1.思想

众所周知，最近一两年卷积神经网络提高效果的方向，要么深（比如ResNet，解决了网络深时候的梯度消失问题）要么宽（比如GoogleNet的Inception），而作者则是从feature入手，通过对feature的极致利用达到更好的效果和更少的参数。

先列下DenseNet的几个优点，感受下它的强大：

1、减轻了vanishing-gradient（梯度消失）

2、加强了feature的传递

3、更有效地利用了feature

4、一定程度上较少了参数数量

2.网络结构

先放一个dense block的结构图。在传统的卷积神经网络中，如果你有L层，那么就会有L个连接，但是在DenseNet中，会有L(L+1)/2个　　连接。简单讲，就是每一层的输入来自前面所有层的输出。如下图：x0是input，H1的输入是x0（input），H2的输入是x0和x1（x1是H1的输出）

.....

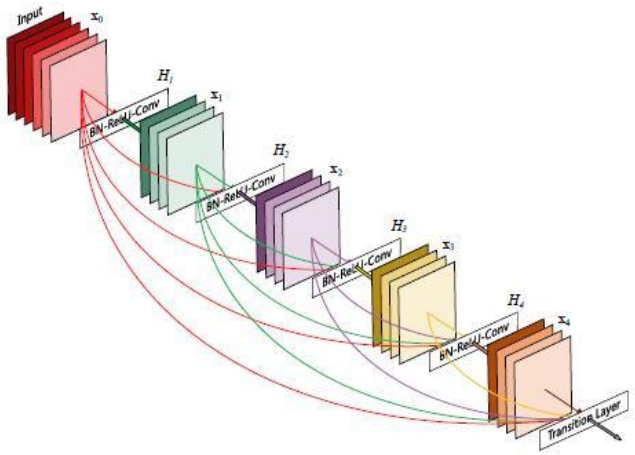

Figure 1. A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

DenseNet的一个优点是网络更窄，参数更少，很大一部分原因得益于这种dense block的设计，后面有提到在dense block中每个卷积层的输出feature map的数量都很小（小于100），而不是像其他网络一样动不动就几百上千的宽度。同时这种连接方式使得特征和梯度的传递更加有效，网络也就更加容易训练。原文的一句话非常喜欢：Each layer has direct access to the gradients from the loss function and the original input signal, leading to an

implicit deep supervision.直接解释了为什么这个网络的效果会很好。前面提到过梯度消失问题在网络深度越深的时候越容易出现，原因就是输入信息和梯度信息在很多层之间传递导致的，而现在这种dense connection相当于每一层都直接连接input和loss，因此就可以减轻梯度消失现象，这样更深网络不是问题。另外作者还观察到这种dense connection有正则化的效果，因此对于过拟合有一定的抑制作用，博主认为是因为参数减少了（后面会介绍为什么参数会减少），所以过拟合现象减轻。

第一个公式是ResNet的。这里的l表示层，xl表示l层的输出，Hl表示一个非线性变换。所以对于ResNet而言，l层的输出是l-1层的输出加上对l-1层输出的非线性变换。

$$\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}.$$

第二个公式是DenseNet的。[x0,x1,...,xl-1]表示将0到l-1层的输出feature map做concatenation。concatenation是做通道的合并，就像Inception那样。而前面resnet是做值的相加，通道数是不变的。Hl包括BN，ReLU和3*3的卷积。

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]),$$

所以从这两个公式就能看出DenseNet和ResNet在本质上的区别，太精辟。

前面的Figure 1表示的是dense block，而下面的Figure 2表示的则是一个DenseNet的结构图，在这个结构图中包含了3个dense block。作者将DenseNet分成多个dense block，原因是希望各个dense block内的feature map的size统一，这样在做concatenation就不会有size的问题。

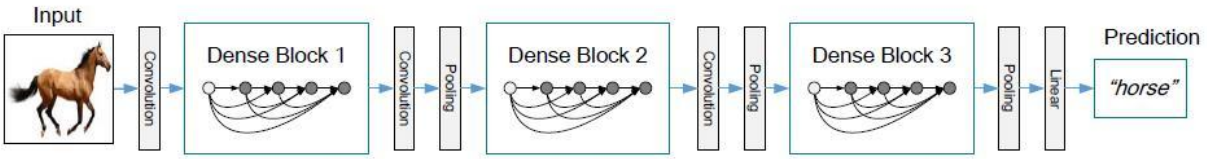

Figure 2. A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature map sizes via convolution and pooling.

Layers	Output Size	DenseNet-121($k=32$)	DenseNet-169($k=32$)	DenseNet-201($k=32$)	DenseNet-161($k=48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 36
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 24
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Table 1. DenseNet architectures for ImageNet. The growth rate for the first 3 networks is $k = 32$, and $k = 48$ for DenseNet-161. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

这个Table1就是整个网络的结构图。这个表中的k=32，k=48中的k是growth rate，表示每个dense block中每层输出的feature map个数。为了避免网络变得很宽，作者都是采用较小的k，比如32这样，作者的实验也表明小的k可以有更好的效果。根据dense block的设计，后面几层可以得到前面所有层的输入，因此concat后的输入channel还是比较大的。**另外这里每个dense block的3*3卷积前面都包含了一个1*1的卷积操作，就是所谓的bottleneck layer，目的是减少输入的feature map数量，既能降维减少计算量，又能融合各个通道的特征，**何乐而不为。**另外作者为了进一步压缩参数，在每两个dense block之间又增加了1*1的卷积操作。**因此在后面的实验对比中，如果你看到DenseNet-C这个网络，表示增加了这个Translation layer，该层的1*1卷积的输出channel默认是输入channel到一半。如果你看到DenseNet-BC这个网络，表示既有bottleneck layer，又有Translation layer。

再详细说下bottleneck和transition layer操作。在每个Dense Block中都包含很多个子结构，以DenseNet-169的Dense Block（3）为例，包含32个1*1和3*3的卷积操作，也就是第32个子结构的输入是前面31层的输出结果，每层输出的channel是32（growth rate），**那么如果不做bottleneck操作，第32层的3*3卷积操作的输入就是31*32+（上一个Dense Block的输出channel），近1000了。**而加上1*1的卷积，代码中的1*1卷积的channel是growth rate*4，也就是128，然后再作为3*3卷积的输入。这就大大减少了计算量，这就是bottleneck。**至于transition layer，放在两个Dense Block中间，是因为每个Dense Block结束后的输出channel个数很多，需要用1*1的卷积核来降维。**还是以DenseNet-169的Dense Block（3）为例，虽然第32层的3*3卷积输出channel只有32个（growth rate），但是紧接着还会像前面几层一样有通道的concat操作，即将第32层的输出和第32层的输入做concat，前面说过第32层的输入是1000左右的channel，所以最后每个Dense Block的输出也是1000多的channel。**因此这个transition layer有个参数reduction（范围是0到1），表示将这些输出缩小到原来的多少倍，默认是0.5，**这样传给下一个Dense Block的时候channel数量就会减少一半，这就是transition layer的作用。文中还用到dropout操作来随机减少分支，避免过拟合，毕竟这篇文章的连接确实多。

3.总结及原作者的解释

文章提出的DenseNet核心思想在于建立了不同层之间的连接关系，充分利用了feature，进一步减轻了梯度消失问题，加深网络不是问题，而且训练效果非常好。另外，利用bottleneck layer，Translation layer以及较小的growth rate使得网络变窄，参数减少，有效抑制了过拟合，同时计算量也减少了。DenseNet优点很多，而且在和ResNet的对比中优势还是非常明显的。

DenseNet 是受什么启发提出来的？

DenseNet 的想法很大程度上源于我们去年发表在 ECCV 上的一个叫做随机深度网络（Deep networks with stochastic depth）工作。当时我们提出了一种类似于 Dropout 的方法来改进ResNet。我们发现在训练过程中的每一步都随机地「扔掉」（drop）一些层，可以显著的提高 ResNet 的泛化性能。这个方法的成功至少带给我们两点启发：

首先，它说明了神经网络其实并不一定要是一个递进层级结构，也就是说网络中的某一层可以不仅仅依赖于紧邻的上一层的特征，而可以依赖于更前面层学习的特征。想像一下在随机深度网络中，当第 l 层被扔掉之后，第 l+1 层就被直接连到了第 l-1 层；当第 2 到了第 l 层都被扔掉之后，第 l+1 层就直接用到了第 1 层的特征。因此，随机深度网络其实可以看成一个具有随机密集连接的 DenseNet。

其次，我们在训练的过程中随机扔掉很多层也不会破坏算法的收敛，说明了 ResNet 具有比较明显的冗余性，网络中的每一层都只提取了很少的特征（即所谓的残差）。实际上，我们将训练好的 ResNet 随机的去掉几层，对网络的预测结果也不会产生太大的影响。既然每一层学习的特征这么少，能不能降低它的计算量来减小冗余呢？

DenseNet 的设计正是基于以上两点观察。**我们让网络中的每一层都直接与其前面层相连，实现特征的重复利用；同时把网络的每一层设计得特别「窄」，即只学习非常少的特征图（最极端情况就是每一层只学习一个特征图），达到降低冗余性的目的。**这两点也是 DenseNet 与其他网络最主要的不同。需要强调的是，第一点是第二点的前提，没有密集连接，我们是不可能把网络设计得太窄的，否则训练会出现欠拟合（under-fitting）现象，即使 ResNet 也是如此。

DenseNet 有什么优点？

省参数。在 ImageNet 分类数据集上达到同样的准确率，DenseNet 所需的参数量不到 ResNet 的一半。对于工业界而言，小模型可以显著地节省带宽，降低存储开销。

省计算。达到与 ResNet 相当的精度，DenseNet 所需的计算量也只有 ResNet 的一半左右。计算效率在深度学习实际应用中的需求非常强烈，从本次 CVPR 会上大家对模型压缩以及 MobileNet 和 ShuffleNet 这些工作的关注就可以看得出来。最近我们也在搭建更高效的 DenseNet，初步结果表明 DenseNet 对于这类应用具有非常大的潜力，即使不用 Depth Separable Convolution 也能达到比现有方法更好的结果，预计在近期我们会公开相应的方法和模型。

抗过拟合。DenseNet 具有非常好的抗过拟合性能，尤其适合于训练数据相对匮乏的应用。这一点从论文中 DenseNet 在不做数据增强（data augmentation）的 CIFAR 数据集上的表现就能看出来。例如不对 CIFAR100 做数据增强，之前最好的结果是 28.20% 的错误率，而 DenseNet 可以将这一结果提升至 19.64%。对于 DenseNet 抗过拟合的原因有一个比较直观的解释：神经网络每一层提取的特征都相当于对输入数据的一个非线性变换，而随着深度的增加，变换的复杂度也逐渐增加（更多非线性函数的复合）。相比于一般神经网络的分类器直接依赖于网络最后一层（复杂度最高）的特征，DenseNet 可以综合利用浅层复杂度低的特征，因而更容易得到一个光滑的具有更好泛化性能的决策函数。实际上，DenseNet 的泛化性能优于其他网络是可以从理论上证明的：去年的一篇几乎与 DenseNet 同期发布在 arXiv 上的论文（AdaNet: Adaptive Structural Learning of Artificial Neural Networks）所证明的结论（见文中 Theorem 1）表明类似于 DenseNet 的网络结构具有更小的泛化误差界。

密集连接不会带来冗余吗？

这是一个很多人都在问的问题，因为「密集连接」这个词给人的第一感觉就是极大的增加了网络的参数量和计算量。**但实际上 DenseNet 比其他网络效率更高，其关键就在于网络每层计算量的减少以及特征的重复利用。DenseNet 的每一层只需学习很少的特征，使得参数量和计算量显著减少。**比如对于 ImageNet 上的模型，ResNet 在特征图尺寸为 7x7 的阶段，每个基本单元（包含三个卷积层）的参数量为 2048x512x1x1+512x512x3x3+512x2048x1x1=4.5M，而 DenseNet 每个基本单元（包含两个卷积层，其输入特征图的数量一般小于 2000）的参数量约为 2000x4x32x1x1 + 4x32x32x3x3 = 0.26M，大幅低于 ResNet 每层的参数量。这就解释了为什么一个 201 层的 DenseNet 参数量和计算量都只有一个 101 层 ResNet 的一半左右。

还有一个自然而然的问题就是，这么多的密集连接，是不是全部都是必要的，有没有可能去掉一些也不会影响网络的性能？论文里面有一个热力图（heatmap），直观上刻画了各个连接的强度。从图中可以观察到网络中比较靠后的层确实也会用到非常浅层的特征。