目录

卷积公式

- 输入为$(N, C_{in}, H_{in}, W_{in})$，输出为$(N, C_{out}, H_{out}, W_{out})$:

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times padding - dilation \times (kernel - 1) - 1}{stride} + 1 \right\rfloor$$

关于卷积层池化层的特点、作用

- s使用小卷积核（如3x3）通过多层叠加可取得与大卷积核（如7x7）同等规模的感受野，此外采用小卷积核有两个优势：
  - 小卷积核需多层叠加，加深了网络深度进而增强了网络容量(model capacity)和复杂度（model complexity)
  - 增强了网络容量的同时减少了参数个数。
- 卷积的作用：通过卷积核的组合以及随着网络后续操作的进行，卷积操作可获取图像区域不同类型特征；基本而一般的模式会逐渐被抽象为具有高层语义的“概念”表示，也就是自动学习到图像的高层特征。
- CNN的结构特点：
  - **局部连接**使网络可以提取数据的局部特征。卷积核尺度远小于输入的维度，这样每个输出神经元仅与前一层特点局部区域的神经元存在连接权重。局部连接的物理意义是，图像、文本、语音等数据一般具有局部的特征结构，可以先学习局部的特征，再将局部的特征组合起来形成更复杂和抽象的特征。
  - **权值共享**大大降低了网络的训练难度，一个Filter只提取一个特征，在整个图片（或者语音 / 文本） 中进行卷积。参数共享的物理意义是使得卷积层具有平移等变性，也就是说神经网络的输出对于平移变换来说是等变的。
  - **池化操作**与多层次结构一起，实现了数据的降维，将低层次的局部特征组合成为较高层次的特征，从而对整个图片进行表示。
- 池化层的作用：
  - 平均池化能够抑制由于领域大小受限造成估计值方差增大的现象，对背景的保留效果更好
  - 最大池化能够抑制网络参数误差造成的估计值偏移现象，能更好地提取纹理信息
  - 增加特征平移、伸缩、旋转操作的不变性。

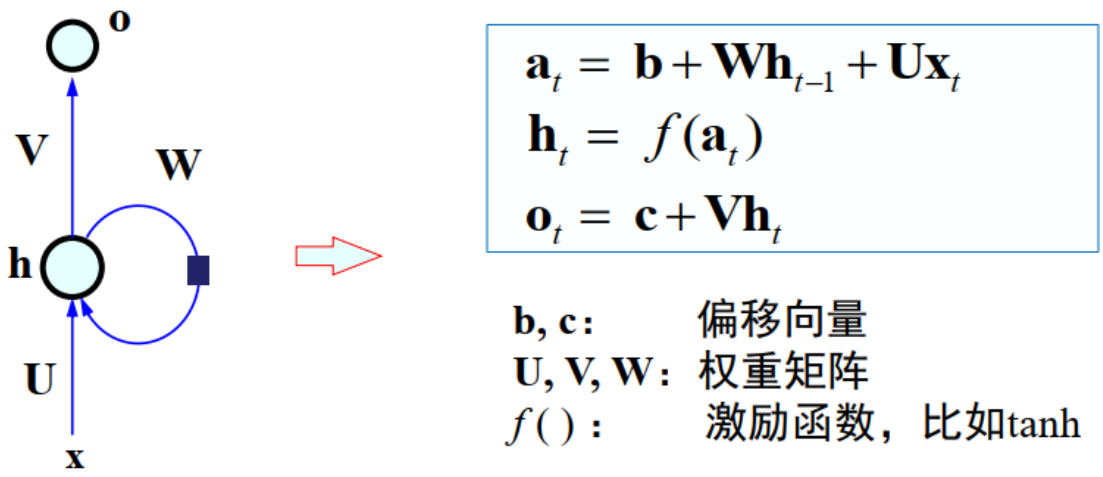
- 减小特征图大小。汇合层对空间局部区域进行下采样，使下一层需要的参数量和计算量减少，并降低过拟合风险。
- 最大汇合可以带来非线性。这是目前最大汇合更常用的原因之一。

1x1卷积的作用

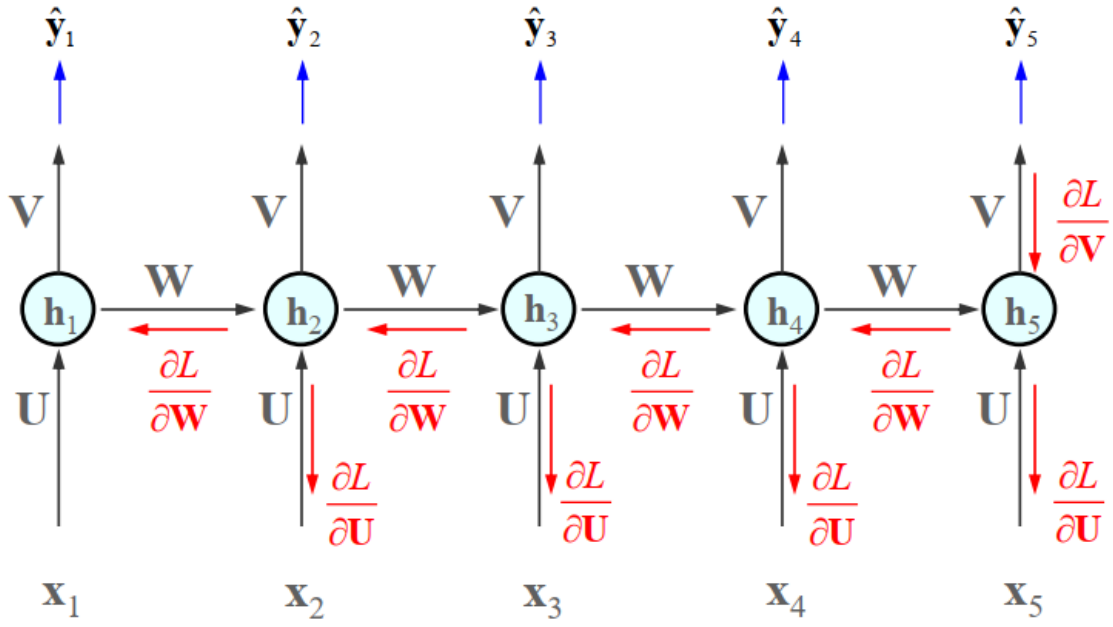
- 降维（ dimension reductionality ）。比如，一张500 * 500且厚度depth为100 的图片在20个filter上做11的卷积，那么结果的大小为500500*20。
- 加入非线性。卷积层之后经过激励层，1*1的卷积在前一层的学习表示上添加了非线性激励（ non-linear activation ），提升网络的表达能力。

RNN原理

- RNN结构：

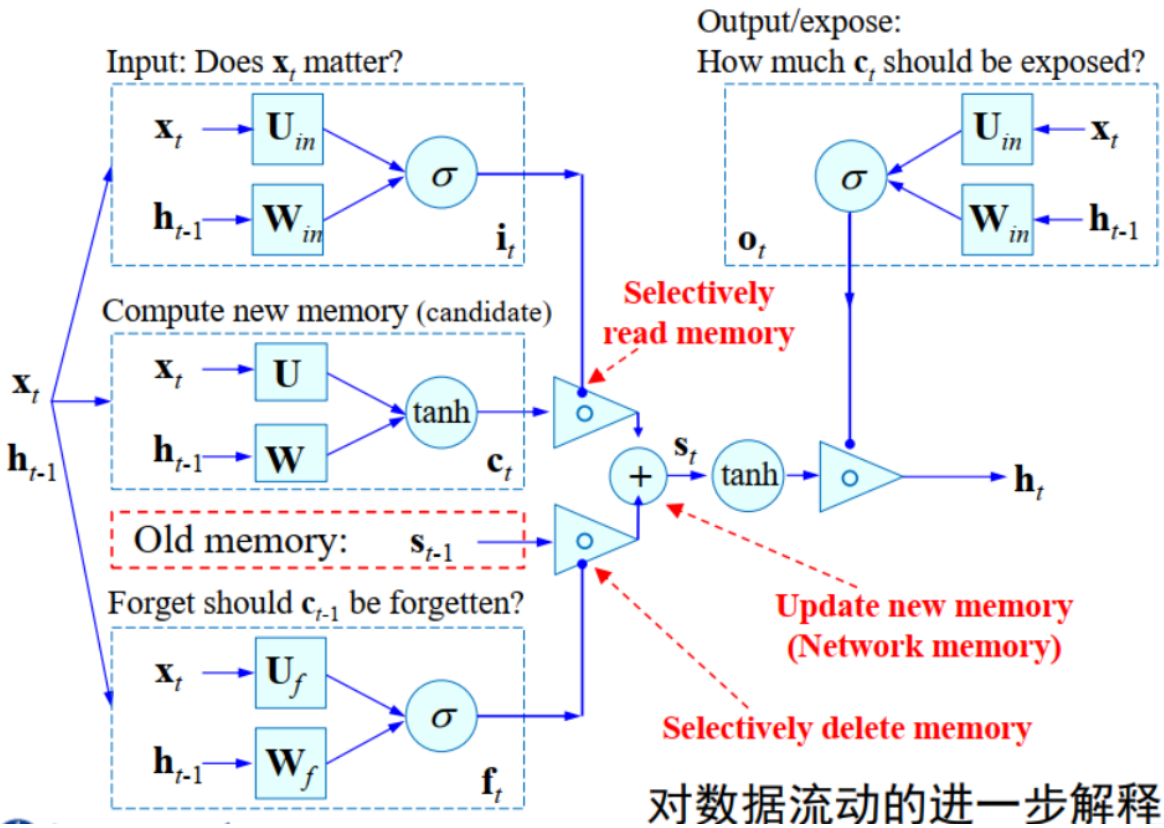


- RNN的训练：



LSTM原理

- LSTM的结构：



- LSTM的门：
  - 输入门：

$$\mathbf{i_t} = sigmoid(\mathbf{b_{in}} + \mathbf{U_{in}}\mathbf{x_t} + \mathbf{W_{in}}\mathbf{h_{t-1}})$$

  - 输出门：

$$\mathbf{o_t} = sigmoid(\mathbf{b_o} + \mathbf{U_o}\mathbf{x_t} + \mathbf{W_o}\mathbf{h_{t-1}})$$

  - 遗忘门：

$$\mathbf{f_t} = sigmoid(\mathbf{b_f} + \mathbf{U_f}\mathbf{x_t} + \mathbf{W_f}\mathbf{h_{t-1}})$$

  - 候选记忆：

$$\mathbf{c_t} = \tanh(\mathbf{b} + \mathbf{U}\mathbf{x_t} + \mathbf{W}\mathbf{h_{t-1}})$$

  - cell的产生的新的记忆：

$$\mathbf{s_t} = \mathbf{f_t} \bigotimes \mathbf{s_{t-1}} + \mathbf{i_t} \bigotimes \mathbf{c_t}$$

  - cell的输出：

$$\mathbf{h_t} = \mathbf{o_t} \bigotimes \tanh(\mathbf{s_t})$$

导致模型不收敛的原因

- 没有对数据做归一化。
- 没有检查过你的结果。这里的结果包括预处理结果和最终的训练测试结果。
- 忘了做数据预处理。
- 忘了使用正则化。
- Batch Size设的太大。
- 学习率设的不对。
- 最后一层的激活函数用的不对。
- 网络存在坏梯度。比如Relu对负值的梯度为0，反向传播时，0梯度就是不传播。
- 参数初始化错误。
- 网络太深。隐藏层神经元数量错误。

深度学习中的正则化

- **参数范数惩罚。**只对权重惩罚，不对偏置惩罚（权重指定了变量如何相互作用，偏置更多影响但变量），不对偏置正则化也不会导致太大的方差，可能到时明显地欠拟合。
  - L_2正则化。在每步执行梯度更新之前，先收缩权重向量。在Hessian矩阵上的表现就是沿着由H的特征向量所定义的轴放缩w^*，缩放因子为$\frac{\lambda_i}{\lambda_i+\alpha}$。H的特征值较大的方向，正则化影响较小，较小的特征值几乎收缩为0，不重要的方向会被衰减掉
  - L_1正则化。L_1会产生更稀疏的解。
  - 可以把参数范数惩罚看作对权重的强加约束，L1是正方形约束，L2是球形约束
- **数据增强。**
- **添加噪声。**
  - 向输入添加方差极小的噪声 $\Longleftrightarrow$ 对权重施加范数惩罚。注入噪声比收缩参数更强大
  - 噪声还可以添加到输出，如标签平滑（能防止模型追求确切概率而不影响模型正确分类）
- **提前终止。**
- **Bagging和其他其他基础方法。**
- **Dropout。**
  - 将一些单元输出乘0，可理解为bagging，不同模型的平均。
  - Dropout中所有模型共享参数，每个模型即父神经网络的不同子集。
  - Dropout用于线性回归，相当于L2权重衰减

BN及其作用作用?

- BN的前向传导公式：

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu)^2$$

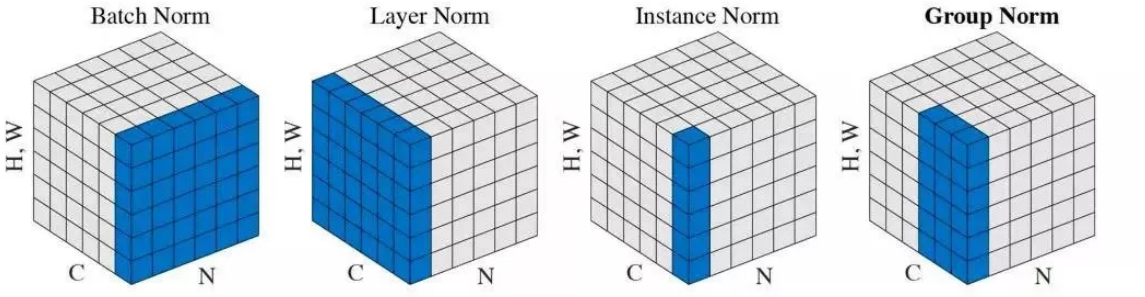
$$\hat{x_i} = \frac{x_i-\mu}{\sqrt{\sigma^2+\epsilon}}$$

$$y_i = \gamma\hat{x_i}+\beta$$

- 注意在卷积网络中，每个卷积核的参数在不同位置的神经元当中是共享的，因此也应该一起被归一化。假设每一批包含b个样本，由f个卷积核生成的的特征图大小为$f \times w \times h$，则每个特征图对应的全部神经元个数为$b \times w \times h$，对应f组不同的γ和β参数
- 可以使用更高的学习率。如果每层的scale不一致，实际上每层需要的学习率是不一样的，同一层不同维度的scale往往也需要不同大小的学习率，通常需要使用最小的那个学习率才能保证损失函数有效下降，Batch Normalization将每层、每维的scale保持一致，那么我们就可以直接使用较高的学习率进行优化。
- 移除或使用较低的dropout。 dropout是常用的防止overfitting的方法，而导致overfit的位置往往在数据边界处，如果初始化权重就已经落在数据内部，overfit现象就可以得到一定的缓解。论文中最后的模型分别使用10%、5%和0%的dropout训练模型，与之前的40%-50%相比，可以大大提高训练速度。
- 降低L2权重衰减系数。还是一样的问题，边界处的局部最优往往有几维的权重（斜率）较大，使用L2衰减可以缓解这一问题，现在用了Batch Normalization，就可以把这个值降低了，论文中降低为原来的5倍。
- 取消Local Response Normalization层。由于使用了一种Normalization，再使用LRN就显得没那么必要了。而且LRN实际上也没那么work。
- Batch Normalization调整了数据的分布，不考虑激活函数，它让每一层的输出归一化到了均值为0方差为1的分布，这保证了梯度的有效性，可以解决反向传播过程中的梯度问题。目前大部分资料都这样解释，比如BN的原始论文认为的缓解了Internal Covariate Shift(ICS)问题。

如何理解Group Normalization?

- Group Normalization（GN）是针对Batch Normalization（BN）在batch size较小时错误率较高而提出的改进算法，因为BN层的计算结果依赖当前batch的数据，当batch size较小时（比如2、4这样），该batch数据的均值和方差的代表性较差，因此对最后的结果影响也较大。
- 下图是是几种归一化方式的对比（Bartch Norm、Layer Norm、Instance Norm和Group Norm），可以一并回顾下BN算法。下图中的立方体是三维，其中两维C和N分别表示channel和batch size，第三维表示H,W，可以理解为该维度大小是H*W，也就是拉长成一维，这样总体就可以用三维图形来表示。可以看出BN的计算和batch size相关（蓝色区域为计算均值和方差的单元），而LN、BN和GN的计算和batch size无关。同时LN和IN都可以看作是GN的特殊情况（LN是group=C时候的GN，IN是group=1时候的GN）。
    - BN在batch的维度上norm，归一化维度为[N，H，W]，对batch中对应的channel归一化；
    - LN避开了batch维度，归一化的维度为[C，H，W]；
    - IN 归一化的维度为[H，W]；
    - 而GN介于LN和IN之间，其首先将channel分为许多组（group），对每一组做归一化，及先将feature的维度由[N, C, H, W]reshape为[N, G，C//G , H, W]，归一化的维度为[C//G , H, W]



VGG使用2个3x3卷积核的优势?

- 减少网络层参数。用两个3x3卷积比用1个5x5卷积拥有更少的参数量，只有后者的2∗3∗3/5∗5=0.72。但是起到的效果是一样的，两个3x3的卷积层串联相当于一个5x5的卷积层，感受野的大小都是5×5，即1个像素会跟周围5*5的像素产生关联。把下图当成动态图看，很容易看到两个3×3卷积层堆叠（没有空间池化）有5×5的有效感受野。
- 更多的非线性变换。2个3x3卷积层拥有比1个5x5卷积层更多的非线性变换（前者可以使用两次ReLU激活函数，而后者只有一次），使得卷积神经网络对特征的学习能力更强。

Relu相比于sigmoid、tanh的优点和局限性

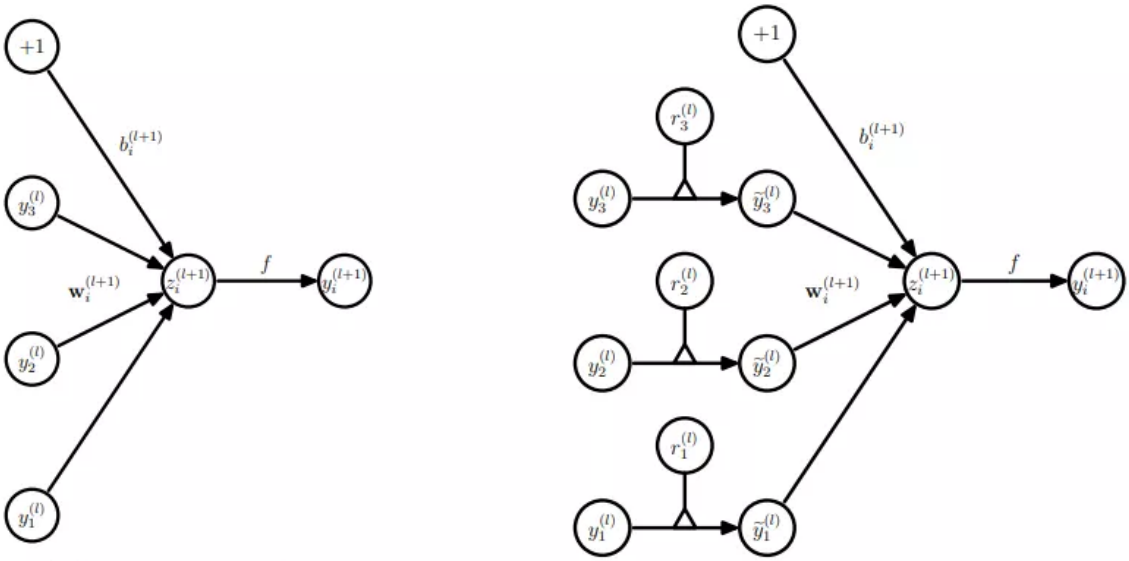
- 优点：
    - **ReLU函数计算简单**，可以减少很多计算量。反向传播求误差梯度时，涉及除法，计算量相对较大，采用ReLU激活函数，可以节省很多计算量；
    - **避免梯度消失问题。**对于深层网络，sigmoid函数反向传播时，很容易就会出现梯度消失问题（在sigmoid接近饱和区时，变换太缓慢，导数趋于0，这种情况会造成信息丢失），从而无法完成深层网络的训练。
    - **可以缓解过拟合问题的发生。**Relu会使一部分神经元的输出为0，这样就造成了网络的稀疏性，并且减少了参数的相互依存关系，缓解了过拟合问题的发生。
    - 相比sigmoid型函数，ReLU函数有助于随机梯度下降方法收敛。
- 局限性：
    - 会导致神经元死亡。

Dropout原理

以标准神经网络为例，正常的流程是：我们首先把输入数据x通过网络前向传播，然后把误差反向传播一决定如何更新参数让网络进行学习。使用dropout之后，过程变成如下：

- 首先随机（临时）删掉网络中一半的隐藏神经元，输入输出神经元保持不变（图3中虚线为部分临时被删除的神经元）；
- 然后把输入x通过修改后的网络进行前向传播计算，然后把得到的损失结果通过修改的网络反向传播。一小批训练样本执行完这个过程后，在没有被删除的神经元上按照随机梯度下降法更新对应的参数（w，b）；
- 然后重复这一过程：
    - 恢复被删掉的神经元（此时被删除的神经元保持原样没有更新w参数，而没有被删除的神经元已经有所更新）
    - 从隐藏层神经元中随机选择一个一半大小的子集临时删除掉（同时备份被删除神经元的参数）。
    - 对一小批训练样本，先前向传播然后反向传播损失并根据随机梯度下降法更新参数（w，b）（没有被删除的那一部分参数得到更新，删除的神经元参数保持被删除前的结果）。

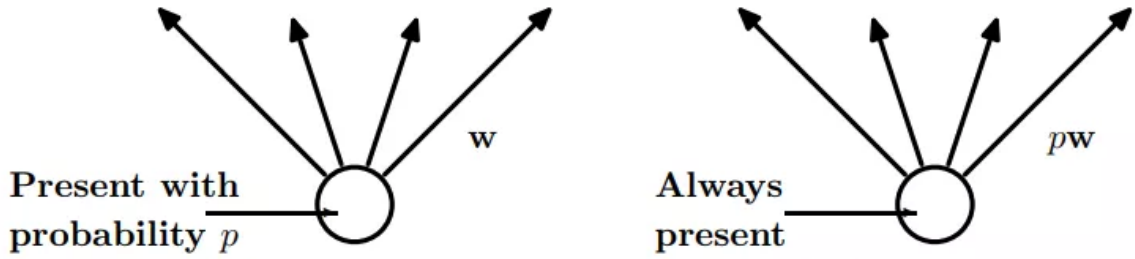
在训练过程中，不可避免的，在训练网络中的每个单元都要添加一道概率流程，标准网络和带有dropout网络的比较图如下所示：



(a) Standard network　　(b) Dropout network

Figure 3: Comparison of the basic operations of a standard and dropout network.

在测试阶段，输入是当前输入，每个神经单元的权重参数要乘以概率p，以恢复在训练过程中该神经元只有p的概率被用于整个网络的前向传播计算



(a) At training time　　(b) At test time

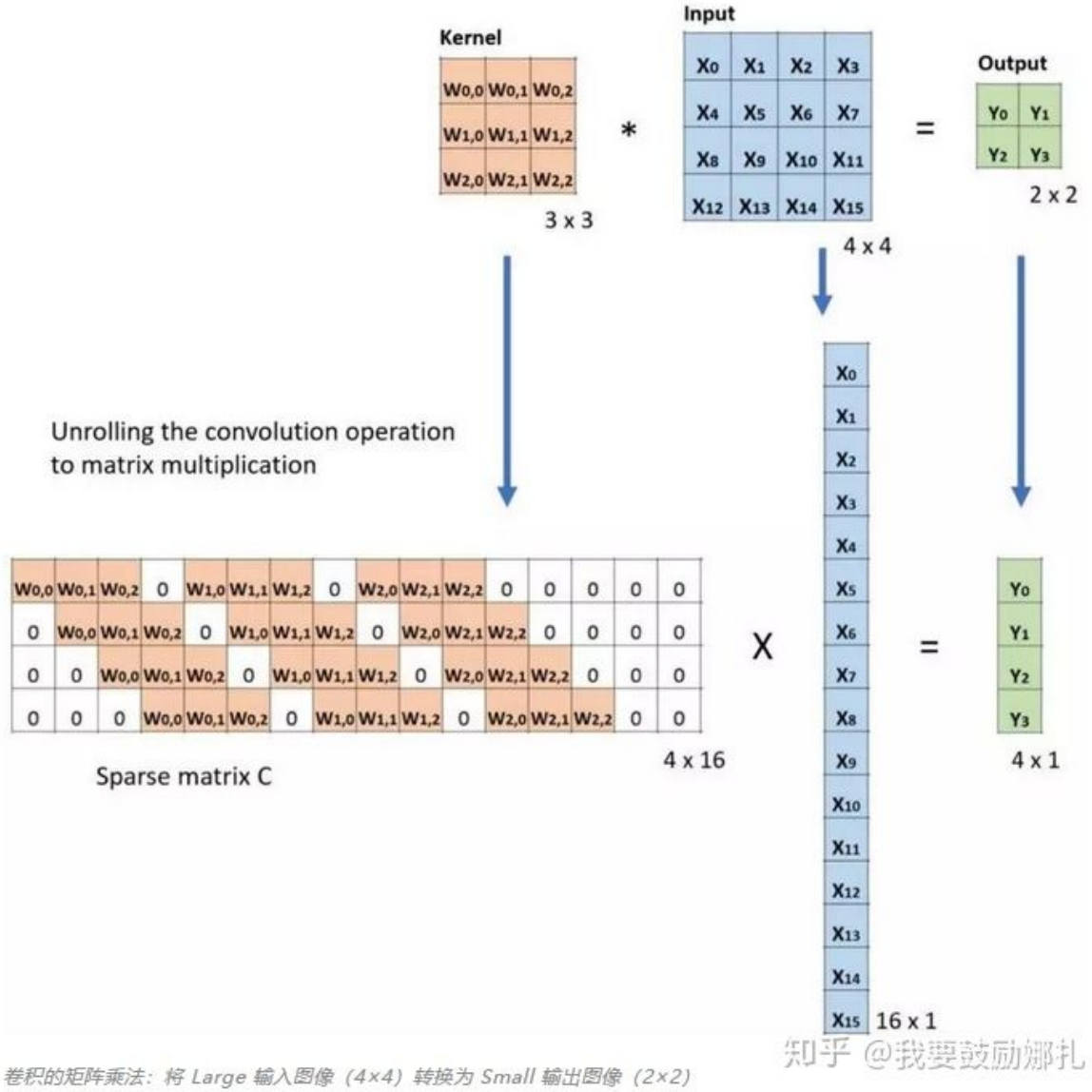
解释空洞卷积原理

- Dilation卷积，通常译作空洞卷积或者卷积核膨胀操作，它是解决pixel-wise输出模型的一种常用的卷积方式。CNN的一些缺陷：

    - Up-sampling / pooling layer (e.g. bilinear interpolation) is deterministic. (a.k.a. not learnable)
    - 内部数据结构丢失；空间层级化信息丢失。
    - 小物体信息无法重建 (假设有四个pooling layer 则 任何小于 2^4 = 16 pixel 的物体信息将理论上无法重建。)
- 在这样问题的存在下，语义分割问题一直处在瓶颈期无法再明显提高精度， 而 dilated convolution 的设计就良好的避免了这些问题。
- Hybrid Dilated Convolution (HDC)的特性：
    - 叠加卷积的 dilation rate 不能有大于1的公约数。比如 [2, 4, 6] 则不是一个好的三层卷积，依然会出现 gridding effect。
    - 我们将 dilation rate 设计成 锯齿状结构，例如 [1, 2, 5, 1, 2, 5] 循环结构。
    - 我们需要满足一下这个式子：$M_i = \max[M_{i+1} - 2r_i, M_{i+1} - 2(M_{i+1} - r_i), r_i]$，$r_i$是第$i$层的dilation rate，$M_i$为第$i$层最大的dilation rate，假设我们应用于 kernel 为 k x k 的话，我们的目标则是$M_2 \leq k$，这样我们至少可以用 dilation rate 1 即 standard convolution 的方式来覆盖掉所有洞。

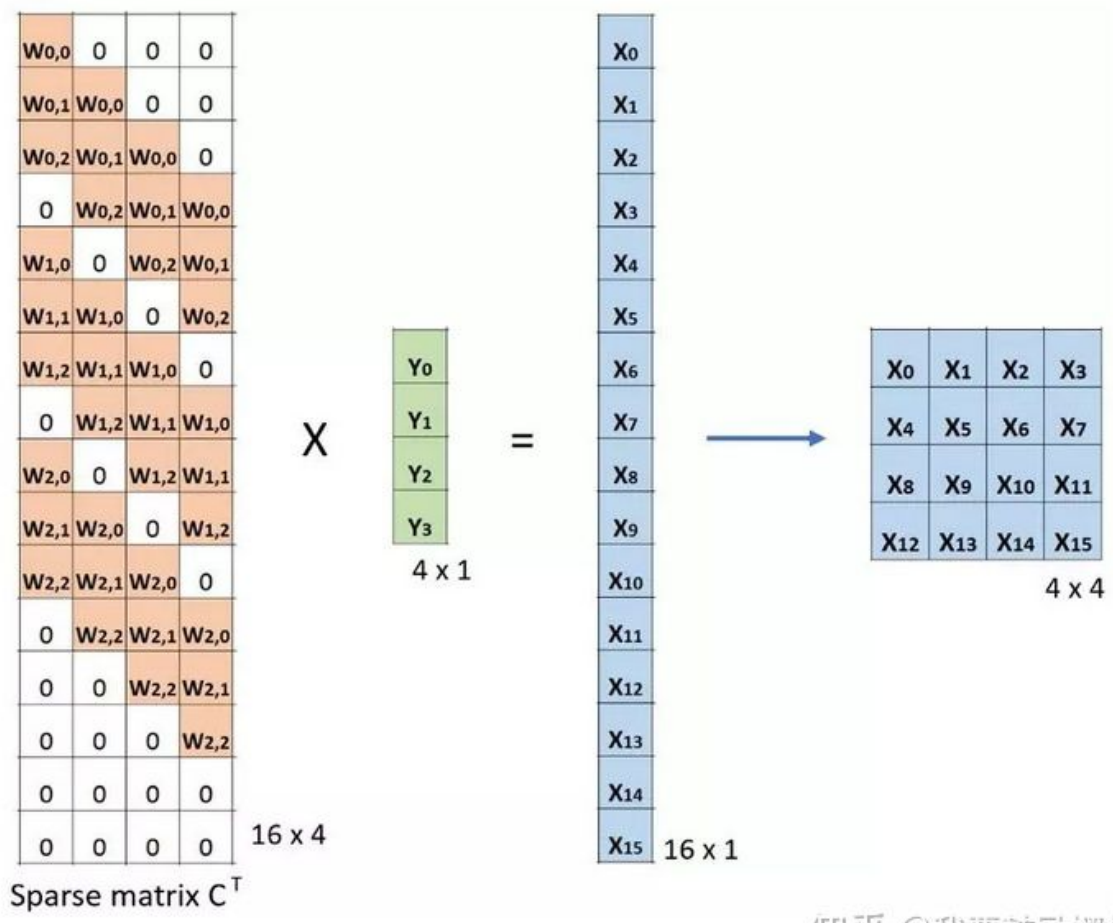
如何理解转置卷积

- **转置卷积**（transposed Convolutions）又名**反卷积**（deconvolution）或是**分数步长卷积**（fractially straced convolutions）。从字面上理解就是卷积的逆过程。值得注意的反卷积虽然存在，但是在深度学习中并不常用。而转置卷积虽然又名反卷积，却不是真正意义上的反卷积。因为根据反卷积的数学含义，通过反卷积可以将通过卷积的输出信号，完全还原输入信号。而事实是，转置卷积只能还原shape大小，而不能还原value。你可以理解成，至少在数值方面上，转置卷积不能实现卷积操作的逆过程。所以说转置卷积与真正的反卷积有点相似，因为两者产生了相同的空间分辨率。但是又名反卷积（deconvolutions）的这种叫法是不合适的，因为它不符合反卷积的概念。
- 在卷积实际计算过程中，我们要转化为矩阵的乘积的形式，**一个转化为Toeplitz matrix 托普利茨矩阵，一个reshape为列矩阵。**普通卷积的计算过程：



卷积的矩阵乘法：将 Large 输入图像（4×4）转换为 Small 输出图像（2×2）

- 转置卷积的计算过程：

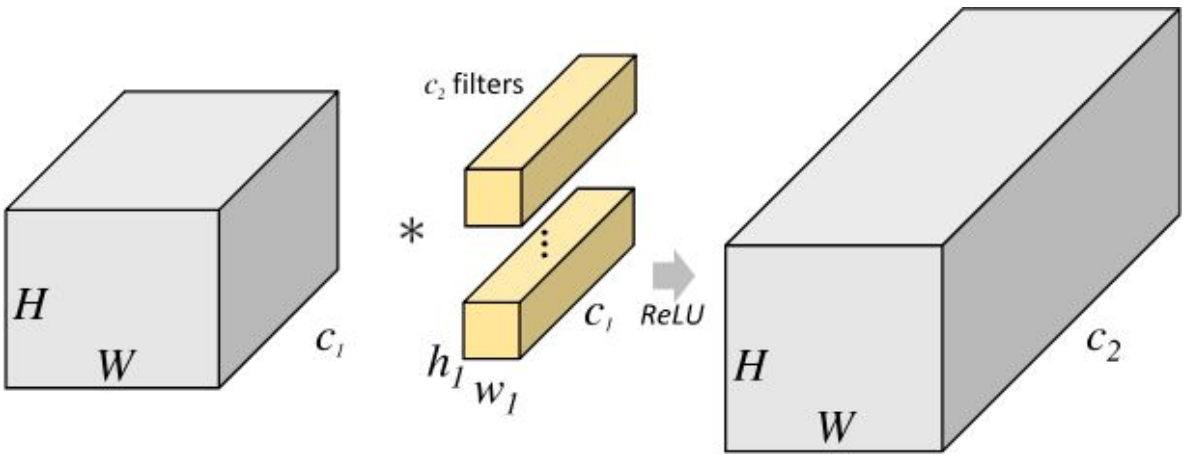


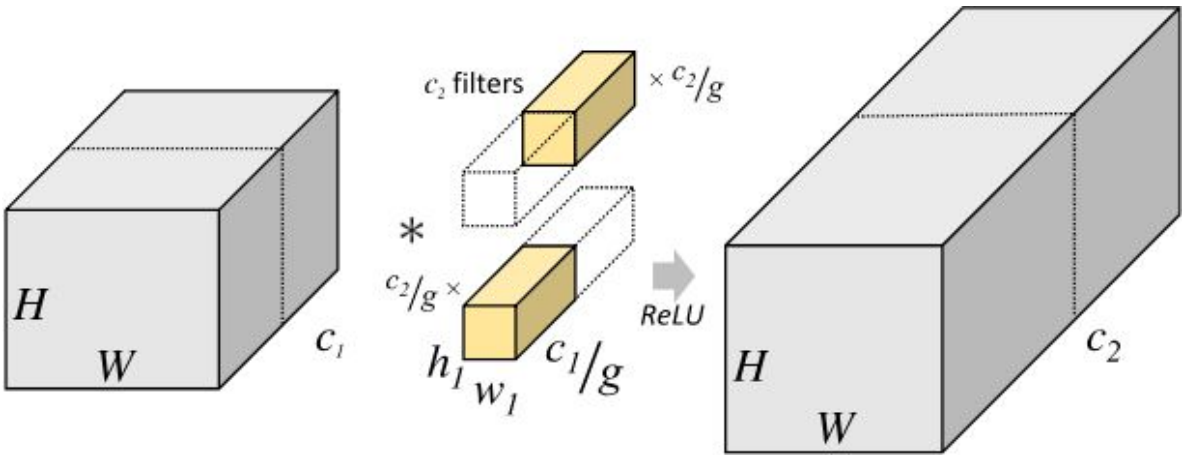
卷积的矩阵乘法：将 Small 输入图像（2×2）转换为 Large 输出图像（4×4）

- 转置卷积输出尺寸计算，输入为$(N, C_{in}, H_{in}, W_{in})$，输出为$(N, C_{out}, H_{out}, W_{out})$:

$$H_{out} = (H_{in-1}) \times stride - 2 \times padding + dilation \times (kernel_size - 1) + output\ padding + 1$$

什么是分组卷积GroupConvolution





第一张图为标准卷积，第二张图为分组卷积。将输入特征图按照通道数分成g组，则每组输入特征图的尺寸为$H \times W \times \left(\frac{c_1}{g}\right)$，对应的卷积核尺寸为$h_1 \times w_1 \times \left(\frac{c_1}{g}\right)$，每组输出特征图尺寸为$H \times W \times \left(\frac{c_2}{g}\right)$。将$g$ 组结果拼接(concat)，得到最终尺寸为$H \times W \times c_2$ 的输出特征图。分组卷积层的参数量为

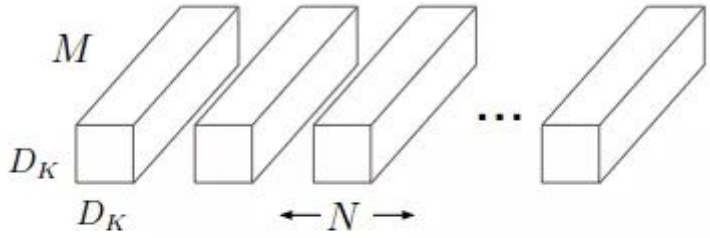
$h_1 \times w_1 \times (\frac{c_1}{g}) \times (\frac{c_2}{g}) \times g = h_1 \times w_1 \times c_1 \times c_2 \times \frac{1}{g}$ 。

什么是深度可分离卷积

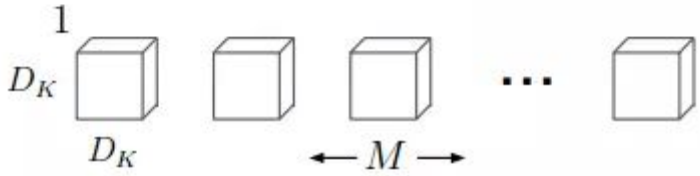
实际速度与理论速度差距较大，解释原因。

假设有一个3×3大小的卷积层，其输入通道为16、输出通道为32。具体为，32个3×3大小的卷积核会遍历16个通道中的每个数据，从而产生16×32=512个特征图谱。进而通过叠加每个输入通道对应的特征图谱后融合得到1个特征图谱。最后可得到所需的32个输出通道。

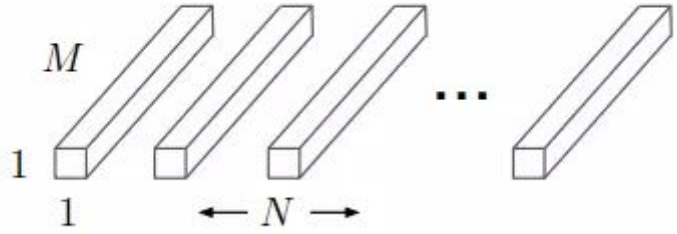
针对这个例子应用深度可分离卷积，用1个3×3大小的卷积核遍历16通道的数据，得到了16个特征图谱。在融合操作之前，接着用32个1×1大小的卷积核遍历这16个特征图谱，进行相加融合。这个过程使用了16×3×3+16×32×1×1=656个参数，远少于上面的16×32×3×3=4608个参数。



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution