AlexNet　　2012 年 ImageNet 比赛冠军

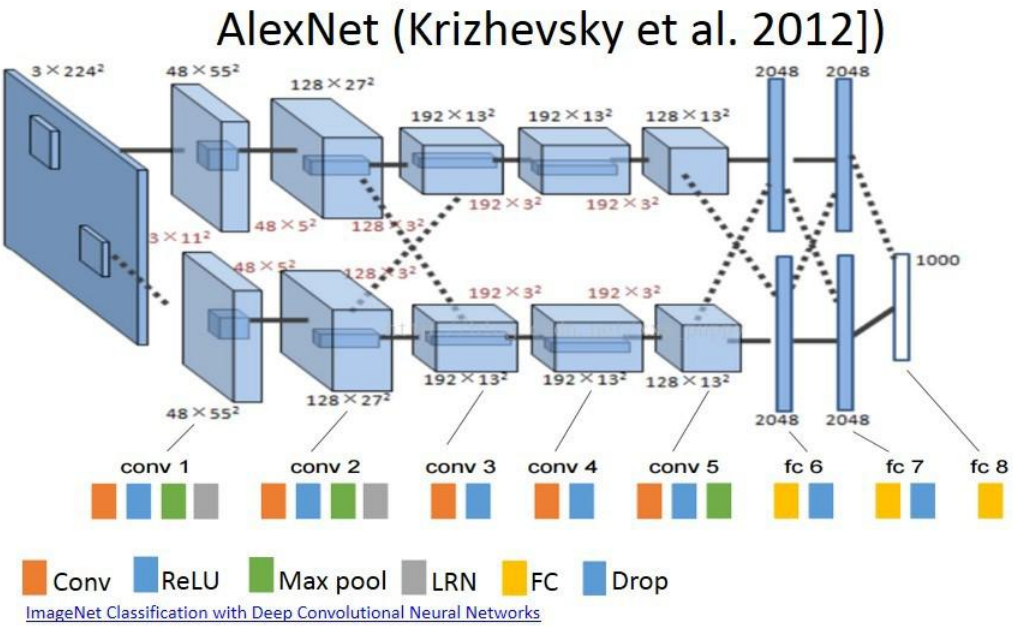
AlextNet 与 LeNet 的设计理念非常相似。但也有显著的区别。

第一，AlexNet 包含 8 层变换，其中有五层卷积和两层全连接隐含层，以及一个全连接输出层。第一层中的卷积窗口形状是 11×11。因为 ImageNet 中绝大多数图像的高和宽均比 MNIST 图像的高和宽大十倍以上，ImageNet 图像的物体占用更多的像素，所以需要更大的卷积窗口来捕获物体。第二层中的卷积窗口形状减小到 5×5，之后全采用 3×3。此外，第一、第二和第五个卷积层之后都使用了窗口形状为 3×3、步幅为 2 的最大池化层。而且，AlexNet 使用的卷积通道数也数十倍大于 LeNet 中的卷积通道数。紧接着最后一个卷积层的是两个输出个数为 4096 的全连接层。这两个巨大的全连接层带来将近 1GB 的模型参数。由于早期 GPU 显存的限制，最早的 AlexNet 使用双数据流的设计使得一个 GPU 只需要处理一半模型。

第二，AlextNet 将 sigmoid 激活函数改成了更加简单的 ReLU 激活函数。ReLU 激活函数的计算更简单，同时在不同的参数初始化方法下使模型更容易训练。

第三，通过丢弃法（dropout）来控制全连接层的模型复杂度。

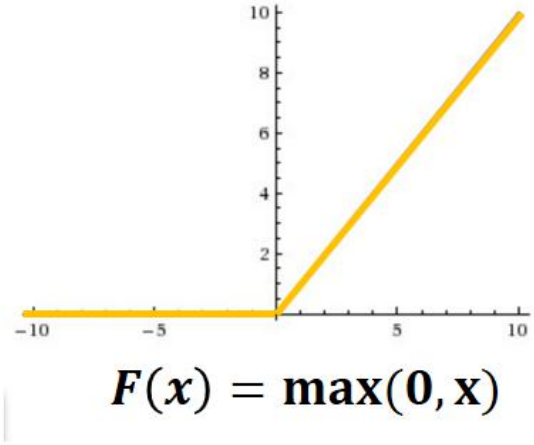
第四，引入了大量的图像增广，例如翻转、裁剪和颜色变化，进一步扩大数据集来缓解过拟合。

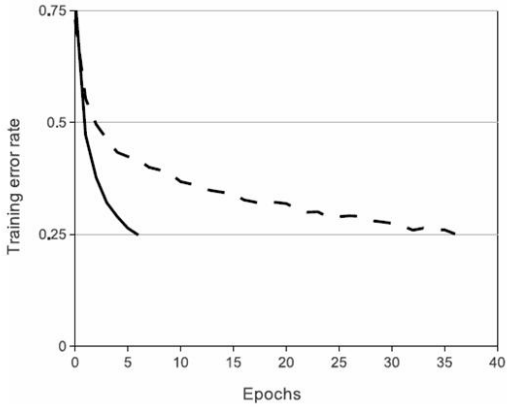


1、使用 ReLU 激活函数

传统的神经网络普遍使用 Sigmoid 或者 tanh 等非线性函数作为激励函数，然而它们容易出现梯度弥散或梯度饱和的情况。以 Sigmoid 函数为例，当输入的值非常大或者非常小的时候，这些神经元的梯度接近于 0（梯度饱和现象），如果输入的初始值很大的话，梯度在反向传播时因为需要乘上一个 Sigmoid 导数，会造成梯度越来越小，导致网络变的很难学习。（详见本公博客的文章：深度学习中常用的激励函数）。

在 AlexNet 中，使用了 ReLU （Rectified Linear Units）激励函数，该函数的公式为：f(x)=max(0,x)，当输入信号<0 时，输出都是 0，当输入信号>0 时，输出等于输入，如下图 1 所示：使用 ReLU 替代 Sigmoid/tanh，由于 ReLU 是线性的，且导数始终为 1，计算量大大减少，收敛速度会比 Sigmoid/tanh 快很多，如下图 2 所示：



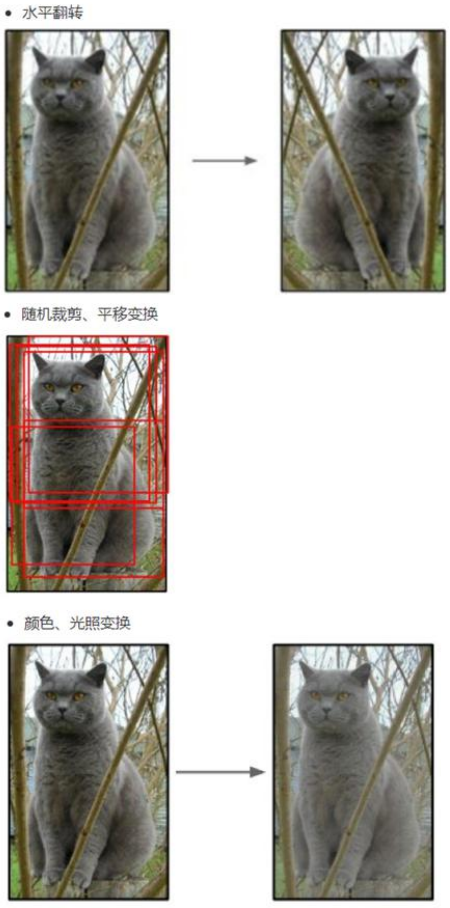


ReLU激活函数（实线）、tanh激活函数（虚线）对CIFAR10数据集的训练误差收敛速度对比图

注：来自经典论文《ImageNet Classification with Deep Convolutional Neural Networks》

2、**数据扩充（Data augmentation）**

有一种观点认为神经网络是靠数据喂出来的，如果能够增加训练数据，提供海量数据进行训练，则能够有效提升算法的准确率，因为这样可以避免过拟合，从而可以进一步增大、加深网络结构。而当训练数据有限时，可以通过一些变换从已有的训练数据集中生成一些新的数据，以快速地扩充训练数据。其中，最简单、通用的图像数据变形的方式：水平翻转图像，从原始图像中随机裁剪、平移变换，颜色、光照变换，如下图所示：



AlexNet 在训练时，在数据扩充（data augmentation）这样处理：

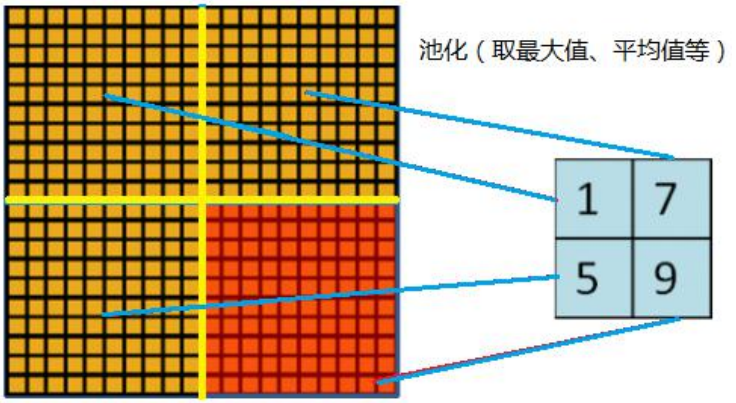
（1）随机裁剪，对 256×256 的图片进行随机裁剪到 224×224，然后进行水平翻转，相当于将样本数量增加了((256-224)^2)×2=2048 倍；

（2）测试的时候，对左上、右上、左下、右下、中间分别做了 5 次裁剪，然后翻转，共 10 个裁剪，之后对结果求平均。作者说，如果不做随机裁剪，大网络基本上都过拟合；

（3）对 RGB 空间做 PCA（主成分分析），然后对主成分做一个（0, 0.1）的高斯扰动，也就是对颜色、光照作变换，结果使错误率又下降了 1%。

3、**重叠池化（Overlapping Pooling）**

一般的池化（Pooling）是不重叠的，池化区域的窗口大小与步长相同，如下图所示：



在 AlexNet 中使用的池化（Pooling）却是可重叠的，也就是说，在池化的时候，每次移动的步长小于池化的窗口长度。AlexNet 池化的大小为 3×3 的正方形，每次池化移动步长为 2，这样就会出现重叠。重叠池化可以避免过拟合，这个策略贡献了 0.3%的 Top-5 错误率。

4、**局部归一化（Local Response Normalization，简称 LRN）**

在神经生物学有一个概念叫做“侧抑制”（lateral inhibitio），指的是被激活的神经元抑制相邻神经元。归一化（normalization）的目的是“抑制”，局部归一化就是借鉴了“侧抑制”的思想来实现局部抑制，尤其当使用 ReLU 时这种“侧抑制”很管用，因为 ReLU 的响应结果是无界的（可以非常大），所以需要归一化。使用局部归一化的方案有助于增加泛化能力。

LRN 的公式如下，核心思想就是利用临近的数据做归一化，这个策略贡献了 1.2%的 Top-5 错误率。

$$b_{x,y}^{i}=a_{x,y}^{i}\Big/\left(k+\alpha\sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)}(a_{x,y}^{j})^{2}\right)^{\beta}$$

$a_{x,y}^{i}$表示使用核i作用于(x,y)然后再采用ReLU非线性函数计算得到的活跃度

N是该层的核的总数目

常数k,n,α,β是超参数，它们的值使用一个验证集来确定

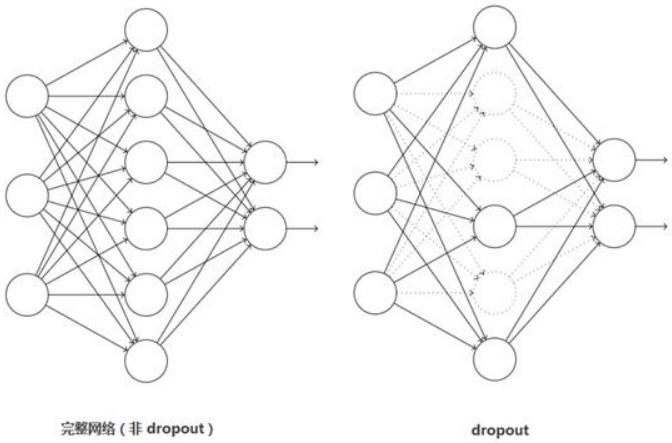
本文使用 $k=2,n=5,\alpha=10^{-4},\beta=0.75$

5、**Dropout**

引入 Dropout 主要是为了防止过拟合。在神经网络中 Dropout 通过修改神经网络本身结构来实现，对于某一层的神经元，通过定义的概率将神经元置为 0，这个神经元就不参与前向和后向传播，就如同在网络中被删除了一样，同时保持输入层与输出层神经元的个数不变，然后按照神经网络的学习方法进行参数更新。在下一次迭代中，又重新随机删除一些神经元（置为 0），直至训练结束。

Dropout 应该算是 AlexNet 中一个很大的创新，以至于“神经网络之父”Hinton 在后来很

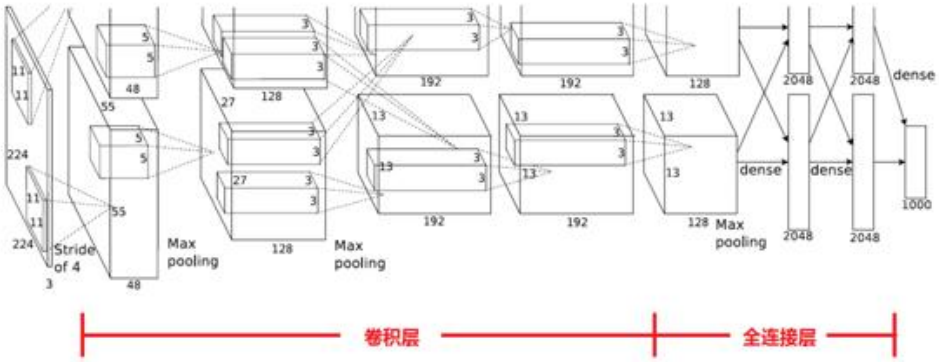
长一段时间里的演讲中都拿 Dropout 说事。Dropout 也可以看成是一种模型组合，每次生成的网络结构都不一样，通过组合多个模型的方式能够有效地减少过拟合，Dropout 只需要两倍的训练时间即可实现模型组合（类似取平均）的效果，非常高效。
如下图所示：



6、**多 GPU 训练**

AlexNet 当时使用了 GTX580 的 GPU 进行训练，由于单个 GTX 580 GPU 只有 3GB 内存，这限制了在其上训练的网络的最大规模，因此他们在每个 GPU 中放置一半核（或神经元），将网络分布在两个 GPU 上进行并行计算，大大加快了 AlexNet 的训练速度。

逐层解析



AlexNet 网络结构共有 8 层，前面 5 层是卷积层，后面 3 层是全连接层，最后一个全连接层的输出传递给一个 1000 路的 softmax 层，对应 1000 个类标签的分布。由于 AlexNet 采用了两个 GPU 进行训练，因此，该网络结构图由上下两部分组成，一个 GPU 运行图上方的层，另一个运行图下方的层，两个 GPU 只在特定的层通信。例如第二、四、五层卷积层的核只和同一个 GPU 上的前一层的核特征图相连，第三层卷积层和第二层所有的核特征图相连接，全连接层中的神经元和前一层中的所有神经元相连接。

1、第一层（卷积层）

（1）卷积

输入的原始图像大小为 224×224×3（RGB 图像），在训练时会经过预处理变为 227×227×3。在本层使用 96 个 11×11×3 的卷积核进行卷积计算，生成新的像素。由于采用了两个 GPU 并行运算，因此，网络结构图中上下两部分分别承担了 48 个卷积核的运算。

卷积核沿图像按一定的步长往 x 轴方向、y 轴方向移动计算卷积，然后生成新的特征图，其大小为：floor((img_size - filter_size)/stride) +1 = new_feture_size，其中 floor 表示向下取整，img_size 为图像大小，filter_size 为核大小，stride 为步长，new_feture_size 为卷积后的特征图大小，这个公式表示图像尺寸减去卷积核尺寸除以步长，再加上被减去的核大小像素对应生成的一个像素，结果就是卷积后特征图的大小。

AlexNet 中本层的卷积移动步长是 4 个像素，卷积核经移动计算后生成的特征图大小为 (227-11)/4+1=55，即 55×55。

（2）ReLU

卷积后的 55×55 像素层经过 ReLU 单元的处理，生成激活像素层，尺寸仍为 2 组 55×55×48 的像素层数据。

（3）池化

RuLU 后的像素层再经过池化运算，池化运算的尺寸为 3×3，步长为 2，则池化后图像的尺寸为 (55-3)/2+1=27，即池化后像素的规模为 27×27×96。

（4）归一化

池化后的像素层再进行归一化处理，归一化运算的尺寸为 5×5，归一化后的像素规模不变，仍为 27×27×96，这 96 层像素层被分为两组，每组 48 个像素层，分别在一个独立的 GPU 上进行运算。

2、第二层（卷积层）

（1）卷积

第二层的输入数据为第一层输出的 27×27×96 的像素层（被分成两组 27×27×48 的像素层放在两个不同 GPU 中进行运算），为方便后续处理，在这里每幅像素层的上下左右边缘都被填充了 2 个像素（填充 0），即图像的大小变为 (27+2+2) ×(27+2+2)。第二层的卷积核大小为 5×5，移动步长为 1 个像素，跟第一层第（1）点的计算公式一样，经卷积核计算后的像素层大小变为 (27+2+2-5)/1+1=27，即卷积后大小为 27×27。

本层使用了 256 个 5×5×48 的卷积核，同样也是被分成两组，每组为 128 个，分给两个 GPU 进行卷积运算，结果生成两组 27×27×128 个卷积后的像素层。

（2）ReLU

这些像素层经过 ReLU 单元的处理，生成激活像素层，尺寸仍为两组 27×27×128 的像素层。

（3）池化

再经过池化运算的处理，池化运算的尺寸为 3×3，步长为 2，池化后图像的尺寸为 (57-3)/2+1=13，即池化后像素的规模为 2 组 13×13×128 的像素层

（4）归一化

然后再经归一化处理，归一化运算的尺度为 5×5，归一化后的像素层的规模为 2 组 13×13×128 的像素层，分别由 2 个 GPU 进行运算。

3、第三层（卷积层）

（1）卷积

第三层输入数据为第二层输出的 2 组 13×13×128 的像素层，为便于后续处理，每幅像素层的上下左右边缘都填充 1 个像素，填充后变为 (13+1+1)×(13+1+1)×128，分布在两个 GPU 中进行运算。

这一层中每个 GPU 都有 192 个卷积核，每个卷积核的尺寸是 3×3×256。因此，每个 GPU 中的卷积核都能对 2 组 13×13×128 的像素层的所有数据进行卷积运算。如该层的结构图所示，两个 GPU 有通过交叉的虚线连接，也就是说每个 GPU 要处理来自前一层的所有 GPU 的输入。
本层卷积的步长是 1 个像素，经过卷积运算后的尺寸为 (13+1+1-3)/1+1=13，即每个 GPU 中共 13×13×192 个卷积核，2 个 GPU 中共有 13×13×384 个卷积后的像素层。

（2）ReLU

卷积后的像素层经过 ReLU 单元的处理，生成激活像素层，尺寸仍为 2 组 13×13×192 的像素层，分配给两组 GPU 处理。

4、第四层（卷积层）

（1）卷积

第四层输入数据为第三层输出的 2 组 13×13×192 的像素层，类似于第三层，为便于后续处理，每幅像素层的上下左右边缘都填充 1 个像素，填充后的尺寸变为 (13+1+1)×(13+1+1)×192，分布在两个 GPU 中进行运算。
这一层中每个 GPU 都有 192 个卷积核，每个卷积核的尺寸是 3×3×192（与第三层不同，第四层的 GPU 之间没有虚线连接，也即 GPU 之间没有通信）。卷积的移动步长是 1 个像素，经卷积运算后的尺寸为 (13+1+1-3)/1+1=13，每个 GPU 中有 13×13×192 个卷积核，2 个 GPU 卷积后生成 13×13×384 的像素层。

（2）ReLU

卷积后的像素层经过 ReLU 单元处理，生成激活像素层，尺寸仍为 2 组 13×13×192 像素层，分配给两个 GPU 处理。

5、第五层（卷积层）

（1）卷积

第五层输入数据为第四层输出的 2 组 13×13×192 的像素层，为便于后续处理，每幅像素层的上下左右边缘都填充 1 个像素，填充后的尺寸变为 (13+1+1)×(13+1+1) ，2 组像素层数据被送至 2 个不同的 GPU 中进行运算。
这一层中每个 GPU 都有 128 个卷积核，每个卷积核的尺寸是 3×3×192，卷积的步长是 1 个像素，经卷积后的尺寸为 (13+1+1-3)/1+1=13，每个 GPU 中有 13×13×128 个卷积核，2 个 GPU 卷积后生成 13×13×256 的像素层。

（2）ReLU

卷积后的像素层经过 ReLU 单元处理，生成激活像素层，尺寸仍为 2 组 13×13×128 像素层，由两个 GPU 分别处理。

（3）池化

2 组 13×13×128 像素层分别在 2 个不同 GPU 中进行池化运算处理，池化运算的尺寸为 3×3，步长为 2，池化后图像的尺寸为 (13-3)/2+1=6，即池化后像素的规模为两组 6×6×128 的像素层数据，共有 6×6×256 的像素层数据。

6、第六层（全连接层）

（1）卷积（全连接）

第六层输入数据是第五层的输出，尺寸为 6×6×256。本层共有 4096 个卷积核，每个卷积核的尺寸为 6×6×256，由于卷积核的尺寸刚好与待处理特征图（输入）的尺寸相同，即卷积核中的每个系数只与特征图（输入）尺寸的一个像素值相乘，一一对应，因此，该层被称为全连接层。由于卷积核与特征图的尺寸相同，卷积运算后只有一个值，因此，卷积后的像素层尺寸为 4096×1×1，即有 4096 个神经元。

（2）ReLU

这 4096 个运算结果通过 ReLU 激活函数生成 4096 个值。

（3）Dropout

然后再通过 Dropout 运算，输出 4096 个结果值。

7、第七层（全连接层）

第七层的处理流程为：全连接-->ReLU-->Dropout 第六层输出的 4096 个数据与第七层的 4096 个神经元进行全连接，然后经 ReLU 进行处理后生成 4096 个数据，再经过 Dropout 处理后输出 4096 个数据。

8、第八层（全连接层）

第七层输出的 4096 个数据与第八层的 1000 个神经元进行全连接，经过训练后输出 1000 个 float 型的值，这就是预测结果。