MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning

Zechun Liu, Haoyuan Mu , Xiangyu Zhang , Zichao Guo , Xin Yang , Tim Kwang-Ting Cheng , Jian Sun
Hong Kong University of Science and Technology
Tsinghua University
Megvii Inc (Face++) Huazhong University of Science and Technology

Presenter: Huidong Xie

Contributions

- Propose MetaPruning for channel pruning.
- MetaPruning liberates human from cumbersome hyperparameter tuning and enables the direct optimization with desired metrics.
- MetaPruning can easily apply hard constraints
- MetaPruning is able to effortlessly prune the channels in ResNet-like network
- MetaPruning is designed to directly find the optimal pruned network structures.
- For channel-pruning task, it is hard to enumerate every choice. They solve this issue by training the PruningNet with weight prediction.

Relative works: pruning

- Weight pruning results in unstructured sparse-filter
- Channel pruning remove entire weigh filters instead of individual weights.
- Traditional channel pruning trim channels based on the importance of each channel.
- In most traditional channel pruning, compression ratio for each layer need to be manually set based on human experts, which is time-consuming and may be trapped in sub-optimal solutions.

Relative works: AutoML

- Through reinforcement learning or an automatic feed-back loop
- AutoML helps to alleviate the manual efforts for tuning hyper-parameters in channel pruning.

Relative works: Meta Learning

- MetaPruning is inspired by Meta-learning for weight prediction.
- Weight prediction: weights of a neural network are predicted by another neural network rather than directly learned.

Relative works: Neural Architecture Search

- Try to find the optimal network structures and hyper-parameters.
- Training multiple choices with drop-path, it designed to search for the path with highest accuracy in the trained network.

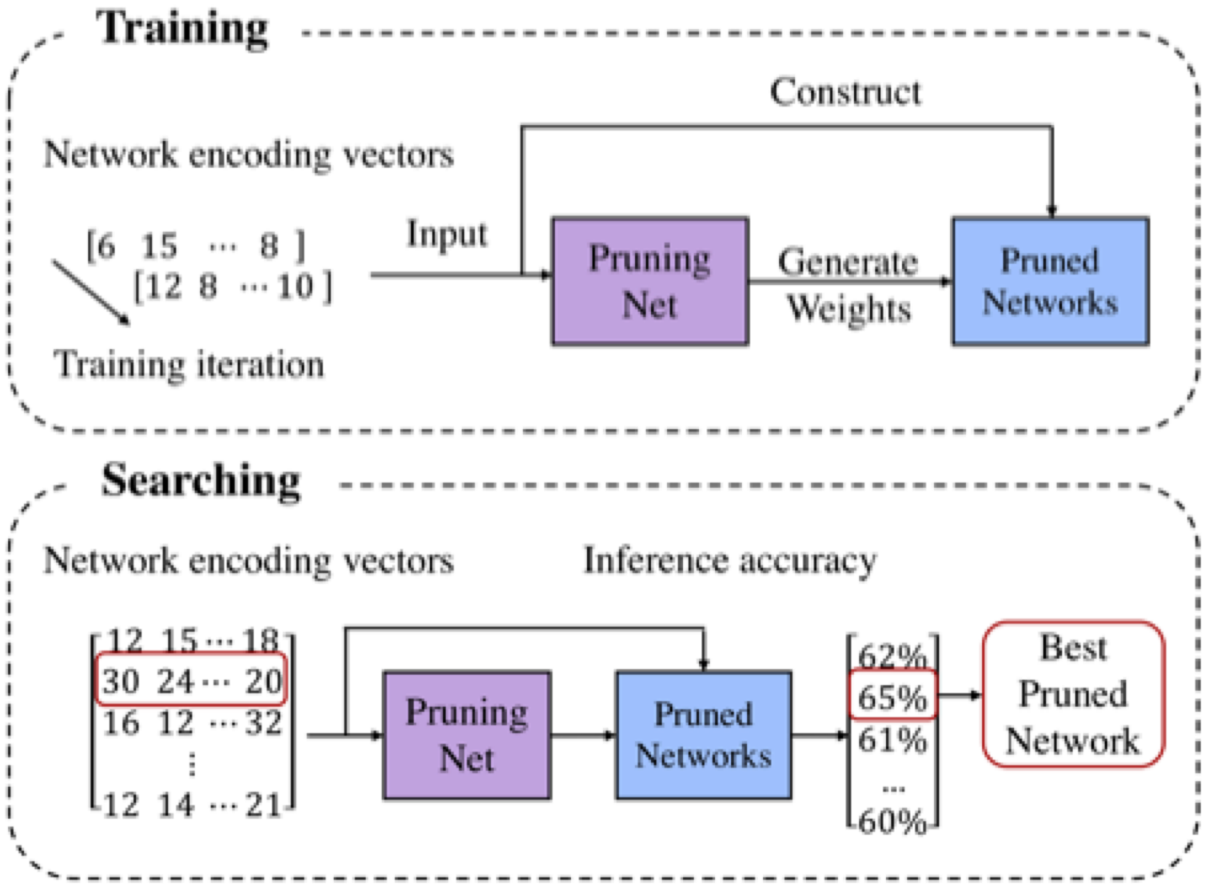
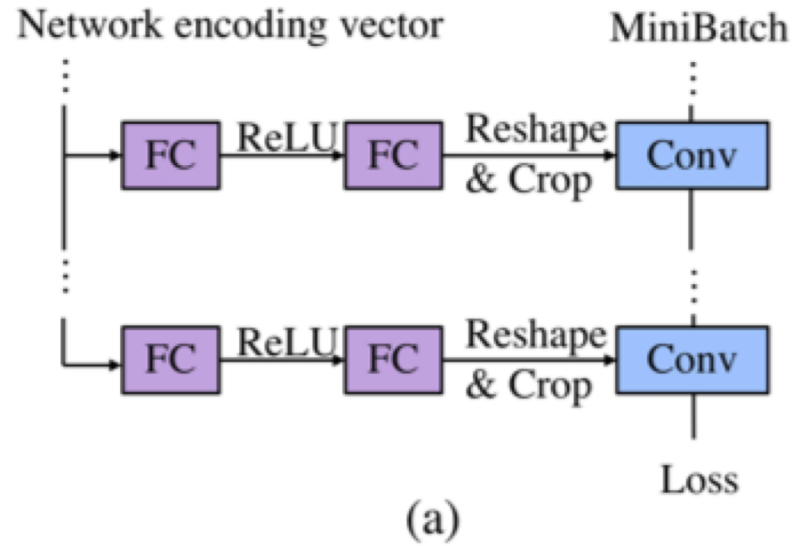
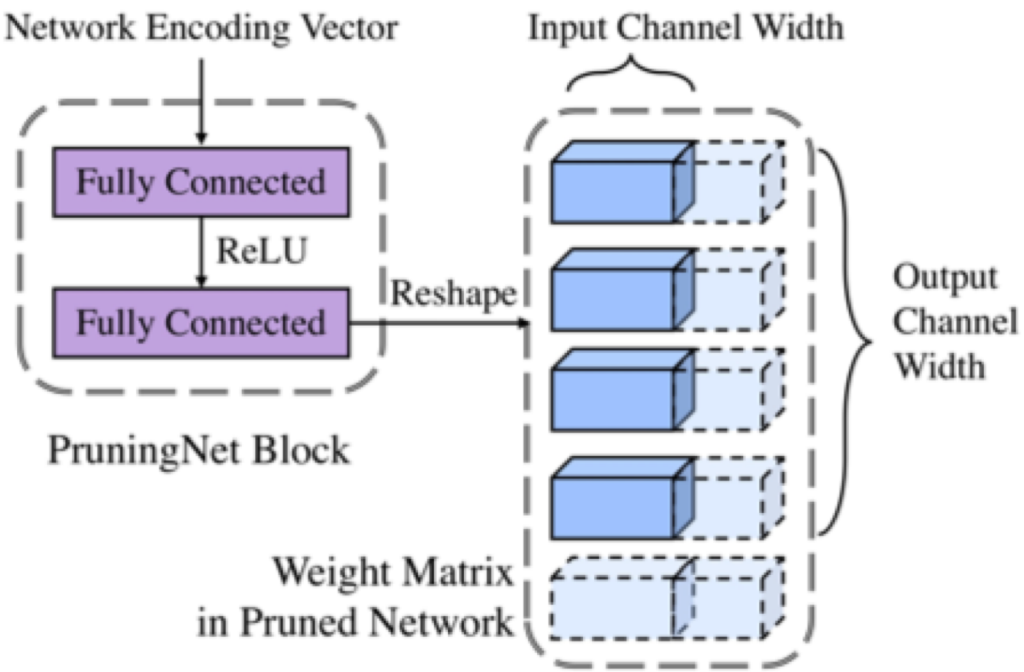


Figure 1. Our MetaPruning has two steps. 1) training a PruningNet. At each iteration, a network encoding vector (i.e., the number of channels in each layer) is randomly generated. The Pruned Network is constructed accordingly. The PruningNet takes the network encoding vector as input and generates the weights for the Pruned Network. 2) searching for the best Pruned Network. We construct many Pruned Networks by varying network encoding vector and evaluate their goodness on the validation data with the weights predicted by the PruningNet. No finetuning or re-training is needed.

$$(c_1, c_2, ...c_l)^* = \arg\min_{c_1, c_2, ...c_l} \mathcal{L}(\mathcal{A}(c_1, c_2, ...c_l; w)) \quad (1)$$
$$s.t. \quad \mathcal{C} < constraint,$$



(a)



(b)

Figure 3. (a) The network structure of PruningNet connected with Pruned Network. The PruningNet and the Pruned Network are jointly trained with input of the network encoding vector as well as a minibatch of images. (b) The reshape and crop operation on the weight matrix generated by the PruningNet block.

Training stages

Algorithm 1 Evolutionary Search Algorithm

Hyper Parameters: Population Size: $\mathcal{P}$, Number of Mutation: $\mathcal{M}$, Number of Crossover: $\mathcal{S}$, Max Number of Iterations: $\mathcal{N}$.
Input: PruningNet: $PruningNet$, Hard constraints $\mathcal{C}$.
Output: Most accurate gene: $\mathcal{G}_{Top}$.

$\mathcal{G}_0$ = Random($\mathcal{P}$), s.t $\mathcal{C}$;
$\mathcal{G}_{topK} = \emptyset$;
for $i = 0 : \mathcal{N}$ **do**
  $\{\mathcal{G}_i, \text{accuracy}\}$ = Inference($PruningNet(\mathcal{G}_i)$);
  $\mathcal{G}_{topK}, accuracy_{topK}$ = TopK($\{\mathcal{G}_i, accuracy\}$);
  $\mathcal{G}_{mutation}$ = Mutation($\mathcal{G}_{topK}, \mathcal{M}$), s.t $\mathcal{C}$;
  $\mathcal{G}_{crossover}$ = Crossover($\mathcal{G}_{topK}, \mathcal{S}$), s.t $\mathcal{C}$;
  $\mathcal{G}_i = (\mathcal{G}_{mutation}, \mathcal{G}_{crossover})$;
end for
$\mathcal{G}_{top}$ $accuracy_{top}$= Top1($\{\mathcal{G}_{\mathcal{N}}, accuracy\}$);
return $\mathcal{G}_{top}$;

- Stage 1: PruningNet is trained from scratch with stochastic training.
- Stage 2: use search algorithm to find the best pruned network and then train it from scratch.

MobileNet V1/V2

- Decode the network encoding vector to the input compression ratio and output compression ratio of each dw-conv-pw-conv block, generating $[\frac{C_{output}^{l-1}}{C_{totaloutput}^{l-1}}, \frac{C_{output}^{l}}{C_{totaloutput}^{l}}]$, and then input to the first FC layer. The first FC layer has output size 64 for l^{th} pruningNet block.
- The second FC layer output vector with length of $C_{out}^{l} \times C_{in}^{l} \times w^{l} \times h^{l}$
- The number of output channels for each dw-conv-pw-conv block is selected in $\left[int\left(0.1 \times C_{totaloutput}^{l}\right), C_{totaloutput}^{l}\right]$ with step $int\left(0.03 \times C_{totaloutput}^{l}\right)$
- $[\frac{C_{output}^{b-1}}{C_{totaloutput}^{b-1}}, \frac{C_{output}^{b}}{C_{totaloutput}^{b}}, \frac{C_{middle}^{b}}{C_{middleoutput}^{b}}]$

MobileNet V1/V2

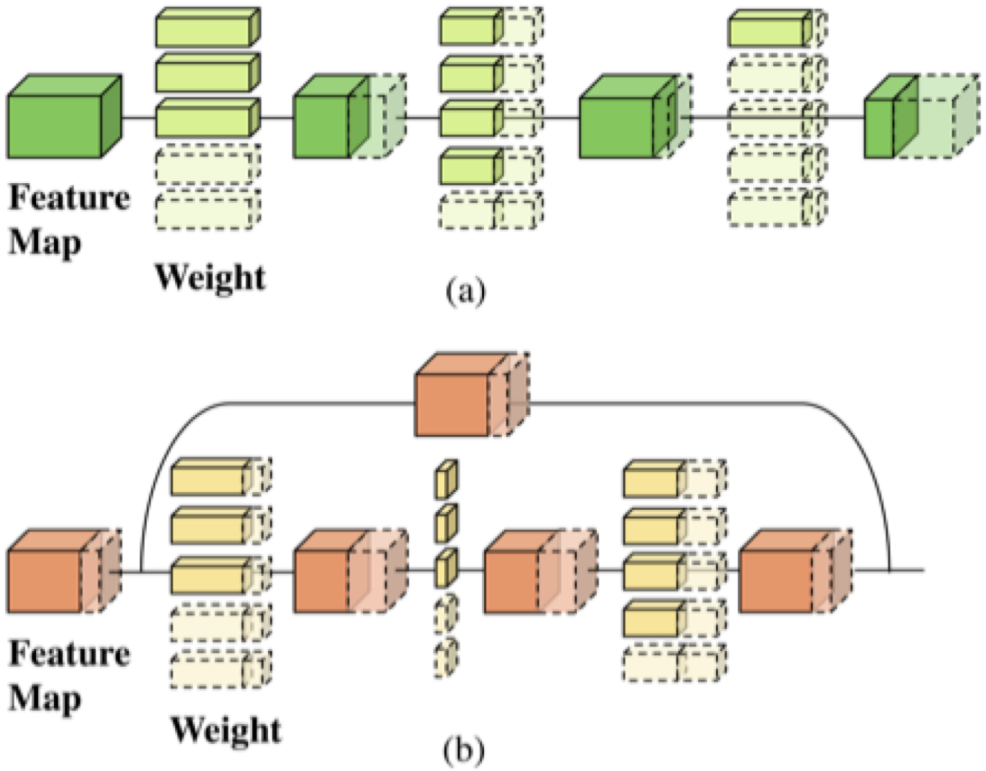


Figure 4. Channel Pruning schemes considering the layer-wise inter-dependency. (a) For the network without shortcut, *e.g.*, MobileNet V1, we crop the top left of the original weight matrix to match the input and output channels. For simplification, we omit the depth-wise convolution here; (b) For the network with shortcut, *e.g.*, MobileNet V2, we prune the middle channels in the blocks while keep the input and output of the block being equal.

Results Comparisons with state-of-the-arts

Uniform Baselines: MobileNets proposes to use multipliers to uniformly prune the channel width in each layer to meet the resource constraints.

Table 1. This table compares the top-1 accuracy of MetaPruning method with the uniform baselines on MobileNet V1 [20].

Uniform Baselines			**MetaPruning**	
Ratio	Top1-Acc	FLOPs	Top1-Acc	FLOPs
1×	70.6%	569M	–	–
0.75×	68.4%	325M	**70.6%**	324M
0.5×	63.7%	149M	**66.1%**	149M
0.25×	50.6%	41M	**57.2%**	41M

Table 2. This table compares the top-1 accuracy of MetaPruning method with the uniform baselines on MobileNet V2 [41]. The MobileNet V2 only reports the accuracy with 585M and 300M FLOPs, so we apply the uniform pruning method on MobileNet V2 to obtain the baseline accuracy for the networks with other FLOPs.

Uniform Baselines		**MetaPruning**	
Top1-Acc	FLOPs	Top1-Acc	FLOPs
74.7%	585M	–	–
72.0%	313M	**72.7%**	291M
67.2%	140M	**68.2%**	140M
66.5%	127M	**67.3%**	124M
64.0%	106M	**65.0%**	105M
62.1%	87M	**63.8%**	84M
54.6%	43M	**58.3%**	43M

Results Comparisons with state-of-the-arts

Table 3. This table compares the top-1 accuracy of MetaPruning method with state-of-the-art AutoML methods.

Network	FLOPs	Top1-Acc
0.75x MobileNet V1 [20]	325M	68.4%
NetAdapt [48]	284M	69.1%
AMC [18]	285M	**70.5%**
MetaPruning	281M	70.4%
0.75x MobileNet V2 [41]	220M	69.8%
AMC [18]	220M	70.8%
MetaPruning	217M	**71.2%**

AMC utilizes reinforcement learning to iteratively prune channels in each layer taking the accuracy as well as the latency as the reward.

NetAdapt automatically decide pruning how much proportion in which layer based on a feedback loop with latency estimated from the device.

Results Pruning under latency constraint

Table 4. This table compares the top-1 accuracy of MetaPruning method with the MobileNet V1 [20], under the latency constraints. Reported latency is the run-time of the corresponding network on Titan Xp with a batch-size of 32

Uniform Baselines			MetaPruning	
Ratio	Top1-Acc	Latency	Top1-Acc	Latency
1×	70.6%	0.62ms	–	–
0.75×	68.4%	0.48ms	**70.5%**	0.48ms
0.5×	63.7%	0.31ms	**67.4%**	0.30ms
0.25×	50.6%	0.17ms	**59.6%**	0.17ms

Table 5. This table compares the top-1 accuracy of MetaPruning method with the MobileNet V2 [41], under the latency constraints. We re-implement MobileNet V2 to obtain the results with 0.65 × and 0.35 × pruning ratio. This pruning ratio refers to uniformly prune the input and output channels of all the layers.

Uniform Baselines			MetaPruning	
Ratio	Top1-Acc	Latency	Top1-Acc	Latency
1.4×	74.7%	0.95ms	–	–
1×	72.0%	0.70ms	**73.2%**	0.67ms
0.65×	67.2%	0.49ms	**71.7%**	0.47ms
0.35×	54.6%	0.28ms	**64.5%**	0.29ms

Pruned results visualization

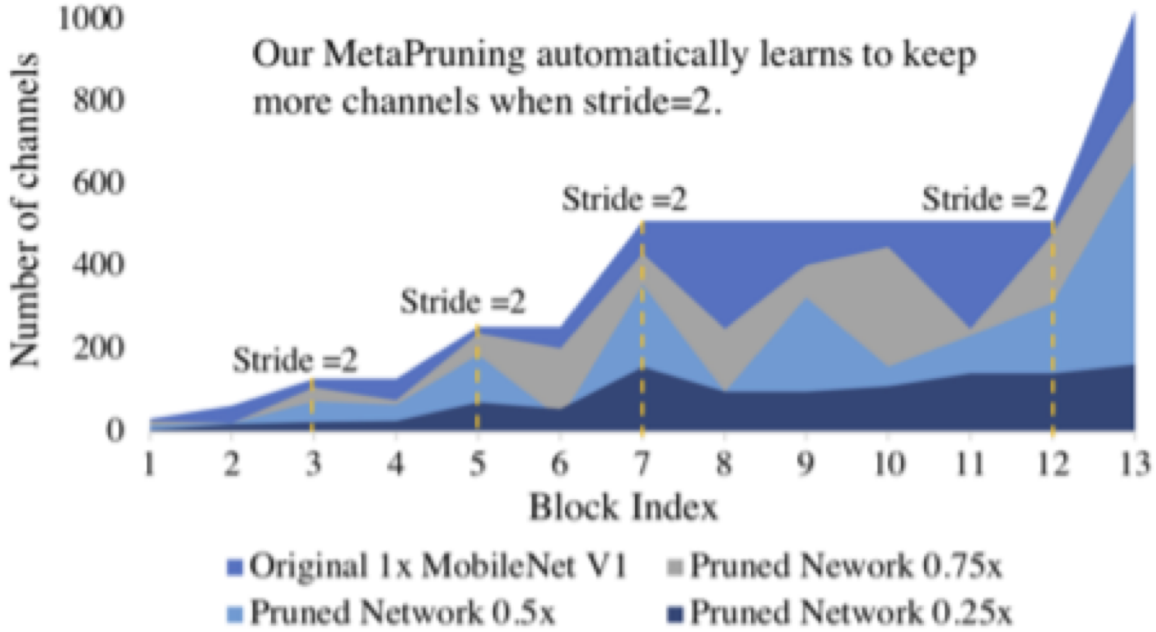


Figure 5. This figure presents the number of output channels of each block of the pruned MobileNet v1. Each block contains a 3x3 depth-wise convolution followed by a 1x1 point-wise convolution, except the first block is composed by a 3x3 convolution only.

Pruned results visualization

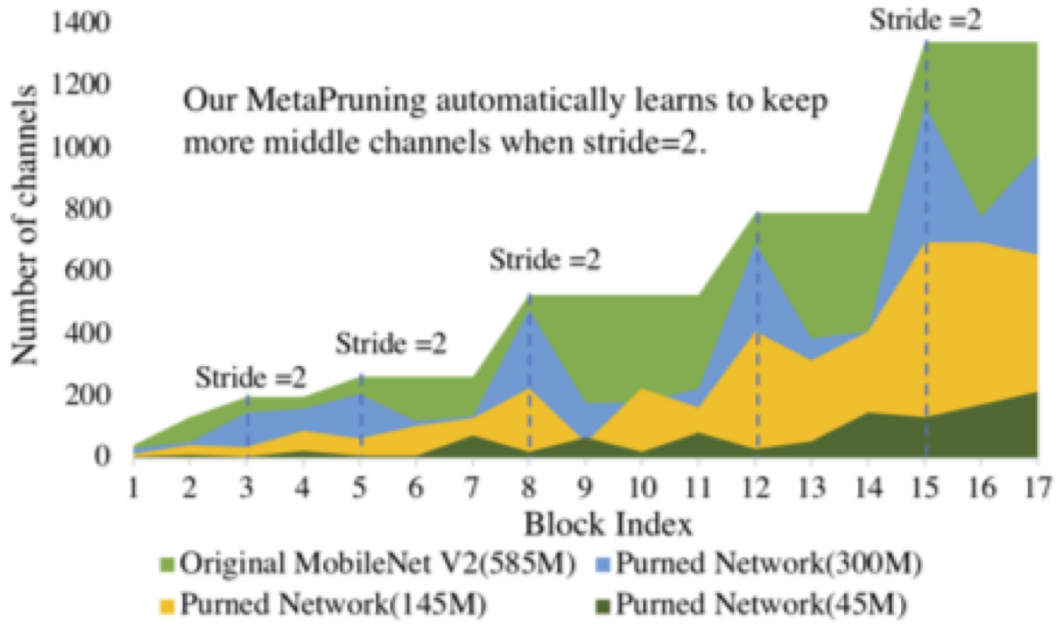


Figure 6. A MobileNet V2 block is constructed by concatenating a 1x1 point-wise convolution, a 3x3 depth-wise convolution and a 1x1 point-wise convolution. This figure illustrates the number of middle channels of each block.

Pruned results visualization

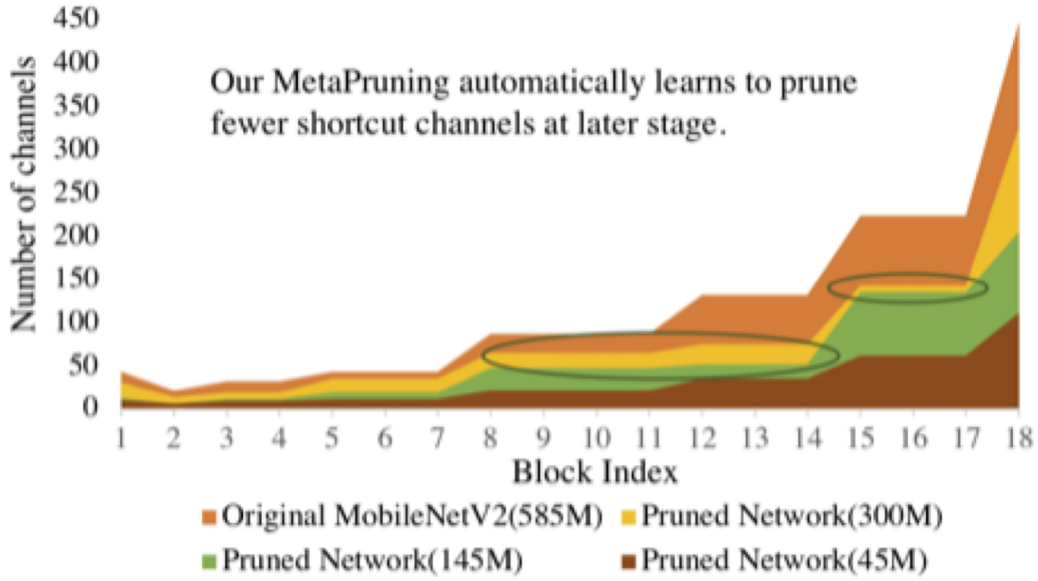


Figure 7. In MobileNet V2, each stage starts with a bottleneck block with differed input and output channels and followed by several repeated bottleneck blocks. Those bottleneck blocks with the same input and output channels are connected with a shortcut. MetaPruning prunes the channels in the shortcut jointly with the middle channels. This figure illustrates the number of shortcut channel in each stage after the MetaPruning.