

# 深度学习大神都推荐入门必须读完这9篇论文

2018-01-16 深度学习世界

差一点  我们就擦肩而过了

来源：大数据挖掘DT数据分析

ID: datadw

## Introduction

卷积神经网络**CNN**，虽然它听起来就像是生物学、数学和计算机的奇怪混杂产物，但在近些年的机器视觉领域，它是最具影响力的创新结果。随着Alex Krizhevsky开始使用神经网络，将分类错误率由26%降到15%并赢得2012年度ImageNet竞赛（相当于机器视觉界的奥林匹克）时，它就开始声名大噪了。从那时起，一票公司开始在它们的核心服务中使用深度学习技术。例如Facebook用它进行自动的图像标签，google用它做照片检索，amazon用它做产品推荐，Pinterest用它做个性化家庭定制推送（？），Instagram用它搭建他们的搜索架构。

基于**CNN**最经典也是最流行的应用应当是图像处理领域。那么，就让我们看看怎样使用**CNN**技术来设计图像分类算法。



<http://blog.csdn.net/>

## The Problem Space

图像分类主要指将输入图像进行硬分类或模糊分类（例如猫图、狗图等）。对于人类来说，这是出生后就应当学会的第一个技能，并在成人后能够做到非常轻松自然地做

到这一点。我们能够不假思索地连续快速地分辨所处的环境，周边事物等。无论我们看到图片还是真实景象，都能够马上对其进行判断并打上标签，有时候这种行为就是下意识的。这种识别技术主要基于人们的先验知识与环境，而这些是我们的机器所无法拥有知识的。



## Inputs and Outputs

当我们的电脑看到一副图片时，机器只是看到一个由像素值组成的矩阵，比如说 $32*32*3$ ，其中32表示其分辨率或图像大小，3表示RGB三原色。为了把问题阐述清楚，我们这里定义一个JPG格式的彩色图像，大小为 $480*480$ ，那么表示的矩阵就是 $480*480*3$ 。矩阵里每一点取值范围0-255，表示为该点的像素强度（灰度值）。在我们人类进行图像识别的时候，这些像素点一点儿意义也没有，它们只是作为机器进行图像识别的输入而已。机器的输出呢，可以是一组概率值，这组概率值表明了当前的图像是某一类图像的可能性有多大。（比如0.8可能性是猫，0.15可能性是狗，0.05可能性是鸟等）

## What We Want the Computer to Do

我们知道了问题的输入和输出，现在要考虑该怎么解决它。我们想要计算机做的是分辨出所有给出的图中所具有的独特特征，例如说狗图或猫图的独特特征，这些特征是在某一分类图中一致，而跟其它类型图不同。这件事在我们自己的脑中同样也是自动完成的。例如，当我们看一幅狗图时，我们能够根据图中物体的爪子或4条腿分辨其是小狗。类似地，计算机也可以通过寻找一些低等级特征，例如边缘或纹理等，并由此通过一系列卷积层来建立更抽象的概念，来实现分类识别。大体上这就是CNN所做的事。接下来让我们更具体地展开说。

## Biological Connection

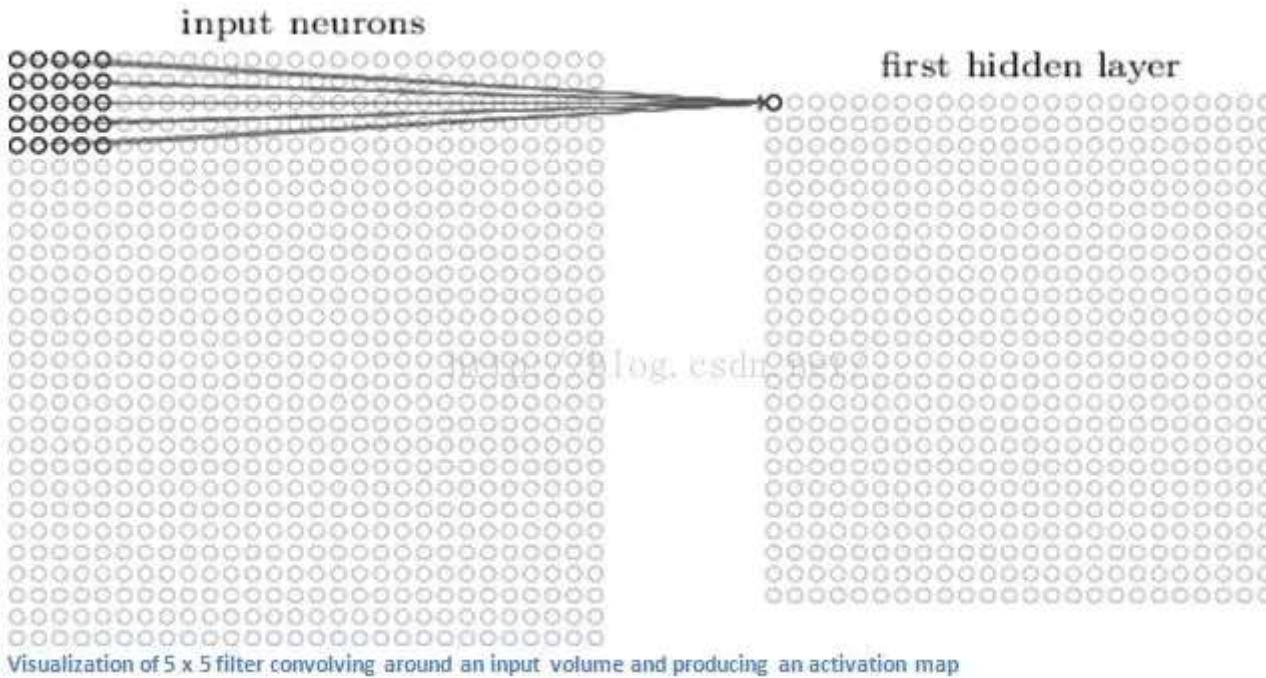
先让我们从一些背景知识开始吧。当你第一次听到卷积神经网络（CNN）这个短语时，也许你会想起生物神经科学领域的一些东西；可以说，某种程度上你是对的。CNN是从神经生物学中视觉皮层这个概念上获取了灵感。在大脑的视觉皮层中，有着许多由细胞组成的，对特定的视觉领域敏感的微小区域。这个理论由Hubel以及Wiesel在1962年通过实验获得了证实。（Video）在实验中，大脑内一些独立的神经元细胞仅对给实验者展示的某些特征放电或有反应，例如特定方向的边缘。例如，在展示垂直边缘、水平边缘或对角线边时，一些神经元开始放电。Hubel和Wiesel发现，这些神经元聚集并被组合成为纵列或柱形的结构，从而产生视觉。在一个系统整体中，不同的部分负责不同的任务（例如视觉皮层的神经元细胞负责感知视觉），这样的构造，正是CNN的基础。

## Structure

回到细节里来。针对CNN的具体行为，一个更加细化的视角是，用户将图像经过一系列卷积、非线性、池化（下采样）、以及全连通层后，获得了输出结果。就像之前说的，这种输出结果也许是一个用于描述该图像的分类结果或是一个分类的可能性。现在的难点在于，如何理解每一层都做了些什么。那么，让我们进入这个最重要的部分。

## First Layer – Math Part

CNN的第一层通常都是卷积层。你必须记住的第一件事应当是卷积层(conv layer)的输入是什么。就像之前说的，输入是一个 $32 \times 32 \times 3$ 的像素数列。要解释这个卷积层，最好的办法就是想象一下下面场景：你举着手电筒将光束打在一幅图像的左上角。我们假定这个光束覆盖的范围是 $5 \times 5$ 。那么现在，想象光束逐渐滑动平移，经过整幅图像。在机器学习的语言里，这个手电筒就被称作滤波器filter（或神经元neuron，或内核kernel），而它照过的区域叫做感知区(receptive field)。这个滤波器也可以表示为一个数组（数组中的数字可称为加权weight或参数parameter）。一个重要的点在于，滤波器的深度必须跟输入图像深度一致（才能保证计算不出错）。那么，这个滤波器的维度就变成了： $5 \times 5 \times 3$ 。现在，以这个滤波器最初的位置为例，它应当处于图像的左上角。随着滤波器的平移（或称卷积convolving），经过整幅图像。它将自身数组中的值与图像对应位置的像素值进行相乘（即点乘），将点乘结果加起来（数学上一共有75次乘加），就有了一个数。记住，这个数只代表滤波器在图像左上角初始位置时的卷积值。现在，我们一边移动滤波器一边重复这个计算（顺序是从左到右，然后从上到下。下一步是往右移一个像素）。这样，图像上每个单独的像素位置都会产生一个滤波（卷积）后的数值。在遍历整个图像后，你会得到一个 $28 \times 28 \times 1$ 的数列，我们称之为激活图activation map或特征图feature map。 $28$ 的原因是 $32 - 5 + 1 = 28$ 。这样一共就有784个值。



(注：上图使用的图片，是从这本特别棒的书中引用的 "Neural Networks and Deep Learning" by Michael Nielsen. 强烈推荐它)

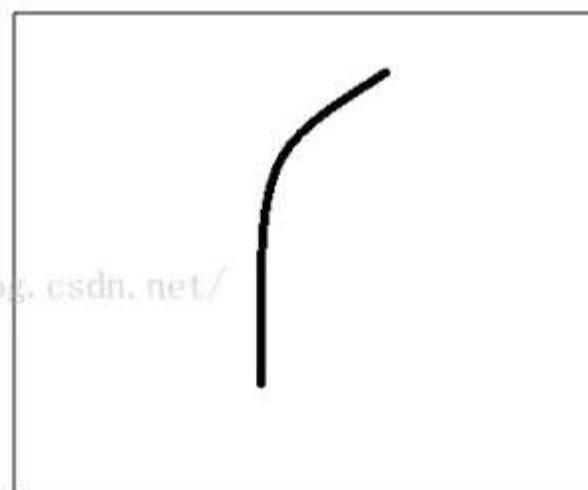
如果现在我们有两个 $5 \times 5 \times 3$ 的滤波器，而不是一个，那么输出结果就会变成 $28 \times 28 \times 2$ ，随着使用越来越多的滤波器，我们能够获取到越来越多的空间维度信息。从数学的角度，这就是卷积层所做的事。

## First Layer – High Level Perspective

让我们讨论一下，从更高的层面上，该如何看待这个卷积层所做的事。事实上，这些滤波器可以看做是特征识别器。这里的特征，指的是那些直线边缘、颜色、纹理等。考虑所有图像中都共通的一些最简单的特征。让我们把第一个滤波器改成大小为 $7 \times 7 \times 3$ 的纹理识别器。（在这个章节，为了降低复杂度，我们忽略滤波器与图像的深度，仅考虑其顶层的灰度数列及其计算）。作为一个纹理识别器，它的像素结构将会在纹理形状的对应区域内具有较高的数值。（记住，所有的滤波器都只是数值和数值的组合）

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

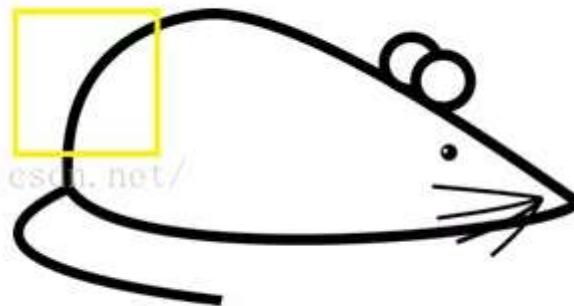


Visualization of a curve detector filter

现在，让我们回到数学上。当我们把滤波器放到图像左上角时，它开始进行 $7 \times 7$ 范围的内积计算。这里我们用给出的特征提取滤波器举一个例子，见下图。



Original image



Visualization of the filter on the image

记住，我们做的事是矩阵对应的像素值相乘再相加。



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

\*

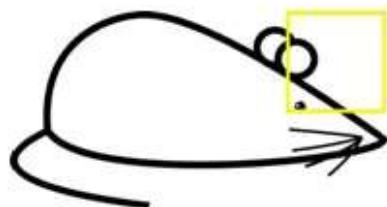
0	0	0	0	0	0	30	0
0	0	0	0	30	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	30	0	0	0	0
0	0	0	0	0	0	0	0

Pixel representation of filter

$$\text{Multiplication and Summation} = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 \text{ (A large number!)}$$

基本上，在输入图像中，如果当前形状跟我们的滤波器大体相像，卷积后的计算结果将会是一个很大的值！那么让我们看看如

如果移动滤波器到其它位置会发生什么。



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

\*

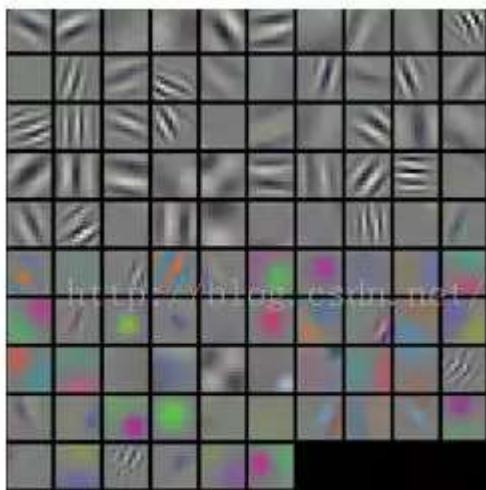
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

计算结果很小！这是因为滤波器和当前形状完全不同。记住，这个卷积层的输出是一个激活图层（activation map）。那么，针对这个单边缘滤波器卷积的简单例子来说，它的激活图层（activation map）将会展示出当前图像中最有可能是边缘特征的区域。在上图中，左上区域的卷积结果为6600，这个大数值表明图中对应位置有这样的边缘特征使得滤波器被激活了。而右上区域的值为0，表示当前区域没有任何能够激活滤波器的特征。（或更简单地说，图中该区域没有对应的图像特征）。记住，这只是一个滤波器。这只是一个检测竖直向右偏线状特征的滤波器。我们可以再添加其它滤波器用于检测竖直左偏或垂直特征等。滤波器越多，激活图层activation map的深度就越深，对于输入图像我们就能够获取到更多的信息。

声明：文中所述的滤波器是为了卷积计算上的简化而定义的。下图给出了一些用于第一卷积层的实际滤波器示例，但主要参数（原理）还是类似的。第一层的滤波器通过对输入图像的卷积和“激活”（或计算极值点），来寻找输入图像中特定的特征。



Visualizations of filters

(注：上图来自于斯坦福Andrey Karpathy以及Justin Johnson的CS 231N course。推荐给需要深入理解CNN的朋友们)

## Going Deeper Through the Network

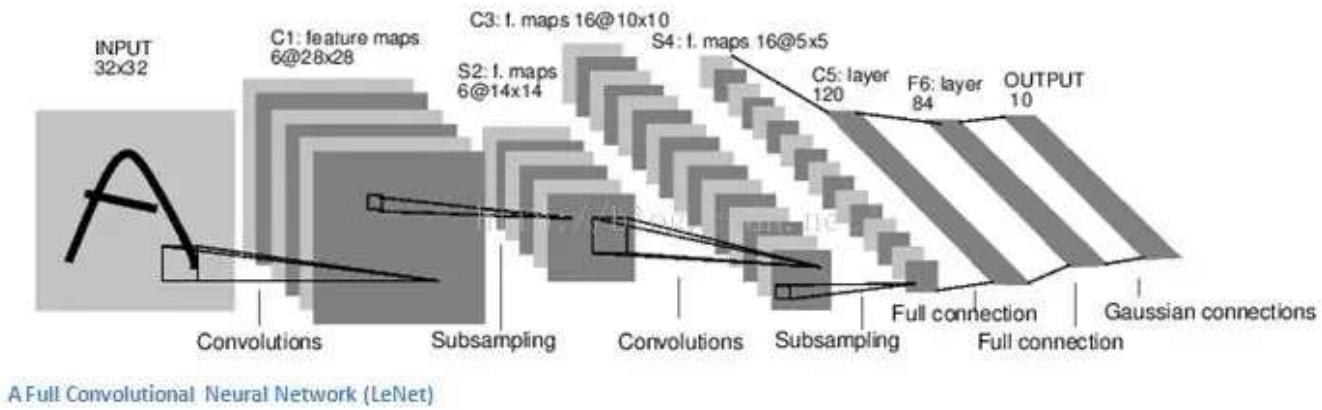
在传统的CNN结构中，在这些卷积层之间还夹杂有其它类型的层。我强烈建议大家去钻研并理解它们的功能和效果，这里仅给出一个大致的介绍。这些层提供了非线性特性nonlinearities与维度保留特性preservation of dimension用于提高整个网络结构的鲁棒性以及控制过拟合（control overfitting）。一个经典的CNN结构像下图所示：

Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool -> Fully Connected

然而，最后一层是非常重要的一层，我们稍后会介绍。现在让我们回过头来看看到目前已经学了哪些东西。我们讨论了第一卷积层的滤波器的设计与检测方法。它们用来检测边缘、纹理这类低阶特征。正如大家想象的，要想预测图像中的事物，我们需要一个网络结构用于识别像手掌、爪子、耳朵这样的高阶特征。然后让我们想一下第一卷积层的输出应当是什么：一个 $28 \times 28 \times 3$ 的数列结构（假设我们用的是三个 $5 \times 5 \times 3$ 的滤波器）。当我们进入第二个卷积层的时候，第一卷积层的输出就变成了第二卷积层的输入。这就比较抽象，比较难以想象了。要知道，当我们讨论第一层时，输入就是未经处理的原始图像；但是，当我们讨论第二卷积层时，输入已经变成了第一层的卷积结果，也就是一个激活图层activation map。这个输入的每一层基本上描述了某些低阶特征在原始图像中出现的位置。现在，当你把它输入给第二层，也就是再使用一系列滤波器对其卷积时，输出就应当是表示更高阶特征的激活图层，例如说半圆（结合了一个曲线和直线边缘）、方形（结合了一些直线边缘）等。当你将数据继续经过更多的卷积层时，你就会得到更加高阶特征的激活图层。在整个网络的最后，你也许会用一些滤波器用以激活图像中的手写特征handwriting、粉红色物体pink object等。如果你想知道更多关于滤波器可视化的信息，Matt Zeiler与Rob Fergus在他们的ConvNets模型与其研究论文research paper中有精彩的阐述。同样，在YouTube上，Jason Yosinski提供了一个非常棒的讲解视频video。另外一件有趣的事是，当你的网络结构越深，滤波器的感知区域范围就越大，这意味着它们在处理时，能够将对应原始图像中更大区域的信息都概括在内（能够对更大的像素空间进行反应和感知）。

## Fully Connected Layer

检测出高阶特征后，我们可以锦上添花地在网络结构的最后添加一个全连通层**fully connected layer**。全连通层输入一个数列（无论这个输入是来自卷积层**conv**、线性整流**ReLU**层还是池化层**pool**），输出一个**N**维向量，**N**是由程序指定的分类类别数量。例如，对于一个数字分类程序，**N**就应该取10（0~9共10个数字）。这个**N**维向量中的每一个数字表示被分到该类的几率。例如，还是针对数字分类程序的分类结果为[0 .1 .1 .75 0 0 0 0 0 .05]，这就表示这个输入的图像为1的概率有10%，为2的概率10%，为3的概率75%，为9的概率5%。（注：输出的表示方法不止这一种，这里只是展示了这种小数概率表示的**softmax**算法）。全连通层的工作原理是，根据之前其它层的输出（表示为高阶特征的激活图层），检测哪些特征与特定的类别相匹配。例如，程序若判定图像是一只狗，那么激活图层中表示狗的高阶特征，像是爪子、四条腿等特征将会具有很高的数值。再比如，程序若判定图像是一只鸟，那么激活图层中表示鸟的高阶特征，像是翅膀、鸟喙等就会具有很高的数值。基本上，全连通层的工作就是寻找与特定类别匹配的高阶特征，并设定相应的权重值。这样当我们把之前层的结果通过权重进行计算后，就能够正确估计其属于某一类别的可能性了。



## Training (AKA:What Makes this Stuff Work)

现在这个部分是我特意在之前的讨论中未提及的，很可能是关于神经网络的最重要的部分。也许你在之前的阅读时会有很多问题。比如，第一卷积层的滤波器怎么知道该如何查找边缘和纹理？而全连通层又是怎样知道该如何查找激活图层的？每一层的滤波器都是怎样设定其值的？计算机在解决这些问题时主要依靠一个训练过程**training process**，称作反向传播算法**backpropagation**（BP算法）。

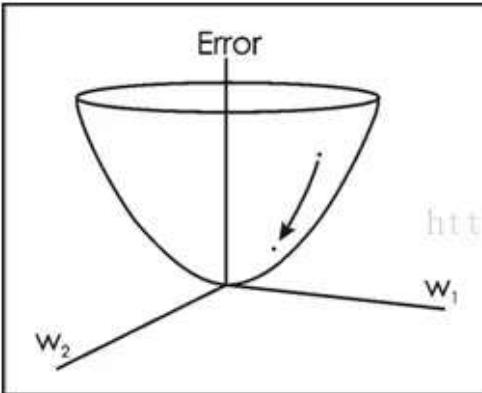
在讨论反向传播之前，我们必须先回过头来讲讲让神经网络（生物学意义上的）工作起来需要什么。当我们出生时，我们的大脑是一片空白的。我们并不知道什么是猫、

狗、鸟儿。类似地，在CNN运行之前，滤波器的数值和权重等参数都是随机的。滤波器并不知道该如何寻找边缘和纹理。高阶层的滤波器同样也不知道该如何寻找爪子、鸟喙这类高阶特征。当我们逐渐长大，父母以及老师会利用照片、影像与之对应的标签来教会我们这些动物的辨别方法。这种利用标签的图片同样也是CNN所使用的训练过程。在深入研究之前，我们先简单地打个比方：我们有一个训练集，其中有几千张图片，绘制着猫、狗、鸟儿，同时每张图片上都有一个标明这是哪种动物的标签**label**。好的，现在回到我们的反向传播算法。

反向传播算法可以分为4个部分：前向传播**forward pass**，损失函数**loss function**，反向传播**backward pass**，权重更新**weight update**。在前向传播的过程中，你将一个训练数据（ $32 \times 32 \times 3$ 的图像数列，表示一个数字）输入整个网络。由于所有权重或滤波器系数都是随机生成的，输出很可能是类似这样的结果[1.1.1.1.1.1.1.1.1.1]，基本上这样的输出难以判断是一个有效分类的。系统以目前的参数/权值是很难寻找低阶特征，并且也很难以此进行可靠的分类的。于是进入算法的损失函数阶段。记得我们刚才使用的输入是训练数据，那么除了图像数列外还附带标签。比方说这个输入的训练数据是3，那么标签就应当是[0 0 0 1 0 0 0 0 0]。通过其输出结果和标签的比较，就能够计算损失函数了。损失函数的定义方法很多，这里就用一个常见的均方差算法**MSE**(mean squared error)，如下式

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

这里我们使用变量**L**保存这个损失函数的结果。可以想象，在训练刚开始时这个值会相当的大。现在让我们用直觉去想想，我们想要找到一个“点”使得预测的标签（也就是网络的输出）和训练标签相同（这意味着我们的网络预测的结果是正确的）。那么我们需要做的就是最小化损失函数**L**。将其具象化，其实就是一个演算**calculus**优化的问题，我们想要找到哪些输入（在我们的系统中就是权值）更直接地影响到损失**L**，或者说误差。



One way of visualizing this idea of minimizing the loss is to consider a 3-D graph where the weights of the neural net (there are obviously more than 2 weights, but let's go for simplicity) are the independent variables and the dependent variable is the loss. The task of minimizing the loss involves trying to adjust the weights so that the loss decreases. In visual terms, we want to get to the lowest point in our bowl shaped object. To do this, we have to take a derivative of the loss (visual terms: calculate the slope in every direction) with respect to the weights.

上图表示，针对这个问题具象化的一个方法是将其放在由一个误差（损失相关）坐标轴、两个权值坐标轴组成的3D图中。（当然大部分神经网络的权值显然大于2，这只是为了简化）。想要最小化误差值就需要不断调节权值 $w_1$ 和 $w_2$ 。用图上的话说，我们要找到这个碗状曲面的z轴最低点。要做到这一点，就需求出误差在各个方向（权值）上的导数（斜率）。

在数学上，这等价于针对某一层的权重 $W$ ，计算导数 $dL/dW$ 。那么现在，我们想要做的就是对网络进行一次反向传播backward pass，以判断哪些权重对损失 $L$ 有较大的影响，并且调整这些权重以减小损失。一旦导数计算出来之后，我们就可以进行最后一个步骤：更新权重。我们将所有滤波器的权重进行更新使得它们随着其梯度方向的变化而改变（change in the direction of the gradient梯度下降法？）

$$w = w_i - \eta \frac{dL}{dW}$$

$w = \text{Weight}$
$w_i = \text{Initial Weight}$
$\eta = \text{Learning Rate}$

学习速率learning rate是由程序员所指定的参数。高学习速率表明在进行权重更新的跨度大，这样模型就能更快地汇集到最优的权重集上面来。但是，过高的学习速率有可能导致跨度过大而难以精确地收敛到最优解上。



这整套流程：前向传递forward pass、损失函数loss function、后向传递backward pass、参数更新parameter update合起来称为一代epoch。针对每个训练图像，程序会重复进行多代epoch的训练。每次迭代结束后，我们都希望能够让网络通过良好地训练，正确地调节其每层的权重。

## Testing

最后，要想验证我们的CNN是否运作正确，我们需要准备另外一套数据和标签（不能与训练集重复！），将其送入CNN进行测试。我们会比较输出结果与金标准ground truth是否一致，以验证网络是否正确。

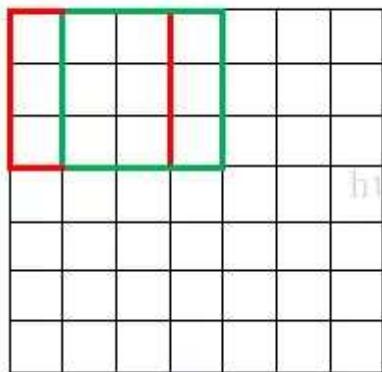
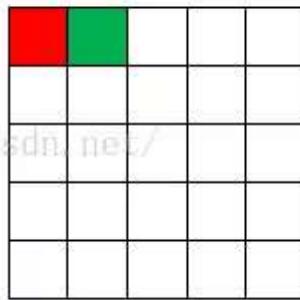
## How Companies Use CNNs

数据，数据，数据。这神奇的单词，那些有着大量数据的公司才能在激烈的市场竞争中获得潜在的先机。在网络训练的过程中，你的训练数据量越大，能够进行的训练迭代就越多，能够进行的权重调整也越多，调优的结果就越好。Facebook与Instagram公司拥有旗下数亿规模用户的照片；Pinterest公司的网站上也有500亿张拼趣图片pin供其使用；Google坐拥海量的用户搜索数据；而Amazon则每天都在接收着几百万项用户产品购买的信息。现在你能明白隐藏在这些数据之下的魔力了吧。

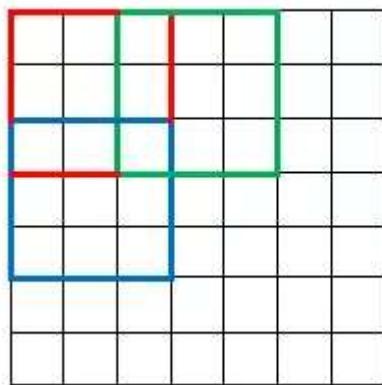
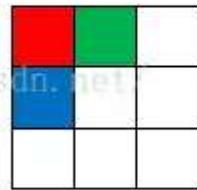
## Stride and Padding

好的，让我们回忆一下我们的老朋友卷积层。还记得那些滤波器、感知区、卷积吗？现在让我们介绍两个卷积层中的重要参数。步长stride和填充padding。

步长**stride**主要用于控制滤波器在输入图像上的卷积行为。如下图例子所示，滤波器在图像上的卷积是每次卷积计算后平移一定距离再次计算。这个距离就是通过步长**stride**来进行控制的。在这个例子中，步长**stride**设为1。通常来说，步长**stride**参数的选择需要考虑到这个步数能够被整除。让我们看一下图中的这个例子：设输入为 $7 \times 7$ 的图像，滤波器尺寸为 $3 \times 3$ （同样忽略了第三维），设定步长为1，那么输出就应该是下图右边的 $5 \times 5$ 图像。

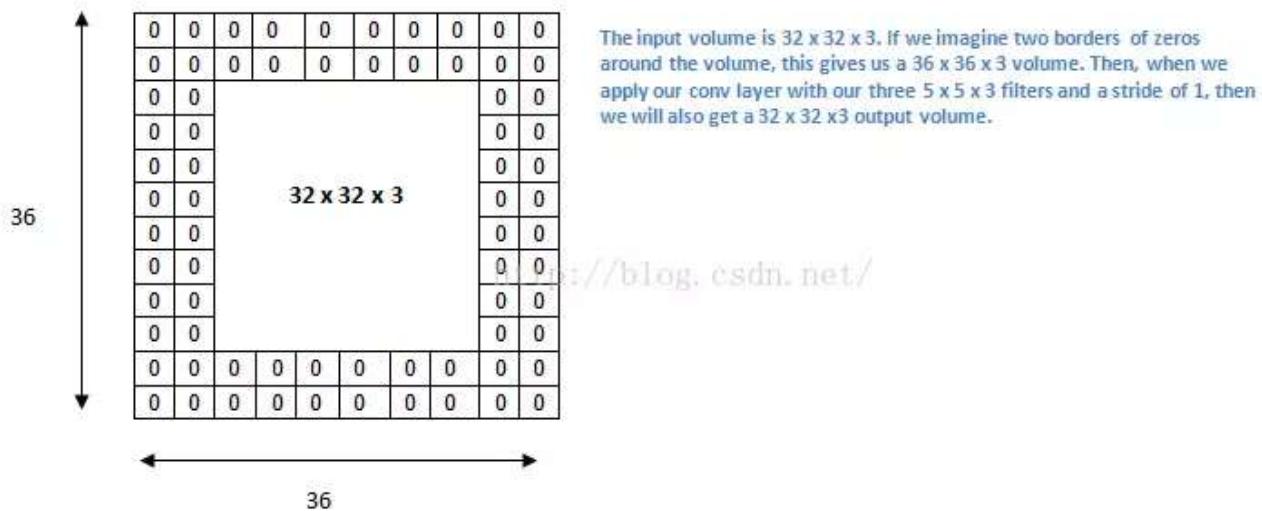
**7 x 7 Input Volume****5 x 5 Output Volume**

还是老样子，不是吗？想想如果我们把步长**stride**设为2的情形。

**7 x 7 Input Volume****3 x 3 Output Volume**

那么，感知区每次就会平移两个单位（像素），那么输出自然就相应的缩小为 $3 \times 3$ 了。如果我们想要把步长**stride**设为3，那么在进行卷积时就会有空间上的问题了，同时也难以保证感知区在平移时是否还处于图像内（译者：原因是步数不能整除？）。通常，在设计卷积层时，步长**stride**越大，那么感知区的重叠就越小，同样地输出的图像也越小。

现在，让我们看一下填充padding参数。先想象一下这样的场景。当你使用3个 $5 \times 5 \times 3$ 的滤波器对一个 $32 \times 32 \times 3$ 的输入图像滤波，那么输出图像应当是 $28 \times 28 \times 3$ 。注意空间尺寸变小了。然后我们把这个图像再次放入卷积层，会发现尺寸更小了。事实上，在整个网络的开始几层，我们并不希望尺寸缩水得这么快。我们希望能够尽量保留下这些输入图像的信息，才能够比较好地提取底层特征。也就是说，我们想要保证输出仍然是 $32 \times 32 \times 3$ 。为了实现这个目的，我们在卷积层上引入了尺寸为2的零填充zero padding。它能够在输入图像的边界上形成一层宽度为2，值为0的边界。那么输入图像就变成了 $36 \times 36 \times 3$ 。如下图所示。



若设定步长stride为1，同时零填充zero padding的尺寸为：

$$\text{Zero Padding} = \frac{(K - 1)}{2}$$

式中K为滤波器尺寸。那么输入和输出图像的尺寸就会永远保持一致。

卷积层的输出图像的尺寸可以通过下面公式得出：

$$O = \frac{(W - K + 2P)}{S} + 1$$

式中O是输出宽/高，W是输入宽/高，K为滤波器尺寸，P为填充padding尺寸，S为步长stride。

## Choosing Hyperparameters

我们该怎样确定在一个系统中，卷积层的数量、滤波器的尺寸、或是步长**stride**、填充**padding**这些参数呢？这些可不是细枝末节的问题，另外这些问题也没有一个放之四海而皆准的答案。这是因为系统参数的设定主要取决于你的数据类型。数据的大小、图像的复杂度、图像处理的目的和方式等，都会随着处理目的的不同而发生变化。选择超参数的正确做法是，在检视数据集时通过抽象概念将数据/图像以合适的尺度正确组合起来。

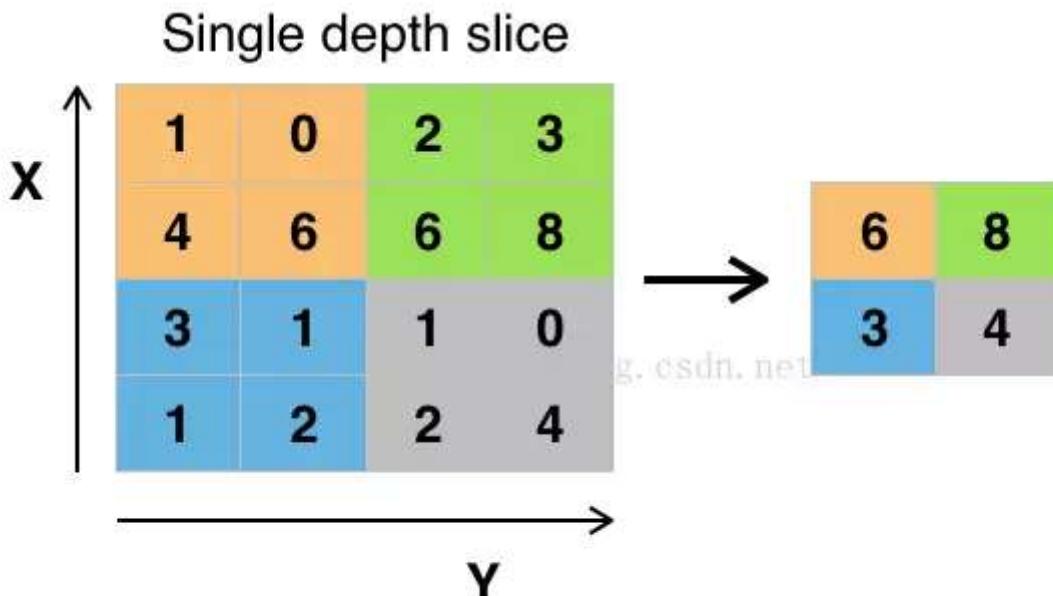
## ReLU (Rectified Linear Units) Layers

按照惯例，每个卷积层之后都紧跟一个非线性层（或激活层**activation layer**）。由于系统在卷积层的计算主要为线性操作（像素/元素级的乘法和加法计算），因此这层的主要目的在于为系统引入非线性性。在过去，研究人员主要利用双曲正切**tanh**或S函数**sigmoid**作为非线性函数进行处理，后来大家发现线性整流层**ReLU**效果更佳，由于其计算效率能够大大加快整个系统训练的速度。同时它能减轻梯度消失问题**vanishing gradient problem**，这个问题主要出现在训练时，由于梯度呈指数下降而导致的底层训练十分缓慢的问题。线性整流层**ReLU**将输入图像的所有元素通过这样的一个整流函数： $f(x) = \max(0, x)$ 。从术语的角度上，本层将所有负激活**negative activations**都改为0，由此提高了模型与整个系统的非线性特性，而不会影响到卷积层的感知区。

相关信息可参考Geoffrey Hinton大神（也就是深度学习之父）的论文Paper。  
<http://www.cs.toronto.edu/~fritz/absps/reluICML.pdf>

## Pooling Layers

通过线性整流层**ReLU**处理后，研究人员可能会在其后添加一个池化**pooling**层，也叫下采样**downsampling**层。这同样有许多可选的方法，其中最常用的是最大池化**maxpooling**方法。这种方法定义一个最大值滤波器（通常 $2*2$ ）以及对应长度的步长**stride**，然后对输入图像进行滤波输出滤波器经过的每个子区域的最大值。如下图。



Example of Maxpool with a 2x2 filter and a stride of 2

池化层也可以用均值或L2范数（L2-norm，欧氏距离）来计算。池化层的直观意义在于，从输入图像中获取到某个特征（会有较高的激活值activation value），它与其它特征的相对位置比其自身的绝对位置更加重要。从图中不难看出，池化层大大降低了输入图像的空间维度（宽和高，不包括深度）。这就达到了两个目的。首先，参数/权重的个数降低为原来的1/4，计算量相应减少了。第二，它能够防止过拟合overfitting。过拟合表示为由于过度的调优导致模型过于执着满足样本的特征（可能有些是局部特征）而失去了泛用性，从而难以识别其它数据。过拟合overfitting的一个征兆就是，模型在训练集上能获得99%或100%的精确度但在测试数据上仅有50%。

## Dropout Layers

在神经网络中，丢包层dropout layer有一个非常特殊的函数。在上一节，我们介绍了过拟合overfitting问题，当训练完成后，由于对训练集过度调优导致的系统对新数据的识别效果不好。实际上丢包层在前向传播的过程中，故意地把一些随机的激活特征activations设为0值，这样就把它们简单地“丢包”了。那么，这样一个看似不必要、违反常理的简单操作有什么好处呢？事实上，它强行地保证了整个系统的冗余性。也就是说，系统需要满足这样的情形：当输入数据的一部分激活特征activations缺失时，系统也能够将其正确识别。丢包层dropout能够保证系统不会跟训练集过于相似，从而从一定程度上解决过拟合overfitting问题。需要注意的是，丢包层dropout仅用于训练环节。

## Network in Network Layers

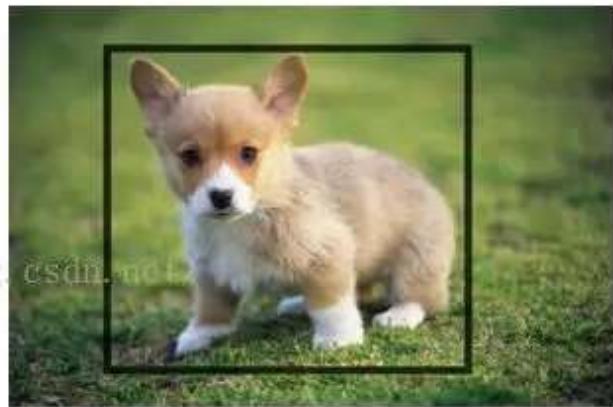
网络中的网络层Network in Network (NIN) 指的是一个拥有 $1*1$ 大小滤波器的卷积层。初看你会奇怪网络层有什么用，它投影出来的感知区区域比待投影对象还大。但是，请记住滤波器还有一个维度：深度N。因此这是一个 $1*1*N$ 的卷积操作。实际上，它进行了一次N-D逐像素乘法，其中N是输入数据的深度。（大概是用子网络结构代替原先的线性卷积）

## Classification, Localization, Detection, Segmentation

在我们之前的图像分类任务那个例子中，系统将输入的图像处理后输出其分类标签（或类别可能性数组）。但是，如果任务变成目标识别，那么除了需要进行分类外，还需要用一个框划出目标的具体位置。如图。



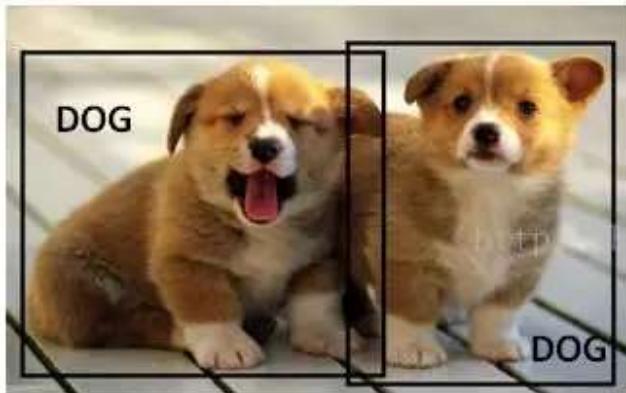
Object Classification is the task of identifying that picture is a dog.



Object Localization involves the class label as well as a bounding box to show where the object is located.

另外在一些目标检测的任务中，图中所有目标的位置都需要确定。因此，就需要输出多个定位框和类别标签。

更进一步地，在目标分割任务中，除开分类标签和定位之外，还需要系统能够将目标的边缘轮廓识别并描绘出来，如下图。



Object Detection involves localization of multiple objects (doesn't have to be the same class).



Object Segmentation involves the class label as well as an outline of the object in interest.

## Transfer Learning

许多人认为只有拥有了google级别的数据量才能训练好一个模型，这是一个常见的对深度学习的误解。诚然，数据量对于整个系统的搭建是至关重要的，然而我们还有迁移学习transfer learning的方法来缓解数据量的需求。迁移学习transfer learning是把已训练的模型（指其他研究人员利用其它大型数据集训练的模型及其参数/权重），利用自己的数据集进行参数调整的过程。方法思路在于，将已训练模型当作一个特征提取器feature extractor。我们需要把模型的最后一层（全连通层？）移除，替换为自己的分类器（取决于我们自己需要解决的问题空间）。然后，冻结其它层的权重/参数，并开展数据训练。冻结是为了防止在梯度下降/优化时参数的改变。

让我们看看它是怎么工作的。假设我们提到的已训练模型是通过ImageNet进行训练的（ImageNet是一个拥有1400万数据量以及1000多个分类标签的数据库）。如同之前描述的，模型的底层通常是一些基本特征的提取，例如边缘和纹理等，这些特征信息在大部分图像处理问题都是必要的（除非你的问题空间和数据集非常独特）。当进行迁移学习transfer learning时，我们就可以使用这些具有共性的权重并保持它们的值，而不是设定一个随机的初始值，这样我们的训练就可以专注于更重要的层（通常是更高层）。当然，如果你的数据集跟已训练集有很大不同，那么你就需要冻结更少层的参数，而进行更多的训练。

## Data Augmentation Techniques

我们现在已经对卷积网络中数据的重要性非常清楚了，那么就谈谈该怎样通过一些简单的转换，进一步扩大我们的现有数据集。就像我们之前所说，当电脑输入一组数据（图像）时，输入的形式通常为灰度值数组。假设我们把整幅图像平移1个像素，从我们的角度来说，这种变化是几乎难以察觉的。但是，对电脑来说，若要保证不影响结果，这种变化就相当明显了。这种在保持结果不变的前提下修改训练数据表现形式的方法称

为数据扩容方法**data augmentation techniques**。这是人工扩展数据集的方法。一些常用的扩容方法包括灰度扩展**grayscale**s，水平翻转**horizontal flips**，垂直翻转**vertical flips**，随机裁剪**random crops**，颜色抖动**color jitters**，翻译（转化？）**translations**，旋转**rotations**等等。应用这些变换方法，我们可以很轻易地成倍扩展我们的数据集。

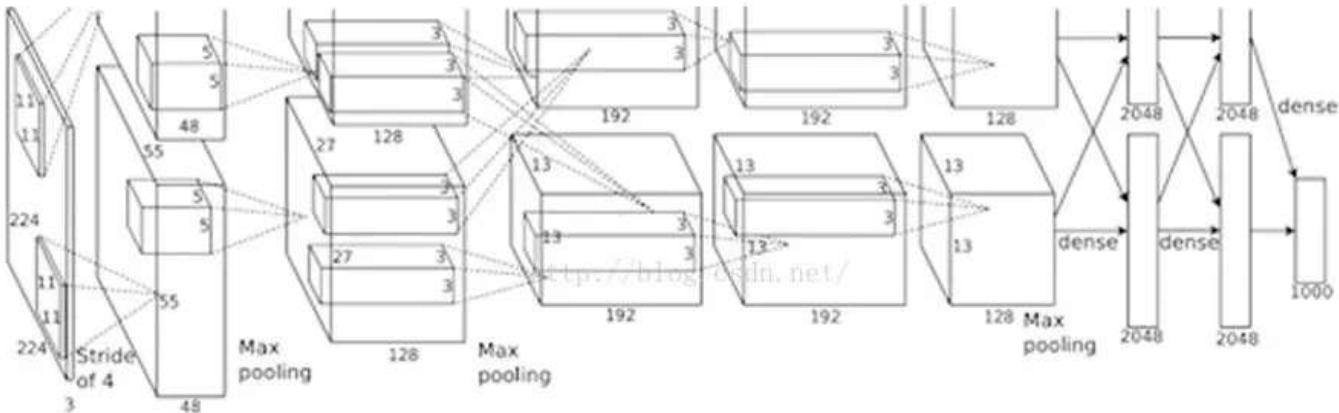
我们将介绍几篇重要的公开发表的论文，讨论它们为何重要。前一半的论文（**AlexNet**到**ResNet**）将主要涉及整体系统架构的发展和演变，后一半论文将主要集中的一些有趣的子领域应用上。

## AlexNet (2012)

<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

这篇文章算是深度学习的起源（尽管有些学者认为Yann LeCun在1998年的论文 paper 才是真正的起源）。文章标题是“ImageNet Classification with Deep Convolutional Networks”，已经获得了共6184次引用，并被广泛认为是业内最具深远影响的一篇。Alex Krizhevsky, Ilya Sutskever, 以及Geoffrey Hinton三人创造了一个“大规模、有深度的卷积神经网络”，并用它赢得了2012年度ILSVRC挑战（ImageNet Large-Scale Visual Recognition Challenge）。ILSVRC作为机器视觉领域的奥林匹克，每年都吸引来自全世界的研究小组，他们拿出浑身解数相互竞争，用自己组开发的机器视觉模型/算法解决图像分类、定位、检测等问题。2012年，当CNN第一次登上这个舞台，在前五测试错误率top 5 test error rate项目上达到15.4%的好成绩。（前五错误Top 5 error指的是当输入一幅图像时，模型的预测结果可能性前五中都没有正确答案）。排在它后面的成绩是26.2%，说明CNN相对其它方法具有令人震惊的优势，这在机器视觉领域引起了巨大的震动。可以说，从那时起CNN就变成了业内家喻户晓的名字。

这篇文章主要讨论了一种网络架构的实现（我们称为**AlexNet**）。相比现在的架构，文中所讨论的布局结构相对简单，主要包括5个卷积层、最大池化层、丢包**dropout**层，以及3个全连通层。该结构用于针对拥有1000个可能的图像类别进行分类。



AlexNet architecture [May look weird because there are two different "streams". This is because the training process was so computationally expensive that they had to split the training onto 2 GPUs]

图中文字：AlexNet架构采用两个不同的数据“流”使得它看起来比较奇怪。这是因为训练过程的计算量极大因此需要将步骤分割以应用两块GPU并行计算。

### 文中要点

- 利用ImageNet数据库进行网络训练，库中包含22000种类的1500万标签数据。
- 利用线性整流层ReLU的非线性函数。（利用线性整流层ReLU后，运行速度比传统双曲正切函数快了几倍）
- 利用了数据扩容方法data augmentation，包括图像变换、水平反射、块提取patch extractions等方法；
- 为解决训练集过拟合问题而引入了丢包层dropout layer。
- 使用批量随机梯度下降法batch stochastic gradient descent进行训练，为动量momentum和权重衰退weight decay设定限定值。
- 使用两块GTX 580 GPU训练了5~6天。

### 本文重要性

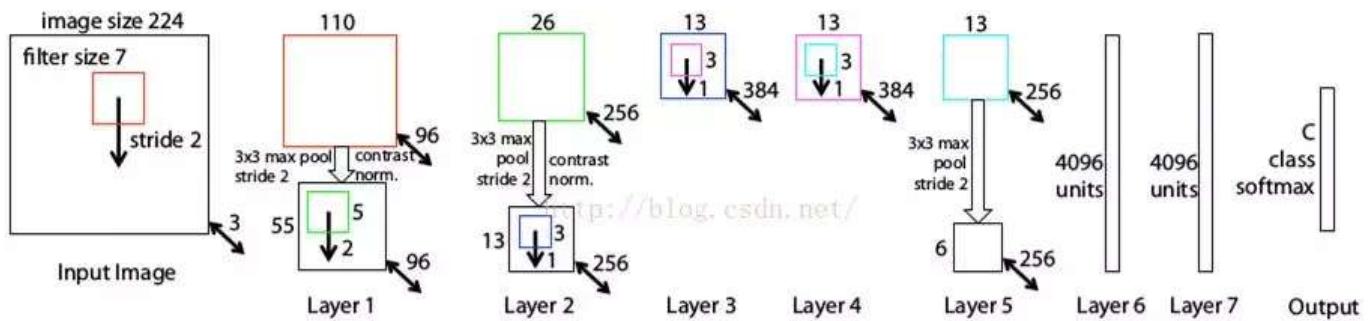
本文的方法是机器视觉领域的深度学习和CNN应用的开山怪。它的建模方法在ImageNet数据训练这一历史性的难题上有着很好的表现。它提出的许多技术目前还在使用，例如数据扩容方法以及丢包dropout层。这篇文章真真切切地用它在竞赛中的突破性表现给业内展示了CNN的巨大优势。

## ZF Net (2013)

<http://arxiv.org/pdf/1311.2901v3.pdf>

AlexNet在2012年大出风头之后，2013年随即出现了大量的CNN模型。当年的的ILSVRC比赛胜者是来自纽约大学NYU的Matthew Zeiler以及Rob Fergus设计的模型，叫做ZF Net。它达到了11.2%的错误率。ZF Net的架构不仅对之前的AlexNet进行了进一步的优化，而且引入了一些新的关键技术用于性能改进。另外一点，文章作者用了很长的篇幅讲解了隐藏在卷积网络ConvNet之下的直观含义以及该如何正确地将滤波器及其权重系数可视化。

本文标题是“Visualizing and Understanding Convolutional Neural Networks”。在文章开头，Zeiler和Fergus提出CNN的复兴主要依靠的是大规模训练集以及GPU带来的计算能力飞跃。他们指出，目前短板在于研究人员对模型的内部运行机理知之甚少，若是不能解决这个问题，针对模型的改进就只能依靠试错。“development of better models is reduced to trial and error”。虽然相较3年前，我们现在对模型有了进一步的了解；然而这依然是一个重要问题。本文的主要贡献是一个改进型AlexNet的细节及其可视化特征图层feature map的表现方式。



ZF Net Architecture

## 文章要点

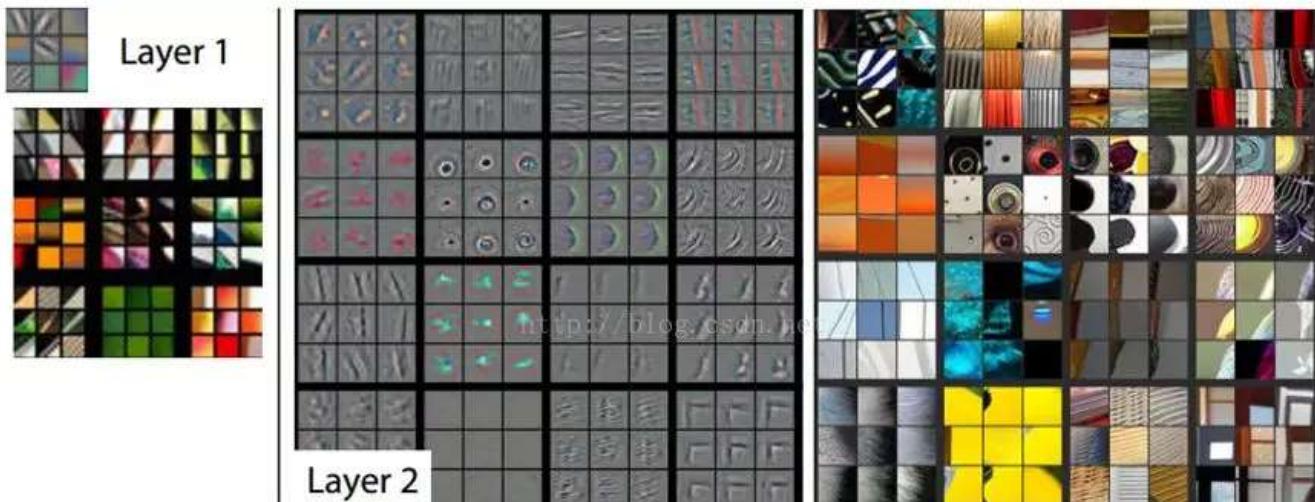
- 除了一些微小改进外，模型架构与AlexNet非常相似
- AlexNet训练集规模为1500万张图像，ZF Net仅为130万张
- 相比AlexNet在第一层使用的11\*11滤波器，ZF Net使用7\*7的滤波器及较小步长。如此改进的深层次原因在于，在第一卷积层中使用较小尺寸的滤波器有助于保留输入数据的原始像素信息。事实证明，在第一卷积层中使用11\*11滤波器会忽略大量相关信息。
- 随着网络层数深入，使用的滤波器数量同样增加。
- 激活方法 activation function 使用了线性整流层 ReLUs，误差函数 error function（疑为作者笔误，应该是损失函数loss function）使用了交叉熵损失函数 cross-entropy loss，训练方法使用了批量随机梯度下降法 batch stochastic gradient descent。

- 用1块GTX580 GPU训练了12天
- 发明一种卷积网络可视化技术，名为解卷积网络Deconvolutional Network，有助于检查不同激活特征以及它们与输入空间的关系。命名为“解卷积网络” "deconvnet"是因为它把特征投影为可见的像素点，这跟卷积层把像素投影为特征的过程是刚好相反的。

## DeConvNet

解卷积的基本工作原理是，针对训练后的CNN网络中的每一层，都附加一个解卷积层 deconvnet用于将感知区回溯path back到图像像素。在CNN的工作流程总，我们把一幅图像输入给CNN，一层一层地计算其激活值activations，这是前向传递。现在，假设我们想要检查第四卷积层中针对某个特征的激活值，我们把这层对应的特征图层中的这个激活值保存起来，并把本层中其它激活值设为0，随后将这个特征图层作为解卷积网络的输入。这个解卷积网络与原先的CNN有相同的滤波器设置。输入的特征图层通过一系列的反池化（最大池化求反），整流（反整流？），以及滤波（反滤波？），随后到达输入端。

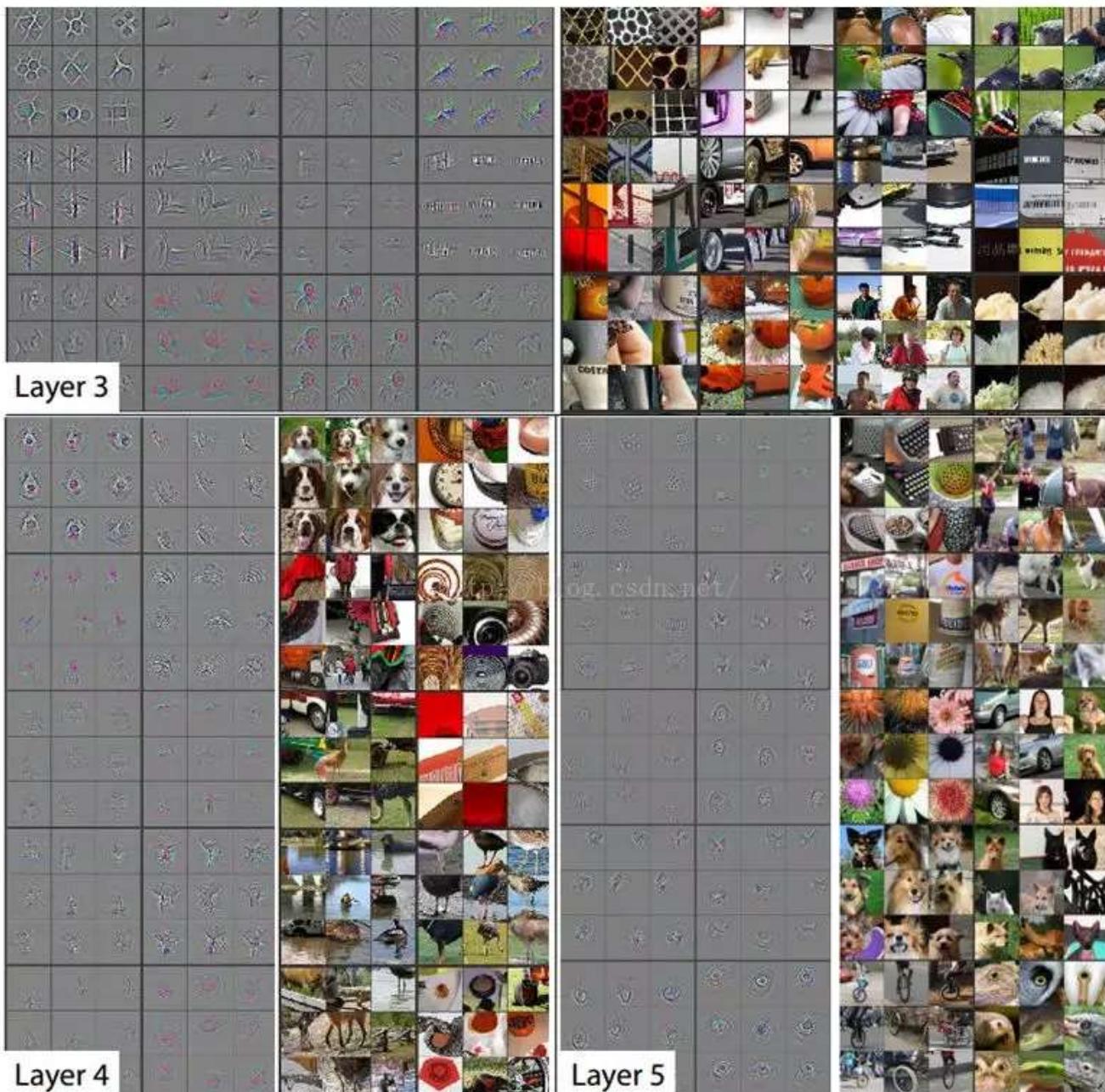
隐藏在这整套流程之下的原因是，我们想要知道当给定某个特征图层时，什么样的图像结构能够激活它。下图给出了第一和第二层的解卷积层的可视化结果。



Visualizations of Layer 1 and 2. Each layer illustrates 2 pictures, one which shows the filters themselves and one that shows what part of the image are most strongly activated by the given filter. For example, in the space labeled Layer 2, we have representations of the 16 different filters (on the left)

图中文字：第一层与第二层的可视化表示。每层都表示为两幅图片：其一表示为滤波器；另一表示为输入原始图像中的一部分结构，在给定的滤波器和卷积层之下，这些结构能够激发最强的激活信号。图中第二解卷积层的左图，展示了16个不同的滤波器。（跟第一层9个组合起来）

图中卷积网络ConvNet的第一层通常是由一些用于检测简单边缘、颜色等信息的低阶特征检测子组成。从图中也可以看出，第二层则是更多的圆形特征。让我们看看下图3，4，5层的情形。



Visualizations of Layers 3, 4, and 5

图中这几层展示出更进一步的高阶特征，例如狗的脸部特征或是花朵的特征等。也许你还记得，在第一卷积层后，我们应用了一个池化层pooling layer用于图像下采样（例如，将 $32*32*3$ 的图像转换为 $16*16*3$ ）。它带来的效果是第二层的滤波器视野（检测范围scope）更宽了。想要获取更多有关解卷积网络以及这篇论文的信息，请参考Zeiler的发表视频presenting。  
<https://www.youtube.com/watch?v=ghEmQSxT6tw>

## 本文重要性

ZF Net不仅仅是2013年度竞赛的冠军，而且它为CNN提供了更加直观的展示能力，同时提供了更多提升性能的技巧。这种网络可视化的方法有助于研究人员理解CNN的内部工作原理及其网络

架构。迷人的解卷积网络可视化以及阻塞实验让这篇文章成了我的最爱。

## VGG Net (2014)

<http://arxiv.org/pdf/1409.1556v6.pdf>

简单但有深度。2014年度ILSVRC其中一个模型最好地利用了这两个特点达到了7.3%的错误率（但并不是当年的冠军）。牛津大学的Karen Simonyan以及Andrew Zisserman两位创造了一个19层的CNN，网络中仅使用了 $3 \times 3$ 尺寸的滤波器，步长stride和填充padding都为1，池化层使用 $2 \times 2$ 的最大池化函数，步长为2。是不是很简单？

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 ( $224 \times 224$ RGB image)					
conv3-64	conv3-64 LRN	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	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 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

The 6 different architectures of VGG Net. Configuration D produced the best results

## 文章要点

- 仅使用 $3 \times 3$ 滤波器，这与之前的AlexNet的首层 $11 \times 11$ 滤波器、ZF Net的 $7 \times 7$ 滤波器都大不相同。作者所阐述的理由是，两个 $3 \times 3$ 的卷积层结合起来能够生成一个有效的 $5 \times 5$ 感知区。因此使用小尺寸滤波器既能保持与大尺寸相同的功能又保证了小尺寸的优势。优势其中之一就是参数量的减少，另一个优势在于，针对两个卷积网络我们可以使用多一个线性整流层ReLU。（ReLU越多，越能降低系统线性性？）
- 3个 $3 \times 3$ 卷积层并排起来相当于一个有效的 $7 \times 7$ 感知区。

- 输入图像的空间尺寸随着层数增加而减少（因为通过每层的卷积或是池化操作），其深度反而随着滤波器越来越多而增加。
- 一个有趣的现象是，每个最大池化层之后，滤波器数量都翻倍，这进一步说明了数据的空间尺寸减少但深度增加。
- 模型不仅对图像分类有效，同样能很好地应用在本地化任务中（翻译任务）。作者在文章中进行了一系列的回归分析说明此事。（论文第10页很好地说明了）
- 用Caffe工具箱进行建模
- 在训练中使用了尺寸抖动技术scale jittering进行数据扩容data augmentation
- 每卷积层后紧跟一个线性整流层ReLU并使用批量梯度下降法batch gradient descent进行训练
- 用4块Nvidia Titan Black GPU进行训练2~3周。

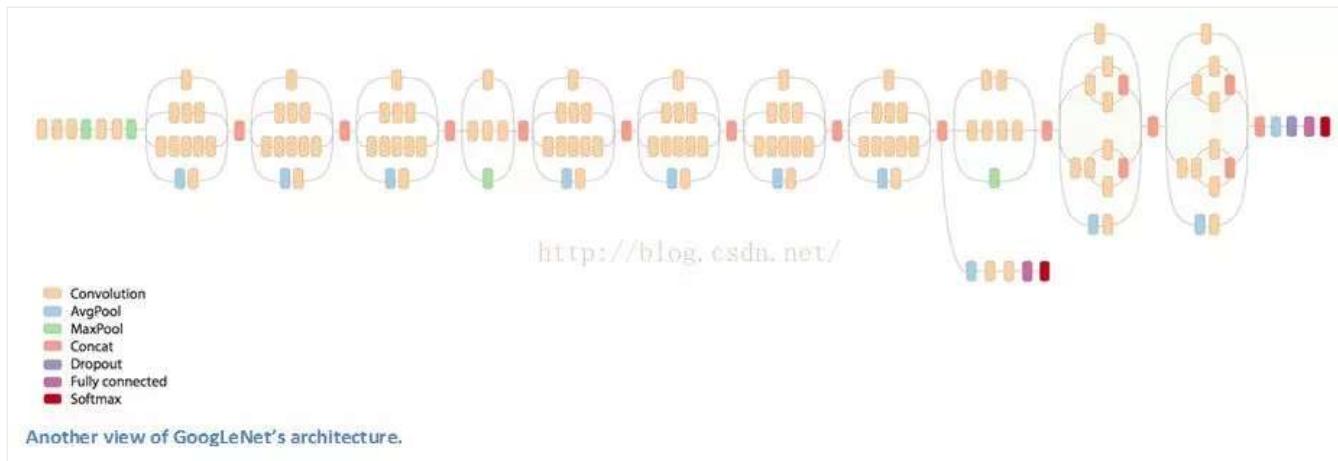
## 本文重要性

VGG Net是我印象中影响最为深远的一篇文章，原因在于它强调了卷积网络中的深度，CNN必须保证拥有一个足够深的网络结构才能体现它在处理视觉数据的层次性。保持深度、保持简单。

## GoogLeNet (2015)

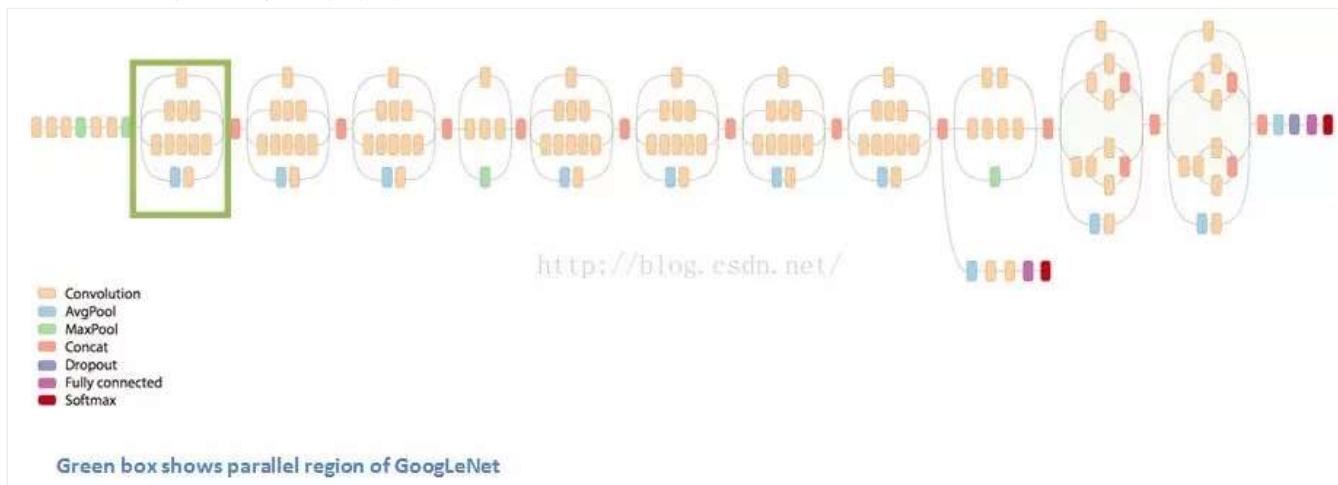
[http://www.cv-foundation.org/openaccess/content\\_cvpr\\_2015/papers/Szegedy\\_Going\\_Deeper\\_With\\_2015\\_CVPR\\_paper.pdf](http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf)

还记得刚才我们所说的简单法则吗？然而Google在自己的架构Inception Module里把这个原则抛到了九霄云外。GoogLeNet是一个22层的CNN，它以6.7%的错误率赢得了2014年度ILSVRC的冠军。据我所知，这是第一个跟传统方法，也就是卷积层与池化层简单叠加以形成序列结构的方法不同的一种CNN的新架构。文章作者强调，他们的新模型也特别重视内存与计算量的使用（这是之前我们没有提到的：多层堆积以及大量滤波器的使用会耗费很多计算与存储资源，同样也会提升过拟合的几率）。

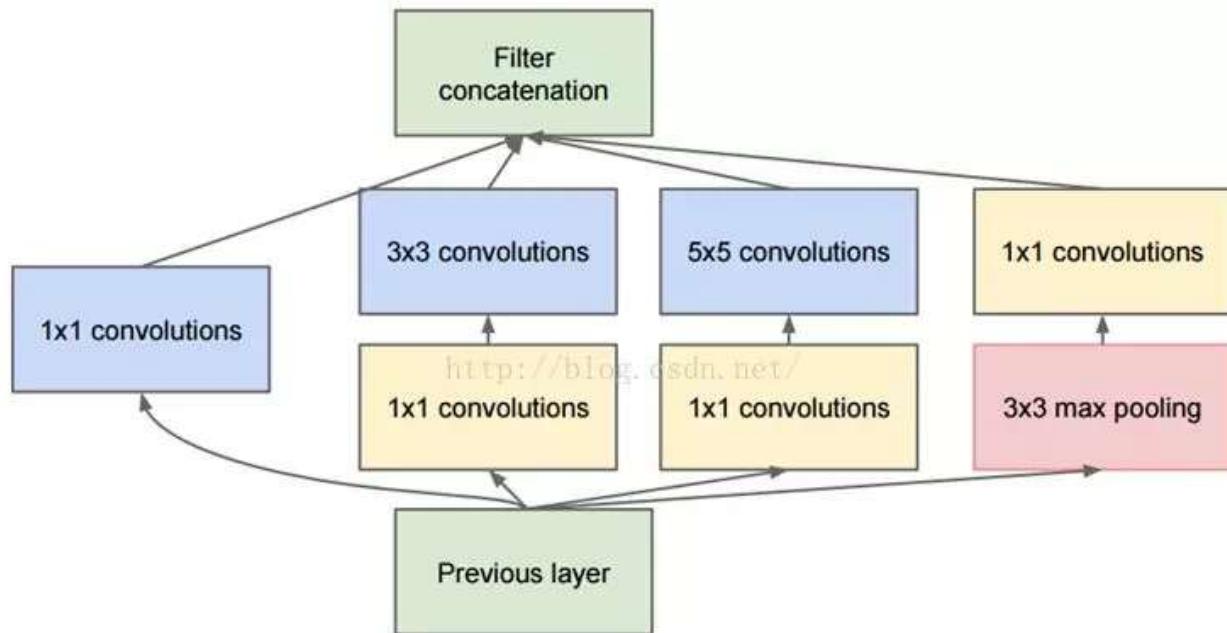


## Inception Module

当我们第一眼看到GoogLeNet的架构时，会发现并不是像之前架构那样，所有流程都是顺序执行的。系统的许多部分是并行执行的。

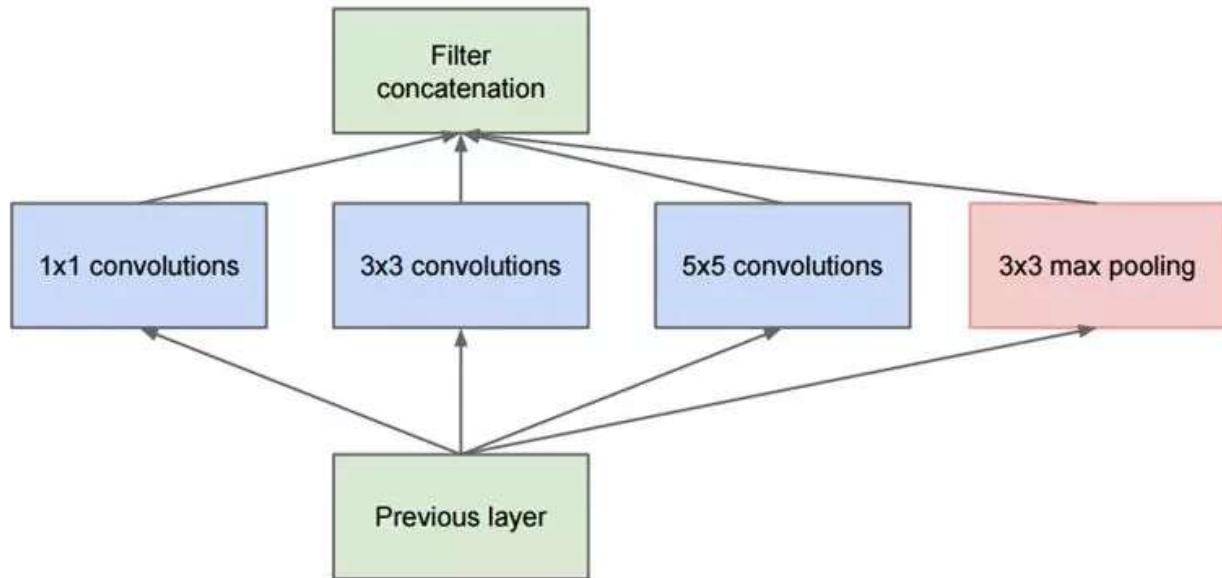


下图就称为Inception module。让我们仔细研究一下它的构成。



Full Inception module

底部的绿色模块就是我们的输入，而顶部绿色模块是输出（把它顺时针转90°就可以跟之前的GoogLeNet架构图对应起来了）。基本上在传统卷积网络ConvNet中，你需要选择当前输入是用于执行池化pooling操作还是卷积操作（同样要选择滤波器尺寸）。然而在Inception module里，你可以让它们同时跑一遍。实际上，这正是作者一开始设计时的“天真”想法。



Naïve idea of an Inception module

为什么说它“天真”呢？答案是它会导致太多的输出。最终我们会得到一个具有极为巨大深度的数组。为了解决这个问题，作者在3\*3以及5\*5卷积层之前，采用了一个1\*1卷积操作。1\*1卷积（或称为网络中的网络架构NIN）提供了降维的效果。打个比方，假设你有一个100\*100\*60的输入图像（尺寸无关紧要，可以看成是其中某一层的输出）。将其进行20个1\*1的卷积操作，则会将尺寸变为100\*100\*20（不太明白了，估计20个滤波器尺寸应当是1\*1\*60）。这意味着之后3\*3以

及 $5 \times 5$ 卷积所要面对的图像数据变少了。这就像是一个“特征池化pooling of features”的操作，就跟在一般模型中的最大池化maxpooling层中降低空间尺寸的操作类似，在这里我们降低了数据的深度。另外一点在于这些滤波器后跟线性整流层ReLU。

你可能会问“这架构有啥用？”事实上，在这个由网络中的网络NIN层，中型滤波器，大型滤波器以及池化操作组成的架构中，NIN层能够从输入数据中提取出极为精细的图像细节信息， $5 \times 5$ 滤波器能够覆盖较大的感知区与提取其内部的信息。同样，池化操作流程能够帮你减少空间尺寸，处理过拟合问题。另外，每个卷积层都配有一个线性整流层ReLU，它能够降低你的系统线性度。基本来说，这个架构能够以一个可接受的计算量处理这些复杂操作。此外，文章中还提到了一个更高层次的用途，是有关稀疏及稠密连接sparsity and dense connections的。

## 文章要点

- 模型里共使用9个Inception module模块，深度总计100层！
- 并没有使用全连通层，而是用一个平均池化层average pool取而代之，将 $7 \times 7 \times 1024$ 的数据降低为 $1 \times 1 \times 1024$ 。这个构造大大降低了参数个数。
- 比AlexNet的参数少了12倍。
- 在测试时，使用相同输入图像的多个副本multiple crops (?) 作为系统输入，将其结果进行归一化指数函数softmax平均操作后得到其最终结果。
- 在模型中引入了区域卷积网络R-CNN的概念（之后会提到）
- Inception module现在不断更新中（现在版本6,7）
- “用一些高端GPU训练1周即可”

## 本文重要性

GoogLeNet是最先提出CNN模型中的非序列叠加模型这一概念的。文章作者通过介绍Inception module模块，为业内展示了一个独具创造性的，有着较高运行效率的模型。本文为随后出现的一些精彩的模型奠定了基石。

## Microsoft ResNet (2015)

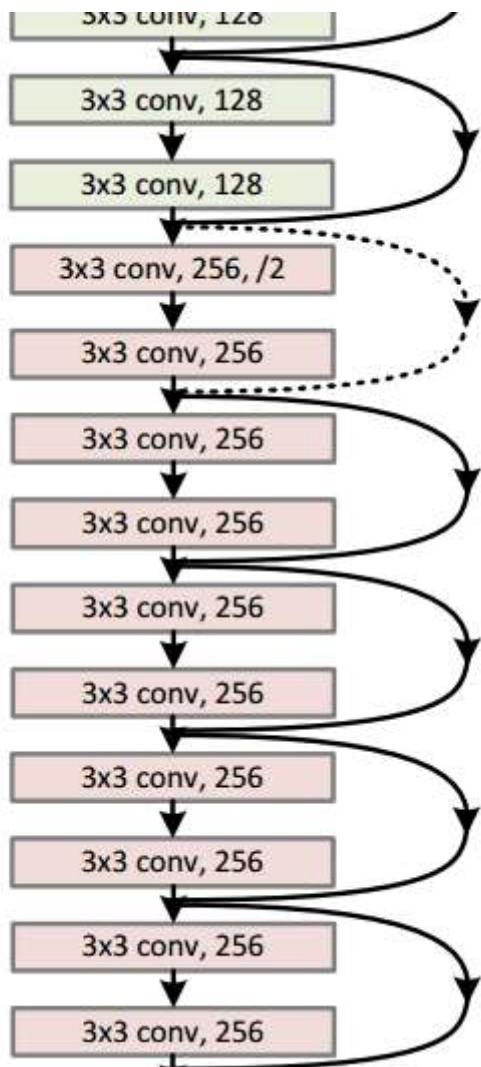
<https://arxiv.org/pdf/1512.03385v1.pdf>

想象一个很深的CNN架构，把它的层数翻两番，它的深度可能还比不上ResNet，它是微软亚研MRA在2015年提出的架构。ResNet是一个拥有152层网络架构的新秀，它集分类、检测与翻译功能于一身。除开层数破了纪录，ResNet自身的表现也破了ILSVRC2015的记录，达到了不可思议

的3.6%（通常人类也只能达到5~10%的出错率，跟专业领域和技能相关。请参考Andrey Karpathy以自身经验撰写的，有关ImageNet挑战中人类与卷积网络ConvNet竞赛的雄文great post

<http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/>

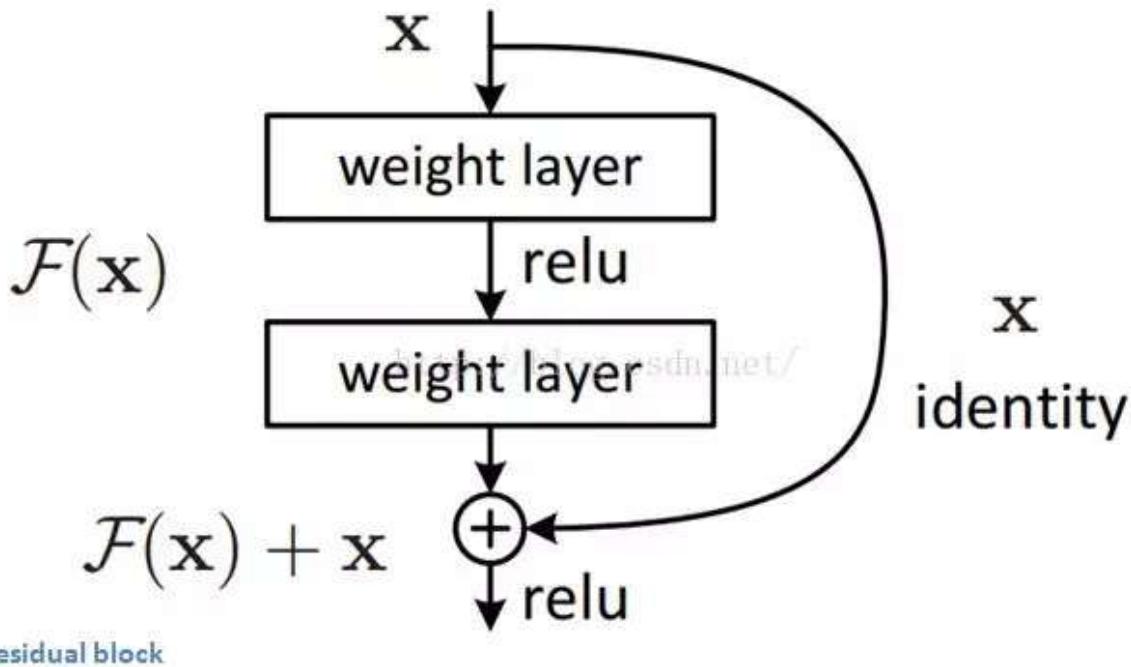
）。



## Residual Block

文章中提出的残差区块residual block概念，其设计思路是这样的：当我们的输入 $x$ 通过卷积-线性整流-卷积系列操作后，产生的结果设为 $F(x)$ ，将其与原始输入 $x$ 相加，就有 $H(x)=F(x)+x$ 。对比传统CNN，只有 $H(x)=F(x)$ 。而ResNet需要把卷积结果 $F(x)$ 与输入 $x$ 相加。下图的子模块表现了这样一个计算过程，它相当于对输入 $x$ 计算了一个微小变化"delta"，这样输出 $H(x)$ 就是 $x$ 与变化 $\delta$ 的叠加（在传统CNN中，输出 $F(x)$ 完全是一个全新的表达，它并不包含输入 $x$ 的信息）。文章作者认

为，“这种残差映射关系residual mapping比起之前的无关映射unreferenced mapping更加容易优化”。



残差区块的另外一个优势在于反向传播操作时，梯度信息流由于这些附加的计算，从而更加容易传播flow easily through the effective。

## 文章要点

- “极度深寒Ultra-deep” - Yann LeCun
- 152层...
- 一个有意思的特点是，最初两层处理后，输入图像的空间尺寸由224\*224压缩至56\*56
- 作者声明若在平层网络plain nets中随意增加层数会导致训练计算量以及错误率上升
- 研究团队曾尝试使用1202层网络架构，结果精确度反而降低了，推测原因是过拟合。
- 训练使用一个8GPU的机器，持续了2~3周

## 文章重要性

模型达到的3.6%错误率本身就极具说服力了。ResNet模型是目前最棒的CNN架构，同时是残差学习residual learning的一项重要创新。2012年以来，随着错误率逐年下降，我很怀疑在ILSVRC2016上是否能看到更好的成绩。我想我们也许已经到了一个瓶颈，仅依靠往模型中堆砌更多的卷积层已经难以获取算法性能上的提升了。就像之前的两年那样，今年的竞赛一定会有更具创造性的新型模型架构。2016.9.16，这是今年比赛结果揭晓之日。别忘了。

## Region Based CNNs (

R-CNN - 2013, <https://arxiv.org/pdf/1311.2524v5.pdf>

Fast R-CNN - 2015, <https://arxiv.org/pdf/1504.08083.pdf>

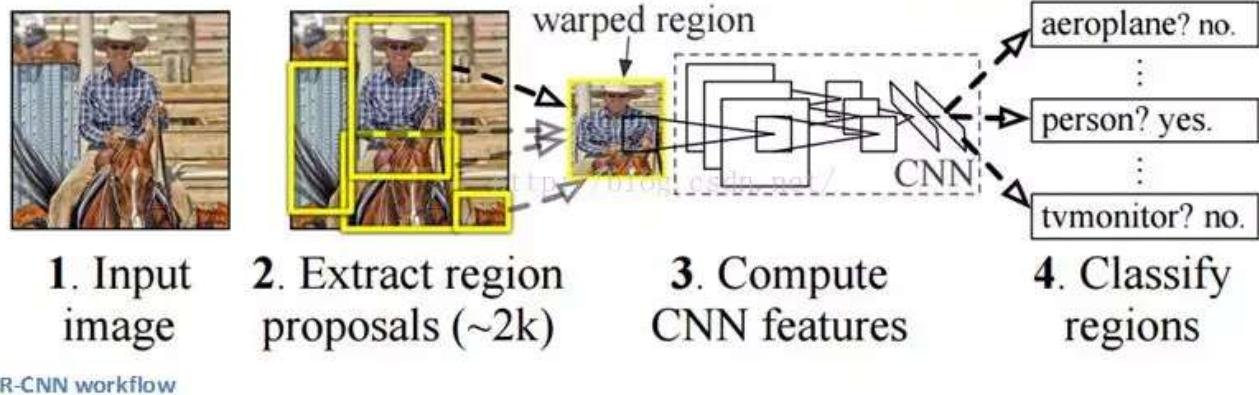
Faster R-CNN - 2015) <http://arxiv.org/pdf/1506.01497v3.pdf>

也许会有人认为比起之前所说的那些新架构，R-CNN才是最重要，对业内影响最大的CNN模型。UC Berkeley的Ross Girshick团队发明了这种在机器视觉领域有着深远影响的模型，其相关论文被引用超过了1600次。如同标题所说的，Fast R-CNN以及Faster R-CNN方法使我们的模型能够更好更快地解决机器视觉中的目标检测问题。

目标检测的主要目的是：给出一副图像，把其中所有物体都框起来。这个过程可以分为两个主要的部分：目标标定、分类。

作者提出，针对区域标定方法，任何类不可知区域检测法class agnostic region proposal method都是合适的。其中Selective Search方法特别适用于RCNN模型。Selective Search算法在运行的过程中会生成2000个不同的，有最大可能性标定图像中的目标的区域标定region proposals。获取到这些标定区域后，算法把它们“变形warped”转换为一幅图像并输入一个已训练好的CNN中（例如AlexNet），进行特征向量的提取。随后将这些向量作为一系列线性SVM分类器的输入进行分类。同样将这些向量输入给区域边界的回归分析器regressor，用于进一步精确获取目标的位置。

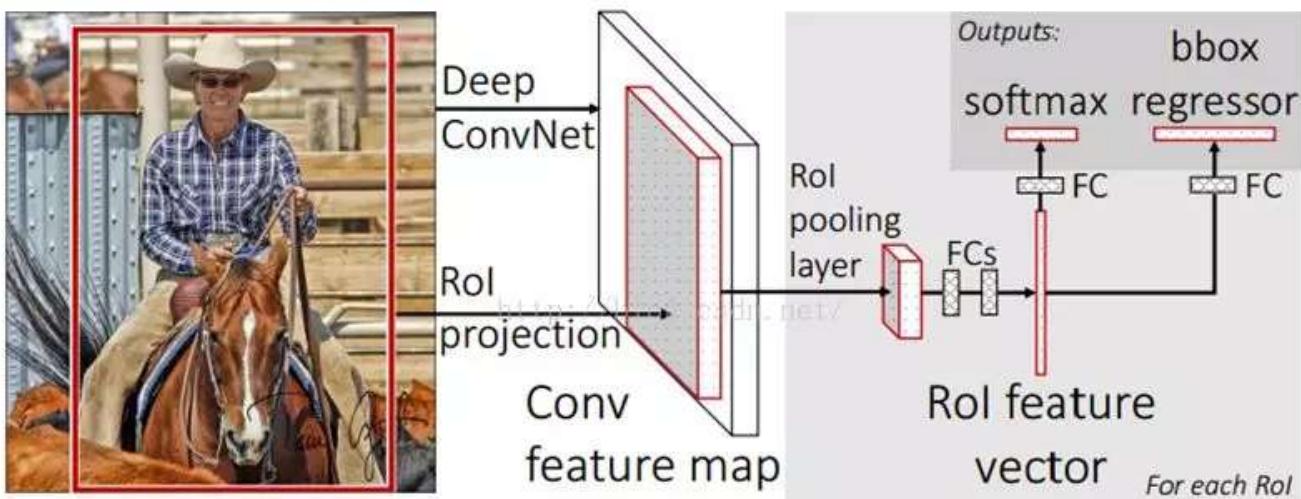
## R-CNN: Regions with CNN features



随后，模型采用一个非极大值抑制算法用于去除那些互相重叠的区域。

### Fast R-CNN

Fast R-CNN针对之前模型的改进主要集中在这3个方面的问题。多个阶段的训练（卷积网络ConvNet、SVM、区域边界回归分析）计算负载很大且十分耗时。Fast R-CNN通过优化流程与改变各生成标定区域的顺序，先计算卷积层，再将其结果用于多个不同的功能计算模块，以此解决速度的问题。在模型中，输入图像首先通过一个ConvNet，从其最后输出的特征图层中获取特征标定区域，最后将其同时输入全连通层、回归分析模块以及分类模块。（译者按：这段基本上为字面翻译，然而有许多不合常理的地方。从图中看出标定区域似乎是在ConvNet之前，跟文中所述矛盾；另外图中似乎应该有多个ROI区域，并行地进行ConvNet，输出结果再并行输入FC，regressor等）

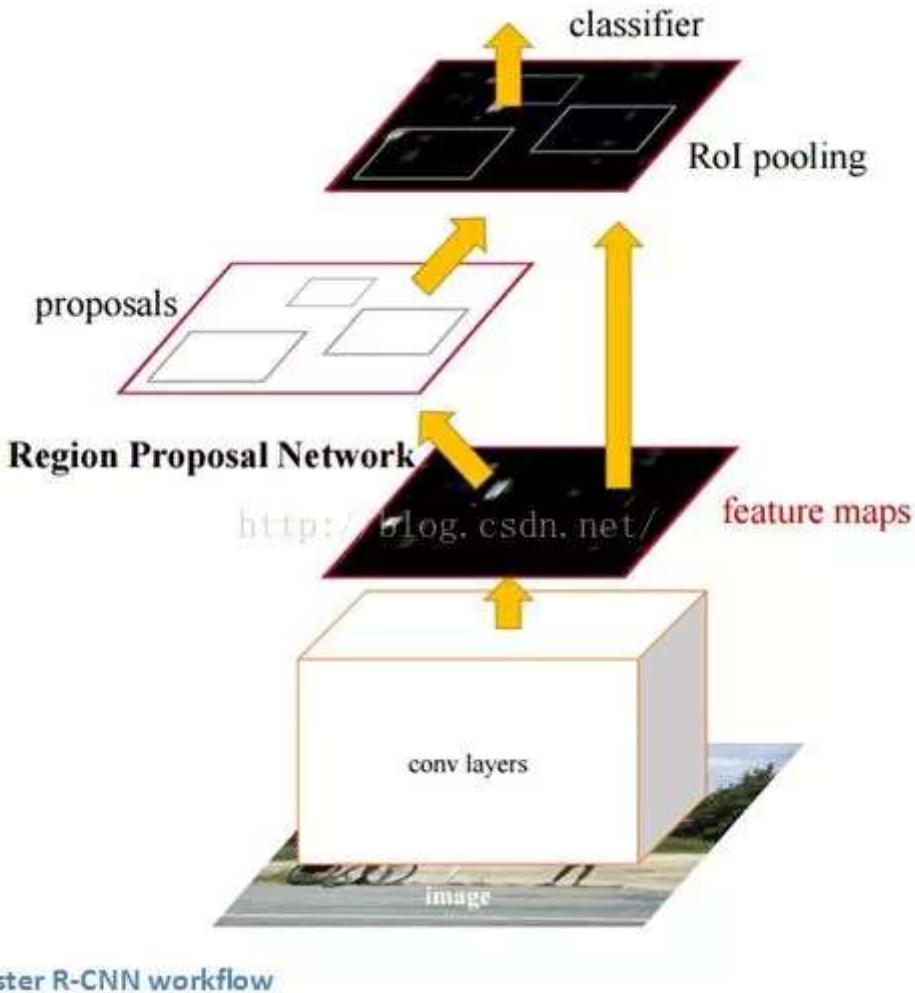


Fast R-CNN workflow

### Faster R-CNN

Faster R-CNN用于解决在R-CNN和Fast R-CNN中的一些复杂的训练流程。作者在最后一层卷积层后插入了一个区域标定网络region proposal network (RPN)。RPN能够从其输入的特征

图层中生成标定区域region proposals。之后流程则跟R-CNN一样（ROI池化、全连通、分类以及回归）



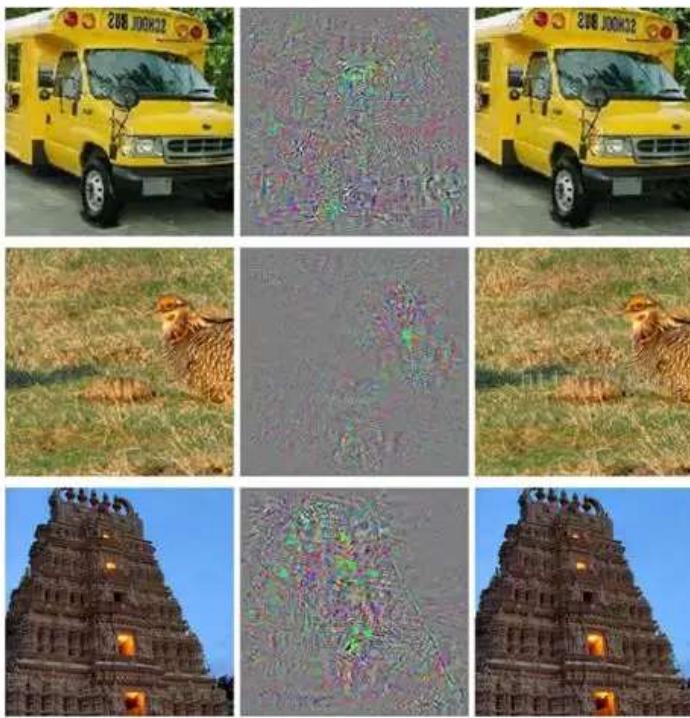
## 文章重要性

首先它能检测图像中的特定物体；更重要的是它能够找到这个物体在图像中的具体位置，这是机器学习的一个重要进步。目前，Faster R-CNN已经成为目标检测算法的标杆。

## Generative Adversarial Networks (2014)

<https://arxiv.org/pdf/1406.2661v1.pdf>

根据Yann LeCun的说法，这个网络架构可以说又是一个大进步。在介绍这篇文章之前，我们先谈谈对抗样本adversarial examples。例如，有一个经过ImageNet数据训练好的CNN，现在给一副图（如下图左）加一些扰动或微小修改（中，右），输入后导致预测错误率增加了许多。虽然图像看起来跟原来似乎是一样的，但是最终分类却与原先已经不同了。归纳起来，对抗样本就是那些故意愚弄并破坏卷积网络ConvNets结果的图像。



The images in the left most column are correctly classified examples. The middle column represents the distortion between the left and right images. The images in the right most column are predicted to be of the class ostrich! Even though the difference between the images on the left and right is imperceptible to humans, the ConvNet makes drastic errors in classification.

图中文字：左列图像为正确样本，中间一列表示左和右图之间的扰动，右列图像的大部分都被归类为鸵鸟ostrich。事实上，人眼几乎难以分辨左右图之间的差异，然而卷积网络ConvNet在分类时竟会产生如此夸张的错误。

对抗样本Adversarial examples 吓到了许多研究人员并马上成为议论的热点。现在让我们谈谈这个generative adversarial networks模型。这里有两个模型：产生模型generative model和判别模型discriminative model。判别模型discriminative model用于判断某幅图像是天然的（直接来自数据集里）还是人为制造的。产生模型generator则创造样本供给判别模型discriminator训练。这可以看成是一个零和zero-sum游戏或是最小最大minimax游戏。文章中用的类比是这样的，产生模型generative model就像是“一群造假币的”，而判别模型discriminative model则像是“抓造假币者的警察”。产生模型不停地试图欺骗判别模型而判别模型试图识破欺骗。随着模型的训练，二者的能力不断提升最后达到“赝品和正品已经完全分不清楚了”的程度。

## 论文重要性

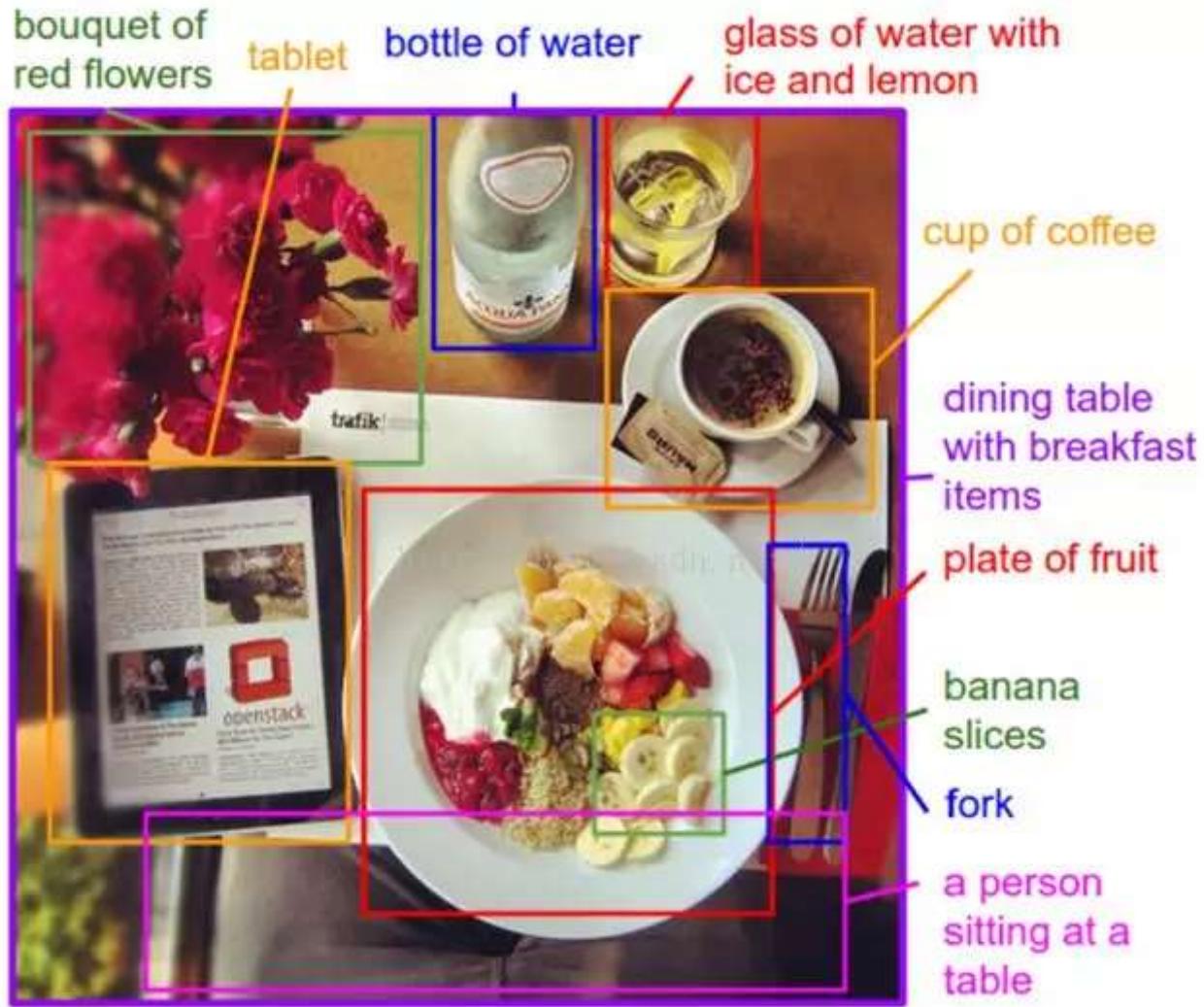
听起来这么的简单，那为什么我们要关注这个模型呢？就像Yann LeCun在Quora网站上的帖子 post <https://www.quora.com/What-are-some-recent-and-potentially-upcoming-breakthroughs-in-deep-learning>

所述，因为判别模型discriminator已经能够识别来自数据集中的真实图像以及人工伪造的图像，因此可以说其探悉了“数据的内在表达”。因此，这个模型可用作CNN中的特征提取器；另外你也可以用它来仿造一些以假乱真的图像。

## Generating Image Descriptions (2014)

<https://arxiv.org/pdf/1412.2306v2.pdf>

当你把CNN和RNN（循环神经网络）结合在一起会产生什么？抱歉，别想错了，你并不能得到R-CNN;-)；但确实能得到一个很不错的模型。Andrej Karpathy（我个人最喜欢的作者之一）和Fei-Fei Li所写的这篇文章就是着重于研究将CNN与双向RNN bidirectional RNN相结合生成用于描述图像区域的自然语言描述器。基本上这个模型通过输入一副图像，产生如下的输出：



Example output of the model

看起来非常不可思议。让我们看看它跟普通CNN有什么不同。在传统的模型中，针对训练数据中的每一张图片，都只有一个确定的标签与之对应。但本文所描述的模型则通过一个句子（或标题）与图像相关联。这种标签形式被称为弱标签，其语句中的成分与图像中的（未知）部分相关联。使用这样的训练集，让一个深度神经网络模型“推断语句成分与其描述的图像区域之间的潜在结合alignment关系（文中语）”；另外还有一个网络模型则将图像作为输入，生成其文字描述。现在让我们分别看看这两个部分：配对alignment与产生generation。

## Alignment Model

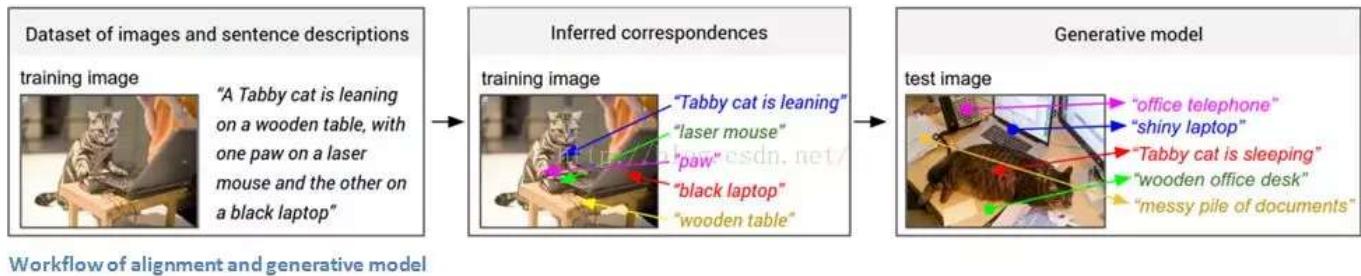
这个部分的主要目的在于将视觉信息和文字信息进行配对结合（图像和描述文字）。模型输入一幅图像与一句话，然后对它们俩的匹配程度进行打分作为输出（有关这个模型工作的具体细节，作者Karpathy引用了另外一篇论文paper <https://arxiv.org/pdf/1406.5679v1.pdf>。模型主要使用兼容/不兼容图文对compatible and incompatible image-sentence pairs进行训练。）

现在看一下该如何表现一幅图像。首先，把一幅图像输入一个用ImageNet数据训练过的RCNN网络，检测其中的物体。前19个检测出来的物体（加上自身）表现为深度为500维的维度空间。那么现在我们有了20个500维向量（文章中表示为 $v$ ），这就是图像中的信息。随后，我们需要获取语句中的信息。我们利用双向RNN架构，把输入语句嵌入同样的多模态维度空间。在模型的最高层，输入的语句内容会以给定的句式（given sentence）表现出来。这样，图像的信息和语句信息就处于同一个建模空间内，我们通过计算其内积就可以求得相似度了。

## Generation Model

刚才说了，配对alignment模型创建了一个存放图像信息（通过RCNN）和对应文本信息（通过BRNN）的数据集。现在我们就可以利用这个数据集来训练产生generation模型，让模型从给定图像中生成一个新的描述文本信息。模型将一幅图像输入CNN，忽略其softmax层，其全连通层的输出直接作为另一个RNN的输入。这个RNN的主要功能则是为语句的不同单词形成一个概率分布函数。（同样需要另外训练）

声明：这绝对是最难懂的文章之一，如果大家对我的讲述有不同意见和建议，请一定在评论区留言。



## 文章重要性

对我来说，本文要点在于利用了看起来似乎不同的两种模型RNN和CNN，创造了一个结合机器视觉和自然语言处理两方面功能的应用。它打开了新世界的大门，提供了一个新的思路，使得深度学习模型更加聪明并能够胜任跨学科领域的任务。

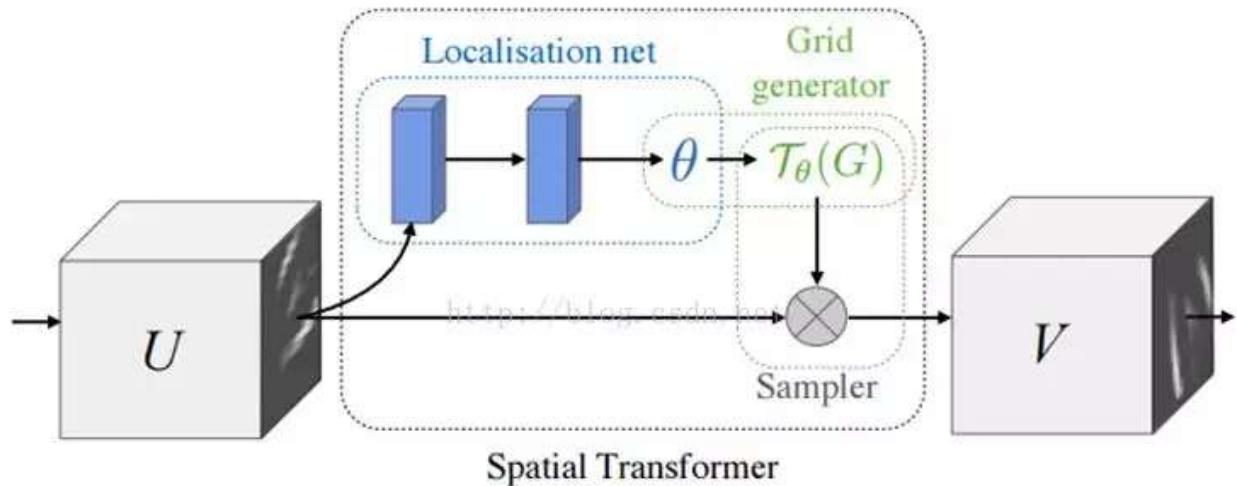
## Spatial Transformer Networks (2015)

<https://arxiv.org/pdf/1506.02025.pdf>

最后，让我们介绍一个最近的文章。这篇文章是由Google Deepmind研究组在一年前撰写的。它提出了一种空间变形模块Spatial Transformer module。模块将输入图像进行某种变形从而使得后续层处理时更加省时省力。比起修改CNN的主要结构，作者更关注于对输入图像进行改造。它进行的改造主要有两条：姿态正规化pose normalization（主要指图像场景中的物体是否倾斜、是否拉伸）以及空间聚焦spatial attention（主要指在一个拥挤的图像中如何聚焦某个物体）。在传统CNN中，如果想要保证模型对尺度和旋转具有不变性，那么需要对应的大量训练样本。而在这个变形模块中，则不需要如此麻烦，下面就让我们看看它是怎么做的。

在传统CNN中，应对空间不变性的模块主要是最大池化maxpooling层。其背后的直观原因在于最大池化层能够提取特征信息（在输入图像中有着高激活值的那些区域）的相对位置作为一个重要属性，而不是绝对位置。而文中所述的空间变形模块则是通过一种动态的方式对输入图像进行变换（扭曲、变形）。这种形式不像传统的最大池化操作那样简单与死板。让我们看看它的组成：

- 一个局部网络结构，通过输入图像计算出应该对图像采用的形变参数并将其输出。形变参数称作 $\theta$ ，定义为一个6维的仿射变换向量。
- 一个正规化网格经过上述参数的仿射变换之后生成的采样网格产物。
- 用作对输入图层变换的采样器sampler



A Spatial Transformer module

这样的一个模块可以插入于CNN网络的任何地方，帮助整个网络结构学习特征图层形变，降低训练成本。

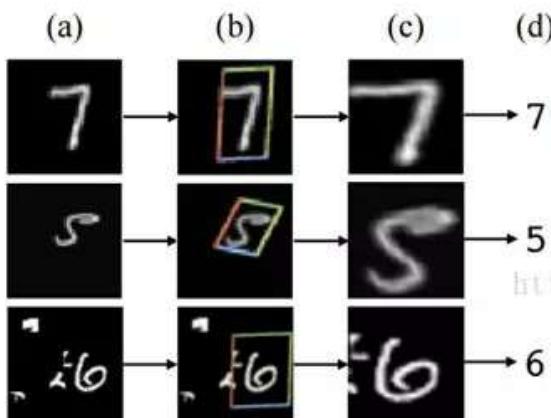


Figure 1: The result of using a spatial transformer as the first layer of a fully-connected network trained for distorted MNIST digit classification. (a) The input to the spatial transformer network is an image of an MNIST digit that is distorted with random translation, scale, rotation, and clutter. (b) The localisation network of the spatial transformer predicts a transformation to apply to the input image. (c) The output of the spatial transformer, after applying the transformation. (d) The classification prediction produced by the subsequent fully-connected network on the output of the spatial transformer. The spatial transformer network (a CNN including a spatial transformer module) is trained end-to-end with only class labels – no knowledge of the groundtruth transformations is given to the system.

[Great overview of the function of a Spatial Transformer module](#)

图中文字：在一个全连通网络架构用于扭曲手写MNIST库的数字识别的项目中，添加空间变形模块spatial transformer作为架构的第一层的运行结果：(a)输入数据是MNIST手写库中的图像，图像上施加了随机变换、缩放、旋转以及其它干扰噪声clutter。(b)空间变形模块预测的图像形变。(c)通过空间变形模块处理后的结果。(d)随后通过全连通网络分类预测后的结果。附带空间变形模块的网络架构在训练时仅使用了最后的正确标签，也就是数字标签，而并没有使用正确变形参数作为标签进行训练。

## 文章重要性

这篇文章吸引眼球的地方在于它提出这样的一种可能性：对CNN的改进并不一定要对网络架构的大规模修改，也不需要创造出另外一个ResNet或Inception module这样的复杂模型。这篇文章通过实现了一个对输入图像进行仿射变换的简单功能从而让模型拥有了很强的形变、伸缩、旋转不变性。如果对本文所述的模型还有兴趣的同学，可以看一下这个Deepmind团队的视频video, [https://drive.google.com/file/d/0B1nQa\\_sA3W2iN3RQLXVFRkNXN0k/view](https://drive.google.com/file/d/0B1nQa_sA3W2iN3RQLXVFRkNXN0k/view)

对CNN加空间形变模块的结果有很好的展示，同时也可以参考这个Quora讨论贴discussion。  
<https://www.quora.com/How-do-spatial-transformer-networks-work>

这就是我们的卷积网络入门的三部曲。再次强烈推荐Stanford的CS 231n视频课程，  
<http://study.163.com/course/courseMain.htm?courseId=1004697005>

# 想与深度学习大咖们交流吗？

长按二维码加小编微信  
拉你进深度学习交流群  
大牛都在这



ID: huzhanli1991



科技

科技  
头条

看新闻,免费发广告!  
流量轻松10万+



点击下方“阅读原文”下载【科技头条】



Read more