李宏毅-Network Compression课程笔记

marsggbo

香港浸会大学 计算机博士在读

已关注

一、方法总结

- Network Pruning
- Knowledge Distillation
- Parameter Quantization
- Architecture Design
- Dynamic Computation

二、Network Pruning

模型通常是过参数的，即很多参数或者neuron是冗余的(例如非常接近0),因此我们可以移除这些参数来对模型进行压缩。

1. 重要性判断

那么怎么判断哪些参数是冗余或者不重要的呢?

- 对权重(weight)而言，我们可以通过计算它的 `l1`，`l2` 值来判断重要程度
- 对neuron而言，我们可以给出一定的数据集，然后查看在计算这些数据集的过程中neuron参数为0的次数，如果次数过多，则说明该neuron对数据的预测结果并没有起到什么作用，因此可以去除。

2. 为什么要pruning?

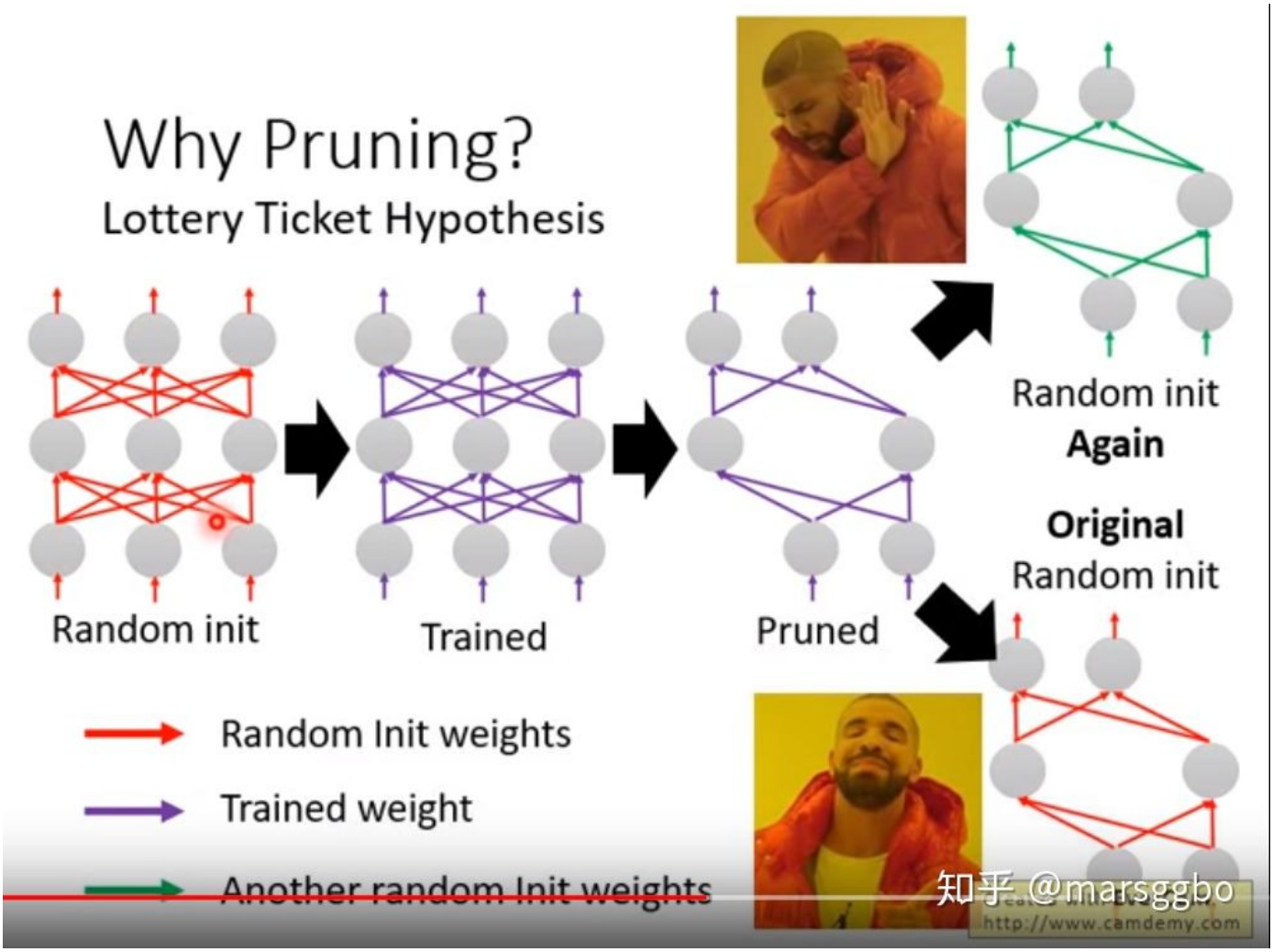
那我们不禁要问，既然最后要得到一个小的network，那**为什么不直接在数据集上训练小的模型，而是先训练大模型?**

- 解释一

一个比较普遍接受的解释是因为模型越大，越容易在数据集上找到一个局部最优解，而小模型比较难训练，有时甚至无法收敛。

- 解释二

首先看最左边的网络，它表示大模型，我们随机初始化它权重参数（红色）。然后我们训练这个大模型得到训练后的模型以及权重参数（紫色）。最后我们对训练好的大模型做pruning得到小模型。



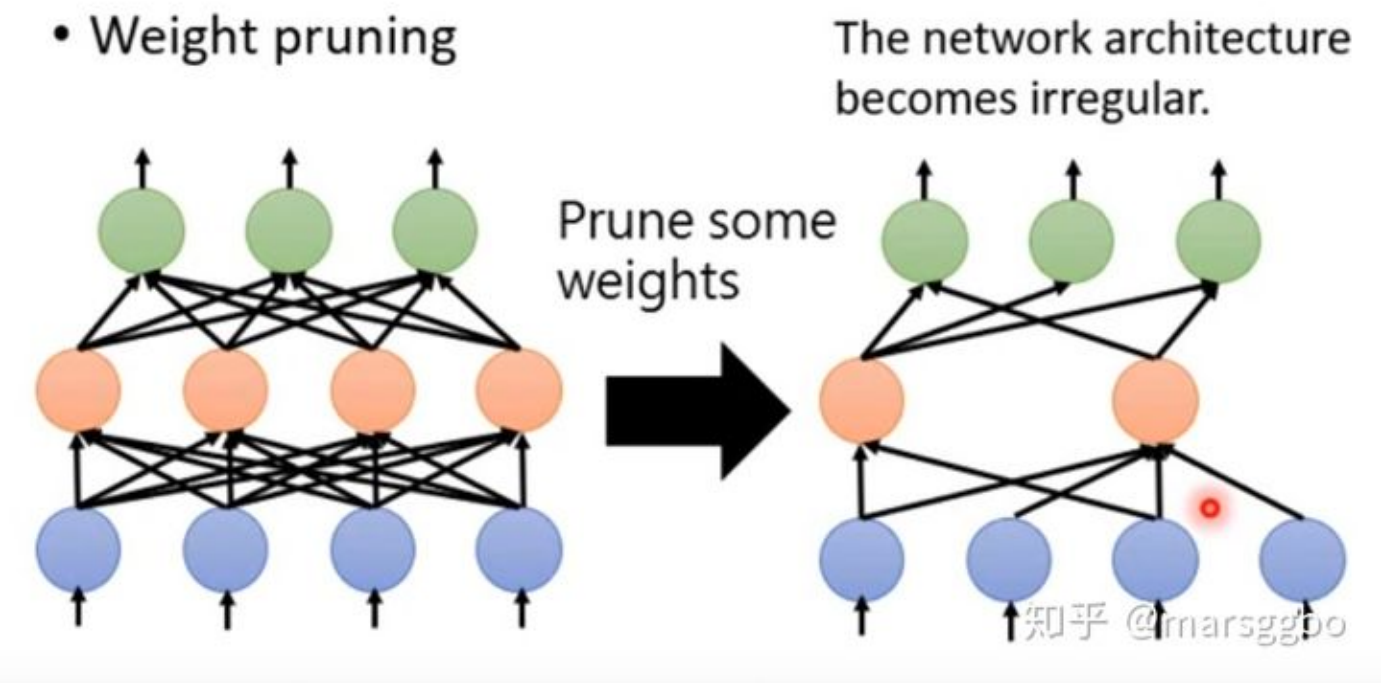
下面就是见鬼的时刻，作者把小模型拿出来后随机初始化参数（绿色，右上角），结果发现根本无法训练。然后他又把最开始的大模型的随机初始化的权重复制到小模型上（即把对应位置的权重参数复制过来，右下角），结果发现正常训练！！！

作者对此的解释是，就像我们买彩票，买的彩票越多，中奖的几率才会越大。而最开始的大模型可以看成是由超级多的小模型组成的，也就是对大模型随机初始化参数会得到各种各样的初始化参数的小模型，有的小模型可能根本就无法训练，但是有的就可以，所以上图中右下角的小模型套用大模型对应位置的初始化参数，其实就是使用了“中奖”了的那个部分的参数。

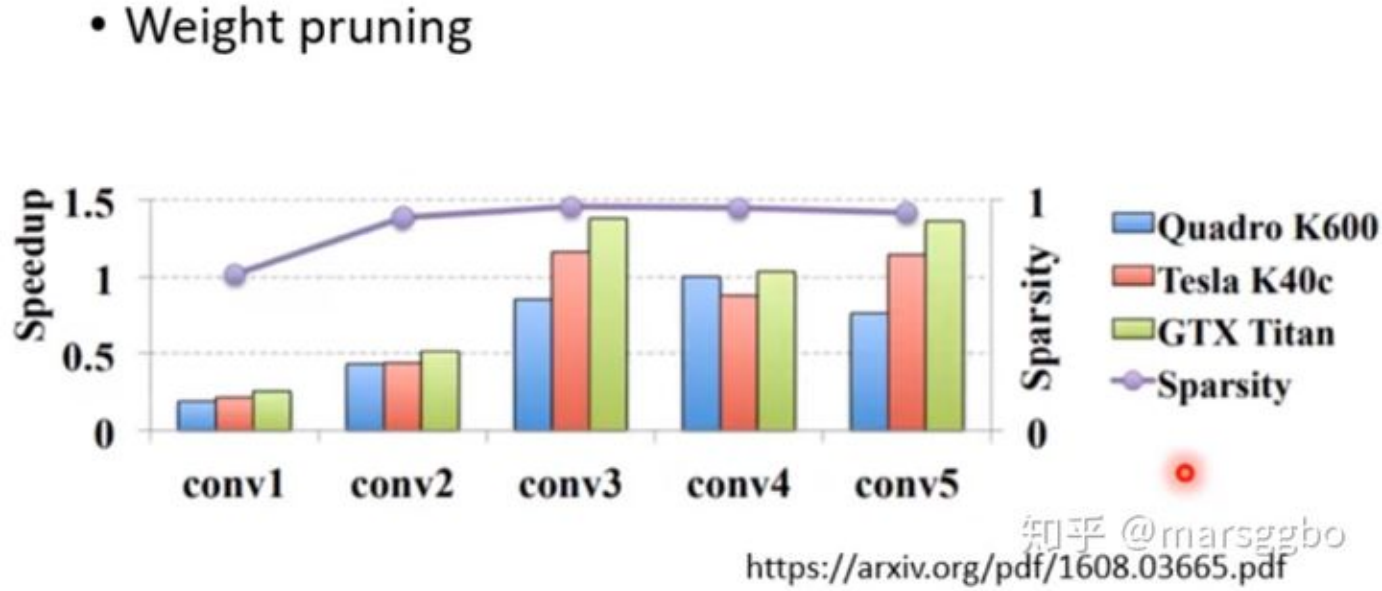
BUT！！！后面有一篇文章Rethinking the value of network pruning做了实验发现随机初始化是可以正常训练的，即他的结论和前面大**大乐透假设**有违背。。。emm，大佬之间大家我们还是做个吃瓜群众吧。

3. 实际操作分析

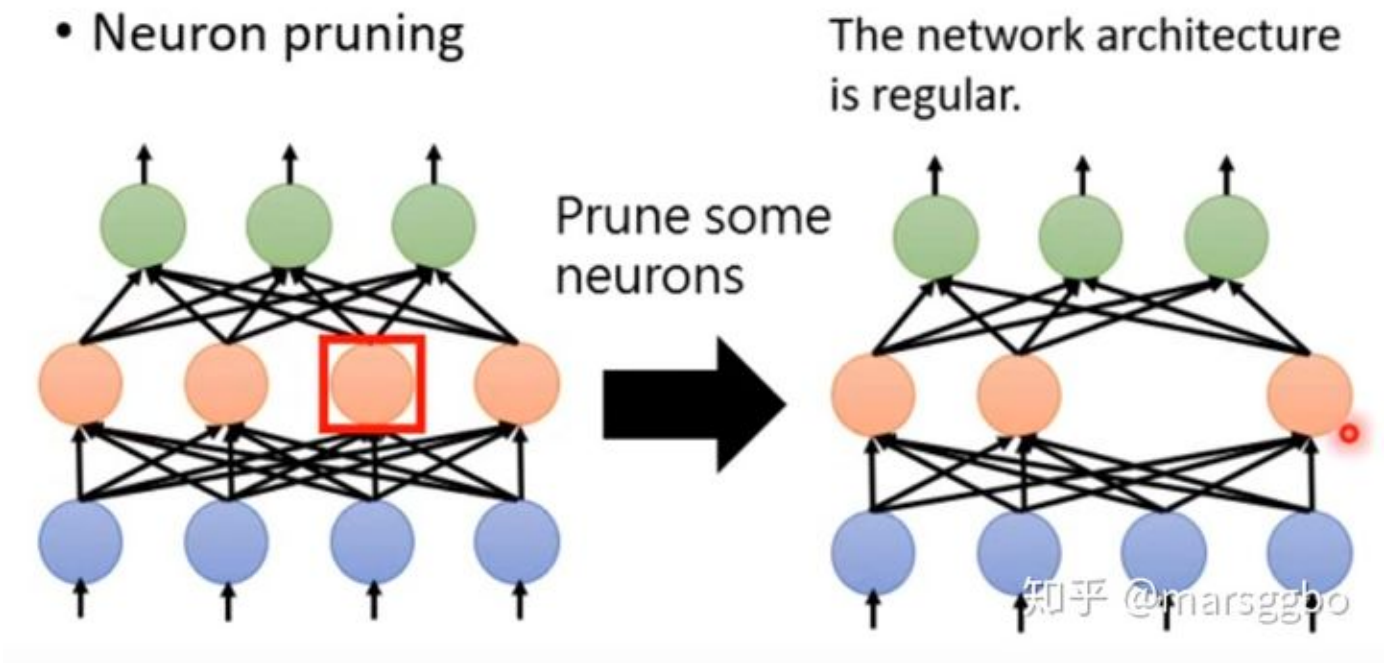
• weight pruning



如上图示，每个节点的输出和输出节点数都会变得不规则，这样一来有两个方面的问题： - 使用Pytorch，Keras实现起来不方便 - GPU是对矩阵运算做加速，现在都变得不规则，看起来似乎GPU面对这种情况也无能为力。2016年的一篇文章对AlexNet就做了这样的实验，实验结果是模型参数去掉了将近90%，最后的准确率只降低了2%左右，说明weight pruning的确能够在保证模型准确率的同时减少了模型大小，but！！！最后实验发现模型计算的速度似乎并没有提速，甚至对有的结构的修改使得速度降低了。



• neuron pruning

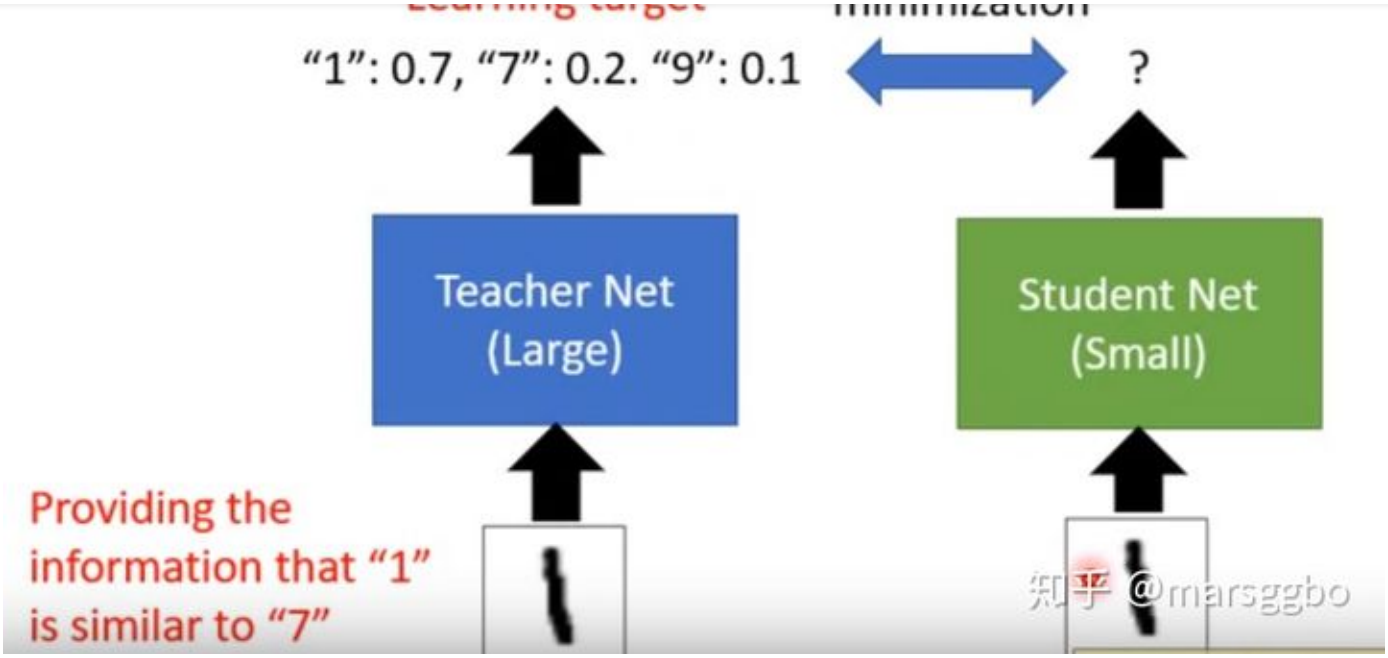


三、Knowledge Distillation

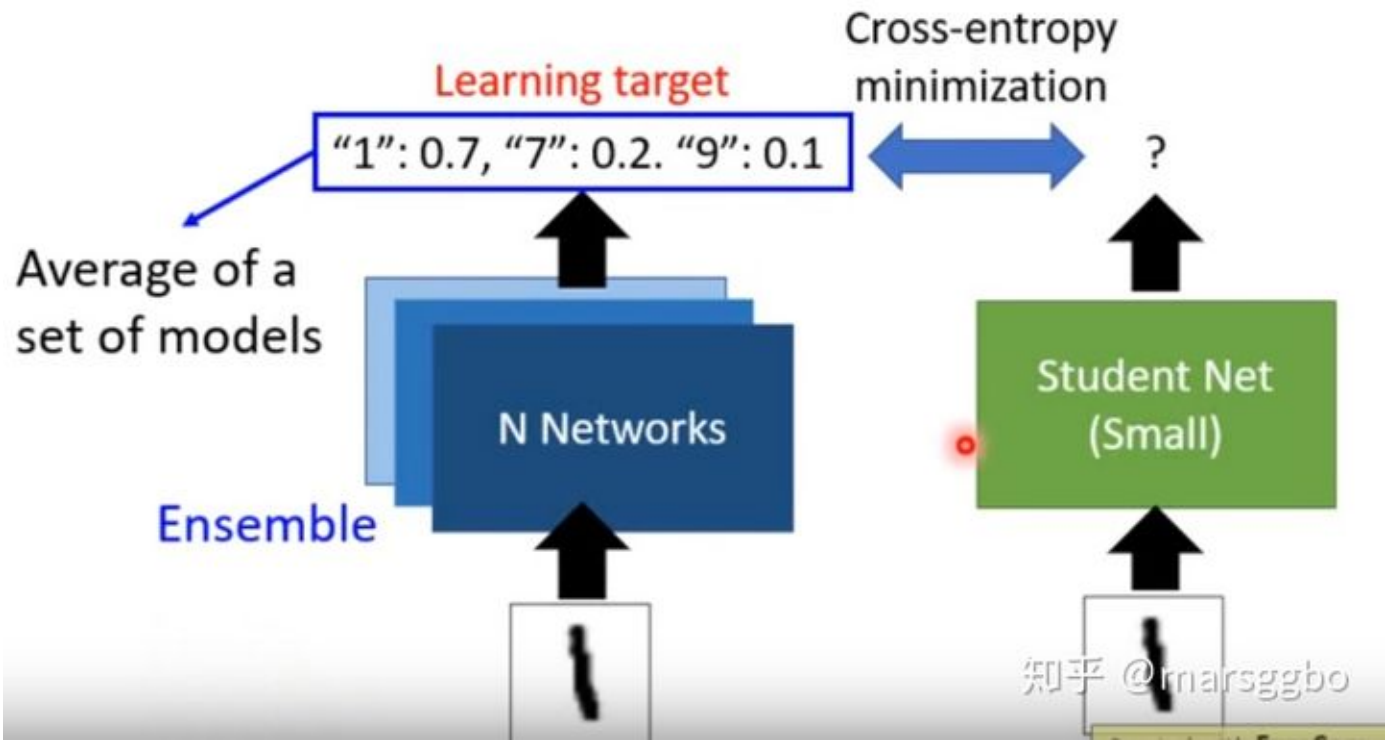
直接看下面的图应该很好理解。整个知识蒸馏过程中会用到两个模型：大模型（Teacher Net）和小模型（Student Net）。

具体方法是我们先用大模型在数据集上学习到收敛，并且这个大模型要学的还不错，因为后面我们要用大模型当老师来教小模型学习嘛，如果大模型本身都没学好还教个锤子，对吧？

我们以MNIST数据集为例，假设大模型训练好了，现在对于一张数字为“1”的图像，大模型的输出结果是由0.7的概率是1,0.2的概率是7,0.1的概率是9，这是不是有一定的道理？相比如传统的one-hot格式的label信息，这样的label包含更多的信息，所以Student Net要做的事情就是对于这张数字为“1”的图像，它的输出结果也要尽量接近Teacher Net的预测结果。



当然，一个更骚气的办法就是让多个老师出谋划策来教学生，即用Ensemble Net来进一步提升预测准确率，让学生学习的知识更加准确。



那Student Net到底如何学习呢？首先回顾一下在多类别分类任务中，我们用到的是softmax来计算最终的概率，即

$$y_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

规的label无异了，所以为了解决这个问题就引入了一个新的参数T,称之为**Temperature**,即有:

$$y_i = \frac{\exp(x_i/T)}{\sum_j \exp(x_j/T)}$$

此时，如果我们令T=100,那么最后的预测概率是y1=0.56,y2=0.23,y3=0.21。（不过李宏毅老师在视频里提到说这个方法在实际使用时貌似用处不大hhhh，感觉这个方法可以回答知乎上的 **什么东西看起来很厉害但是没什么用?** 哈哈哈哈哈哈哈哈哈哈或或）

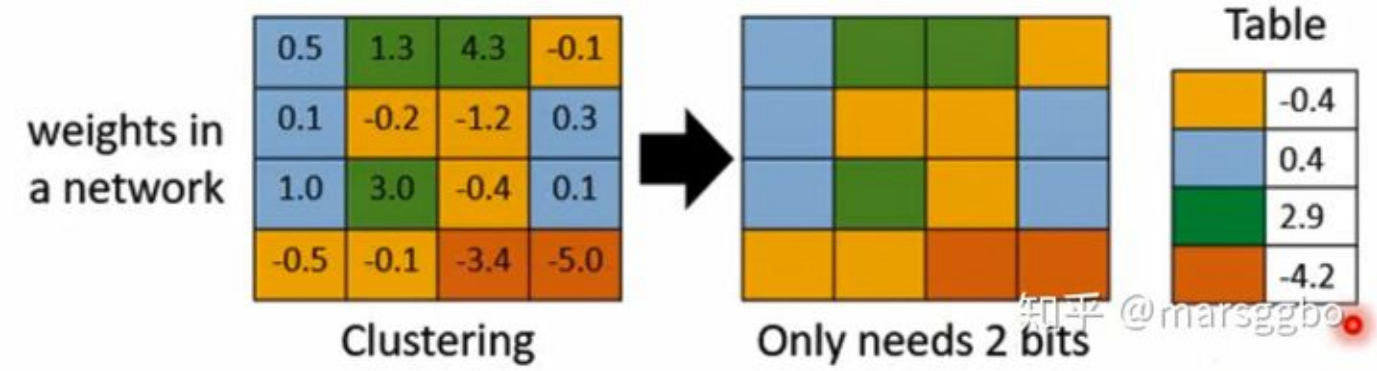
四、Parameter Quantization

1. less bits

一个很直观的方法就是使用更少bit来存储数值，例如一般默认是32位，那我们可以用16或者8位来存数据。

2. weight clustering

如下图所示，最左边表示网络中正常权重矩阵，之后我们对这个权重参数做聚类，比如最后得到了4个聚类，那么为了表示这4个聚类我们只需要2个bit，即用00,01,10,11来表示不同聚类。之后每个聚类的值就用均值来表示。这样的一个缺点就是误差可能会比较大。



3. Huffman Encoding

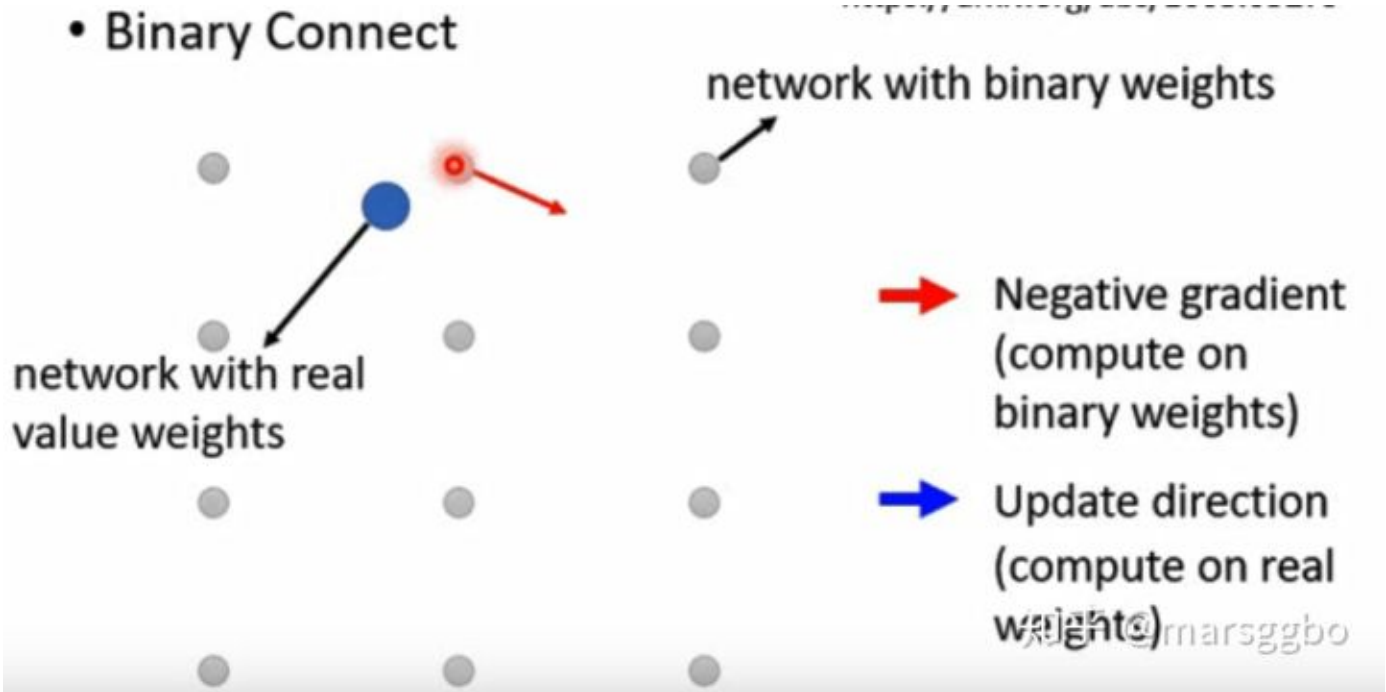
思路很简单，以上面的图为例，就是对于常出现的聚类用少一点的bit来表示，而那些很少出现的聚类就用多一点的bit来表示。

4. Binary Weights

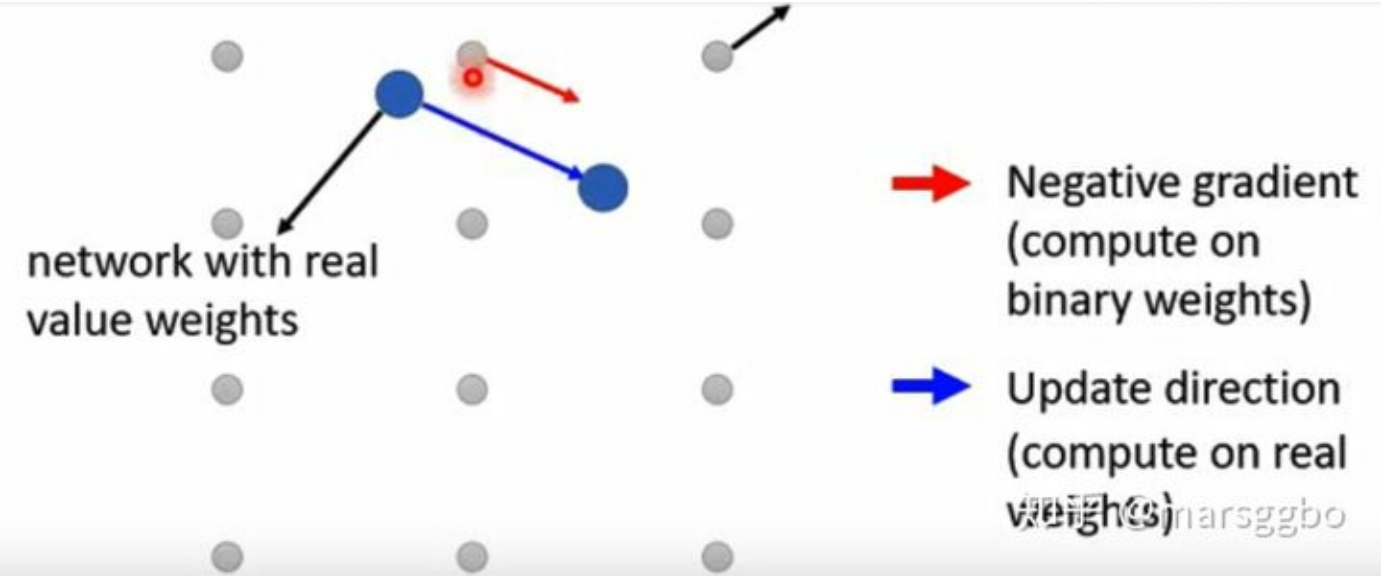
- BInary Connect
- BInary Network
- XNOR-Net

下面简单介绍一下**Binary Connect**的思路，如下图示，灰色节点表示使用binary weight的神经元，蓝色节点可以是随机初始化的参数，也可以是真实的权重参数。

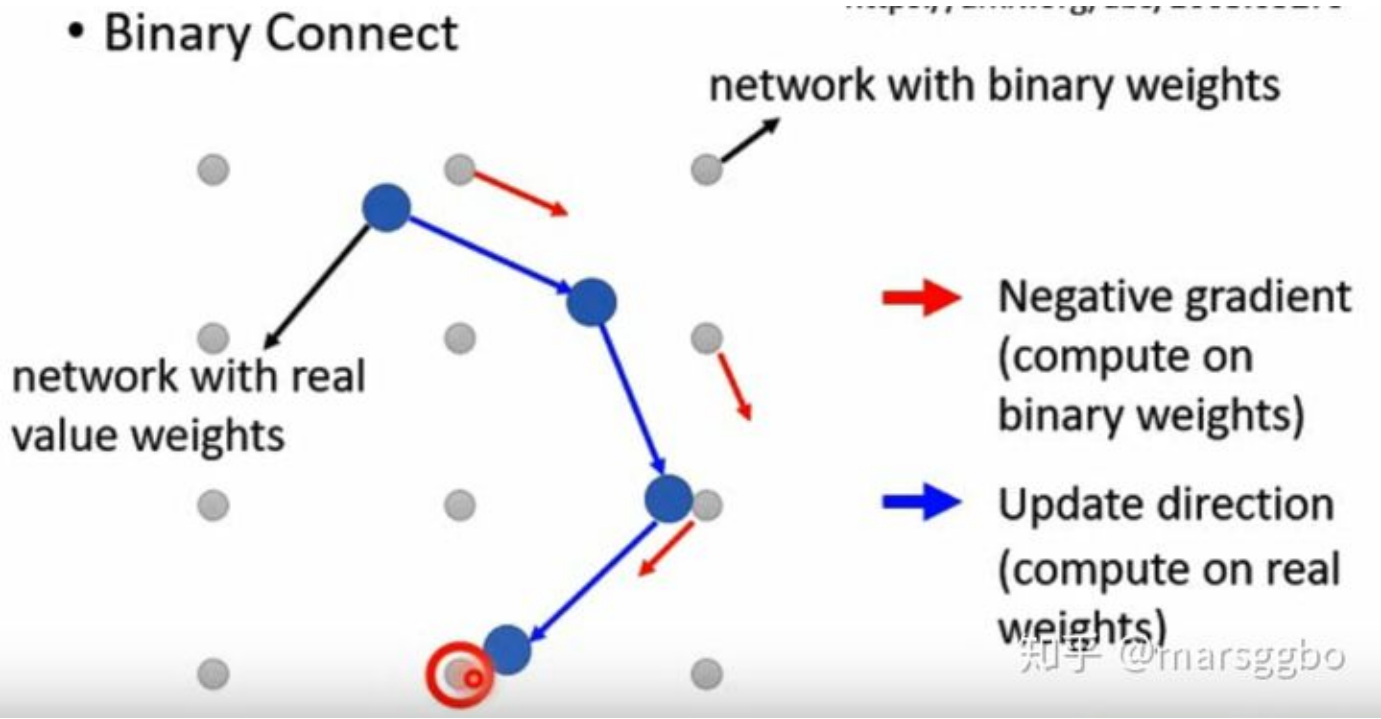
第一步我们先计算出和蓝色节点最接近的二元节点，并计算出其梯度方向（红色剪头）。



第二步，蓝色节点的更新方向则是按照红色箭头方向更新，而不是按照他自身的梯度方向更新。如下图示，梯度下降后，蓝色节点到了一个新的位置。



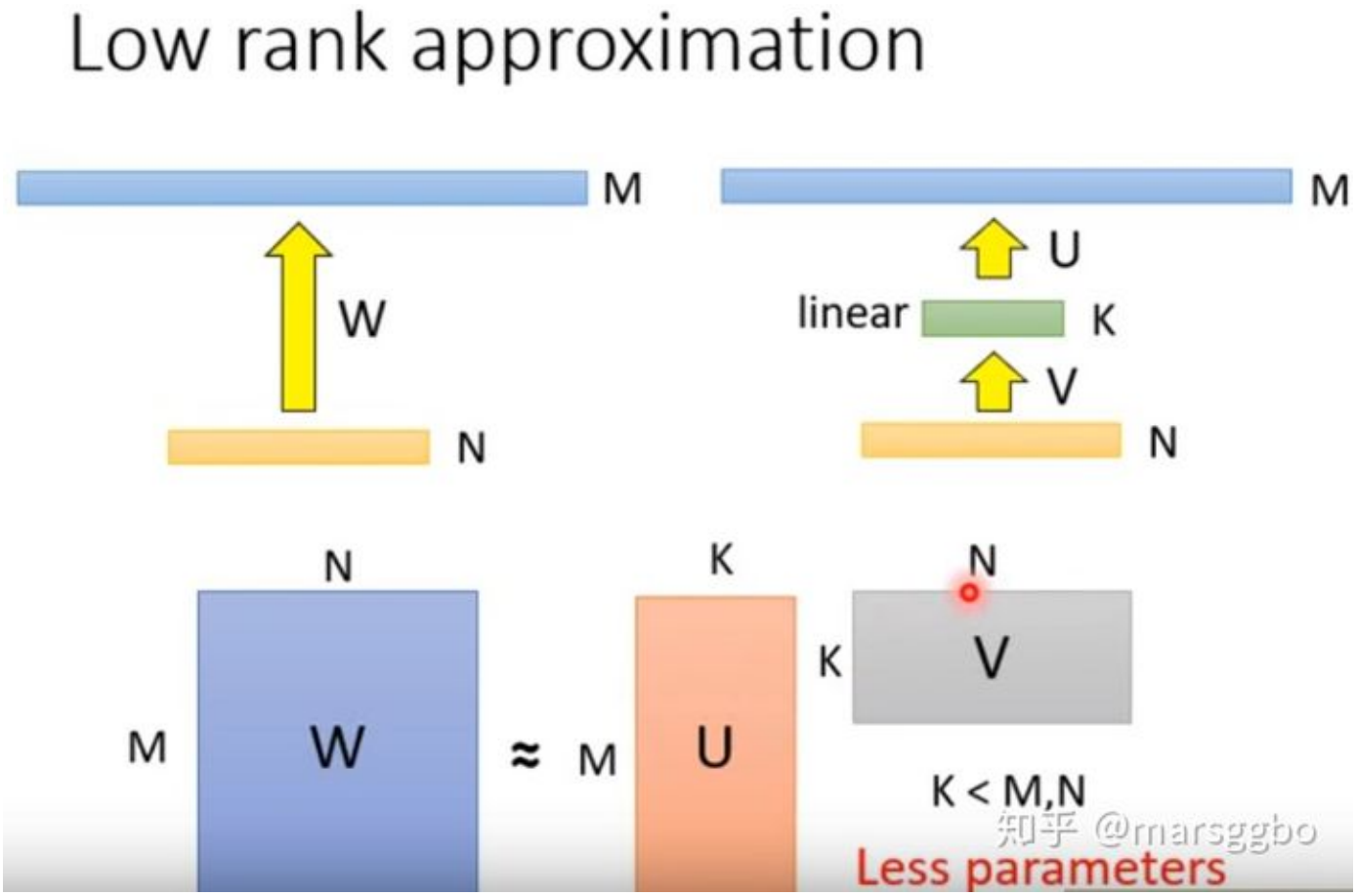
最后在满足一定条件后(例如训练之最大epoch),蓝色节点会停在一个灰色节点附近,那么我们就是用该灰色节点的权重。



五、Architecture Design

1. Low Rank Approximation(低秩近似)

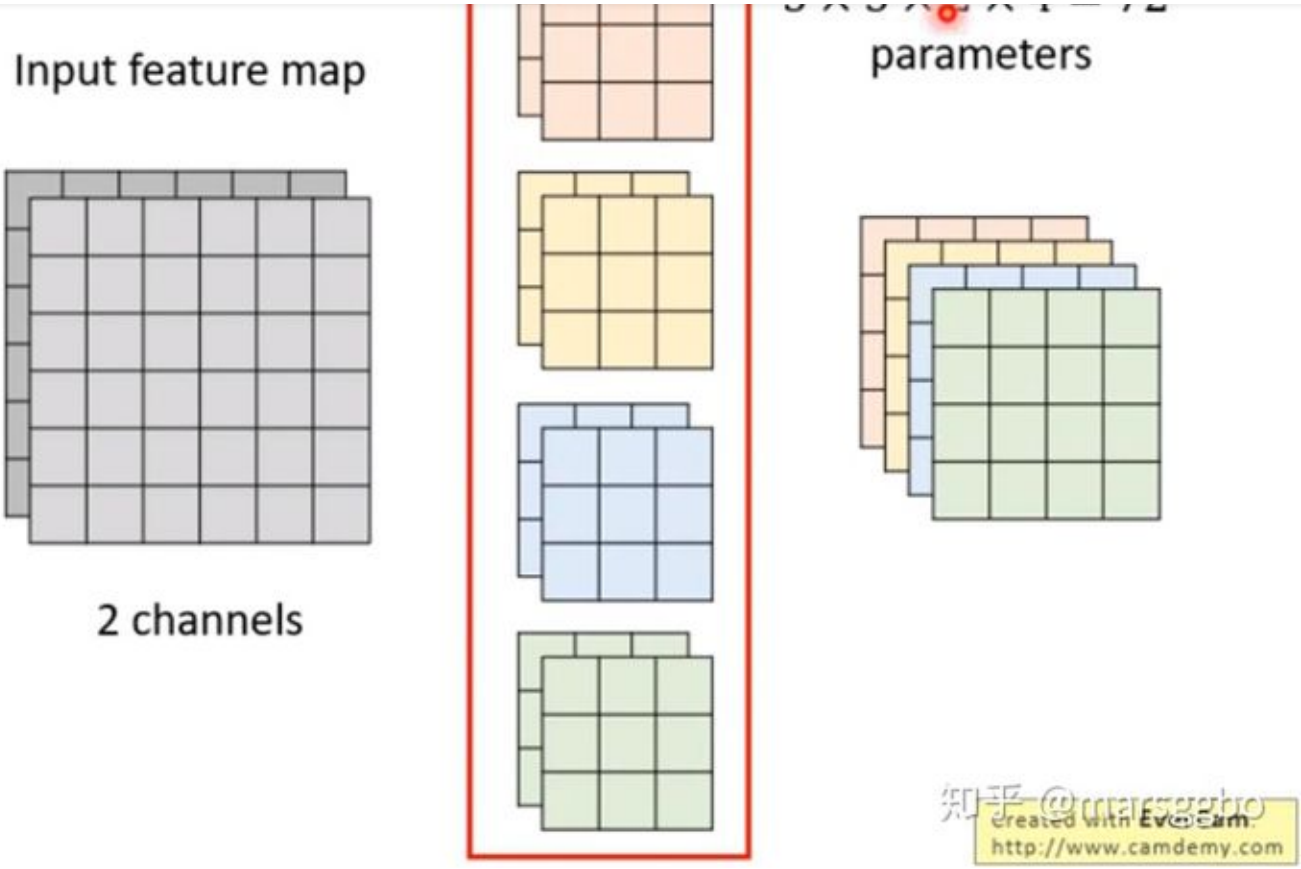
下图是低秩近似的简单示意图，左边是一个普通的全连接层，可以看到权重矩阵大小为 $M \times N$ ，而低秩近似的原理就是在两个全连接层之间再插入一层K。是不是很反直观？插入一层后，参数还能变少？



但是低秩近似之所以叫**低秩**，是因为原来的矩阵的秩最大可能是min(M,N),而新增一层后可以看到矩阵U和V的秩都是小于等于K的，我们知道rank(AB)≤min(rank(A),rank(B)), 所以相乘之后的矩阵的秩一定还是小于等于K。那么这样会带来什么影响呢？那就是原先全连接层能表示更大的空间，而现在只能表示小一些的空间了。

2. Depthwise Separable Convolution

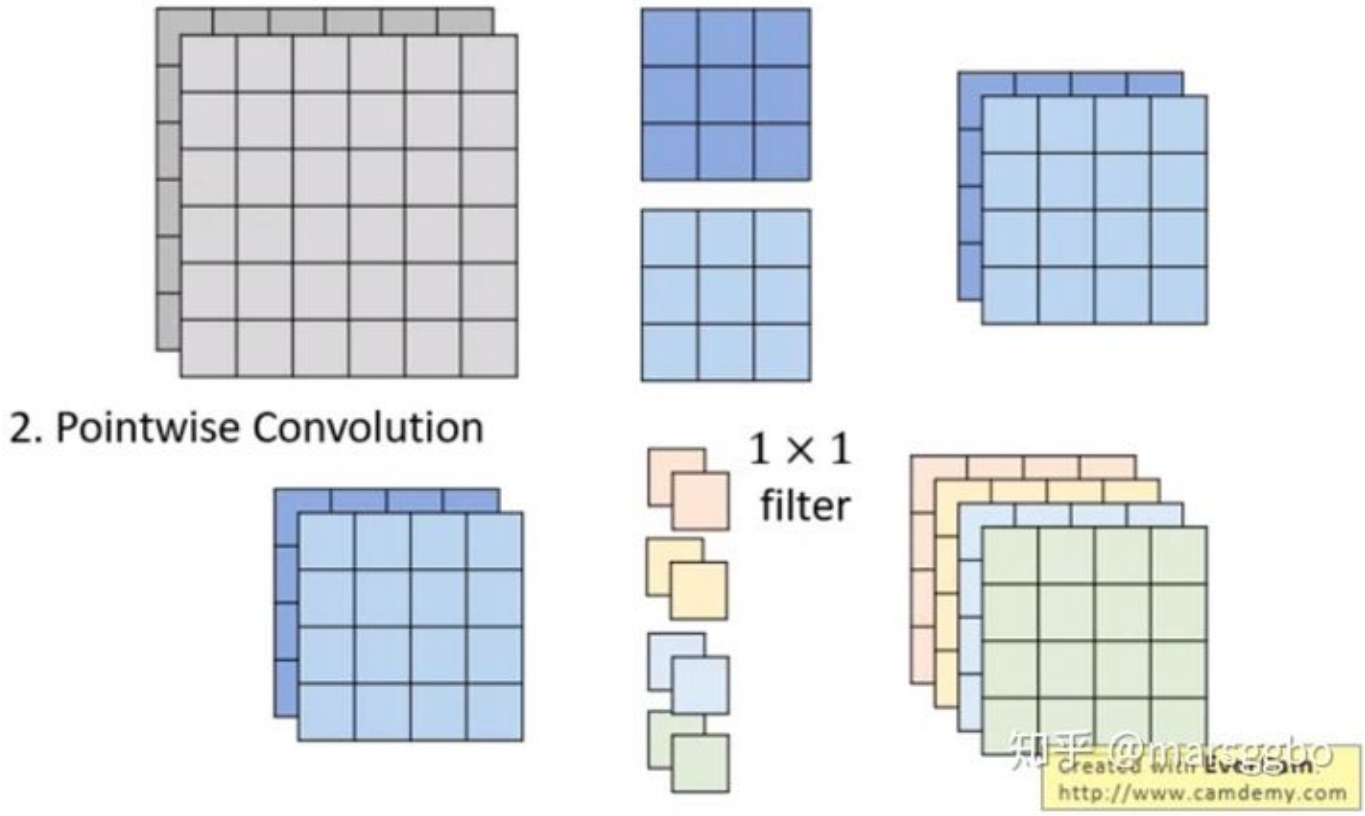
首先看一下标准卷积所需要的参数量。如下图示，输入数据由两个6*6的feature map组成，之后用4个大小为3*3的卷积核做卷积，最后的输出特征图大小为4*4*4。每个卷积核参数数量为2*3*3=18,所以总共用到的参数数量为4*18=72。



而Depthwise Separable卷积分成了两步，如下图示。

首先是输入数据的每个通道只由一个二维的卷积核负责，即卷积核通道数固定为1，而不是像上面那样，每个卷积核的通道数和输入通道数保持一致。这样最后得到的输出特征图的通道数等于输入通道数。

因为第一步得到的输出特征图是用不同卷积核计算得到的，所以不同通道之间是独立的，因此我们还需要对不同通道之间进行关联。为了实现关联，在第二步中使用了1*1大小的卷积核，通道数量等于输入数据的通道数量。另外1*1卷积核的数量等于预期输出特征图的通道数，在这里等于4。最后我们可以得到和标准卷积一样的效果，而且参数数量更少：3*3*2+(1*1*2)*4=26。

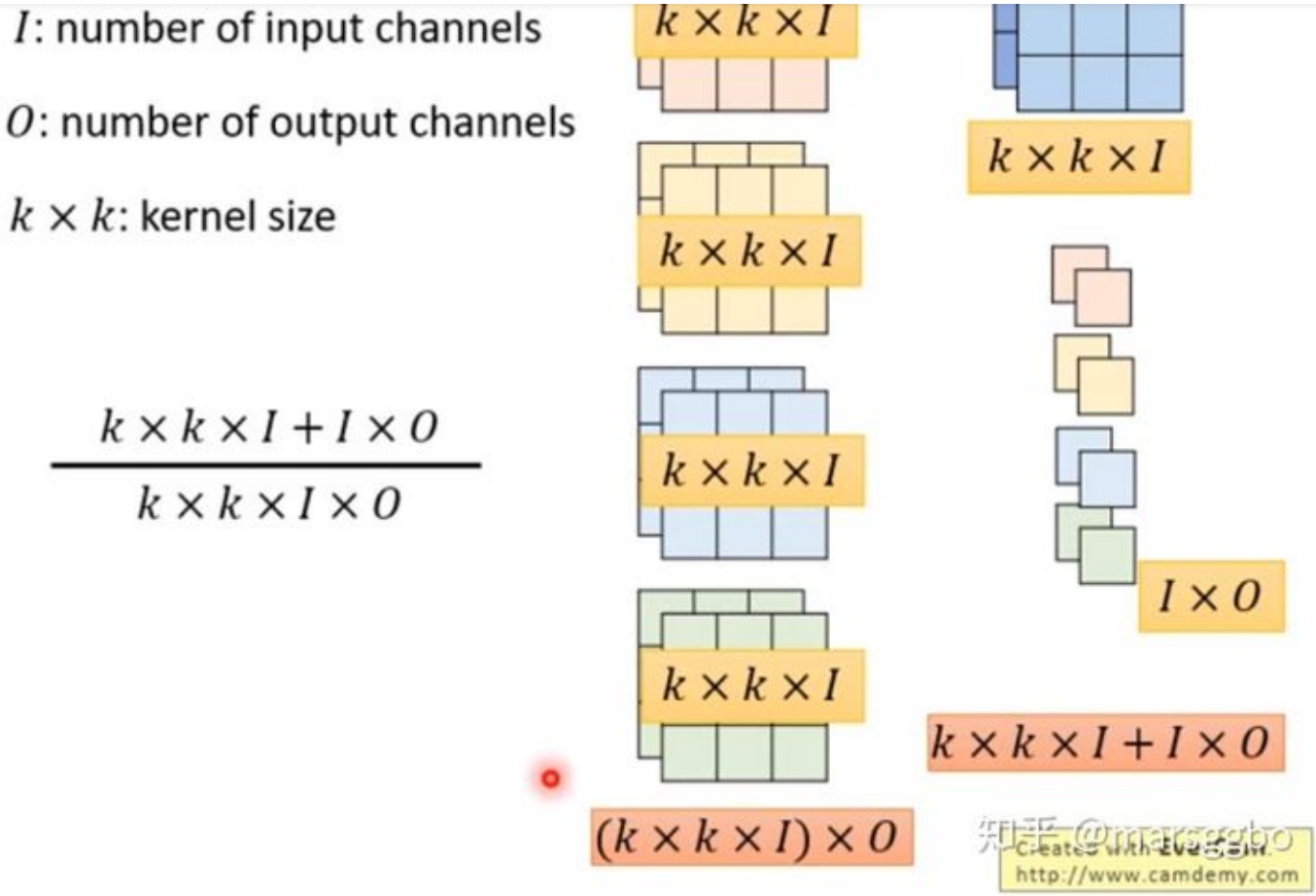


下面我们算一下标准卷积和Depthwise Separable卷积参数数量大小关系：假设输入特征图通道数为I,输出特征图通道数为O,卷积核大小为$k\times k$。

标准卷积参数数量= $\boldsymbol{k} \times \boldsymbol{k} \times \boldsymbol{I} \times \boldsymbol{O}$

Depthwise Separable卷积参数数量= $\boldsymbol{k} \times \boldsymbol{k} \times \boldsymbol{I} + \boldsymbol{I} \times \boldsymbol{O}$ 。

因为通常输出特征图的通道数 $\boldsymbol{O}$ 会设置的比较大，所以可以看到Depthwise Separable卷积的参数量会明显少于标准卷积。



这样的卷积设计广泛运用在各种小网络上，如**SqueezeNet,MobileNet,ShuffleNet,Xception**

六、Dynamic Computation

该方法的主要思路是如果目前的资源充足（比如你的手机电量充足），那么算法就尽量做到最好，比如训练更久，或者训练更多模型等；反之，如果当前资源不够（如电量只剩10%），那么就先算出一个过得去的结果。

那么如何实现呢?

1. 训练更多的model

比如说我们提前训练多种网络，比如大网络，中等网络和小网络，那么我们就可以根据资源情况来选择不同的网络。但是这样的缺点是我们需要保存多个模型，这在移动设备上的可操作性不高。

2. 使用中间层输出结果

这样的思路其实也挺直观的，就是比如说我们做分类任务，当资源有限时，我们可能只是基于前面几层提取到的特征做分类预测，但是一般而言这样得到的结果会打折扣，因为前面提取到的特征是比较细腻度的，可能只是一些纹理，而不是比较高层次抽象的特征。

右下角的图则展示了在不同中间层插入分类器对于模型训练的影响，可以看到越靠近输入层插入分类器，对模型的影响越大。其实也很好理解，因为一般而言，前面的网络结构负责提取浅层的特征，但是当我们在前面就插入分类器后，那么分类器为了得到较好的预测结果会强迫前面的网络结构提取一些抽象的特征，进而扰乱了后面特征的提取。具体的分析可以阅读这篇文章Multi-Scale Dense Networks for Resource Efficient Image Classification。