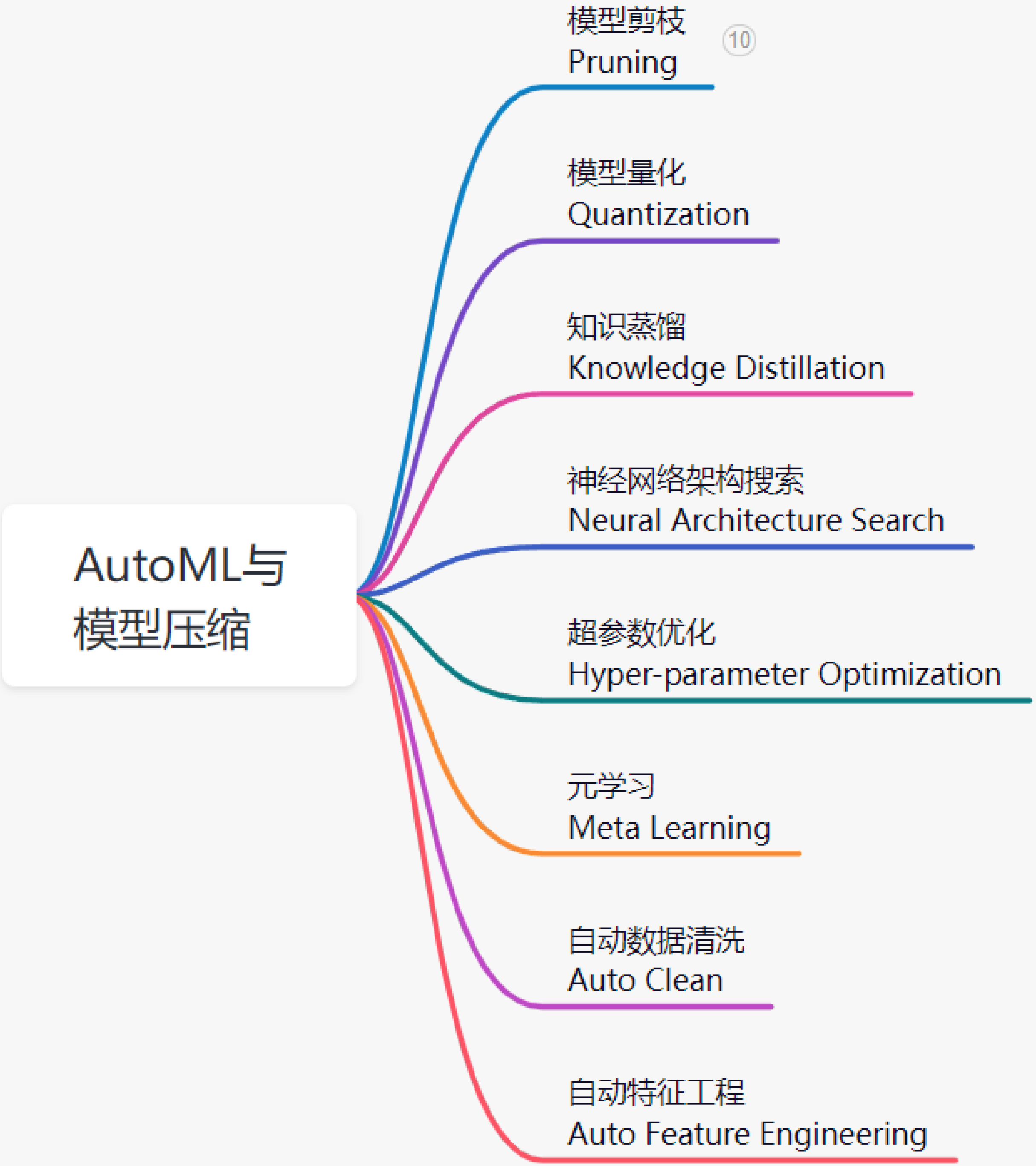
# 神经网络剪枝方法概述
# 与混合剪枝方法介绍

报告人：吴宇航
报告时间：2021.12.1

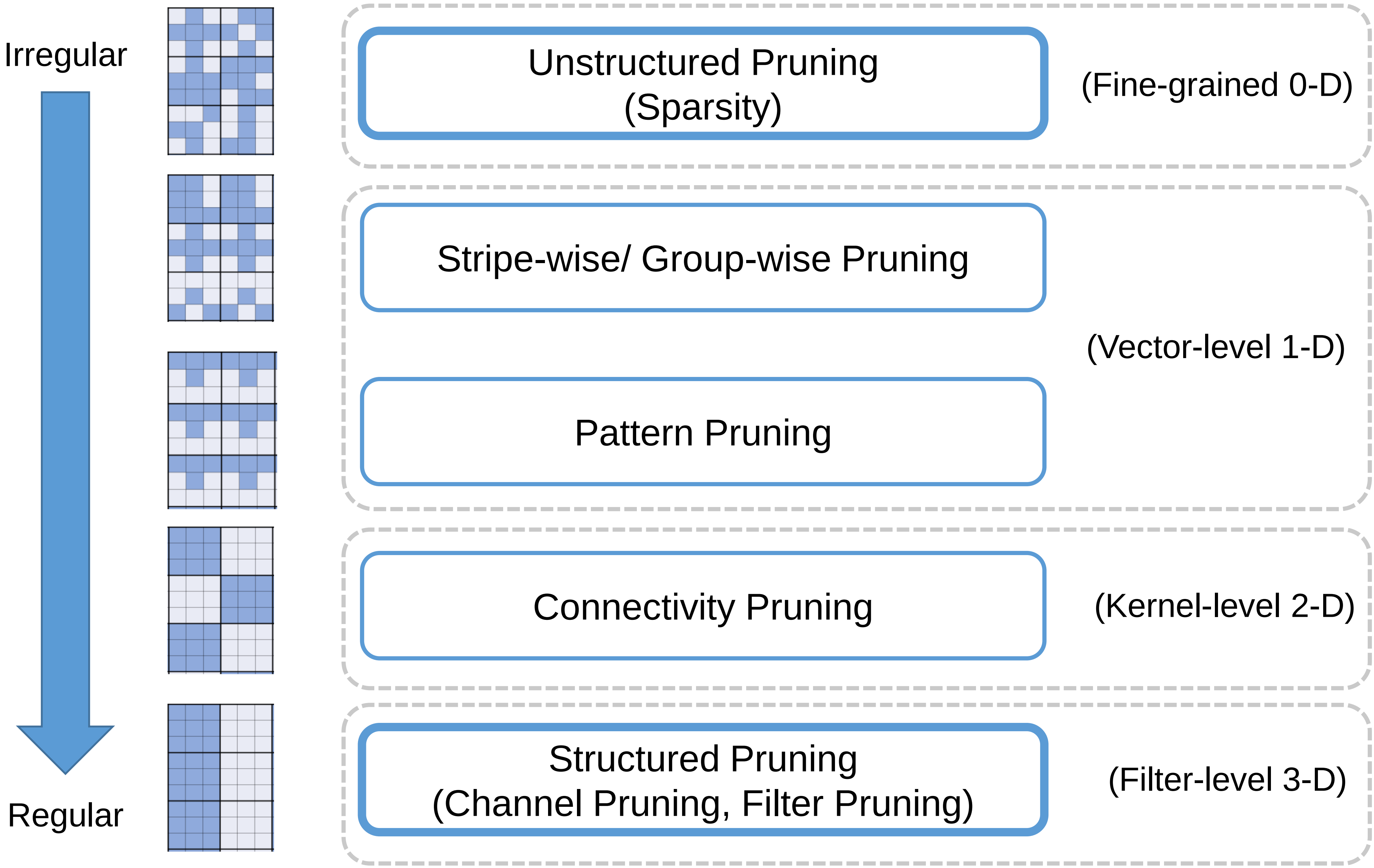
# 前言

深度神经网络被广泛应用于计算机视觉、自然语言处理、推荐搜索等领域。其模型内部大量有待训练的权重占用了大量的内存空间，给深度神经网络在一些嵌入式平台、移动端上的部署带来了很大的困难。此外，庞大的计算量也能耗巨大，计算成本昂贵。在模型轻量化需求催化下，模型压缩技术应运而生。其主要目标就是在尽可能不牺牲模型精度（甚至在一些场景能提升精度）的前提下，减小模型的内存与算力消耗。



# 目录 Contents

# 剪枝粒度


Irregular
Regular
Unstructured Pruning
(Sparsity)
(Fine-grained 0-D)
Stripe-wise/ Group-wise Pruning
Pattern Pruning
(Vector-level 1-D)
Connectivity Pruning
(Kernel-level 2-D)
Structured Pruning
(Channel Pruning, Filter Pruning)
(Filter-level 3-D)

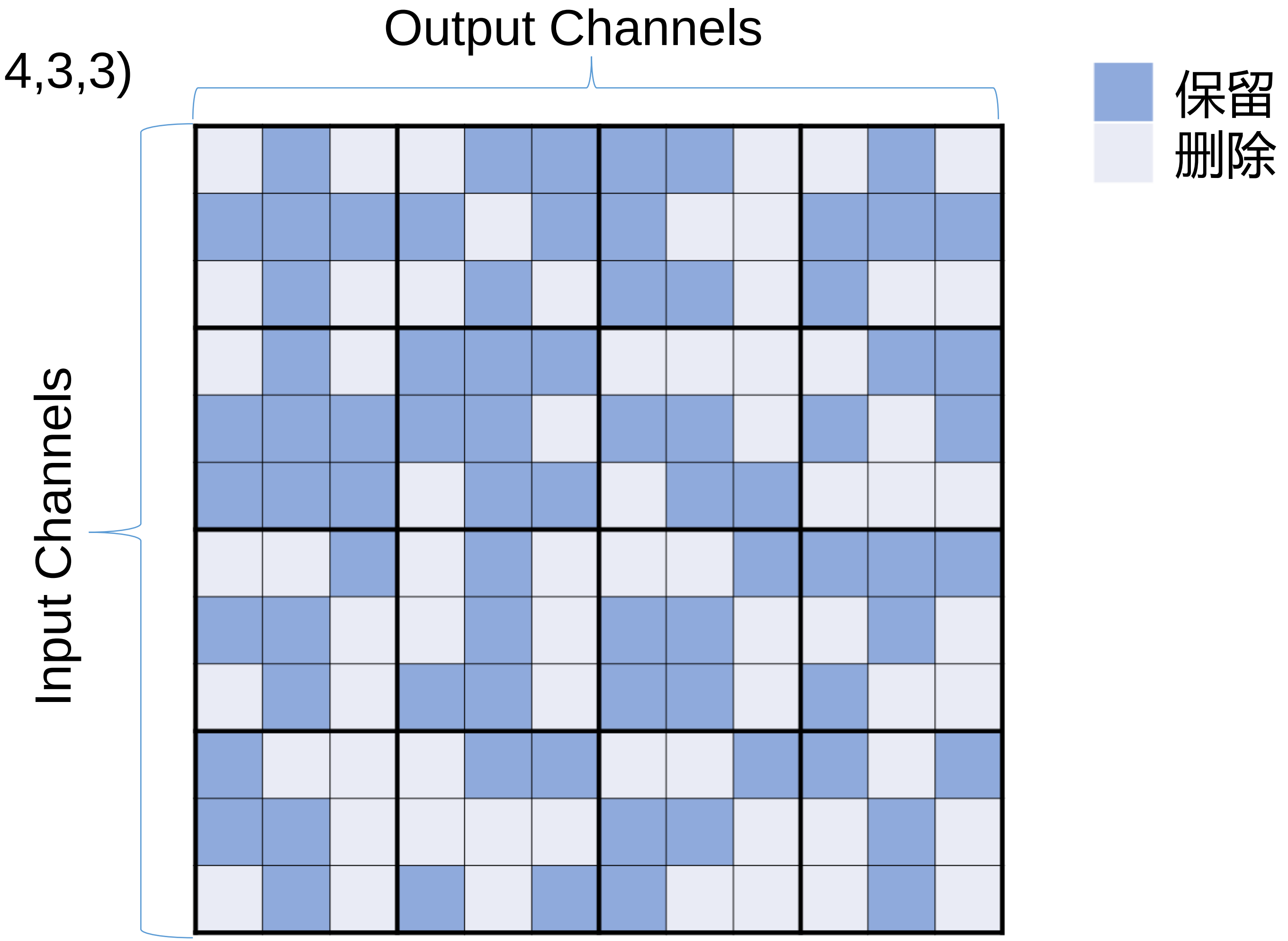
# 剪枝粒度 Unstructured Pruning (Sparsity)

3*3 Conv:
(C_out, C_in, k_H, k_W)= (4,4,3,3)

剪枝粒度为单个权重数值

每个卷积核里的某些权重置为0，且数量不定，只有每层的稀疏率，

**不容易进行硬件加速，需要定制硬件支持。**

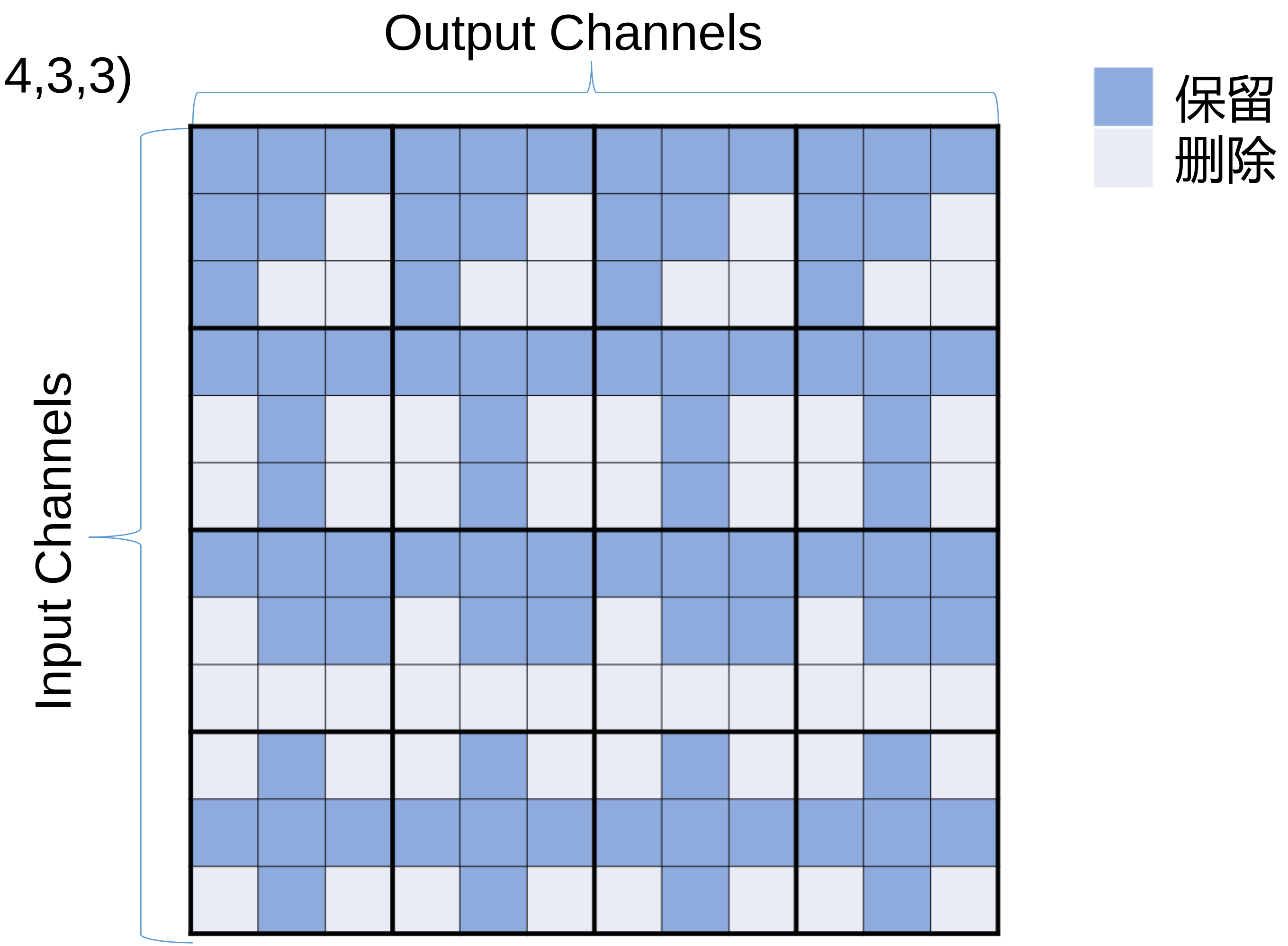


# 剪枝粒度 Stripe-wise/ Group-wise Pruning

3*3 Conv:
(C_out, C_in, k_H, k_W)= (4,4,3,3)

剪枝粒度为卷积参数张量中伸展方向为输入或输出维度的一些向量。

**需要特殊设计的硬件，例如 NVIDIA A100。**



# 剪枝粒度 Stripe-wise/ Group-wise Pruning

For conv layer (N,C,H,W)

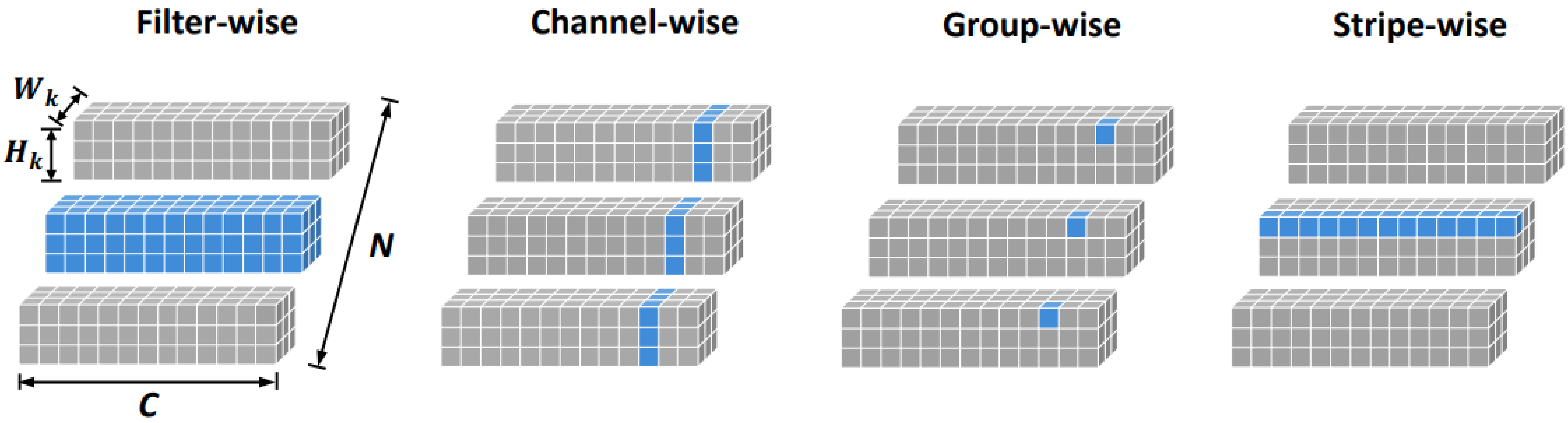



Figure 2: The visualization of different types of pruning.

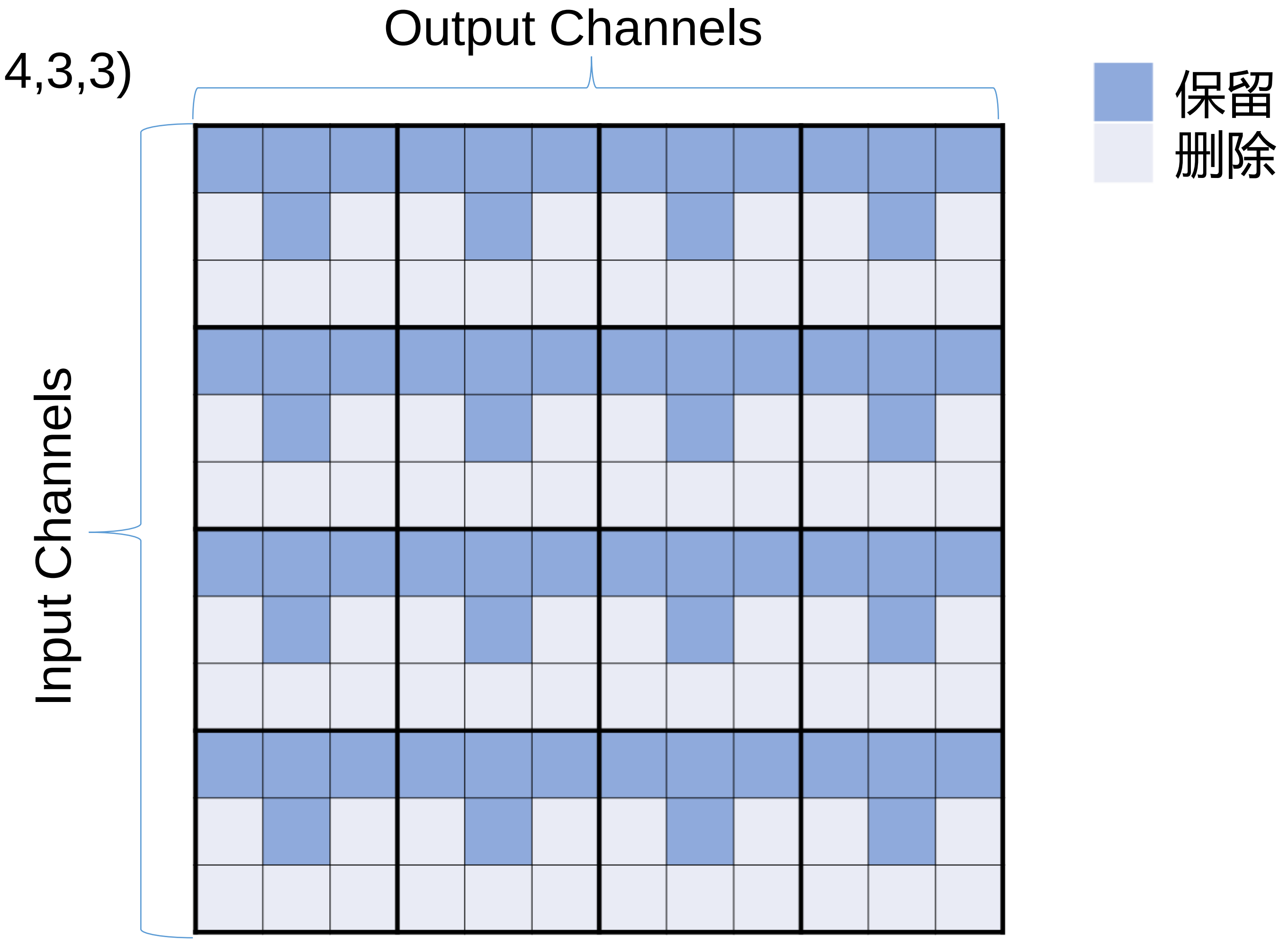
Meng, Fanxu, et al. "Pruning Filter in Filter." NeurIPS. 2020.

# 剪枝粒度 Pattern Pruning

3*3 Conv:
(C_out, C_in, k_H, k_W)= (4,4,3,3)

剪枝粒度为一组固定模式（位置）的权重，卷积核固定位置的权重置为0，一般一个模式的适用范围为一层或整个模型。

**有论文(Patdnn)提出了可以进行模式剪枝硬件加速的计算框架。**

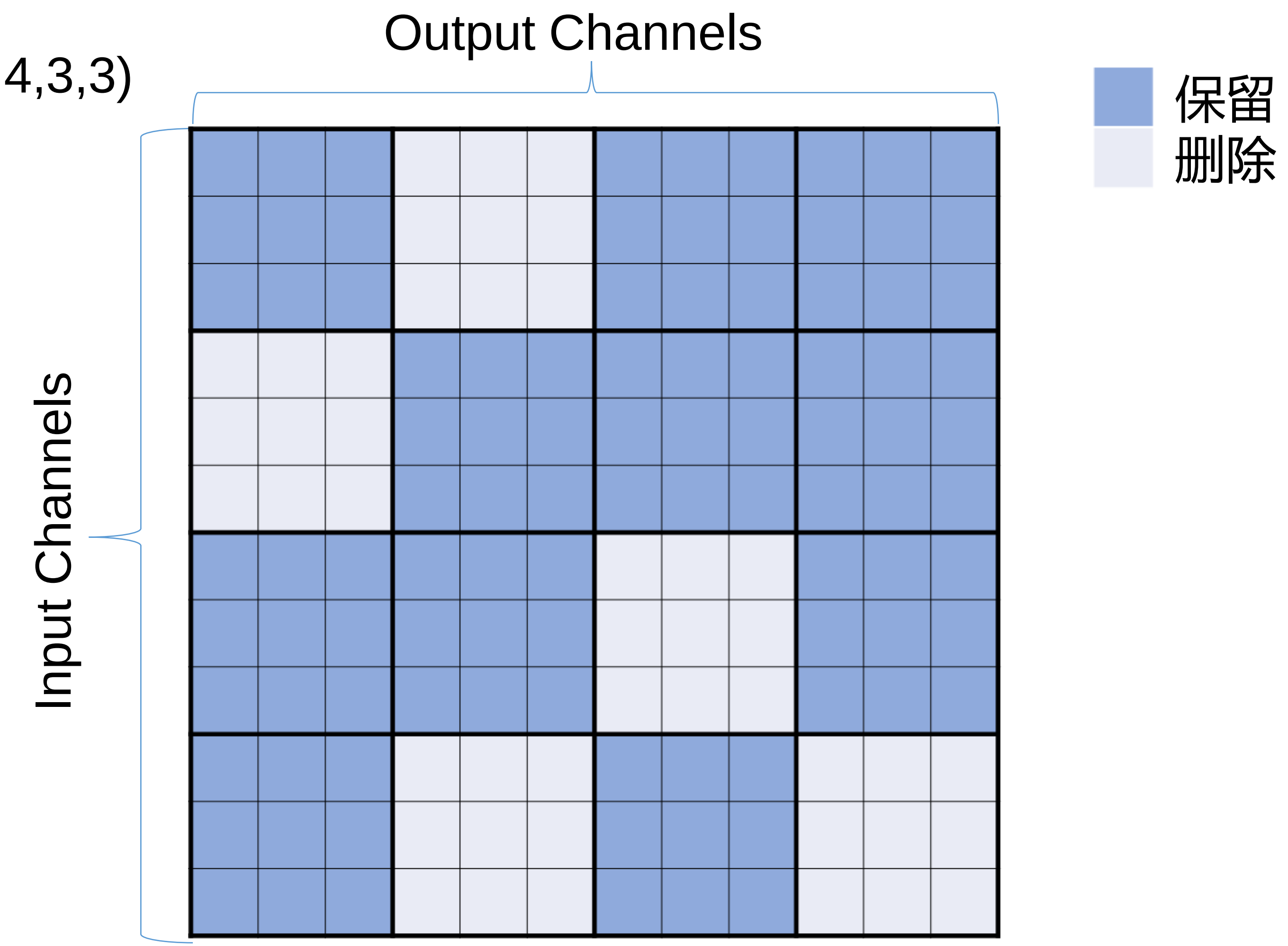



*Niu, Wei, et al. "Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning." Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. 2020.*

# 剪枝粒度 Connectivity Pruning

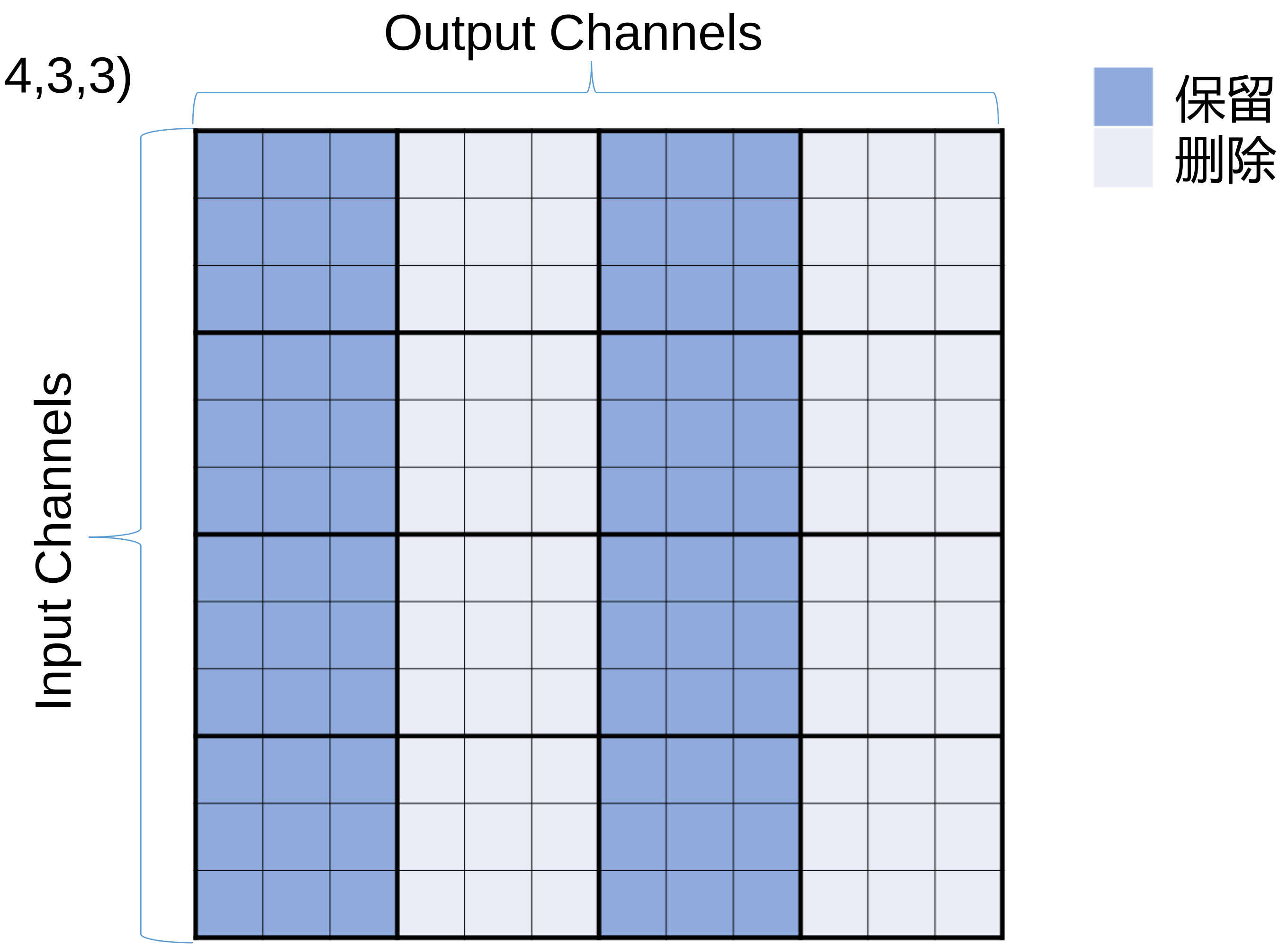
3*3 Conv:
(C_out, C_in, k_H, k_W)= (4,4,3,3)

剪枝粒度为卷积核，是不定位置的卷积核，被剪枝的卷积核所有权重置为0。**相对于稀疏来说比较容易进行硬件加速，仍需要编译器的专门优化才能加速。**



# 剪枝粒度 Structured Pruning (Channel/ Filter Pruning)

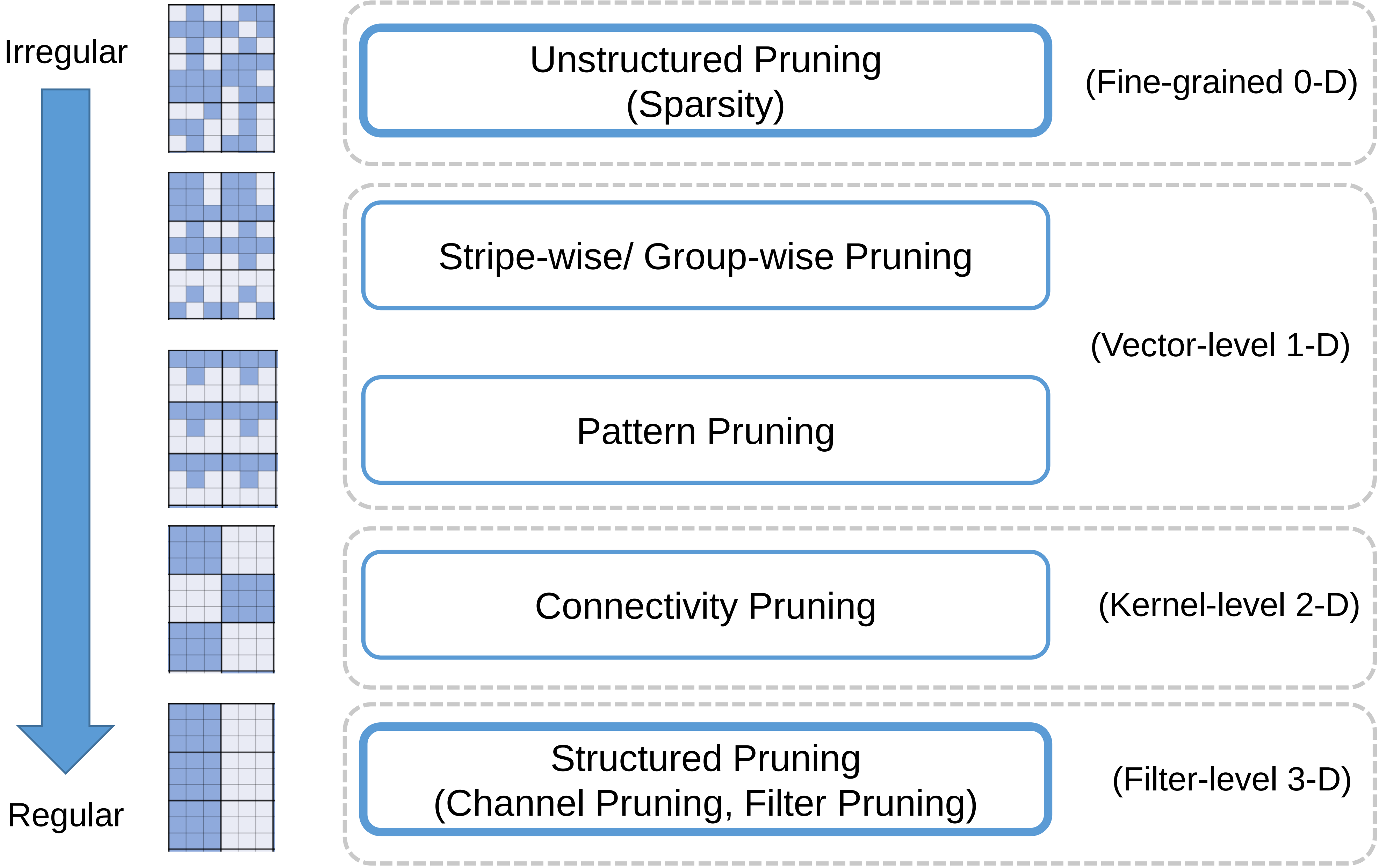
3*3 Conv:
(C_out, C_in, k_H, k_W)= (4,4,3,3)




剪枝粒度为一组通道的卷积核，相当于直接改变了模型的宽度。

**不需要任何特殊改造就可以有加速效果，在CPU上效果更显著。**

# 剪枝粒度


Irregular
Regular
Unstructured Pruning
(Sparsity)
(Fine-grained 0-D)
Stripe-wise/ Group-wise Pruning
Pattern Pruning
(Vector-level 1-D)
Connectivity Pruning
(Kernel-level 2-D)
Structured Pruning
(Channel Pruning, Filter Pruning)
(Filter-level 3-D)

# 目录 Contents

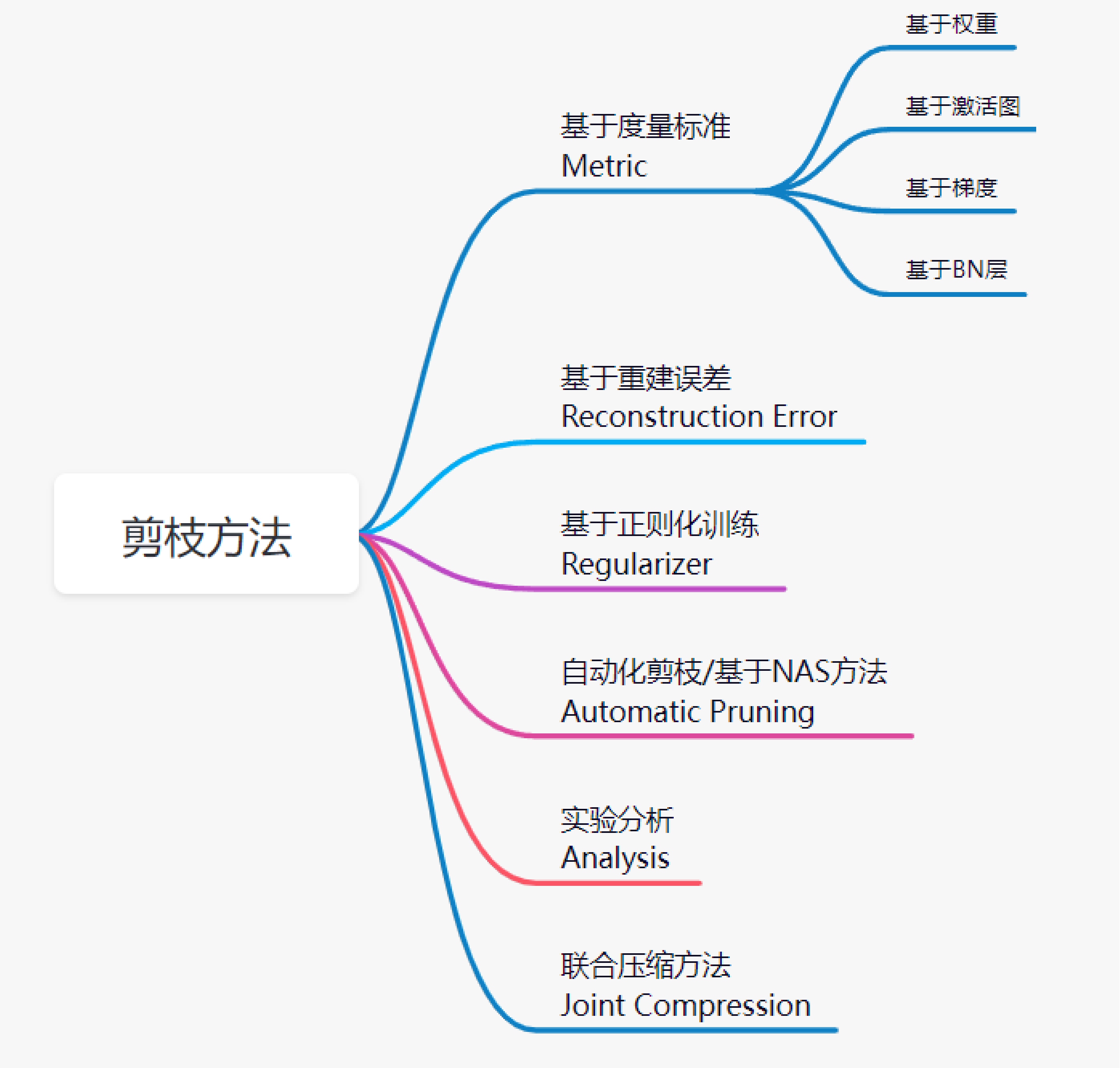
# 剪枝方法

模型剪枝：

- 给出剪枝单元的重要性
- 按一定比例把不重要的单元剪掉
- 得到满足约束的小模型

如何给出重要性？两方面：

1. 模型本身结构
2. 模型的训练/使用流程



# 剪枝方法 基于度量标准 – 权重

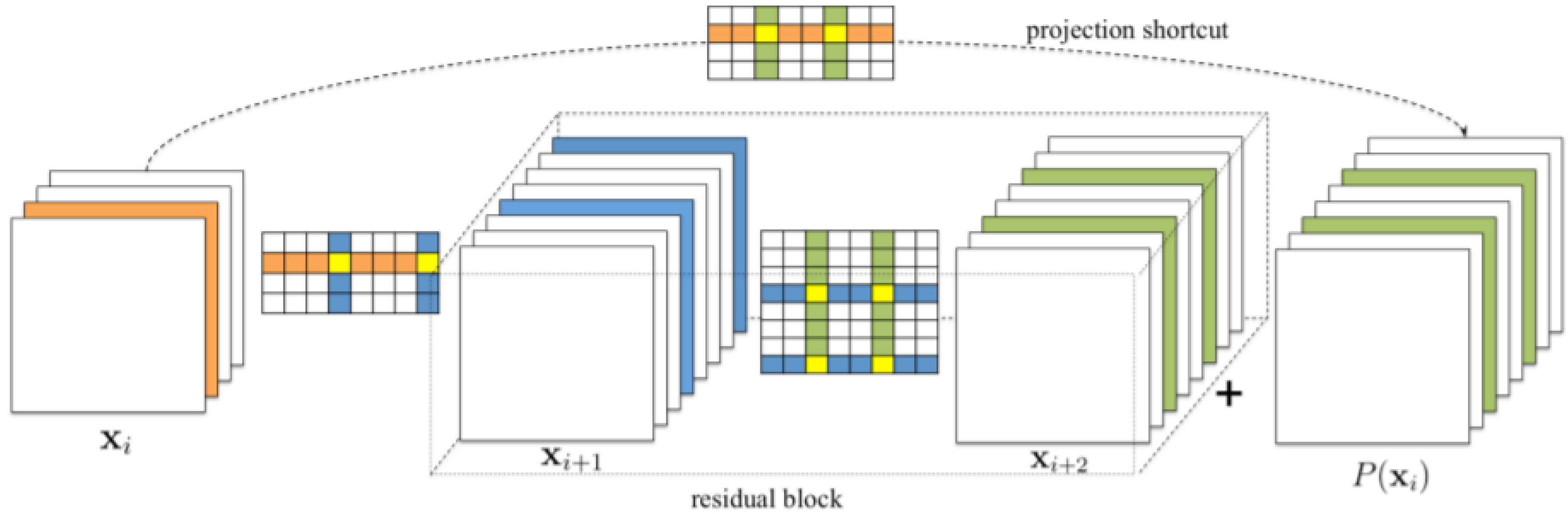
**经典方法：L1剪枝**　Pruning Filters for Efficient ConvNets  ICLR 2017

剪枝粒度：Channel pruning

剪枝过程：

- 计算 Filter 中所有权值的绝对值（L1范数）之和
- 根据求和大小排列 Filter
- 删除数值较小的 Filter （权重数值越小，代表权重的重要性越弱）
- 对删除之后的 Filter 重新组合，生成新Filter矩阵

确立复杂结构剪枝原则：
对当前操作在计算图中的
**后继节点的所有前驱节点，**
按照同一个剪枝Recipt进行操作

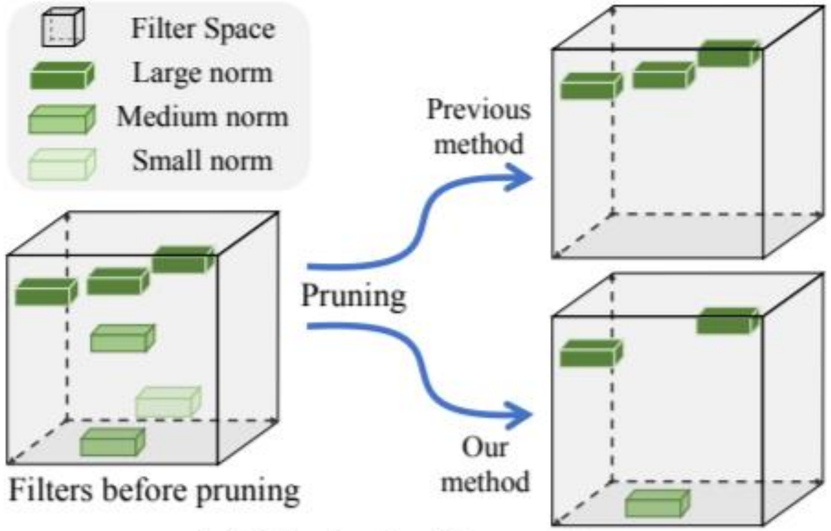


# 剪枝方法 基于度量标准 – 权重

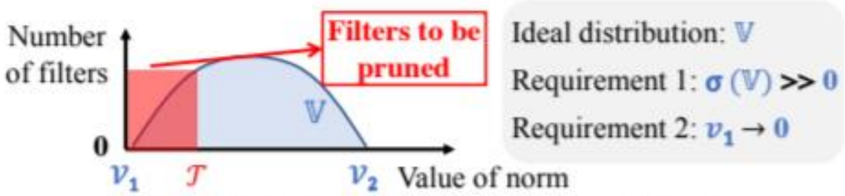
**经典方法：FPGM 几何中心剪枝**
CVPR 2019

Filter Pruning via Geometric Median for Deep Convolutional Neural Network Acceleration

剪枝粒度：Channel pruning




(a) Criterion for filter pruning




(b) Requirements for norm-based criterion

Motivation: 基于范数进行剪枝的方法，会遇到参数分布与先验分布差异大的情况，

Metric: 滤波器与几何中心的距离

**Algorithm 1** Algorithm Description of FPGM

**Input:** training data: $\mathbf{X}$.
**Given**: pruning rate $P_i$
**Initialize**: model parameter $\mathbf{W} = \{\mathbf{W}^{(i)}, 0 \le i \le L\}$
**for** $epoch = 1$; $epoch \le epoch_{max}$; $epoch++$ **do**
    Update the model parameter $\mathbf{W}$ based on $\mathbf{X}$
    **for** $i = 1$; $i \le L$; $i++$ **do**
        Find $N_{i+1}P_i$ filters that satisfy Equation 6
        Zeroize selected filters
    **end for**
**end for**
Obtain the compact model $\mathbf{W}^*$ from $\mathbf{W}$
**Output:** The compact model and its parameters $\mathbf{W}^*$

# 剪枝方法 基于重建误差

**经典方法：ThiNet**

剪枝粒度：Channel pruning

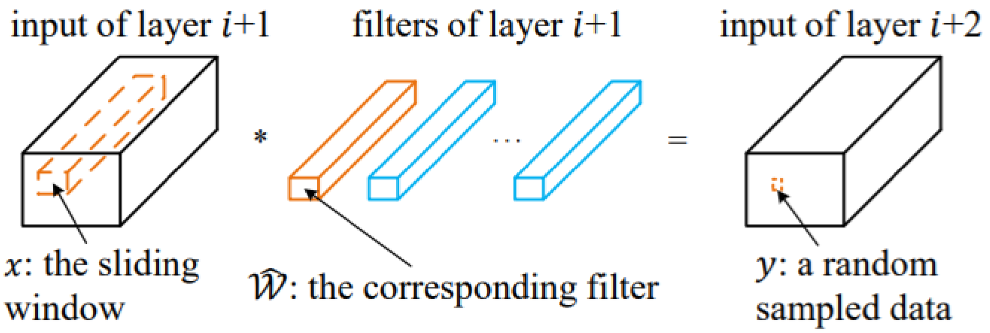
ThiNet:A Filter Level Pruning Method for Deep Neural Network Compression ICCV2017

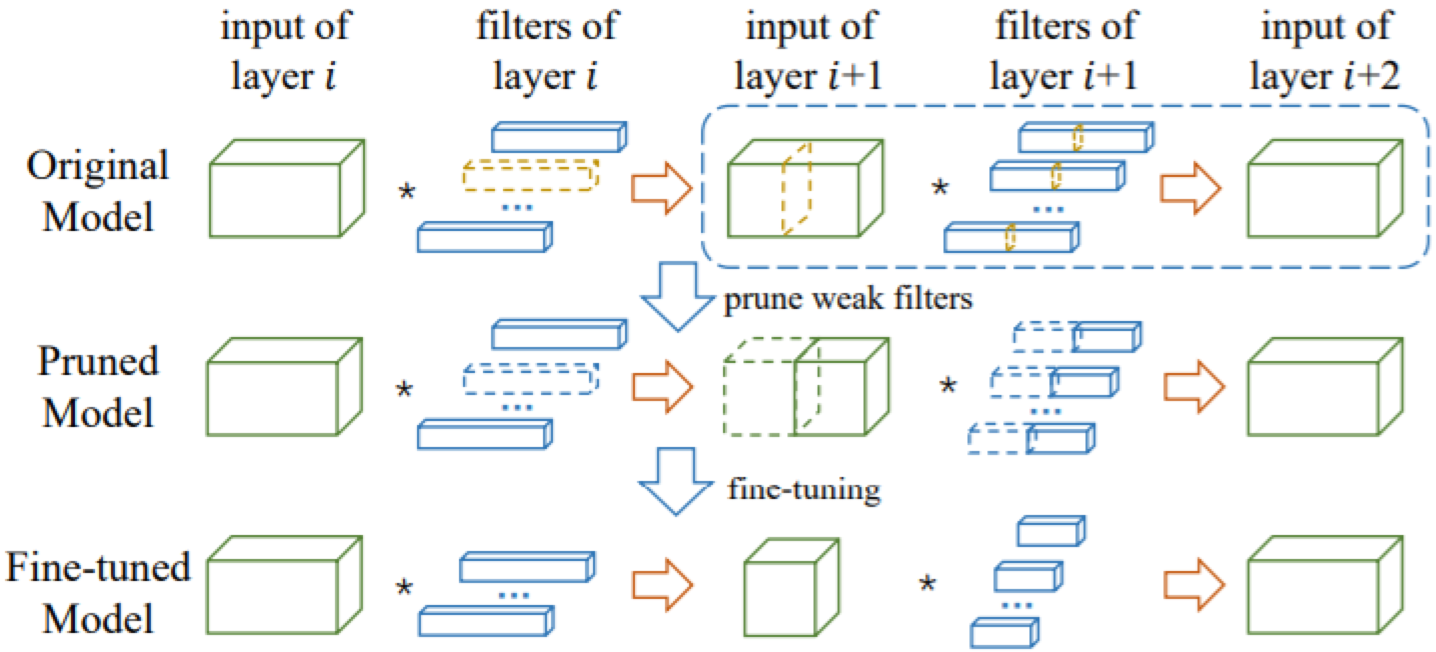
卷积可表示为:

$$y = \sum_{c=1}^{C} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \widehat{\mathcal{W}}_{c,k_1,k_2} \times x_{c,k_1,k_2} + b.$$

优化目标:

$$\underset{S}{\arg\min} \sum_{i=1}^{m} \left( \hat{y}_i - \sum_{j \in S} \hat{\mathbf{x}}_{i,j} \right)^2$$
$$\text{s.t.} \quad |S| = C \times r, \quad S \subset \{1, 2, \ldots, C\}.$$




**Algorithm 1** A greedy algorithm for minimizing Eq. 6

**Input:** Training set $\{(\hat{\mathbf{x}}_i, \hat{y}_i)\}$, and compression rate $r$

**Output:** The subset of removed channels: $T$

```
T ← ∅; I ← {1, 2, ..., C};
while |T| < C × (1 − r) do
    min_value ← +∞;
    for each item i ∈ I do
        tmpT ← T ∪ {i};
        compute value from Eq. 6 using tmpT;
        if value < min_value then
            min_value ← value; min_i ← i;
        end if
    end for
    move min_i from I into T;
end while
```

# 剪枝方法 自动化剪枝

**经典方法：AMC**

剪枝粒度：Channel pruning

AMC: AutoML for Model Compression and Acceleration on Mobile Devices ECCV2018

强化学习结构：

- Actor-Critic

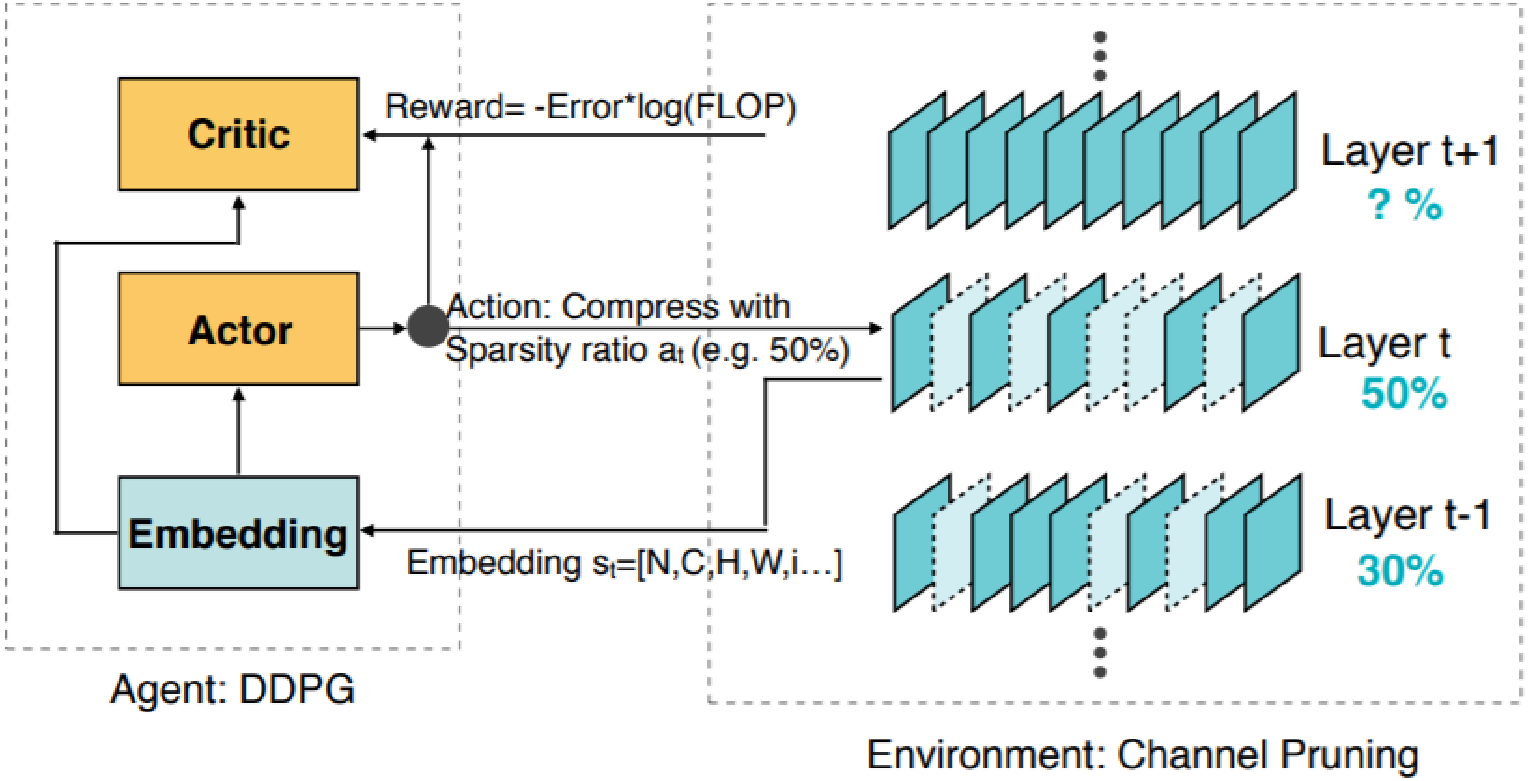
奖励函数设计：

- R=-Error*log(FLOP)
- 在精度损失时惩罚
- 在模型缩小和加速时奖励

梯度更新：DDPG

状态空间：

$$(t, n, c, h, w, stride, k, FLOPs[t], reduced, rest, a_{t-1})$$



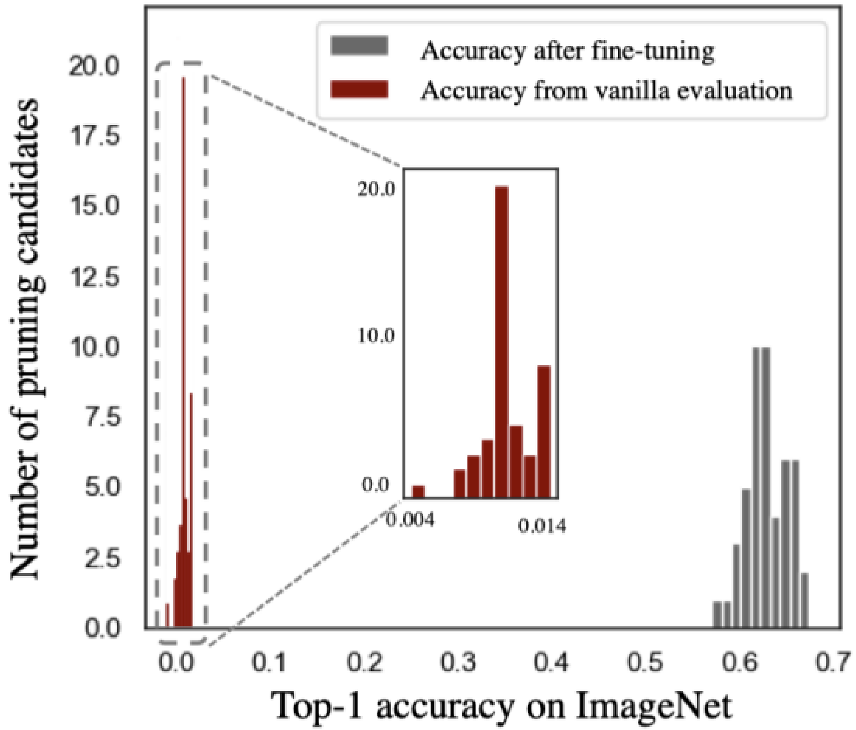
# 剪枝方法 NAS方法

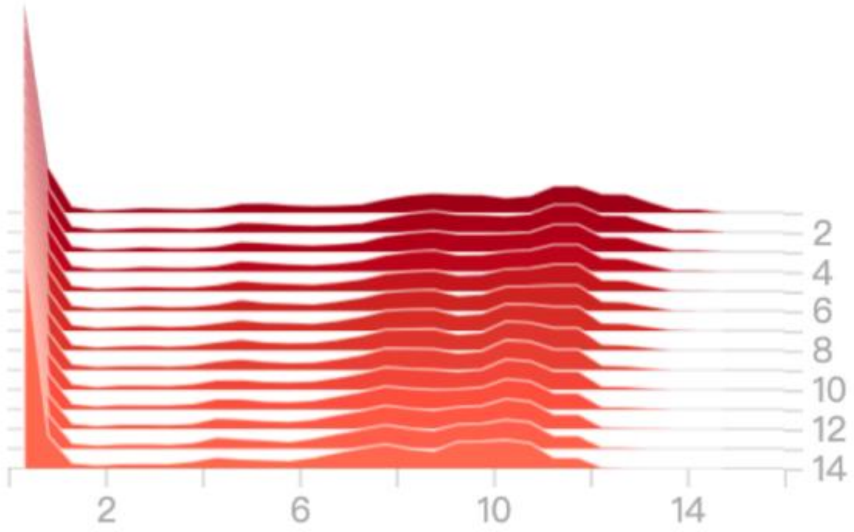
**SOTA方法：EagleEye**

剪枝粒度：Channel pruning

EagleEye: Fast Sub-net Evaluation for Efficient Neural Network Pruning ECCV 2020

Motivation: 通道剪枝后的模型，准确度下降剧烈，如下图，而Finetune后精度快速回升，但Finetune开销大，能否找到一种方法快速对剪枝后的子模型进行**模型表现排序的评估**




随着Finetune的进行，权重L1范数分布变化

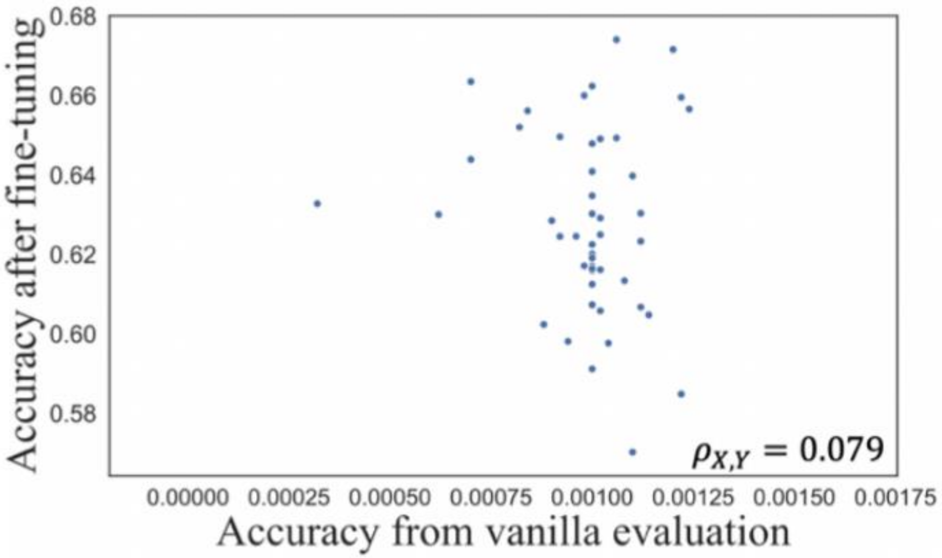
# 剪枝方法 NAS方法

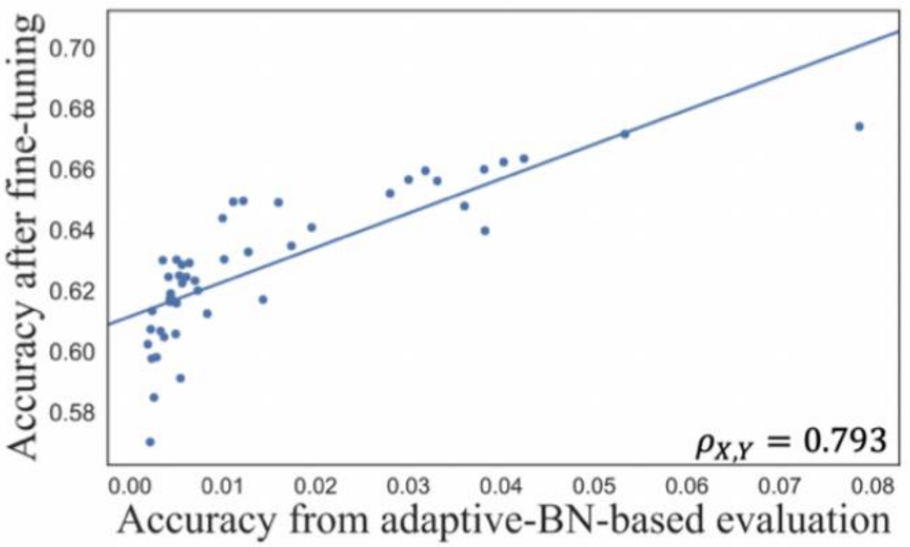
## SOTA方法：EagleEye

EagleEye: Fast Sub-net Evaluation for Efficient Neural Network Pruning ECCV 2020

提出Adaptive-BN：

剪枝后子模型不进行finetune，而是**通过几个batch的数据更新BN的参数。**




原BN层:

$$y = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta,$$

统计量的更新:

$$\mu_{\mathcal{B}} = E[x_{\mathcal{B}}] = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \sigma_{\mathcal{B}}^2 = Var[x_{\mathcal{B}}] = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_{\mathcal{B}})^2.$$

$$\mu_t = m\mu_{t-1} + (1-m)\mu_{\mathcal{B}}, \quad \sigma_t^2 = m\sigma_{t-1}^2 + (1-m)\sigma_{\mathcal{B}}^2,$$

# 剪枝方法 NAS方法

## SOTA方法：EagleEye

EagleEye: Fast Sub-net Evaluation for Efficient Neural Network Pruning ECCV 2020

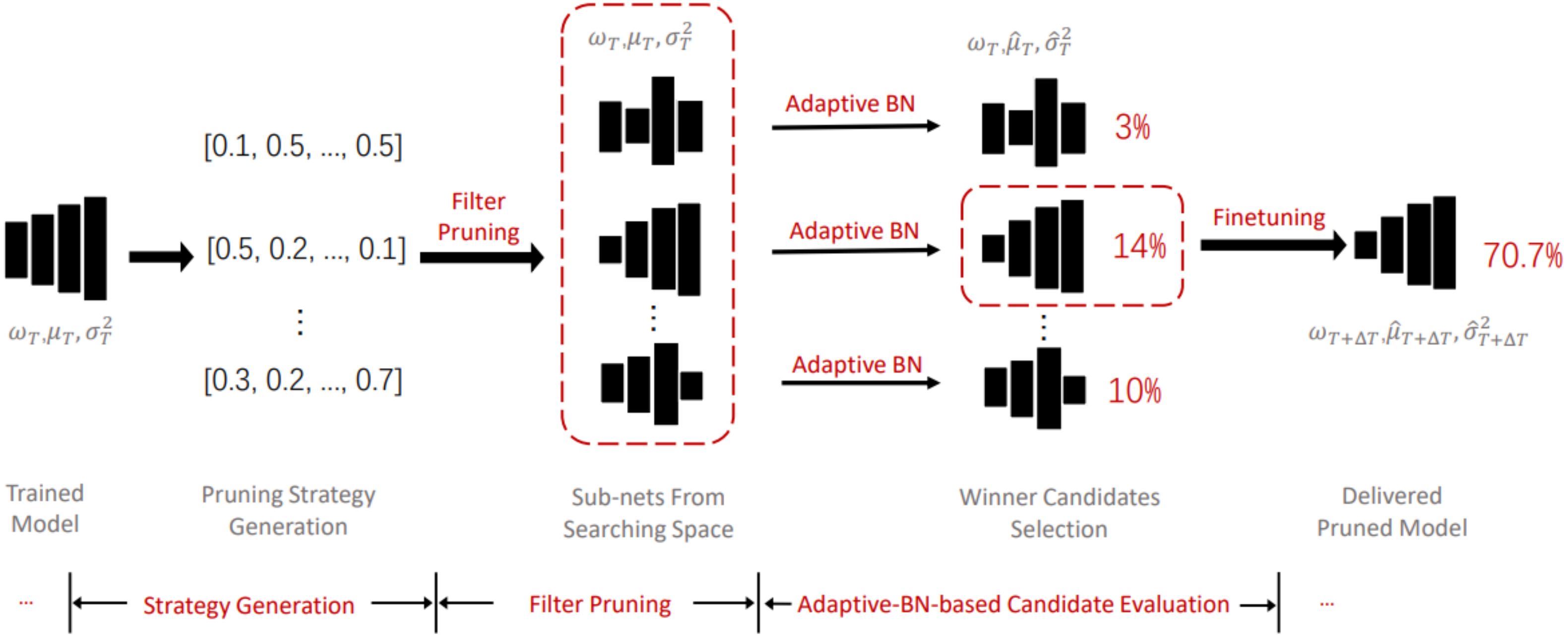



**Fig. 6.** Workflow of the EagleEye Pruning Algorithm

| Method | FLOPs | Top1-Acc |
|---|---|---|
| 0.75 × MobileNetV1 [9] | 325M | 68.4% |
| AMC [7] | 285M | 70.5% |
| NetAdapt [26] | 284M | 69.1% |
| Meta-Pruning [20] | 281M | 70.6% |
| **EagleEye** | 284M | **70.9%** |

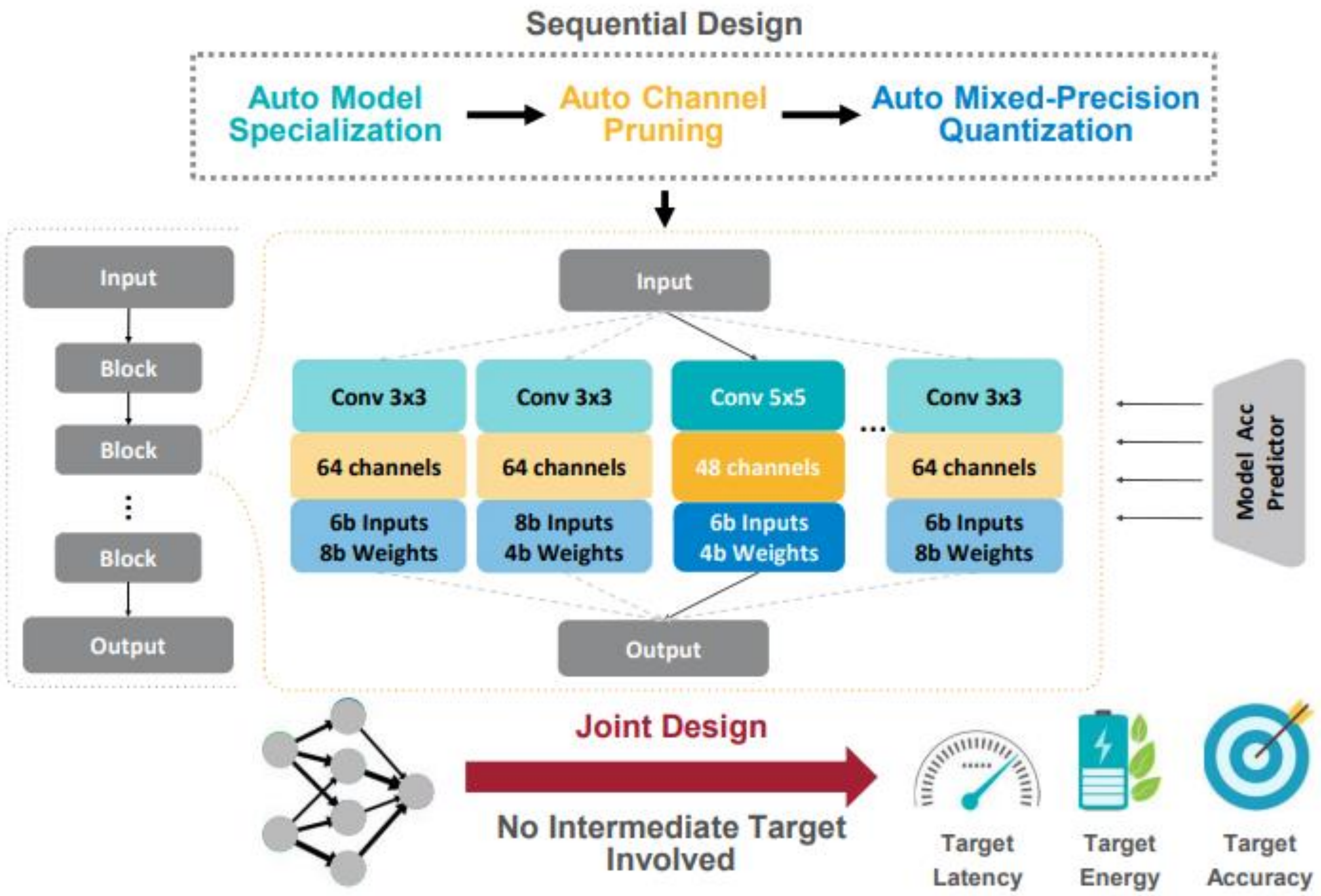
# 剪枝方法 联合压缩方法

## SOTA方法：APQ

剪枝粒度：Channel pruning

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

模型的部署分为 模型结构设计（Architecture），剪枝（Pruning），量化（Quantization）三个步骤。本文提出一种将这三个步骤联合进行端到端搜索的方法。该论文基于OFA框架。



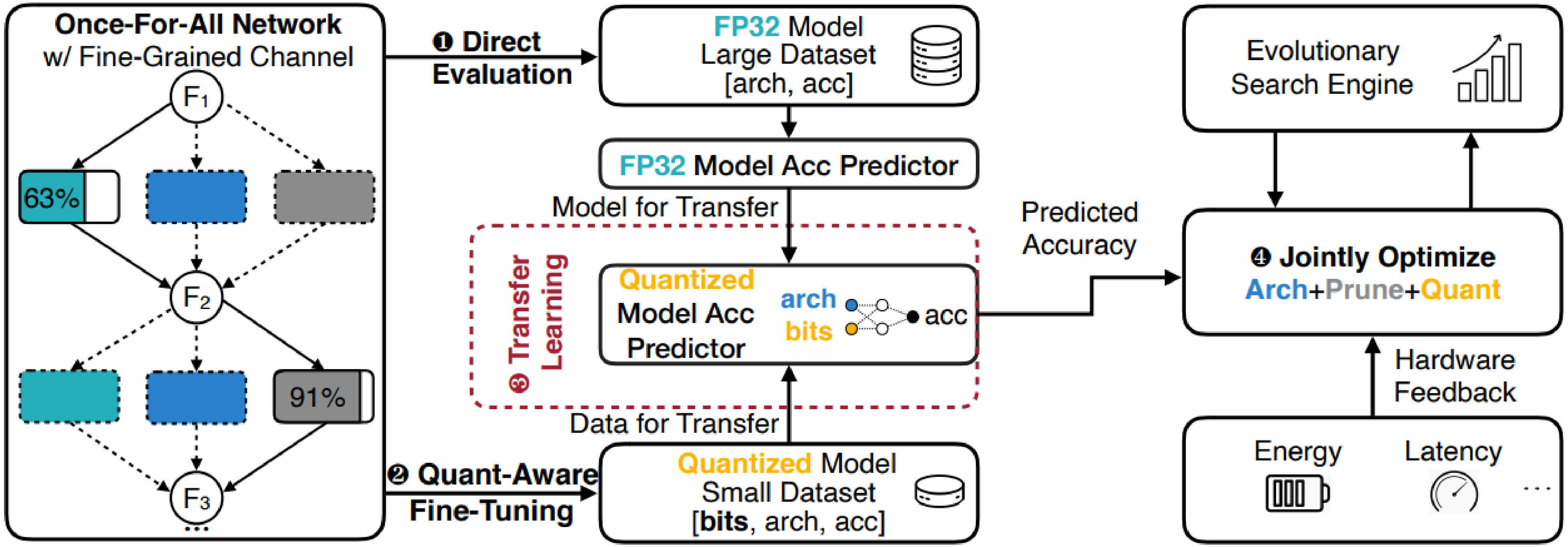
# 剪枝方法 联合压缩方法

**SOTA方法：APQ**

剪枝粒度：Channel pruning

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

**流程图：** 在OFA基础上进行联合压缩

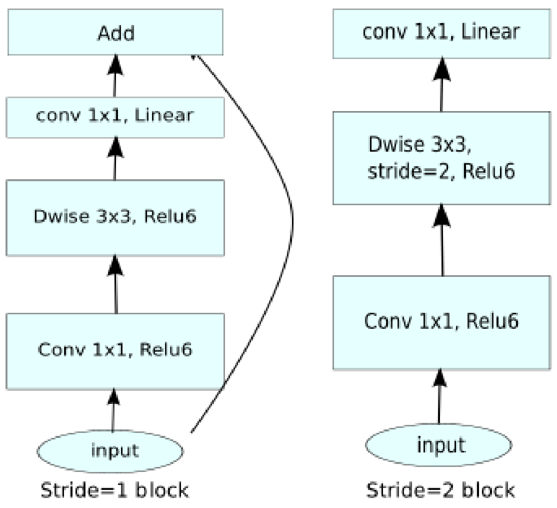



主要包括三个部件：OFA超网，量化精度预测器，进化搜索引擎

# 剪枝方法 联合压缩方法

## OFA介绍

Once-for-All: Train One Network and Specialize it for Efficient Deployment




(d) Mobilenet V2

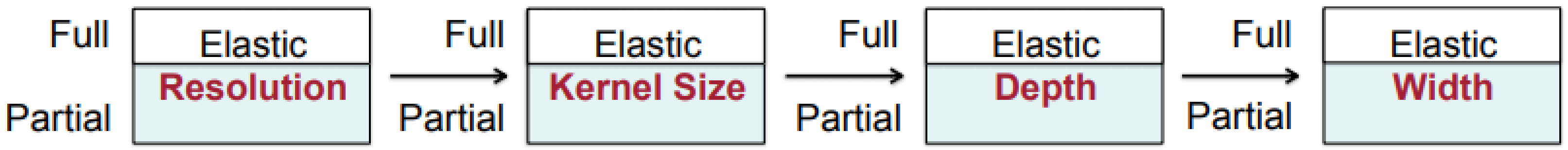
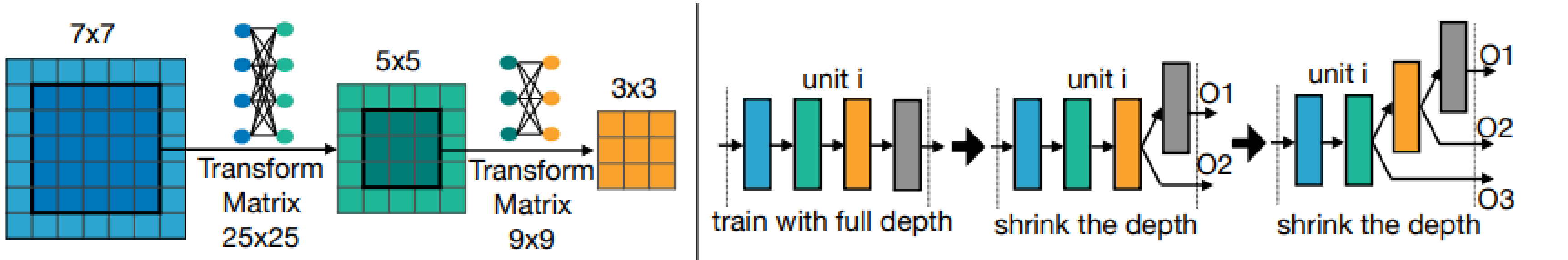
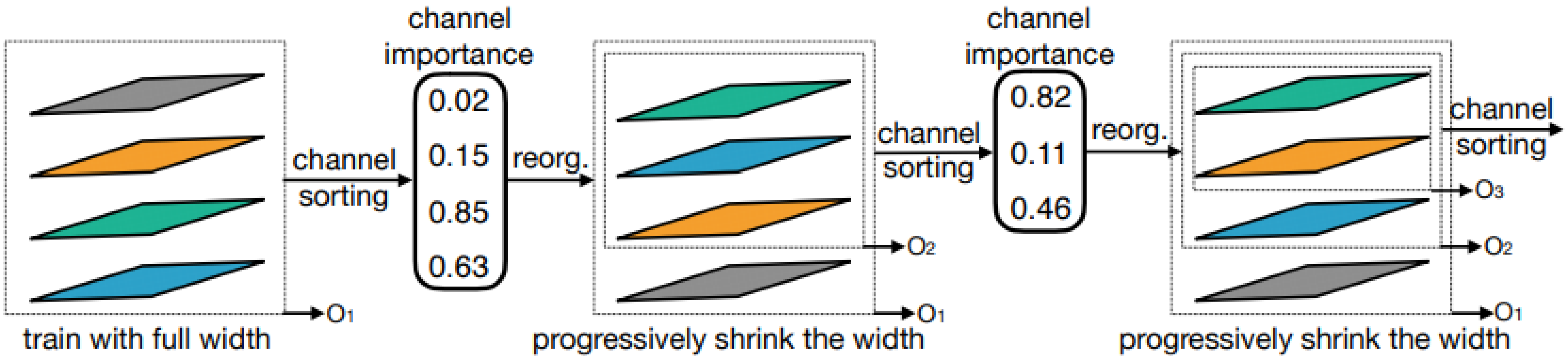



Figure 3: Illustration of the progressive shrinking process to support different depth $D$, width $W$, kernel size $K$ and resolution $R$. It leads to a large space comprising diverse sub-networks ($> 10^{19}$).




Width Expand_ratio = {3,4,6}




搜索空间：Supernet
搜索方法：Oneshot
评估方法：Weight-sharing

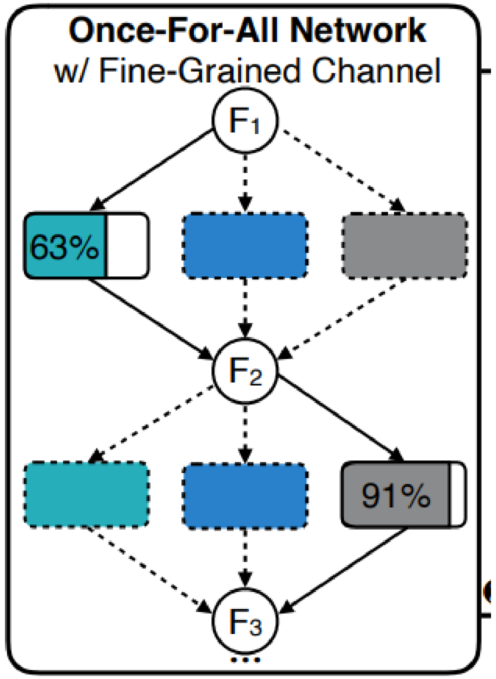
# 剪枝方法 联合压缩方法

## SOTA方法：APQ

**关于剪枝:** 提出 Fine-grained

原OFA的宽度选项：{3,4,6}*256

Fine-grained宽度：[768,776,…,1536]

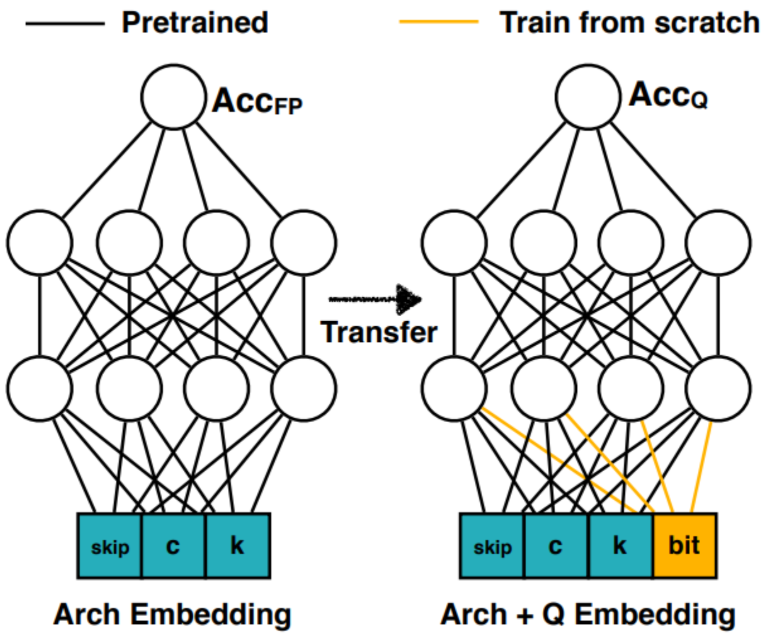



**关于量化:** 混合精度量化比特数 {4,8}，量化公式为：

$$w' = \max(0, \min(2v, round(\frac{2w}{2^b - 1}) \cdot v)) - v$$

**如何训练?** 使用量化**预测器**，三层MLP网络

1. 在超网训练完成后，采样出 <arch,acc> 结构精度对，组成全精度训练集 **A**
2. MLP在 **A** 上训练
3. 采样出 <bits,arch,acc> 数据集 **B**
4. MLP在**B**上进行Finetune



# 剪枝方法 联合压缩方法

## SOTA方法：APQ

剪枝粒度：Channel pruning

APQ: Joint Search for Network Architecture, Pruning and Quantization Policy

**Algorithm 1: APQ framework**

**Input:** Pretrained once-for-all network $\mathcal{S}$, evolution round $iterMax$, population size $N$, mutation rate $prob$, architecture constraints $C$.

Use $\mathcal{S}$ to generate FP32 model dataset $\mathcal{D}_{FP}$ $\langle$arch, acc$\rangle$ and quantized model dataset $\mathcal{D}_{MP}$ $\langle$quantization policy, arch, acc$\rangle$.

Use $\mathcal{D}_{FP}$ to train a full precision (FP) accuracy predictor $\mathcal{M}_{FP}$.

Use $\mathcal{D}_{MP}$ and $\mathcal{M}_{FP}$ (pretrained weight to transfer) to train a mixed precision (MP) accuracy predictor $\mathcal{M}_{MP}$.

Randomly generate initial population $\mathcal{P}$ $\langle$quantization policy, arch$\rangle$ with size $N$ satisfying $C$.

**for** $i = 1 \dots iterMax$ **do**

  Use $\mathcal{M}_{MP}$ to predict accuracy for candidates in $\mathcal{P}$ and update $Top_k$ with the candidates having Top k highest accuracy.

  $\mathcal{P}_{crossover} = \text{Crossover}(\text{Top}_k, N/2, C)$

  $\mathcal{P}_{mutation} = \text{Mutation}(\text{Top}_k, N/2, prob, C)$

  $\mathcal{P} = \mathcal{P} \cup \mathcal{P}_{crossover} \cup \mathcal{P}_{mutation}$

**Output:** Candidate with best accuracy in $Top_k$.

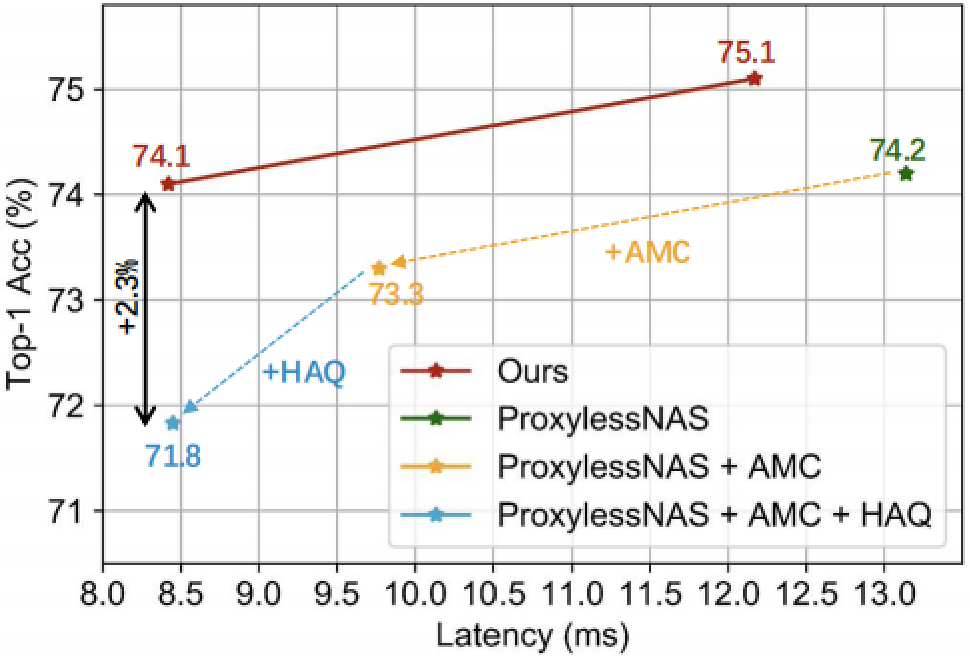
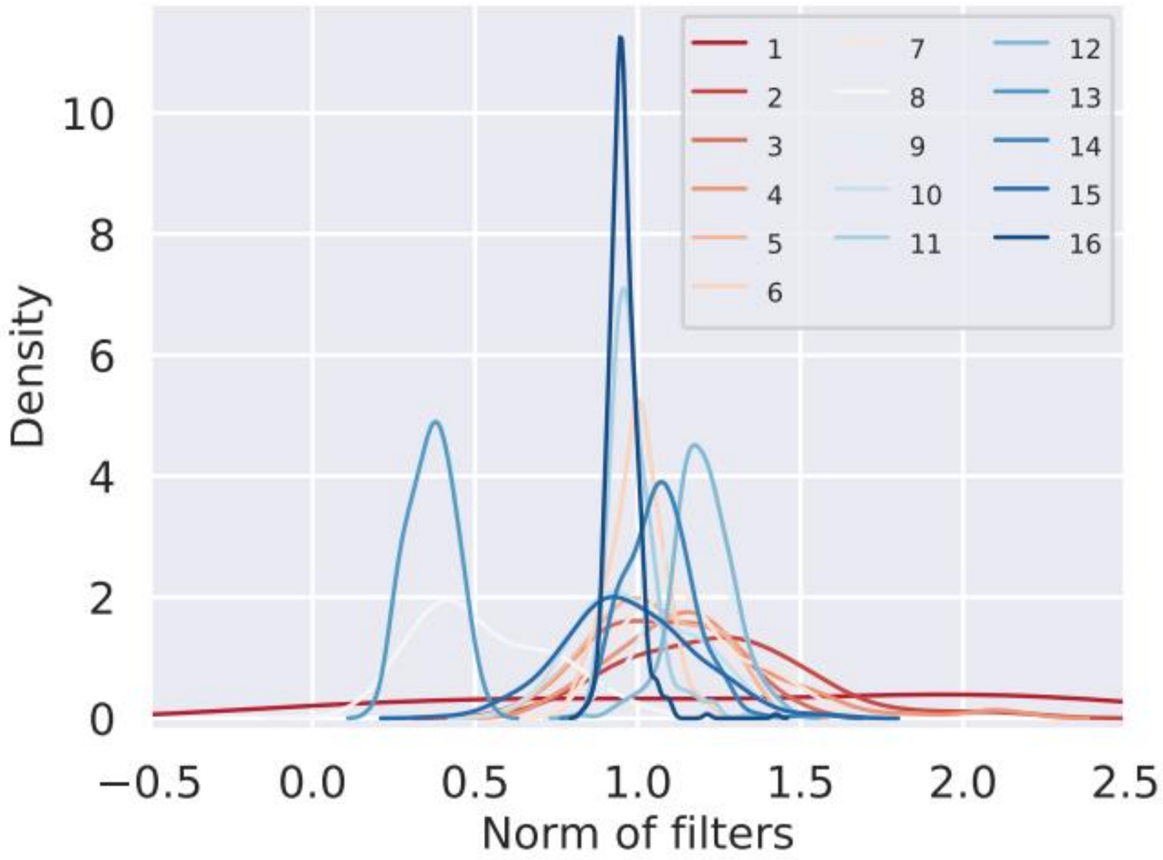



Figure 5. Comparison with *sequentially designed* mixed-precision models searched by AMC and HAQ [5, 12, 36] under latency constraints. Our joint designed model while achieving better accuracy than sequentially designed models.

# 目录 Contents

# Hybrid Search For pruning

问题由来：不同层参数分布不同




对ResNet18每层权重进行核密度估计，得到的曲线分布如左图

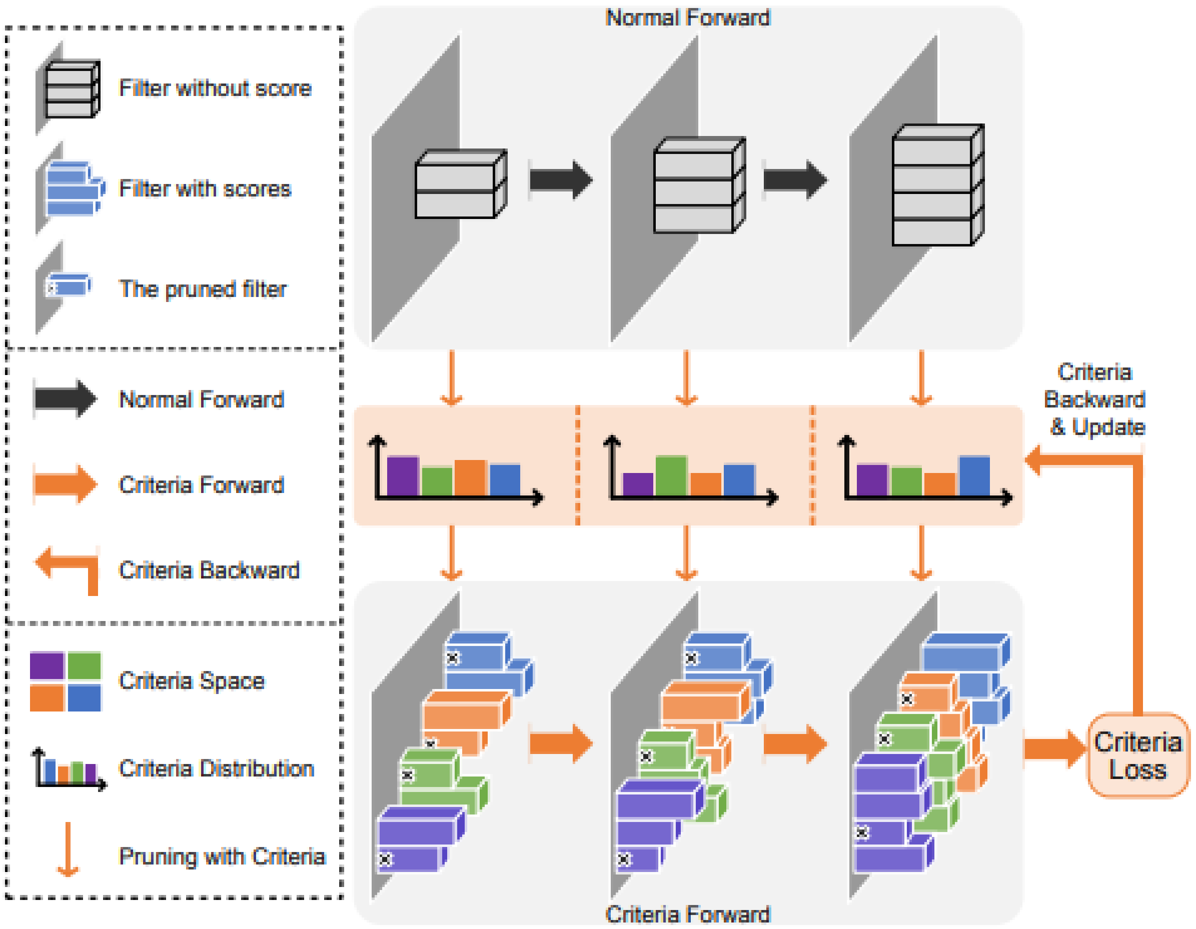
使用同一个剪枝标准对不同分布的参数进行剪枝，就会出现无法充分剪枝，达不到最优效果的问题


He, Y., Ding, Y., Liu, P., Zhu, L., Zhang, H., & Yang, Y. (2020). Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2009-2018).

# Hybrid Search For pruning

## 相关工作

Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration (LFPC)




**中稿情况：**CVPR2020

**主要贡献：**

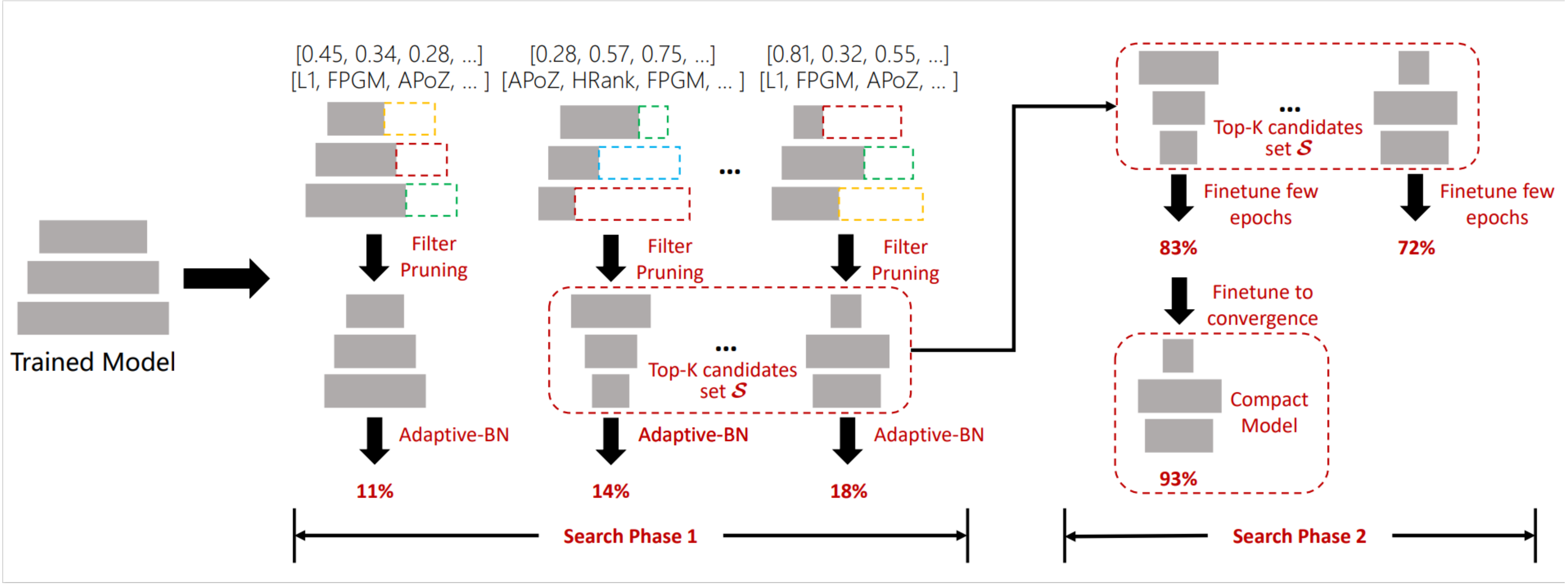
- 构造**剪枝方法搜索空间**：对每层采用不同的剪枝标准 Criteria
- 使用可微分方法进行该空间内的搜索，效果达到当时的SOTA

**不足：**

- 搜索策略：效率低下，搜索开销与预训练开销在同一个数量级
- 实验结果：只在ResNet上做实验

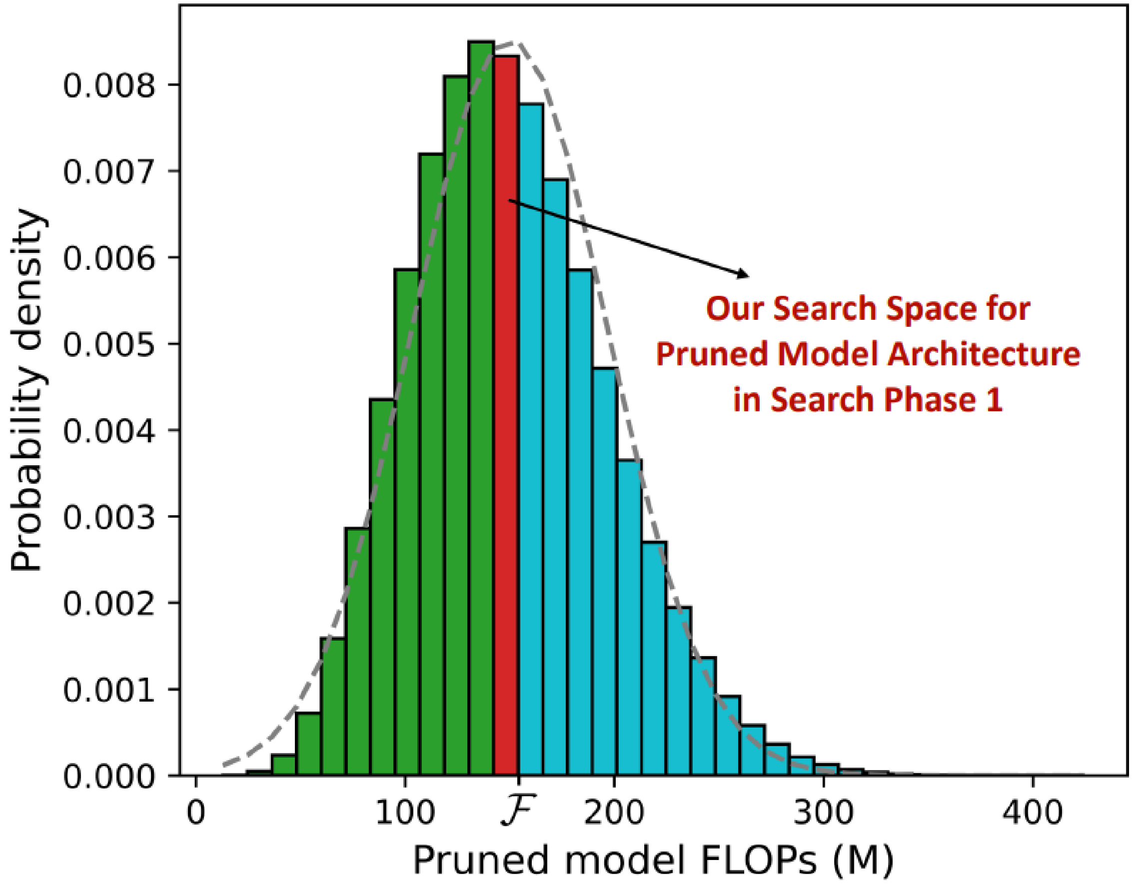
# Hybrid Search For pruning

## 流程图



# Hybrid Search For pruning

## Constraint Search Space




**针对问题：**随机采样，得到的结构FLOPs分布太广，搜索效率低

**方法：**对每次生成的模型结构进行缩放，将结构缩放到目标区间，并据此更新结构。

$$\mathrm{P} = \min(\max(P * scale, bound_{lower}), bound_{upper})$$

# Hybrid Search For pruning

## 搜索空间设计

与LFPC类似，数个候选剪枝标准

| | | | | |
|---|---|---|---|---|
| L1 | L2 | FPGM | APOZ | .... |

## 评估策略

从头训练开销巨大不可行
采用 Adaptive BN 快速评估

## 搜索方法

进化算法，同时搜索：
- 每层剪枝比例
- 每层应用的剪枝方法

种群中的个体包括两部分信息：比例和方法

| | | | | |
|---|---|---|---|---|
| 0.34 | 0.25 | 0.56 | … | 0.42 |
| L1 | APOZ | FPGM | … | L1 |

Li, Bailin, et al. "Eagleeye: Fast sub-net evaluation for efficient neural network pruning." *European Conference on Computer Vision*. Springer, Cham, 2020.

# Hybrid Search For pruning

## 实验结果

| Method | Top1 Acc | FLOPs | #Params |
|---|---|---|---|
| Base model | 94.47% | 313.73M | 14.98M |
| $\ell$-1 (Li et al. 2016) | 93.40% | 206.00M | 5.40M |
| SSS (Huang and Wang 2018) | 93.02% | 183.13M | 3.93M |
| HRank (Lin et al. 2020a) | 93.43% | 145.61M | 2.51M |
| NSPP (Zhuang et al. 2020) | 93.88% | 144.21M | 2.49M |
| **Hybrid Search** ($\mathcal{F}$ = 150M) | **94.30%** | 148.90M | 4.84M |
| GAL-0.05 (Lin et al. 2019) | 92.03% | 189.49M | 3.36M |
| HRank (Lin et al. 2020a) | 92.34% | 108.61M | 2.64M |
| **Hybrid Search** ($\mathcal{F}$ = 100M) | **94.21%** | 98.90M | 2.63M |
| GAL-0.1 (Lin et al. 2019) | 90.73% | 171.89M | 2.67M |
| HRank (Lin et al. 2020a) | 91.23% | 73.70M | 1.78M |
| ABCPruner (Lin et al. 2020c) | 93.08% | 82.81M | 1.67M |
| DPFPS (Ruan et al. 2021) | 93.52% | 91.24M | 1.00M |
| **Hybrid Search** ($\mathcal{F}$ = 70M) | **93.95%** | 69.70M | 1.53M |

Table 3: Top-1 accuracy of VGGNet on CIFAR-10.

| Method | Top1 Acc (%) | FLOPs (M) |
|---|---|---|
| MobileNet-Base | 70.6 | 569 |
| Uniform(0.75×) | 68.4 | 325 |
| NetAdapt (Yang et al. 2018) | 69.1 | 284 |
| AMC (He et al. 2018b) | 70.5 | 285 |
| MetaPruning (Liu et al. 2019) | 70.6 | 281 |
| **Hybrid Search** ($\mathcal{F}$ = 285M) | **70.8** | 283 |
| Uniform(0.5×) | 63.7 | 149 |
| MetaPruning (Liu et al. 2019) | 66.1 | 149 |
| **Hybrid Search** ($\mathcal{F}$ = 150M) | **67.5** | 150 |
| **Hybrid Search** ($\mathcal{F}$ = 100M) | 65.3 | 100 |
| Uniform(0.25×) | 50.6 | 41 |
| MetaPruning (Liu et al. 2019) | 57.2 | 41 |
| **Hybrid Search** ($\mathcal{F}$ = 50M) | **59.8** | 49 |

Table 5: Pruning results of MobileNet on ImageNet.

# Hybrid Search For pruning

## 实验结果

| Model | Method | Top1 Acc (%) | Top5 Acc (%) | FLOPs (M) | Ratio↓ |
|---|---|---|---|---|---|
| ResNet-18 | Base model | 69.66 | 89.08 | 1824.52 | - |
| | SFP (He et al. 2018a) | 67.10 | 87.78 | 1061.87 | 41.8% |
| | DSA (Ning et al. 2020) | 68.61 | 88.35 | 1080.12 | 40.8% |
| | MiL (Dong et al. 2017) | 66.07 | 86.77 | 1193.24 | 34.6% |
| | FPGM (He et al. 2019) | 68.41 | **89.63** | 1080.12 | 40.8% |
| | ABCPruner-100 (Lin et al. 2020c) | 67.80 | 88.00 | 968.13 | 46.9% |
| | **Hybrid Search** ($\mathcal{F} = 1000$M) | **69.44** | 88.83 | 990.90 | 45.7% |
| ResNet-34 | Base model | 73.28 | 91.45 | 3679.23 | - |
| | SFP (He et al. 2018a) | 71.83 | 90.33 | 2167.07 | 41.1% |
| | FPGM (He et al. 2019) | 72.63 | 91.08 | 2167.07 | 41.1% |
| | ABCPruner-90 (Lin et al. 2020c) | 70.98 | 90.05 | 2170.77 | 41.0% |
| | DPFPS (Ruan et al. 2021) | 72.25 | 90.80 | 2170.77 | 41.0% |
| | **Hybrid Search** ($\mathcal{F} = 2000$M) | **73.20** | 91.00 | 1986.10 | 46.0% |
| ResNet-50 | Base model | 76.01 | 92.96 | 4135.70 | - |
| | SFP (He et al. 2018a) | 74.61 | 92.06 | 2406.98 | 41.8% |
| | FPGM (He et al. 2019) | 75.59 | **92.87** | 2167.07 | 47.6% |
| | ABCPruner-80 (Lin et al. 2020c) | 73.86 | 91.69 | 2390.43 | 42.2% |
| | SRR-GR (Wang, Li, and Wang 2021) | 75.11 | 92.50 | 1856.62 | 55.1% |
| | SCOP (Lin et al. 2020c) | 75.26 | 92.50 | 1877.29 | 54.6% |
| | DPFPS (Ruan et al. 2021) | 75.55 | 92.54 | 2224.63 | 46.2% |
| | **Hybrid Search** ($\mathcal{F} = 1800$M) | **75.74** | 92.56 | 1736.20 | 58.0% |

Table 4: Pruning results of ResNet on ImageNet.

# Hybrid Search For pruning

## 消融实验

只采用一种剪枝标准进行搜索的消融实验

| Model | $c_i$ in layers | Top-1 Acc | FLOPs |
|---|---|---|---|
| VGG-16 | L1 | 93.1% | 69.1M |
| | FPGM | 93.3% | 69.8M |
| | APoZ | 93.7% | 69.2M |
| | HRank | 93.7% | 68.1M |
| | Search | 94.0% | 69.7M |

Table 7: Pruning results with different pruning algorithms.

在相同的训练开销下，对不同阶段进行消融实验

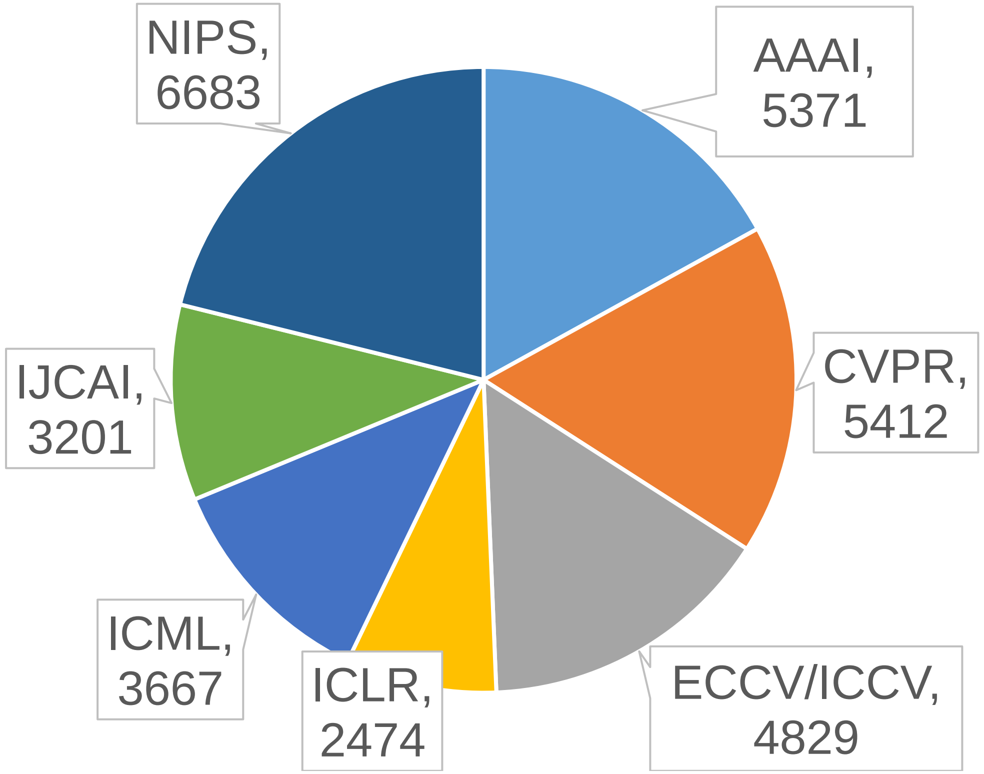
| Model | Phase 1 | Phase 2 | Top-1 Acc (%) | FLOPs |
|---|---|---|---|---|
| MobileNet | ✗ | ✓ | 62.3 | 144M |
| | ✓ | ✗ | 65.5 | 147M |
| | ✓ | ✓ | 67.5 | 150M |

Table 8: Pruning results after removing one of the search phase.

# AI_paper_collector

**A simple, lightweight, easy use pythonic AI Conference paper collector**

| 2018-2021 |
|---|
| AAAI |
| CVPR |
| ECCV |
| ICCV |
| ICML |
| ICLR |
| IJCAI |
| NIPS |




Repo Link

https://github.com/dercaft/AI_paper_collector

**Totally Collect 31637 papers in xlsx/csv file**

# Thanks

# Q&A


Derek
南极洲
扫一扫上面的二维码图案，加我微信