Network Compression

> 以下技术只是在软件上进行压缩，不考虑硬件加速的部分。

1. Network Pruning (网络剪枝)
2. Knowledge Distillation (知识蒸馏)
3. Parameter Quantization (参数量化)
4. Architecture Design (结构设计)
5. Dynamic Computation (动态计算)

1 - Network Pruning

> 原理： 树大必有枯枝，将大的网络中没有用的那些参数找出来

Paper : Optimal Brain Damage

做法： 每次都去修剪少量的参数或者神经元

效果： 将大网络变成小网络，然后正确率(性能)不会差很多

具体步骤：

1. 首先训练一个大的(或者超大的)网络，且性能达到要求。
2. 去评价重要性，这里有两种
    1. 评价每一个参数的重要性

        这里如何去评估参数重不重要呢？方法：

        1 - 看它的绝对值大小

        2 - Life Long

        3 - 等等
    2. 评价每一个神经元的重要性

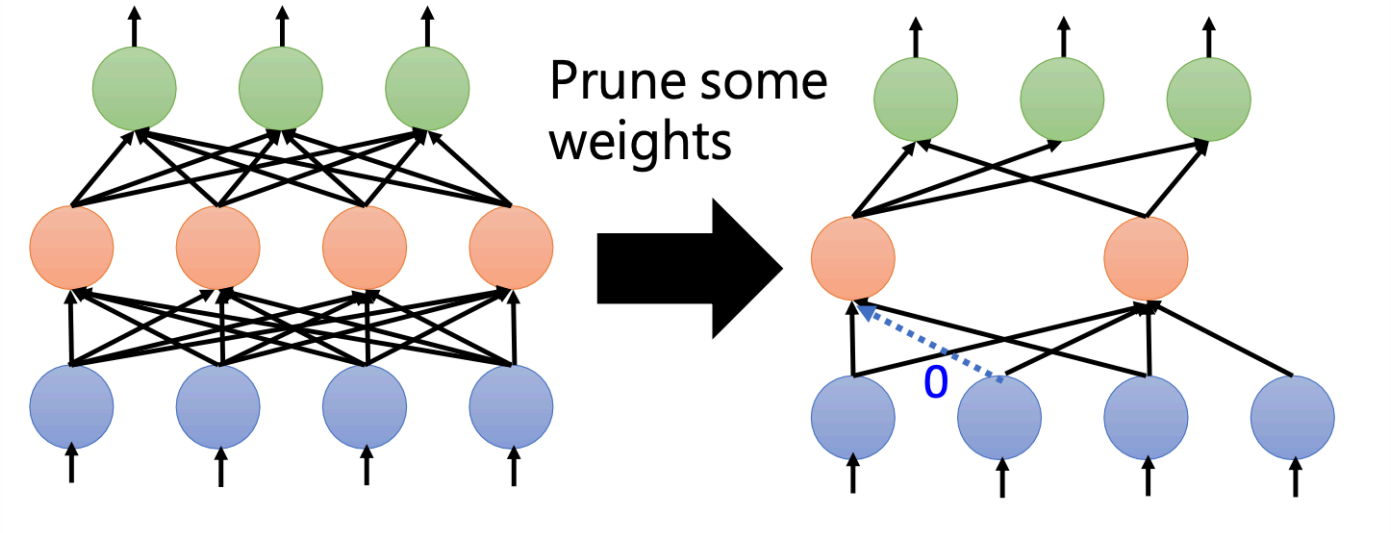
        1 - 计算当前神经元输出不为零的次数

        2 - 等
3. 对于评估出来的不重要的神经元或者参数移除掉
4. 对于步骤3得到的网络进行 Fine-tune (微调)
5. 重复若干次步骤2到步骤4，直到得到满意的网络

如果一次性直接剪掉大量的参数，可能对网络的影响太大了，导致就算使用 Fine-tune 也没办法复原到原来差不多的性能。

重新考虑步骤2

以参数为单位进行评估和裁剪

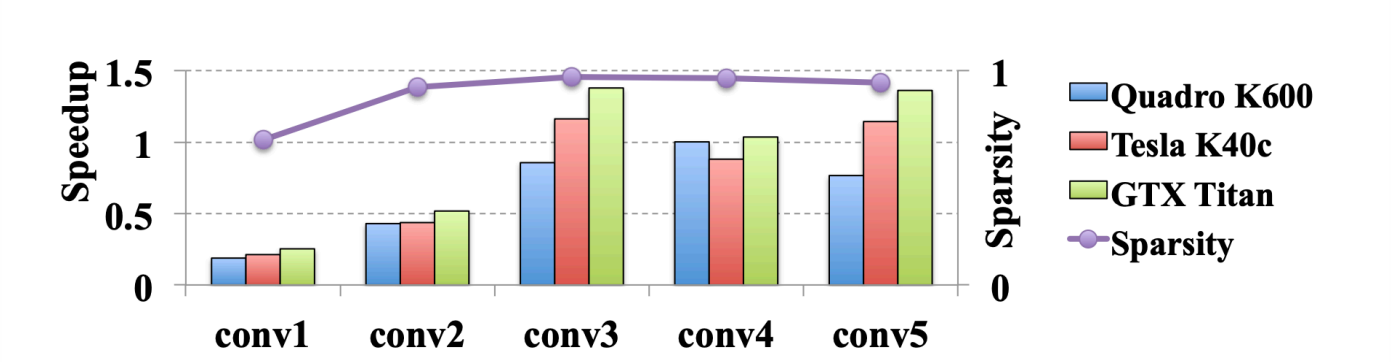


裁剪掉部分不重要的参数后，网络形状变得不规则。会导致一系列的问题：

1. 代码不好写。因为在定义网络模型的时候是固定数量的神经元。
2. 不规则形状的网络很难利用GPU进行加速训练。(加速是指的GPU将网络当作一个矩阵进行计算)

如果对裁剪掉的参数直接补 0 的话，就可以解决网络形状改变的问题。虽然这么做可以比较容易用GPU加速，但是根本没有达到网络变小的效果。

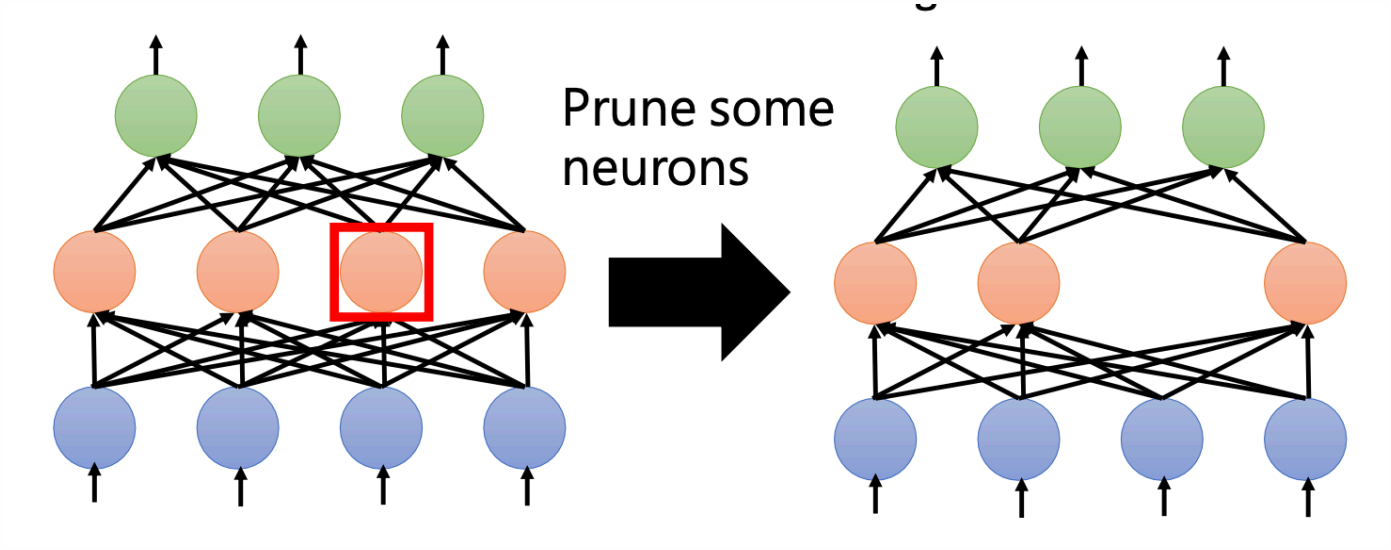
（Learning Structured Sparsity in Deep Neural Networks）



紫色的线 表示有多少百分比的参数被Prune掉了。(可以看到接近95%的参数都被去掉了)

三条 蓝绿红 的竖线表述：在三中不同计算资源上的加速程度。大于1的时候才有加速的效果，小于1其实是变慢了速度。可以发现，在多数情况下根本就没有加速。

以神经元为单位进行评估和裁剪



以神经元为单位会更好实现一些，也能用得上GPU加速。

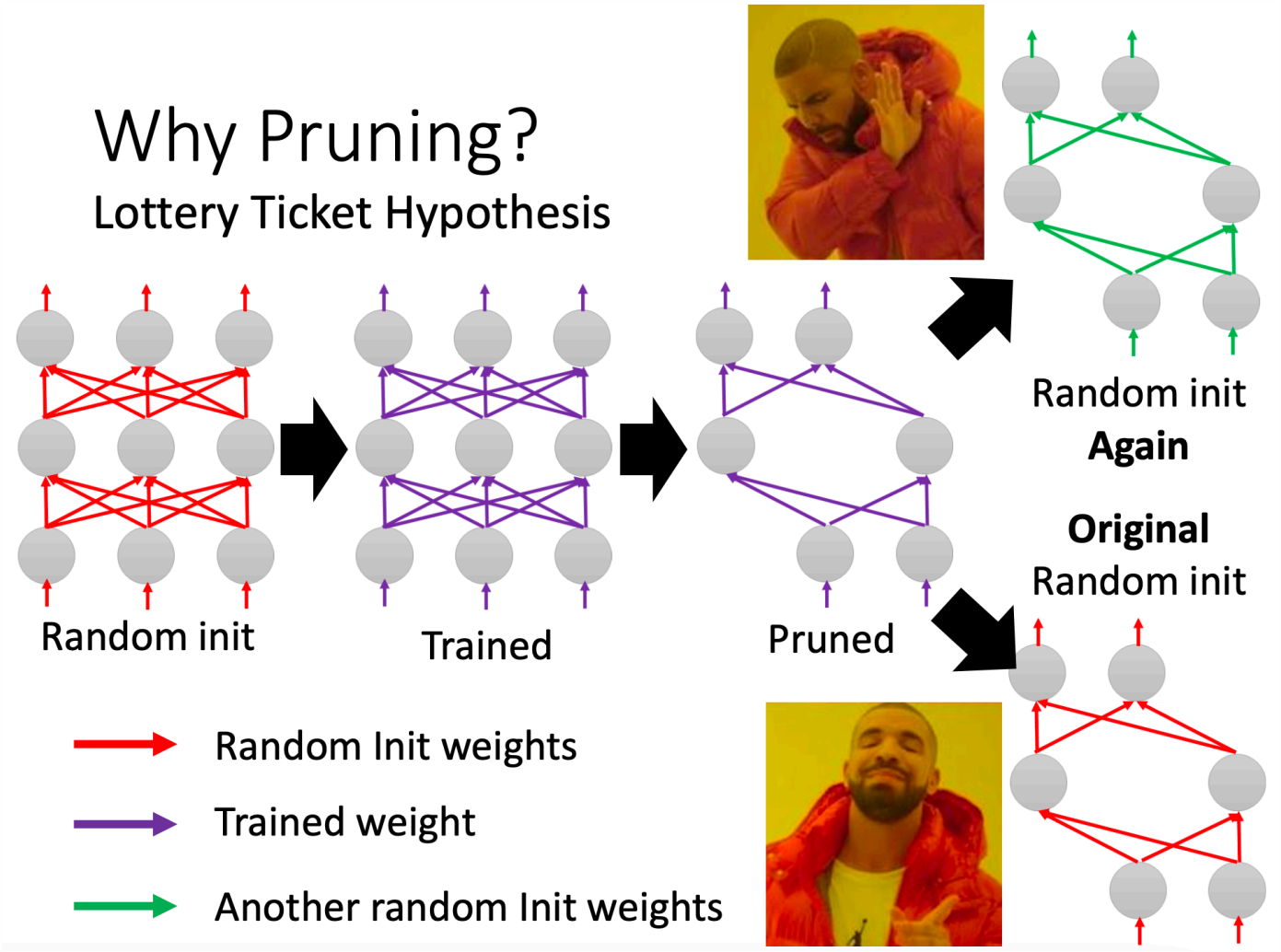
问题： 如果先训练一个小的网络，再慢慢把它变大，会不会有不错的效果呢?

答案： 不会。

Why？？

原因1 - 大的网络比较好训练。

（大乐透假说： 大的网络里面包含很多小的网络，在训练大的网络的时候，可以看作同时训练多个小的网络



（解析大乐透假说

结论： 对于一个好的初始化，只要不改变参数的正负号，小的网络就可以训练起来。即 正负号是初始化的关键

$$(0.9,\ 3.1,\ -1,\ 4.3)\ ----->(+\alpha,\ +\alpha,\ -\alpha,\ +\alpha) \tag{1}$$

结论2： 对于一个大网络来说，会不会有某个sub-网络，它连训练都不用，就已经是一个好的模型了。

在解构大乐透假说之前，Weight Agnostic Neural Networks 中提到：设置了一个神奇的网络，网络中所有的数值要么是随机的，要么通通都设置 1.5 这样。也有可能得到好的性能。

Rethinking the Value of Network Pruning (打脸大乐透假说) 里面说只有当学习率比较小的时候或者剪枝的时候以参数为单位才会发现 大乐透假说的现象。

Knowledge Distillation

Paper：

Do Deep Nets Really Need to be Deep?

Knowledge Distillation

大的网络为 Teacher Net

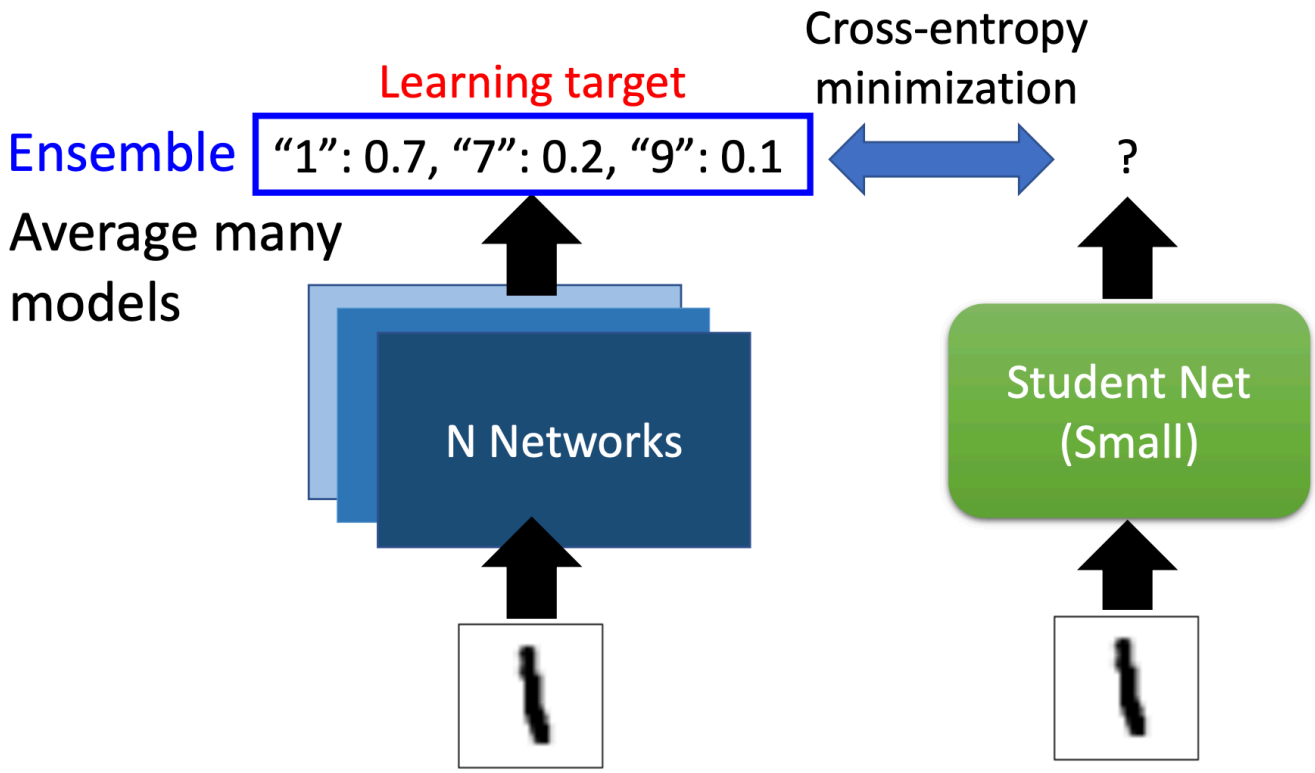
小的网络为 Student Net

E.g 对于手写数字识别任务而言。

假设输入的 Image - 1，大网络输出的为 $1:0.7, 7:0.2, 9:0.1$。则小网络会直接去学习大网络的输出。 这样做的好处是可以告诉小网络哪些数字有相似的联系。

甚至学生网络在训练的时候没有见过 数字7 的图片，也是有可能去识别出来 7 的。

Ensemble



这里可以用平均N个模型的输出，也可以去平均N个模型的参数。

Temperature for softmax

$$y_i' = \frac{\exp(y_i)}{\sum_j \exp(y_i)} \rightarrow y_i' = \frac{\exp(y_i/T)}{\sum_j \exp(y_i/T)} \tag{2}$$

T 是一个超参数。当$T > 1$时，将比较集中的分布变得平滑一些。

设 $T = 100$。

$$\begin{array}{llll} y_1 = 100 & y_1' = 1 & y_1/T = 1 & y_1' = 0.56 \\ y_2 = 10 & y_2' \approx 0 & y_2/T = 0.1 & y_2' = 0.23 \\ y_3 = 1 & y_3' \approx 0 & y_3/T = 0.01 & y_3' = 0.21 \end{array} \tag{3}$$

这样会使得学生网络在学习的时候更容易一些，告诉学生哪些类别是比较相像的。否则直接和正确的label去学习没有区别。

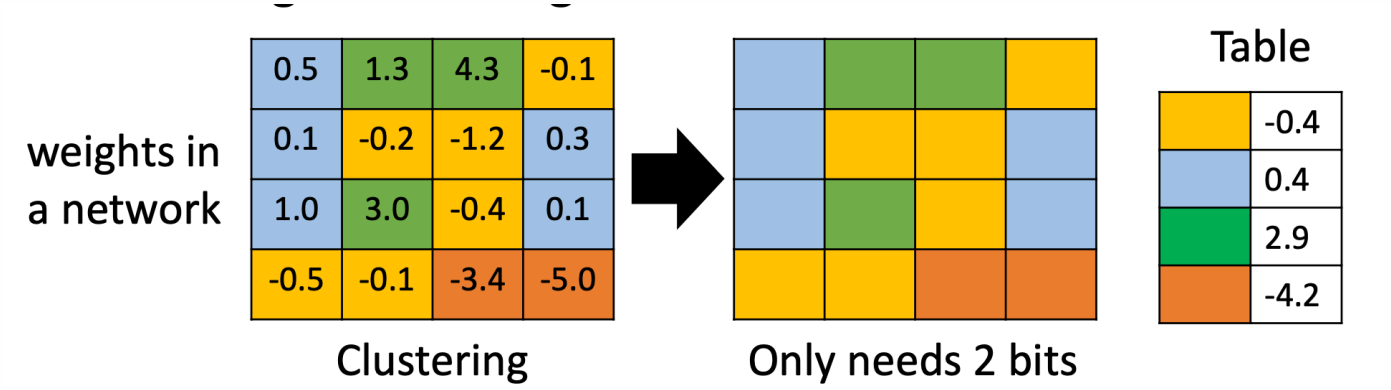
当然也可以用softmax前的输出去学习，甚至也可以用每一层的输出去输出。一般而言，加入更多的限制往往会得到更好的结果。

甚至甚至，可以在老师网络和学生网络之间在加一个中介网络。

Parameter Quantization

思路： 尽可能去使用少的空间(bits 比特)去存储参数。

Weight clustering 权重聚类



4个群(4个颜色)只需要2个bit就可以了，再加一个Table。

> 注：1bit只能存0或者1，对于颜色表来说，每一个格子的表示最多需要两位(00, 01, 10, 11)

Huffman Encoding

可以更减少存储。

Binary Weights

> 仅仅使用 1bit 来存储参数，即参数不是1就是-1.

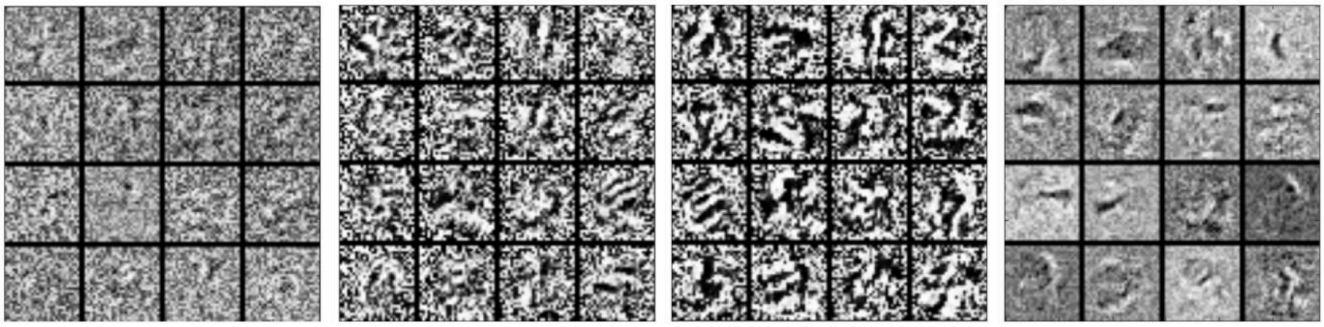
Binary Connect

Binary Network

XNOR-net

https://arxiv.org/abs/1511.00363

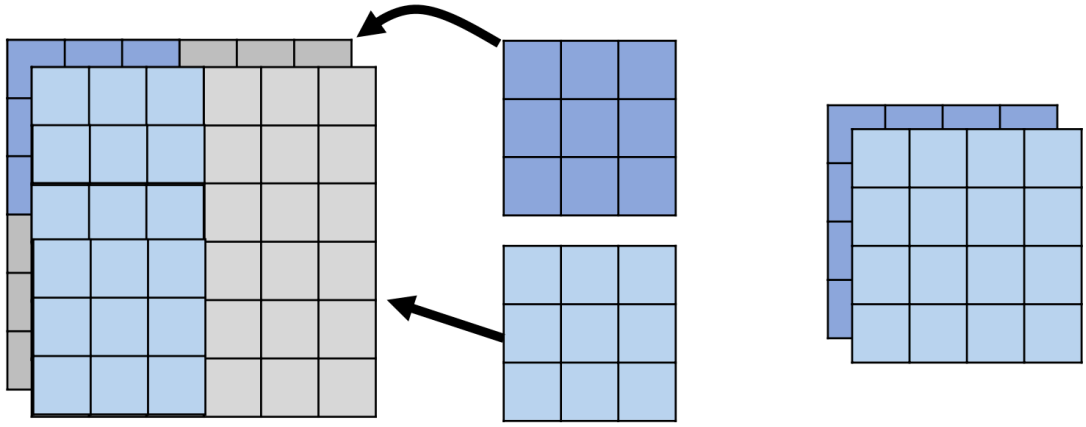
Method	MNIST	CIFAR-10	SVHN
No regularizer	$1.30 \pm 0.04\%$	10.64%	2.44%
BinaryConnect (det.)	$1.29 \pm 0.08\%$	9.90%	2.30%
BinaryConnect (stoch.)	$1.18 \pm 0.04\%$	$\mathbf{8.27\%}$	2.15%
50% Dropout	$1.01 \pm 0.04\%$		



Architecture Design

Depthwise Separable Convolution

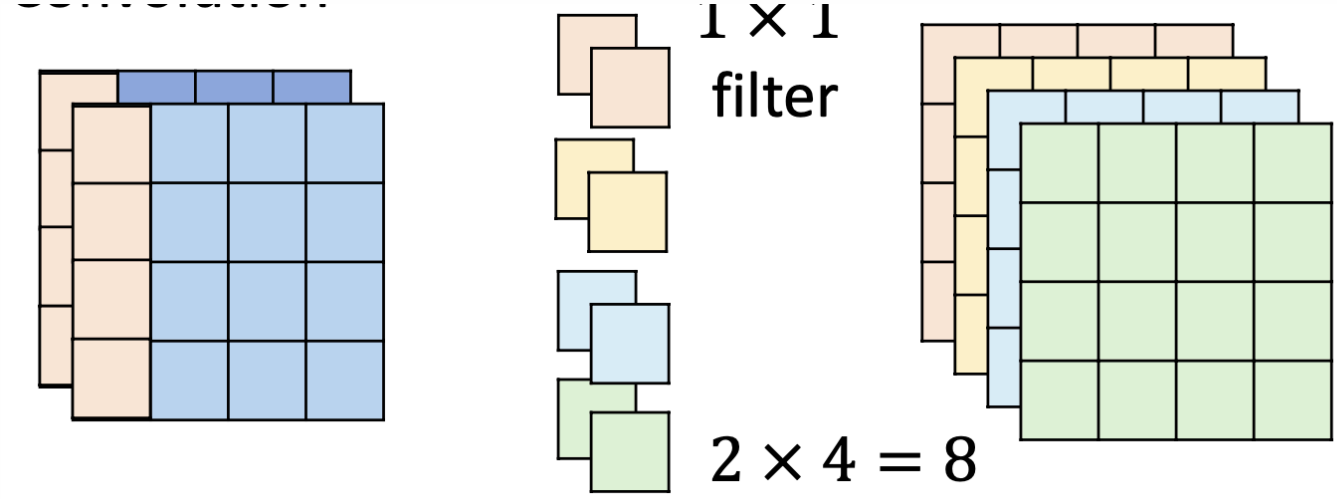
1 - Depthwise Convolution



- Filter number = Input channel number
- Each filter only considers one channel.
- The filters are $k \times k$ matrices
- There is no interaction between channels.

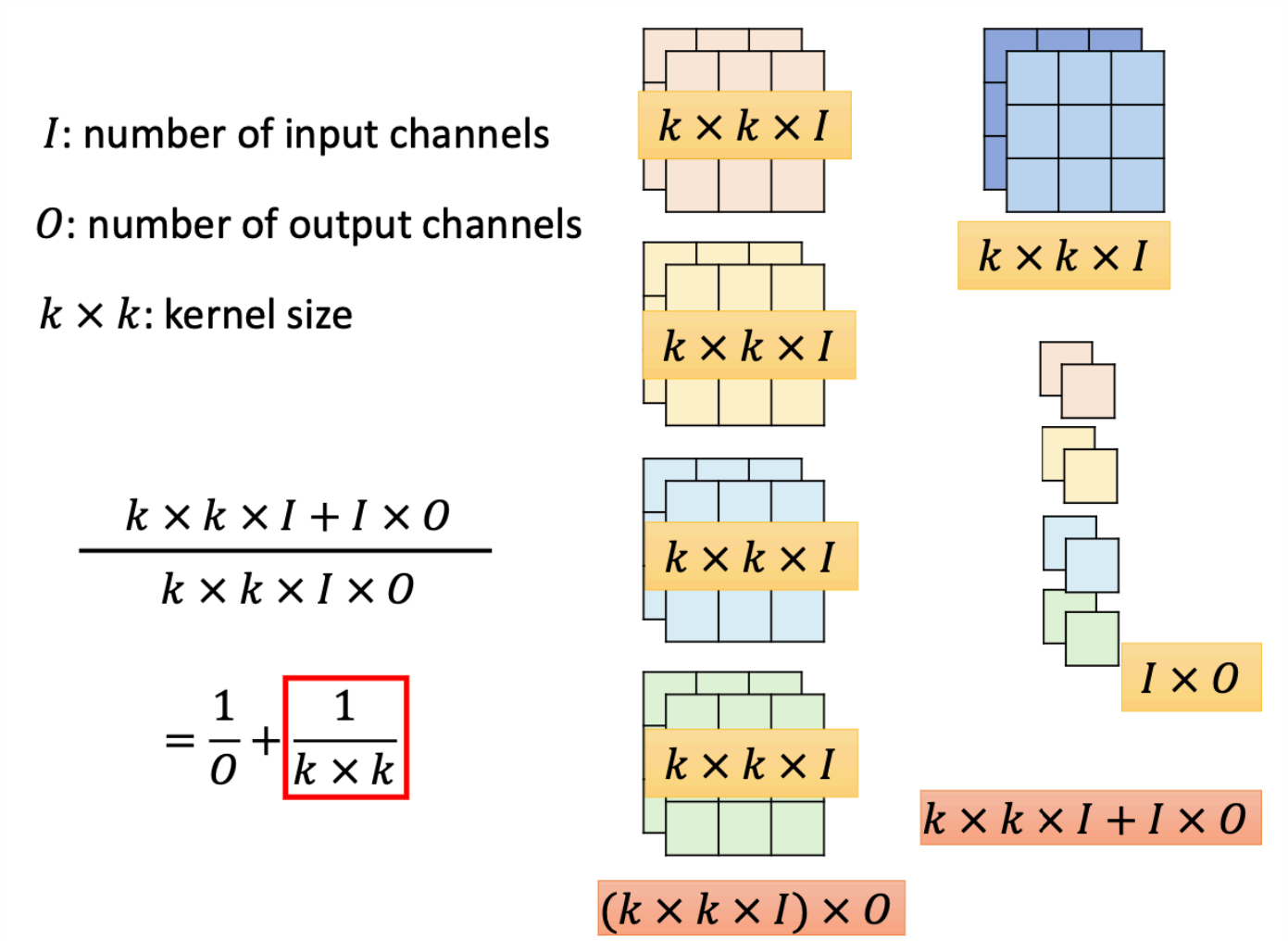
这样做的卷积没有考虑到通道之间的信息，即损失掉了空间信息。

2 - Pointwise Convolution



比较

前提： 两种卷积方式最终得到的特征图的尺寸是一样的。

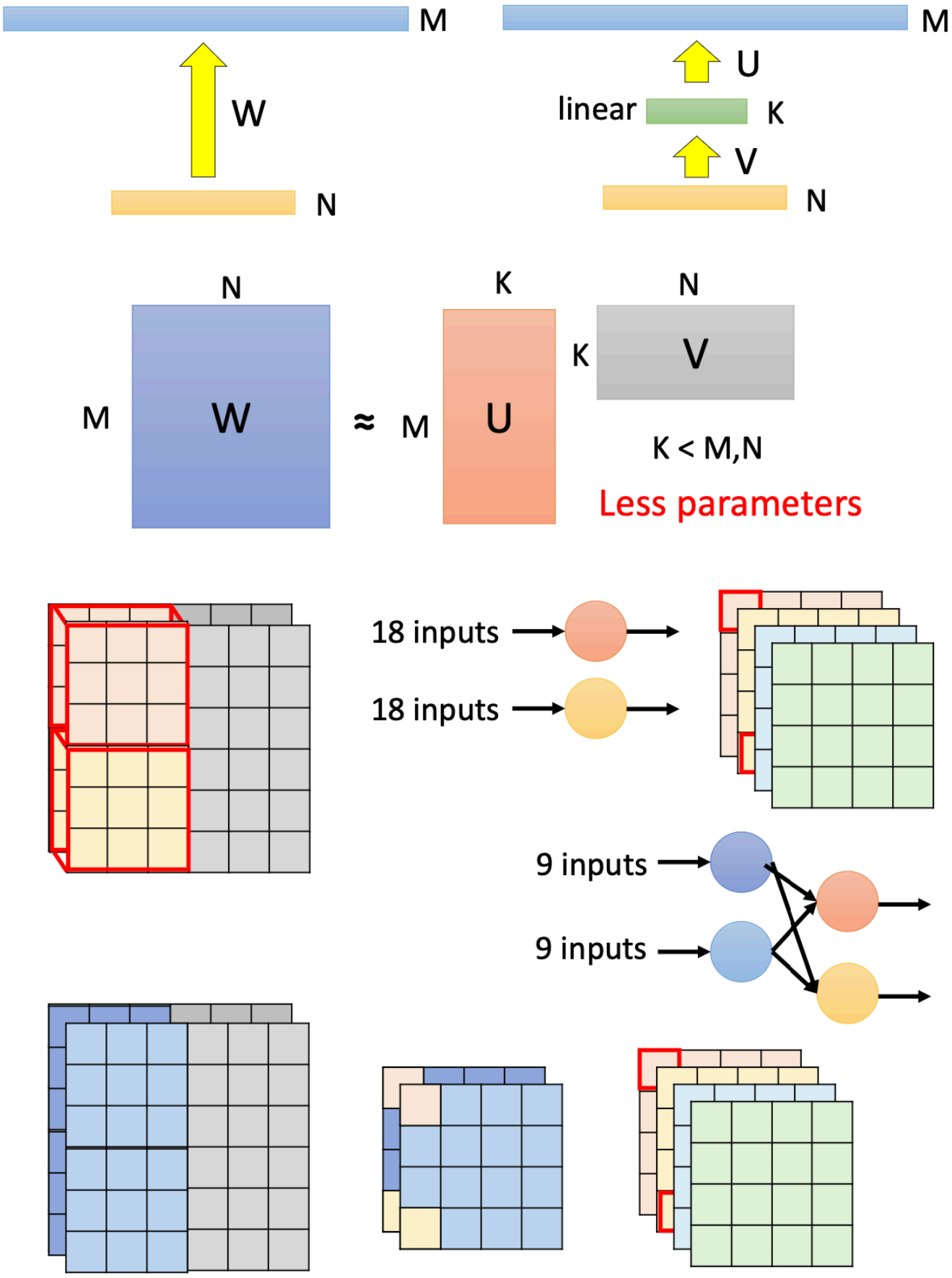


$$\frac{k \times k \times I + I \times O}{k \times k \times I \times O}$$

$$= \frac{1}{O} + \boxed{\frac{1}{k \times k}}$$

Low rank approximation

> 解释 为什么深度可分离卷积是有效的



把一层拆成两层，这样会减少参数。

Paper：

SqueezeNet

MobileNet

ShuffleNet

Xception

GhostNet

Dynamic Computation

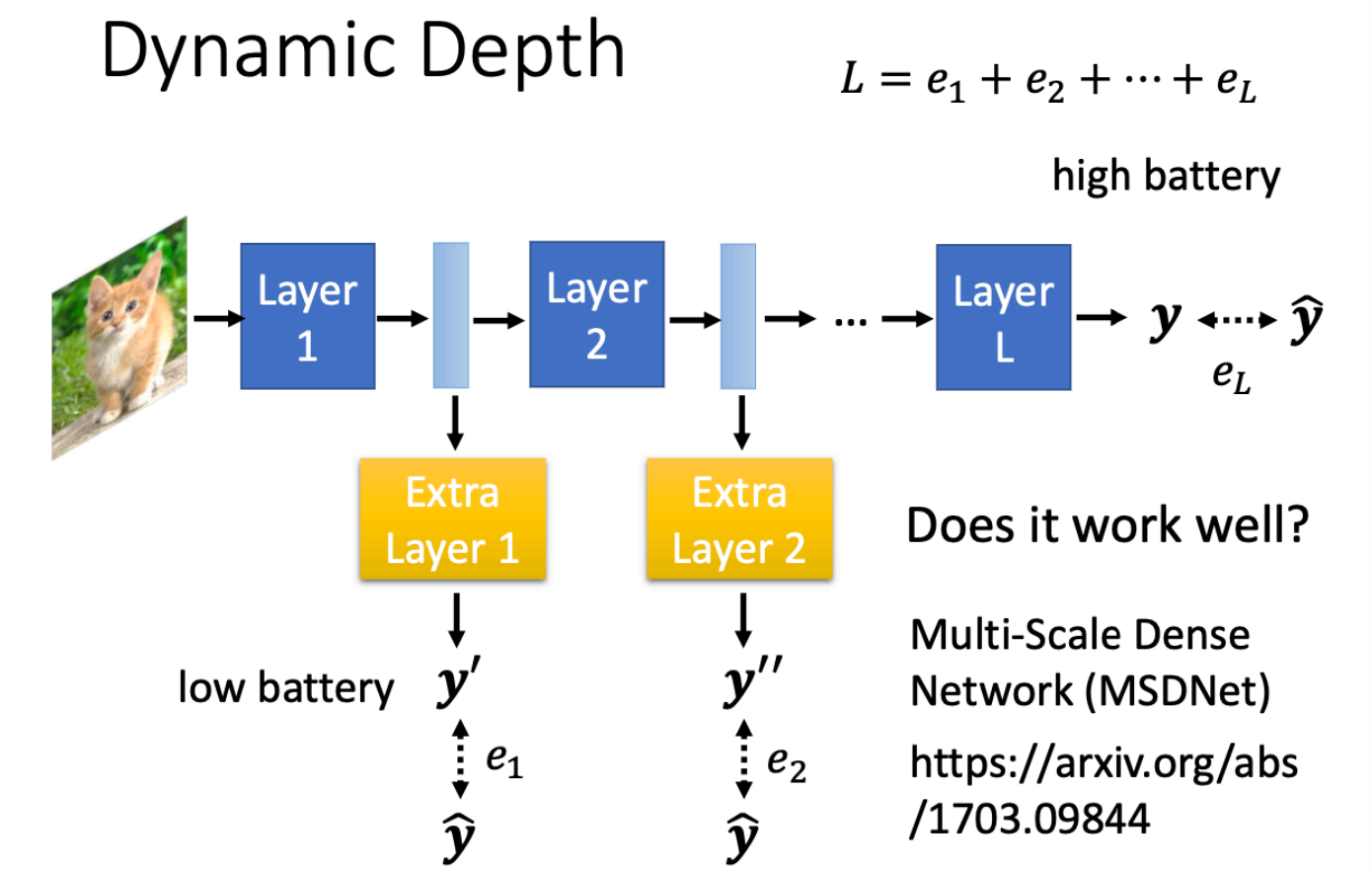
> 不同于之前的四个技术。它们是把网络变小。而动态计算是希望网络可以自由调整它的运算量。

Q： 为什么需要自由调整它的运算量?

A： 1 - 不同设备的计算资源是不一样的，除非每次针对不同的设备去修改网络。

2 - 对于同一个设备而言，设备的电量可能会导致网络每次得到的计算资源不一样多。

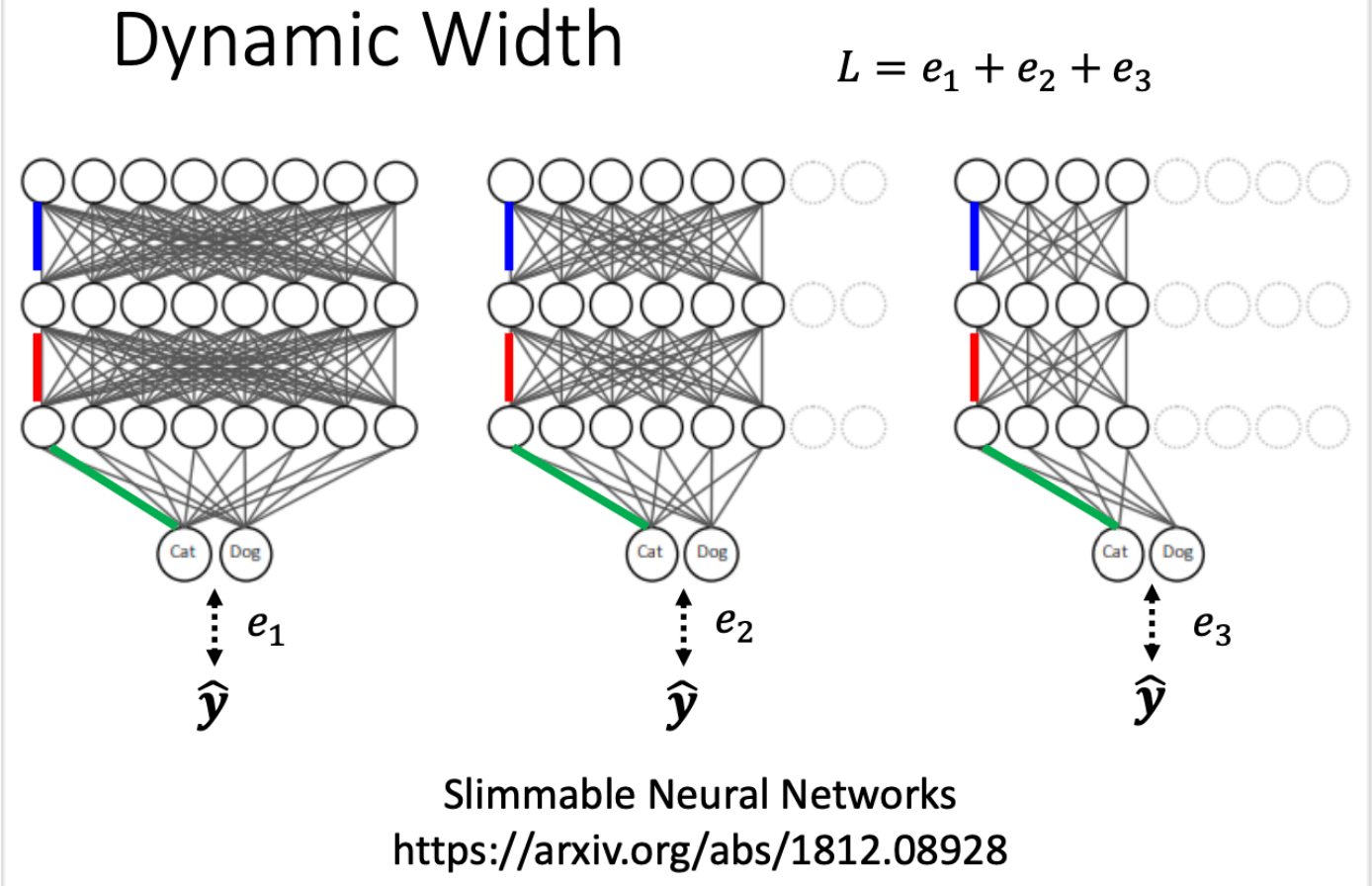
Dynamic Depth



当资源充足的时候，可以跑深层的网络，当资源不充足的时候，就只能跑浅层的网络。

另一种方法 Multi-Scale Dense Network (MSDNet)

Dynamic Width



根据环境或者难度进行宽带或深度的选取

对于简单的输入，可能通过1或者2层就可以得到答案了，但是对于负责/困难的输入，可能需要很多层才能得到正确的答案。

Paper

SkipNet: Learning Dynamic Routing in Convolutional Networks

RuntimeNeuralPruning

BlockDrop: Dynamic Inference Paths in Residual Networks