Revisiting Random Channel Pruning for Neural Network Compression

Yawei Li, Kamil Adamczewski, Wen Li, Shuhang Gu, Radu Timofte, Luc Van Gool; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 191-201

Presentation of Federico Bussolino(s317641), Abolfazl Javidian(s314441), Francesco Baracco(s317743)

Channel Pruning for Neural Network

- **Definition:**

Channel pruning is a technique used in the optimization of neural networks, particularly deep convolutional neural networks (CNNs).

- **Goal:**

The goal of channel pruning is to "**reduce the computational cost and memory requirements maintaining a good accuracy**" of a neural network by identifying and removing unnecessary channels (feature maps) in each layer.

- **Note:**

While channel pruning can significantly reduce the computational cost of a neural network, the degree of pruning and its impact on model performance depend on the specific architecture, dataset, and task at hand. Therefore, "**careful experimentation and validation**" are essential when applying channel pruning to a neural network.

Why and How

- **Convolutional Layers and Channels:**

In a CNN, convolutional layers consist of multiple channels, each producing a set of feature maps. These feature maps capture different patterns and spatial hierarchies in the input data. However, **not all channels are equally important for the network's performance.**

- **Importance Metrics:**

Channel pruning methods typically involve defining metrics to measure the importance of each channel. These **metrics can include the L1 or L2 norm of the channel's weights, activation values, or gradients during training**. Channels that contribute less to the network's performance are candidates for pruning.

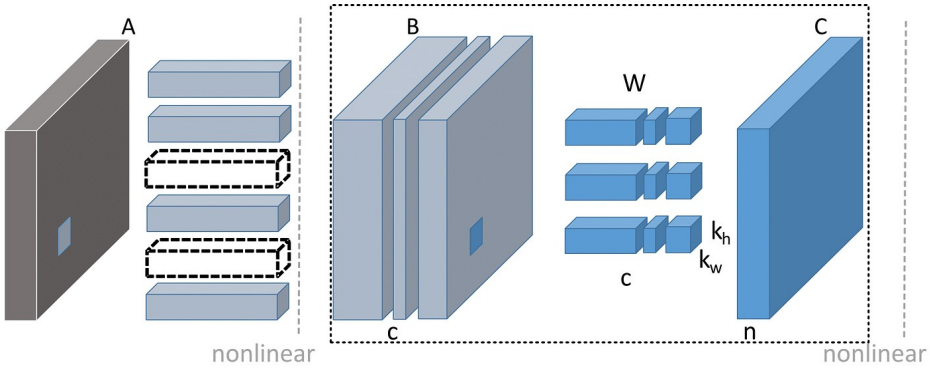
- **Thresholding:**

A threshold is applied to identify channels whose importance falls below a certain criterion. Channels with values below this threshold are considered less crucial and are candidates for removal. The threshold for pruning can be determined using various criteria, such as a fixed percentage of the lowest importance channels, a global threshold for the entire layer, or an adaptive threshold based on a validation set.

- **Types of pruning:**
  - ✓ **Structured**: preserves spatial structures by removing entire filters or filter groups, maintaining the connectivity patterns within each channel.
  - ✓ **Unstructured**: removes individual channels without considering their spatial location in the input data.

Pruning

Channel level pruning

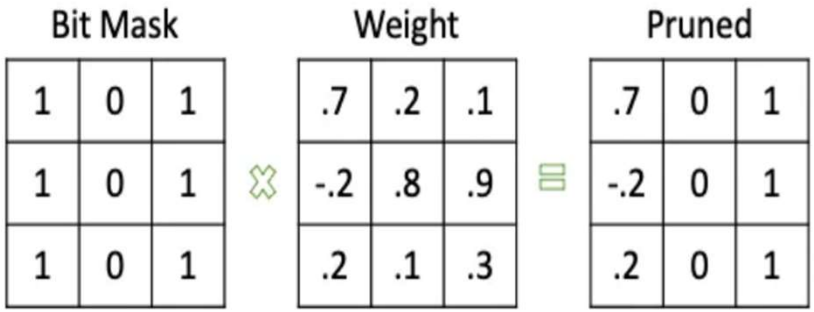


Removing N filter from one layer the size of next layer's filters becomes (C-N)xHxW where C is number of the channel of layer L in the non-pruned network.

→ **We have to recalculate the weights**

→ $\boldsymbol{\mu, \sigma}$ in layer (or group) normalization **must be updated**

Weight level pruning



Removing just weights doesn't change the shape of next layer filters.

Properties of channel pruning space

- **Property 1:**

The channel configuration space is discrete, so conducting a differentiable analysis in this space is impossible. This property constitutes a major challenge for channel pruning and architecture search methods. To search for the space, reinforcement learning, evolutionary algorithm, and proximal gradient descent have been utilized.

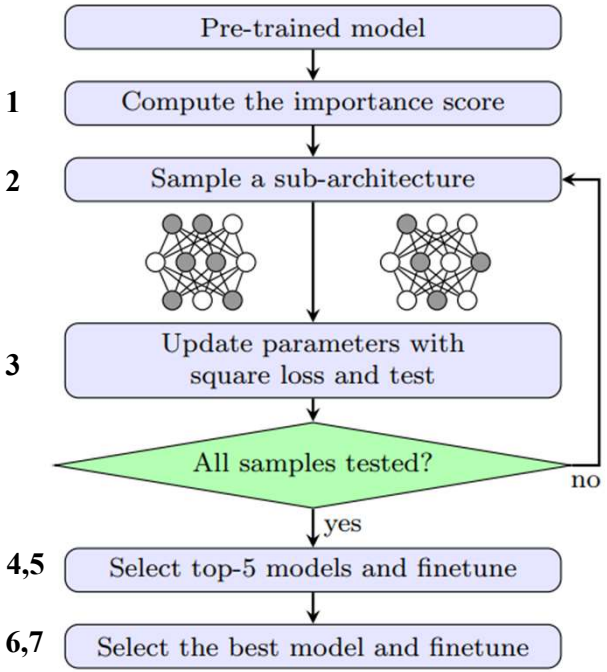
- **Property 2:**

Slightly changing the channel number of a network does not change the accuracy of the network too much. Regularization-based methods gradually update from initial networks to the optimal solutions. By contrast, random pruning only needs to get a sample in the neighborhood of the optimal solution instead of the optimal solution itself.

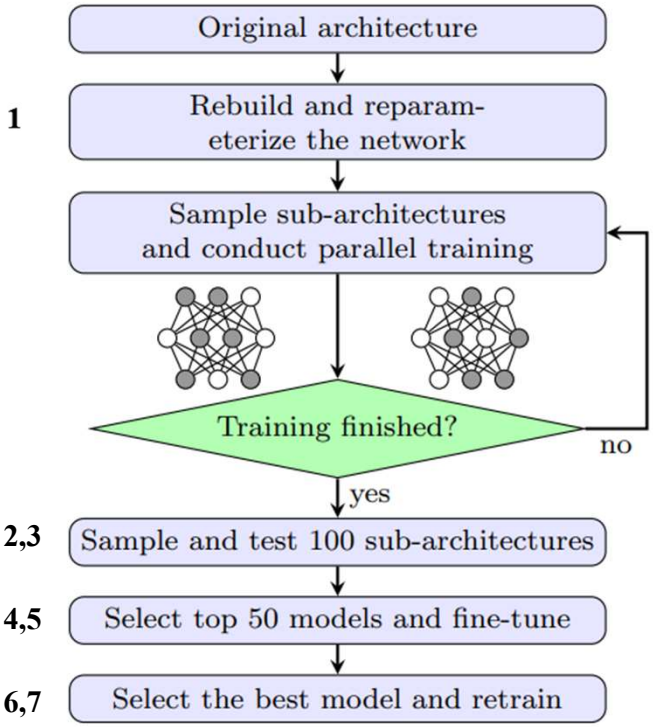
Methodology

The paper aims to show performances of random pruning in two different contexts:

- **pruning pre-trained network**



- **pruning network from scratch**



Pruning Pre-trained network

Importance score and sampling sub-networks

1. Calculate the importance score of the channels.
2. Select **N sub-architectures. Each has to be chosen with following procedure:**

   For each layer:

   a. Randomly select $\#remaining\_channel$.

   b. Verify that $1 > \frac{\#remaining\_channel}{\#original\_channel} > \eta$, where η is chosen equal to or slightly smaller than the overall objective pruning ratio γ.

   c. Remove $(\#original\_channel - \#remaining\_channel)$ channels from the layer starting from the channel with lower importance score.

   Verify that network respect*: $\left|\frac{C_{prune}}{C_{orig}} - \gamma\right| \leq T$.

*C_{prune} and C_{orig} denotes the floating point operations (FLOPs) of the pruned network and the original network, respectively. γ is the overall pruning ratio of the network and T is the threshold that confines the difference between the actual and target pruning ratio.

Pruning Pre-trained network

Updating parameters and selecting model

3. For each of the N sub-architectures:
    a. Update parameter by minimizing difference between feature maps of pruned and original networks*
    b. Evaluate accuracy on validation set
4. Top 5 performing models are selected among the N proposed
5. For each of those 5:
    a. Fine-tune the selected model for several epochs
    b. Evaluate accuracy on validation set
6. Select the model with best accuracy
7. Fine-tune for more epochs the best model

*Parameter updating is described in next slide

Pruning Pre-trained network

Updating parameters by minimizing distance between feature maps

Since updating parameter by fine-tuning all the N networks is expensive an alternative is proposed on this paper:

Let $\mathbf{F}_p \in \mathbb{R}^{n' \times d}$ and $\mathbf{F}_o \in \mathbb{R}^{n \times d}$ denote the feature map of the pruned network and the original network, respectively. Since the network is pruned, its feature map has less channels than the original network ($n' < n$).

The parameters in the pruned network is updated by minimizing the following loss function:

$$\mathcal{L} = \arg\min_{\mathbf{X}} \|\hat{\mathbf{F}}_o - \mathbf{X}\mathbf{F}_p\|_2^2$$

Where $\hat{\mathbf{F}}_o \in \mathbb{R}^{n' \times d}$ is the feature map of the original network (calculated with original filters) with the corresponding channels removed and $\mathbf{X} \in \mathbb{R}^{n' \times n'}$ is the additional parameter that updates the pruned network.

The parameter **X** can be derived with least square solvers. It **can be further merged with the original parameter in the layer of the network.**

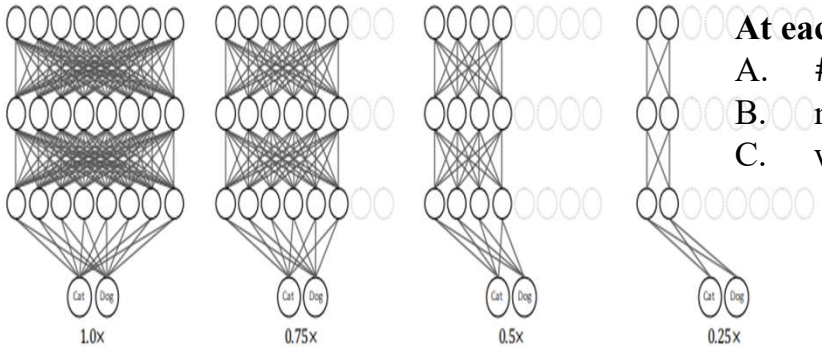
This parameter updating procedure is done layer-wise.

NOTE: **only for 1^{st} feature map $\hat{\mathbf{F}}_o = \mathbf{F}_p$**, for this reason we need to minimize the difference.

Pruning network from scratch

Training slimmable NN

1. Train a slimmable neural network (neural network that can be runned with different width, 4 in parallel)*



At each mini-batch iteration select 3 networks randomly respecting:

A. $\#tot\_remaining\_channels$ must be multiples of 8.
B. min number of channels for each layer is rounded to multiples of 8.
C. width ($\#tot\_remaining\_channel$) cannot increase.

Figure shows an example of a slimmable network that can switch between four model variants with different numbers of active channels. The parameters of all model variants are shared.

→ **Thanks to parallel training the network will gain the capability of interpolating the accuracy of unsampled networks and similarly to pre-trained network it take less epoch to give good results.**

Pruning network from scratch

Sampling sub-networks and selecting best model

2. Compute importance score and **Random sample a population of N* subnetwork (same process):**

   For each layer:

   a. Randomly select $\#remaining\_channel$

   b. Verify that $1 > \frac{\#remaining\_channel}{\#original\_channel} > \eta$, where η is chosen equal to or slightly smaller than the overall objective pruning ratio γ

   c. Remove $(\#original\_channel - \#remaining\_channel)$ channels from the layer starting from the channel with lower importance score.

   Verify that network respect this characteristic*: $\left|\frac{C_{prune}}{C_{orig}} - \gamma\right| \leq T$

3. For each of the N sub-architectures:

   Directly evaluate accuracy on validation set using slimmable NN

4. Top 50 performing models are selected among the N proposed
5. For each of those 50:
   a. Further train the selected model for some epochs
   b. Evaluate accuracy on validation set
6. Select the model with best accuracy
7. Re-train the best model from scratch

*N=100 in the image

Experimental results

Comparing importance score metrics

Here we report the network that obtained best benefit from random pruning

Benchmarking random channel pruning criteria on ImageNet:

Criterion	Top-1 Error (%)	Top-5 Error (%)	FLOPs [G] / Ratio (%)	Params [M] / Ratio (%)
VGG16, Target FLOPs Ratio 70 %				
Baseline	26.63	8.5	15.50 /100.00	138.4 /100.00
L1	27.14	8.68	11.11 /71.67	112.3 /81.14
L2	27.01	8.82	10.87 /70.10	128.8 /93.09
GM	27.96	8.77	11.09 /71.55	108.4 /78.34
TE	27.04	8.77	10.65 /68.70	130.4 /94.27
ES	26.76	8.60	10.38 /66.98	130.9 /94.63
KL	27.22	8.88	10.91 /70.37	128.9 /93.15

Benchmarking random channel pruning criteria on CIFAR10:

Criterion	Top-1 Error (%)	Top-5 Error (%)	FLOPs [G] / Ratio (%)	Params / Ratio (%)
VGG, CIFAR10				
Baseline	5.67	0.58	313.80 /100.00	14.73M /100.00
L1	6.1	0.69	160.50 /51.15	5.05M /34.32
L2	6.06	0.67	150.60 /47.99	6.20M /42.11
GM	5.99	0.52	154.60 /49.27	4.13M /28.04
TE	6.51	0.61	157.00 /50.03	5.84M /39.63
ES	6.21	0.64	157.20 /50.10	7.06M /47.90
KL	6.19	0.66	161.50 /51.47	6.52M /44.26

Experimental results

Comparing random channel pruning with other methods

From the result we can see that Random Pruning perform well even when compared with more complex pruning methods.

NOTE: method that outperforms random pruning are usually trained for more epochs.

Methods	Epoch	Top-1 Err. (%)	Top-5 Err. (%)	FLOPs Ratio	Params Ratio
ResNet50, ImageNet					
SFP [17]	100	25.39	7.94	58.2	–
GAL-0.5 [38]	30	28.05	9.06	56.97	83.14
SSS [23]	100	28.18	9.21	56.96	61.15
HRank [37]	480	25.02	7.67	56.23	63.33
Random Pruning	25	25.85	8.01	50.72	54.99
Random Pruning	75	25.22	7.69	50.72	54.99
Random Pruning	120	24.87	7.48	48.99	54.12
AutoPruner [43]	32	25.24	7.85	48.79	–
Adapt-DCP [39]	120	24.85	7.70	47.59	45.01
FPGM [19]	90	25.17	7.68	46.5	–
DCP [66]	60	25.05	7.68	44.50	48.44
ThiNet [45]	87	27.97	9.01	44.17	–
MetaPruning [40]	160	24.60	–	48.78	–
AutoSlim [57]	150	24.40	–	80.60	–
MobileNetV2, ImageNet2012					
MetaPruning [40]	160	28.80	–	72.33	–
Random Pruning	120	29.10	–	70.87	–
AMC [18]	120	29.20	–	70.00	–
Adapt-DCP [39]	310	28.55	–	68.92	–
ResNet56, CIFAR10					
GAL-0.5 [38]	100	6.62	–	63.40	88.20
[28]	40	6.94	–	62.40	86.30
NISP [60]	–	6.99	–	56.39	57.40
Random Pruning	50	6.52	–	51.03	55.08
CaP [46]	–	6.78	–	50.20	–
ENC [25]	–	7.00	–	50.00	–
AMC [18]	–	8.1	–	50.00	–
Hinge [30]	300	6.31	–	50.00	48.73
KSE [34]	200	6.77	–	48.00	45.27
FPGM [19]	200	6.74	–	47.4	–
SFP [17]	300	6.65	–	47.4	–

Conclusion

- There are no clear winners among different channel importance evaluation methods.
- Random pruning is a simple, general and explainable baseline which performs well.

Our understanding

- **Pros**
  - No use of sophisticated algorithm
  - Great way to have an idea of performance of pruned model
- **Suggestions**
  - With more complex algorithms we may obtain better results

Related works

Previous work about structured pruning:

Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. ACM Journal on Emerging Technologies in Computing Systems (JETC), 13(3):1–18, 2017

Before that the research was more focused on unstructured pruning

Other related paper investigates the different channel importance metrics:

L1/L2: Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In Proceedings of International Conference on Learning Representations, 2018

GM: Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition

TE: Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In Advances in Neural Information Processing Systems, pages 164–171, 1993

KL: Jian-Hao Luo and Jianxin Wu. Neural network pruning with residual-connections and limited-data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1458–1467, 2020

ES: Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. arXiv preprint arXiv:1911.07412, 2019

Works that followed:

Using random pruning as a baseline: ThinResNet: A New Baseline for Structured Convolutional Networks Pruning,

Multi-Granularity Pruning for Model Acceleration on Mobile Devices

Tianli Zhao, Xi Sheryl Zhang, Wentao Zhu, Jiaxing Wang Sen Yang , Ji Liu, and Jian Cheng

In book: Computer Vision – ECCV 2022, 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI (pp.484-501)

Study of the paper

Method	Type	Latency	Accuracy
Fast	Weight Pruning	88.94 ms	72.0%
DMCP	Channel Pruning	82.95 ms	72.4%
Fast	Weight Pruning	35.89 ms	65.2%
DMCP	Channel Pruning	33.50 ms	62.7%

Issue: The two mainstream pruning methods, channel and weight pruning, have varying impacts on latency and accuracy.

Objective: Can a refined network pruning method be designed to achieve an improved latency-accuracy trade-off?

Problem formulation

Model $\mathcal{A}$ with L layers, we denote the umber of channels of each layer by $\mathcal{C}_{\mathcal{A}} = \{\mathcal{C}_{\mathcal{A}}^{(l)}\}_{l=1}^{L}$, and the weight sparsity of each layer by $\mathcal{S}_{\mathcal{A}} = \{\mathcal{S}_{\mathcal{A}}^{(l)}\}_{l=1}^{L}$. In this way, each sub-network $\mathcal{A}$ can be represented by a pair of vectors: $\mathcal{A} = \{\mathcal{C}_{\mathcal{A}}, \mathcal{S}_{\mathcal{A}}\}$. Our goal is to accelerate the inference of networks by applying channel pruning and weight pruning simultaneously, while at the same time minimizing the accuracy loss:

$$\mathcal{A}^* = \arg\min_{\mathcal{A}} \{\mathcal{T}(C_{\mathcal{A}}, S_{\mathcal{A}}), \mathcal{E}(C_{\mathcal{A}}, S_{\mathcal{A}})\},$$

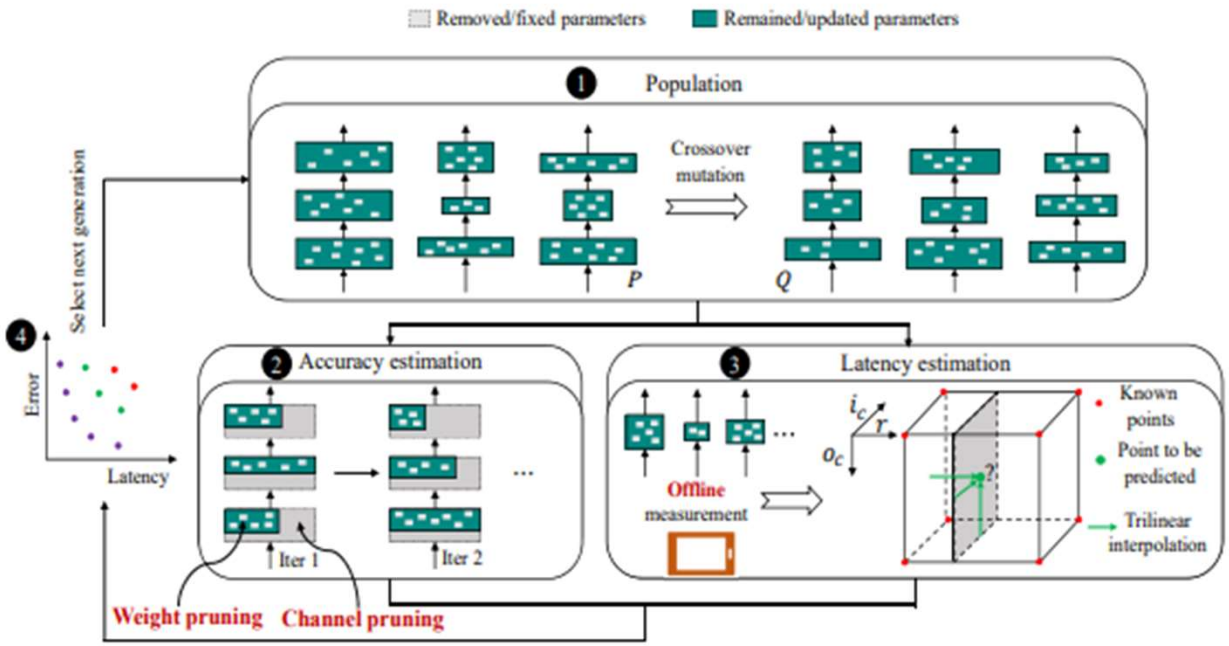
Where $\mathcal{T}(\mathcal{C}_{\mathcal{A}}, \mathcal{S}_{\mathcal{A}})$ and $\mathcal{E}(\mathcal{C}_{\mathcal{A}}, \mathcal{S}_{\mathcal{A}})$ denote the inference latency and task specific error of the model, respectively.

Issues:

1. It is difficult to find the optimal balance between channel pruning and weight pruning by hand.

2. There are more than one objective (the latency and accuracy) to be optimized, yielding a multi-objective optimization (MOO) problem.

Joint Channel and Weight Pruning



By jointly exploring the optimal number of channels and layer-wise weight sparsity simultaneously, JCW automatically finds a balanced trade-off between channel and weight pruning for various latency budgets → Pareto-optimization evolutionary algorithm

High level procedure

The search of the best architecture is performed using evolutionary algorithm

Randomly mutate and recombine C and S where:

→C = [number of channel to remove from a layer]

→S = [probability to prune weight from a layer]

Ex: C=[3, 4, 5, 2, 6, …], S=[0.4, 0, 0.2, 0.3, 0, …]

Functioning:

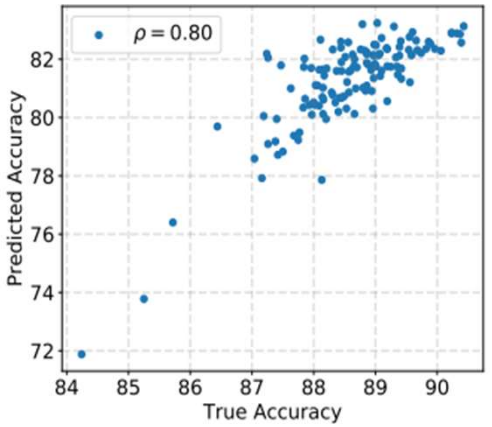
Set of n well-performing models P $= \{(C_i, S_i)\}_{i=1}^{n}$ with different number of channels C and sparsity S.

for #generations:

1. A new set of models $Q = \{(C_{ne\ \ i}, S_{ne\ \ i})\}_{i=1}^{n}$ are generated from P through crossover and mutation operators.
2. Estimate the accuracy (A) and latency (T) of all the models in $P \cup Q$.
3. Based on a non-dominated sorting algorithm select P for next generation.

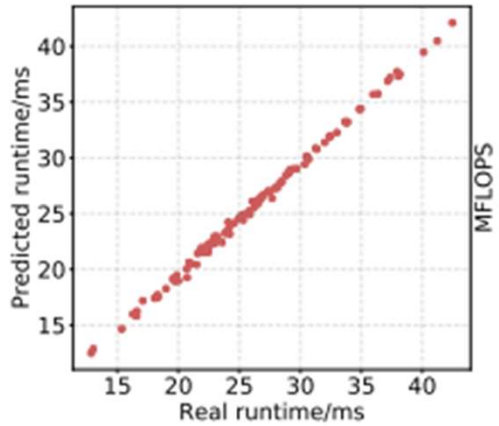
Accuracy estimation:

- Done by training a super-net with parameter sharing
- Framework only focus is on accuracy rank of a finite number of architectures
    - The super-net is reconstructed at the beginning of each evolutionary generation, containing only models to be evaluated



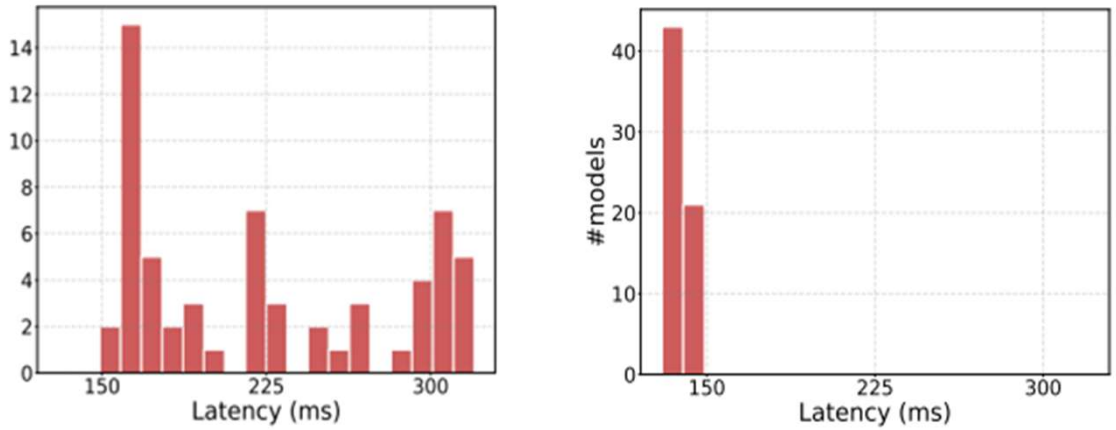
Latency Estimation:

- Done by using trilinear interpolation
- Latency of a model can be represented by the summation of the latency of each layer
- Given any input/output channels and weight sparsity, it's easy to approximate the latency through trilinear interpolation of the 3-D array



Uniform non-dominated sorting selection:

- This selection process generates diverse architectures with various latencies while searching for well-performed models



Selection with non-dominating sorting (left) and without non-dominating sorting (right)

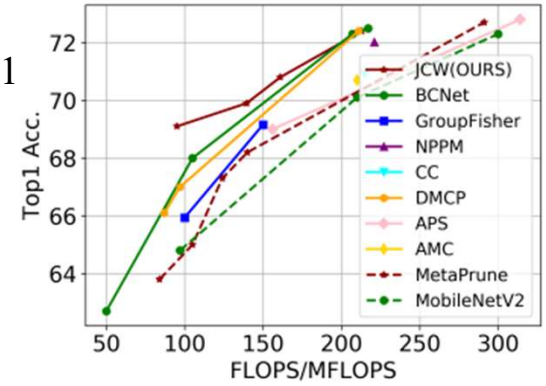
Experiments

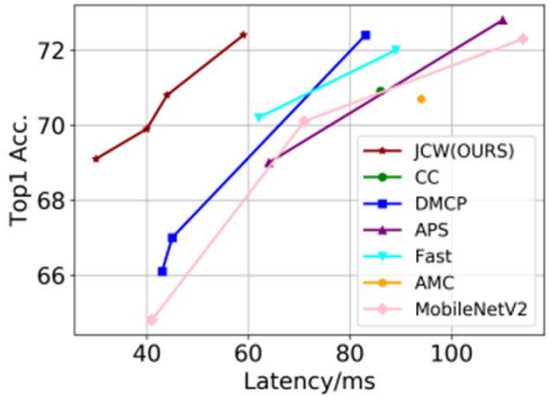
- Conducted on ResNet18, MobileNetV1 and MobileNetV2 using the ImageNet dataset
- Evolutionary search with a population size (n) of 64 and 128 search steps (generations).
- Supernet for accuracy estimation is trained for 30 epochs with batch size of 256
- Supernet training on a subset of ImageNet, followed by re-training on the entire dataset after the search stage
- Implemented sparse convolution with improvements for weight pruning, grouping parameters for efficient data reuse.
- Latency measured on:
    - ARM Cortex-A72 CPU (1 core)
    - ARM Cortex-A53 CPU (1 core/4 core)

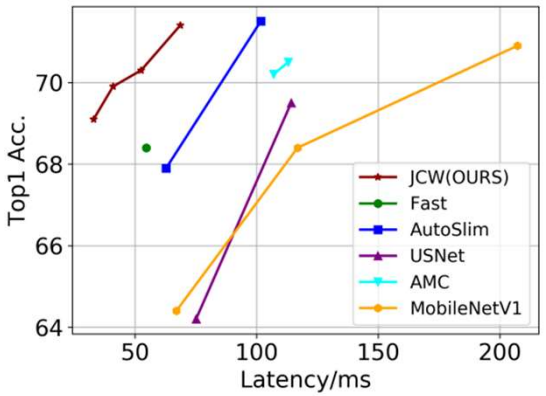
Results

FLOPS | Real architecture latency (1 CORE) | Real architecture latency (4 CORE)

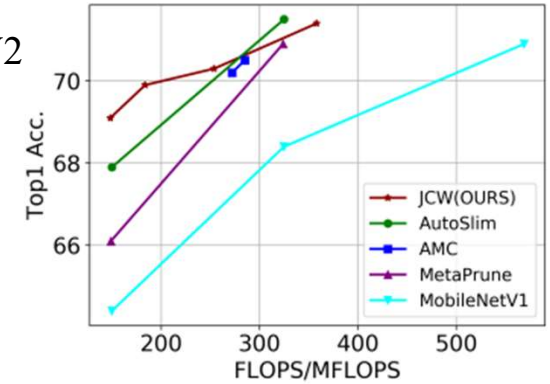
MobileNetV1

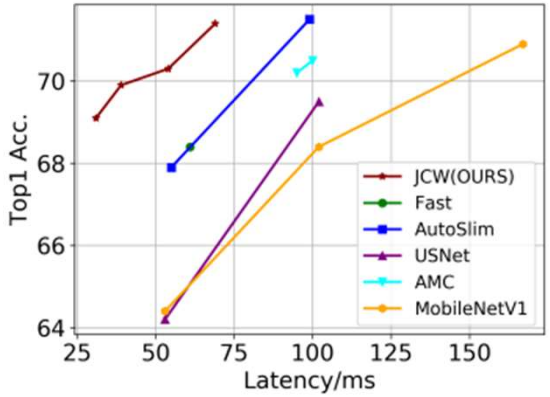


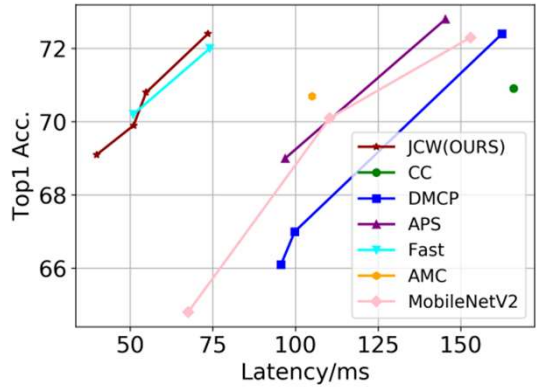




MobileNetV2







Conclusions

The methods is capable of attaining same accuracy of the majority of other pruning method while reducing more the computational cost.

Our understanding

- **Pros:**
  - This pruning method achieves a better proportion between the channel and weight pruning getting both benefits of them.

Related works

MOO in CNN compression:

Dong, J.D., Cheng, A.C., Juan, D.C., Wei, W., Sun, M.: Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 517–531 (2018)

Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., Banzhaf, W.: Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 419– 427 (2019)

Weight pruning:

Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. In: Advances in Neural Information Processing Systems

LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Advances in neural information processing systems. pp. 598–605 (1990)

There are different studies about using EA in network search, one of the earliest:

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002).