卷积神经网络框架

LeNet

背景

LeNet5 诞生于 1994 年，是最早的卷积神经网络之一，并且推动了深度学习领域的发展。自从 1988 年开始，在多年的研究和许多次成功的迭代后，这项由 Yann LeCun 完成的开拓性成果被命名为 LeNet5。

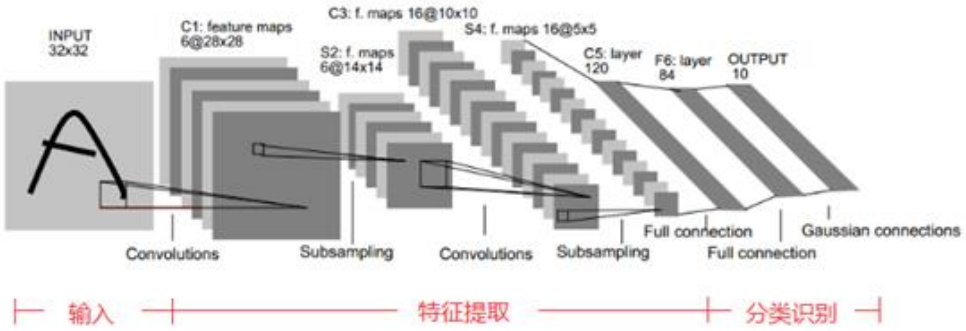
1989 年，Yann LeCun 等人在贝尔实验室的研究首次将反向传播算法进行了实际应用，并且认为学习网络泛化的能力可以通过提供来自任务域的约束来大大增强。他将使用反向传播算法训练的卷积神经网络结合到读取“手写”数字上，并成功应用于识别美国邮政服务提供的手写邮政编码数字。这即是后来被称为 LeNet 的卷积神经网络的雏形。同年，Yann LeCun 在发表的另一篇论文中描述了一个小的手写数字识别问题，并且表明即使该问题是线性可分的，单层网络也表现出较差的泛化能力。而当在多层的、有约束的网络上使用有位移不变性的特征检测器（shift invariant feature detectors）时，该模型可以在此任务上表现得非常好。他认为这些结果证明了将神经网络中的自由参数数量最小化可以增强神经网络的泛化能力。

1990 年他们发表的论文再次描述了反向传播网络在手写数字识别中的应用，他们仅对数据进行了最小限度的预处理，而模型则是针对这项任务精心设计的，并且对其进行了高度约束。输入数据由图像组成，每张图像上包含一个数字，在美国邮政服务提供的邮政编码数字数据上的测试结果显示该模型的错误率仅有 1%，拒绝率约为 9%。

其后 8 年他们的研究一直继续，直到 1998 年，Yann LeCun，Leon Bottou，Yoshua Bengio 和 Patrick Haffner 在发表的论文中回顾了应用于手写字符识别的各种方法，并用标准手写数字识别基准任务对这些模型进行了比较，结果显示卷积神经网络的表现超过了其他所有模型。他们同时还提供了许多神经网络实际应用的例子，如两种用于在线识别手写字符的系统和能每天读取数百万张支票的模型。

他们的研究取得了巨大的成功，并且激起了大量学者对神经网络的研究的兴趣。在今天向过去回首，目前性能最好的神经网络的架构已与 LeNet 不尽相同，但这个网络是大量神经网络架构的起点，并且也给这个领域带来了许多灵感。

LeNet 框架



LeNet5 由 7 层 CNN（不包含输入层）组成，上图中输入的原始图像大小是 32×32 像素，卷积层用 Ci 表示，子采样层（pooling，池化）用 Si 表示，全连接层用 Fi 表示。下面逐层介绍其作用和示意图上方的数字含义。

1、C1 层（卷积层）：6@28×28

该层使用了 6 个卷积核，每个卷积核的大小为 5×5，这样就得到了 6 个 feature map（特征图）。

（1）特征图大小

每个卷积核（5×5）与原始的输入图像（32×32）进行卷积，这样得到的 feature map（特征图）大小为（32-5+1）×（32-5+1）= 28×28

卷积核与输入图像按卷积核大小逐个区域进行匹配计算，匹配后原始输入图像的尺寸将变小，因为边缘部分卷积核无法越出界，只能匹配一次，匹配计算后的尺寸变为 Cr×Cc=（Ir-Kr+1）×（Ic-Kc+1），其中 Cr、Cc，Ir、Ic，Kr、Kc 分别表示卷积后结果图像、输入图像、卷积核的行列大小。

（2）参数个数

由于参数（权值）共享的原因，对于同个卷积核每个神经元均使用相同的参数，因此，参数个数为（5×5+1）×6= 156，其中 5×5 为卷积核参数，1 为偏置参数

（3）连接数

卷积后的图像大小为 28×28，因此每个特征图有 28×28 个神经元，每个卷积核参数为（5×5+1）×6，因此，该层的连接数为（5×5+1）×6×28×28=122304

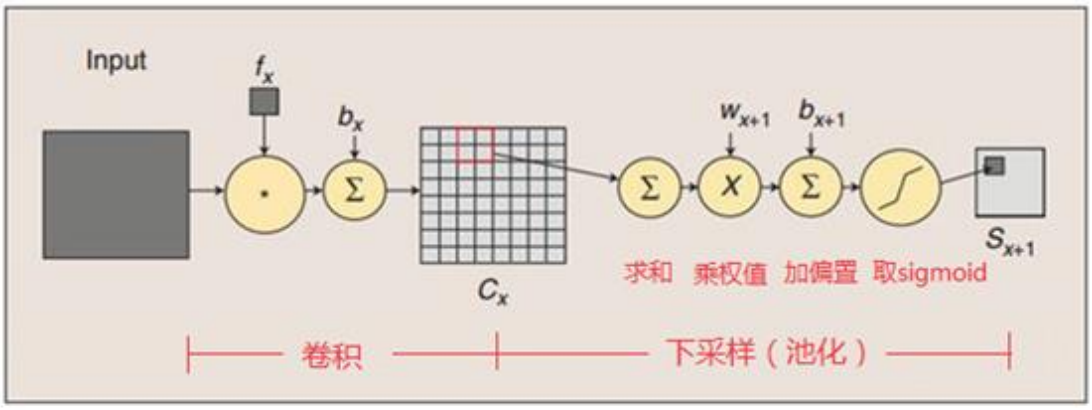
2、S2 层（下采样层，也称池化层）：6@14×14

（1）特征图大小

这一层主要是做池化或者特征映射（特征降维），池化单元为 2×2，因此，6 个特征图的大小经池化后即变为 14×14。回顾本文刚开始讲到的池化操作，池化单元之间没有重叠，在池化区域内进行聚合统计后得到新的特征值，因此经 2×2 池化后，每两行两列重新算出一个特征值出来，相当于图像大小减半，因此卷积后的 28×28 图像经 2×2 池化后就变为 14×14。

这一层的计算过程是：2×2 单元里的值相加，然后再乘以训练参数 w，再加上一个偏置参

数 b（每一个特征图共享相同的 w 和 b），然后取 sigmoid 值（S 函数：0-1 区间），作为对应的该单元的值。卷积操作与池化的示意图如下：



（2）参数个数

S2 层由于每个特征图都共享相同的 w 和 b 这两个参数，因此需要 2×6=12 个参数

（3）连接数

下采样之后的图像大小为 14×14，因此 S2 层的每个特征图有 14×14 个神经元，每个池化单元连接数为 2×2+1（1 为偏置量），因此，该层的连接数为（2×2+1）×14×14×6 = 5880

3、C3 层（卷积层）：16@10×10

C3 层有 16 个卷积核，卷积模板大小为 5×5。

（1）特征图大小

与 C1 层的分析类似，C3 层的特征图大小为（14-5+1）×（14-5+1）= 10×10

（2）参数个数

需要注意的是，C3 与 S2 并不是全连接而是部分连接，有些是 C3 连接到 S2 三层、有些四层、甚至达到 6 层，通过这种方式提取更多特征，连接的规则如下表所示：

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

3　　4　　6

例如第一列表示C3层的第0个特征图（feature map）只跟S2层的第0、1和2这三个feature maps相连接，计算过程为：用3个卷积模板分别与S2层的3个feature maps进行卷积，然后将卷积的结果相加求和，再加上一个偏置，再取sigmoid得出卷积后对应的feature map了。其它列也是类似（有些是3个卷积模板，有些是4个，有些是6个）。因此，C3层的参数数目为（5×5×3+1）×6 +（5×5×4+1）×9 +5×5×6+1 = 1516

（3）连接数

卷积后的特征图大小为10×10，参数数量为1516，因此连接数为1516×10×10= 151600

4、S4（下采样层，也称池化层）：16@5×5

（1）特征图大小

与S2的分析类似，池化单元大小为2×2，因此，该层与C3一样共有16个特征图，每个特征图的大小为5×5。

（2）参数数量

与S2的计算类似，所需要参数个数为16×2 = 32

（3）连接数

连接数为（2×2+1）×5×5×16 = 2000

5、C5层（卷积层）：120

（1）特征图大小

该层有120个卷积核，每个卷积核的大小仍为5×5，因此有120个特征图。由于S4层的大小为5×5，而该层的卷积核大小也是5×5，因此特征图大小为（5-5+1）×（5-5+1）= 1×1。这样该层就刚好变成了全连接，这只是巧合，如果原始输入的图像比较大，则该层就不是全连接了。

（2）参数个数

与前面的分析类似，本层的参数数目为120×（5×5×16+1） = 48120

（3）连接数

由于该层的特征图大小刚好为1×1，因此连接数为48120×1×1=48120

6、F6层（全连接层）：84

（1）特征图大小

F6层有84个单元，之所以选这个数字的原因是来自于输出层的设计，对应于一个7×12的比特图，如下图所示，-1表示白色，1表示黑色，这样每个符号的比特图的黑白色就对应于一个编码。

该层有 84 个特征图，特征图大小与 C5 一样都是 1×1，与 C5 层全连接。

（2）参数个数

由于是全连接，参数数量为（120+1）×84=10164。跟经典神经网络一样，F6 层计算输入向量和权重向量之间的点积，再加上一个偏置，然后将其传递给 sigmoid 函数得出结果。

（3）连接数

由于是全连接，连接数与参数数量一样，也是 10164。

7、OUTPUT 层（输出层）：10

Output 层也是全连接层，共有 10 个节点，分别代表数字 0 到 9。如果第 i 个节点的值为 0，则表示网络识别的结果是数字 i。

（1）特征图大小

该层采用径向基函数（RBF）的网络连接方式，假设 x 是上一层的输入，y 是 RBF 的输出，则 RBF 输出的计算方式是：

$$y_i = \sum_j (x_j - w_{ij})^2$$

上式中的 Wij 的值由 i 的比特图编码确定，i 从 0 到 9，j 取值从 0 到 7×12-1。RBF 输出的值越接近于 0，表示当前网络输入的识别结果与字符 i 越接近。

（2）参数个数

由于是全连接，参数个数为 84×10=840

（3）连接数

由于是全连接，连接数与参数个数一样，也是 840

AlexNet

背景

在计算机视觉领域 object detection & recognition 通常用机器学习的方法来解决。为了提高识别的效果，我们可以通过收集更多的可训练的数据来让模型的泛化性能提高。目前，在以一万为单位的数量级层面的数据(称为简单的识别任务)已经获得了非常好的性能，例如：MNIST 手写数字识别任务，最好的性能已经达到了<0.3%的误差。但是现实中的物体存在相当多的变化属性，所以学习识别它们需要更多的数据。事实上，小的图像训练数据有很多的缺点，无论以我们的直觉想还是理论证明都是有依据的，理论上论文《Why is real-world visual object recognition hard?》给出了研究方法。随着互联网技术的发展，以及智能手机的普及图像数据获取可以说越来越容易。所以就有组织去收集这些现实中事物的图像并进行标记和分割。例如：LabelMe(Labelme: a database and web-based tool for image annotation.)，包含了成百上千的全分割图像。 ImageNet(ImageNet: A Large-Scale Hierarchical Image Database. I),包含 15 billion 标记的高分辨率图像，包含超过了 22000 种现实中的事物。

文章中说该模型有 5 层卷积，去掉任意一层都会使结果不好，所以这个网络的深度似乎是很重要的，这样的话难免引起我们的思考，记得不知道哪位大神在一篇论文中证明了，神经网络可以模拟任意多项式，只要神经元数量足够多，并且和深度关系不大。但这里的实验却表示深度会对网络的性能有影响。

文章中还提到了，他们用 5-6 天训练了这个模型，并且限制了网络的大小，因为现有的硬件智能允许那么大的内存，用更好的设备还可以获得更好的效果。

AlexNet 框架

AlexNet 的网络结构是这样的，为啥我感觉这样表示的网络很丑呀，哈哈。

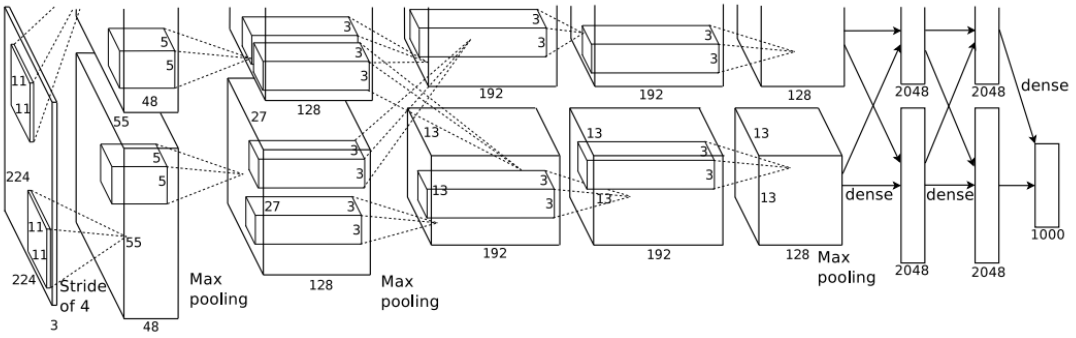


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input 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.

感觉这个网络很复杂呀，需要怎么理解好呢？首先这幅图分为上下两个部分的网络，论文中提到这两部分网络是分别对应两个 GPU，只有到了特定的网络层后才需要两块 GPU 进行交互，这种设置完全是利用两块 GPU 来提高运算的效率，其实在网络结构上差异不是很大。为了更方便的理解，我们假设现在只有一块 GPU 或者我们用 CPU 进行运算，我们从这个稍微简化点的方向区分析这个网络结构。网络总共的层数为 8 层，5 层卷积，3 层全连接层。

第一层：卷积层 1，输入为 224×224×3

224×224×3 的图像，卷积核的数量为 96，论文中两片 GPU 分别计算 48 个核；卷积核的大小为 11×11×3

11×11×3; stride = 4, stride 表示的是步长， pad = 0, 表示不扩充边缘;
卷积后的图形大小是怎样的呢？
wide = (224 + 2 * padding - kernel_size) / stride + 1 = 54
height = (224 + 2 * padding - kernel_size) / stride + 1 = 54
dimention = 96
然后进行 (Local Response Normalized), 后面跟着池化 pool_size = (3, 3), stride = 2, pad = 0 最终获得第一层卷积的 feature map
最终第一层卷积的输出为

第二层：卷积层 2, 输入为上一层卷积的 feature map， 卷积的个数为 256 个，论文中的两个 GPU 分别有 128 个卷积核。卷积核的大小为：5×5×48

5×5×48; pad = 2, stride = 1; 然后做 LRN， 最后 max_pooling, pool_size = (3, 3), stride = 2;

第三层：卷积 3，输入为第二层的输出，卷积核个数为 384，kernel_size = (3×3×256

3×3×256)， padding = 1，第三层没有做 LRN 和 Pool

第四层：卷积 4，输入为第三层的输出，卷积核个数为 384，kernel_size = (3×3

3×3)，padding = 1，和第三层一样，没有 LRN 和 Pool

第五层：卷积 5，输入为第四层的输出，卷积核个数为 256，kernel_size = (3×3

3×3)，padding = 1。然后直接进行 max_pooling，pool_size = (3，3)，stride = 2;

第 6,7,8 层是全连接层，每一层的神经元的个数为 4096，最终输出 softmax 为 1000,因为上面介绍过，ImageNet 这个比赛的分类个数为 1000。全连接层中使用了 RELU 和 Dropout 上面的结构是假设在一块 GPU 上面的，和论文的两块 GPU 有差别，但是为了方便理解，还是采用越简单的结构越好。

VGG

VGG 背景

VGGNet 由牛津大学计算机视觉组合和 Google DeepMind 公司研究员一起研发的深度卷积神经网络。它探索了卷积神经网络的深度和其性能之间的关系，通过反复的堆叠 3*3 的小型卷积核和 2*2 的最大池化层，成功的构建了 16~19 层深的卷积神经网络。VGGNet 获得了 ILSVRC 2014 年比赛的亚军和定位项目的冠军，在 top5 上的错误率为 7.5%。目前为止，VGGNet 依然被用来提取图像的特征。

创新点

VGGNet 全部使用 3*3 的卷积核和 2*2 的池化核，通过不断加深网络结构来提升性能。网络层数的增长并不会带来参数量上的爆炸，因为参数量主要集中在最后三个全连接层中。同时，两个 3*3 卷积层的串联相当于 1 个 5*5 的卷积层，3 个 3*3 的卷积层串联相当于 1 个 7*7 的卷积层，即 3 个 3*3 卷积层的感受野大小相当于 1 个 7*7 的卷积层。但是 3 个 3*3 的卷积层参数量只有 7*7 的一半左右，同时前者可以有 3 个非线性操作，而后者只有 1 个非线性操作，这样使得前者对于特征的学习能力更强。

使用 1*1 的卷积层来增加线性变换，输出的通道数量上并没有发生改变。这里提一下 1*1 卷积层的其他用法，1*1 的卷积层常被用来提炼特征，即多通道的特征组合在一起，凝练成较大通道或者较小通道的输出，而每张图片的大小不变。有时 1*1 的卷积神经网络还可以用来替代全连接层。

其他小技巧。VGGNet 在训练的时候先训级别 A 的简单网络，再复用 A 网络的权重来初始化后面的几个复杂模型，这样收敛速度更快。VGGNet 作者总结出 LRN 层作用不大，越深的网络效果越好，1*1 的卷积也是很有效的，但是没有 3*3 的卷积效果好，因为 3*3 的网络可以学习到更大的空间特征。

网络结构

VGGNet 的网络结构如下图所示。VGGNet 包含很多级别的网络，深度从 11 层到 19 层不等，比较常用的是 VGGNet-16 和 VGGNet-19。VGGNet 把网络分成了 5 段，每段都把多个 3*3 的卷积网络串联在一起，每段卷积后面接一个最大池化层，最后面是 3 个全连接层和一个 softmax 层。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

图 6-6 VGGNet 各级别网络结构图

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

图 6-7 VGGNet 各级别网络参数量（单位为百万）

GoogLeNet

GoogLeNet 发展历程

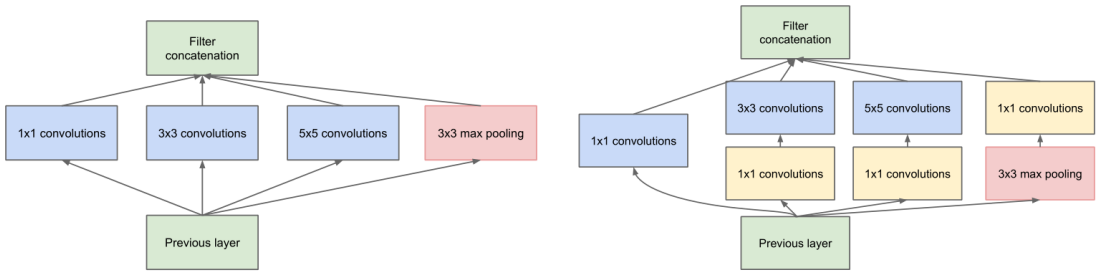
1. Inception v1 的网络，打破了常规的卷积层串联的模式，将 1x1，3x3，5x5 的卷积层和 3x3 的 pooling 池化层并联组合后 concatenate 组装在一起的设计思路；
2. Inception v2 的网络在 Inception v1 的基础上，进行了改进，一方面了加入了 BN 层，减少了 Internal Covariate Shift（内部神经元分布的改变），使每一层的输出都规范化到一个 N(0, 1)的高斯，还去除了 Dropout、LRN 等结构；另外一方面学习 VGG 用 2 个 3x3 的

卷积替代 inception 模块中的 5x5 卷积，既降低了参数数量，又加速计算；
3. Inception v3 一个最重要的改进是分解（Factorization），将 7x7 分解成两个一维的卷积（1x7,7x1），3x3 也是一样（1x3,3x1）。这样的好处，既可以加速计算（多余的计算能力可以用来加深网络），又可以将 1 个 conv 拆成 2 个 conv，使得网络深度进一步增加，增加了网络的非线性，可以处理更多更丰富的空间特征，增加特征多样性。还有值得注意的地方是网络输入从 224x224 变为了 299x299，更加精细设计了 35x35/17x17/8x8 的模块；
4. Inception v4 结合了微软的 ResNet，发现 ResNet 的结构可以极大地加速训练，同时性能也有提升，得到一个 Inception-ResNet v2 网络，同时还设计了一个更深更优化的 Inception v4 模型，能达到与 Inception-ResNet v2 相媲美的性能。

重点介绍 Inception V1

1. 概述

Google Inception Net 首次出现在 ILSVRC 2014 的比赛中，以较大优势取得了第一名。那届比赛中的 Inception Net 通常被称为 Inception V1，它最大的特点是控制了计算量和参数量的同时，获得了非常好的分类性能——top-5 错误率 6.67%，只有 AlexNet 的一半不到。Inception V1 有 22 层深，比 AlexNet 的 8 层或者 VGGNet 的 19 层还要更深。但其计算量只有 15 亿次浮点运算，同时只有 500 万的参数量，仅为 AlexNet 参数量（6000 万）的 1/12，却可以达到远胜于 AlexNet 的准确率，可以说是非常优秀并且非常实用的模型。Inception V1 降低参数量的目的有两点，第一，参数越多模型越庞大，需要供模型学习的数据量就越大，而目前高质量的数据非常昂贵；第二，参数越多，耗费的计算资源也会更大。Inception V1 参数少但效果好的原因除了模型层数更深、表达能力更强外，还有两点：一是去除了最后的全连接层，用全局平均池化层（即将图片尺寸变为 1*1）来取代它。全连接层几乎占据了 AlexNet 或 VGGNet 中 90%的参数量，而且会引起过拟合，去除全连接层后模型训练更快并且减轻了过拟合。用全局平均池化层取代全连接层的做法借鉴了 NetworkIn Network（以下简称 NIN）论文。二是 Inception V1 中精心设计的 InceptionModule 提高了参数的利用效率，其结构如图 1 所示。这一部分也借鉴了 NIN 的思想，形象的解释就是 Inception Module 本身如同大网络中的一个小网络，其结构可以反复堆叠在一起形成大网络。不过 Inception V1 比 NIN 更进一步的是增加了分支网络，NIN 则主要是级联的卷积层和 MLPConv 层。一般来说卷积层要提升表达能力，主要依靠增加输出通道数，但副作用是计算量增大和过拟合。每一个输出通道对应一个滤波器，同一个滤波器共享参数，只能提取一类特征，因此一个输出通道只能做一种特征处理。而 NIN 中的 MLPConv 则拥有更强大的能力，允许在输出通道之间组合信息，因此效果明显。可以说，MLPConv 基本等效于普通卷积层后再连接 1*1 的卷积和 ReLU 激活函数。



(a) Inception module, naïve version　　(b) Inception module with dimension reductions

图 1 Inception Module

2. InceptionModule

Inception Module 的基本结构如图 1，有 4 个分支：第一个分支对输入进行 1*1 的卷积，这其实也是 NIN 中提出的一个重要结构。1*1 的卷积是一个非常优秀的结构，它可以跨通道组织信息，提高网络的表达能力，同时可以对输出通道升维和降维。可以看到 Inception Module 的 4 个分支都用到了 1*1 卷积，来进行低成本（计算量比 3*3 小很多）的跨通道的特征变换。第二个分支先使用了 1*1 卷积，然后连接 3*3 卷积，相当于进行了两次特征变换。第三个分支类似，先是 1*1 的卷积，然后连接 5*5 卷积。最后一个分支则是 3*3 最大池化后直接使用 1*1 卷积。有的分支只使用 1*1 卷积，有的分支使用了其他尺寸的卷积时也会再使用 1*1 卷积，这是因为 1*1 卷积的性价比很高，用很小的计算量就能增加一层特征变换和非线性化。Inception Module 的 4 个分支在最后通过一个聚合操作合并（在输出通道数这个维度上聚合）。Inception Module 中包含了 3 种不同尺寸的卷积和 1 个最大池化，增加了网络对不同尺度的适应性，这一部分和 Multi-Scale 的思想类似。早期计算机视觉的研究中，受灵长类神经视觉系统的启发，Serre 使用不同尺寸的 Gabor 滤波器处理不同尺寸的图片，Inception V1 借鉴了这种思想。Inception V1 的论文中指出，InceptionModule 可以让网络的深度和宽度高效率地扩充，提升准确率且不致于过拟合。

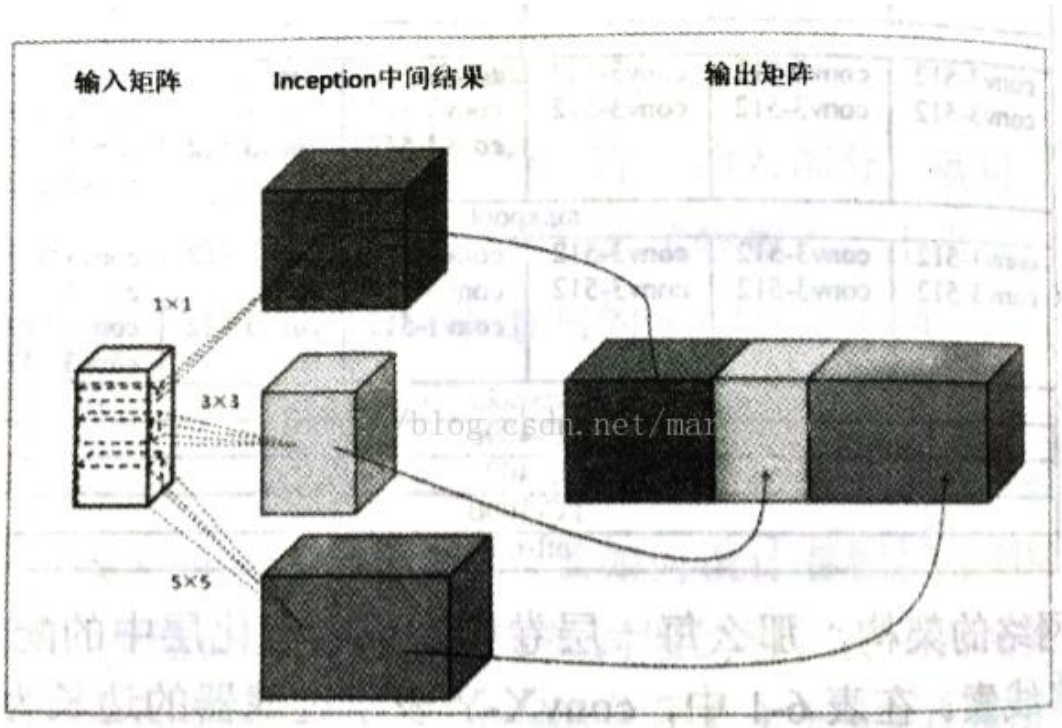


图 6-16 Inception 模块示意图。

稀疏结构是非常适合神经网络的一种结构，尤其是对非常大型、非常深的神经网络，可以减轻过拟合并降低计算量，例如卷积神经网络就是稀疏的连接。Inception Net 的主要目标就是找到最优的稀疏结构单元 Inception Module，论文中提到其稀疏结构基于 Hebbian 原理，这里简单解释一下 Hebbian 原理：神经反射活动的持续与重复会导致神经元连接稳定性的持久提升，当两个神经元细胞 A 和 B 距离很近，并且 A 参与了对 B 重复、持续的兴奋，那么某些代谢变化会导致 A 将作为能使 B 兴奋的细胞。总结一下即“一起发射的神经元会连在一起”（Cells that fire together, wire together），学习过程中的刺激会使神经元间的突触强度增加。受 Hebbian 原理启发，另一篇文章 Provable Bounds for Learning Some Deep Representations 提出，如果数据集的概率分布可以被一个很大很稀疏的神经网络所表达，

那么构筑这个网络的最佳方法是逐层构筑网络：将上一层高度相关的节点聚类，并将聚类出来的每一个小簇（cluster）连接到一起，如图 2 所示。这个相关性高的节点应该被连接在一起的结论，即是从神经网络的角度对 Hebbian 原理有效性的证明。

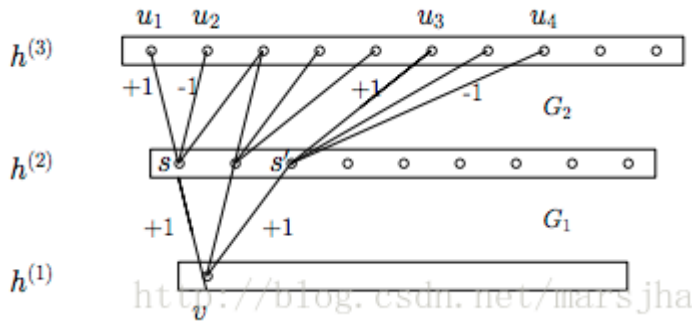


图 2 将高度相关的节点连接在一起，形成稀疏网络

因此一个“好”的稀疏结构，应该是符合 Hebbian 原理的，我们应该把相关性高的一簇神经元节点连接在一起。在图片数据中，天然的就是临近区域的数据相关性高，因此相邻的像素点被卷积操作连接在一起。而我们可能有多个卷积核，在同一空间位置但在不同通道的卷积核的输出结果相关性极高。因此，一个 1*1 的卷积就可以很自然地把这些相关性很高的、在同一个空间位置但是不同通道的特征连接在一起，这就是为什么 1*1 卷积这么频繁地被应用到 Inception Net 中的原因。1*1 卷积所连接的节点的相关性是最高的，而稍微大一点尺寸的卷积，比如 3*3、5*5 的卷积所连接的节点相关性也很高，因此也可以适当地使用一些大尺寸的卷积，增加多样性（diversity）。Inception Module 通过 4 个分支中不同尺寸的 1*1、3*3、5*5 等小型卷积将相关性很高的节点连接在一起，就完成了其设计初衷，构建出了很高效的符合 Hebbian 原理的稀疏结构。

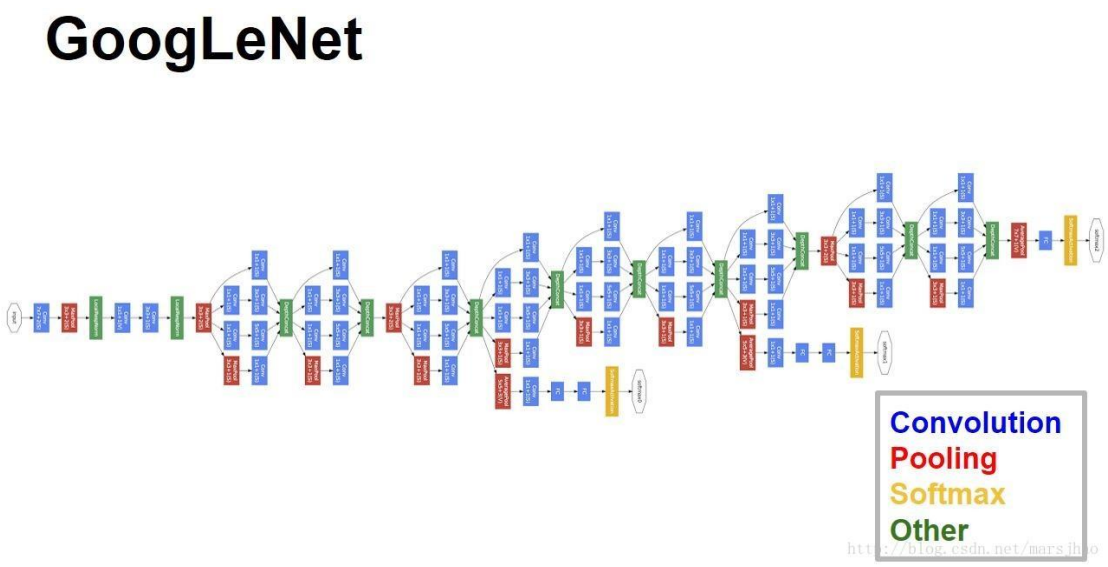
3. InceptionNet 网络结构

在 Inception Module 中，通常 1*1 卷积的比例（输出通道数占比）最高，3*3 卷积和 5*5 卷积稍低。而在整个网络中，会有多个堆叠的 Inception Module，我们希望靠后的 InceptionModule 可以捕捉更高阶的抽象特征，因此靠后的 Inception Module 的卷积的空间集中度应该逐渐降低，这样可以捕获更大面积的特征。因此，越靠后的 InceptionModule 中，3*3 和 5*5 这两个大面积的卷积核的占比（输出通道数）应该更多。

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

nception Net 有 22 层深，除了最后一层的输出，其中间节点的分类效果也很好。因此在 Inception Net 中，还使用到了辅助分类节点（auxiliary classifiers），即将中间某一层的输出用作分类，并按一个较小的权重（0.3）加到最终分类结果中。这样相当于做了模型融合，同时给网络增加了反向传播的梯度信号，也提供了额外的正则化，对于整个 Inception Net 的训练很有裨益。Inception V1 也使用了 Multi-Scale、Multi-Crop 等数据增强方法，并在不同的采样数据上训练了 7 个模型进行融合，得到了最后的 ILSVRC 2014 的比赛成绩——top-5 错误率 6.67%。



对上图做如下说明：

1. 显然 GoogLeNet 采用了模块化的结构，方便增添和修改；
2. 网络最后采用了 average pooling 来代替全连接层，想法来自 NIN，事实证明可以将 TOP1 accuracy 提高 0.6%。但是，实际在最后还是加了一个全连接层，主要是为了方便以后大家 finetune；

3. 虽然移除了全连接，但是网络中依然使用了 Dropout ；
4. 为了避免梯度消失，网络额外增加了 2 个辅助的 softmax 用于向前传导梯度。文章中说这两个辅助的分类器的 loss 应该加一个衰减系数，但看 caffe 中的 model 也没有加任何衰减。此外，实际测试的时候，这两个额外的 softmax 会被去掉。