DenseNet翻译

摘要

近期的研究发现，如果在靠近输入和靠近输出的层之间存在连接的话，卷积神经网络可以更深，更加准确，并且能够更有效地进行训练。本文提出DenseNet。每一层与其他层之间使用反馈的方式进行连接。传统的L层卷积神经网络有L个连接，DenseNet有L(L+1)/2个直接连接。对于每一层，前面层的所有特征图作为输入，输出的特征图作为之后层的输入。DenseNet有以下几个优势：能够有效缓和梯度消失问题，增强特征传输，特征复用，同时减少参数的个数。在四个主要的目标识别任务上进行验证，DenseNet的结果在大多数的数据集上表现较好，在实现较高准确率的基础下，减少计算量。

Introduction

目前存在的问题：随着卷积层深度的增加，关于输入的信息或者梯度传递多层，在到达网络最后时，很有可能出现梯度小时和梯度爆炸现象。目前的方法例如ResNet能够减少这种现象，但都需要从前面的层到后面的层之间建立连接。

本文提出一种简单的连接模式，为了保证在网络中最大的信息流，将所有的层进行直接连接（匹配的特征图大小）。为保留反馈特征，每一层从前面的所有层获取输入，然后将其输出的特征图作为后面层的输入。与ResNet对比，没有使用求和来进行特征的连接，而是进行级联（concatenating）。因此，第l 层有l个输入，包括前面所有卷积模块的feature map。

Densenet的连接方式使得其比传统的卷积网络需要更少的参数，因为不需要对冗余的feature map重复进行学习。传统的反馈结构可以看作是一个状态，在层与层之间进行传递，每个层从前面的层读入状态并写入后面的层。在改变状态的同时保留有用的信息。ResNet通过额外的输入变换来保留信息，ResNet的一些变形算法发现，很多层的贡献比较小，在训练过程中可以随机drop。这使得ResNet与RNN相似，但是ResNet网络的参数很多，因为每一层都有权重。DenseNet 探索增加到网络的信息以及保留的信息之间的区别。将一小部分的特征图作为额外信息增加到网络，并维持保留信息不变，最终的分类器基于所有的特征图进行决策。

除了参数的有效性，DenseNet的另一个优势是，对于网络的信息以及梯度数据流进行优化，使得训练更容易。每一层能够直接获取损失函数的梯度以及输入的原始信号，因此网络的层次可以加深。另外，Dense连接方式具有正则化效果，减少过拟合。

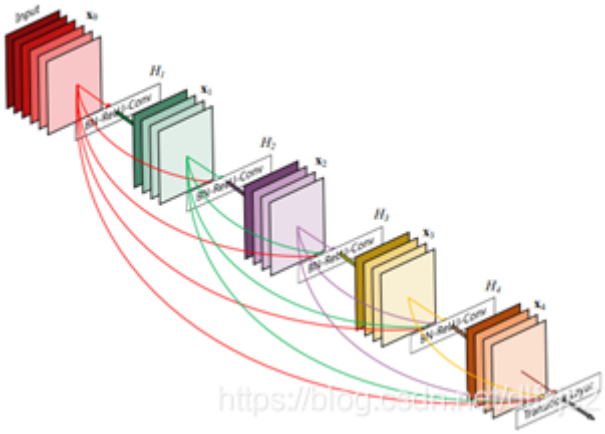
DenseNet

一幅图片输入一个L层卷积神经网络，每一层都有一个非线性变换$H_l(\cdot)$，l表示层的索引，$H_l(\cdot)$可以是BN，RELU，Pooling，或者卷积的混合操作符，第l层的输出为x_l。

ResNets

传统卷积神经网络将第l层的输出作为第$l+1$层的输入，表达式为：$\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1})$，ResNet增加了跨层连接$\mathbf{x}_\ell = H_\ell(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1}$，ResNet能够将梯度从后面层直接传递给前面层，但是identity函数和Hl的输出是使用加法进行连接，在网络中可能会阻碍信息流的的传播。

Dense连接



此时l层的表示为：$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]), [\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]$

表示$0, \ldots, \ell-1$层的特征图。

混合函数

$H_\ell(\cdot)$表示三个操作的混合：BN，RELU，以及3x3的卷积。

池化层

当特征图的大小改变时，连接操作不可行。但是下采样能够改变feature map的大小。为了应用下采样技术，将网络分成不同的dense模块。将block之间层成为过渡层，进行卷积和池化操作。在本文中使用的过渡层包括：一个BN层，一个1x1卷积层以及一个2x2的平均池化层。

增长速度

如果每个 $H_\ell(\cdot)$ 产生k个特征图，则第l层一共有 $k_0 + k \times (\ell - 1)$ 个输入特征图，k0为输入层的通道数。与已经存在的网络相比，DenseNet有更窄的层，比如k=12。将超参数k作为网络的增长速度，后续实验环节证明很小的增长速度就能够达到很好的效果。对此，可以解释为每一层都能够获取之前的曾的feature map，以及网络的“集体智慧”。可以将feature map作为网络的全局状态。每一层将其输出的k个特征图添加到这个状态中，增长率对每一层增加到全局状态的信息进行约束。全局信息已经写入，网络的所有层都可以获取，不需要从一个层到另外一个层进行复制。

Bottleneck层

尽管每一层都只有k个输出的feature map，但是有很多输入。1x1卷积可以作为bottleneck层放在3x3之前，来减少输入特征图的个数，因此，为了提高计算效率。本文使用的bottleneck层：BN-ReLU-Conv(1x1)-BN-ReLU-Conv(3x3)，称其为DenseNet-B。在实验中，使用1x1的卷积产生4k个feature map。

压缩

为了提高模型的紧促性，需要减少过渡层的 feature map 个数。如果一个 dense 模块中包含 m 个特征图，则其后面的过渡层产生 $\lfloor \theta m \rfloor$ 个输出，其中 $0 < \theta \leq 1$ 作为压缩比。当 $\theta < 1$ 时，DenseNet-C，实验中取值 0.5。既使用 bottleNet，又使用压缩的网络称为 DenseNet-BC。

实现细节

在实验中，使用三个dense模块，每个模块有相同数量的层。在进入第一个dense模块之前，使用一个16输出通道的卷积层对输入图片进行卷积（或者对于DenseNet-BC而言是增长速度的二倍）。对于每个3x3的卷积层，每个输入都使用0填充来保持特征图的大小固定。在两个连续的dense模块之间，使用1x1的卷积和2x2的平均池化层。在最后一个dense模块后面，使用全局平均池化，然后使用softmax分类器。Dense模块的特征图大小为32x32,16x16,8x8。基础结构{L=40,k=12},{L=100,k=12},{L=100,k=24}，对于DenseNet，网络的默认参数为{L=100,k=12},{L=250,k=24},{L=190,k=40}。

在实验中，使用DenseNet-BC结构，有四个dense模块，对224*224的输入图片，最开始的卷积层2k个7x7的卷积，stride大小为2，其他所有层的feature map个数设置为k。

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [32]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [34]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [42]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

总结

DenseNet的优势在于

1. 有效解决梯度消失问题
2. 加强了特征的传播
3. 有效地减少了网络参数
4. 特征重用性高