# 一 摘要

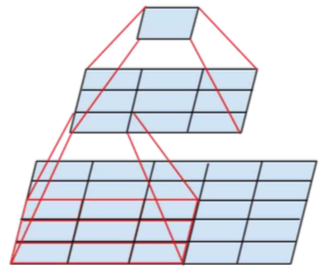
遇到的问题: 计算效率和低参数计数仍是各种应用场景的限制因素.

目前通过适当的分解卷积,并用积极正则化来增大网络.

Inception架构的复杂性使得更难以对网络进行更改。如果单纯地放大架构，大部分的计算收益可能会立即丢失。

# 二 inception优化参数思路

## 使用更小的卷积



利用两个3x3的网络代替5x5的网络,第二个3x3网络实际上做一个fc的工作.这样可以保证感受野和5x5一致.

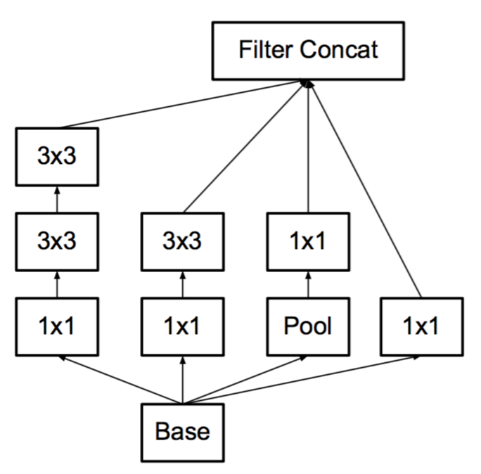
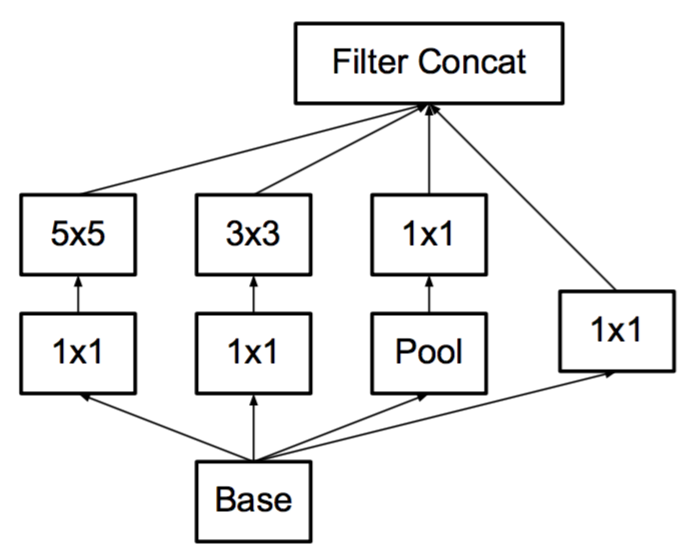
比如如下代码实现,第二层的3x3是采用stride=2来做的.这样期望的感受野是一样的,当然了,5x5和之后的2个3x3的weights param肯定是不相同的,是train出来的.但是从感受野角度是一样的.

**关于节省的点:**

1. 肯定是节省在参数量上,具体到,5x5的是有25个参数.他们后面接fc层.3x3虽然有两个,但是只计算第一组的是9个.第二组不算的原因是,第二组是fc.
2. 性能上,应该是2组3x3的更加耗时,因为有更多的乘法操作.但是在服务器上训练,这部分应该不纳入考虑.
3. 我们说的性能不是soc的性能,而是让参数少,这样train的更少,这样更加容易收敛.这个时间应该远大于soc上乘法操作多一些而增加的耗时.
4. 我们最终得到一个计算量减少到(9+9)/25的网络，通过这种分解导致了28％的相对增益。

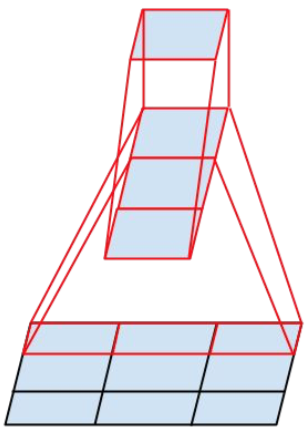
|  |
| --- |
| branch\_1 = slim.conv2d(branch\_1, depth(96), [3, 3],  scope='Conv2d\_0b\_3x3')  branch\_1 = slim.conv2d(branch\_1, depth(96), [3, 3], **stride=2**,  padding='VALID', scope='Conv2d\_1a\_1x1') |

比如,最初的inception模块(左),和改进后的inception模块(右):



把5x5的卷积拆成两个3x3.

## 使用非对称的卷积



将3x3的卷积拆解成3x1和1x3.这样感受野一致,节省1/3的参数.

**怎么理解节省1/3:**

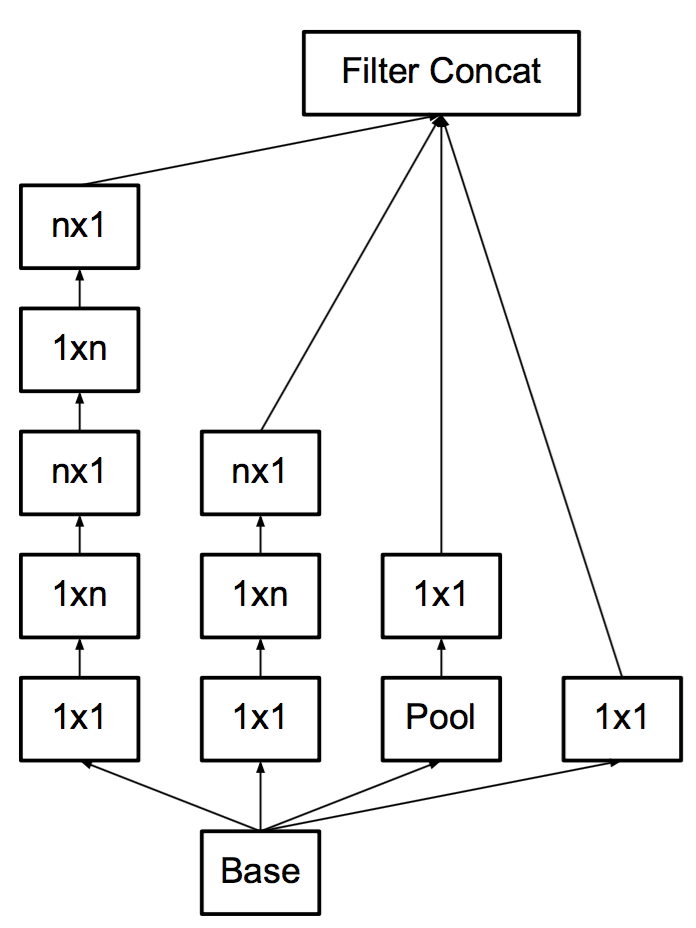
1. 从第一层3x1来看,引入”权值共享”,第一层的参数是3个,第二层的参数也是3个.这样有6个参数.
2. 3x3会有9组参数.
3. 这个也是参数数量上的优化,但是并非soc乘法操作的领域.

进一步引申,把nxn卷积拆解成nx1和1xn的结构:

共节省: (n\*n-2n)/(n\*n)=(n-2)/n

并且随着n增长，计算成本节省显著增加,但实际效果会变差.

|  |  |
| --- | --- |
| n | (n-2)/n |
| 3 | 1/3 |
| 5 | 3/5 |



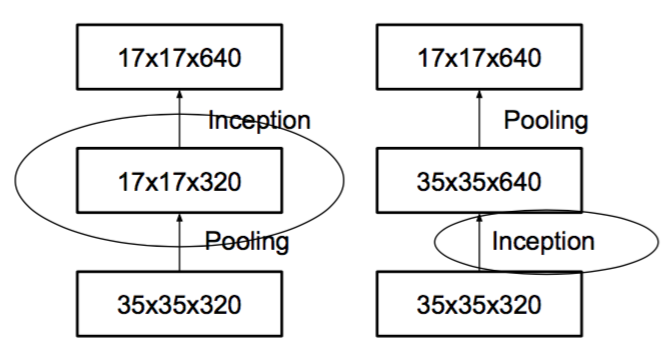
注意: 我们发现，采用这种分解在前面的层次上不能很好地工作，但是**对于中等网格尺寸（在m×m特征图上，其中m范围在12到20之间），其给出了非常好的结果**。

## 有效的网格尺寸减少

对比下图的两个网络结构:

左边的先做pooling,参数下降到1/4. 然后接inception.这样运算操作会减少很多.但是由于传入的图小了,inception会表现很差,认为是遇到表现瓶颈.

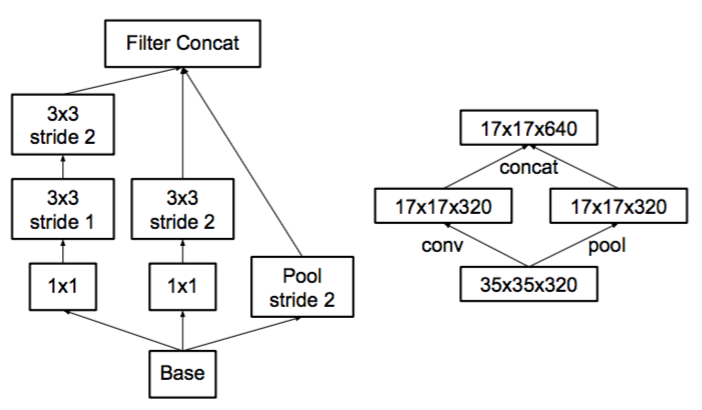
右边的先做inception在做pooling,虽然inception的表现力很好,但是运算很多.



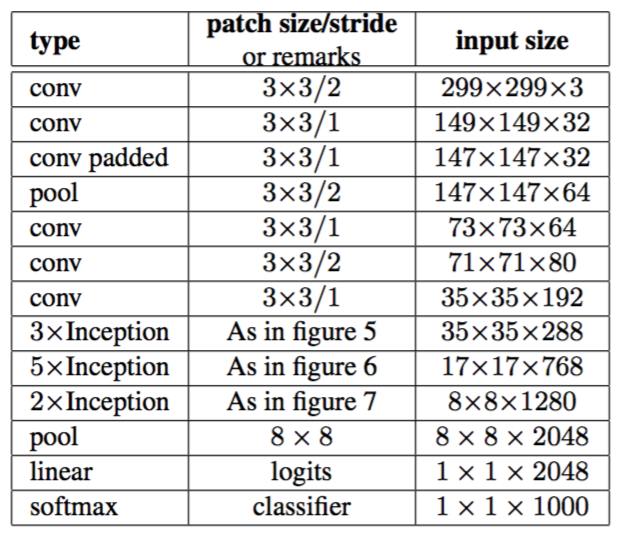
为了解决这组矛盾,引入如下的网络:

右图是数据量视角的等价图.

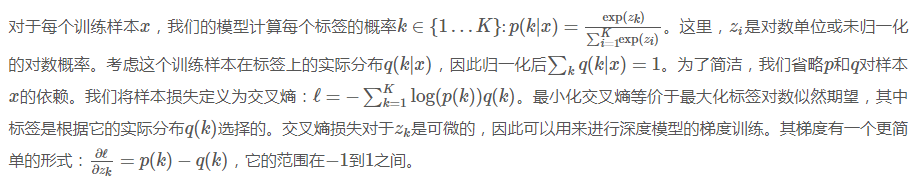
输入的是大图,过程中有pooling. 缩减网格尺寸的同时扩展滤波器组的Inception模块。它不仅廉价并且避免了原则1中提出的表示瓶颈。



网络结构:



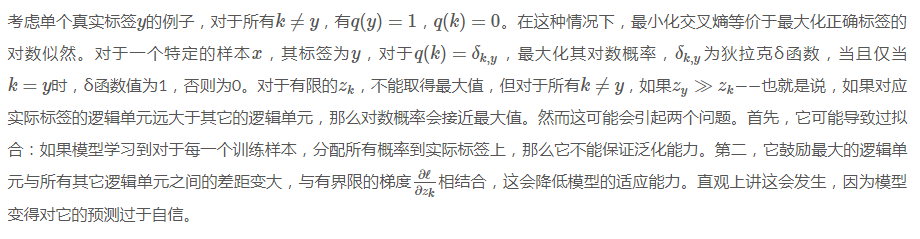
## 通过标签平滑进行模型正则化



这里引入了对于每个样本标签的分布.q(k|x)是其实际分布.

自然地, 我们会定义一个交叉熵的损失函数.

交叉熵对zk是可微的.因此可以用梯度下降来求解.



**这种标签分布有什么弊端呢?**

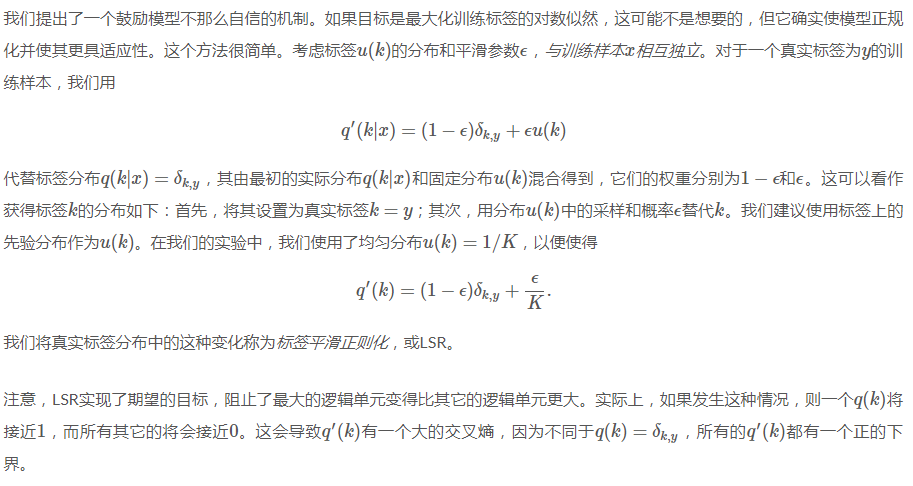
这种分布是狄拉克δ函数.在真是标签上q(y)=1,其他的标签上为零.它对真值过于自信,因而有较差的泛化能力.

什么是狄拉克δ函数?

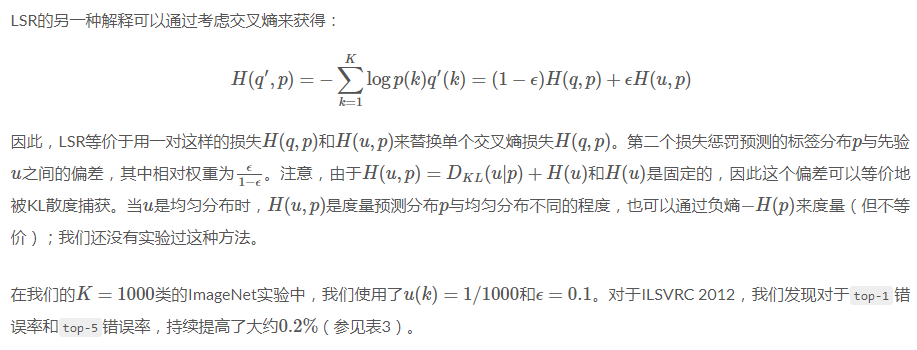
|  |
| --- |
| **狄拉克δ函数**是一个广义函数，在物理学中常用其表示质点、点电荷等理想模型的密度分布，该函数在除了零以外的点取值都等于零，而其在整个定义域上的积分等于1。  狄拉克δ函数在概念上，它是这么一个“函数”：在除了零以外的点函数值都等于零，而其在整个[定义域](https://baike.baidu.com/item/%E5%AE%9A%E4%B9%89%E5%9F%9F/7879679)上的积分等于1。  https://gss3.bdstatic.com/-Po3dSag_xI4khGkpoWK1HF6hhy/baike/w%3D268%3Bg%3D0/sign=f3aae413a84bd11304cdb0346294c332/30adcbef76094b3697901b8da6cc7cd98d109d4b.jpg |

**采用什么方法抑制?**

采用一个模糊的方法.



用狄拉克δ函数和一个样本的先验分布组合.这样缩小真值和伪值之间的gap.



采用此法更新下交叉熵.

## 低分辨率输入上的性能

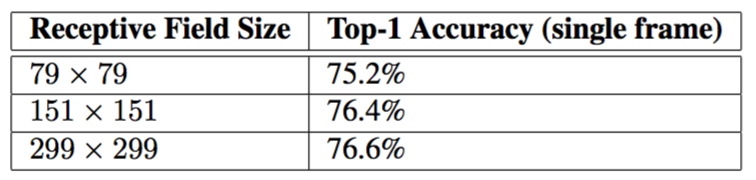
在较低分辨率输入的情况下减少前两层的步长，或者简单地移除网络的第一个池化层。

为了这个目的我们进行了以下三个实验：

步长为2，大小为299×299的感受野和最大池化。

步长为1，大小为151×151的感受野和最大池化。

步长为1，大小为79×79的感受野和第一层之后没有池化。



准确率有稍微的损失.