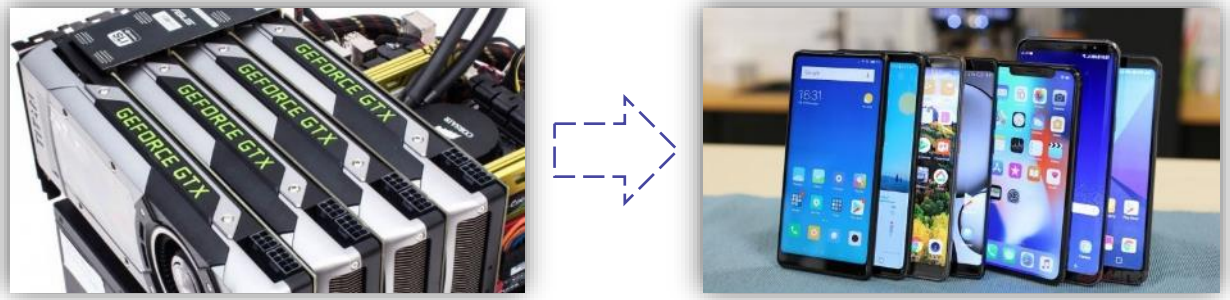
卷积神经网络压缩

吴建鑫

南京大学

背景介绍

- 深度神经网络会消耗大量的计算资源

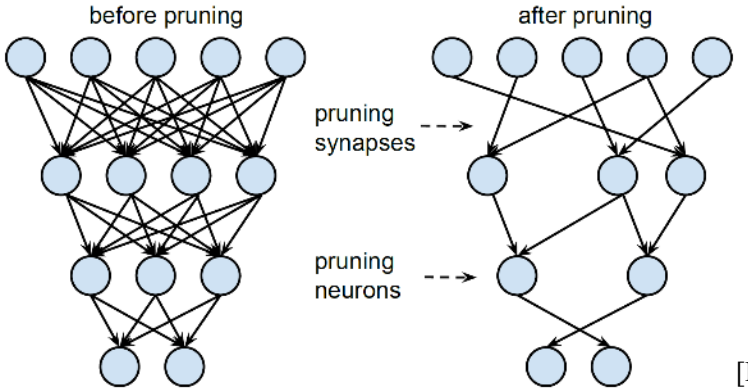


Q：如何在资源受限的小型设备上运行深度网络？

A：轻量级网络设计、参数量化、网络压缩 …

背景介绍

- 深度学习时代的网络剪枝



[Han et al., NeurIPS 2015]

准则：所有权重低于某阈值的连接都会被移除

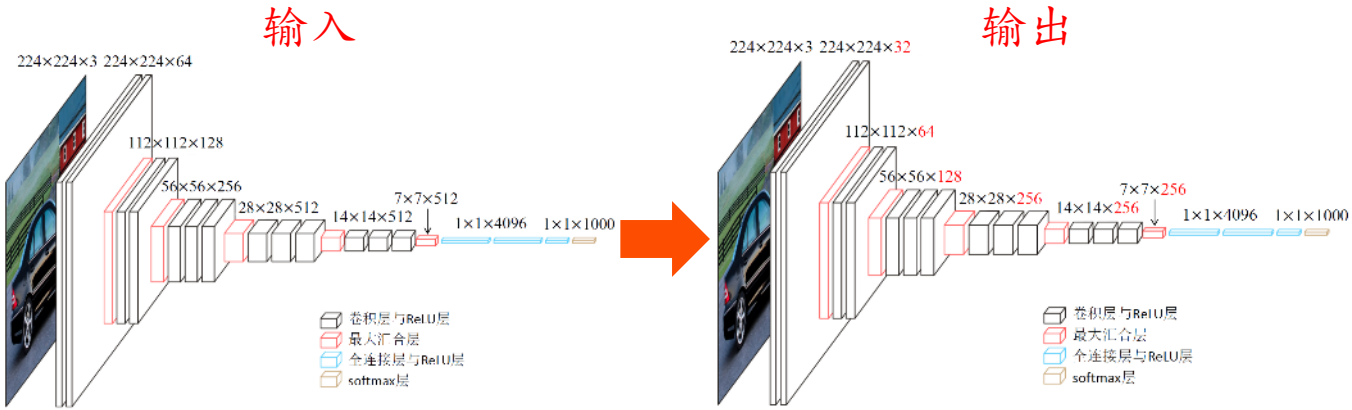
缺点：非结构化的模型很难获得实际加速效果

问题1：怎么剪？

- 简单答案：**结构化剪枝**！

背景介绍

- 滤波器级别的深度网络剪枝



- 结构化剪枝的优点
  - ✓ 高实用性：可同时减少计算量与内存占用
  - ✓ 低依赖性：结构化的剪枝不会破坏模型结构，无缝部署
  - ✓ 可扩展性：模型剪枝可与其他网络压缩技术相结合

问题2：剪哪些?

- 简单答案：**数学优化** vs. 启发式！

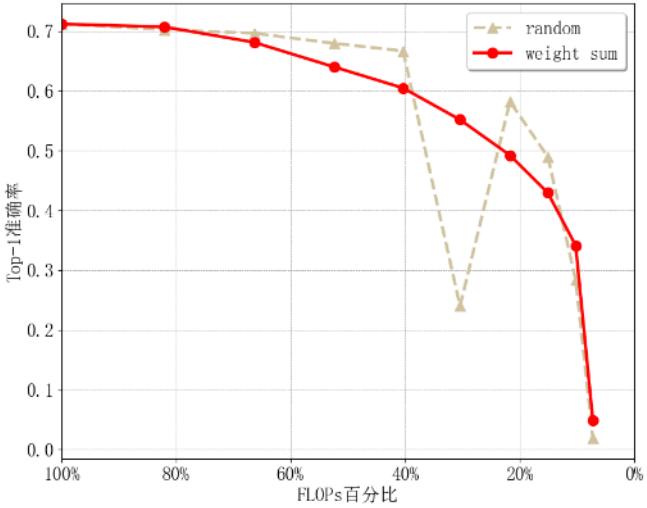
背景介绍

- 启发式剪枝算法的缺点

  - 基线方法
    - weight sum [ICLR'17]：将每个滤波器权重的绝对值作为其重要性得分
    - random：随机丢弃一些滤波器

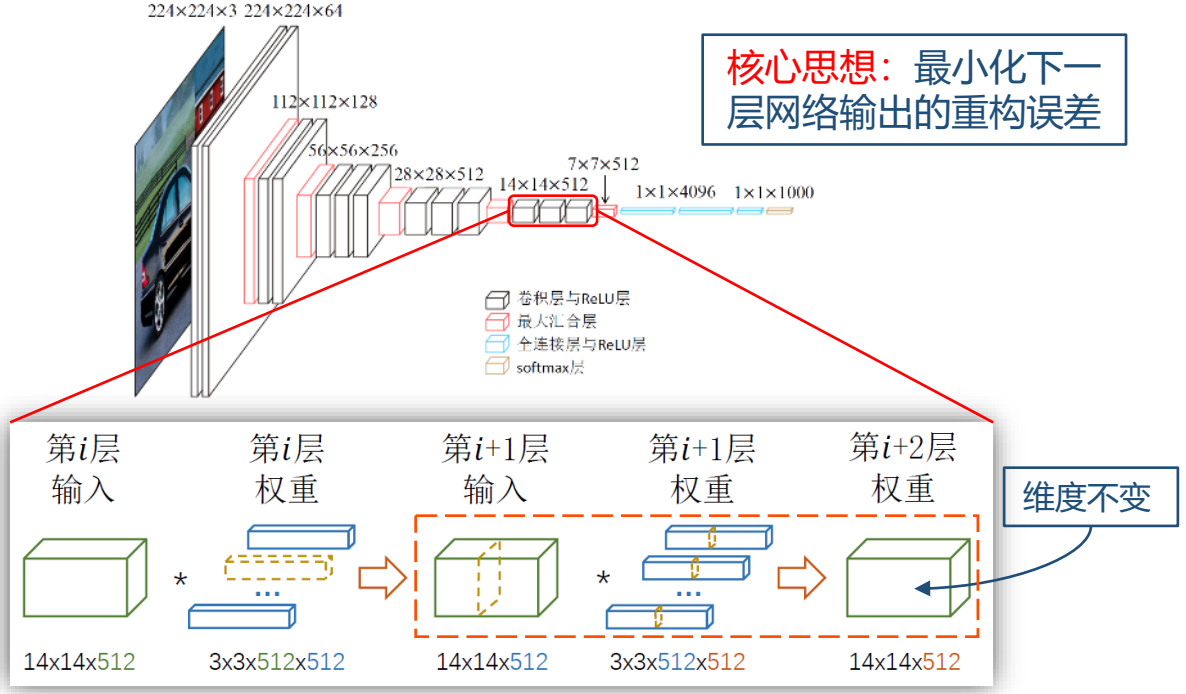
  - 缺点
    - 并不总是有效，找不到成功或失败的原因
    - 缺乏理论解释

启发：能否对剪枝进行形式化建模，通过数学推导给出一定的保障？

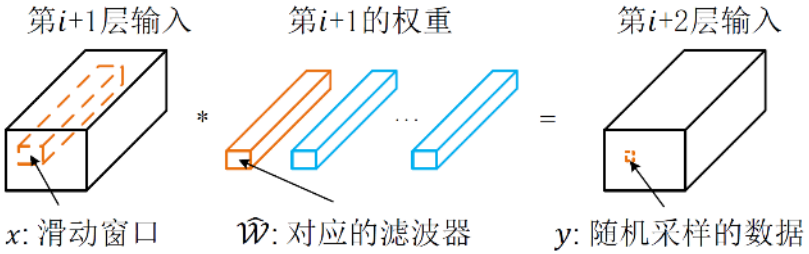


方法介绍

- ThiNet


224×224×3
224×224×64
112×112×128
56×56×256
28×28×512
14×14×512
7×7×512
1×1×4096
1×1×1000
卷积层与ReLU层
最大汇合层
全连接层与ReLU层
softmax层
核心思想：最小化下一层网络输出的重构误差
第i层 输入
第i层 权重
第i+1层 输入
第i+1层 权重
第i+2层 权重
维度不变
14x14x512
3x3x512x512
14x14x512
3x3x512x512
14x14x512

方法介绍

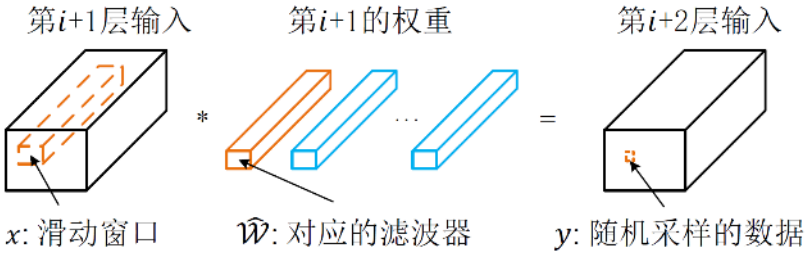


正常卷积：
$$y = \sum_{c=1}^{C} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \widehat{\mathcal{W}}_{c,k_1,k_2} \times x_{c,k_1,k_2} + b.$$

定义：
$$\hat{x}_c = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \widehat{\mathcal{W}}_{c,k_1,k_2} \times x_{c,k_1,k_2}, \quad \hat{y} = y - b.$$

代入可得：
$$\hat{y} = \sum_{c=1}^{C} \hat{x}_c,$$ 其中，C是前一层滤波器的数量

方法介绍



剪枝：在C个通道里选择一个子集S，并移除那些不属于S的滤波器

用数学语言来描述

目标：找到子集 $S \subset \{1, 2, \ldots, C\}$ 使得下式尽可能成立

$$\hat{y} = \sum_{c \in S} \hat{x}_c$$

方法：收集m个训练样本$\{(\hat{\mathrm{X}}_i, \hat{y}_i)\}$，最小化$\hat{y}$的重构误差

方法介绍

1. 转化为优化目标：

$$\arg\min_{S} \sum_{i=1}^{m} \left(\mathbf{y}_i - \sum_{j \in S} \mathbf{X}_{i,j} \right)^2$$

$$\text{s.t.} \quad |S| = C \times r, \quad S \subset \{1, 2, \ldots, C\}$$

2. 通道放缩：

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_{i=1}^{m} (\mathbf{y}_i - \widehat{\mathbf{X}}_i \mathbf{w})^2$$

$$\hat{\mathbf{w}} = (\widehat{\mathbf{X}}^{\mathrm{T}} \widehat{\mathbf{X}})^{-1} \widehat{\mathbf{X}}^{\mathrm{T}} \mathbf{y}$$

3. 最终的优化目标：

$$\arg\min_{S} \sum_{i=1}^{m} \left(\mathbf{y}_i - \sum_{j \in S} \hat{w}_j \mathbf{X}_{i,j} \right)^2$$

$$\text{s.t.} \quad |S| = C \times r, \quad S \subset \{1, 2, \ldots, C\}$$

实验分析

- 在小数据集上的剪枝

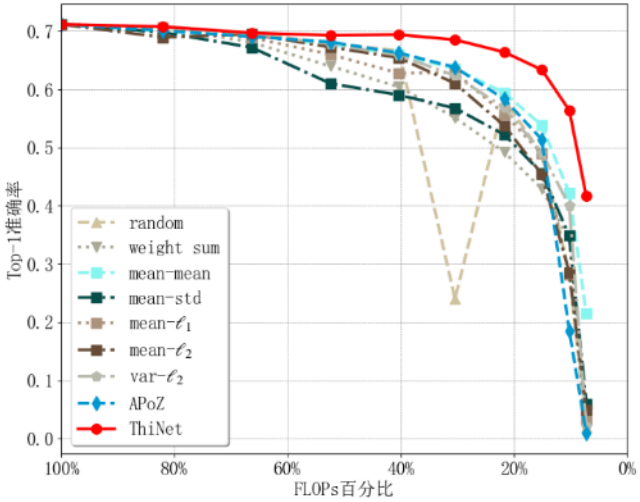
- 基线方法：
  - random：随机丢弃一些滤波器
  - weight sum：将每个滤波器权重的绝对值作为其重要性得分
  - APoZ：计算输出张量中每个通道的稀疏度作为其重要性得分
  - mean-mean：$s_i = \frac{1}{N}\sum \text{mean}(\text{I}(i,:,:))$
  - mean-std：$s_i = \frac{1}{N}\sum \text{std}(\text{I}(i,:,:))$
  - mean-ℓ_1：$s_i = \frac{1}{N}\sum \|\text{I}(i,:,:)\|_1$
  - mean-ℓ_2：$s_i = \frac{1}{N}\sum \|\text{I}(i,:,:)\|_2$
  - var-ℓ_2：$s_i = \frac{1}{N}\sum \text{var}(\|\text{I}(i,:,:)\|_2)$

- 数据集：
  - CUB200：鸟类数据集，200类，5994张训练图像，5794张测试图像
  - Indoor67：室内场景数据集，67类，5360 张训练图像，1340 张测试图像

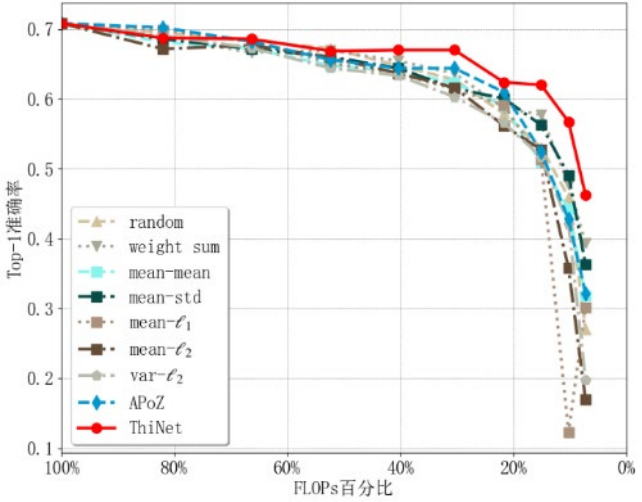
实验分析

- 在小数据集上的剪枝

CUB200数据集



Indoor67数据集



ThiNet算法在不同压缩率指标下均取得了良好的压缩效果

实验分析

- 在ImageNet数据集上的剪枝 – VGG

算法	Top-1	Top-5	# 参数.	#FLOPs
VGG-16[8] [1]	71.50%	90.01%	138.34M	30.94B
连接级剪枝[20]	68.66%	89.12%	10.30M	6.50B
APoZ-1[25]	66.20%	87.60%	67.81M	–
APoZ-2[25]	70.17%	89.69%	51.24M	–
Taylor-1[26]	–	87.00%	–	11.50B
Taylor-2[26]	–	84.50%	–	8.00B
weight sum[24] [2]	69.35%	89.13%	131.44M	9.58B
Channel-Pruning (5×)[27] [3]	67.80%	88.10%	130.87M	7.03B
从头训练[4]	67.00%	87.45%	131.44M	9.58B
ThiNet-Conv-1	69.74%	89.41%	131.44M	9.58B
ThiNet-Conv-2	69.11%	88.86%	130.50M	6.91B
ThiNet-GAP	67.69%	88.13%	8.32M	9.34B

实验分析

- 在ImageNet数据集上的剪枝 – ResNet

模型	Top-1	Top-5	# 参数	#FLOPs	速度（图片/秒）
原始模型[1]	75.30%	92.20%	25.56M	7.72B	295.12
ThiNet-70	74.03%	92.11%	16.94M	4.88B	334.87
ThiNet-50	72.03%	90.99%	12.38M	3.41B	373.92
ThiNet-30	68.17%	88.86%	8.66M	2.20B	397.86

[1] ResNet 的精度是按照官方中心裁剪的设定进行测试的：原图保持长宽比短边缩放到 256，之后进行 224 × 224 中心裁剪（https://github.com/KaimingHe/deep-residual-networks）。

小结

- 针对以往剪枝算法多为启发式，缺少理论保障的缺点，对剪枝过程进行了形式化建模，提出了一种基于重构误差最小化的剪枝算法：ThiNet

- Jian-Hao Luo, Jianxin Wu and Weiyao Lin. ThiNet: A filter level pruning method for deep neural network compression. **In Proceedings of the IEEE International Conference on Computer Vision (ICCV)**, Venice, Italy, October 2017, pp. 3505-3513.

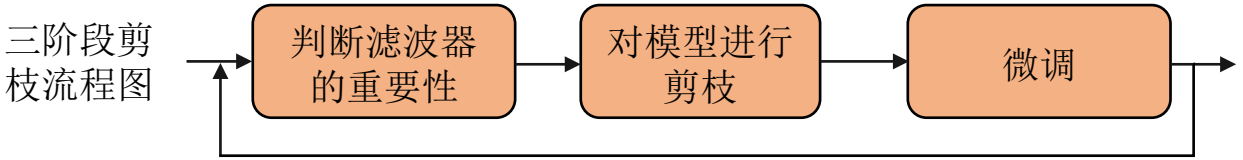
- Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu and Weiyao Lin. ThiNet: Pruning CNN Filters for a Thinner Net. **IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)**, 41(10), 2019: 2525-2538.

问题3：能不能自动化?

- 简单答案：能，且应该自动化！

背景介绍

- 三阶段剪枝的缺点

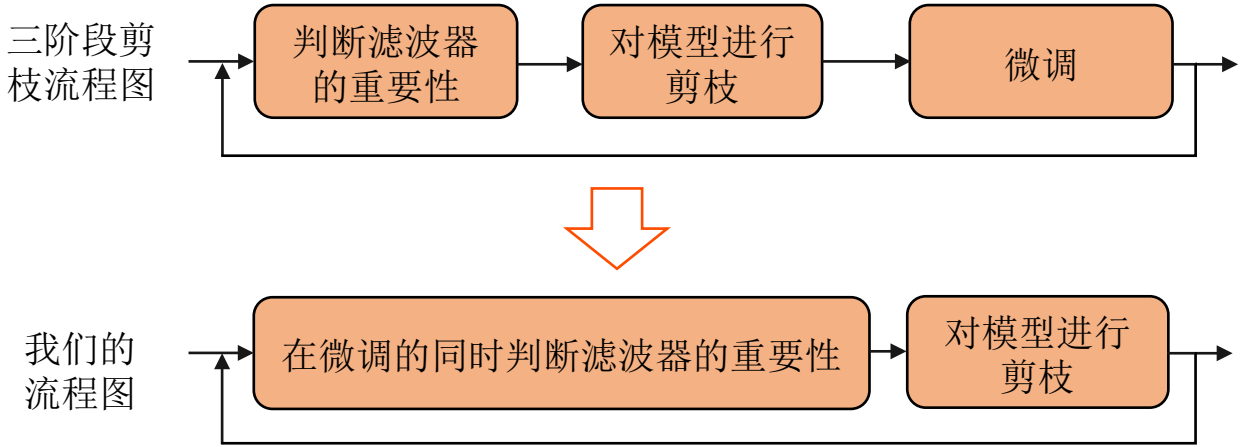


- 三阶段剪枝策略：
  - 衡量重要性——移除弱滤波器——微调恢复精度
  - 过于依赖人工手动设计的重要性衡量标准
  - 重要性衡量标准无法适用于所有的情况
  - 每个层的压缩比都相等

启发：能否让网络自己来决定哪些滤波器需要被剪枝？

背景介绍

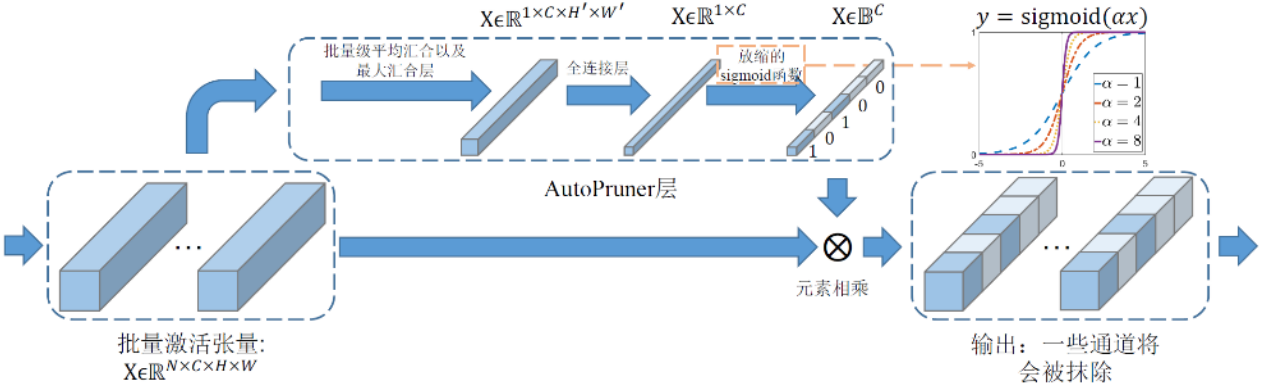
- 新的剪枝流程



解决方案：设计新的网络层，通过**端到端**的方式**自动**完成剪枝

方法介绍

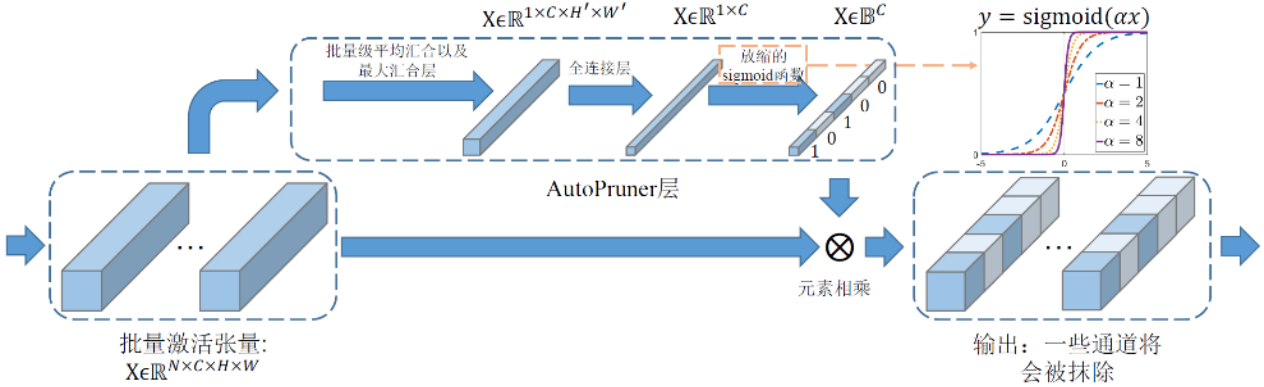
- AutoPruner算法框架



- 网络结构：将AutoPruner层添加到卷积-激活函数之后，如，
  conv1-1 – ReLU – AutoPruner – conv1-2 ···
- 输入：批大小为N的4阶张量
- 输出：C维二值向量，0元素所对应的通道和滤波器将被移除

方法介绍

- AutoPruner算法框架



- 具体实现：
  1. 汇合 (pooling)：利用批均值汇合和最大汇合，综合不同图像之间的信息
  2. 编码：使用全连接层得到C维向量
  3. 二值化：使用放缩sigmoid函数进行逐步二值化

方法介绍

- 稀疏控制与损失函数

- 损失函数由分类损失与压缩率控制两部分组成：

$$\mathcal{L} = \mathcal{L}_{\text{classification}} + \lambda \left\| \frac{\|\mathbf{v}\|_1}{C} - r \right\|_2^2$$

  - $\mathcal{L}_{\text{classification}}$：标准的分类损失项，如cross-entropy；
  - v：$\mathrm{v} = \mathrm{sigmoid}(\alpha x)$是AutoPruner层的输出，由实值逐渐变为二值；
  - r: 预设的压缩率；

- 如何控制α的值？
  - α的值过大：v的值将很快变为二值，造成剪枝提前完成。此时，AutoPruner将退化为随机选择；
  - α的值过小：v有可能很难甚至根本无法收敛到二值；
  - α在不同网络模型、不同网络层的合适值可能不大相同；

解决方案：提出了一种自适应的二值化控制方法

实验分析

- 在CUB200数据集上的剪枝效果对比

方法	压缩率 $r = 0.2$			压缩率 $r = 0.5$		
	top-1 (%)	top-5 (%)	#FLOPs	top-1 (%)	top-5 (%)	#FLOPs
微调后的 VGG16	76.68	94.06	30.93B	76.68	94.06	30.93B
随机选择	57.28	83.52	2.62B	70.25	91.16	9.63B
ThiNet	63.12	87.54	2.62B	73.00	92.27	9.63B
AutoPruner	**65.06**	**87.93**	2.62B	**73.45**	**92.56**	9.63B

AutoPruner超过了前面介绍的ThiNet算法

实验分析

- 在ImageNet上剪枝ResNet50模型

方法	Top-1	Top-5	#FLOPs	加速比[1]
原始 ResNet-50 模型[2]	76.15%	92.87%	8.18B	1.00×
AutoPruner (r = 0.3)	**73.05%**	**91.25%**	2.78B	2.94×
Taylor-FO-BN-56%[78]	71.69%	90.59%	2.68B	2.88×
GAL-1-joint[31]	69.31%	89.12%	2.22B	3.48×
ThiNet-30	68.42%	88.30%	2.20B	3.51×
AutoPruner (r = 0.5)	**74.76%**	**92.15%**	4.00B	2.05×
AutoPruner with block pruning (r = 0.5)	73.84%	91.75%	4.56B	1.79×
Channel Pruning (2×)[27][4]	72.30%	90.80%	5.22B	1.48×
SSS (ResNet-26)[30]	71.82%	90.79%	4.00B	1.93×
ThiNet-50	71.01%	90.02%	3.41B	2.27×

实验分析

- 对轻量级网络进行剪枝

算法	精度		MACs	速度	训练时间		
	top-1	top-5			剪枝阶段	微调阶段	总时间
MobileNetV2-1.0[1]	72.19%	90.53%	300.79M	0.138s	-	-	-
MobileNetV2-0.75	69.95%	88.99%	209.08M	0.119s	-	-	-
AMC (70% FLOPs)[82]	70.85%	89.91%	210.63M	0.105s	22.52s×800	1687s×150	3.14d
Slimmable NN[33]	68.93%	88.34%	209.08M	0.120s	-	8618s× 480	47.88d
AutoPruner	**71.18%**	**89.95%**	207.93M	0.099s	4898s×5	1658s×150	3.16d

[1] https://github.com/d-li14/mobilenetv2.pytorch

AutoPruner同样适用于MobileNet的剪枝，精度更高、速度更快、所消耗的计算代价比较适中

实验分析

- 剪枝后网络的泛化性

算法	MobileNetV2-1.0	MobileNetV2-0.75	AMC[82]	Slimmable NN[33]	AutoPruner
mAP	0.6743	0.6529	0.6571	0.6602	0.6576

剪枝对模型的泛化性没有显著影响，但优势也并不明显，如何对分类以外的模型进行剪枝仍然值得进一步研究

小结

- 针对以往剪枝算法过于依赖人工经验的缺点，我们在本章提出了一种可端到端训练的自动剪枝算法AutoPruner

- Jian-Hao Luo and Jianxin Wu. AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference. **Pattern Recognition (PR)**, **2020**.

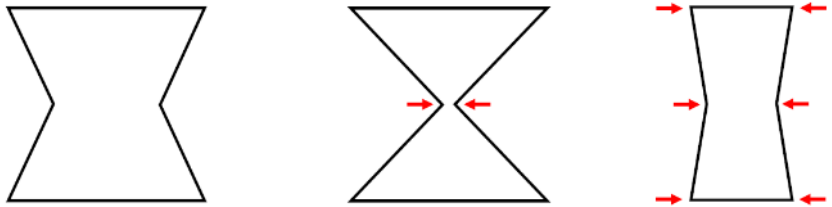
问题4，5：走向实用化

- 问题4：小数据怎么办？
- 简单答案：数据增强与知识蒸馏！

- 问题5：残差结构怎么办？
- 简单答案：残差内外一视同仁！

背景介绍

- 如何剪枝残差连接?
  - ThiNet、AutoPruner等算法均只剪枝残差结构的中间层



(a) 瓶颈结构　(b) 沙漏结构　(c) 钱包结构

同时对残差结构的中间层与输出层进行剪枝是一种更好的策略

背景介绍

- 如何直接在小数据集上进行剪枝?

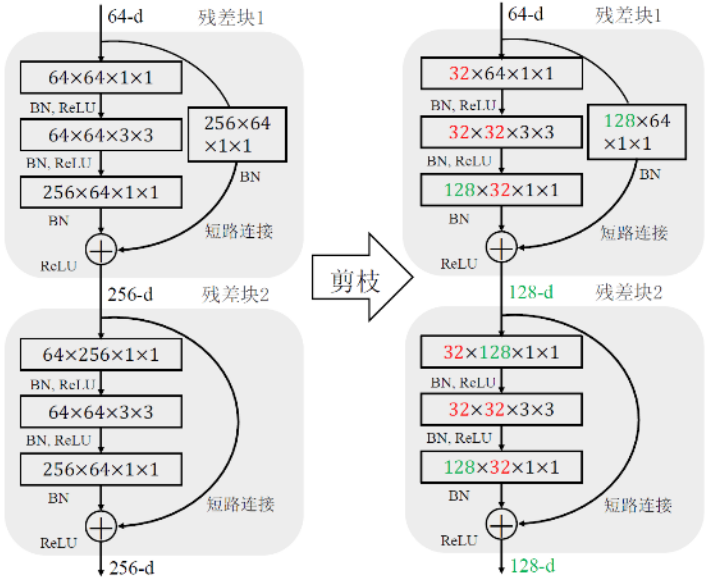
  - 直接在小数据集行进行剪枝效果不佳
    - T1：先在大数据集上进行剪枝，再迁移到小数据集上进行训练
    - T2：直接在小数据集上进行剪枝训练
    - 效果：T1 > T2

  - 直接在小数据集行进行剪枝应用更加实用
    - 由于一些实际情况（如隐私保护、法律问题等），原始大数据集可能无法获取到
    - 在大数据集上进行剪枝，代价昂贵，不利于迭代开发

如何提高小数据集上的直接剪枝效果?

方法介绍

- 如何剪枝残差连接?



分析原因

由于残差结构限制，同一组内的残差块的输出维度需时刻保持一致来完成残差相加的操作

解决方法

将不同残差块的输出维度看作一个整体，同时进行剪枝

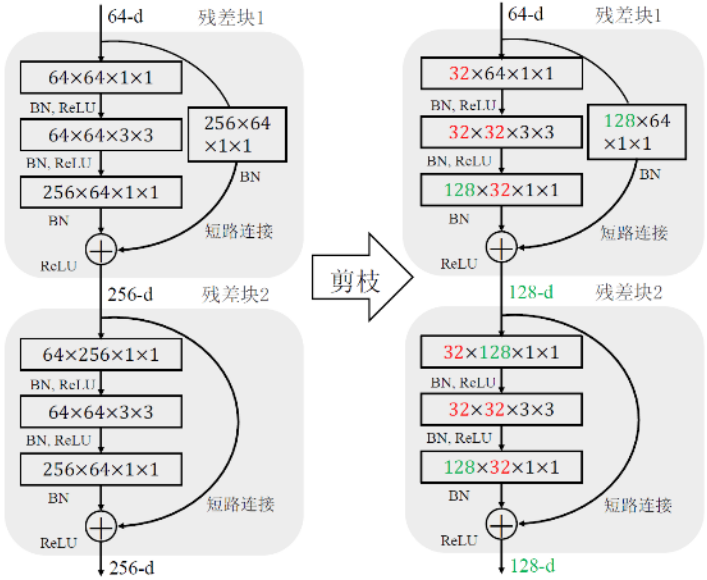
新的挑战

如何衡量多个输出维度的重要性

评价标准：关注于网络输出，计算移除输出维度后的信息损失

方法介绍

- 如何剪枝残差连接?



- 剪枝输出维度（绿色部分）：同时移除多个残差块同一输出维度的滤波器

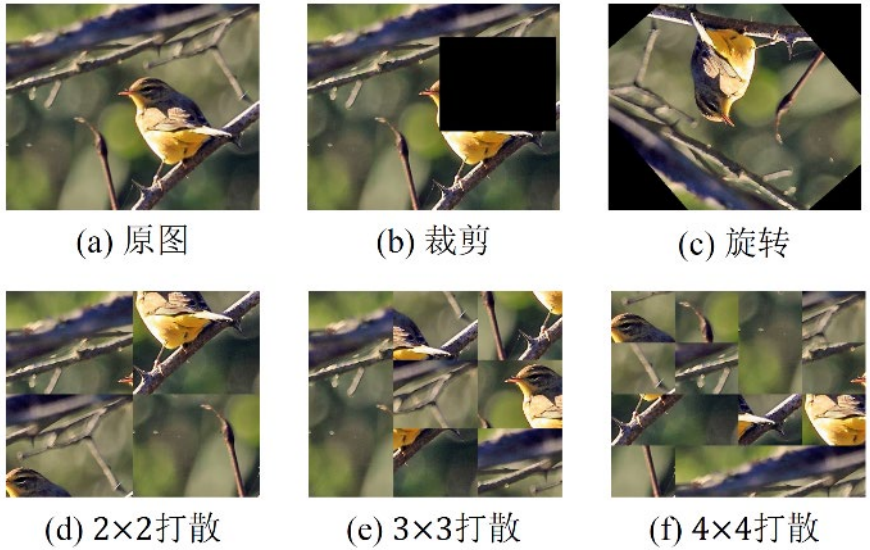
- 全局重要性衡量标准：

$$s = D_{KL}(p \parallel q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$

p：原网络的预测概率输出
q：移除滤波器后的概率输出

方法介绍

- 如何直接在小数据集上进行剪枝?
  - 分析原因：数据量不足+监督信息弱
  - 解决数据量不足的问题：图像变换以扩充数据集

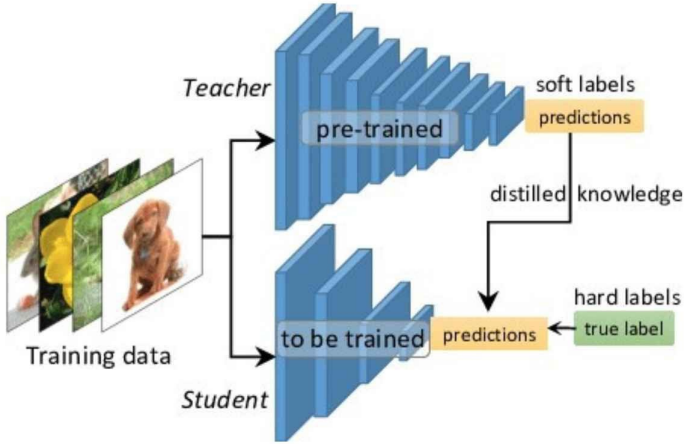


(a) 原图　(b) 裁剪　(c) 旋转

(d) 2×2打散　(e) 3×3打散　(f) 4×4打散

方法介绍

- 如何直接在小数据集上进行剪枝?
  - 分析原因：数据量不足 + 监督信息弱
  - 解决监督信息弱的问题：使用知识蒸馏技术



- 老师模型：原始网络
- 学生模型：剪枝后的网络

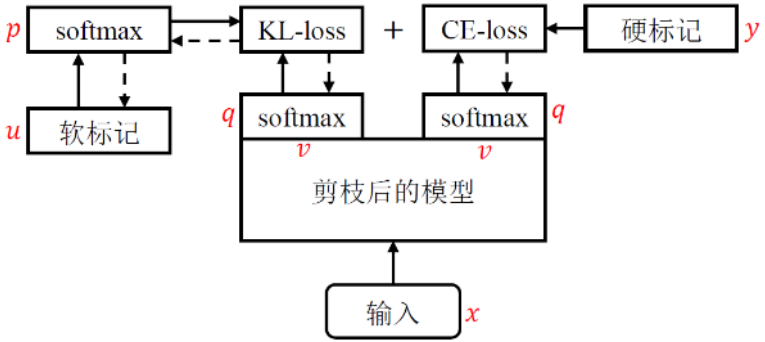
老师模型未见过扩充后的图像，其logits含噪声，不利于学生模型的学习

方法介绍

- 如何直接在小数据集上进行剪枝?

□ 解决方法：标记细化

- 第一步：在原始数据集上使用mixup和知识蒸馏技术微调剪枝后的模型，获得一个精确的模型
- 第二步：在扩充后的数据集上使用标记细化来移除噪声标记的干扰



- p：老师模型的输出；
- q：学生模型的输出；

$$\mathcal{L} = \alpha \mathrm{T}^2 \mathcal{L}_{\mathrm{KL}}(q, p) + (1 - \alpha)\, \mathcal{L}_{\mathrm{CE}}(q, y)$$

- 同时对软标记进行更新以移除噪声标记的影响
- 逆转p，q以促使网络更关注于软标记

- Kun Yi and Jianxin Wu. Probabilistic End-to-end Noise Correction for Learning with Noisy Labels. **In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, Long Beach, CA, USA, June 2019.

实验分析

- 在ImageNet上剪枝ResNet50

方法	Top-1	Top-5	MACs	# 参数
ResNet50	76.15%	92.87%	4.09G	25.56M
ThiNet-30	68.42%	88.30%	1.10G	8.66M
GAL-1-joint[31]	69.31%	89.12%	1.11G	10.21M
GDP-0.5[93]	69.58%	90.14%	1.57G	-
Taylor-FO[78]	71.69%	-	1.34G	7.90M
Slimmable NN[33]	72.10%	90.57%	1.05G	6.92M
AutoPruner	73.05%	91.25%	1.39G	12.60M
CURL	**73.39%**	**91.46%**	1.11G	6.67M

CURL的速度更快、精度更高、模型更小，证明了我们剪枝策略的有效性

实验分析

- 小数据集上的剪枝

基模型	剪枝方法	CUB200[60]		Pets[71]		Dogs[94]		Car[95]	
		精度	MACs	精度	MACs	精度	MACs	精度	MACs
MobileNetV2	MobileNetV2-1.0	78.77%	299.77M	89.94%	299.56M	78.94%	299.67M	87.27%	299.76M
	MobileNetV2-0.5	73.96%	96.12M	86.32%	95.91M	72.15%	96.02M	81.50%	96.11M
	Slimmable NN[33]	72.20%	96.12M	83.37%	95.91M	67.81%	96.02M	87.02%	96.11M
	CURL	**78.72%**	96.07M	**86.89%**	95.91M	**74.72%**	96.04M	**87.64%**	96.15M
ResNet50	ResNet50	84.76%	4.09G	92.59%	4.09G	83.82%	4.09G	91.89%	4.09G
	ResNet50-CURL	81.33%	1.11G	**90.84%**	1.11G	**81.60%**	1.11G	88.60%	1.11G
	Slimmable NN[33]	80.05%	1.05G	89.67%	1.05G	77.80%	1.05G	91.47%	1.05G
	CURL	**83.64%**	1.10G	90.30%	1.11G	79.79%	1.11G	**92.19%**	1.11G

直接在小数据集上剪枝效果不佳

- CURL能够在小数据集上实现良好的压缩效果

小结

- 针对如何更好地剪枝残差结构以及如何在小数据集上进行网络剪枝的问题，我们提出了一种新的模型剪枝方案CURL
- 我们会同时剪枝残差结构的中间层与输出维度，实现更快的运行速度、更高的模型精度，并有效降低了网络的内存消耗
- 为了解决小数据集上训练困难的问题，我们使用图像变换来扩充训练数据集，并使用标记细化的技术来避免标记噪声的干扰

- Jian-Hao Luo and Jianxin Wu. Neural Network Pruning with Residual-Connections and Limited-Data. **In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, Seattle, WA, USA, June 2020, accepted. (CCF-A类会议)

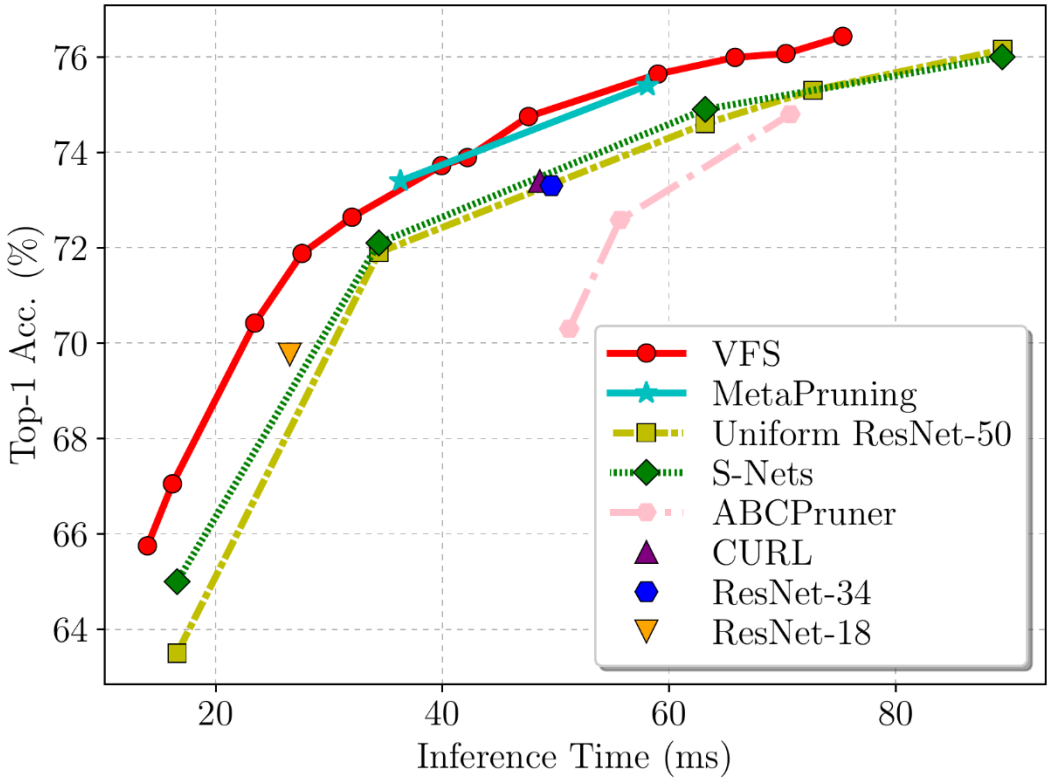
展望未来

多设备模型生成

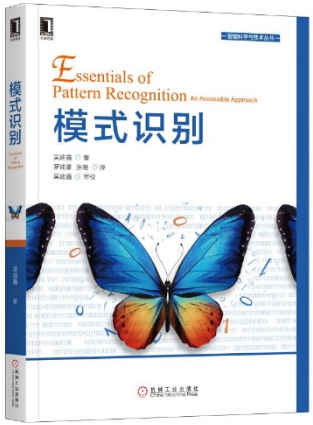
- 问题6：同时压缩滤波器、层、块？多应用压缩？(Versatile)
- 问题7：全光谱快速压缩？（Full-spectrum）
- 问题8：生成时间短、推断速度快？ （Swift）

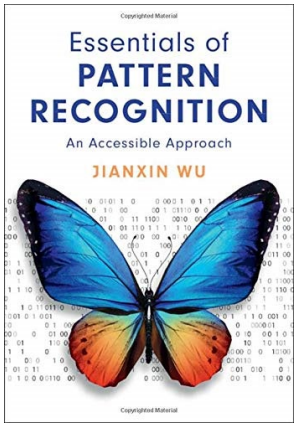
根据实际应用环境的需求
确定需要解决的问题！

VFS!


76
74
72
70
68
66
64
Top-1 Acc. (%)
20
40
60
80
Inference Time (ms)
VFS
MetaPruning
Uniform ResNet-50
S-Nets
ABCPruner
CURL
ResNet-34
ResNet-18

Thank you!


Essentials of
Pattern Recognition
An Accessible Approach
模式识别


Essentials of
PATTERN
RECOGNITION
An Accessible Approach
JIANXIN WU