# Channel Pruning via Automatic Structure Search

Charles Kuo

# About the paper

- Published in IJCAI in 2020
- Cited by 129 as of today

## Channel Pruning via Automatic Structure Search

**Mingbao Lin[1], Rongrong Ji[1*], Yuxin Zhang[1], Baochang Zhang[2], Yongjian Wu[3], Yonghong Tian[4]**

[1]Media Analytics and Computing Laboratory, Department of Artificial Intelligence, School of Informatics, Xiamen University, China

[2]School of Automation Science and Electrical Engineering, Beihang University, China

[3]Tencent Youtu Lab, Tencent Technology (Shanghai) Co., Ltd, China

[4]School of Electronics Engineering and Computer Science, Peking University, Beijing, China

lmbxmu@stu.xmu.edu.cn, rrji@xmu.edu.cn, yxzhangxmu@163.com, bczhang@buaa.edu.cn, littlekenwu@tencent.com, yhtian@pku.edu.cn

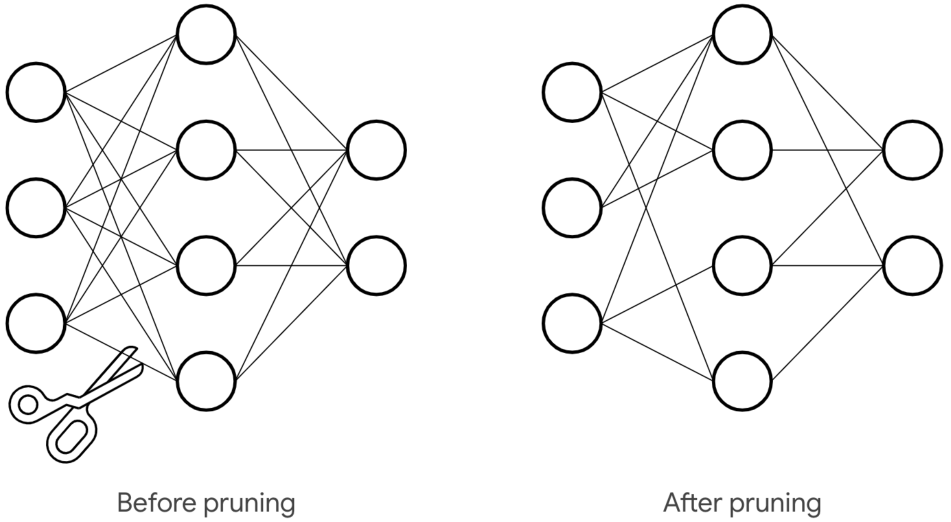
### Abstract

Channel pruning is among the predominant approaches to compress deep neural networks. To this end, most existing pruning methods focus on selecting channels (filters) by importance/optimization or regularization based on rule-of-thumb designs, which defects in sub-optimal pruning. In this paper, we propose a new channel pruning method based on artificial bee colony algorithm (ABC), dubbed as ABCPruner, which aims to efficiently find optimal pruned structure, *i.e.*, channel number in each layer, rather than selecting "important" channels as previous works did. To solve the intractably huge combinations of pruned structure for deep networks, we first propose to shrink the combinations where the preserved channels are limited to a specific space, thus the combinations of pruned structure can be significantly

CNNs [Luo *et al.*, 2017; He *et al.*, 2017; He *et al.*, 2018b; He *et al.*, 2018a; Liu *et al.*, 2019a; Wang *et al.*, 2019b].

Channel pruning targets at removing the entire channel in each layer, which is straightforward but challenging because removing channels in one layer might drastically change the input of the next layer. Most cutting-edge practice implements channel pruning by selecting channels (filters) based on rule-of-thumb designs. Existing works follow two mainstreams. The first pursues to identify the most important filter weights in a pre-trained model, which are then inherited by the pruned model as an initialization for the follow-up fine-tuning [Hu *et al.*, 2016; Li *et al.*, 2017; He *et al.*, 2017; He *et al.*, 2018a]. It usually performs layer-wise pruning and fine-tuning, or layer-wise weight reconstruction followed by a data-driven and/or iterative optimization to recover model accuracy, both of which however are time-cost. The second typically performs channel pruning based on handcrafted rules to regularize the retraining of a full model followed by pruning

# Channel Pruning




- Pruning
  - Compress model size, accelerate speed
  - Remove portion of filters in CNN, while keeping accuracy

- Weight pruning: Removes individual neurons in the filters or connections across different layers.
- Channel pruning: Structural model compression approach, remove the entire redundant filters directly.

# Motivation

- Existing channel pruning methods: prune channels (filters) based on rule-of- thumb designs. -> not automatic, defects in sub-optimal pruning
- **Propose ABCPruner**

# ABCPruner

- Find optimal pruned structure (channel number) in each layer, instead of selecting “important” channels
- Apply **artificial bee colony algorithm** to automatically find optimal pruned structure(channel number).

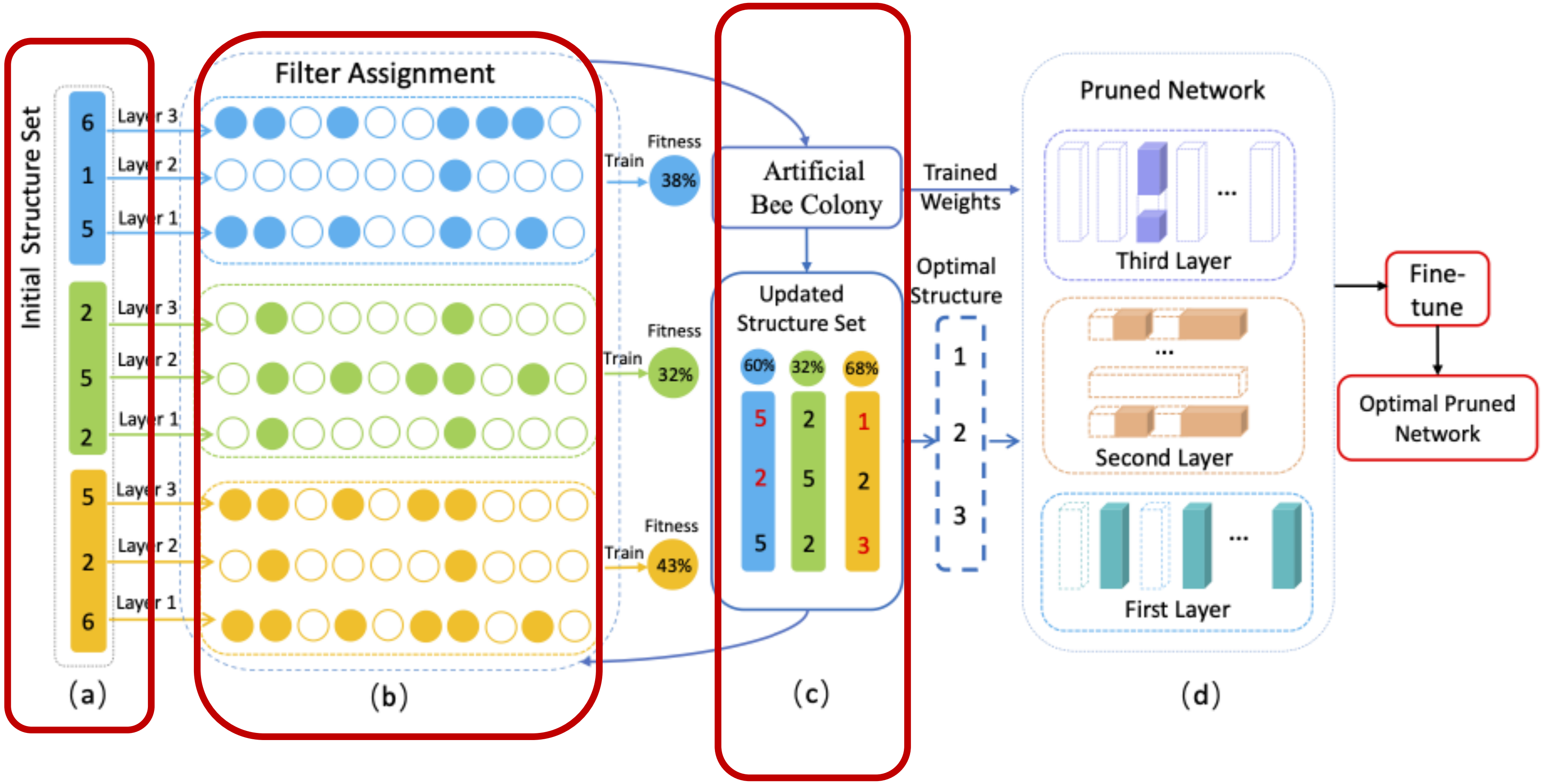
# Implementation

(a) Initialize structure set

(b) Randomly assign filter to each structure, then train and measure fitness

(c) Use ABC algorithm to update structure set and recalculate fitness.

Repeat (b) and (c) for several cycle, then find the optimal pruned structure with best fitness.



# Experiment

- Tested on VGGNet, ResNet, GoogLeNet with CIFAR-10 dataset.

| Model | Top1-acc | ↑↓ | Channel | Pruned | FLOPs | Pruned | Parameters | Pruned |
|---|---|---|---|---|---|---|---|---|
| VGGNet-16 Base | 93.02% | 0.00% | 4224 | 0.00% | 314.59M | 0.00% | 14.73M | 0.00% |
| **VGGNet-16 ABCPruner-80%** | 93.08% | 0.06%↑ | 1639 | 61.20% | 82.81M | 73.68% | 1.67M | 88.68% |
| GoogLeNet Base | 95.05% | 0.00% | 7904 | 0.00% | 1534.55M | 0.00% | 6.17M | 0.00% |
| **GoogLeNet ABCPruner-30%** | 94.84% | 0.21%↓ | 6150 | 22.19% | 513.19M | 66.56% | 2.46M | 60.14% |
| ResNet-56 Base | 93.26% | 0.00% | 2032 | 0.00% | 127.62M | 0.00% | 0.85M | 0.00% |
| **ResNet-56 ABCPruner-70%** | 93.23% | 0.03%↓ | 1482 | 27.07% | 58.54M | 54.13% | 0.39M | 54.20% |
| ResNet-110 Base | 93.50% | 0.00% | 4048 | 0.00% | 257.09M | 0.00% | 1.73M | 0.00% |
| **ResNet-110 ABCPruner-60%** | 93.58% | 0.08%↑ | 2701 | 33.28% | 89.87M | 65.04% | 0.56M | 67.41% |

# Compare with other methods

- Obtains better FLOP reduction and accuracy performance comparing with other importance-based pruning methods.
- Requires less training epochs.

| Model | FLOPs | Top1-acc | Baseline-acc | Epochs |
|---|---|---|---|---|
| ThiNet-30 | 1.10G | 68.42% | 76.01% | 244 (196 + 48) |
| **ABCPruner-30%** | 0.94G | 70.29% | 76.01% | 102 (12+90) |
| SSS-26 | 2.33G | 71.82% | 76.01% | 100 |
| GAL-0.5 | 2.33G | 71.95% | 76.01% | 150 (90 + 60) |
| GAL-0.5-joint | 1.84G | 71.80% | 76.01% | 150 (90 + 60) |
| ThiNet-50 | 1.71G | 71.01% | 76.01% | 244 (196 + 48) |
| **ABCPruner-50%** | 1.30G | 72.58% | 76.01% | 102 (12+90) |
| SSS-32 | 2.82G | 74.18% | 76.01% | 100 |
| CP | 2.73G | 72.30% | 76.01% | 206 (196 + 10) |
| **ABCPruner-100%** | 2.56G | 74.84% | 76.01% | 102 (12+90) |
| MetaPruning-0.50 | 1.03G | 69.92% | 76.01% | 160 (32 + 128) |
| **ABCPruner-30%** | 0.94G | 70.29% | 76.01% | 102 (12+90) |
| MetaPruning-0.75 | 2.26G | 72.17% | 76.01% | 160 (32 + 128) |
| **ABCPruner-50%** | 1.30G | 72.58% | 76.01% | 102 (12+90) |
| MetaPruning-0.85 | 2.92G | 74.49% | 76.01% | 160 (32 + 128) |
| **ABCPruner-100%** | 2.56G | 74.84% | 76.01% | 102 (12+90) |

FLOPS: Floating point operations (measure of computer performance, less-> better)

# Conclusion

- ABCPruner
  - Find optimal channel number in each layer, instead of selecting “important” channels
  - Apply **artificial bee colony algorithm** to automatically find optimal channel number.