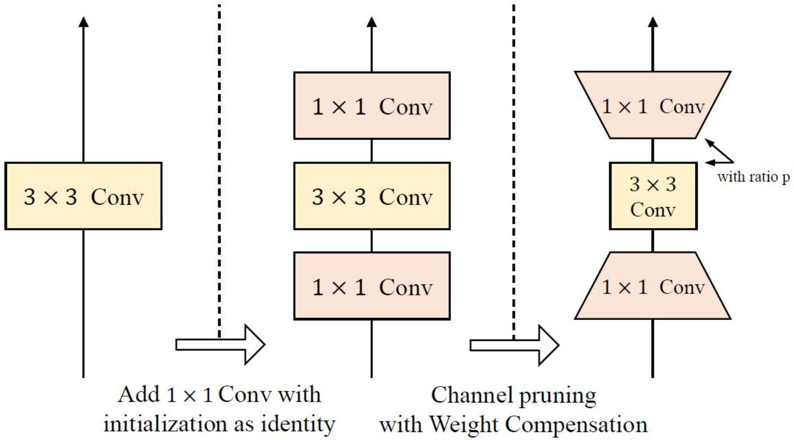
$$f_{:,j} = \sum_{l \in S} \lambda_{j,l} f_{:,l} + \epsilon_j, \forall j \notin S$$

We pose a sparse approximation problem for finding **S** and $\boldsymbol{\lambda}$

$$S^*, \lambda^* = \text{argmin}_{|S| \leq (1-\beta)n, \lambda} \sum_{j \in \{1,2,..,n\}} ||f_{:,j} - \sum_{l \in S} \lambda_{j,l} f_{:,l}||^2$$

where **S** is the set of the selected/retained filters in a layer, **n** is the total number of filter in that layer, and $\boldsymbol{\beta}$ is the pruning fraction

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:

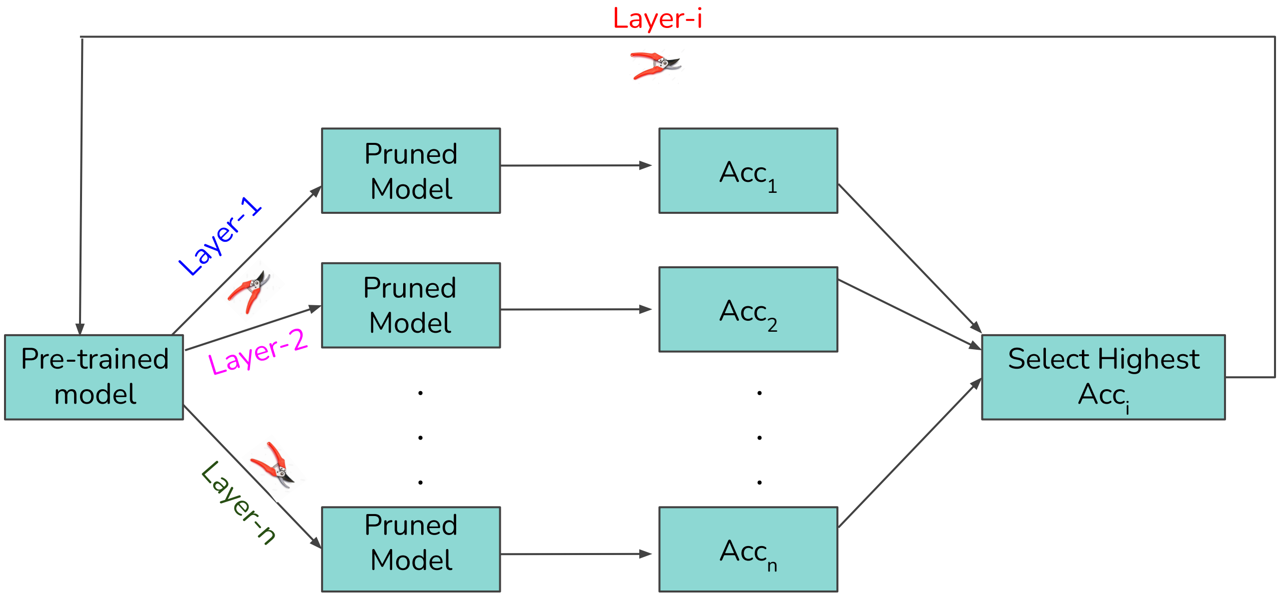
- The difference in output feature map of the pruned and unpruned layer will get adjusted by the updation of weights of the $1x1$ convolution.
- Usage of $1x1$ convolution enables the pruning of any network, regardless of its architecture.

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

FP-OMP-Search for non-uniform pruning



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.
- The 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.)

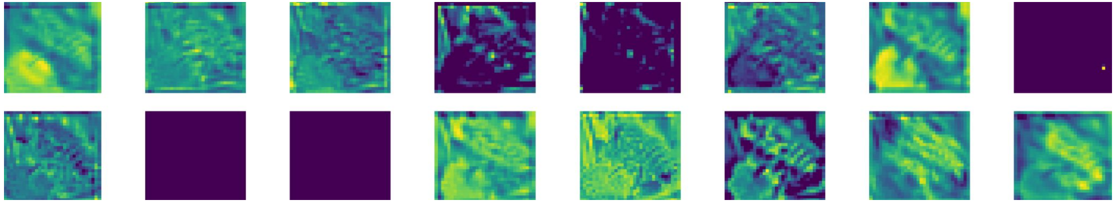


Figure: Visualisation of output feature map of ResNet-32 4^{th} layer on CIFAR-100

❖ 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.

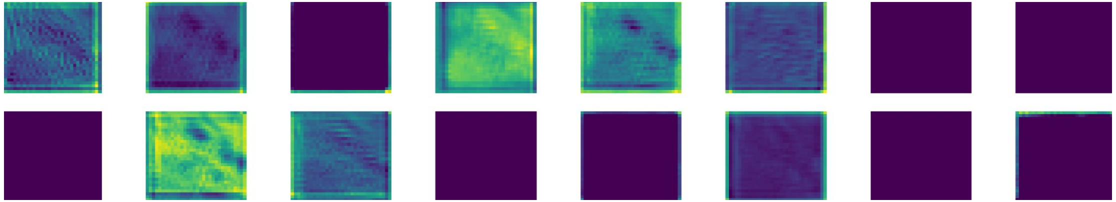


Figure: Visualisation of output feature map of ResNet-32 10^{th} layer on CIFAR-100

❖ 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

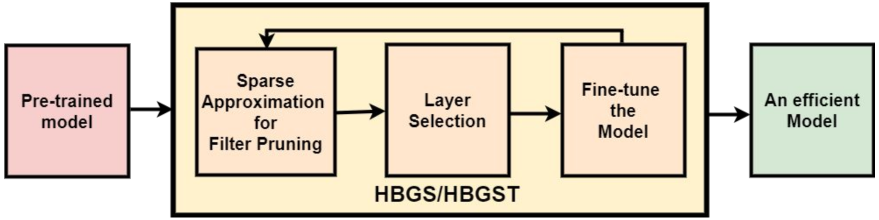
- We proposed FP-OMP and FP-OMP Search algorithms, a fresh and efficient channel pruning technique using a sparse approximation method.
- We performed extensive experiments on the 3 datasets with 2 different architectures.

A Hierarchical Approach to Non-Uniform Filter-Pruning for highly-efficient CNNs

Contribution

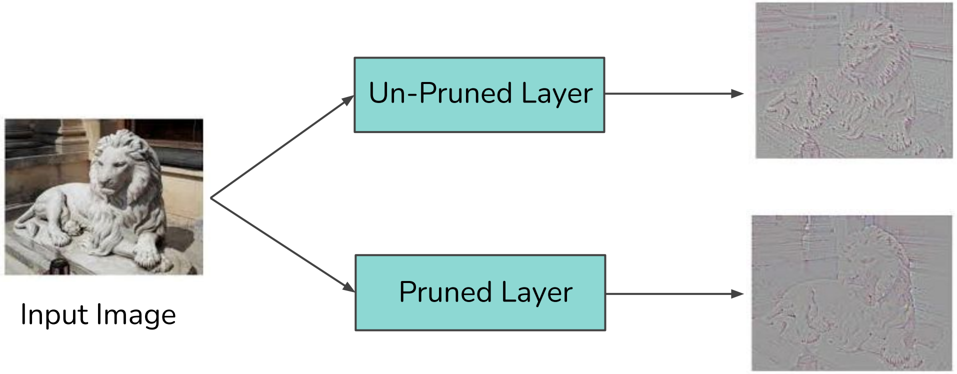
- We developed faster non-uniform pruning methods.
- We used a hierarchical scheme with two-levels:
  - filter pruning - this step identifies the most appropriate filters to be pruned from each layer.
  - layer selection - this step selects the best layer to currently prune from.

We apply these two steps iteratively to achieve a non-uniform pruning.



HBGS for Layer Selection

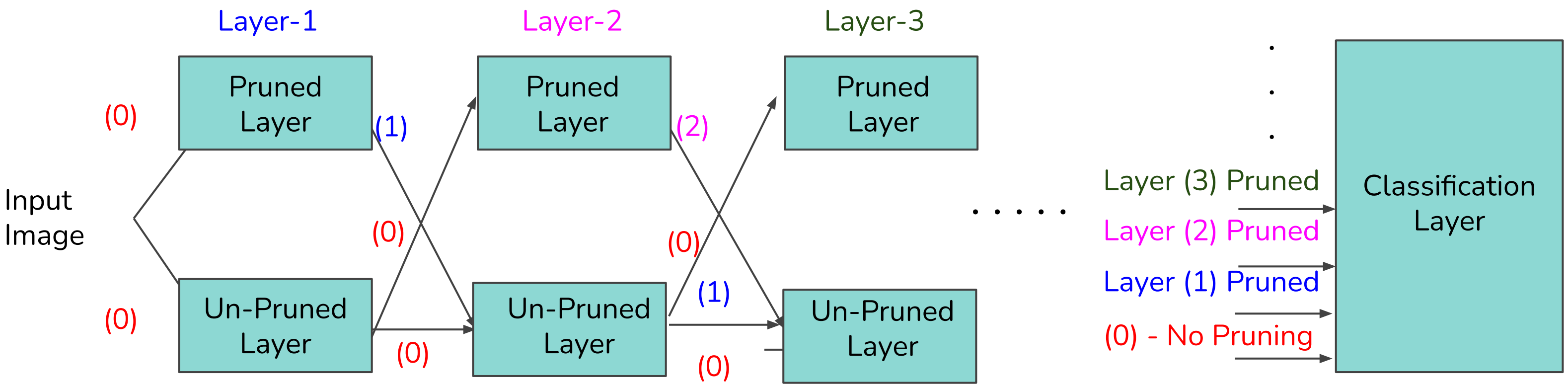
- We develop Hierarchical Backward Greedy Search (HBGS) for selecting the best layer to currently prune from.
- Key idea here is to calculate the relative reconstruction error between the pruned layer output and unpruned layer output
    - and then finally choose the layer with minimum error to currently prune from.



We want to minimise the difference b/w the output feature maps of pruned and unpruned layers

HBGTS for Layer Selection

- We develop Hierarchical Backward Greedy Tree Search (HBGTS) for selecting the best layer to currently prune from.
- Key idea here is to calculate the error in final layer output, if layer j ∈ {1, ..., C} is pruned
  - and then finally choose the layer with minimum error to currently prune from.



Results and Analysis

Method	VGG16/CIFAR100 Test Acc (%)	Acc ↓ (%)	Param ↓ (%)	FLOPs ↓ (%)	ResNet18/CIFAR10 Test Acc (%)	Acc ↓ (%)	Param ↓ (%)	FLOPs ↓ (%)
Dense	67.1 ± 0.01	0 ± 0	-	-	94.5 ± 0.02	0 ± 0	-	-
Random	55.5 ± 0.16	11.6 ± 0.16	98.0	86.0	86.3 ± 0.06	8.2 ± 0.06	93.7	75.0
EarlyCroP-S (Rachwan et al. 2022)	62.8 ± 0.52	4.3 ± 0.52	97.9	88.0	91.0 ± 0.52	3.5 ± 0.52	95.1	65.8
DLRFC (He et al. 2022)	63.5 ± 0.09	3.56 ± 0.09	97.1	53.7	-	-	-	-
SAP (Diao et al. 2023)	-	-	-	-	91.4 ± 0.03	3.1 ± 0.03	94.9	64.9
PL (Chen et al. 2023b)	63.5 ± 0.03	3.6 ± 0.03	97.3	87.9	-	-	-	-
LRF (Joo et al. 2021)	64.0 ± 0.31	3.1 ± 0.31	97.9	88.0	91.5 ± 0.37	3.0 ± 0.37	95.1	65.8
FP-OMP (Purohit et al. 2023)	$\mathbf{66.4} \pm 0.13$	0.7 ± 0.13	97.9	88.0	$\mathbf{93.1} \pm 0.17$	1.4 ± 0.17	95.1	65.8
FP-Backward	66.2 ± 0.11	0.9 ± 0.11	97.9	88.0	92.9 ± 0.15	1.6 ± 0.15	95.1	65.8
HBGS	$\mathbf{67.3} \pm 0.17$	-0.2 ± 0.17	98.3	89.6	$\mathbf{93.9} \pm 0.24$	0.6 ± 0.24	95.3	66.2
HBGS-B	67.2 ± 0.15	-0.1 ± 0.15	98.1	89.4	93.7 ± 0.22	0.8 ± 0.22	95.2	66.0
HBGTS	$\mathbf{67.8} \pm 0.23$	-0.7 ± 0.23	98.5	89.8	$\mathbf{94.7} \pm 0.28$	-0.2 ± 0.28	95.6	66.7
HBGTS-B	67.6 ± 0.21	-0.5 ± 0.21	98.4	89.5	94.6 ± 0.24	-0.1 ± 0.24	95.4	66.5

Table: Performance comparison between different pruning methods on VGG16/CIFAR100 at 98% parameter reduction and ResNet18/CIFAR10 at 95% parameter reduction

- We can clearly see that our methods outperform other pruning algorithms.
- The drop in params and flops is equivalent or more compared to other methods

Results and Analysis (Cont.)

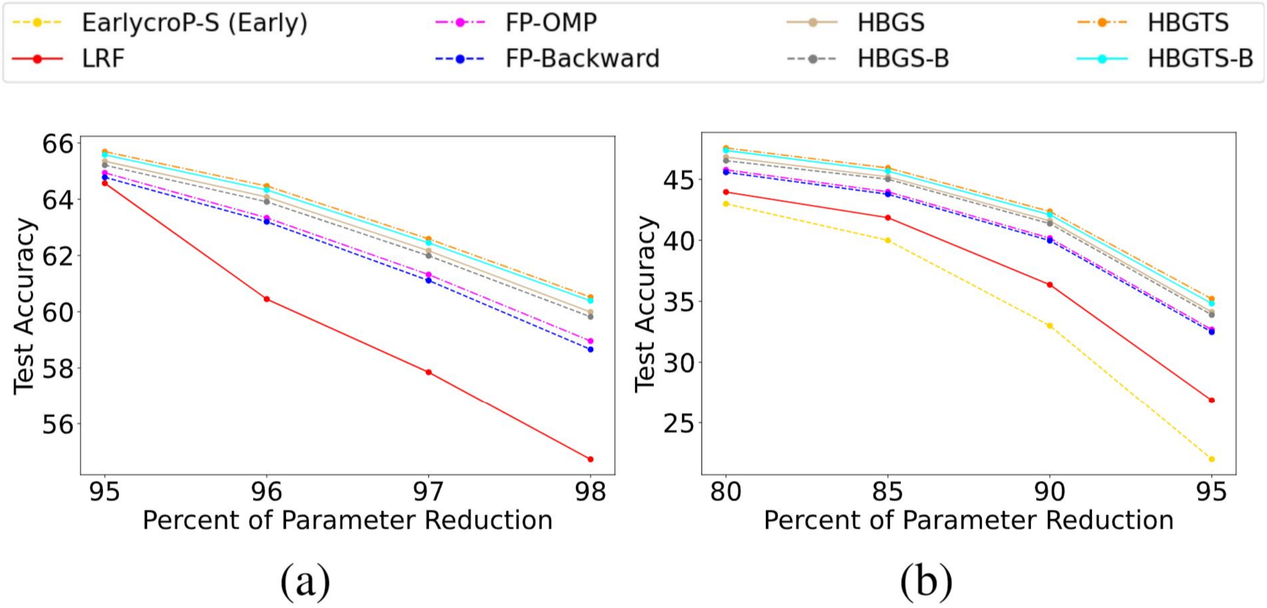


Figure: Test accuracy for (a) ResNet56/CIFAR100 and (b) ResNet18/Tiny-Imagenet with increasing parameter reduction

- We can clearly see that our methods outperform other pruning algorithms.
- As the percentage of parameter reduction increases, the difference in test accuracy between our proposed methods and state-of-the-art methods also grows.

Results and Analysis (Cont.)

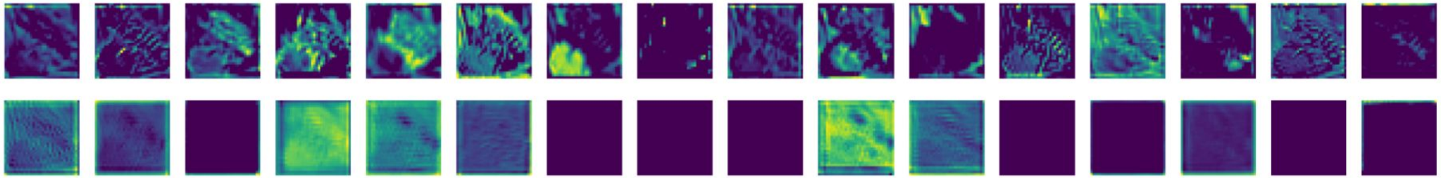


Figure: Visualisation of output feature map of ResNet32 2th layer (top row) and 10th layer (bottom row) on CIFAR100

- Feature map of Layer 2 has a diverse set of filter outputs, indicates its usefulness in capturing different features of the inputs. Our HBGST prunes only 31.25% of its filters.
- Feature map outputs from Layer 10 looks very similar, denoting its redundancy in filter outputs. 93.75% of its filters are removed by our HBGST method.

Results and Analysis (Cont.)

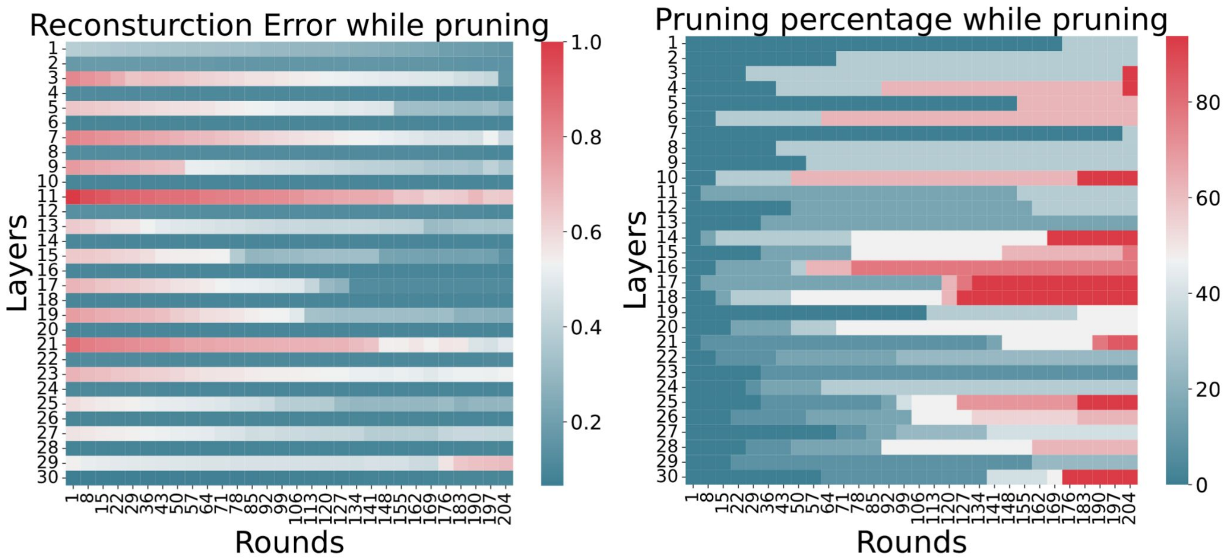


Figure: Heat map for relative reconstruction error and pruning percentage while pruning ResNet32 on CIFAR100 at 63% parameter reduction.

- Pruning percentage increases with each round, but not uniformly.
- Relative reconstruction error also decreases with pruning rounds but is not uniform across layers.
- Our method selects the layer with the least relative reconstruction error for pruning.

Conclusion

- We used a hierarchical scheme with two-levels for faster non-uniform pruning.
- Filter Pruning step identifies the most appropriate filters to be pruned from each layer.
- HBGS and HBGST algorithms selects the best layer to currently prune from.

References

1. 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.
2. 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.
3. 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
4. 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.
5. Tropp, J. A.; and Gilbert, A. C. 2007. Signal recovery from random measurements via orthogonal matching pursuit. IEEE Transactions on information theory, 53(12): 4655–4666.

THANK YOU
FOR
YOUR ATTENTION!!!

https://github.com/kiranpurohit/

@kiranpurohit08