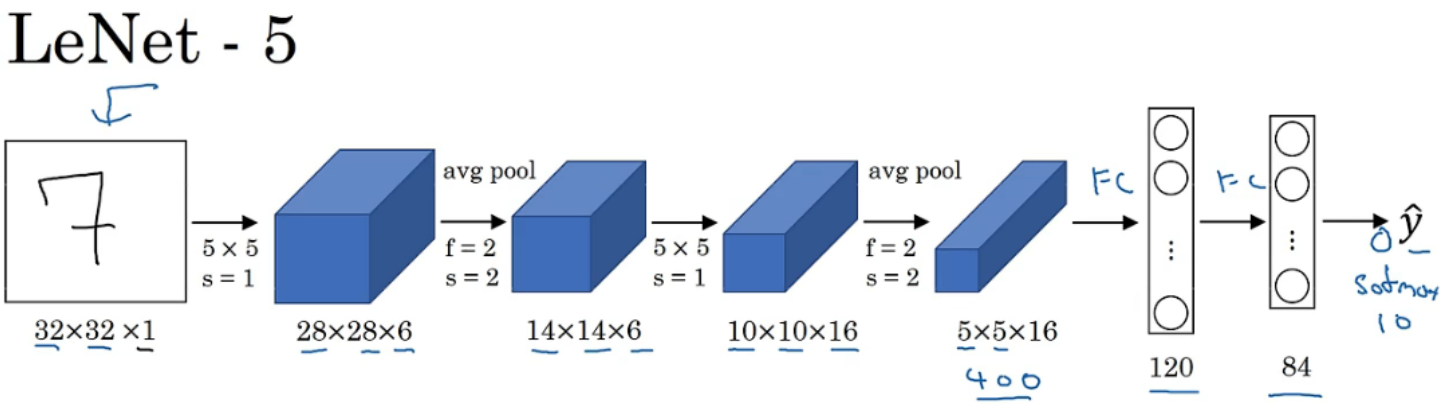
1. 经典的网络

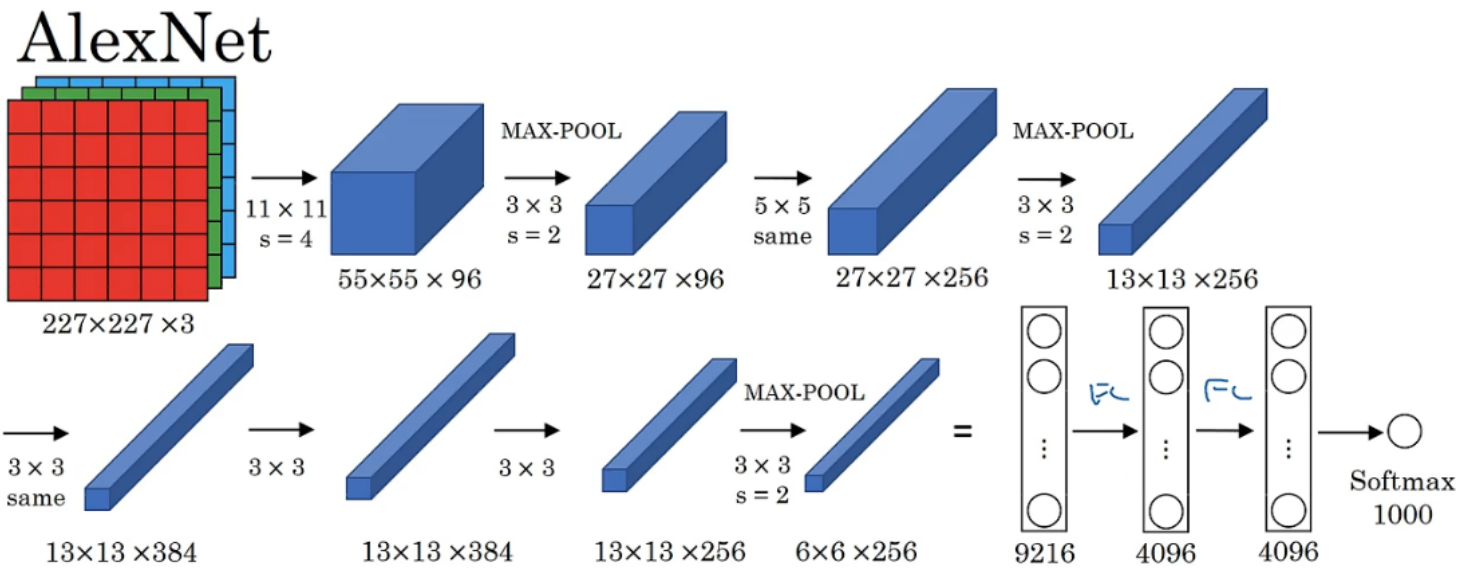
1.1 LeNet-5

针对灰度图像进行设计，input的大小为$32 \times 32 \times 1$，参数大约60k个。



- 随着网络层次的加深，图像的大小减少，但是图像的通道数量增加。
- Conv层后面跟着pooling层。

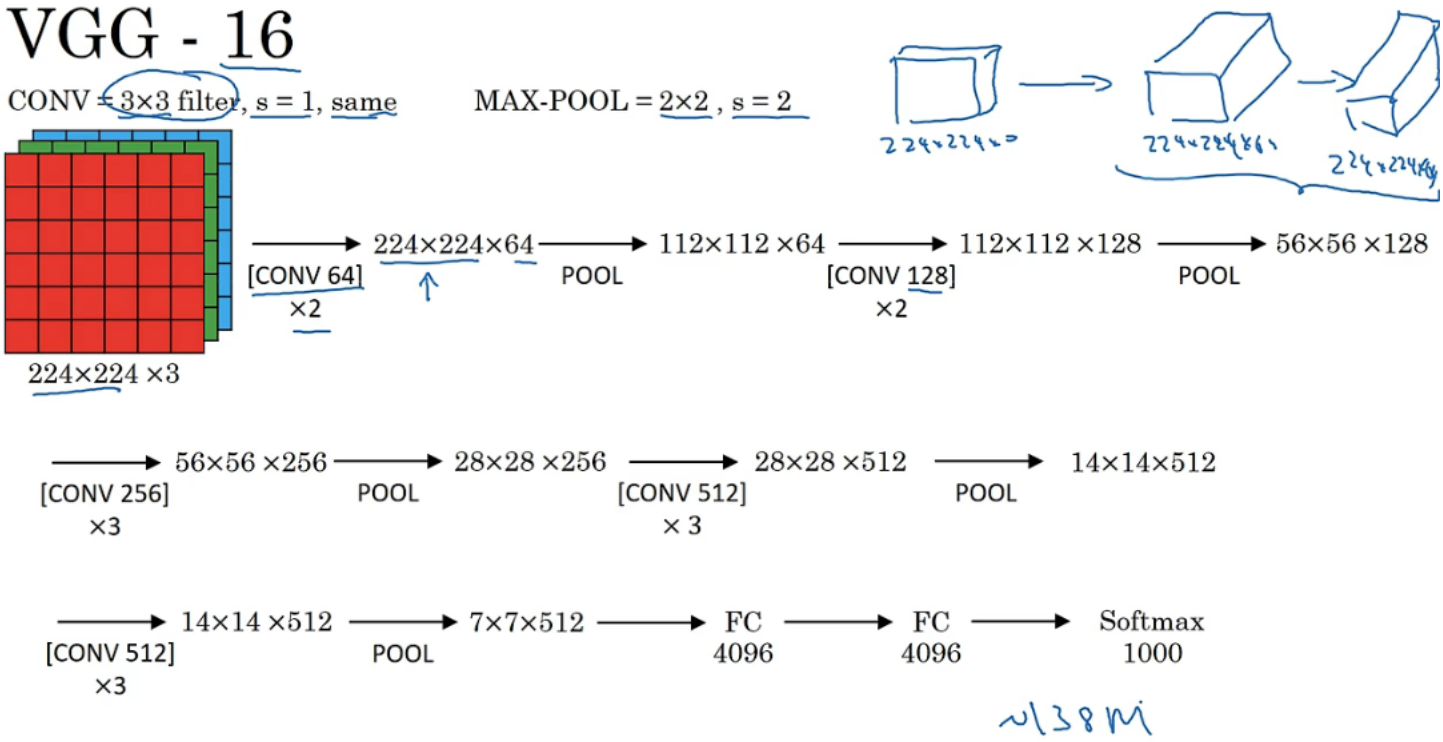
1.2 AlexNet



- AlexNet大约包含6000万个参数，性能比LeNet-5更出色；
- 使用了ReLU激活函数；
- 使用了多个GPUs；
- LRN层：没什么太大的用处。

1.3 VGG-16

- Conv：3×3大小的filter，stride=1，padding为same
- Pooling：2×2大小的filter，stride=1

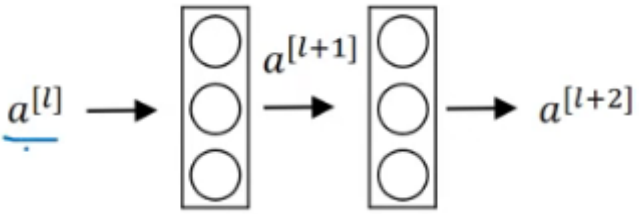


随着网络的加深，每次池化后图像的尺寸都会缩小一半，信道数量在不断增加。

2. ResNet

2.1 Residual block

ResNet由残差块构成。对于一个神经网络



其传播过程如下描述：

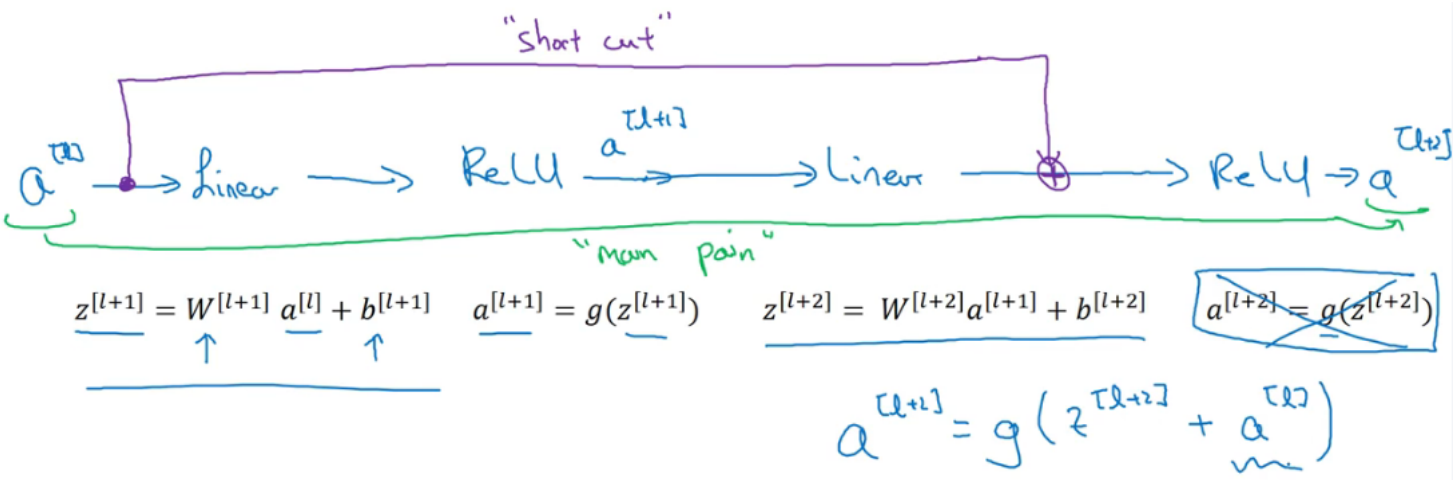
- Linear: $z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]}$
- ReLU: $a^{[l+1]} = g(z^{[l+1]})$
- Linear: $z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]}$
- ReLU: $a^{[l+2]} = g(z^{[l+2]})$

ResNet相当于在第$l+2$层使用ReLU前加上第l层的输出$a^{[l]}$，被称为short cut/skip connection：

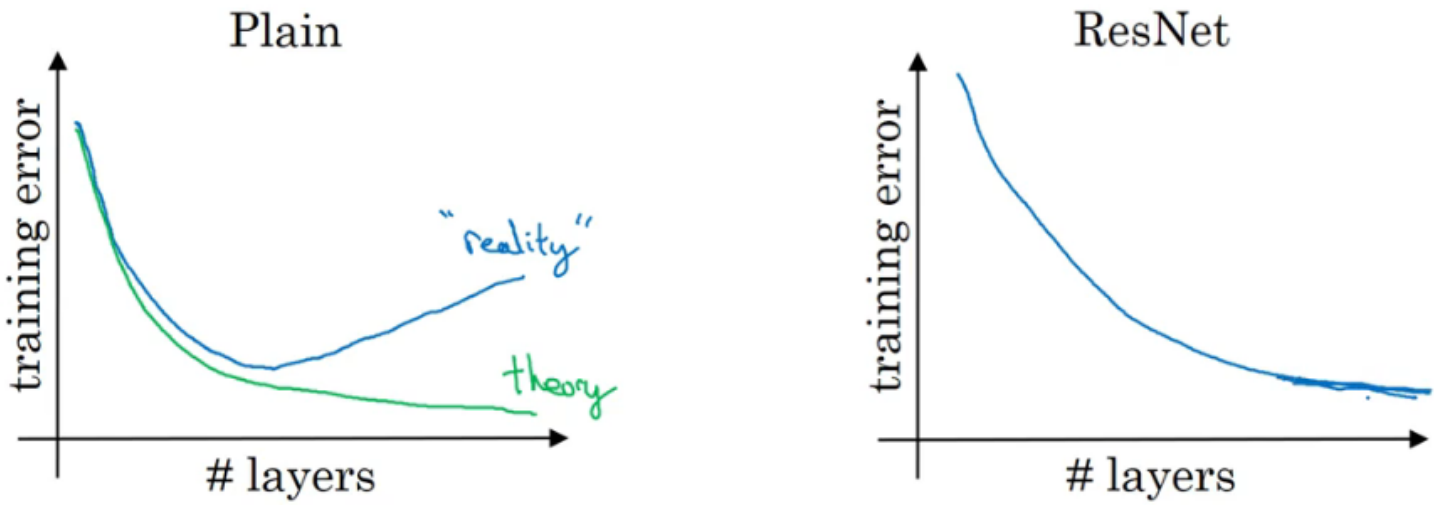
$$a^{[l+2]} = g(z^{[l+2]}) \rightarrow a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

2.2 Residual network

两层的神经元构成一个Residual block，ResNet的网络结构如下：



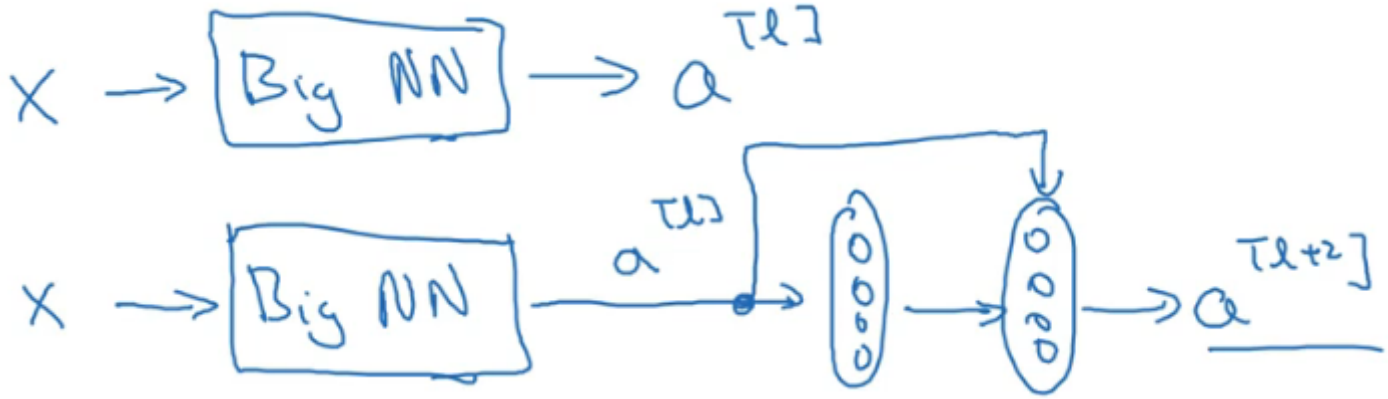
使用ResNet前后error的变化：



使用ResNet能够很好地解决深层网络中梯度消失和梯度爆炸的问题，保证良好的性能。

2.3 为什么ResNet有用?

假设一个非常大的神经网络，输入为X，输出为$a^{[l]}$，使用残差块让网络增加两层：



假设网络使用ReLU作为激活函数，因此输出都会大于零。

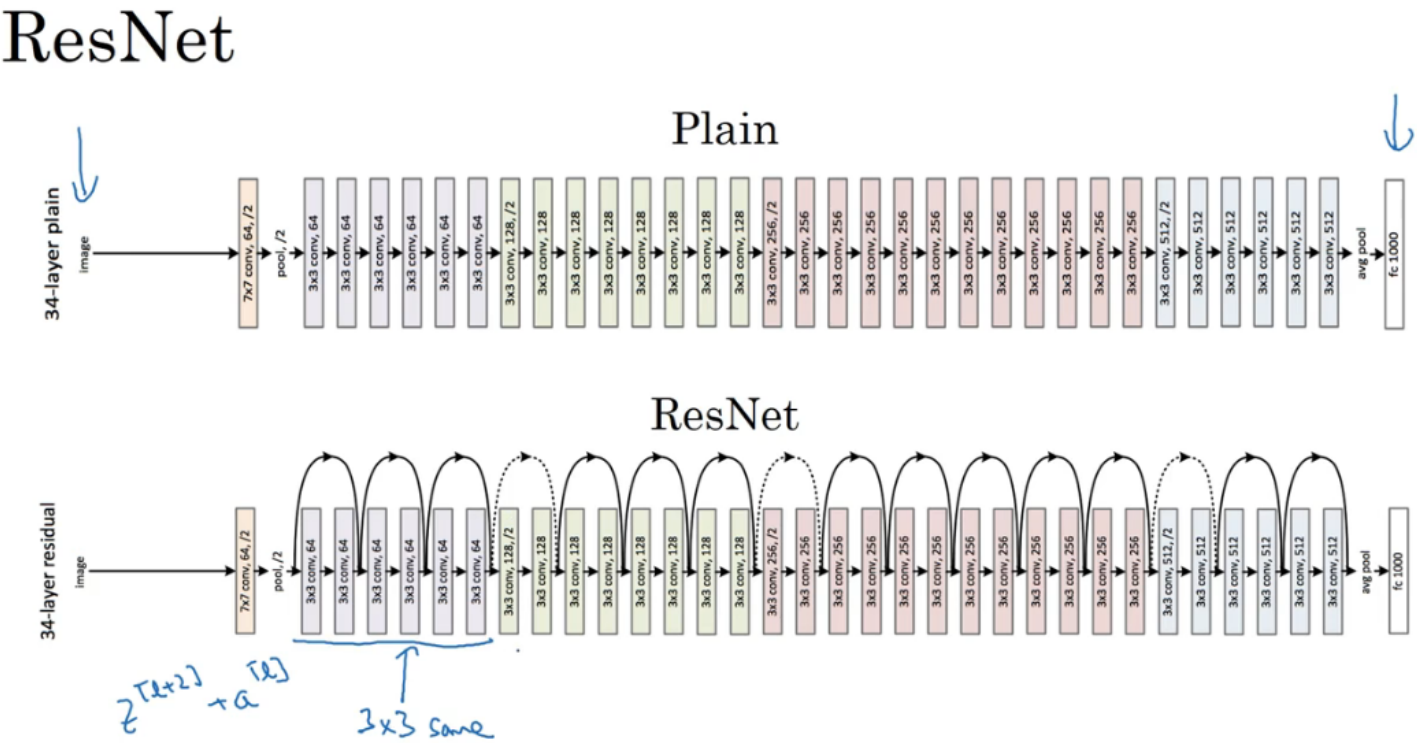
因为使用了ResNet，$a^{[l+2]} = g(z^{[l+2]} + a^{[l]}) = g(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]})$

如果使用l_2正则化或者是权重衰减，会压缩$W^{[l+2]}$的值，$b^{[l+2]}$也同理。如果$W^{[l+2]}$和$b^{[l+2]}$都为零，那么上式便退化为：$a^{[l+2]} = g(a^{[l]}) = ReLU(a^{[l]}) = a^{[l]}$。

这个等式对于ResNet来说非常容易，无论是把残差块放到网络的中间还是末端，网络的性能都不会受到影响。

根据上式可知$z^{[l+2]}$与$a^{[l]}$的维度相同，因此ResNet中使用了很多相同的卷积。

2.4 将普通网络转化为ResNet

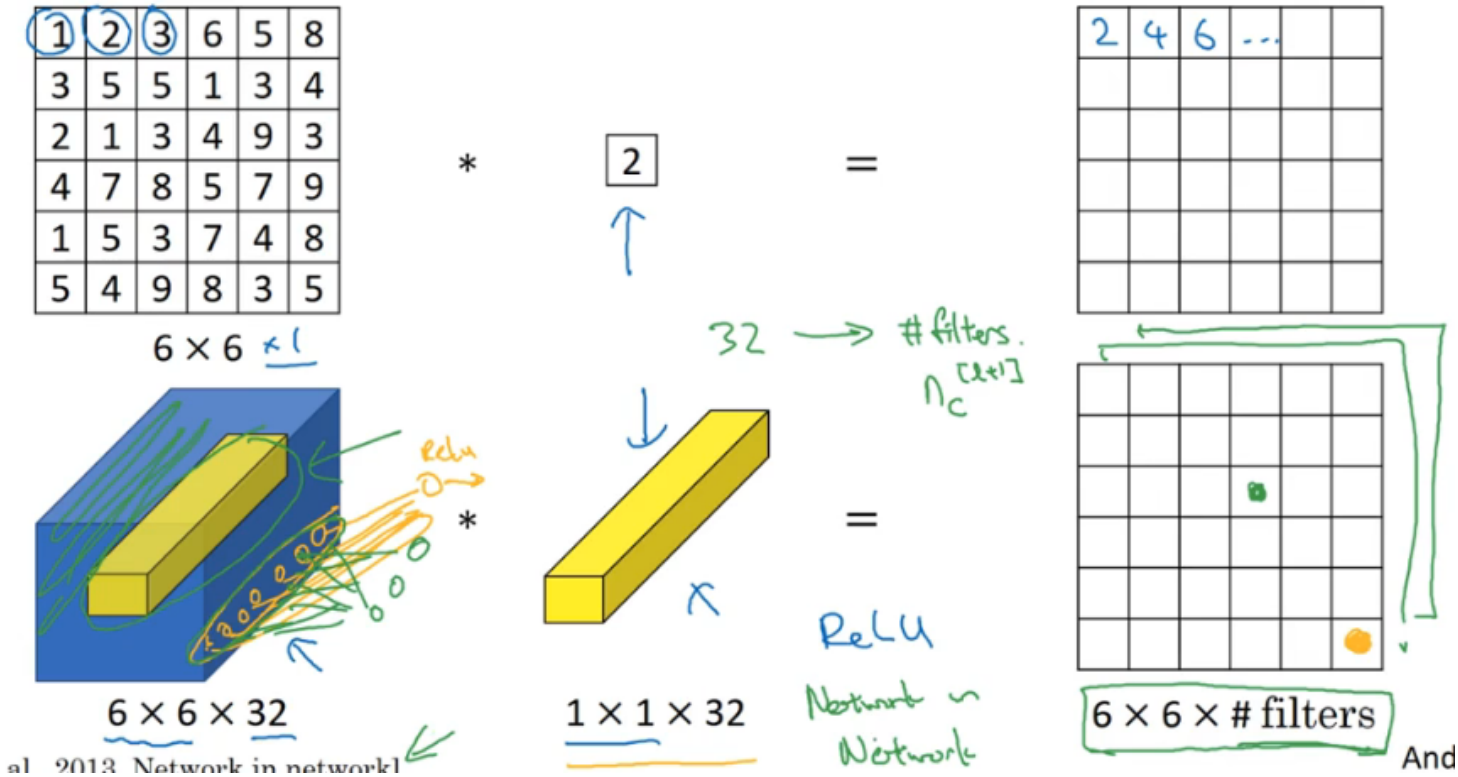


在两个相同的卷积层之间增加short cut。

3. 1×1卷积

如果是在二维上的卷积，那么相当于每个元素与卷积核相乘。

如果是在三维上与$1 \times 1 \times n_c$的卷积核进行卷积，相当于图像的$1 \times 1 \times n_c$的切片与卷积核进行点乘，再通过ReLU函数，输出结果。本质上是对切片上的单元使用了一个全连接的神经网络。



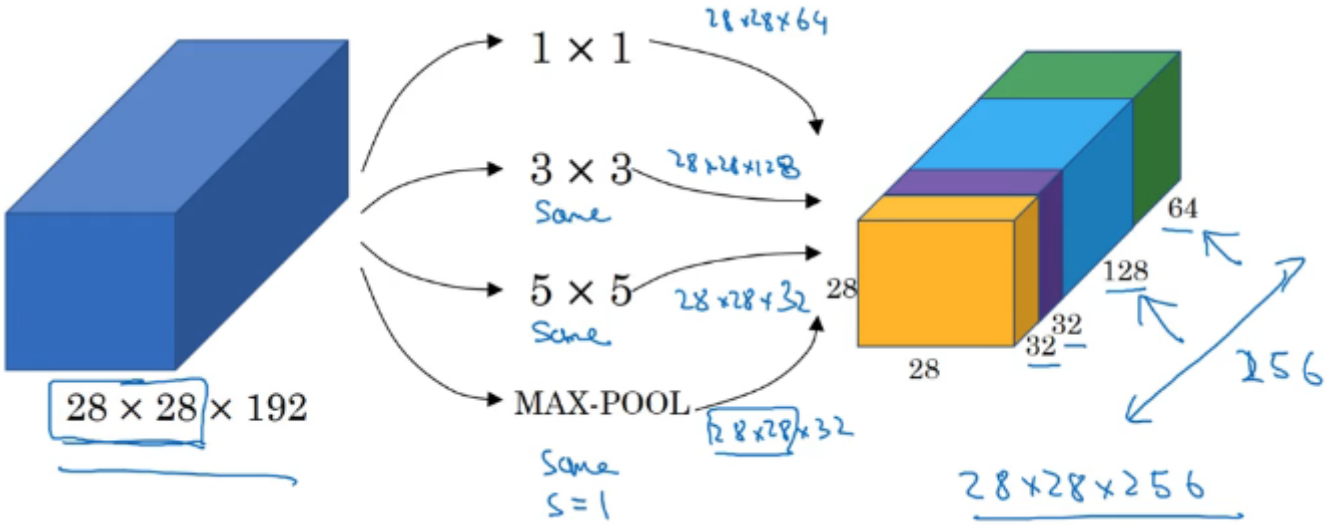
3.1 应用

- 维度压缩：新的通道数取决于你使用的1×1卷积核的个数。
- 增加非线性：使用的1×1卷积核的个数与原通道数相同。

4. Inception network

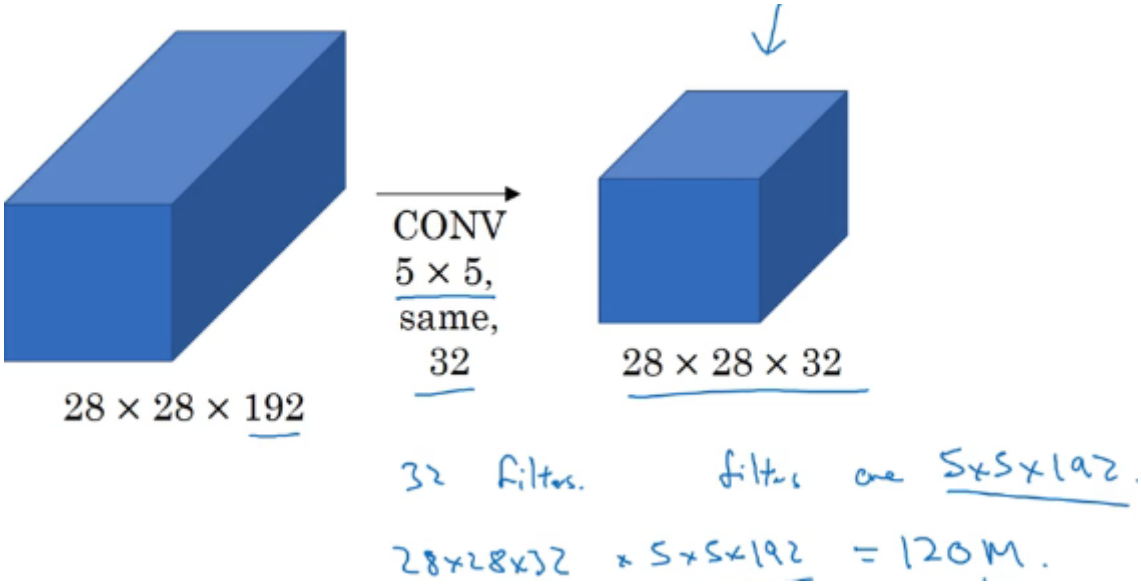
Inception network的作用是在构建深度卷积神经网络时无需考虑卷积核的大小以及是否添加池化层。

4.1 网络结构



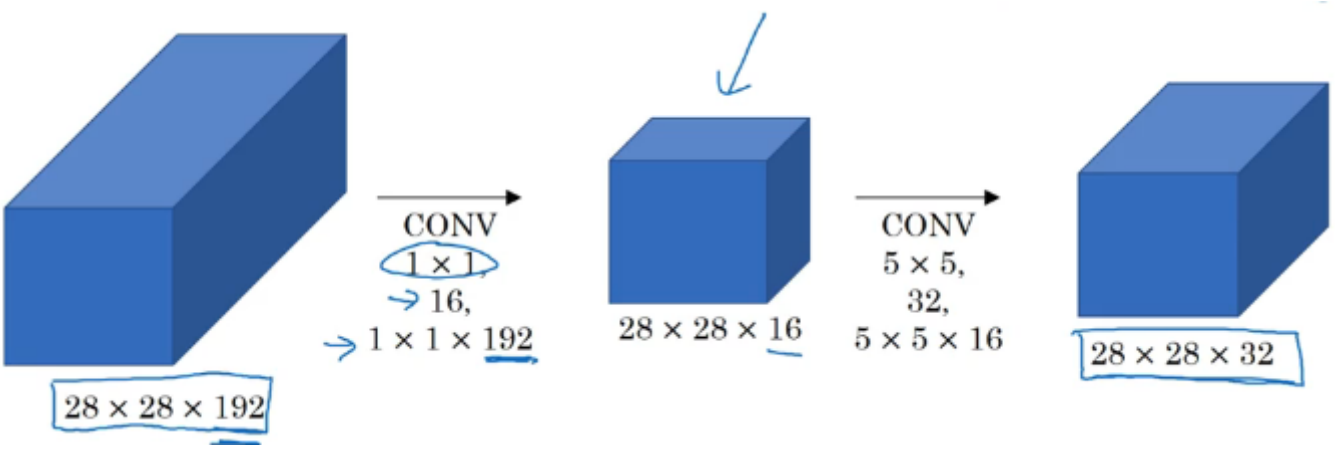
使用了不同大小的卷积核以及带padding的池化层，将输出结果堆叠。这个操作相当于增加了通道的数量。

4.2 计算成本



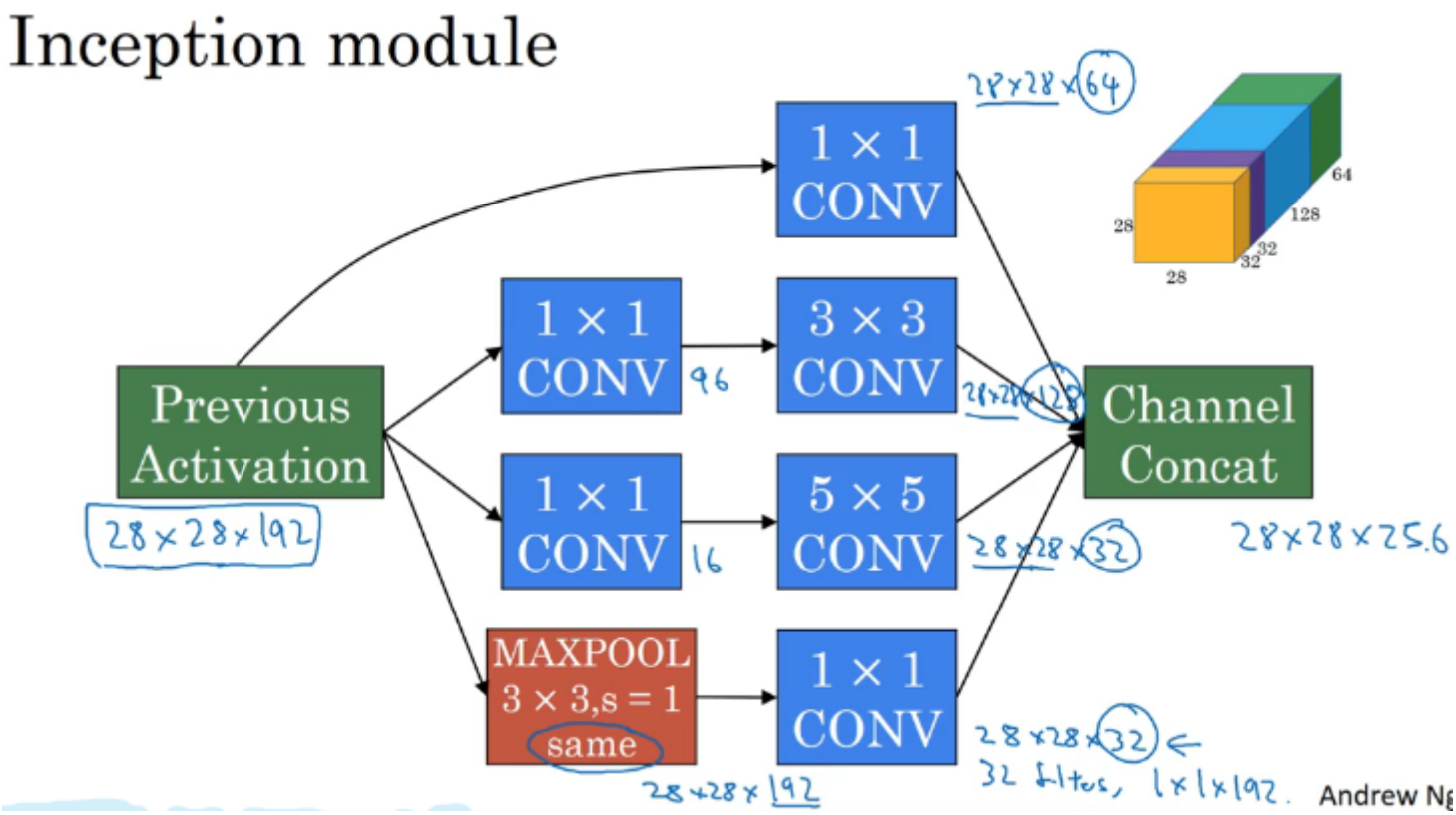
- filter的大小为$5 \times 5 \times 192$，个数为32。
- $computingcost = 28 \times 28 \times 32 \times 5 \times 5 \times 192 = 120M$

利用1×1的卷积核过渡，这一层被称为"bottleneck layer"：

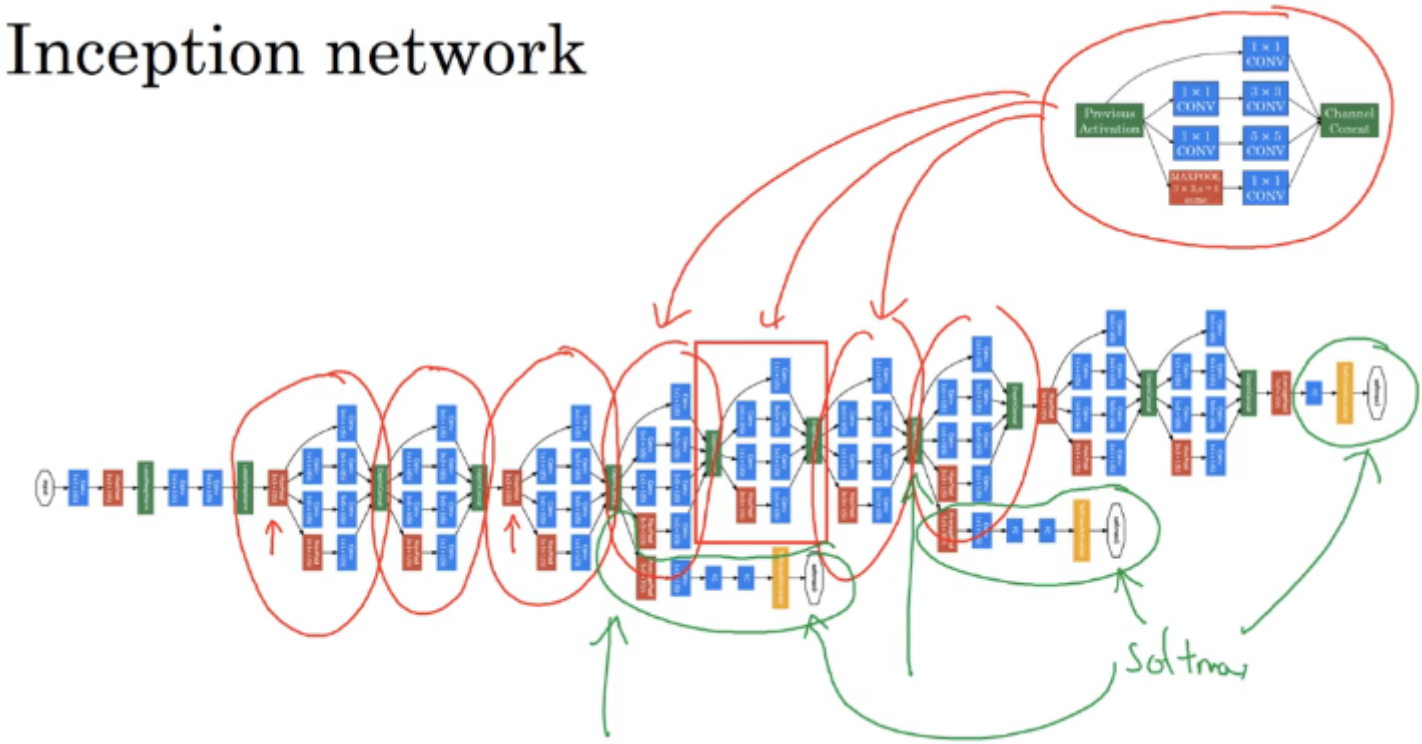


- 1×1的计算成本：$28 \times 28 \times 16 \times 1 \times 1 \times 192 = 2.4M$
- 5×5的计算成本：$28 \times 28 \times 32 \times 5 \times 5 \times 16 = 10.0M$
- $total = 2.4M + 10.0M = 12.4M$

4.3 Inception module



4.4 GoogleNet

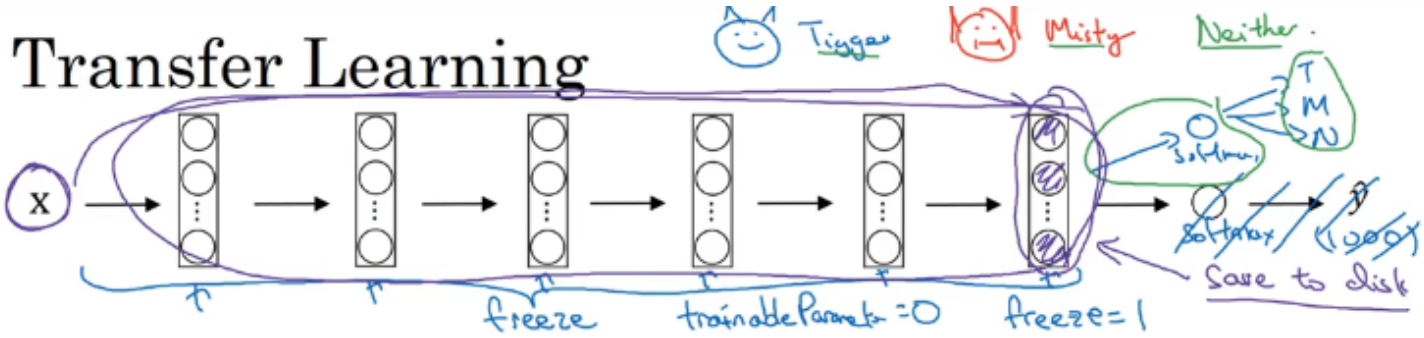


对网络的后几层的隐藏输出都用于预测，起到正则化的作用，能够防止网络的过拟合。

5. 迁移学习

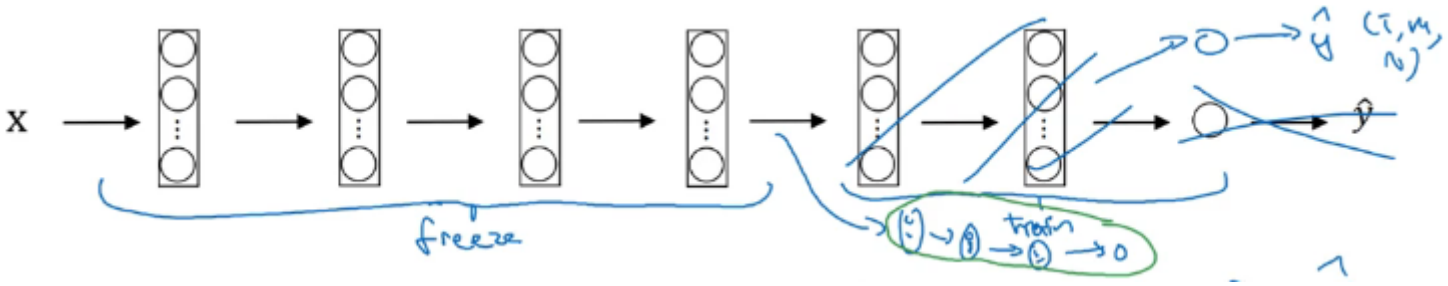
5.1 小数据集

利用迁移学习的思想，把已经预训练的模型的softmax层去掉，使用自己的少量数据训练自定义的softmax层。这样能够在小数据集上达到良好的效果。



5.2 大数据集

freeze更少的层数，增加网络末端训练的层数。数据集越大，freeze的层数就越小。如果一个数据集非常庞大，可以训练所有的参数，将预训练好的参数作为参数的初始值以此来代替随机初始化。

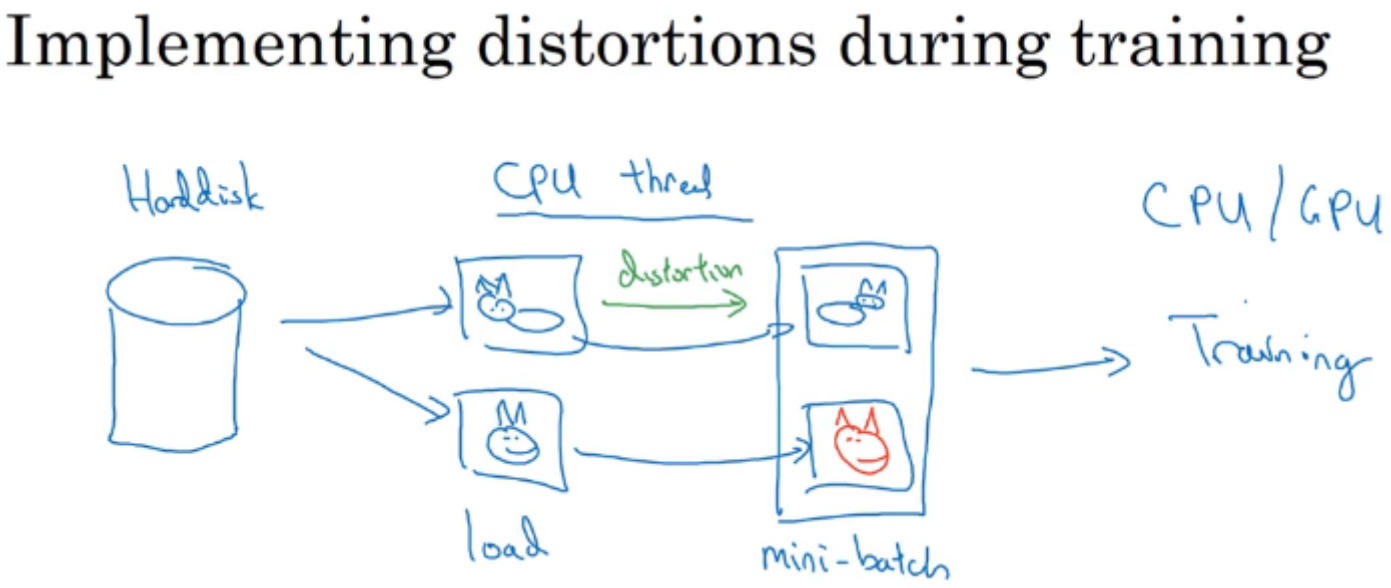


6. Data augmentation

6.1 常用方法

- 镜像翻转（Mirroring）
- 随机裁剪（Random Cropping）
- 色彩转换（Color shifting）：给RGB的每个通道增减数值。PCA颜色增强对图片的主色变化较大，图片的次色变化较小，使总体的颜色保持一致。

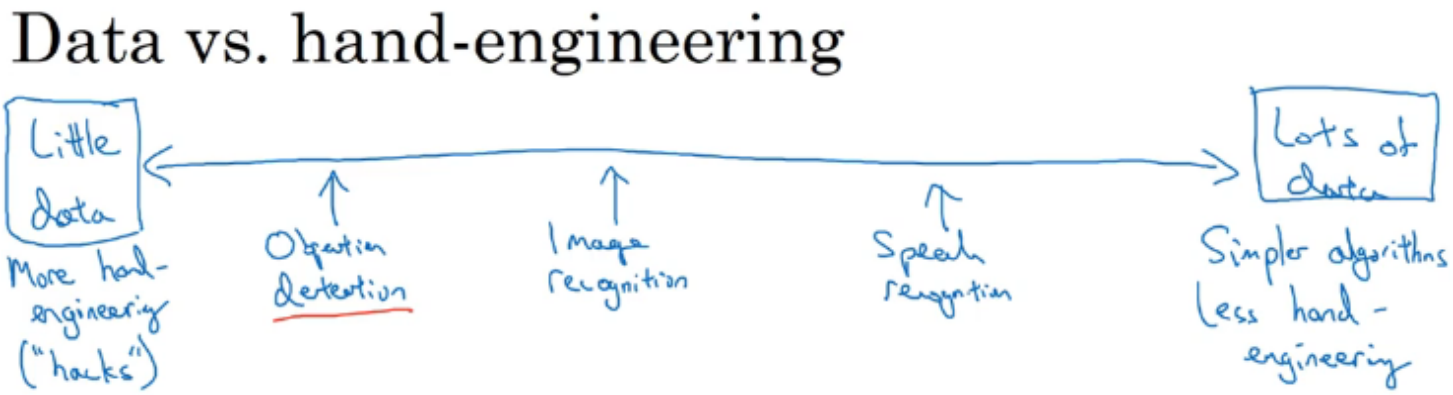
6.2 Data augmentation in training



Data augmentation和training可以利用的CPU的多线程来并行运行。

7. 计算机视觉现状

7.1 Data vs. hand-engineering



- 在有大量数据的时候，更倾向于使用更简单的算法和更少的hand-engineering。因为这时不需要为这个问题精心设计特征，我们使用一个大型的网络或者一个简单的架构就能够解决。
- 在只有少量数据时，应该有更多的hand-engineering。因为模型很难从少量的数据中获取到足够的特征。

对于机器学习的数据，主要有两种知识来源：

- 被标记的数据；
- 手工特征工程/网络结构/其他构建。

7.2 Doing well on benchmarks

- Ensembling: 独立训练多个网络，输出结果平均或者加权平均；
- 测试时的Multi-crop: 在测试图片的多个版本上运行分类器，取平均结果。