Adaptive CNN filter pruning using global importance metric

Supplementary Material

1. Importance of class specific measures toward filter pruning

We define two metrics Φ and Ψ to measure the differential strength of activation of a feature map across different classes. Φ determines the normalized difference between activation strengths for the class that is maximally activated and the class that is minimally activated. A characteristic of a CNN is that in the initial layers, most of the feature maps are activated for almost all training examples as low-level features are common across classes as shown in figure 1. The Ψ score of a feature map computes the normalized difference between the first and the second highest class-wise activation strengths. The number of feature maps which are activated with high intensity for a specific class increases for deeper layers as shown in figure 2. Φ score of the j^{th} feature map of the l^{th} layer is defined by,

$$\Phi_j^{[l]} = \frac{I(k_j^{[l]}) - M(k_j^{[l]})}{I(k_j^{[l]})} \tag{1}$$

Ψ score of the j^{th} feature map of the l^{th} layer is defined by,

$$\Psi_j^{[l]} = \frac{I(k_j^{[l]}) - S(k_j^{[l]})}{I(k_j^{[l]})} \tag{2}$$

where,

$$\mu_j^{c[l]} \leftarrow \frac{\sum_{i=1}^{n_c} \|\mathbf{z}_j^{c[l](i)}\|_1}{n_c W_j^{[l]} H_j^{[l]}} \tag{3}$$

$$I(k_j^{[l]}) \leftarrow \max_{c \in \{1,2,\ldots,C\}} \mu_j^{c[l]} \tag{4}$$

$$M(k_j^{[l]}) \leftarrow \min_{c \in \{1,2,\ldots,C\}} \mu_j^{c[l]} \tag{5}$$

$$S(k_j^{[l]}) \leftarrow_{c \in \{1,2,\ldots,C\}} \mu_j^{c[l]} \tag{6}$$

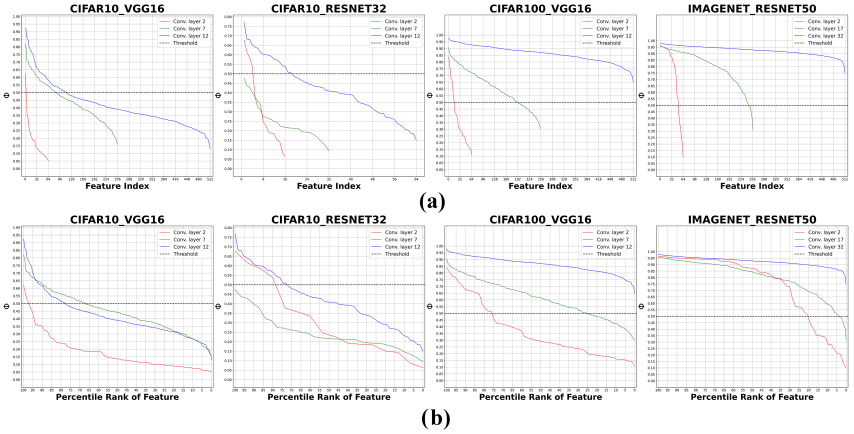

Figure 1: Φ scores of features, sorted in descending order, are highlighted here with **(a)** feature index and with **(b)** percentile rank of features. When Φ is small for a feature map, it indicates that the feature map is activated with similar strength irrespective of the class of the input example. We can observe that, feature belonging to the 75^{th} percentile in the second layer and features belonging to the 25^{th} percentile in the seventh layer, and no features from the twelfth layer have a Φ value less than 0.5 (CIFAR100_VGG16 classification task). It is evident from the figure that more features have Φ 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.

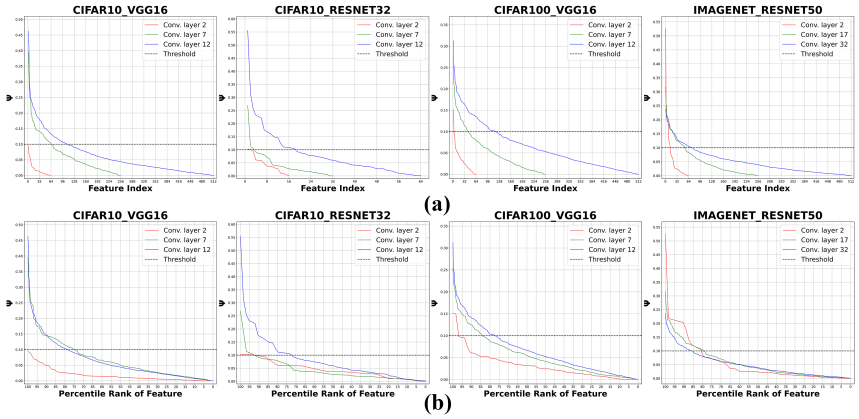

Figure 2: Ψ scores of features, sorted in descending order, are highlighted here with **(a)** feature index and with **(b)** percentile rank of features. When Ψ is large for a feature map, it indicates that the feature map is activated with high strength for training examples that belong to a particular 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 Ψ value greater than the threshold of 0.1 (CIFAR100_VGG16 classification task). It is evident from the figure that more features have Ψ 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.

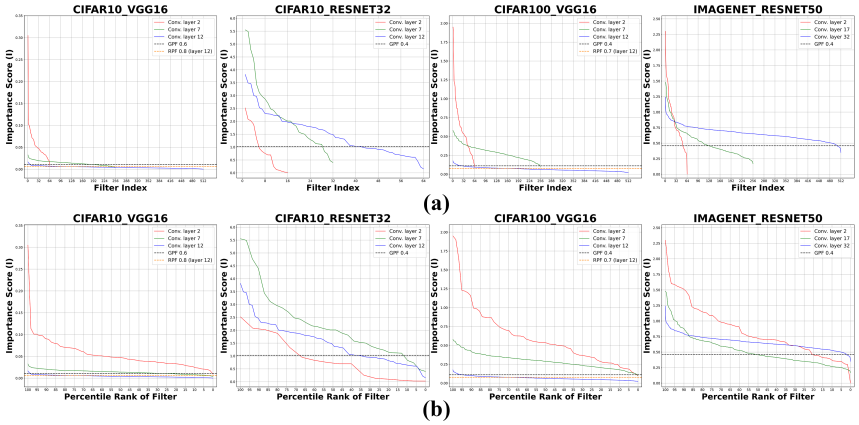

Figure 3: The importance scores of filters, sorted in descending order, is highlighted here with **(a)** filter index and with **(b)** percentile rank of filters. The importance scores of filters belonging to an initial layer, a middle layer, and a final layer is compared with the threshold provided 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.

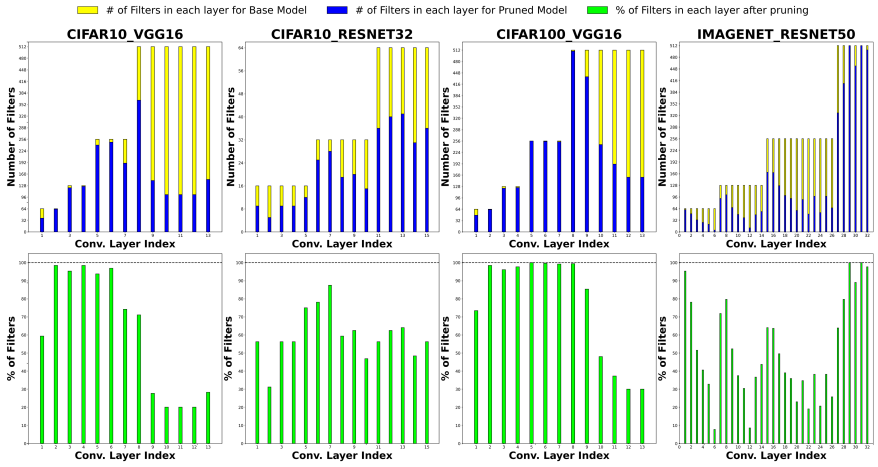

Figure 4: Retained filters in each convolutional layer after pruning the architectures with different pruning configurations. **(top)** Total number of retained filters. **(bottom)** Percentage of retained filters in these pruning configurations - **(leftmost)** 44% FLOPs and 78% parameters reduction in CIFAR10_VGG16 (GPF = 0.6, RPF = 0.8), **(center left)** 40% FLOPS and 41% Parameters Reduction for CIFAR10_RESNET32 (GPF = 0.4), **(center right)** 22% FLOPS and 53% parameters reduction for CIFAR100_VGG16 (GPF = 0.4, RPF = 0.7), **(rightmost)** 52% FLOPs and 31% parameters reduction for IMAGENET_RESNET50 (GPF = 0.4)

2. Layer sensitivity towards pruning

The following indicates how the performance of the CNN changes while pruning filters from different convolutional layers. To observe the sensitivity of layers in filter pruning we perform experiments on VGG16. The details for the experiments are discussed in section 3. Figures 2-4 show the test accuracies of the pruned network before and after retraining. Figure 2 depicts a case of pruning a single-layer from the VGG16 network, and figure 3 and figure 4 depicts the pruning of two and three layers respectively. The table data of the corresponding plots are given below.

Table 1: Performance of the pruned network after pruning 99% of the total number of filters from a **single** layer of **VGG16**, trained for **CIFAR10**. Each experiment was run three times to calculate the mean and standard deviation.

Layer	Accuracy before pruning		Accuracy after pruning		Retained
	mean	std	mean	std	parameters (%)
0.0	28.87	0.0	93.53	0.138	99.87
1.0	71.64	0.0	93.62	0.15	99.64
2.0	68.84	0.0	93.703	0.183	99.27
3.0	73.62	0.0	93.386	0.033	98.55
4.0	86.18	0.0	93.74	0.138	97.1
5.0	89.9	0.0	93.65	0.155	96.13
6.0	91.09	0.0	93.98	0.032	94.2
7.0	94.05	0.0	94.27	0.067	88.39
8.0	94.51	0.0	94.533	0.0368	84.53
9.0	94.53	0.0	94.593	0.032	84.53
10.0	94.48	0.0	94.543	0.0573	84.53
11.0	94.46	0.0	94.55	0.054	84.53
12.0	94.62	0.0	94.59	0.0286	91.4

Table 2: Performance of the pruned network after pruning 99% of the total number of filters from **two** layers of **VGG16**, trained for **CIFAR10**. Each experiment was run three times to calculate the mean and standard deviation.

Layer	Accuracy before pruning		Accuracy after pruning		Retained
	mean	std	mean	std	parameters (%)
0.0	10.5	0.0	93.13	0.265	99.57
1.0	21.6	0.0	93.29	0.106	99.03
2.0	25.38	0.0	93.12	0.094	98.06
3.0	57.51	0.0	92.926	0.0492	96.13
4.0	78.98	0.0	93.193	0.168	94.19
5.0	76.72	0.0	93.46	0.261	91.29
6.0	88.85	0.0	93.756	0.095	84.53
7.0	93.95	0.0	94.323	0.0449	76.79
8.0	94.49	0.0	94.573	0.04027	72.92
9.0	94.26	0.0	94.56	0.069	72.92
10.0	94.35	0.0	94.553	0.009	72.92
11.0	94.54	0.0	94.616	0.0286	79.8
12.0	94.46	0.0	94.583	0.0492	90.11

Table 3: Performance of the pruned network after pruning 99% of the total number of filters from **three** layers of **VGG16**, trained for **CIFAR10**. Each experiment was run three times to calculate the mean and standard deviation.

Layer	Accuracy before finetuning		Accuracy after finetuning		Retained
	mean	std	mean	std	parameters (%)
0	12.05	0.0	92.99333	0.028674	98.96
1	14.57	0.0	92.85	0.163095	97.82
2	16.53	0.0	92.91333	0.151511	95.64
3	45.67	0.0	92.92667	0.041899	93.23
4	60.1	0.0	93.22	0.117757	89.36
5	71.88	0.0	93.53333	0.129701	81.62
6	88.02	0.0	93.83667	0.156276	72.92
7	93.49	0.0	94.33333	0.038586	65.18
8	93.31	0.0	94.60333	0.020548	61.32
9	94.04	0.0	94.59667	0.016997	61.32
10	93.98	0.0	94.65333	0.060185	68.19
11	94.45	0.0	94.61333	0.020548	78.51
12	94.38	0.0	94.66667	0.018856	89.66

Table 4: Performance of the pruned network after pruning 99% of the total number of filters from a **single** layer of **VGG16**, trained for **CIFAR100**. Each experiment was run three times to calculate the mean and standard deviation.

Layer	Accuracy before pruning		Accuracy after pruning		Retained
	mean	std	mean	std	parameters (%)
0.0	40.69	0.0	73.223	0.217	99.87
1.0	29.85	0.0	72.62	0.163	99.64
2.0	27.51	0.0	73.006	0.197	99.28
3.0	36.92	0.0	72.283	0.132	98.55
4.0	37.91	0.0	72.3	0.259	97.11
5.0	51.12	0.0	72.563	0.287	96.14
6.0	52.93	0.0	72.396	0.109	94.21
7.0	60.72	0.0	72.916	0.246	88.43
8.0	67.11	0.0	73.0	0.134	84.57
9.0	72.04	0.0	73.263	0.196	84.57
10.0	72.15	0.0	73.3	0.107	84.57
11.0	73.14	0.0	73.28	0.123	84.57
12.0	73.14	0.0	73.593	0.07	91.43

Table 5: Performance of the pruned network after pruning 99% of the total number of filters from **two** layers of **VGG16**, trained for **CIFAR100**. Each experiment was run three times to calculate the mean and standard deviation.

Layer	Accuracy before pruning		Accuracy after pruning		Retained
	mean	std	mean	std	parameters (%)
0.0	5.84	0.0	72.686	0.216	99.57
1.0	4.98	0.0	72.31	0.064	99.03
2.0	5.55	0.0	71.953	0.259	98.07
3.0	11.92	0.0	71.57	0.073	96.14
4.0	21.82	0.0	71.9	0.145	94.21
5.0	22.49	0.0	71.473	0.219	91.32
6.0	37.77	0.0	72.06	0.164	84.57
7.0	50.59	0.0	72.41	0.142	76.86
8.0	61.75	0.0	72.836	0.247	73.0
9.0	70.46	0.0	73.23	0.155	73.0
10.0	71.4	0.0	73.456	0.088	73.0
11.0	72.02	0.0	73.733	0.14	79.86
12.0	72.43	0.0	73.933	0.193	90.14

Table 6: Performance of the pruned network after pruning 99% of the total number of filters from **three** layers of **VGG16**, trained for **CIFAR100**. Each experiment was run three times to calculate the mean and standard deviation.

Layer	Accuracy before pruning		Accuracy after pruning		Retained
	mean	std	mean	std	parameters (%)
0.0	2.22	0.0	72.323	0.166	98.97
1.0	1.92	0.0	70.89	0.302	97.83
2.0	2.58	0.0	70.55	0.163	95.66
3.0	6.14	0.0	70.246	0.155	93.25
4.0	6.26	0.0	69.5	0.254	89.39
5.0	11.27	0.0	70.063	0.147	81.68
6.0	27.0	0.0	70.81	0.131	73.0
7.0	41.93	0.0	71.983	0.024	65.29
8.0	59.94	0.0	72.966	0.061	61.43
9.0	69.53	0.0	73.606	0.15	61.43
10.0	69.95	0.0	74.113	0.087	68.29
11.0	71.41	0.0	74.333	0.054	78.57
12.0	72.09	0.0	74.35	0.131	89.54

3. Feature size after pruning

Table 7: Size of the feature map in each convolutional layer after pruning filters from VGG16 while performing CIFAR100 classification. Table shows the size (height, width, channel) of the feature maps for the case of 22% FLOPs reduction from VGG16 using GFI-AP.

Layer	Feature Size	
	Base model	Pruned model
0	(32, 32, 3)	(32, 32, 3)
1	(32, 32, 64)	(32, 32, 47)
2	(32, 32, 64)	(32, 32, 46)
3	(16, 16, 128)	(16, 16, 122)
4	(16, 16, 128)	(16, 16, 120)
5	(8, 8, 256)	(8, 8, 251)
6	(8, 8, 256)	(8, 8, 256)
7	(8, 8, 256)	(8, 8, 253)
8	(4, 4, 512)	(4, 4, 507)
9	(4, 4, 512)	(4, 4, 435)
10	(4, 4, 512)	(4, 4, 210)
11	(2, 2, 512)	(2, 2, 92)
12	(2, 2, 512)	(2, 2, 56)
13	(2, 2, 512)	(2, 2, 46)

Here, we observe the effects of pruning on the size of the feature map for the cases VGG16_CIFAR100 and ResNet50_ImageNet in table 7 and in table 8 respectively. As we can observe from both the tables that GFI-AP only reduces the number of channels of feature map, it does not change the spatial size of feature maps. Here, size of the feature map for layer 0 indicates the size of a single input image. For VGG16, GFI-AP reduces the number of channels for different layer by different amount as shown in table 7. In ResNet50, each residual block consists of three convolutional layers and one shortcut path. GFI-AP prunes filters from every layer of each residual block except the last layer of a block. So the size of the feature maps remains same even after pruning for the last convolutional layer of each residual block as shown in table 8 as we do not prune filters from the last convolutional layer of residual block.

Table 8: Size of the feature map in each convolutional layer after pruning for ResNet50_ImageNet case. Here, Block.Layer indicates the index of the residual block and the layer in which a feature map belongs to. Table shows the size (height, width, channel) of the feature maps for the case of 48% FLOPs reduction from ResNet50 using GFI-AP.

Block.Layer	Feature Size	
	Base Model	Pruned Model
0	(224, 224, 3)	(224, 224, 3)
1	(112, 112, 64)	(112, 112, 64)
2.1	(56, 56, 64)	(56, 56, 62)
2.2	(56, 56, 64)	(56, 56, 53)
2.3	(56, 56, 256)	(56, 56, 256)
3.1	(56, 56, 64)	(56, 56, 36)
3.2	(56, 56, 64)	(56, 56, 29)
3.3	(56, 56, 256)	(56, 56, 256)
4.1	(56, 56, 64)	(56, 56, 24)
4.2	(56, 56, 64)	(56, 56, 5)
4.3	(56, 56, 256)	(56, 56, 256)
5.1	(28, 28, 128)	(28, 28, 96)
5.2	(28, 28, 128)	(28, 28, 106)
5.3	(28, 28, 512)	(28, 28, 512)
6.1	(28, 28, 128)	(28, 28, 70)
6.2	(28, 28, 128)	(28, 28, 59)
6.3	(28, 28, 512)	(28, 28, 512)
7.1	(28, 28, 128)	(28, 28, 46)
7.2	(28, 28, 128)	(28, 28, 13)
7.3	(28, 28, 512)	(28, 28, 512)
8.1	(28, 28, 128)	(28, 28, 56)
8.2	(28, 28, 128)	(28, 28, 62)
8.3	(28, 28, 512)	(28, 28, 512)
9.1	(14, 14, 256)	(14, 14, 173)
9.2	(14, 14, 256)	(14, 14, 171)
9.3	(14, 14, 1024)	(14, 14, 1024)
10.1	(14, 14, 256)	(14, 14, 141)
10.2	(14, 14, 256)	(14, 14, 109)
10.3	(14, 14, 1024)	(14, 14, 1024)
11.1	(14, 14, 256)	(14, 14, 104)
11.2	(14, 14, 256)	(14, 14, 71)
11.3	(14, 14, 1024)	(14, 14, 1024)
12.1	(14, 14, 256)	(14, 14, 101)
12.2	(14, 14, 256)	(14, 14, 69)
12.3	(14, 14, 1024)	(14, 14, 1024)
13.1	(14, 14, 256)	(14, 14, 114)
13.2	(14, 14, 256)	(14, 14, 70)
13.3	(14, 14, 1024)	(14, 14, 1024)
14.1	(14, 14, 256)	(14, 14, 115)
14.2	(14, 14, 256)	(14, 14, 84)
14.3	(14, 14, 1024)	(14, 14, 1024)
15.1	(7 , 7 , 512)	(7 , 7 , 361)
15.2	(7 , 7 , 512)	(7 , 7 , 429)
15.3	(7 , 7 , 2048)	(7 , 7 , 2048)
16.1	(7 , 7 , 512)	(7 , 7 , 512)
16.2	(7 , 7 , 512)	(7 , 7 , 471)
16.3	(7 , 7 , 2048)	(7 , 7 , 2048)
17.1	(7 , 7 , 512)	(7 , 7 , 512)
17.2	(7 , 7 , 512)	(7 , 7 , 509)
17.3	(7 , 7 , 2048)	(7 , 7 , 2048)

4. Experiments on Tiny-ImageNet

We apply our GFI-AP method on TinyImageNet dataset using ResNet18 architecture. TinyImageNet is a small version of ImageNet dataset containing 200 clases. We modify the ResNet18 model for TinyImageNet classification as described in [1]. We find that overfitting happens while training with ResNet18 model, so we achieve 99.95% train accuracy but 37.26% test accuracy for this classification task for unpruned model. We observe that the test accuracy does not reduce even with 78% FLOPs reduction. So, GFI-AP provides a compact pruned model when severe overfitting happens while training the base model.

Table 9: 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.

Pruning fraction (pf)	Baseline accuracy (%)	Pruned accuracy (%)	FLOPs (M)	FLOPs reduction (%)
0.8	37.26	38.99(±0.52)	64.79	56.58
0.85	37.26	38.54(±0.24)	56.09	62.41
0.9	37.26	38.31(±0.35)	46	69.17
0.95	37.26	37.98(±0.23)	32.53	78.2

References

[1] C. M. J. Tan, M. Motani, Dropnet: Reducing neural network complexity via iterative pruning, in: International Conference on Machine Learning, PMLR, 2020, pp. 9356–9366.