SkipConv: Skip Convolution for Computationally Efficient Deep CNNs

Pravendra Singh
Indian Institute of Technology Kanpur
psingh@iitk.ac.in

Vinay P. Namboodiri
Indian Institute of Technology Kanpur
vinaypn@iitk.ac.in

***Abstract*—Convolution operation in deep convolutional neural networks is the most computationally expensive as compared to other operations. Most of the model computation (FLOPS) in the deep architecture belong to convolution operation. In this paper, we are proposing a novel skip convolution operation that employs significantly fewer computation as compared to the traditional one without sacrificing model accuracy. Skip convolution operation produces structured sparsity in the output feature maps without requiring sparsity in the model parameters for computation reduction. The existing convolution operation performs the redundant computation for object feature representation while the proposed convolution skips redundant computation. Our empirical evaluation for various deep models (VGG, ResNet, MobileNet, and Faster R-CNN) over various benchmarked datasets (CIFAR-10, CIFAR-100, ImageNet, and MS-COCO) show that skip convolution reduces the computation significantly while preserving feature representational capacity. The proposed approach is model-agnostic and can be applied over any architecture. The proposed approach does not require a pretrained model and does train from scratch. Hence we achieve significant computation reduction at training and test time. We are also able to reduce computation in an already compact model such as MobileNet using skip convolution. We also show empirically that the proposed convolution works well for other tasks such as object detection. Therefore, SkipConv can be a widely usable and efficient way of reducing computation in deep CNN models.**

***Index Terms*—Computation (FLOPS) compression, Convolutional neural network, Image recognition, Deep learning, Computationally efficient CNN**

I. Introduction

Recent advances in deep learning have led to impressive and significant breakthrough in various domains, such as computer vision, NLP, information retrieval tasks and generative model etc. Pushing the performance further typically leads to models with overly complex and deeper architecture that tends to increase the model size (number of parameters, depth, and breadth of layers) and FLOPS considerably. Therefore, training or test time computational requirements for such complex models are significantly increased.

The main reason for the success of deep learning is its automatic representation learning capability over traditionally handcrafted [1]–[3] representation learning approaches. These deep learning models consist of a set of convolutional layers, and each layer contains a set of convolutional filters. Each convolutional filter learns some kind of representation from the input. These high-end deep learning models are highly

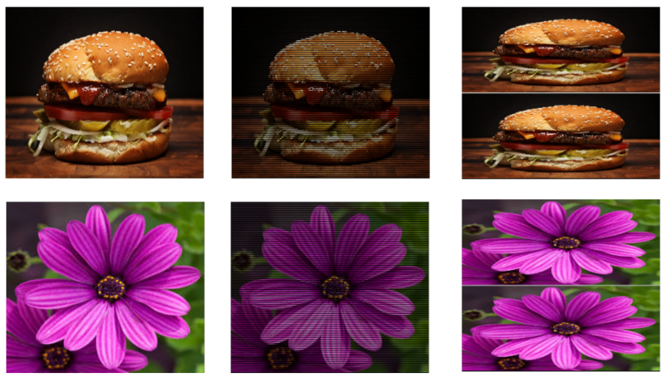

Fig. 1. The left figure shows the original images, middle figure shows the images after making alternative rows to zero, and right figure shows the images after removing even/odd rows from the left side images (best viewed in color).

redundant in terms of computation due to the presence of spatial redundancy. One way to reduce the computation of the deep CNNs is to take advantage of spatial redundancy in the feature maps and bypass redundant computation.

As shown in figure 1 left images, most of the time, each pixel is surrounded by very similar pixels in the neighborhood. Therefore each pixel and its nearby pixels contain similar information except when the pixels lie on edges. Hence, if we apply the convolution operation by skipping the nearby pixels (horizontally or vertically), then also we can capture the necessary information, as shown in figure 1. This skip convolution operation produces structured sparsity in the output feature maps. This approach is also motivated by the Nyquist sampling theorem that states that a signal, when sampled at twice the highest frequency rate, can be reconstructed without loss of information. For an image, the highest frequency would be attained if one had a checkerboard pattern with alternate values, which is a special case. For this pattern, one would have to sample all the pixels to meet the Nyquist criterion. Other than this particular case, most images have continuous signals, and sampling alternate rows or columns satisfy this criterion. Furthermore, a significant amount of computation can be reduced by skipping convolution operations, which results in a significant reduction in training/test time. We can also use structured sparsity in the output feature maps for computation reduction by internally skipping corresponding rows of kernels of the convolutional filter. Note that, these skip convolutions are performed over the feature maps. Hence, the proposed approach is independent of deep CNNs architecture and can be employed on any deep CNNs for computation reduction.

The edges in the image are the most informative part. If all the convolutions by all the convolutional filters skip the edge pixels, then we may lose some information. Therefore, the model results in inadequate representation learning and model performance degrade significantly. However, the chance of this situation to happen is rare because each layer contains many convolutional filters. To avoid this rare situation, we are using a combination of the dense and skip convolutions at every layer. A subset of filters is selected for the dense convolutions which produce dense output feature maps. These filters are also able to capture the useful information from the edge pixels because of its dense nature. Remaining filters perform skip convolutions to produce structured sparsity in the output feature maps. Skip convolutions skip the rows with some predefined gap to bypass redundant computation. The major contributions of this work are as follows:

- We propose a model-agnostic SkipConv for reducing computation in deep CNNs.
- The proposed SkipConv can be directly plugged into any deep CNNs architecture to reduce model computation.
- Efficient deep CNNs models are obtained by training from scratch and does not require a pretrained model.
- We can significantly reduce the training as well as test time computation using the proposed SkipConv.
- We demonstrate the usability of our proposed SkipConv for various vision tasks such as object classification and object detection. Our approach outperforms the compared state-of-the-art FLOPS (computation) compression methods [4]–[17].

II. Related Work

The recent development of the deep neural network [18]–[21] achieves state-of-the-art results over the various task. The high performance comes with the cost of high computational requirements, which restrains the use of the deep model for resource-constrained devices. Therefore various approaches [22]–[32] have been proposed to reduce the computation of deep convolutional neural networks. The works on model compression can be divided into the following categories:

A. Efficient Convolutional Filter

Groupwise Convolution (GWC) [18], Depthwise convolution (DWC) [33] and Pointwise Convolution (PWC) [20] are the popular convolutional filters used in compact architectures to reduce the computation. MobileNet [33] uses DWC and PWC to reduce FLOPS. ResNet [20] uses a bottleneck structure to reduce computation.

B. Model Compression

Another popular approach for computational reduction is model compression. These methods can further categorised as: 1- Connection Pruning [34], 2- Filter Pruning [12], [15], [17], [35]–[37] and 3- Bits Compression [38]. Filter pruning approaches are more effective as compared to other approaches because they prune the whole convolutional filter from the model. Also, filter pruning approaches do not require any special hardware/software support to achieve a reduction in computation.

Various filter pruning approach calculates the importance of the filter and prunes them based on some criteria followed by re-training to recover the accuracy drop in the CNN model. [35] uses a l_1 norm to get filter importance for model compression. [10] uses a scaling factor to scale the outputs of specific structures in the CNN model and perform filter pruning by making some of the factors to zero. SPP [11] uses a probabilistic based approach to compresses the deep CNN. CP [36] proposes an iterative two-step procedure to prune convolutional filters in the deep CNN. ThiNet [9] discard convolutional filters based on next layer statistics. VCNNP [7] proposes a variational bayesian scheme for compressing the CNN model. SFP [12] proposes a soft filter pruning approach to update pruned filters during training. GDP [13] proposes a global and dynamic model compression technique. GAL [6] proposes a structured pruning approach using a generative adversarial learning idea. DBSR [8] uses block sparsity for model compression. WAE [14] proposes a Wavelet-like auto-encoder that decomposes the original input image into low-frequency information and high-frequency to accelerate deep CNN. NISP [15] uses feature ranking techniques to rank the importance of each neuron in the final response layer. AMC [16] proposes a reinforcement learning based AutoML method for pruning.

Most of the methods in model compression require a pretrained model and then perform iterative pruning and fine-tuning to get a compressed model, which is a time-consuming process. The proposed work to get a computationally efficient model is more flexible in the sense that it does not require a pretrained model and reduces training as well as test time computation.

The above-discussed work increases the model efficiency, either using new convolutional filters or by discarding the redundant filters. In the proposed approach, we reduce the redundant computation by exploiting structured sparsity in feature maps using the skip convolution operation. Our approach is model-agnostic and can be applied over any CNN architecture to increase the model efficiency (speed) further. Other model compression methods are complementary to the proposed approach and can be used in conjunction with SkipConv to get a further reduction in computation. We show empirically that SkipConv can reduce the significant FLOPS (computation) even in an already compact model (MobileNet) without degrading the model performance.

III. SkipConv: Skip Convolution

Let us assume input feature maps of size $D_i \times D_i \times M$. Where D_i is the spatial width and height of the input square feature map, and M is the input depth (number of channels

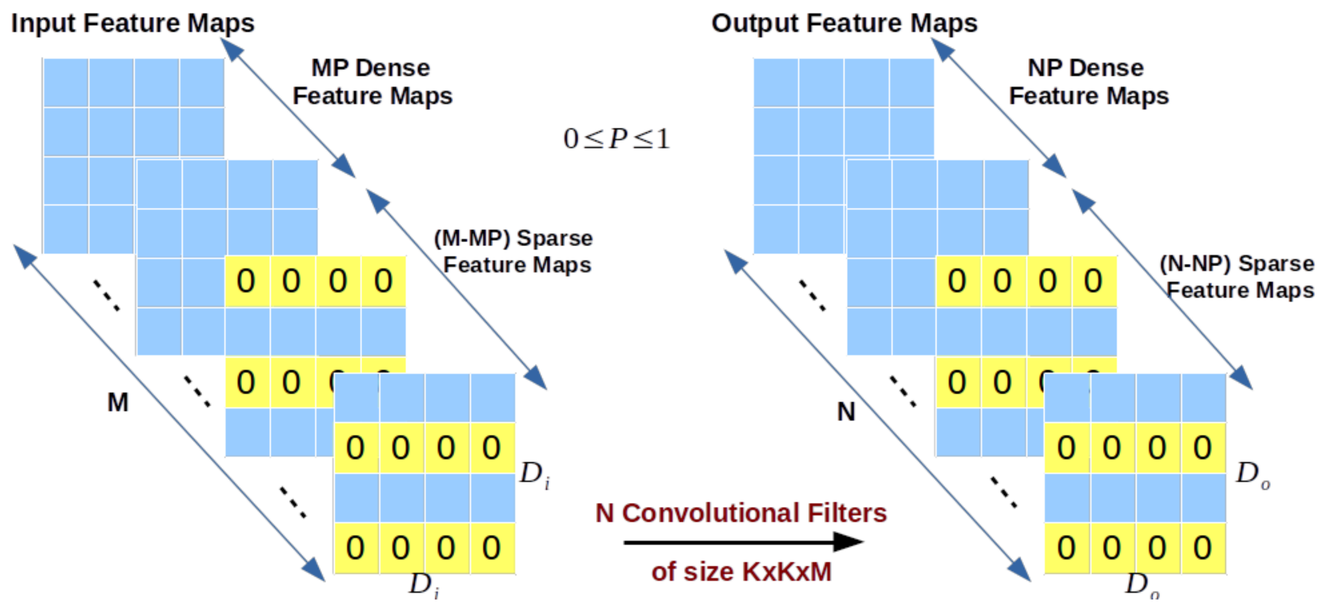

Fig. 2. The figure shows the input and output feature maps. The blue color is used to represent nonzero values, and yellow color is used to represent zero values in the feature maps (best viewed in color).

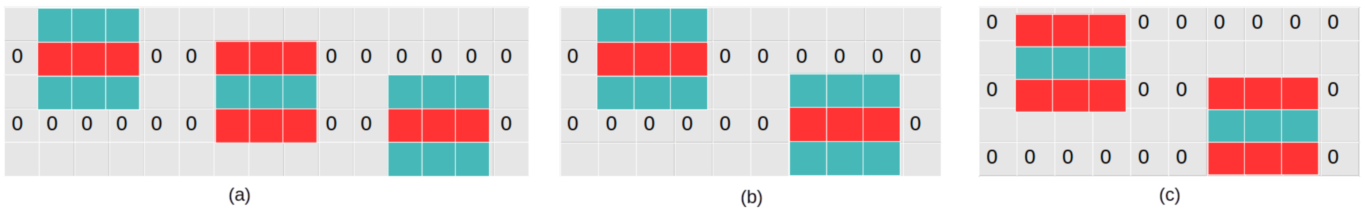

Fig. 3. The figure shows the convolution of 3×3 kernel over a sparse feature map in the input under various positions. The red color row of 3×3 kernel will always be skipped during convolution because corresponding values in the feature map are zero (best viewed in color).

in input feature maps). An output feature maps of size $D_o \times D_o \times N$ is obtained by applying N convolutional filters of size $K \times K \times M$. Where K is the kernel size, D_o is the spatial width and height of the output square feature map, and N is the output depth (number channels in the output feature maps). Note that P is a parameter which controls the number of dense feature maps and $P \in [0, 1]$.

Out of N, NP convolutional filters will produce NP dense feature maps in output as shown in figure 2. Therefore NP convolutional filters will have $X_{stride} = 1$ and $Y_{stride} = 1$. The remaining $(N - NP)$ convolutional filters will produce $(N-NP)$ sparse feature maps in the output. The $(N-NP)/2$ convolutional filters will produce even sparse feature maps (even rows are zero), and the remaining $(N - NP)/2$ convolutional filters will produce odd sparse feature maps (odd rows are zero) as shown in figure 2. The sparse feature map can be produced by using $X_{stride} = 1$, $Y_{stride} = 2$ (skipping rows during convolution) and then zero padding at missing rows position to match spatial size of $D_o \times D_o$. Therefore, there will be NP dense feature maps and $(N - NP)$ sparse feature maps in the output. Similarly, there will be MP dense feature maps and $(M - MP)$ sparse feature maps in the input. Note that the sparsity pattern in the sparse feature map is alternative (alternative zero rows).

Next, we will define the convolution operation on this type of feature maps. Let us assume the convolutional filter of size $K \times K \times M$ with M kernels of size $K \times K$. The convolution of $K \times K$ kernel over dense feature map in the input is the same as traditional convolution. The convolution of $K \times K$ kernel over a sparse feature map in the input can be categorized into two cases.

First Case - Output feature map produced by convolutional filter is dense ($X_{stride} = 1$, $Y_{stride} = 1$): The convolution of $K \times K$ kernel over sparse feature map in the input for different positions is shown in figure 3 (a). First 3×3 kernel position in figure 3 (a) shows that row-2 of 3×3 kernel (red color) can be skipped during convolution. Second 3×3 kernel position in figure 3 (a) shows that row-1 and row-3 of 3×3 kernel (red color) can be skipped during convolution. Therefore in this case, for the same 3×3 kernel, either row-2 or (row-1, row-3) will be skipped depending on the position of the kernel during convolution. The figure 3 (a) illustrates this case in more detail.

Second Case - Output feature map produced by convolutional filter is sparse ($X_{stride} = 1$, $Y_{stride} = 2$): The convolution of $K \times K$ kernel over sparse feature map in the input for different positions is shown in figure 3 (b) and (c).

Figure 3 (b) shows that row-2 of 3×3 kernel (red color) will always be skipped during convolution. Similarly, figure 3 (c) shows that row-1 and row-3 of 3×3 kernel (red color) will always be skipped during convolution. There will be exactly $(M - MP)/2$ feature maps with even zero rows and $(M-MP)/2$ feature maps with odd zero rows in input feature maps.

We are creating a structured sparsity in the output feature maps to reduce the computation in the deep CNNs. In the proposed approach, we are not introducing any sparsity in the model parameters. Sparse convolution can be used in conjunction with the proposed approach to get further computational reductions by introducing sparsity in the model parameters during training.

A. Comparision with Strided Convolution

The strided convolution (stride > 1) is used to downsample the feature maps. We have used strided convolution ($X_{stride} = 1$, $Y_{stride} = 2$) and then zero row padding at missing rows position to match spatial size of feature map $D_0 \times D_0$ to produce sparse feature maps, as shown in figure 2. This way, instead of downsampling, we are introducing structured sparsity (alternate zero rows) in the feature map. We are also skipping some rows of kernels of a convolutional filter to take benefit from a sparse feature map during convolution as shown in figure 3, which makes it different from strided convolution.

IV. Computational Complexity Analysis

Let us assume input feature maps of size $D_i \times D_i \times M$, where D_i is the spatial width and height of the input feature map, and M is the input depth (number of channels in input feature maps). An output feature map can be obtained by applying N convolutional filters of size $K \times K \times M$, where K is the kernel size. Therefore, output feature maps are of the size $D_o \times D_o \times N$, where D_o is the spatial width and height of output feature map, and N is the output depth (number of channels in output feature maps). Therefore, the total computational cost (FLOPS) at this layer can be given as:

$$O_F = D_o D_o KKMN \tag{1}$$

It is clear from equation-1 that the computational cost depends on the kernel size (K), feature map size, input channels M, and output channels N. This computational cost is quite high and can be reduced by carefully designing the skip convolution operation.

Let us define P ($P \in [0,1]$), which is a parameter to control the number of dense feature maps. Out of total N convolutional filters, NP convolutional filters will produce NP dense feature maps in output as shown in figure 2. There are MP dense feature maps and $(M - MP)$ sparse feature maps in the input. Note that the sparsity pattern in the sparse feature map is alternative zero rows. The computational cost for NP dense convolutional filters over MP dense input feature-maps is:

$$D_o D_o KK(MP)(NP)$$

and the computational cost over $(M - MP)$ sparse input feature maps is:

$$D_o D_o KK((M - MP)/2)(NP)$$

Note that it is divided by two because $(M-MP)$ input feature maps are 50% sparse (alternative zero rows). Therefore, half rows of kernels of convolutional filters are skipped during convolution (figure 3 (a)).

The total computational cost to produce NP dense output feature maps can be given as:

$$\begin{aligned} D_F =& [(D_o D_o KK(MP)) \\ &+ (D_o D_o KK((M - MP)/2))](NP) \\ =& \frac{1}{2}[D_o D_o KKM(1+P)(NP)] \end{aligned} \tag{2}$$

The remaining $(N-NP)$ convolutional filters will produce $(N - NP)$ sparse feature maps in the output. The sparse feature map can be produced by using $X_{stride} = 1$, $Y_{stride} = 2$ (skipping rows during convolution) and then zero padding at missing rows position to match spatial size of $D_o \times D_o$.

Therefore, total computational cost to produce $(N - NP)$ sparse output feature maps can be given as:

$$\begin{aligned} S_F =& \frac{1}{2}[(D_o D_o KK(MP)) \\ &+ (D_o D_o KK((M - MP)/2))](N - NP) \\ =& \frac{1}{4}[D_o D_o KKM(1+P)(N-NP)] \end{aligned} \tag{3}$$

Now the total computational cost to produce output feature maps is given as:

$$\begin{aligned} T_F =& D_F + S_F \\ =& \frac{1}{2}[D_o D_o KKM(1+P)(NP)] \\ &+ \frac{1}{4}[D_o D_o KKM(1+P)(N-NP)] \\ =& D_o D_o KKMN\left(\frac{2P+1+PP}{4}\right) \end{aligned} \tag{4}$$

The total reduction in FLOPS (computation) can be given as:

$$\begin{aligned} R_f =& 1 - \frac{D_F + S_F}{O_F} \\ =& 1 - \left(\frac{2P+1+PP}{4}\right) \end{aligned} \tag{5}$$

The reduction in FLOPS is $R_f = 44\%$ for P=1/2 and the reduction in FLOPS is $R_f = 75\%$ for P=0. Note that this reduction in FLOPS is independent of the model architecture. The propose SkipConv can be applied to any complex deep architecture [20], [21] as well as any efficient architecture [33], [39] (already compact model) to achieve further reduction in computation.

Speedup over standard convolution can be given as:

$$Speedup = \left(\frac{4}{2P+1+PP}\right) \tag{6}$$

As shown in Figure 4, speedup increases as P value decreases. We can use a P value for the trade-off between the

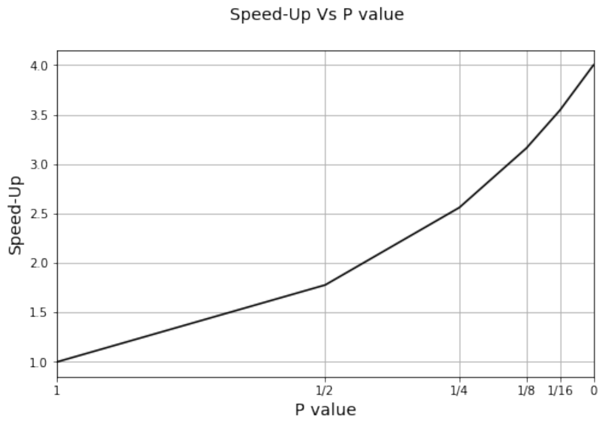

Fig. 4. Speedup over standard convolution for different values of P for the SkipConv convolution.

accuracy and speedup. If we increase the P value, the resulting convolution will be closer to the standard dense convolution. For $P = 1$, it will become standard dense convolution. Value of P is user define, and the user can choose P value depending on the desired speedup before training starts.

TABLE I
THE TABLE SHOWS THE RESULTS FOR VGG-16, AND MOBILENETV2 OVER CIFAR-10 AND CIFAR-100 DATASETS (ACC: ACCURACY, FR: FLOPS REDUCED, AND PM: PRETRAINED MODEL).

Model	Dataset	PM used	VGG-16		MobileNetV2	
			Acc (%)	FR (%)	Acc (%)	FR (%)
Baseline	CIFAR-10	No	93.5	–	94.5	–
P=1/2	CIFAR-10	No	93.5	44%	94.3	44%
P=1/2	CIFAR-10	Yes	93.5	44%	94.4	44%
Baseline	CIFAR-100	No	73.0	–	75.8	–
P=1/2	CIFAR-100	No	72.9	44%	75.4	44%
P=1/2	CIFAR-100	Yes	73.0	44%	75.6	44%

TABLE II
THE TABLE SHOWS THE RESULTS FOR RESNET-56, AND RESNET-164 OVER CIFAR-10 AND CIFAR-100 DATASETS (ACC: ACCURACY, FR: FLOPS REDUCED, AND PM: PRETRAINED MODEL).

Model	Dataset	PM used	ResNet-56		ResNet-164	
			Acc (%)	FR (%)	Acc (%)	FR (%)
Baseline	CIFAR-10	No	93.6	–	94.2	–
P=1/2	CIFAR-10	No	93.6	44%	94.2	44%
P=1/2	CIFAR-10	Yes	93.6	44%	94.2	44%
Baseline	CIFAR-100	No	71.5	–	75.0	–
P=1/2	CIFAR-100	No	71.3	44%	74.8	44%
P=1/2	CIFAR-100	Yes	71.4	44%	74.9	44%

V. EXPERIMENTS AND RESULTS

To show the effectiveness of the proposed approach, we perform extensive experiments over large as well as small scale datasets. We test the proposed approach on a simple architecture (VGG-16) [40], a very deep architecture (ResNet-164) [20], and a compact architecture (MobileNet) [41]. We conduct experiments over the most popular datasets such as CIFAR-10, CIFAR-100 [42], and ImageNet [43]. To show the generalization ability of our approach beyond classification, we experiment with a faster-RCNN object detector [19] over the large scale MS-COCO [44] dataset. We present extensive ablation analysis to show the efficacy of the proposed approach. In the ablation study, we explore different types of skip patterns.

A. VGG-16 on CIFAR-10 and CIFAR-100 datasets

CIFAR [42] dataset consists of 50,000 training and 10,000 test images, where all images are RGB images with a dimension of 32×32. We perform a standard data augmentation technique of random cropping to a size of 32×32 (zero-padded on each side with four pixels before taking a random crop) and random horizontal flipping. For optimization, stochastic gradient descent (SGD) is used with momentum 0.9 and a minibatch size of 128. Initially, the learning rate is set to 0.1 and is decreased by a factor of 5 after every 50 epoch. We set weight decay to 0.0001, and all models are trained from scratch for around 250 epochs in the PyTorch framework. For evaluation, the test images are used.

We take the original VGG-16 model as the baseline. We plug the proposed skip convolution with $P = 1/2$ in the VGG-16 architecture to achieve computation reduction. Experimental results show that we can reduce model computation using proposed SkipConv without sacrificing accuracy. The results are shown in the table I.

B. MobileNet on CIFAR-10 and CIFAR-100 datasets

For CIFAR experiments, we use the same hyper-parameters and settings as described earlier. We train MobileNetV2 [41] on the CIFAR-10 dataset and achieve only 91% accuracy because of three downsample layers, leading to only 4×4 feature maps before the last avg pooling. Therefore, we changed the number of downsampling layers from three to two and achieved 94.5% accuracy. We use the publicly available code[1] for these experiments, which uses only two downsampled layers.

MobileNet [33], [41] is a highly compact model. Therefore, reducing computation in an already compact model is a challenging task. We take the original MobileNetV2 model as the baseline. We plug the proposed skip convolution with $P = 1/2$ in the MobileNetV2 architecture to achieve computation reduction. Experimental results show that we can further reduce the computational requirements of MobileNetV2 using the proposed SkipConv. The results are shown in the table I.

C. ResNet-50/56/164 on CIFAR-10, CIFAR-100, and ImageNet datasets

For CIFAR experiments, we use the same hyper-parameters and settings, as described in the above section. ResNet-56/164 is a deeper architecture as compare to VGG-16. We take the original ResNet-56/164 model as the baseline. We plug the proposed skip convolution with $P = 1/2$ in the ResNet-56/164 architecture to achieve computation reduction. Experimental

[1]https://github.com/tinyalpha/mobileNet-v2_cifar10

TABLE III
THE TABLE SHOWS THE RESULTS FOR RESNET-50 OVER THE IMAGENET DATASET. BOLD VALUES INDICATE THE BEST RESULTS OBTAINED BY OUR METHOD IN THE COMPARISON.

Model	Accuracy (%)	FLOPS Reduced (%)
ResNet-50 (Baseline)	92.2	–
SSS-32 [10]	91.9	31.1%
SPP [11]	90.4	33.3%
CP [36]	90.8	33.3%
ThiNet-70 [9]	92.1	36.8%
VCNNP [7]	92.1	40.0%
SFP [12]	92.0	41.8%
GDP-0.7 [13]	91.1	42.0%
SSS-26 [10]	90.8	43.0%
GAL-0.5 [6]	90.9	43.0%
DBSR [8]	91.8	43.1%
WAE [14]	90.4	46.8%
ResNet-50 P=1/2	**92.1**	44.0%

results show that we can reduce model computation using proposed SkipConv without sacrificing accuracy. The results are shown in the table II.

We conduct large scale ImageNet experiments on ResNet-50 architecture to validate the efficacy of the proposed approach. For ImageNet experiments, we perform standard data augmentation methods of random cropping to a size of 224×224 and random horizontal flipping. For optimization, stochastic gradient descent (SGD) is used with momentum 0.9 and a minibatch size of 256. Initially, the learning rate is set to 0.1 and is decreased by a factor of 10 after every 30 epoch. The models are trained from scratch for around 90 epochs. For evaluation, the validation images are subjected to center cropping of size 224×224, and accuracy (Top-5) on the validation set is reported. SkipConv with $P = 1/2$ shows similar results as baseline[2] with significant FLOPS compression (44%) as shown in table III.

We have also compared the proposed approach with other model compression methods [6]–[14], [36]. Most of the model compression methods require a pretrained model and perform time-consuming iterative pruning and finetuning steps to get a reduction in computation, while the proposed approach further reduces the computational requirements at training time. Therefore, it is not fair to directly compare our approach with model compression methods, but for the sake of completeness, we include this. As shown in table III, computationally efficient models produced by our approach are always better than compressed models produced by other methods [6]–[14], [36].

VI. ABLATION STUDY

We conduct a wide range of experiments to validate the proposed approach. First, we perform ablation on skip density (by varying P values) to create the trade-off between model accuracy and model computation. Next, we compare our approach with additional baseline "Prune-Scratch". Last, we analyze the effect of the skip pattern on model performance.

[2]https://github.com/KaimingHe/deep-residual-networks

TABLE IV
THE TABLE SHOWS THE ERROR (%), FLOPS REDUCED (%), AND COMPARISON WITH OTHER COMPUTATION REDUCTION APPROACHES FOR VARIOUS P VALUES OVER THE CIFAR-10 DATASET. BOLD VALUES INDICATE THE BEST RESULTS OBTAINED BY OUR METHOD IN THE COMPARISON.

Model	Error (%)	FLOPS Reduced (%)
ResNet-56 P=1	6.4	–
VCNNP [7]	7.7	20.3%
Li-B [35]	6.9	28%
GAL-0.6 [6]	6.6	37.6%
NISP [15]	7.0	44%
ResNet-56 P=1/2	**6.4**	**44%**
CP [36]	8.2	50%
AMC [16]	8.1	50%
ENC [5]	7.0	50%
CaP [4]	6.8	50.2%
SFP [12]	6.7	53%
GAL-0.8 [6]	8.4	60.2%
AFP-G [17]	7.1	61%
ResNet-56 P=1/4	**6.6**	**61%**
ResNet-56 P=1/8	6.9	68.4%
ResNet-56 P=0	7.1	75%

A. Ablation on Skip Density

In the proposed approach, FLOPS reduction depends on the P values as shown in table IV, where $P \in [0, 1]$. If we decrease P values, then the density of skip convolutions will increase (low P value results in a high reduction in computation). We evaluate the model's performance over different P value and observe that with a negligible drop in accuracy, we can obtain highly computationally efficient models. For $P = 1/8$, only $(100/8)$ % of the feature maps will be dense. As shown in table IV, we have 68.4% FLOPS reduction with only 0.5% accuracy drop from the baseline. As shown in table IV, computationally efficient models produced by our approach show higher accuracy than other methods [4]–[7], [12], [15]–[17], [35], [36].

B. Comparision with Prune-Scratch

We create another baseline to validate the proposed approach. To create the "Prune-Scratch" baseline, we first prune some $\alpha\%$ of convolutional filters from every layer to reduce FLOPS close to 44% and then pruned model is trained from scratch using same hyper-parameters settings and training schedule.

We observe that the model produced by the proposed approach exhibits far better performance as compare to the Prune-Scratch model over the CIFAR-10 dataset, as shown in table V. The same trend is also observed over the CIFAR-100 dataset.

C. Ablation on Skip Pattern

In the proposed approach, we use an alternative zero row pattern in sparse feature maps. We also test contiguous half zero rows pattern (upper-half or lower-half part is zero in sparse feature map), but the results are not satisfactory as shown in table VI because of information loss in the sparse feature map. This again validates the proposed approach

TABLE V
COMPARISON BETWEEN THE MODELS PRODUCED BY THE PROPOSED APPROACH AND PRUNE-SCRATCH MODELS HAVING THE SAME FLOPS. BOLD VALUES INDICATE THE BEST RESULTS OBTAINED BY OUR METHOD IN THE COMPARISON (FR: FLOPS REDUCED).

Model	Dataset	Accuracy (%)	FR (%)
ResNet-56 Prune-Scratch	CIFAR-10	92.9	44%
ResNet-56 P=1/2	CIFAR-10	**93.6**	44%
ResNet-56 Prune-Scratch	CIFAR-100	70.2	44%
ResNet-56 P=1/2	CIFAR-100	**71.3**	44%

TABLE VI
THE TABLE SHOWS THE RESULTS FOR VARIOUS SKIP PATTERNS OVER THE CIFAR-100 DATASET (FR: FLOPS REDUCED).

Model	Skip Pattern	Accuracy (%)	FR (%)
ResNet-56	None	71.5	0%
ResNet-56 P=1/2	Alternative Rows	71.3	44%
ResNet-56 P=1/2	Contiguous Half Rows	70.1	44%

for model computation reduction. Contiguous half zero rows pattern in a sparse feature map can be produced by skipping contiguous half rows during convolution.

D. Reduction in Computation for Pretrained Models

Our approach does not require a pretrained model and does train from scratch. But if the pretrained model is available, then the proposed convolution (SkipConv) can be directly applied on the pretrained model to get computation reduction. In our approach, we are skipping convolutions to reduce redundant computation. Therefore after loading model weights from the pretrained model, we finetune the model for the proposed SkipConv convolutions. As shown in table I, II, computationally efficient models using pretrained models show similar or slightly better accuracy performance than models trained from scratch.

VII. GENERALIZATION ABILITY

To show the generalization ability of the model produced by our proposed approach, we experiment on the F-RCNN [19] object detector over MS-COCO [44] dataset. MS-COCO detection dataset contains 80 object categories [44]. Here all 80k train images and 35k val images are used for training (trainval35K) [45].

We are reporting the detection accuracies over the 5k unused val images (minival). To create a baseline, we trained Faster-RCNN with the ImageNet pretrained ResNet-50 base model. The results are shown in table VII. In the second experiment, we use ResNet-50 P=1/2 model (table III) as a base network in Faster-RCNN P=1/2. We use a publicly available code[3] for Faster R-CNN with ResNet-50 as a base network. In the Faster-RCNN implementation, we use ROI Align and use stride=1 for the last block of the convolutional layer (layer4) in the base network.

[3]https://github.com/jwyang/faster-rcnn.pytorch

TABLE VII
GENERALIZATION RESULTS OVER MS-COCO DATASET. RESNET-50 P=1/2 USED AS A BASE MODEL FOR FASTER-RCNN OBJECT DETECTOR.

Model	data	Avg. Precision, IoU:	
		0.5:0.95	0.5
F-RCNN (Baseline)	trainval35K	30.3	51.3
F-RCNN P=1/2	trainval35K	30.3	51.2

From our results, we show that the Faster-RCNN based on ResNet-50 gives similar results (table VII) with our ResNet-50 P=1/2 model. Therefore, our ResNet-50 P=1/2 model retains its feature representation capacity even after computation reduction.

VIII. IMPLEMENTATION DETAILS

None of the existing frameworks support the proposed SkipConv convolution. We, therefore, implemented this in PyTorch. Practical speedup highly depends on the frameworks, hardware, and efficient implementation. We have implemented an initial implementation of the proposed convolution ourselves. This is not highly optimized, and therefore we believe it cannot be directly compared with existing highly efficiently implemented convolutions in terms of absolute numbers. We, therefore, presented an analysis of computational complexity (FLOPS) for the proposed work because it is independent of implementation.

We implement SkipConv convolution by dividing existing convolution into multiple parts. Similarly, we divided input feature maps corresponding to each part of the convolution and performed the convolution operation. After that, we combine (add) the results from all sub-convolutions after using appropriate zero row paddings.

IX. CONCLUSION

In this work, we have proposed a novel SkipConv for reducing computation in the deep CNNs. Our proposed SkipConv can be plugged into any deep architecture for computational reduction. We empirically show the usability of SkipConv in object classification and detection tasks. Our extensive experimental results validate the strength of the proposed approach. We experimentally show the usability of SkipConv for already compact architectures (MobileNet). Using the proposed SkipConv, we can reduce computation at training as well as at test time. Proposed SkipConv is complementary to all other model compression approaches and hence can be used in conjunction with other model compression (quantization, pruning, and sparsity in model parameters) approaches to get a further reduction in computation. We present an extensive ablation study to validate the SkipConv (skip density, prune-scratch, and skip pattern) approach.

REFERENCES

[1] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, pp. 91–110, 2004.
[2] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *international Conference on computer vision & Pattern Recognition (CVPR’05)*, vol. 1, 2005, pp. 886–893.

[3] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE transactions on pattern analysis and machine intelligence*, pp. 500–513, 2011.

[4] B. Minnehan and A. Savakis, "Cascaded projection: End-to-end network compression and acceleration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 715–10 724.

[5] H. Kim, M. U. K. Khan, and C.-M. Kyung, "Efficient neural network compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 569–12 577.

[6] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured cnn pruning via generative adversarial learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2790–2799.

[7] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational convolutional neural network pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2780–2789.

[8] D. teja Vooturi, G. Varma, and K. Kothapalli, "Dynamic block sparse reparameterization of convolutional neural networks," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[9] J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, and W. Lin, "Thinet: pruning cnn filters for a thinner net," in *IEEE transactions on pattern analysis and machine intelligence*. IEEE, 2018.

[10] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 304–320.

[11] H. Wang, Q. Zhang, Y. Wang, and H. Hu, "Structured probabilistic pruning for convolutional neural network acceleration," in *The British Machine Vision Conference (BMVC)*, 2018.

[12] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 2234–2240.

[13] S. Lin, R. Ji, Y. Li, Y. Wu, F. Huang, and B. Zhang, "Accelerating convolutional networks via global & dynamic filter pruning." in *IJCAI*, 2018, pp. 2425–2432.

[14] T. Chen, L. Lin, W. Zuo, X. Luo, and L. Zhang, "Learning a wavelet-like auto-encoder to accelerate deep neural networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[15] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[16] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[17] X. Ding, G. Ding, J. Han, and S. Tang, "Auto-balanced filter pruning for efficient convolutional neural networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[22] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri, "Hetconv: Heterogeneous kernel-based convolutions for deep cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4835–4844.

[23] P. Singh, V. S. R. Kadi, N. Verma, and V. P. Namboodiri, "Stability based filter pruning for accelerating deep cnns," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1166–1174.

[24] P. Singh, V. K. Verma, P. Rai, and V. Namboodiri, "Leveraging filter correlations for deep model compression," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 835–844.

[25] P. Singh, R. Manikandan, N. Matiyali, and V. Namboodiri, "Multi-layer pruning framework for compressing single shot multibox detector," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1318–1327.

[26] V. K. Verma, P. Singh, V. Namboodri, and P. Rai, "A "network pruning network" approach to deep model compression," in *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[27] P. Singh, P. Mazumder, and V. Namboodiri, "Accuracy booster: Performance boosting using feature map re-calibration," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 884–893.

[28] P. Singh, V. S. R. Kadi, and V. P. Namboodiri, "Falf convnets: Fatuous auxiliary loss based filter-pruning for efficient deep cnns," *Image and Vision Computing*, p. 103857, 2019.

[29] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri, "Hetconv: Beyond homogeneous convolution kernels for deep cnns," *International Journal of Computer Vision*, pp. 1–21, 2019.

[30] P. Mazumder, P. Singh, and V. Namboodiri, "Cpwc: Contextual point wise convolution for object recognition," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 4152–4156.

[31] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri, "Play and prune: Adaptive filter pruning for deep model compression," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[32] P. Singh, M. Varshney, and V. Namboodiri, "Cooperative initialization based deep neural network training," in *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[33] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[34] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *International Conference on Learning Representations*, 2016.

[35] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *International Conference on Learning Representations*, 2017.

[36] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[37] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5058–5066.

[38] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *The European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 525–542.

[39] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[41] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[42] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," in *International Journal of Computer Vision (IJCV)*, 2015, pp. 211–252.

[44] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *The European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[45] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection." in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, p. 4.