AI300学院

网络的压缩与加速原理

罗浩
浙江大学

AI300学院

大纲

- 模型压缩与加速概述
- 前端压缩
- 后端压缩

模型压缩与加速概述

深度学习的困境

移动设备：算不好


穿戴设备：算不了


数据中心：算不起

[Valse2018: 深度神经网络加速和压缩年度进展报告(2017) 程健]

AI300学院

模型压缩与加速概述

理论基础

- **必要性**

在许多网络结构中，如VGG-16网络，参数数量1亿3千多万，占用500MB空间，需要进行309亿次浮点运算才能完成一次图像识别任务。

- **可行性**

论文<Predicting parameters in deep learning>提出，其实在很多深度的神经网络中存在着显著的冗余。仅仅使用很少一部分（5%）权值就足以预测剩余的权值。该论文还提出这些剩下的权值甚至可以直接不用被学习。也就是说，仅仅训练一小部分原来的权值参数就有可能达到和原来网络相近甚至超过原来网络的性能（可以看作一种正则化）。

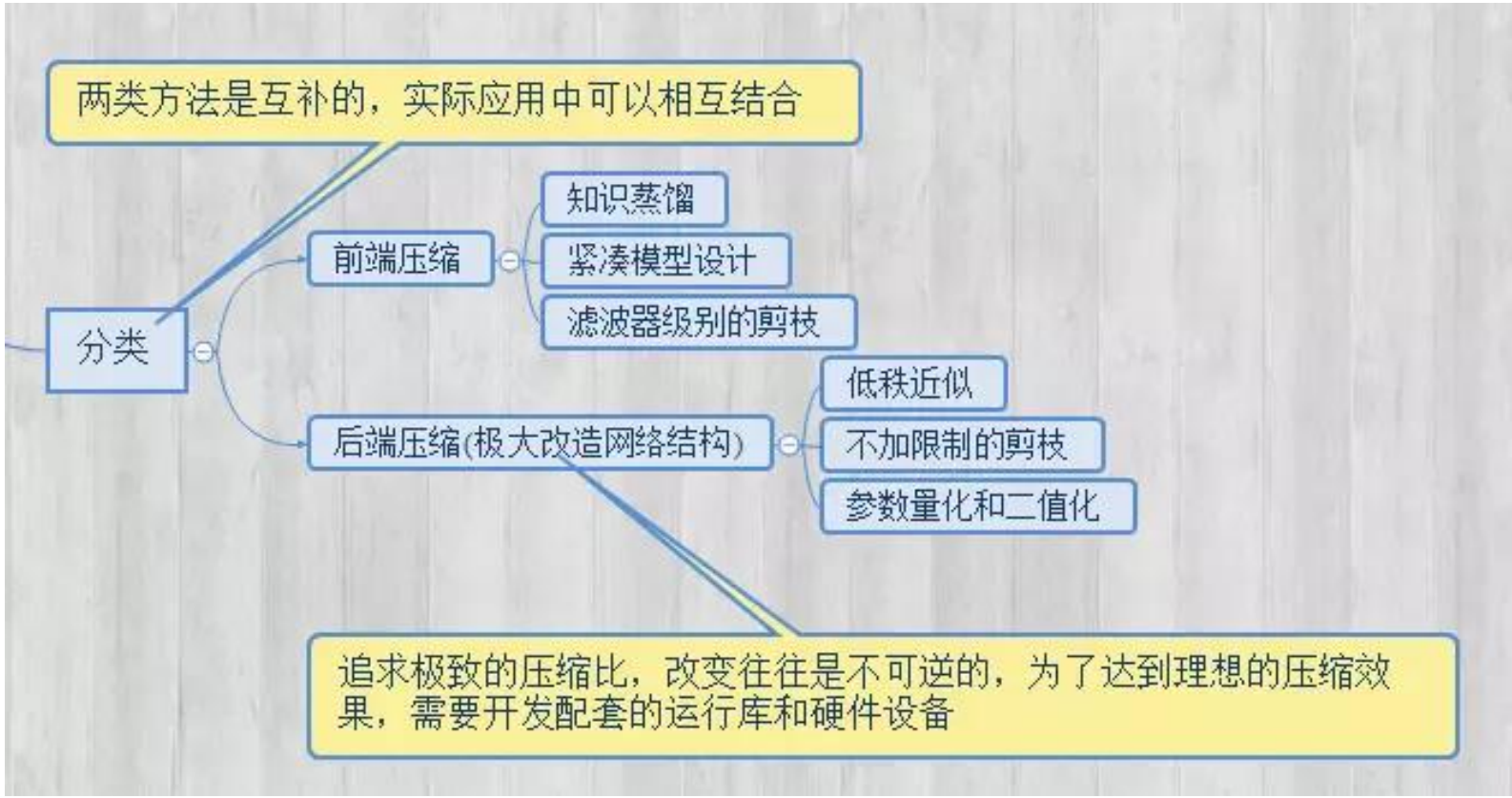
- **最终目的**

最大程度的减小模型复杂度，减少模型存储需要的空间，也致力于加速模型的训练和推测

作者：CodePlayHu
链接：https://www.jianshu.com/p/e73851f32c9f

模型压缩与加速概述

方法概述



作者：CodePlayHu
链接：https://www.jianshu.com/p/e73851f32c9f

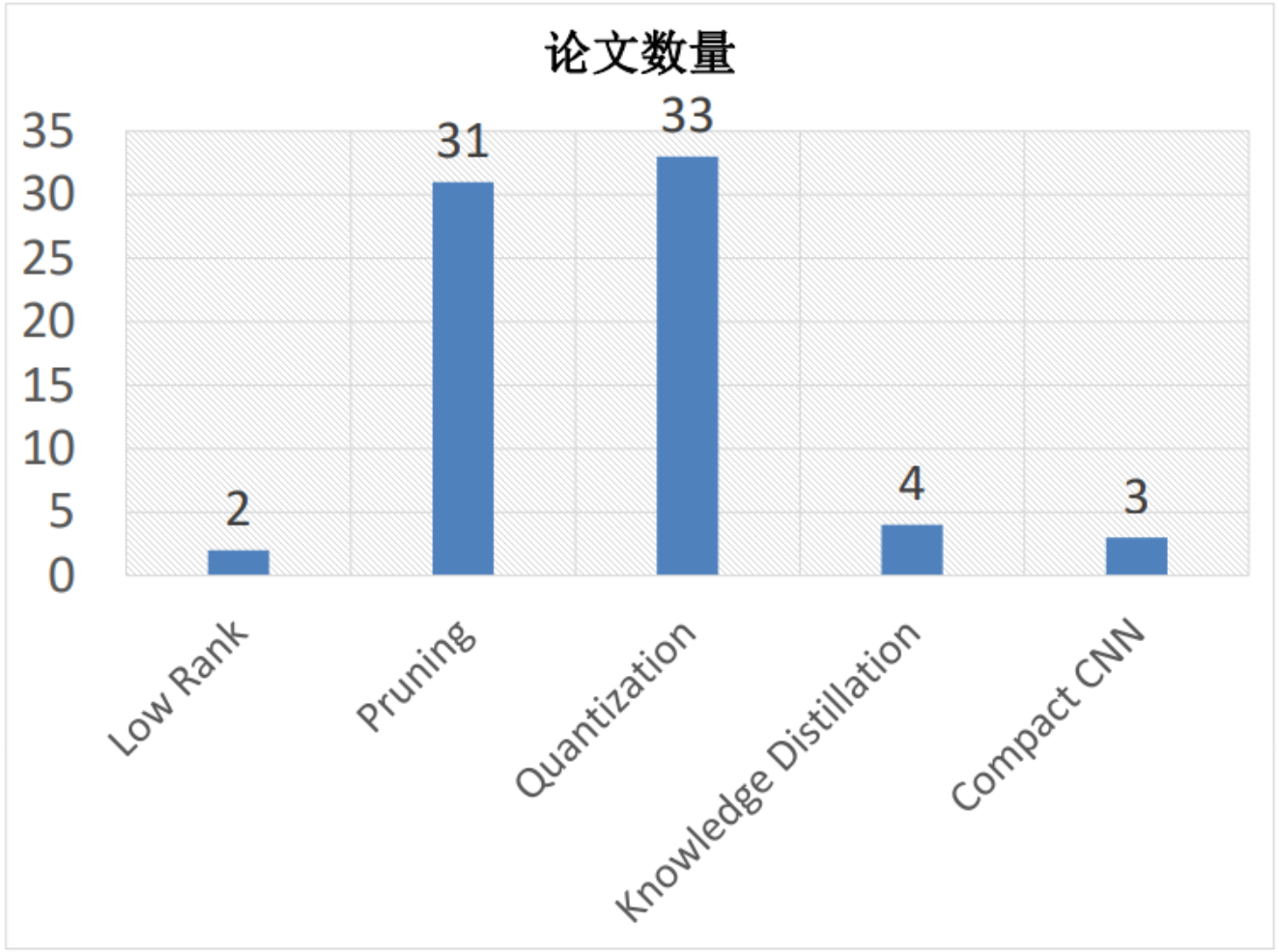
模型压缩与加速概述

各种方法的特点

方法名称	描述	应用场景	方法细节
低秩分解	使用矩阵对参数进行分解估计	卷积层和全连接层	标准化的途径，很容易实施，支持从零训练和预训练
剪枝	删除对准确率影响不大的参数	卷积层和全连接层	对不同设置具有鲁棒性，可以达到较好效果，支持从零训练和预训练
量化	减少表示每个权重所需的比特数来压缩原始网络	卷积层和全连接层	多依赖于二进制编码方式，适合在FPGA、单片机等平台上部署
知识蒸馏	训练一个更紧凑的神经网络来从大的模型蒸馏知识	卷积层和全连接层	模型表现对应用程序和网络结构较为敏感，只能从零开始训练
转移、紧凑卷积核	设计特别的卷积核来保存参数	只有卷积层	算法依赖于应用程序，通常可以取得好的表现，只能从零开始训练

模型压缩与加速概述

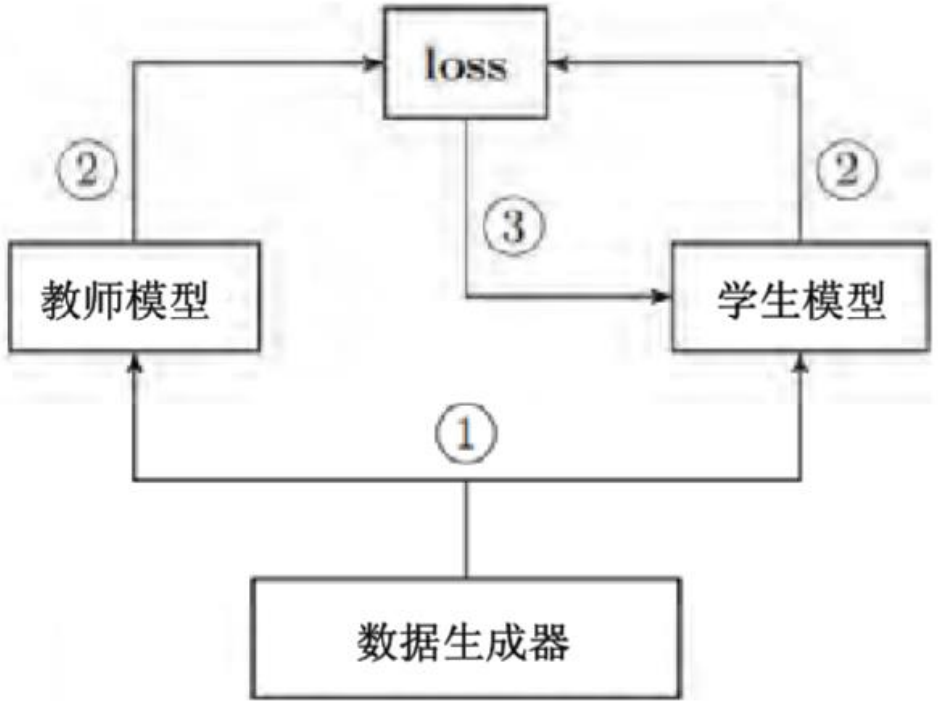
过去一年会议论文分布



- Quantization becomes popular
  - *efficient training using quantization*
  - *low-bit representation*
  - *binary convolutional neural networks*
- Pruning is still a hot topic
  - *small accuracy drop*
  - *efficient structured pruning*
- Few low-rank based method
  - *tensor decomposition is not efficient for current network structure*

前端压缩

知识蒸馏（knowledge distillation）



蒸馏模型采用的是迁移学习，通过采用预先训练好的教师模型（Teacher model）的输出作为监督信号去训练另外一个轻量化的网络（ Student model ）。

教师模型：大&强

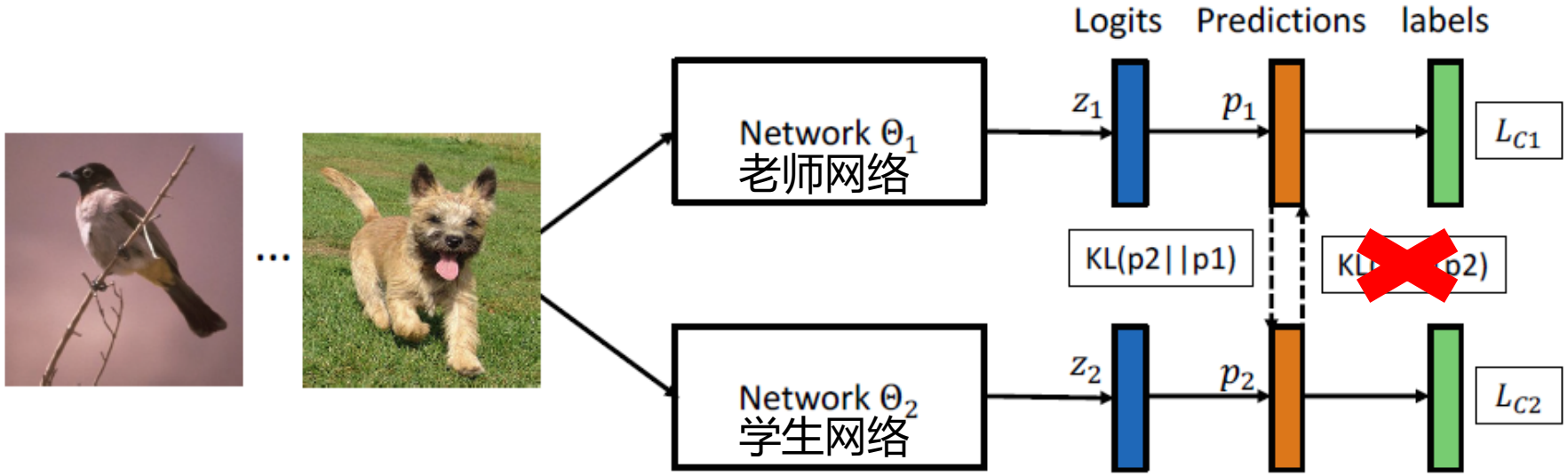
学生模型：小&快

监督的信息可以为：

- 概率分布
- 输出的特征
- 中间层feature map
- Attention map
- 中间过程

知识蒸馏

概率分布——KL散度



$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

$$D_{KL}(\boldsymbol{p}_2\|\boldsymbol{p}_1) = \sum_{i=1}^{N}\sum_{m=1}^{M} p_2^m(\boldsymbol{x}_i)\log\frac{p_2^m(\boldsymbol{x}_i)}{p_1^m(\boldsymbol{x}_i)}.$$

$$L_{\Theta_2} = L_{C_2} + D_{KL}(\boldsymbol{p}_1\|\boldsymbol{p}_2).$$

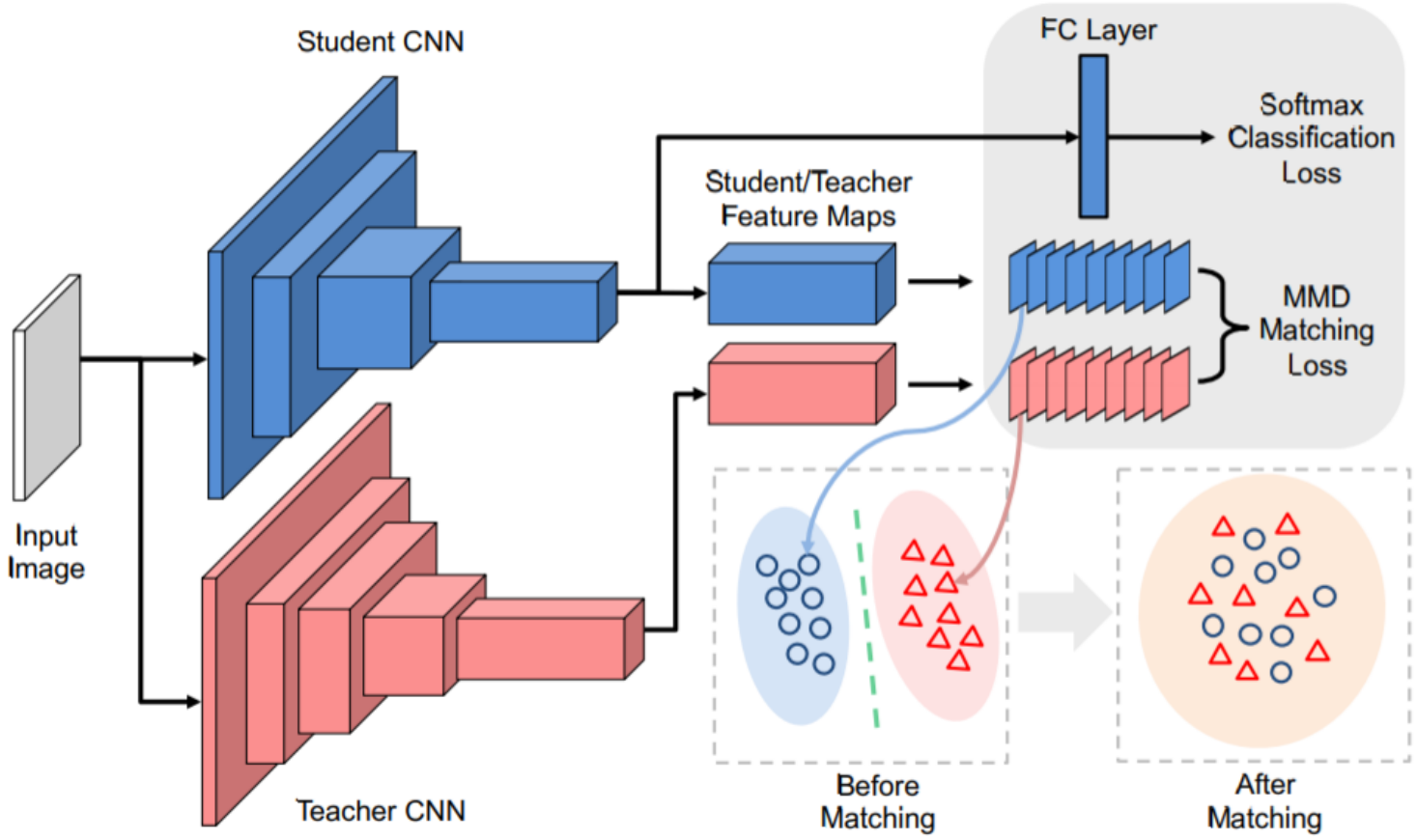
KL散度(Kullback–Leibler divergence)，是描述两个概率分布P和Q差异的一种方法。它是非对称的，在信息论中，D(P||Q)表示当用概率分布Q来拟合真实分布P时

torch.nn.KLDivLoss(*size_average=None, reduce=None, reduction='elementwise_mean'*)

```
loss = nn.KLDivLoss(size_average=False)
L = loss(log_probs2, probs1) / batch_size
```

知识蒸馏

Feature Map——MMD Loss



MMD Loss：最大均值差异（Maximum Mean Discrepancy）损失，可以评价两个域的相似性

$$MMD(X,Y) = ||\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) - \frac{1}{m}\sum_{j=1}^{m}\phi(y_j)||_H^2$$

$\phi()$ 将数据映射到再生希尔伯特空间（RKHS）中进行度量

https://blog.csdn.net/a529975125/article/details/81176029

知识蒸馏

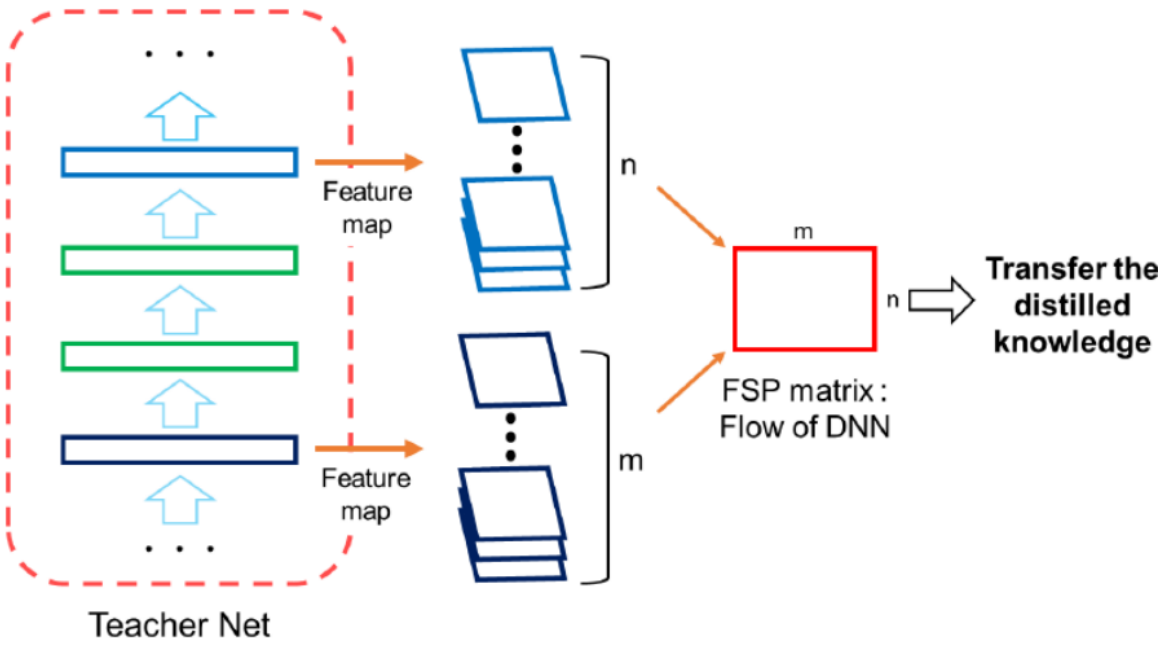
中间过程——L2损失

- Knowledge Definition: (FSP) flow of solution procedure matrix

$$G_{i,j}(x;W) = \sum_{s=1}^{h}\sum_{t=1}^{w} \frac{F^1_{s,t,i}(x;W) \times F^2_{s,t,j}(x;W)}{h \times w}$$

- Loss Definition: L2 loss

$$L_{FSP}(W_t, W_s) = \frac{1}{N}\sum_{x}\sum_{i=1}^{n} \lambda_i \times \|(G^T_i(x;W_t) - G^S_i(x;W_s)\|^2_2,$$

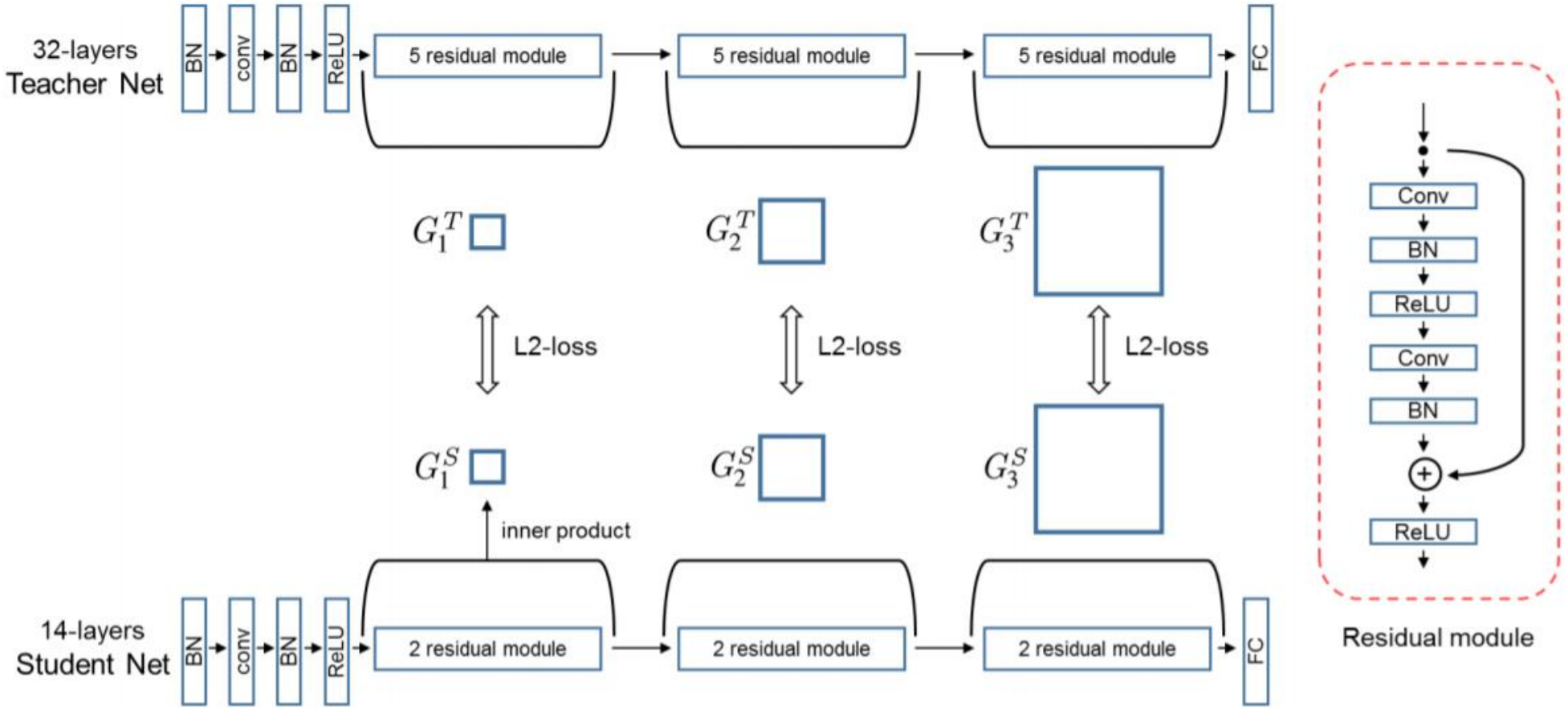


利用Gram矩阵来拟合层与层之间的关系

torch.nn.MSELoss(*size_average=None, reduce=None, reduction='elementwise_mean'*)

知识蒸馏

中间过程——L2损失



Yim J, Joo D, Bae J, et al. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning[C]//The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017, 2.

前端压缩

紧致的网络结构设计

常见的紧致网络：

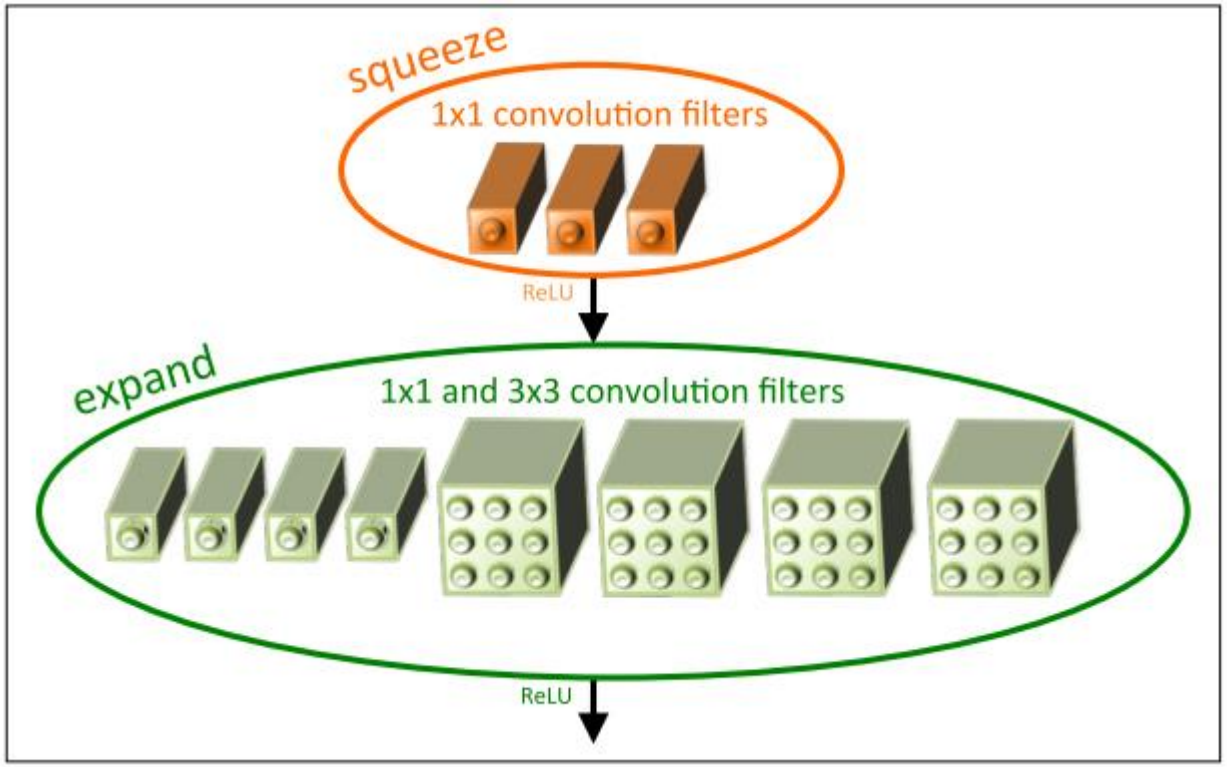
- ShuffleNet
- MobileNet v1
- MobileNet v2
- SqueezeNet

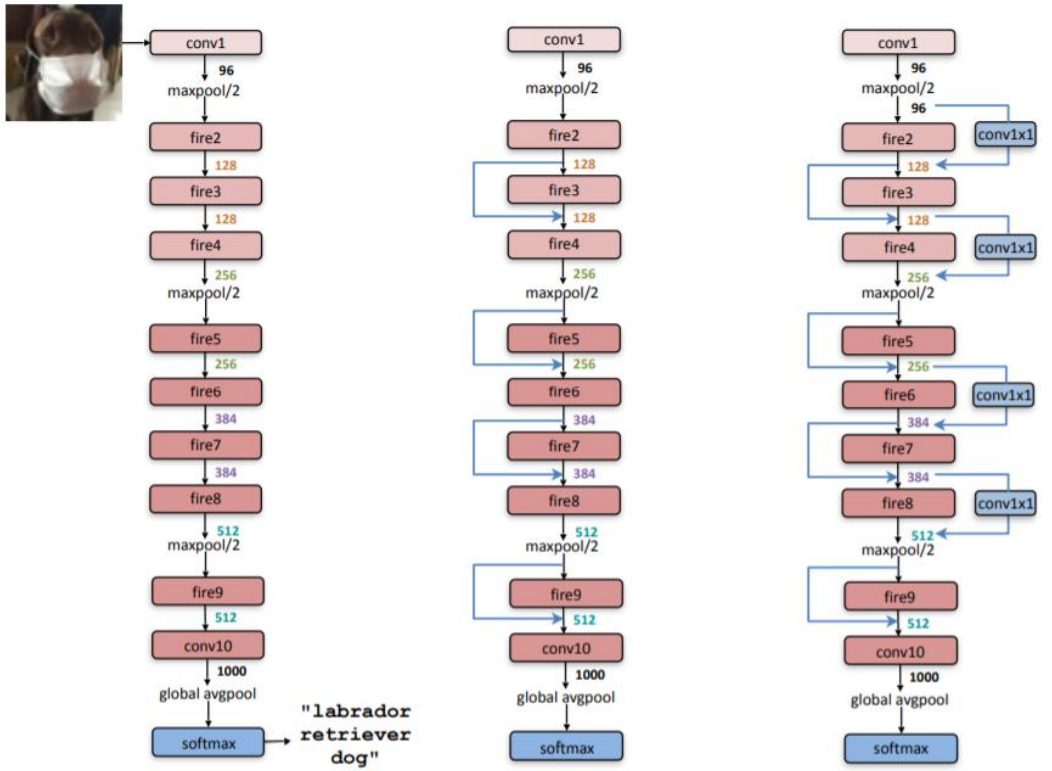
Model	
SqueezeNet	1*1 filters , input channels , Downsample late
MobileNet v1	DWconv + Pconv ,Width Multiplier ,Resolution Multiple
MobileNet v2	Inverted residuals ,Linear bottlenecks
ShuffleNet	DWconvolution with Channel Shuffle

直接设计又小又快又好的网络，这就是紧致网络设计（Compact Network Design）的方法

紧致的网络结构设计

SqueezeNet

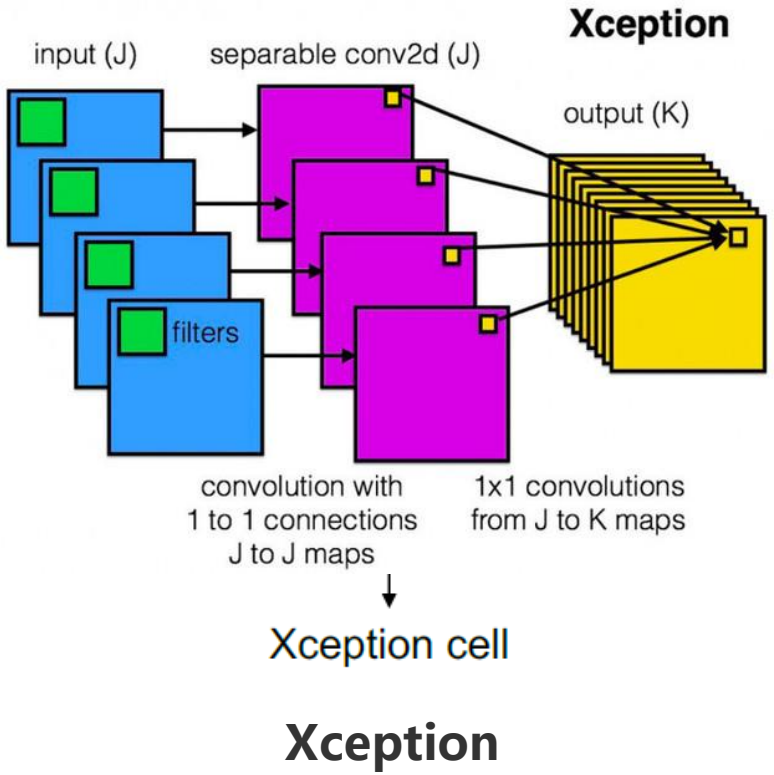




1.使用更小的1*1卷积核来替换3*3卷积核
2.减少输入3*3卷积的特征图的数量
3.减少pooling

紧致的网络结构设计

MobileNet v1



Xception

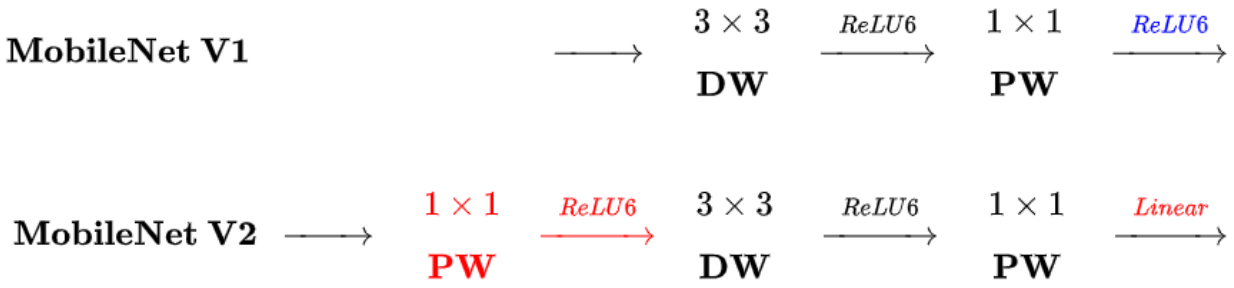
Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

- MobileNet v1大量使用3×3 Depthwise卷积和1×1卷积组合的Xception结构

紧致的网络结构设计

MobileNet v2



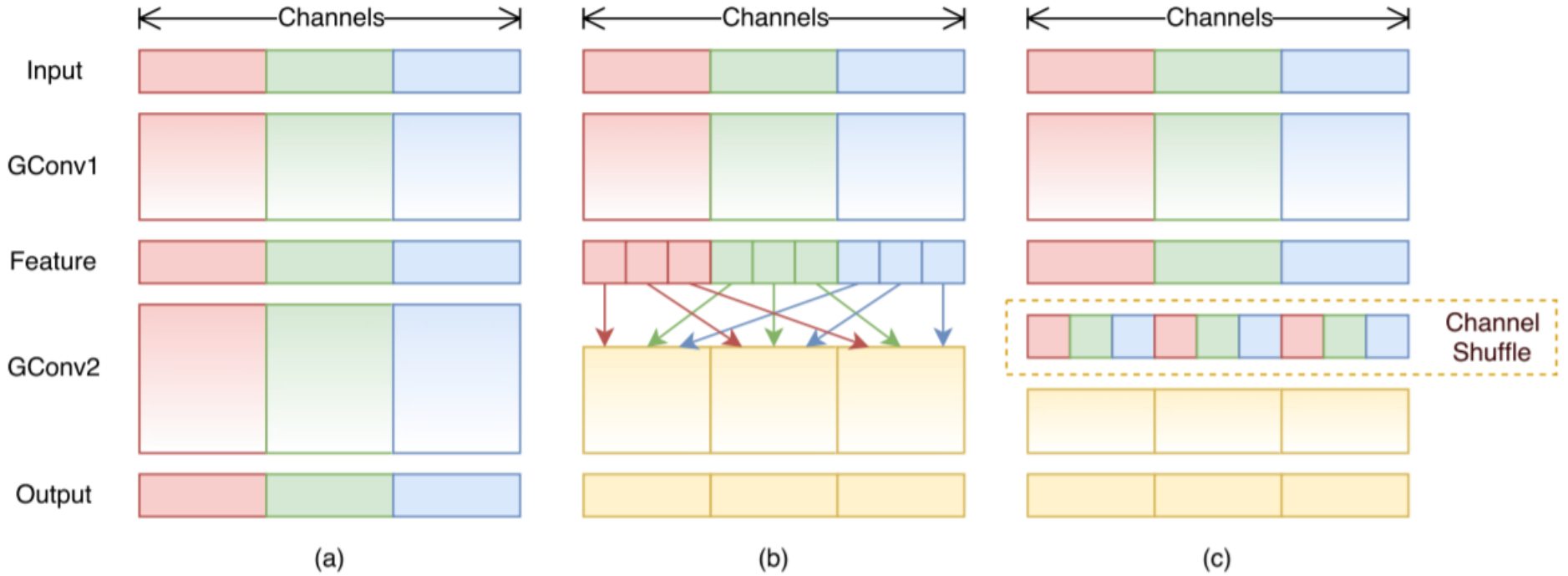
Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$28^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times k$	conv2d 1x1	-	k	-	

- V2 在 DW 卷积之前新加了一个 1×1卷积，增加网路通道数。
- V2 去掉了第二个 PW 的激活函数，防止特征丢失。
- V2增加了ResNet的shortcut结构

- 通过改变网络的参数设置，V2比V1进一步压缩了约50%，并且泛化能力提升。

紧致的网络结构设计

ShuffleNet



- ShuffleNet使用通道打乱操作来代替1×1卷积，实现通道信息的融合
- Shuffle操作通过Reshape操作实现，不包含参数

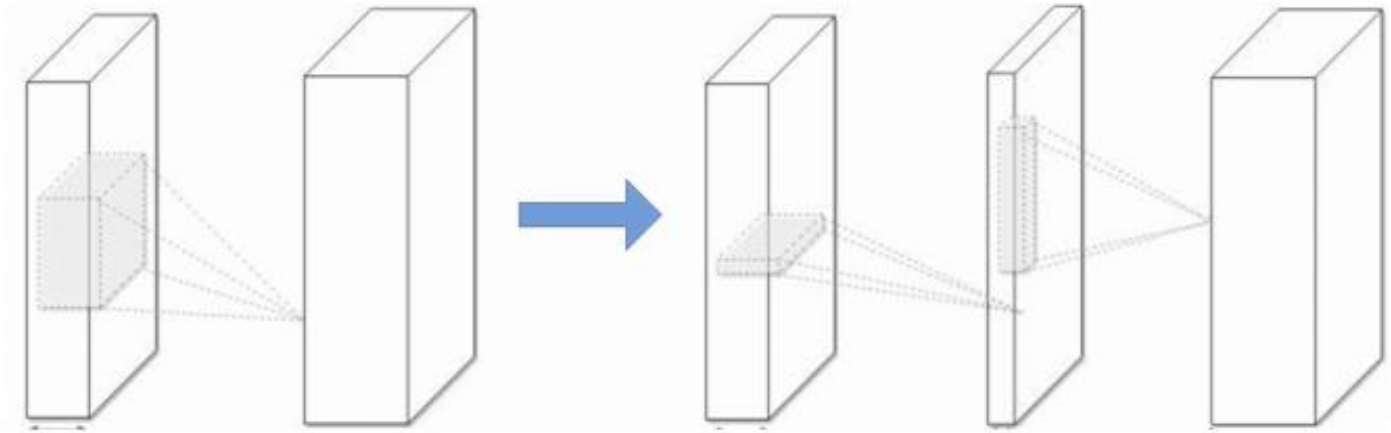
紧致的网络结构设计

紧致的网络结构使用情况对比

Model	Caffe	Tensorflow	Keras	PyTorch	Migration network	Recommendation level
SqueezeNet	1★	4★	2★	2★	AlexNet,faster-rcnn	3★
MobileNet v1	4★	5★	3★	3★	Mobilenet-SSD, MXNet,faster-rcnn	3★
MobileNet v2	3★	5★	2★	4★	MobileNetv2-SSDLite	5★
ShuffleNet	5★	4★	2★	3★	Shufflenet-SSD,	4★

后端压缩

低秩近似（Low Rank）



一个典型的 CNN 卷积核是一个 4D 张量，需要注意的是这些张量中可能存在大量的冗余。而基于张量分解的思想也许是减少冗余的很有潜力的方法。而全连接层也可以当成一个 2D 矩阵，低秩分解同样可行。

低秩近似

卷积分解

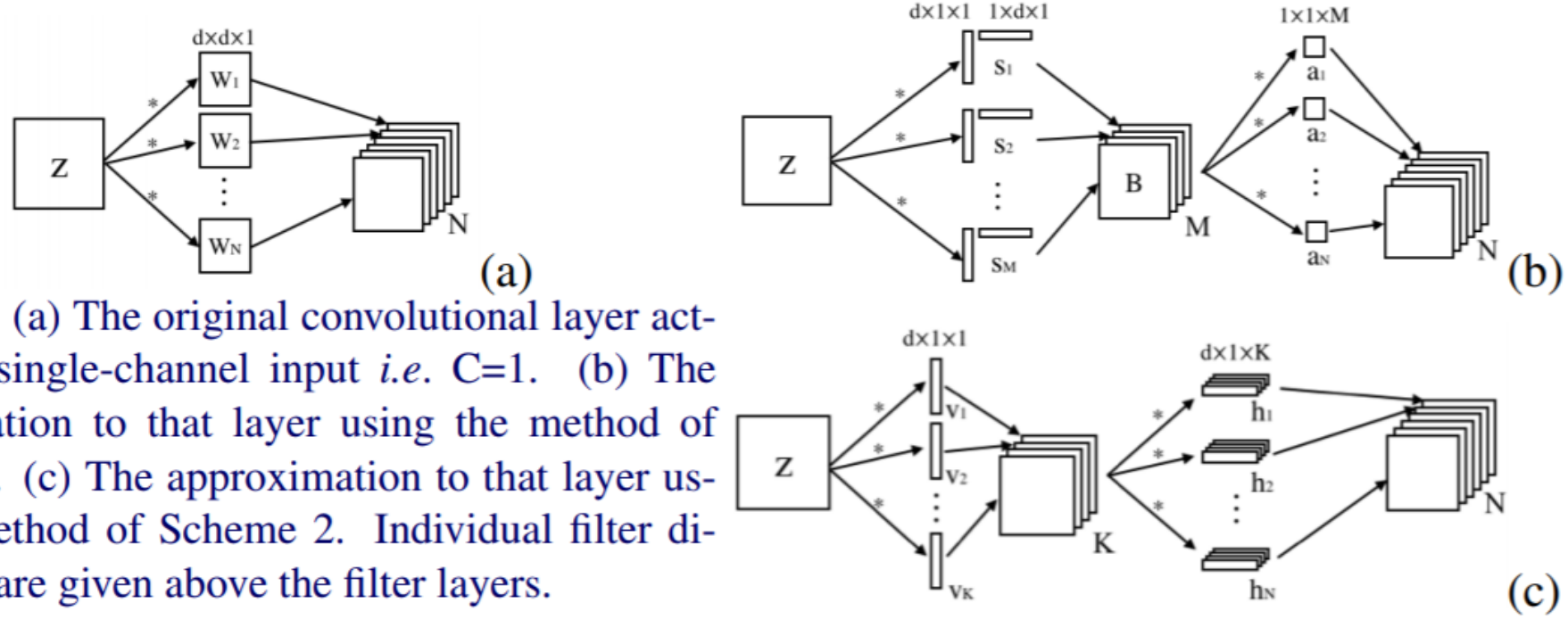


Figure 1: (a) The original convolutional layer acting on a single-channel input *i.e.* C=1. (b) The approximation to that layer using the method of Scheme 1. (c) The approximation to that layer using the method of Scheme 2. Individual filter dimensions are given above the filter layers.

- 将d×d卷积核分解为d×1和1×d卷积核
- 损失精度小于1%

低秩近似

SVD分解

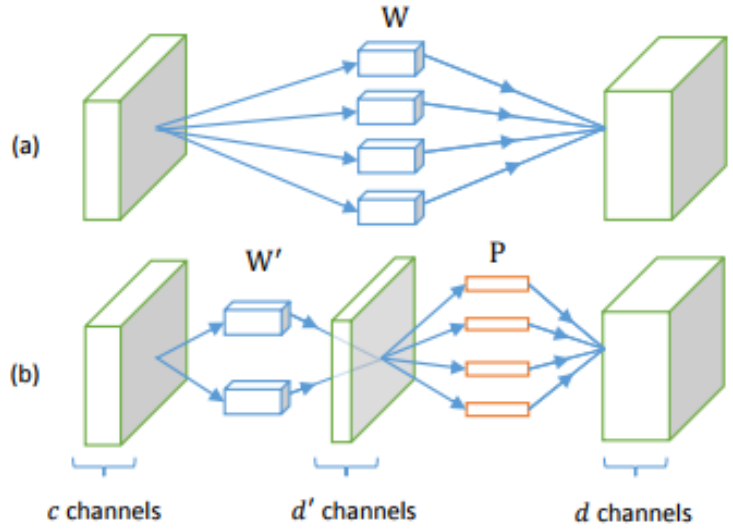


Figure 1: Illustration of the decomposition. (a) An original layer with complexity $O(dk^2c)$. (b) An approximated layer with complexity reduced to $O(d'k^2c) + O(dd')$.

SVD分解

- 对矩阵进行SVD分解，保留特征值最大的几项

$$\mathbf{y} = \mathrm{M}(\mathbf{y} - \bar{\mathbf{y}}) + \bar{\mathbf{y}}$$

$$\mathbf{y} = \mathrm{MW}\mathbf{x} + \mathbf{b},$$

$$\mathbf{b} = \bar{\mathbf{y}} - \mathrm{M}\bar{\mathbf{y}}$$

$$\mathrm{M} = \mathrm{PQ}^\top$$

$$\mathrm{M} = \mathrm{U}_{d'}\mathrm{S}_{d'}\mathrm{V}_{d'}{}^\top$$

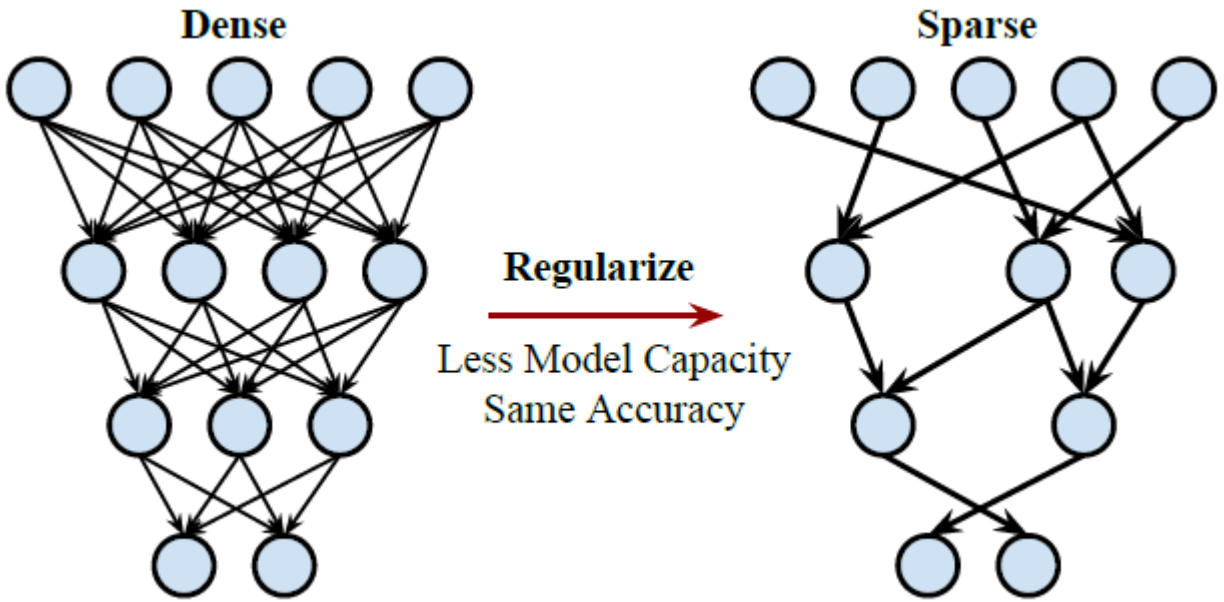
$$\mathrm{P} = \mathrm{U}_{d'}\mathrm{S}_{d'}^{1/2} \text{ and } \mathrm{Q} = \mathrm{V}_{d'}\mathrm{S}_{d'}^{1/2}$$

$$\min_{\mathrm{M}} \sum_i \|(\mathbf{y}_i - \bar{\mathbf{y}}) - \mathrm{M}(\mathbf{y}_i - \bar{\mathbf{y}})\|_2^2,$$

$$s.t. \quad rank(\mathrm{M}) \leq d'.$$

后端压缩

剪枝（Pruning）



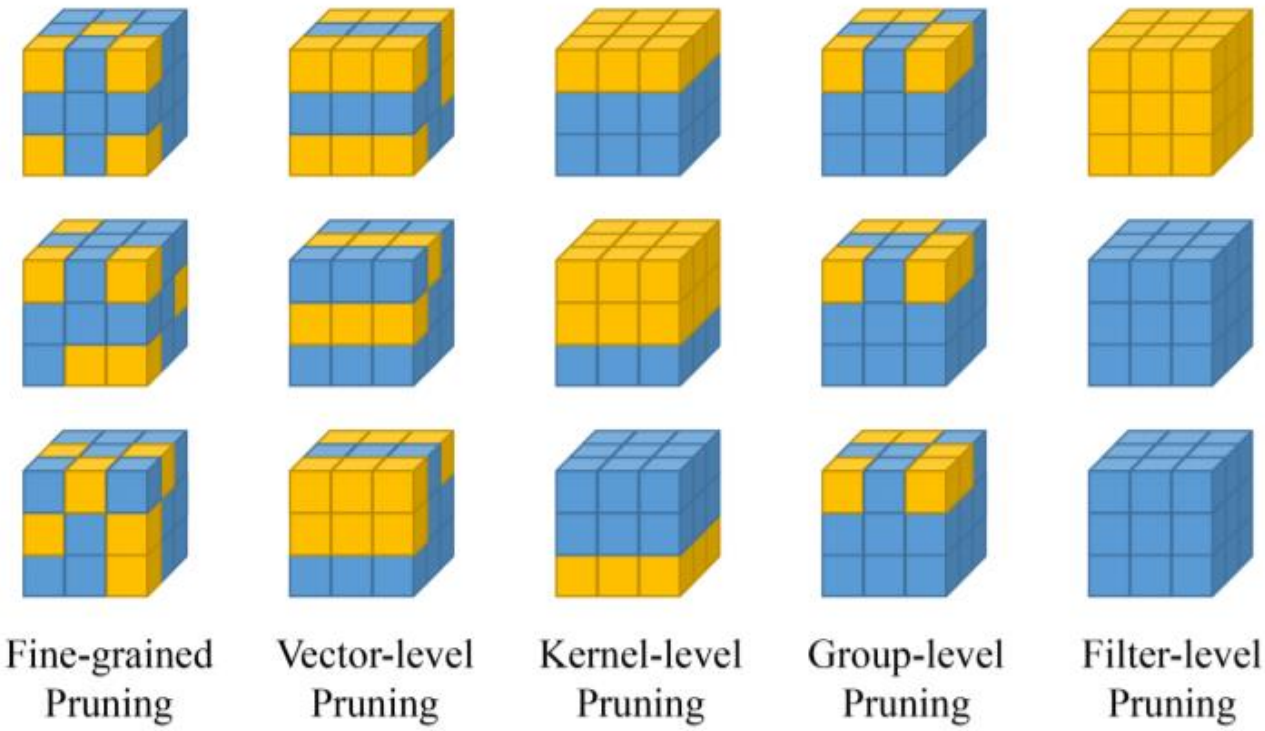
网络训练完成后，裁剪掉贡献信息量不大的网络参数。如果权值越少，就认为经过激活之后所产生的影响几乎可以忽略不计。去除权重比较小的连接，使得网络稀疏

剪枝的基本流程：

1. 衡量神经元的重要程度
2. 移除一部分不重要的神经元
3. 对网络进行微调
4. 返回第一步，进行下一轮剪枝

后端压缩

剪枝（Pruning）



剪枝类型主要有：

- 细粒度剪枝
- 向量级别剪枝
- 核级别剪枝
- 组级别剪枝
- 滤波器级别剪枝

剪枝

剪枝方法分类

- **权重衰减法**

  通过在网络目标函数中引入表示结构复杂性的正则化项来使训练的网络权重趋向稀疏化。缺点是正则化参数/权重剪枝阈值对剪枝结果影响很大

- **灵敏度量化法**

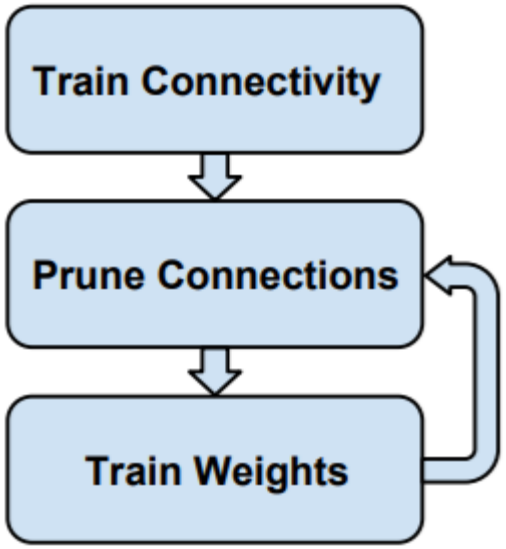
  通过设计一个灵敏度评价指标，衡量网络节点对于网络误差的贡献（灵敏度），删除那些贡献不大的节点。

- **根据相关性剪枝**

  根据网络节点间的相关性或者相互作用进行剪枝，然后合并相关性较大的节点。类似于PCA一样，去除网络中信息冗余的网络节点。

剪枝

权重衰减法



L1正则化： $C = C_0 + \frac{\lambda}{n} \sum_w |w|.$

L2正则化： $C = C_0 + \frac{\lambda}{2n} \sum_w w^2,$

基本流程：

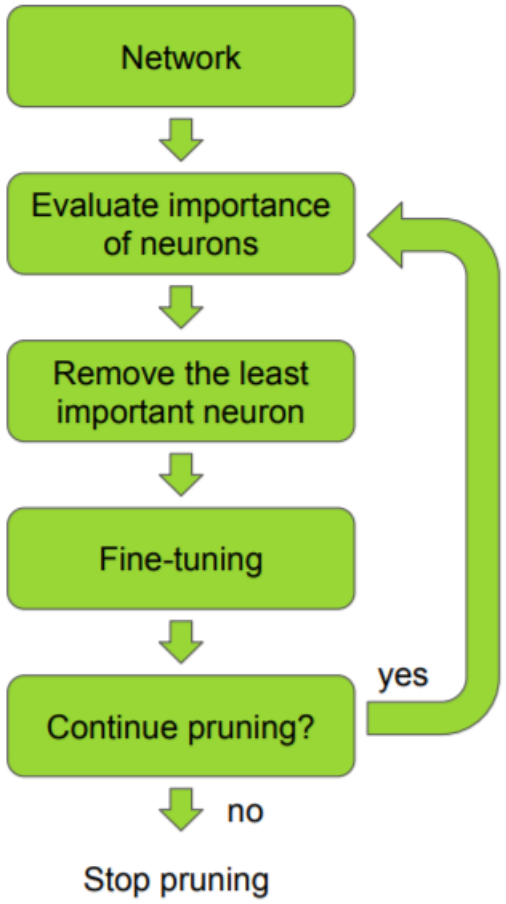
1. 在网络损失函数中加入正则化项保证网络稀疏
2. 设置一个阈值，删除网络权重低于阈值的网络节点
3. 对网络进行训练
4. 返回第一步，进行下一轮剪枝

缺点：

1. 传统方法需要在多个待测试阈值上重复迭代
2. 权重阈值在所有层共享，难以寻找合适的阈值
3. 对硬件支持并不友好

剪枝

灵敏度量化法



1. 对于网络节点h，我们假设其被裁剪时(h=0)，对网络损失的影响
2. 在h=0处进行泰勒展开

$$\mathcal{C}(\mathcal{D}|\mathcal{W},h=0) = \mathcal{C}(\mathcal{D},\mathcal{W}) - \frac{\delta\mathcal{C}}{\delta h}h + R_1(h=0)$$

3. 得到网络节点h对于网络损失的灵敏度表达

$$\left|\mathcal{C}(\mathcal{D}|\mathcal{W},h=0) - \mathcal{C}(\mathcal{D}|\mathcal{W})\right| = \left|\mathcal{C}(\mathcal{D}|\mathcal{W}) - \frac{\delta\mathcal{C}}{\delta h}h - \mathcal{C}(\mathcal{D}|\mathcal{W})\right| = \left|\frac{\delta\mathcal{C}}{\delta h}h\right|.$$

4. 迭代删除重要度最低的filter，并进行fine tune

剪枝

根据相关性剪枝

第 i 个隐层节点和第 j 个隐层节点的相关度公式为：

$$R_{ij}=\frac{\sum_p v_{ip}v_{jp}-pv_iv_j}{\left(\sum_p(v_{ip}-\overline{v_i})^2\right)^{\frac{1}{2}}\left(\sum_p(v_{jp}-\overline{v_j})^2\right)^{\frac{1}{2}}}$$

各节点的输出的方差公式如下：

$$s_i^2=\sum_p(v_{ip}-\overline{v_i})^2$$

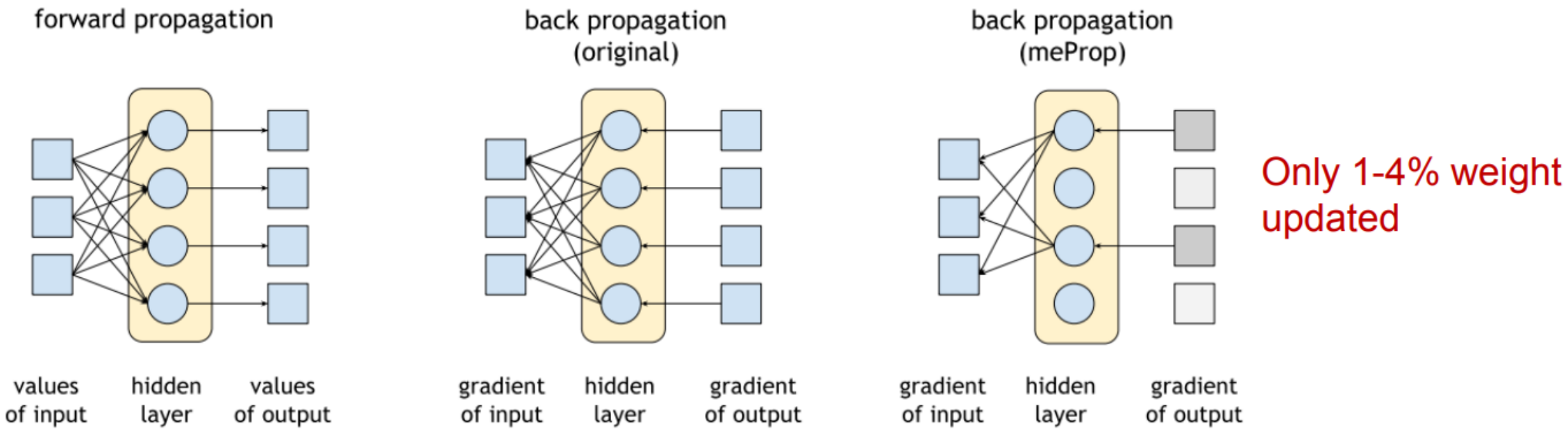
- 按以上公式计算后输出序列呈以下 5 种情况：
  1. 两个隐节点输出序列高度正相关；→ 合并
  2. 两个隐节点输出序列高度负相关；→ 合并
  3. 两个隐节点输出序列相关性不高；
  4. 某隐节点输出序列方差较小； → 近似为常数
  5. 某隐节点输出序列方差较大。

- 可以处理的是线性相关度高的以及方差较小的隐层 节点。

剪枝

其他一些方法

- Keep top-k gradients of neural nodes , and prune rest of them
- Since some neural nodes' gradients are zeros, back propagation can be accelerated.

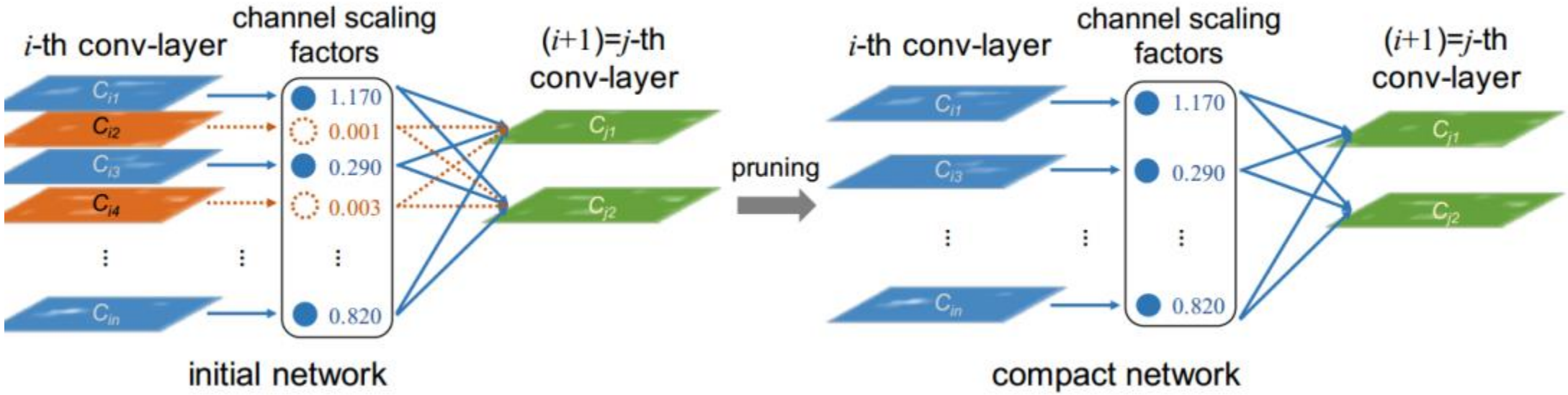


对梯度进行剪枝，每次只更新梯度大的神经元

剪枝

其他一些方法

- Associate a scaling factor with each channel
- Impose sparsity regularization on these scaling factors
- Prune those channels with small scaling factors



通过设计一个尺度因子，实现滤channel维度上的剪枝

后端压缩

量化（Quantization）

一般而言，神经网络模型的参数都是用的32bit长度的浮点型数表示，实际上不需要保留那么高的精度，可以通过量化，比如用0~255表示原来32个bit所表示的精度，通过牺牲精度来降低每一个权值所需要占用的空间。此外，SGD（Stochastic Gradient Descent）所需要的精度仅为6~8bit，因此合理的量化网络也可保证精度的情况下减小模型的存储体积。

- 二值量化（+1，-1）
- 三值量化（+1，-1，0）
- 多值量化（8bit，16bit等）

作者：CodePlayHu
链接：https://www.jianshu.com/p/e73851f32c9f

量化的对象有：

- 网络权重
- 网络特征
- 网络梯度

量化

二值神经网络（Binary-Weight-Network）

- 基于符号函数Sign的确定性(deterministic)方法：

$$x^b = \text{Sign}(x) = \begin{cases} +1 & \text{if } x \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

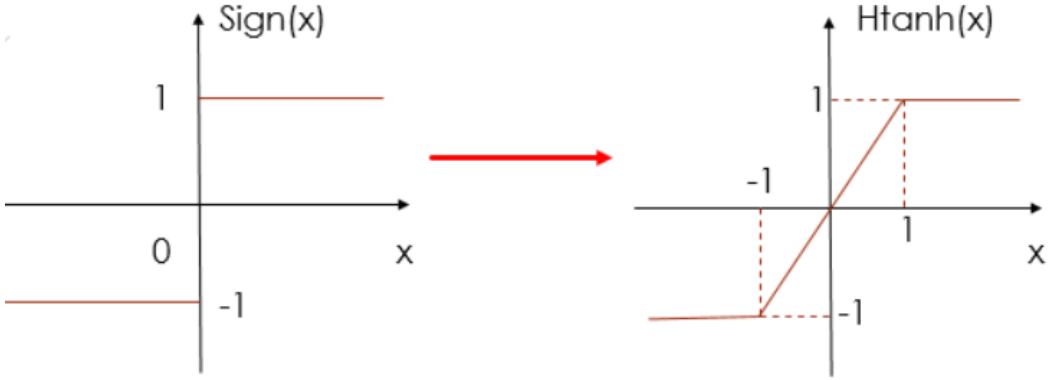
- 基于概率分布的随机二值化(stochastic)方法：

$$x^b = \begin{cases} +1 & \text{with probability } p = \sigma(x), \\ -1 & \text{with probability } 1 - p, \end{cases}$$

$$\sigma(x) = \text{clip}(\frac{x+1}{2}, 0, 1) = \max(0, \min(1, \frac{x+1}{2}))$$

- 对符号函数进行松弛求解来得到梯度：

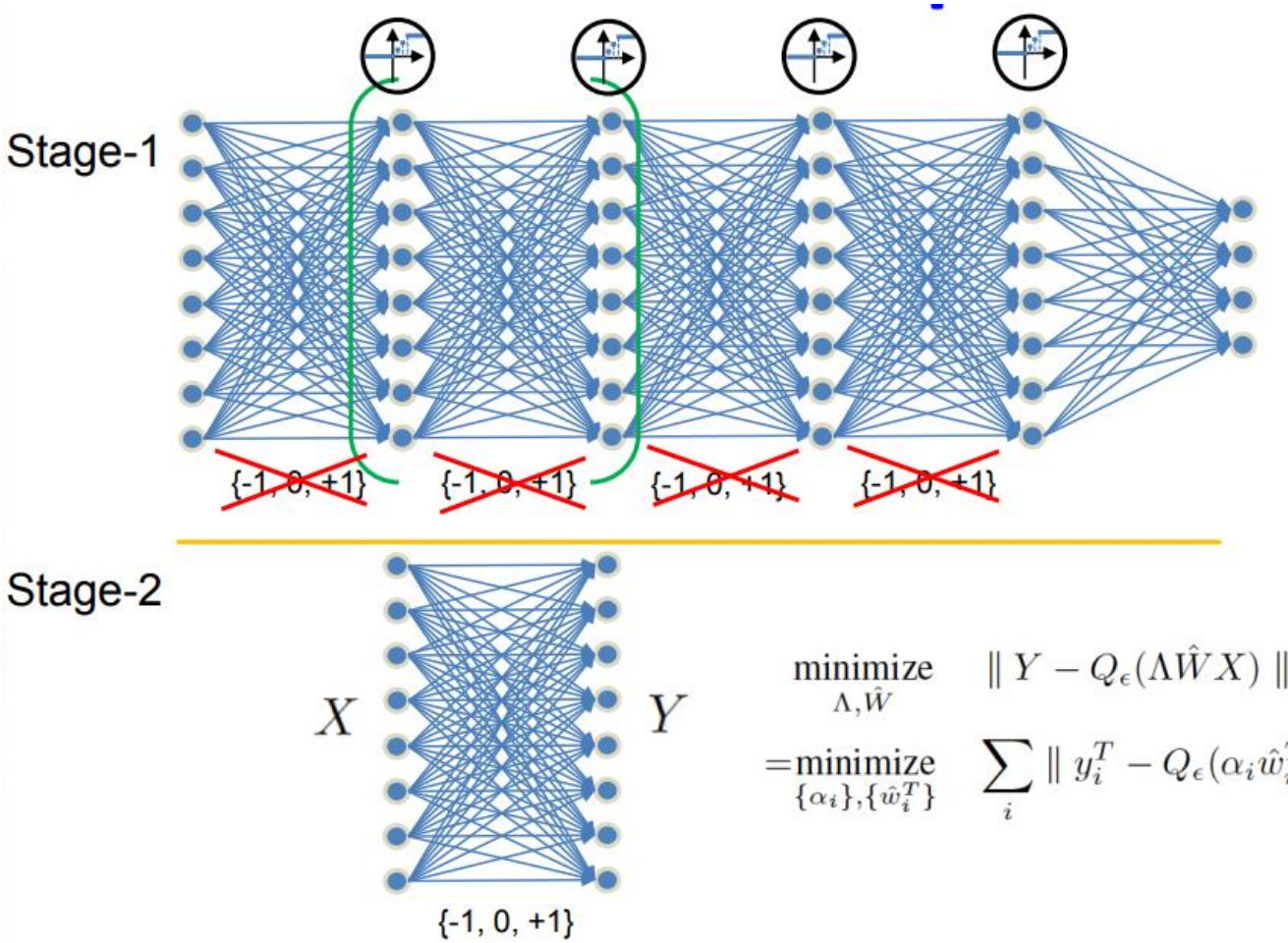
$$\text{Htanh}(x) = \text{Clip}(x, -1, 1) = \max(-1, \min(1, x)).$$



Courbariaux M, Hubara I, Soudry D, et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1[J]. arXiv preprint arXiv:1602.02830, 2016.

量化

三值量化——示例



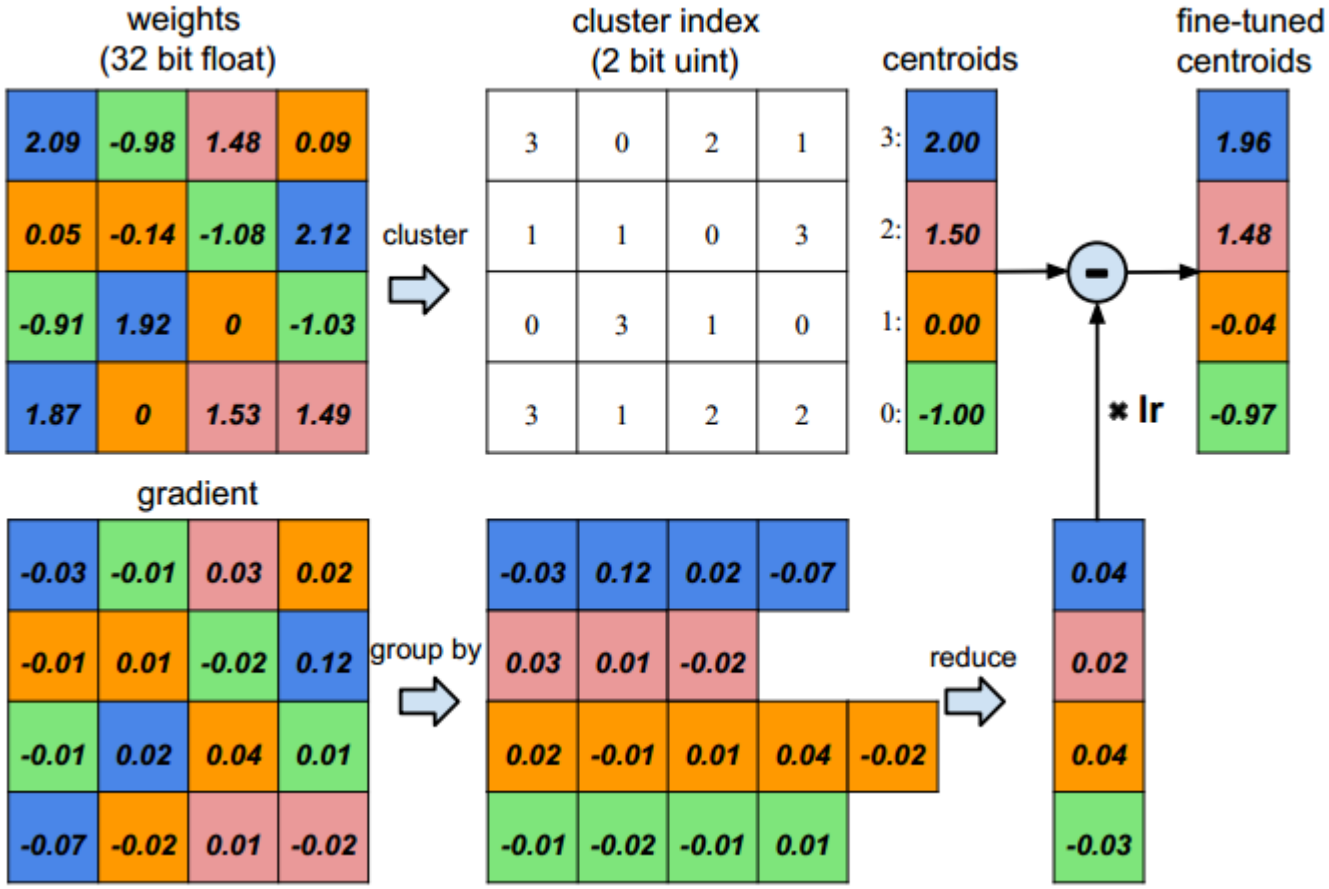
$$\underset{\Lambda,\hat{W}}{\text{minimize}} \quad \| Y - Q_\epsilon(\Lambda \hat{W} X) \|_F^2$$

$$= \underset{\{\alpha_i\},\{\hat{w}_i^T\}}{\text{minimize}} \quad \sum_i \| y_i^T - Q_\epsilon(\alpha_i \hat{w}_i^T X) \|_2^2$$

- Quantize both weight and actition is diffifult
- Inspired by Two-Step Hashing, Two-Step quantization method first quantizes the activation, then quantize the weights

量化

其他方法



- 先对权重、梯度进行聚类
- 根据聚类结果生成索引
- 根据索引得到结果

量化

网络量化对比

Method	Quantization			Acceleration	
	Weight	Activation	Gradient	Training	Testing
BinaryConnect [10]	Binary	Full	Full	No	Yes
BWN [65]	Binary	Full	Full	No	Yes
BWNH [32]	Binary	Full	Full	No	Yes
TWN [48]	Binary	Full	Full	No	Yes
FFN [81]	Ternary	Full	Full	No	Yes
INQ [99]	Ternary-5bit	Full	Full	No	Yes
BNN [65]	Binary	Binary	Full	No	Yes
XNOR [65]	Binary	Binary	Full	No	Yes
HWGQ [4]	Binary	2bit	Full	No	Yes
DoReFa-Net [100]	Binary	1-4bit	6bit, 8bit, Full	Yes	Yes

AI300学院

课后思考

1. 有哪些方法比较适合于嵌入式芯片部署
2. 无监督与自监督网络压缩方法在未来是否会出现
3. 阅读两篇相关综述：

- 2017-A Survey of Model Compression and Acceleration for Deep Neural Networks
- 2018-Recent Advances in Efficient Computation of Deep Convolutional Neural Networks

AI300学院

欢迎关注AI300学院