

**FILTER PRUNING FOR COMPACT AND  
EFFICIENT CONVOLUTIONAL NEURAL  
NETWORKS**

**MILTON MONDAL**



**DEPARTMENT OF ELECTRICAL ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY DELHI**

**APRIL 2023**

**FILTER PRUNING FOR COMPACT AND  
EFFICIENT CONVOLUTIONAL NEURAL  
NETWORKS**

by

**MILTON MONDAL**

**DEPARTMENT OF ELECTRICAL ENGINEERING**

submitted

in fulfillment of the requirements of the degree of Doctor of

Philosophy

to the



**INDIAN INSTITUTE OF TECHNOLOGY DELHI**

**APRIL 2023**



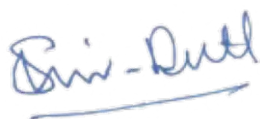
*Dedicated to  
My Teachers  
Family  
&  
Friends*

## CERTIFICATE

---

This is to certify that the thesis entitled, “**Filter Pruning for Compact and Efficient Convolutional Neural Networks**”, being submitted by **Mr. Milton Mondal**, to the Indian Institute of Technology Delhi, for the award of degree of **Doctor of Philosophy** in the Department of Electrical Engineering is a record of bonafide research work carried out by him under our supervision and guidance. He has fulfilled the requirements for the submission of the thesis, which to the best of our knowledge has reached the required standard.

The material contained in the thesis has not been submitted in part or full to any other University or Institute for the award of any degree or diploma.



**Prof. Shiv Dutt Joshi**

Date: April 2023

Department of Electrical Engineering  
Indian Institute of Technology Delhi  
New Delhi -110016



**Prof. Brejesh Lall**

Department of Electrical Engineering  
Indian Institute of Technology Delhi  
New Delhi -110016



**Dr. Pushendra Singh**

Department of Electronics & Communication Engineering  
National Institute of Technology Hamirpur  
Himachal Pradesh - 177005

## ACKNOWLEDGEMENTS

I would like to start by thanking my supervisors, Prof. Shiv Dutt Joshi, Prof. Brejesh Lall, and Dr. Pushpendra Singh, for their support, guidance, and encouragement throughout my Ph.D. journey. I am grateful to Prof. Shiv Dutt Joshi, from whom I have learned a lot of subjects with a deeper understanding beyond my research. His trust in my abilities and his encouragement to continue my research with ease, have been a source of inspiration to me. I would also like to extend my sincere gratitude to Prof. Brejesh Lall, who helped me in every possible way to conduct my experiments for advanced deep models. Also, his insightful feedback and constructive criticism have helped me to improve my research. I am obliged to Dr. Pushpendra Singh for his time and effort in guiding me, especially regarding manuscript preparation and revision. My thanks also go to all my SRC members who provided valuable feedback during my semester presentations.

This Ph.D. would not have been possible without the support of my parents, Mr. Rajendra Nath Mondal and Mrs. Anima Mondal, and all my family members. Their unwavering love, sacrifice, and emotional support have been the driving force behind my academic journey. It is a true blessing for me to have such a wonderful family. I am also grateful to my friends, including Himanshu, Bishshoy, and Ayan, for the beautiful memories we have shared. I count myself fortunate to have a friend like Himanshu, who has helped me throughout my academic journey, and with whom I have shared numerous enjoyable moments. My special thanks to Bishshoy, with whom I have discussed the most during my research, and he always supported me whenever I required any assistance.

I would like to express my deep gratitude to my yoga friends Akash, Hardik, Hemant, Sunaina, Shilpa, Mansi, and my yoga teacher, Ms. Haimanti Mukhopadhyay, for their invaluable influence in helping me to lead a healthy and happy life. I am indebted to all my spiritual masters for their teachings and for guiding me to become a better person every day. Lastly, I am eternally grateful to the entire universe for the love, support, and opportunities it has bestowed upon me to make a positive impact on the world.

New Delhi  
27 April 2023

Milton Mondal  
(2016EEZ8498)

## ABSTRACT

Convolutional neural networks (CNNs) have become the primary tool for solving most computer vision tasks. The networks have become deeper over time to extract more abstract features for improving generalization performance. Currently, workstations and GPU servers only can use these deep CNNs due to their capacity to store a large number of parameters and execute a high number of floating point operations (FLOPs). The main objective of this thesis is to design compact and efficient CNNs. The first question that comes to our mind is whether all filters present in a network are essential for a task or not, or if they are important, are their importance the same or different? To address these questions, we observe the performance of the networks after pruning filters from them. We find that with retraining the pruned model can recover the performance degradation. A lot of filter-pruning algorithms have been proposed in recent years. However, the literature lacks a clear understanding of whether we should prune a filter at a uniform rate or non-uniformly from each layer. To find the answer, we conduct several experiments while pruning the same percent filters from different layers. We observe that the performance drop in each case is different, demonstrating that different layers are sensitive to filter pruning differently. We also discover that existing methods manually set the pruning fraction per layer by observing the layer-wise sensitivity for each layer. However, this manual task of determining the pruning fraction for each layer becomes extremely tedious for very deep models. We solve this problem by designing the filter importance in such a manner that they become comparable across the layers.

We also eliminate some of the major problems of existing filter pruning methods. A filter pruning method can be either feature-dependent or feature-independent. Existing feature-dependent methods do not consider the class label of a training example when calculating filter importance using feature map activations. However, our observations indicate that a feature can have high intensity for one class but not for others, indicating that the feature is useful for that class. Our proposed method, Global Filter Importance based Adaptive Pruning (GFI-AP) resolves this issue by defining filter importance based on the strength of class-specific feature maps. GFI-AP outperforms feature-dependent methods that do not consider class-specific feature map strength. Similarly, the problem in existing feature-independent filter pruning methods is that they determine the filter index that needs to be pruned based only on that filter weight. However, when we prune

a filter from a network, the corresponding channel of all filters in the next layer is also eliminated to maintain structural consistency. Thus, unlike existing methods, our feature-independent pruning method, Filter Pruning by Successive Layers (FPSL) prunes a filter through successive layers analysis. FPSL removes a filter from a layer so that the feature maps of the subsequent layer remain close to their unpruned state. Additionally, FPSL eliminates the need for layer-by-layer retraining, manual per-layer pruning percentage selection, and intensive hyperparameter search for finetuning. In a CNN, there is always a possibility of redundancy in the number of filters used as we do not ensure that different filters should pick up different information of the input. Thus, pruning helps in eliminating those unimportant filters from the base model. Here we propose an alternate approach Multi-band CNN (M-CNN) which modifies the architecture design so that different filters capture complementary bands of the input signal. It is an attempt to prevent the generation of redundant filters in the base model. Incorporating multi-band filtering into CNN has provided a novel means of building a compact and efficient network. The proposed M-CNN maintains the model performance but reduces the number of filters that needs to be trained by a factor of four for the first or last convolutional layer. The advancements and improvements achieved by our proposed methods over existing approaches would enable the deployment of compact and efficient deep learning models in edge devices, allowing them to be more beneficial for vision and other related tasks.

## सार

कनवॉल्यूशनल न्यूरल नेटवर्क (CNN) अधिकांश कंप्यूटर दृष्टि कार्यों को हल करने के लिए प्राथमिक उपकरण बन गए हैं। बेहतर परिणाम पाने के लिए समय के साथ नेटवर्क गहरा हो गया है ताकि और अच्छा फीचर्स पाया जाये। वर्तमान में, वर्कस्टेशन और जीपीयू सर्वर केवल बड़ी संख्या में पैरामीटर्स को संग्रहीत कर पाते हैं और बड़ी संख्या में फ्लोटिंग पॉइंट ऑपरेशंस (FLOPs) को निष्पादित करने की क्षमता रखते हैं इस कारण इन गहरे सीएनएन का उपयोग कर सकते हैं। इस थीसिस का मुख्य उद्देश्य कॉम्पैक्ट और कुशल सीएनएन डिजाइन करना है। सबसे पहला प्रश्न जो हमारे दिमाग में आता है वह यह है कि क्या किसी नेटवर्क में मौजूद सभी फिल्टर किसी कार्य के लिए आवश्यक हैं या नहीं, या यदि वे महत्वपूर्ण हैं, तो क्या उनका महत्व समान है या अलग है? इन सवालों के समाधान के लिए, हम नेटवर्क से फिल्टर (filter) की छंटाई के बाद उनके प्रदर्शन का निरीक्षण करते हैं। हम पाते हैं कि छंटनी किए गए मॉडल को फिर से प्रशिक्षित करने से प्रदर्शन में गिरावट को ठीक किया जा सकता है। हाल के वर्षों में बहुत सारे फिल्टर-प्रूनिंग एल्गोरिदम प्रस्तावित किए गए हैं। हालाँकि, मौजूदा तरीके में इस बात की स्पष्ट समझ का अभाव है कि क्या हमें प्रत्येक परत से एक समान दर पर या गैर-समान रूप से फिल्टर को छंटाई करनी चाहिए। उत्तर खोजने के लिए, हम अलग-अलग परतों से समान प्रतिशत फिल्टर की छंटाई करते हुए कई प्रयोग करते हैं। हम देखते हैं कि प्रत्येक परतों में प्रदर्शन में गिरावट अलग है, यह प्रदर्शित करता है कि अलग-अलग परतें अलग-अलग तरीके से फिल्टर छंटाई के प्रति संवेदनशील हैं। हम यह भी पता लगाते हैं कि मौजूदा तरीके प्रत्येक परत के लिए परत-वार संवेदनशीलता को देखते हुए मैनुअल रूप से प्रति परत छंटाई अंश निर्धारित करते हैं। हालाँकि, प्रत्येक परत के लिए छंटाई अंश निर्धारित करने का यह मानव आधारित कार्य बहुत गहरे मॉडल के लिए अत्यंत कठिन हो जाता है। हम फिल्टर के महत्व को इस तरह से

डिजाइन करके इस समस्या को हल करते हैं कि वे परतों के बीच तुलनीय हो जाते हैं। हम मौजूदा फिल्टर प्रूनिंग विधियों की कुछ प्रमुख समस्याओं को भी समाप्त कर देते हैं। फिल्टर छंटाई विधि फीचर (feature) –निर्भर या फीचर –स्वतंत्र हो सकती है। फीचर सक्रियण का उपयोग करके फिल्टर महत्व की गणना करते समय मौजूदा फीचर–निर्भर विधियाँ प्रशिक्षण उदाहरण के वर्ग पर विचार नहीं करती हैं। हालाँकि, हमारी टिप्पणियों से संकेत मिलता है कि एक फीचर, एक वर्ग के लिए उच्च तीव्रता का हो सकती है, लेकिन दूसरों के लिए नहीं, यह दर्शाता है कि यह फीचर उस वर्ग के लिए उपयोगी है। हमारी प्रस्तावित विधि, ग्लोबल फिल्टर इंपोर्टेंस–बेस्ड एडेप्टिव प्रूनिंग (GFI–AP) वर्ग–विशिष्ट फीचर सक्रियण की ताकत के आधार पर फिल्टर महत्व को परिभाषित करके इस समस्या को हल करती है। जीएफआई–एपी फीचर–निर्भर तरीकों से बेहतर प्रदर्शन करता है जो वर्ग–विशिष्ट फीचर की ताकत पर विचार नहीं करते हैं। इसी तरह, मौजूदा फीचर–स्वतंत्र फिल्टर प्रूनिंग विधियों में समस्या यह है कि वे फिल्टर महत्व का निर्धारण करते हैं केवल उस फिल्टर के आधार पर। हालाँकि, जब हम किसी नेटवर्क से छंटाई करते हैं, तो नेटवर्क का संरचना को बनाए रखने के लिए अगली परत में सभी फिल्टर के संबंधित चैनल को भी हटा दिया जाता है। इस प्रकार, मौजूदा तरीकों से अलग, हमारी फीचर–स्वतंत्र प्रूनिंग विधि, सक्सेसिव लेयर्स द्वारा फिल्टर प्रूनिंग (FPSL) क्रमिक परतों के विश्लेषण के माध्यम से एक फिल्टर की छंटाई करती है। FPSL एक परतों से एक फिल्टर हटाता है ताकि बाद की परतों के फीचर उनकी पहले का स्थिति के करीब रहें। इसके अतिरिक्त, FPSL परत दर परत पुनर्प्रशिक्षण, मानव आधारित प्रति–परत छंटाई प्रतिशत चयन, और फ़ाइनट्यूनिंग (fine–tuning) के लिए गहन हाइपरपैरामीटर (hyperparameter) खोज की आवश्यकता को समाप्त करता है। CNN में, उपयोग किए जाने वाले फिल्टरों की संख्या में हमेशा अतिरेक की संभावना होती है क्योंकि हम यह सुनिश्चित नहीं करते हैं कि विभिन्न फिल्टर अलग–अलग जानकारी प्राप्त करें। इस प्रकार, छंटाई मॉडल से उन महत्वहीन फिल्टरों को हटाने में मदद करती है। यहां हम एक वैकल्पिक दृष्टिकोण मल्टी–बैंड सीएनएन (M–CNN) का प्रस्ताव करते हैं जो मॉडल निर्माण को संशोधित करता है ताकि विभिन्न फिल्टर इनपुट सिग्नल के पूरक बैंड को कैप्चर कर सकें। यह बेस मॉडल में निरर्थक फिल्टर के निर्माण को रोकने का एक प्रयास है। CNN में मल्टी–बैंड फिल्टरिंग को शामिल करने से एक कॉम्पैक्ट और कुशल नेटवर्क बनाने का एक नया साधन उपलब्ध हुआ है। प्रस्तावित M–CNN मॉडल के प्रदर्शन को बनाए रखता है, लेकिन पहले या अंतिम संकेंद्रित परत

के लिए चार के एक कारक द्वारा प्रशिक्षित किए जाने वाले फिल्टर की संख्या को कम करता है। मौजूदा दृष्टिकोणों पर हमारे प्रस्तावित तरीकों द्वारा हासिल की गई उन्नति और सुधार बढ़त उपकरणों में कॉम्पैक्ट और कुशल गहन मॉडल की तैनाती को सक्षम करेगा, जिससे वे दृष्टि और अन्य संबंधित कार्यों के लिए अधिक फायदेमंद हो सकेंगे।

# Contents

<b>List of Abbreviations</b>	vii
<b>List of Figures</b>	ix
<b>List of Tables</b>	xvii
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Filter Pruning . . . . .	3
1.2.1 Feature Dependent vs Independent . . . . .	3
1.2.2 Uniform vs Non-uniform . . . . .	5
1.3 Problem Definition and Scope of the Thesis . . . . .	6
1.4 Outline of the Thesis . . . . .	9
<b>2 Basic Preliminaries and Literature Survey</b>	<b>13</b>
2.1 Different types of CNN . . . . .	13
2.2 Flow of filter pruning algorithms . . . . .	17
2.3 Literature Survey . . . . .	19
<b>3 Class-Specific Feature Dependent Adaptive Filter Pruning</b>	<b>23</b>
3.1 Introduction . . . . .	23
3.2 Analysis of layer sensitivity towards pruning . . . . .	24
3.3 Importance of class specific measures toward filter pruning . . . . .	28
3.4 Global filter importance based adaptive pruning (GFI-AP) . . . . .	31
3.4.1 Proposed algorithm . . . . .	33
3.5 Results and discussion . . . . .	35
3.5.1 Adaptive Pruning vs Uniform Pruning . . . . .	35
3.5.2 Comparison with other methods . . . . .	38

3.5.3	More explorations	41
3.6	Conclusion	46
<b>4</b>	<b>Feature Independent Iterative Filter Pruning</b>	<b>47</b>
4.1	Introduction	47
4.2	Proposed Method	50
4.2.1	Neuron elimination in FCFN	51
4.2.2	Analysis without non-linearity and batch normalization for FCFN	52
4.2.3	FCFN with non-linearity and batch norm	53
4.3	Feature map elimination in CNN using FPSL	55
4.3.1	Globally comparable normalized filter importance score	56
4.4	Experiments	58
4.4.1	Experimental Setup	58
4.4.2	Results and Analysis	59
4.4.3	More Explorations	68
4.5	Conclusion	72
<b>5</b>	<b>Multi-band CNN for Efficient Utilization of Model Parameters</b>	<b>73</b>
5.1	Introduction	73
5.2	Proposed Method	74
5.3	Results	75
5.4	Conclusion	78
<b>6</b>	<b>Conclusions and Future Prospects</b>	<b>81</b>
6.1	Conclusions	81
6.2	Future Prospects	82
References		84
6.2.1	Journal Publications	94
6.2.2	Publications not included in thesis	94

## List of Abbreviations

AP	Adaptive Pruning
APN	Adaptively Pruned Network
BLAS	Basic Linear Algebra Subprograms
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
FCFN	Fully Connected Feedforward Neural network
FLOPs	Floating Point Operations
FPSL	Filter Pruning by Successive Layers
GFI-AP	Global Filter Importance based Adaptive Pruning
GPF	Global Pruning Fraction
GPU	Graphics Processing Unit
M-CNN	Multi-band Convolutional Neural Network
PCA	Principal Component Analysis
RPF	Restricted Pruning Fraction
UP	Uniform Pruning
UPN	Uniformly Pruned Network

## List of Figures

1.1	Pruning a single filter from a CNN. (a) unpruned model (b) pruned model.	
	Eliminating the first filter of $l^{th}$ layer, removes the first feature map of $l^{th}$	
	layer and reduces channel dimension of all the filters in $(l + 1)^{th}$ layer by	
	one. However the number of features remain same in $(l + 1)^{th}$ layer even	
	after pruning. . . . .	4
1.2	Different methods that help in producing compact and efficient CNNs. Our	
	proposed methods (GFI-AP, FPSL and MCNN) have been highlighted in	
	‘sky blue’ color, and the primary motivation for each of the proposed meth-	
	ods has been shown in ‘pink’ color. . . . .	7
2.1	Convolutional layers of a single-branch and multi-branch CNN, (a) single	
	branch network, (b) and (c) are two different types of bottleneck layers of a	
	multi branch network, (b) identity shortcut, (c) projection shortcut. . . . .	14
2.2	Structural changes on ResNet50 due to a single filter pruning from a con-	
	volutional layer (a) Base (Unpruned) Model, Pruned model when a filter is	
	pruned from the (b) last convolutional layer or (c) first convolutional layer or	
	(d) second convolutional layer of residual block 2. Green indicates the layer	
	on which a single filter pruning operation is being invoked. Red indicates	
	all other layers which are affected by the aforementioned pruning operation. . . . .	15
2.3	Basic building blocks of a MobileNetV2. The red dotted rectangular box	
	shows how a MobileNet replaces a conventional convolutional layer by two	
	layers. First layer has $1 \times 1$ filters with depth $n$ and second layer has $3 \times 3$	
	filters with depth 1. The first layer which contains $1 \times 1$ filters, perform	
	convolution over channels, then $3 \times 3$ filters perform spatial convolution in	
	a depth-wise separable manner. . . . .	16

2.4	Steps involved to obtain a compact model without significant performance drop due to filter pruning	17
3.1	Performance of the pruned network after pruning 99% of the total number of filters from a single layer of VGG16, trained for <b>(top)</b> CIFAR10, <b>(bottom)</b> CIFAR100. The x-label (layer index) indicates the layer from which the filters are trimmed. The test accuracy of the base model (baseline test accuracy) reduces from 94.48% to 28.87% for CIFAR10, and it drops from 73.97% to 40.69% for CIFAR100, while trimming the 1 <sup>st</sup> layer. We verified that even after retraining for forty epochs, the baseline accuracy cannot be recovered. On the contrary, we can recover almost the same classification accuracy as the unpruned network within twelve epochs of retraining if the same percentage of filters are pruned from the 13 <sup>th</sup> layer.	24
3.2	Performance of the pruned network after pruning 99% of filters from each of the two consecutive layers of VGG16, trained on <b>(top)</b> CIFAR10, <b>(bottom)</b> CIFAR100. The x-label (layer index) indicates that the starting index of two consecutive layers from which filters are deleted. Trimming filters from the first two layers reduces the test accuracy to 10.5% and after retraining it reaches 93.13% in the CIFAR10 classification task. The test accuracy remains at 94.36% after pruning filters from 10 <sup>th</sup> and 11 <sup>th</sup> layers. Interestingly, the accuracy reaches to 94.6% within twelve epochs of retraining which is better than the unpruned network's performance. The pruned network becomes less complex for CIFAR10 classification and thus provides lower generalization error than the unpruned network.	25
3.3	Performance of the pruned network after pruning 99% filters from each of three consecutive layers of VGG16, trained on <b>(top)</b> CIFAR10, <b>(bottom)</b> CIFAR100. The x-label (layer index) indicates the starting index of the three consecutive layers from which filters are deleted. In the CIFAR100 classification task, 98.97% and 89.54% parameters are retained while trimming from the first and the last three layers respectively as shown by the 'green' lines. When the deeper layers of the network are pruned, we get a smaller accuracy drop with retraining or even without retraining, than the case in which the initial layers of the network are pruned.	26

3.4  $\Phi$  scores of features, sorted in descending order, are highlighted here with (a) feature index and with (b) percentile rank of features. When  $\Phi$  is small for a feature map, it indicates that the feature map is activated with similar strength irrespective of the class of the input example. We can observe that, feature belonging to the 75<sup>th</sup> percentile in the second layer and features belonging to the 25<sup>th</sup> percentile in the seventh layer, and no features from the twelfth layer have a  $\Phi$  value less than 0.5 (CIFAR100\_VGG16 classification task). It is evident from the figure that more features have  $\Phi$  value less than a chosen threshold for the initial layers, whereas the percentage of features below the selected threshold decreases as the layer depth increases. We observe the same pattern for all datasets and all architectures that we have experimented with. . . . . 30

3.5  $\Psi$  scores of features, sorted in descending order, are highlighted here with (a) feature index and with (b) percentile rank of features. When  $\Psi$  is large for a feature map, it indicates that the feature map is activated with high strength for training examples that belong to a particular class. We can observe that, only 2 features of the second layer, 41 features of the seventh layer, and 118 features of the twelfth layer have a  $\Psi$  value greater than the threshold of 0.1 (CIFAR100\_VGG16 classification task). It is evident from the figure that more features have  $\Psi$  value greater than a chosen threshold for deep layers. It indicates that more number of features becomes more and more class specific as layer depth increases. We observe the same pattern for all datasets and all architectures that we have experimented with. . . . . 31

3.6 Illustration of GFI-AP’s pruning pipeline. Here, the  $(l - 1)^{th}$  layer’s output ( $a^{[l-1]}$ ) having  $(d_{l-1})$  channels is convolved with four filters to produce  $Z^{[l]}$ . The four filters’ class specific normalized mean is computed in  $d_l \times C$  matrix (top-left) and the filter importance is determined by the normalized mean of the maximally activated class. Filter scores for  $(l + 1)^{th}$  layer is computed in a similar fashion. Then to prune  $p\%$  filters from the entire network, the importance scores of all filters are compared globally and the least  $p\%$  are pruned. Importance score of retained and pruned filters are highlighted in green and red respectively in the Global Importance scores vector (top-right). Ticks and crosses indicate filters that are retained and pruned respectively. Blue blocks, purple blocks and purple rectangles indicate feature tensors. Green cubes indicate filters of a particular layer. . . . . 32

3.7 Performance comparison of uniformly pruned network (UPN) vs GFI-based adaptively pruned network (APN) with total pruned parameters **(a)** 78% in VGG16\_CIFAR10, **(b)** 53% in VGG16\_CIFAR100, **(c)** 41% in ResNet32\_CIFAR10. The x-label (layer index) indicates the last layer up to which the filters are trimmed. **(a)** There is no substantial variation in the performance of UPN and APN when only the first layer is trimmed in VGG16\_CIFAR10. However, APN provides  $(99.75 - 97.65) = 2.1\%$  and  $(92.8 - 89.81) = 2.99\%$  accuracy improvement over the corresponding UPN for the training and test set respectively after removing filters from all thirteen convolutional layers. **(b)** For VGG16\_CIFAR100, APN provides  $(70.9 - 69.46) = 1.44\%$  test accuracy improvement over the UPN after pruning filters from all convolutional layers. **(c)** Thirteen prunable convolutional layers are present in ResNet32. APN provides  $(96.96 - 94.08) = 2.88\%$  and  $(89.66 - 87.94) = 1.72\%$  training and test accuracy improvement respectively over the corresponding UPN method. . . . . 35

3.8	Accuracy comparisons among the base network, uniformly pruned network (UPN) and adaptively pruned network (APN) after fine tuning. APN provides $(94.23 - 92.05) = 2.18\%$ test accuracy improvement over UPN for the VGG16_CIFAR10 classification task. GFI-AP performs better than uniform pruning by $(73.68 - 71.57) = 2.11\%$ for the VGG16_CIFAR100 classification task. APN performs better in general than UPN across different architectures and datasets.	36
3.9	Retained FLOPs in each layer after applying GFI-AP. Total reduction of FLOPs is <b>(top left)</b> 22% in VGG16_CIFAR100, <b>(top right)</b> 40% in ResNet32_CIFAR10, and <b>(bottom)</b> 48% in ResNet50_ImageNet. Numeric values underneath the diagrams represent the number of FLOPs (in millions) present in each layer in the pruned network and the numeric value above the diagrams represent the fraction of FLOPs retained per layer. In residual networks, we prune all convolutional layers except the last convolutional layer of a residual block (highlighted in dark blue). <b>(top left)</b> shows that GFI-AP prunes more filters from deeper layers than initial layers in VGG16. The last two convolutional layers are pruned using a restricted pruning fraction (RPF), highlighted in 'dark pink'. It matches our previous observation that deeper layers are less sensitive towards pruning for the VGG16 architecture. However, <b>(top right)</b> shows that different amounts of filters are trimmed from different convolutional layers irrespective of the layer position when filters are trimmed adaptively in ResNet32. <b>(bottom)</b> shows that the filter importance metric is comparable across all prunable convolutional layers.	40

3.10	The importance scores of filters, sorted in descending order, is highlighted here with <b>(a)</b> filter index and with <b>(b)</b> percentile rank of filters. The importance scores of filters belonging to an initial layer, a middle layer, and a final layer is compared with the threshold provided by Global Pruning Fraction (GPF). We prune those filters which have lower importance score than the threshold provided by GPF. If a convolution layer is pruned heavily ( $>$ RPF), then we restrict the maximum number of filters that can be pruned by the threshold provided by Restricted Pruning Fraction (RPF). We limit the maximum allowable pruning fraction of a few deep layers in the VGG16 architecture. Filters belonging to the twelfth convolution layer of VGG16 architecture are pruned by the threshold provided by RPF instead of GPF.	42
3.11	Retained filters in each convolutional layer after pruning the architectures with different pruning configurations. <b>(top)</b> Total number of retained filters. <b>(bottom)</b> Percentage of retained filters in these pruning configurations - <b>(leftmost)</b> 44% FLOPs and 78% parameters reduction in CIFAR10_VGG16 (GPF = 0.6, RPF = 0.8), <b>(center left)</b> 40% FLOPs and 41% Parameters Reduction for CIFAR10_RESNET32 (GPF = 0.4), <b>(center right)</b> 22% FLOPs and 53% parameters reduction for CIFAR100_VGG16 (GPF = 0.4, RPF = 0.7), <b>(rightmost)</b> 52% FLOPs and 31% parameters reduction for IMAGENET_RESNET50 (GPF = 0.4).	43
3.12	Performance of GFI-AP on CIFAR10 classification using ResNet32 for various values of $p$ (pruning fraction). At $p=0.25$ , GFI-AP reduces 30% FLOPs without any drop in accuracy. At $p=0.55$ , the total FLOPs reduces by more than half, with just 1% drop in test accuracy. The red line shows the accuracy of the pruned model at different pruning fractions. The blue line indicates the reduction in FLOPs.	44
4.1	Neuron elimination from $l^{th}$ layer of a Fully Connected Feedforward Neural network (FCFN). It shows the structural change in $(l + 1)^{th}$ layer due to elimination of one neuron from $l^{th}$ layer.	50

4.2	Pipeline of FPSL. (i) Low $\ \mathbf{f}_{1,:}^{[l]}\ _1$ (green, i.e. first filter of $l^{th}$ layer) indicates low $\ \mathbf{z}_1^{[l]}\ _1$ , (ii) If $\ \mathbf{f}_{:,1}^{[l+1]}\ _1$ (green, i.e. first channel of all the filters of $(l+1)^{th}$ layer) is also low, then $\mathbf{f}_{1,:}^{[l]}$ should be pruned as both the first feature $\mathbf{z}_1^{[l]}$ and $\mathbf{f}_{:,1}^{[l+1]}$ has not contributed significantly to produce feature maps of $(l+1)^{th}$ layer. However if one of the $\ell_1$ norm is large then the contribution of the first filter of $l^{th}$ layer need not be necessarily low as both the $l^{th}$ (present) layer and $(l+1)^{th}$ (next) layer filter contribute to generate $\mathbf{z}^{[l+1]}$ . So FPSL takes both $\ \mathbf{f}_{j,:}^{[l]}\ _1$ and $\ \mathbf{f}_{:,j}^{[l+1]}\ _1$ to determine the $j^{th}$ filter that needs to be pruned from $l^{th}$ layer. In filter pruning, eliminating $\mathbf{f}_{1,:}^{[l]}$ imposes elimination of $\mathbf{f}_{:,1}^{[l+1]}$ to match the structural size after pruning. Here, $\mathbf{f}_j^{[l]} = \mathbf{f}_{j,:}^{[l]} \in \mathbb{R}^{1 \times n \times s \times s}$ and $\mathbf{f}_{:,j}^{[l+1]} \in \mathbb{R}^{p \times 1 \times s \times s}$ .	54
4.3	Performance of FPSL on CIFAR10 with ResNet56 at varying FLOPs reduction. It indicates that FPSL can reduce 50% FLOPs while maintaining the baseline accuracy for ResNet56_CIFAR10 configuration	60
4.4	Performance of FPSL on CIFAR10 with ResNet110 at varying FLOPs reduction. It indicates that FPSL can reduce 60% FLOPs while maintaining the baseline accuracy for ResNet110_CIFAR10 configuration	61
4.5	Top-1 accuracy vs FLOPs for different pruning methods for ResNet50 while using ImageNet dataset. FPSL provides the best test accuracy compared to state of the art (SOTA) methods for similar target FLOPs.	65
4.6	Filter importance of all the filters for ResNet50_ImageNet configuration. Horizontal green line indicates the threshold, filters with importance score lesser than the threshold is pruned. Epoch number and the reduction in total FLOPs compared to base model is highlighted on the top of each subplot.	67
4.7	Performance comparison between FPSL, FPSL_C and FPSL_S for ResNet56 and ResNet110 for CIFAR10 classification. FPSL_C indicates when only the current layer filter information is used for filter importance. similarly, FPSL_S indicates when only the succeeding layer filter information is used for filter importance. FPSL outperforms both FPSL-C and FPSL-S in all cases as it uses both the current and the succeeding layer filter information to compute filter importance.	69

4.8	Performance comparison between FPSL and FPSL NP for different datasets and architectures. FPSL NP means when FPSL applied to a base model which is Not Pretrained. Top-1 accuracy drop for both FPSL and FPSL NP are very close to each other for similar amount of FLOPs reduction for both VGG and Residual nets while working with CIFAR datasets . . . . .	71
5.1	Block diagram of a Multi-band CNN. Here, conventional convolutional layer is replaced by multi-band convolutional layer for the first layer. . . . .	74
5.2	Frequency response of a filter bank. The frequencies for which a filter has non-zero response, is highlighted in gray. If the frequency response of base filter is (a) $H_1(u, v)$ then the response of derived filters will be (b) $H_2(u, v)$ , (c) $H_3(u, v)$ , (d) $H_4(u, v)$ . . . . .	76
5.3	Performance comparison between M-CNN and CNN. MCNN-F and MCNN-L are used when the first and last (final) convolutional layer are replaced with a multi-band layer respectively. . . . .	77

## List of Tables

3.1	Hyperparameter details to perform retraining and finetuning for GFI-AP. . .	38
3.2	FLOPs comparison across different pruning methods. Arch Data denotes the architecture and the dataset used in the experiments. GFI-AP provides higher reduction of FLOPs while achieving better test accuracies than the existing methods. . . . .	38
3.3	Comparison of accuracy and FLOPs across different pruning methods for CIFAR10 classification with ResNet32. Global pruning fractions for GFI-AP are indicated by value of $p$ . All experiments have been conducted three times to get the mean and standard deviation. Best results are highlighted in bold. Acc.↓ and FLOPs↓ indicate the drop in accuracy and the reduction in FLOPs respectively for the pruned model compared to unpruned model.	39
3.4	Comparison of accuracy and FLOPs across different pruning methods for ImageNet classification with ResNet50. Global pruning fractions for GFI-AP are indicated by the value of $p$ . Best results are highlighted in bold. Acc.↓ and FLOPs↓ indicate the drop in accuracy and the reduction in FLOPs respectively for the pruned model compared to unpruned model. . . . .	39
3.5	Performance of GFI-AP on Tiny-ImageNet classification using ResNet18 for various values of $p$ (pruning fraction). All experiments have been conducted three times to generate mean and standard deviation. We observe no degradation in the performance of the pruned model even with 78% FLOPs reduction. . . . .	43

3.6	Sensitivity of GFI-AP while modifying the spatial resolution of the input image for ResNet32 CIFAR10 configuration. Size of the input image becomes (24, 24, 3) and (48, 48, 3) via downscaling and upscaling operation respectively. The performance of GFI-AP is not affected due to change in the size of input image as we observe similar accuracy drop for same percentage of FLOPs reduction in all the cases.	44
3.7	Performance of GFI-AP while freezing some of the initial layers. Freeze initial layer 'n' indicates that we have not pruned any filters from the first 'n' convolutional layers, while maintaining close to the same target reduction in FLOPs. These experiments are conducted on a system with 1x Nvidia RTX 2080 Ti.	45
3.8	Comparison between the performance of GFI-AP and GFI-AP-NC. GFI-AP-NC is the version of GFI-AP that is Not Class specific. GFI-AP-NC determines the filter importance by taking the normalized mean using all training examples irrespective of their class.	45
4.1	Pruning Results of ResNet56 on CIFAR10.	62
4.2	Pruning Results of ResNet110 on CIFAR10.	63
4.3	Pruning Results of VGG16 on CIFAR datasets.	64
4.4	Pruning Results of MobileNetV2 on ImageNet.	64
4.5	FLOPS comparison across different pruning methods for ImageNet classification with ResNet50.	68
4.6	Computation time to compute filter importance using GFI-AP (feature dependent) and FPSL (feature independent) method on ResNet56 and ResNet110 while performing CIFAR10 classification	70
5.1	Training time comparison between CNN and MCNN.	78
5.2	Performance of MCNN-F on ResNet50 for ImageNet classification	78