读AlexNet

《ImageNet Classification with Deep Convolutional Neural Networks》

Abstract

你看这个卷积神经网络，他又大又神，我们用他处理1.2百万个高分辨率图的分类问题，1000个类别。（ImageNet LSVRC-2010那个比赛）然后我们成绩很吊，top1 top5是37.% 17.0% （*查了一下：top1就是猜一次猜对的概率，top5是猜5次。*）我们这个网络有6千万个参数，和65万个神经元，由5个卷积层组成，一些卷积层后边跟着max-pooling，然后还有3个全连接层，最后是softmax。为了让他更快，我们用了non-saturating neurons和和GPU（*non-saturating neurons不懂，gpu那块不看*）。然后为了降低过拟合，用了dropout。我们还参加了另一个比赛，成绩也很吊。

Introduction

现在的目标识别的方法都用刀了机器学习的方法。为了提高他们的表现，我们需要一个更大的训练数据，来得到更牛逼的模型，还有要用更好的技术来防止过拟合。直到最近，被标记的图像还相对比较小，只有几万个。这个体量的数据可以解决一些简单的任务，尤其是借助data augmentation的方法。比如，现在数字识别的问题上，已经很接近人类水平，错误率<0.3%。但是现实中的对象有更大的不确定性。所以学习识别他们需要更大的训练集。实际上，数据量小的缺点已经被广泛认识到了，但直到最近，搜集百万级别的图片才变得可能。新的大数据包括LabelMe有几十万个完全分隔的图片，还有ImageNet有超过一千五百万个被标注的高分辨率图像，超过22000个类别。

从千万张图例学习几千种物体，我们需要一个学习能力更厉害的模型。然而呢，即使有ImageNet提供的这么大的数据集，但对象识别问题本身包含的巨大的复杂性，导致这问题还是不好解决，所以我们的模型还需要先验知识来补偿我们所欠缺的数据。（*额原来说的复杂性就是图片还是不够吗*）卷积神经网络中有一类这样的模型[16,11,13,18,15,22,26]。可以通过改变他们的深度和广度，来控制他们的学习能力，而且他们天生可以处理图片的自然属性（*后一点是说局部特征吧，前边不太懂，插一个*

看吴恩达采访lecun时候lecun说，alex在给那些做对象识别的人讲这个论文时候，他以为在做的毕竟都是大佬，应该都知道卷积网络，结果其实下边人都呆了，不知道是什么东西。 所以那个年代大家都用传统的方法去竞赛时候，出来一个人用什么卷积玩意儿拿了第一，hh）,因此，相对于传统的用尺寸差不多的隐藏层前向传播的神经网络，虽然cnns在理论上的最佳表现会稍差一点点，但是由于cnns的连接和参数少得多，所以他好训练的多。

但即使，cnns有这样的迷人特质，而且因为注重局部特征的结构让他们相对高效，他们仍然不能处理这么多这么大且像素高的的图片。幸运的是，如果一个足够优化的2d卷积搭配GPU，就可以训练训练这些有趣的大cnn了，在训练ImageNet这样足够多的标注数据时也不会有严重的过拟合。

The Dataset

1，2段介绍了ImageNet和ILSVRC数据，略。

ImageNet的图像大小不一样，而我们的系统需要输入是一个固定的尺寸。因此我们把他下采样到固定的分辨率256*256上，然后截出固定的尺寸。 就可以了。不会再做其他预处理，所以我们是直接在像素点的RGB值上做训练。

The Architecture

我们的网络结构在图2里。他包括8个可学习的层，其中5个卷积层和3个全连接层。下边主要描述一些这个结构的创新点和特点。3.1 - 3.4章节按重要度排序讲这些。

3.1 ReLU Nonlinearity

标准的激活函数是tanh或者sigmoid。考虑到梯度下降，这些"saturating nonlinearities"比"non-saturating nonlinearity"（比如relu）慢太多了。所以我们参考了Nair和Hinton的，选择用了Rectified Linear Units，即relu。 卷积神经网络比tanh快了几倍，如图1所示。通过这个图能看到，如果用传统的激活函数，可能根本没办法训练这么大的神经网络。

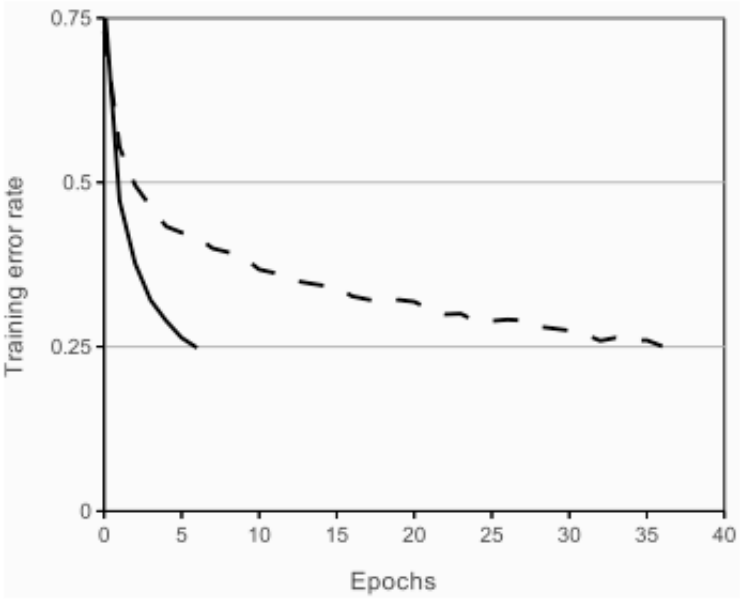

Figure 1: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons **(dashed line)**. The learning rates for each network were chosen independently to make training as fast as possible. No regularization of any kind was employed. The magnitude of the effect demonstrated here varies with network architecture, but networks with ReLUs consistently learn several times faster than equivalents with saturating neurons.

图1

我们不是第一个考虑改变激活函数的，比如Jarrett用过tanh的绝对值做激活函数，在他们那个问题里效果很好。但是在他们主要是为了防止过拟合，所以他们的结论和我们这里说的加速模型训练不一样。在用大模型训练大数据时候，更快的学习对最终表现影响非常大。

3.2 Training on Multiple GPUs

略

3.3 Local Response Normalization

ReLUs有一个很好的属性，就是他们不需要通过把输入归一化来防止saturating。（*这个saturating到底是什么……，查了一下，是饱和，就是sigmoid 数大*

和小时候，梯度消失）只要有至少一个输入是正数，神经元就能学到东西。不过，我们发现局部归一化有助于泛化。a^i_x,y 表示，kernel_i在(x,y)计算的结果，传给relu激活后的值。对应的归一化的b^i_x,y值是：

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

这个式子不太懂....后边也不太想看了。先跳过这一节了

3.4 Overlapping Pooling

传统的pooling层的感受野是非重叠的，我们用了重叠的。 比起s=2,z=2，我们用s=2,z=3，在top-1和top-5中分别降低了0.4和0.3，并且输出的维度是一样的。（*算了一下，在上一层形状是奇数时候，是一样的 n-2/2和 n-3/2，向下取整。*）我们就是观察发现，用重叠的pooling时候，可以轻微的减少过拟合。（*也就是说，别问我为什么，问就是「结果表明」，不够也有道理，确实=3的时候感受野要大，等于相对于=2参照了更大范围的环境，因此不会让模型过度依赖某个区域，虽然减小的也不多就是了*）

3.5 Overall Architecture

现在我们可以来描述这个cnn完整结构了。如图2，他有8个可训练的层，前5个卷积层，剩下的3个是全连接层。输出是一个输出1000, 的softmax，用来描述分类结果，训练目标是最大化多项式逻辑回归，等价于最大化预测分布下训练样本正确标签的对数概率的均值。

第一层卷积层用96个11*11*3，步长为4的kernel，来处理224*224*3的输入图像。然后经过归一化和池化层后，作为输入传到第二层卷积层。第二层有256个5*5*48的kernels。第三，四，五层之间没有pooling和normalization层。第三层有384个3_3_256个kernels（*不用*了，总是被自动加成斜体，真烦。。。*）的输入是2层输出经过归一化和池化后的。第4层是384个3_3_192 kernels，第五层是256个3_3_192个。后边的全连接层都是各有4096个神经元。

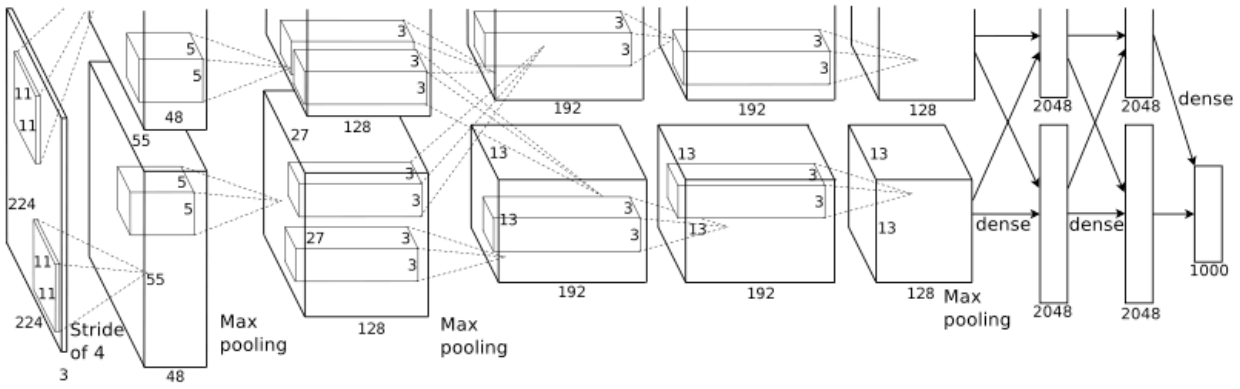

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

图2

（我不知道224 → 55 怎么算的，不是224-11+1/4吗向上取整吗，就是54啊.....还有96个kernels，为啥要分成2部分上边一块下边一块..... ）

4. Reducing Overfitting

降低过拟合最简单和常用的方法就是增加数据的数量。我们用两种不同的增加数据的方式，都可以用很少的计算得到新的图片，所以生成的图片不需要存在硬盘上。在我们的实现中，生成图片的代码是python写的， 跑在cpu上，同时gpu在训练上一个batch的图。所以实际上，生成图片在时间上是"免费"的。

第一个生成图片的方式是通过图像变换和水平翻转（*变换是什么.....*）我们通过从256*256中随机提取一些224*224，以及他们的水平翻转，然后在这些图片上训练。这增大了训练集2048倍（*256-224的平方再乘以2，翻转*），尽管这些结果和原图非常像。 没有这个模式的话，我们的网络会有较大的过拟合，那我们就只能用比较小的网络了。然后在测试时候，网络会从图片中提取5个224乘224的部分，4个从角落，1个从中间，对他们做预测，然后取平均值作为最终的预测结果（*我记得好像这个叫10crop吧*）

第二种方法是通过改rgb通道的值，*balbala，不感兴趣，略.....*

4.2 Dropout

我们在图2中的前两个全连接层使用失活。如果没有失活，我们的网络表现出大量的过拟合。失活大致上使要求收敛的迭代次数翻了一倍。

5. Details of learning

1. 少量的权重衰减对于模型的学习很重要，权重衰减的作用不只是一个正则化，他减少了模型的训练误差。（*这里是momentum吗，这个学的时候没注意，明天去补一下*）
2. 手动调整学习率。学习率初始化位0.01，后来发现validation_error不变的时候，就把学习率除以10。在训练结束前，我们调了3次。120万图像上训练了大概90个epochs，在NVIDIA GTX 5890 3GB GPU上花了5-6天。

6. Results

好。

7. Discussion

深度很重要，去掉任何中间层，都会损失大约2% top-1性能。