# 基于im2txt的图像叙事探究

郑子浩2017202117

# 问题简介

图像标注问题其本质是视觉到语言（Visual-to-Language，即V2L）的问题，解释起来很简单，就是四个字：看图说话。就像老师要求小朋友们在看图说话作业中完成的任务一样，我们也希望算法能够根据图像给出能够描述图像内容的自然语言语句。然而这种对于人类实在是小事一桩的小儿科级任务，在计算机视觉领域却不能不说是一个挑战：因为图像标注问题需要在两种不同形式的信息（图像信息到文本信息）之间进行“翻译”。

随着深度学习领域的发展，一种将深度卷积神经网络（Deep Convolutional Neural Network）和循环神经网络（Recurrent Neural Network）结合起来的方法在图像标注问题上取得了显著的进步。由于该方法的成功，使得基于该方法的对图像标注问题研究迅速地火热起来，在2016年的IEEE国际计算机视觉与模式识别会议（即IEEE Conference on Computer Vision and Pattern Recognition，缩写为CVPR）上专门有一个小型会议（session）的主题就是图像标注。

图像标注模型的发展，说是发展，其实时间也并不长，将CNN和RNN结合的模型用于解决图像标注问题的研究最早也就从2014开始提出，在2015年开始对模型各部分组成上进行更多尝试与优化，到2016年CVPR上成为一个热门的专题。

在这个发展中，将RNN和CNN结合的核心思路没变，变化的是使用了更好更复杂的CNN模型，效果更好的LSTM，图像特征输入到RNN中的方式，以及更复合的特征输入等。正由于其发展时间跨度较短，通过阅读该领域的一些重要文章，可以相对轻松地理出大牛们攻城拔寨的思路脉络，这对我们自己从事研究的思路也会有所启发。

# 图像标注数据集

到目前为止，深度学习依旧是一种需要大量数据来进行驱动的方法。小样本学习尚未有突破性的进展，所以数据对于基于深度学习的算法依旧非常重要。在图像标准问题研究的过程中，研究者们对于基准数据库的选择偏好也在发生变化，一些数据集运用的越来越广泛，而一些数据集则越来越少地被使用。本小节将基于图像标注问题，对这些数据集做简要的介绍和对比。

## Microsoft COCO Caption数据集

Microsoft COCO Caption数据集的推出，是建立在Microsoft Common Objects in COntext (COCO)数据集的工作基础上的。在论文《Microsoft COCO Captions: Data Collection and Evaluation Server》中，作者们详细介绍了他们基于MS COCO数据集构建MS COCO Caption数据集的工作。

简要地来说，就是对于原COCO数据集中约330,000张图像，使用亚马逊公司的“土耳其机器人（Mechanical Turk）”服务，人工地为每张图像都生成了至少5句标注，标注语句总共超过了约150万句。至于亚马逊的“土耳其机器人”服务，其实也就是另一种形式的雇人拿钱干活而已。

实际上，COCO Caption数据集包含了两个数据集：

第一个数据集是MS COCO c5。它包含的训练集、验证集合测试集图像和原始的MS COCO数据库是一致的，只不过每个图像都带有5个人工生成的标注语句。

第二个数据集是MS COCO c40。它只包含5000张图片，而且这些图像是从MS COCO数据集的测试集中随机选出的。和c5不同的是，它的每张图像都有用40个人工生成的标注语句。

之所以要做MS COCO c40数据集，是因为如果有更多的参考标注语句，很多对于算法生成的标注的自动计算标准能够和人类判断有更高的相关性。下一步可能将MS COCO验证集中所有的图像都加上40个人工生成的标注语句。

作者们的另一个主要工作就是搭建了一个评价服务器，实现了当前最流行的评价标准（BLEU, METEOR, ROUGE and CIDEr）。使用MS COCO Caption数据集训练并用验证机调参后，研究者可以按照固定的JSON格式，向服务器上传自己算法对于测试集图像生成的标注语句，服务器将自动地给出各种评价标准的得分。要上传结果，需要在CodaLab注册账号，且每个账号能够提交结果的次数是有限的。微软在Github上也提供了能够在本地对验证集数据生成标注进行评价的代码。

## Flickr8K和30K

Flickr8K和Flickr30K数据集的特性从它们的命名就能很方便地猜测出来：

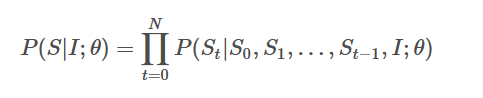
图像数据来源是雅虎的相册网站Flickr；数据集中图像的数量分别是8,000张和30,000张（确切地说是31,783）；

这两个数据库中的图像大多展示的是人类在参与到某项活动中的情景。每张图像的对应人工标注依旧是5句话。这两个数据库本是同根生，所以其标注的语法比较类似。数据库也是按照标准的训练集、验证集合测试集来进行分块的。

相较于MS COCO Caption数据集，Flickr8K和Flickr30K数据集的明显劣势就在于其数据量不足。

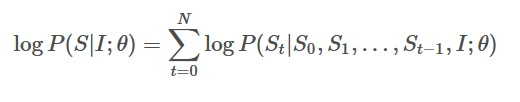
# 任务描述

将理解图片和生成描述这两个子任务统一到一起的话，那么Image Caption任务的训练过程可以描述为这个形式：对于训练集的一张图片，其对应的描述为序列（其中代表句子中的词）。对于模型，给定输入图片，模型生成序列的概率为

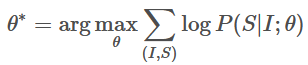


拆解成连乘的形式后，问题就变成了建模条件概率 。通常来说，RNN是当仁不让的第一选择，因为理论上它可以保留全部上文信息（对于“长时依赖”问题，使用LSTM来缓解），而不像n-gram或者CNN那样只能取一个窗口。

将似然函数取对数，得到对数似然函数：



模型的训练目标就是最大化全部训练样本的对数似然之和：



式中为训练样本。这种极大似然估计的方式等价于使用对数损失函数的经验风险最小化。

训练好模型后，对于生成过程，则是根据输入的一张图片，推断出最可能的序列来输出：



当然，计算全部序列的概率然后选出概率最大的序列当然是不可行的，因为每个位置都有词表规模的词作为候选，搜索规模会随序列的长度而指数级增长，所以需要用beam search来缩小搜索空间。

**而这个过程就好像：是机器翻译，自动摘要，即encoder-decoder。**

# 评价指标

下面说一下自动评价方法的评价指标。常用的指标是BLEU、Meteor、ROUGE、CIDEr和SPICE。前两个是评测机器翻译的，第三个是评测自动摘要的，最后两个应该是为caption定制的。下面简单介绍一下前四个。

对于英文来说，模型生成的caption和人工给出的参考caption都经过了预处理：**词条化（tokenization）**，去除标点。一般来说，评价生成的序列与参考序列的差异时，都是基于 n-gram 的，进而考察局部性质而非只考察单个词。另外，COCO官方提到 n-gram 中的词没有经过**词干还原（stemming，启发式地去掉单词两端词缀）**这个步骤。

## Gram

**N-Gram是一种基于统计语言模型的算法。它的基本思想是将文本里面的内容按照字节进行大小为N的滑动窗口操作，形成了长度是N的字节片段序列。**

每一个字节片段称为gram，对所有gram的出现频度进行统计，并且按照事先设定好的阈值进行过滤，形成关键gram列表，也就是这个文本的向量特征空间，列表中的每一种gram就是一个特征向量维度。

**该模型基于这样一种假设，第N个词的出现只与前面N-1个词相关，而与其它任何词都不相关，整句的概率就是各个词出现概率的乘积。**这些概率可以通过直接从语料中统计N个词同时出现的次数得到。常用的是二元的Bi-Gram和三元的Tri-Gram。

n-gram模型用于评估语句是否合理

如果我们有一个由m个词组成的序列（或者说一个句子），我们希望算得概率 ，根据链式规则，可得：



这个概率显然并不好算，不妨利用马尔科夫链的假设，即当前这个词仅仅跟前面几个有限的词相关，因此也就不必追溯到最开始的那个词，这样便可以大幅缩减上述算式的长度。即



这个马尔科夫链的假设为什么好用？我想可能是在现实情况中，大家通过真实情况将n=1，2，3，....这些值都试过之后，得到的真实的效果和时间空间的开销权衡之后，发现能够使用。

下面给出一元模型，二元模型，三元模型的定义：

当 n=1, 一个一元模型（unigram model)即为 ：



当 n=2, 一个二元模型（bigram model)即为 ：



当 n=3, 一个三元模型（trigram model)即为



然后下面的思路就很简单了，在给定的训练语料中，利用贝叶斯定理，将上述的条件概率值（因为一个句子出现的概率都转变为右边条件概率值相乘了）都统计计算出来即可。而求马尔可夫式可以这样：



具体推导（当bigram时）是：



## BLEU

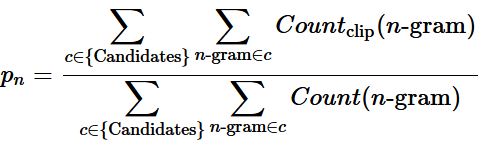
BLEU是事实上的机器翻译评测标准，n 常取1到4，基于准确率（precision）的评测。我们首先看一下它在机器翻译里是怎么做的。

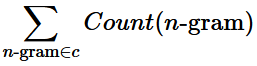
需要明确，在机器翻译中，翻译的评测是以句子为单位的。

(1) 首先来看一个最简单的思路：对于一个源语言句子，计算模型生成的译文中的 n-gram 的个数，然后再计算这些 n-gram 中有多少是同时出现在了参考译文（不止一句）中，从而计算出一个百分比来作为 precision 。

但这样的做法存在一些问题：比如说参考句子1是“the cat is on the mat”，参考句子2是“there is a cat on the mat”，而模型生成的句子是“the the the the the the the”，那么按照上述定义，考察 n=1 的情况，也就是unigram，模型生成的句子包含7个 unigram ，这7个 unigram 全部出现在了参考句子集合中，所以将得到 7/7 这样的满分，但是这样的译文显然没有意义。为了获得较高的指标，模型完全可以在任何位置都去生成一个“百搭”的词，使得分子随着分母的增长而增长。

1. 为了解决“百搭”词的问题，需要修正 precision 的计算方式。考虑模型生成的句子 c 的全部 n-gram ，考察其中的任一 n-gram ：首先计算其在 c 中出现的次数 Count(n-gram) ；然后统计其在各参考句子中分别出现的次数的最大值，将该值与 Count(n-gram) 的较小者记作该 n-gram 的匹配次数 Countclip(n-gram) 。之后，再把每个 n-gram 的计算结果累加起来，得到句子的结果。所以precision可以用如下方式计算：

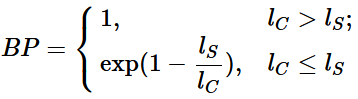


式中 {Candidates} 代表需要评测的多句译文的集合。当n取1时，就是句子 c 的长度。

回过头来看上面那个例子，译文句子的 unigram 只有“the”，它在译文中出现了7次，故 Count(the)=7；在参考句子1中出现2次，参考句子2中出现1次，最大值为2，所以“the”的匹配次数为 Countclip(the)=min{7,2}=2 ，因此precision为 2/7 。

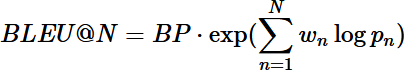
（3）但是这样的计算方式仍然存在问题：比如模型生成的句子是“the cat is on”，那么从 n 不论取1、2、3还是4，得分都是1，满分。换言之，由于评价的是precision，所以会倾向于短句子，如果模型只翻译最有把握的片段，那么就可以得到高分，因此要对短句子进行惩罚。

惩罚的方式就是在原先的评价指标值上乘一个惩罚因子（brevity penalty factor）：当模型给出的译文句子 c的长度 lc 要比参考句子的长度 ls 长时，就不进行惩罚，即惩罚因子为1，比如说有三个参考句子的长度分别为12、15、17，模型给出的译文句子长度为12，那么就不进行惩罚，比较的是各参考句子长度里最接近的那个；否则就惩罚：



式中的代表模型给出的测试集全部句子译文的长度总和，代表与模型给出译文句子长度最接近的参考译文的长度（语料级别）。

综合起来，BLEU的评分公式采用的是对数加权平均值（这是因为当n增大时评分会指数级减小），再乘上惩罚因子：



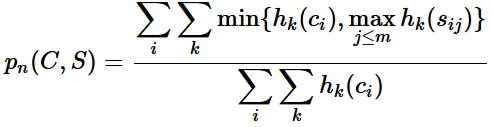
式中的N通常取4，权重wn通常取1/N（几何平均）。最终评分在0到1之间，1表示完全与人工翻译一致。

BLEU的优点是它考虑的粒度是 n-gram 而不是词，考虑了更长的匹配信息；BLEU的缺点是不管什么样的 n-gram 被匹配上了，都会被同等对待。比如说动词匹配上的重要性从直觉上讲应该是大于冠词的。

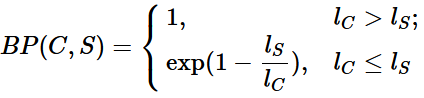
(4) 在caption任务中，处理方式和机器翻译是一样的：多张图片就相当于翻译里的多个源语言句子。为了描述清楚还是给一下公式：

对于测试集的一张图片，模型生成的caption记为∈C（C是全部构成的集合），且将任一n-gram 记作；人工给出的参考caption的集合为 （S是全部Si构成的集合），为句子，m为参考caption的数量（数据集里的一张图片通常会有多个参考caption，比如Flickr、COCO数据集上每张图片都有5个参考caption）。将某个n-gram 出现在句子 ci 中的次数记作，类似地，可定义。

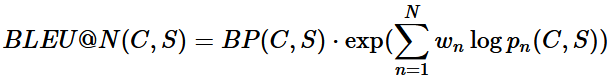
在整个测试集上，precision值为



惩罚因子的值为



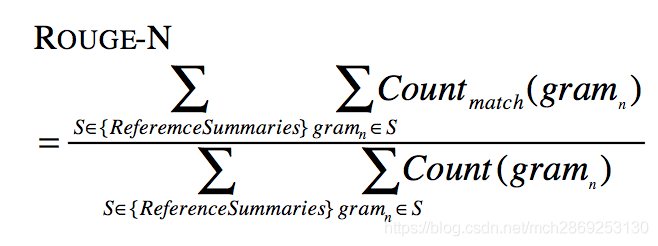
所以BLEU@N的值为



## ROUGE

ROUGE基于摘要中n元词(n-gram)的共现信息来评价摘要，是一种面向n元词召回率的评价方法。基本思想为由多个专家分别生成人工摘要，构成标准摘要集，将系统生成的自动摘要与人工生成的标准摘要相对比，通过统计二者之间重叠的基本单元(n元语法、词序列和词对)的数目，来评价摘要的质量。通过与专家人工摘要的对比，提高评价系统的稳定性和健壮性。

ROUGE准则由一系列的评价方法组成，包括ROUGE-N(N是n-gram中n，取值有1，2，3，4)，ROUGE-L，ROUGE-S, ROUGE-W，ROUGE-SU等。在自动文摘相关研究中，一般根据自己的具体研究内容选择合适的ROUGE方法。

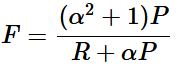


分母是人工摘要（也就是标准摘要）中n-gram的个数，分子是人工摘要和机器生成的自动摘要共现（重合）的n-gram的个数。可以看出，ROUGE与召回率的定义很相似。

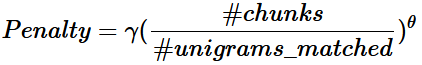
## Meteor

Meteor也是来评测机器翻译的，对模型给出的译文与参考译文进行词对齐，计算词汇完全匹配、词干匹配和同义词匹配等各种情况的准确率、召回率和F值。

首先计算 unigram 情况下的准确率 P 和召回率 R（计算方式与BLEU、ROUGE类似），得到调和均值（F值）



Meteor的特别之处在于，它不希望生成很“碎”的译文：比如参考译文是“A B C D”，模型给出的译文是“B A D C”，虽然每个 unigram 都对应上了，但是会受到很严重的惩罚。惩罚因子的计算方式为：



式中的 #chunks 表示匹配上的语块个数，如果模型生成的译文很碎的话，语块个数会非常多；#unigrams\_matched 表示匹配上的 unigram 个数。所以最终的评分为

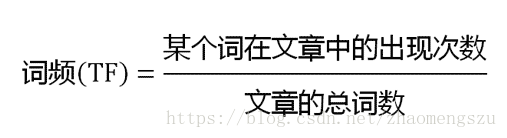
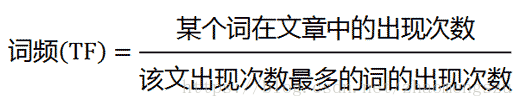


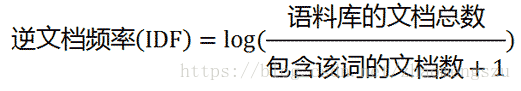
用于机器翻译评测时，通常取 α=3 、γ=0.5 和 θ=3 。

## CIDEr

对于测试集的一张图片，模型生成的caption记为∈C（C是全部构成的集合），且将任一n-gram 记作；人工给出的参考caption的集合为 （S是全部Si构成的集合），为句子，m为参考caption的数量（数据集里的一张图片通常会有多个参考caption，比如Flickr、COCO数据集上每张图片都有5个参考caption）。将某个n-gram 出现在句子 ci 中的次数记作，类似地，可定义。

这个指标将每个句子都看作“文档”，将其表示成 tf-idf 向量的形式，然后计算参考caption与模型生成的caption的余弦相似度，作为打分。换句话讲，就是向量空间模型。考虑一张图片（I是全部测试集图片的集合）：

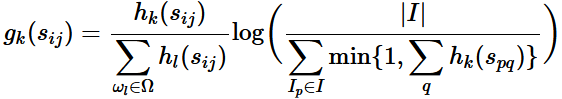
或者



IMG_256

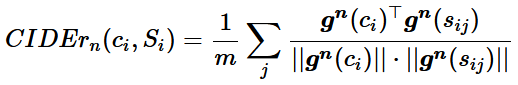
知道了"词频"（TF）和"逆文档频率"（IDF）以后，将这两个值相乘，就得到了一个词的TF-IDF值。某个词对文章的重要性越高，它的TF-IDF值就越大。所以，排在最前面的几个词，就是这篇文章的关键词。

对于一个 n-gram 和参考caption ，tf-idf 计算方式为：

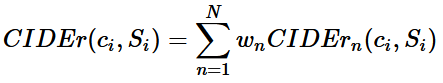


式中的 Ω 是全部 n-gram 构成的词表。可以看出 idf 的分母部分代表的是 ωk出现于参考caption的图片个数。

那么，CIDEr的值可以用余弦相似度的平均值来计算：



类似于BLEU的做法：



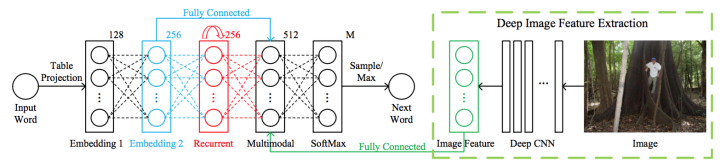
这个指标的motivation之一是刚才提到的BLEU的一个缺点，就是对所有匹配上的词都同等对待，而实际上有些词应该更加重要。

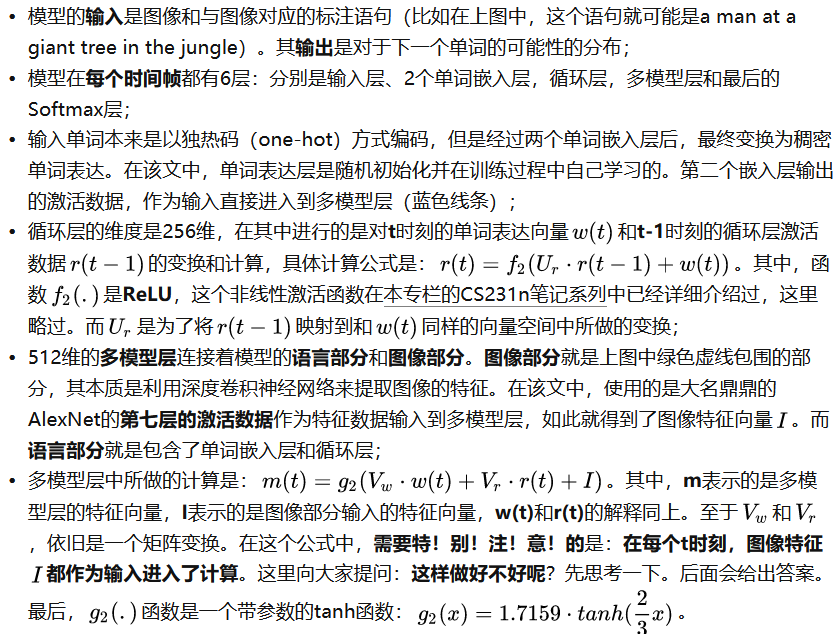
# RNN模型

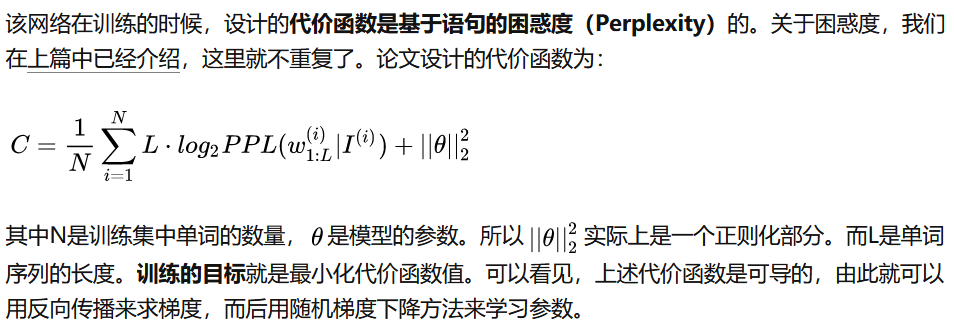
2014年10月，百度研究院的Junhua Mao和Wei Xu等人在arXiv上发布论文《Explain Images with Multimodal Recurrent Neural Networks》，提出了multimodal Recurrent Neural Network（即m-RNN）模型，创造性地将深度卷积神经网络CNN和深度循环神经网络RNN结合起来，用于解决图像标注和图像和语句检索等问题。

在后续的几篇优秀论文中，m-RNN都被作为一个基准方法用于比较和超越。因此，首先介绍百度研究院的m-RNN模型，在于其创造性工作。

论文中，在对原始RNN结构进行简要说明后，提出了m-RNN模型如下：







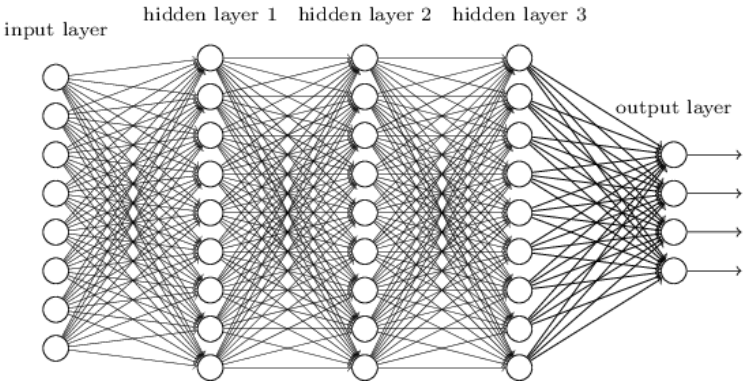
# 知识补充

基于Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge

## Encoder-Decoder框架

## CNN

DNN(全连接深度神经网络)结构图如下：



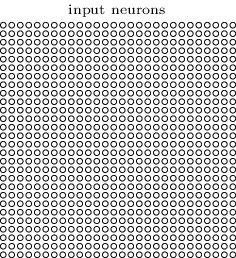
全连接深度神经网络，顾名思义，每个神经元都与相邻层的神经元连接。在这个实验中，每个数字的image是28\*28，也就是784(=28\*28)个数值，每个数值对应一个像素值，值的大小反应像素点的强度。这就意味着我们网络的输入层有784个神经元。输出层呢？由于我们是预测0-9这几个数字，输出层当然就是10个神经元了。至于隐藏层节点的个数我们可以自行选定，本实验中选的是500.

我们想想为什么DNN在训练后能够正确地分类？那肯定是它学到了东西，学到什么东西呢？它学到了图片中的某些空间结构，不同数字它们的空间结构肯定是不一样的，而这样的空间结构就是由像素点与像素点之间的关系形成。我们再仔细看DNN输入层和第一个隐藏层，发现它对我们输入的784个像素点是同等对待的，也就是说它此时并没有考虑像素点与像素点之间的关系。有没有一个好点的模型能够考虑到这点呢？那就是CNN。

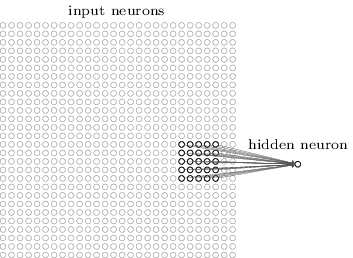
**CNN有三个基本思想，局部感受野(local receptive fields) 权值共享(shared weights) 池化(pooling)。**

### 局部感受野

刚刚我们在DNN中是把输入层784个神经元排成了一条长线，这里我们还原图片原来的样子(28\*28)，如下图

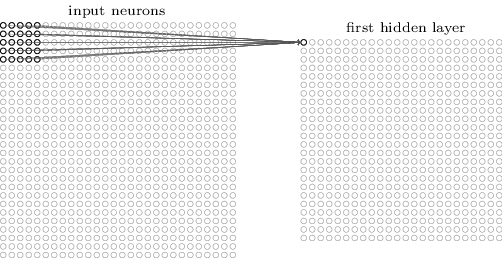


DNN中，我们会把输入层的每个神经元都与第一个隐藏层的每个神经元连接。而在CNN中我们这样做的，第一个隐藏层的神经元只与局部区域输入层的神经元相连。下图就是第一个隐藏层的某个神经元与局部区域输入层的神经元相连的情况。

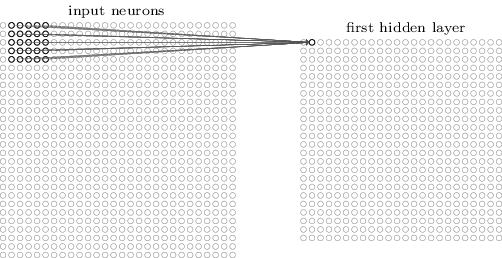


这里的局部区域就是局部感受野，它像一个架在输入层上的窗口。你可以认为某一个隐藏层的神经元学习分析了它”视野范围“(局部感受野)里的特征。图中一个隐藏层的神经元有5\*5个权值参数与之对应。

我们移动这样一个窗口使它能够扫描整张图，每次移动它都会有一个不同的节点与之对应。我们从输入层左上角开始，如下



然后，我们一个像素往右滑动一个像素，如下

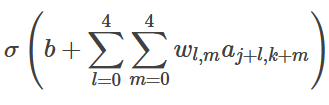


以此类推可以形成第一个隐藏层，注意我们的图片是28\*28的，窗口是5\*5的，可以得到一个24\*24(24=28-5+1)个神经元的隐藏层

这里我们的窗口指滑动了一个像素，通常说成**一步(stride)**，也可以滑动多步，这里的stride也是一个超参，训练是可以根据效果调整，同样，**窗口大小**也是一个超参。

### 权值共享

一个隐藏层的神经元有5\*5个权值参数与之对应。这里要补充下，这24\*24个隐藏层的神经元它们的权值和偏移值是共享的，用公式描述下



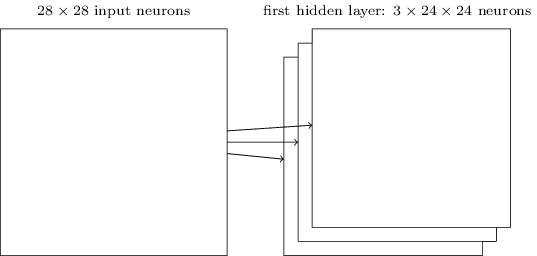
σ代表的是激活函数，如sigmoid函数等，b就是偏移值，w就是5\*5个共享权值矩阵，我们用矩阵a表示输入层的神经元，ax,y表示第x+1行第y+1列那个神经元(注意，这里的下标默认都是从0开始计的，a0,0表示第一行第一列那个神经元）所以通过矩阵w线性mapping后再加上偏移值就得到公式中括号里的式子，表示的是隐藏层中第j+1行k+1列那个神经元的输入。这部分原理和DNN是一样的，如果把w改成28\*28的矩阵就变成了全连接，就是DNN了。

简化一下这个公式

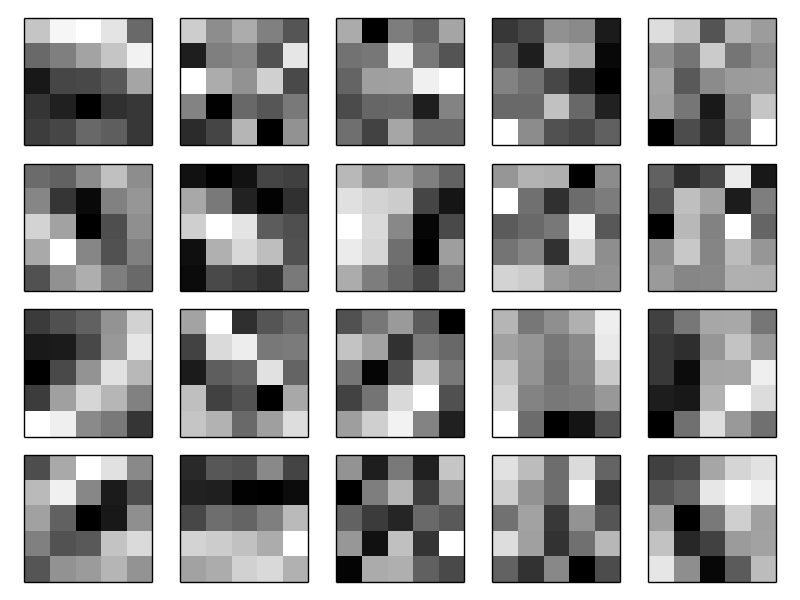


a1表示隐藏层的输出，a0表示隐藏层的输入，而∗就表示**卷积操作(convolution operation)** 这也正是卷积神经网络名字的由来。

由于权值共享，窗口移来移去还是同一个窗口，也就意味着第一个隐藏层所有的神经元从输入层**探测(detect)**到的是同一种**特征(feature)**，只是从输入层的不同位置探测到(图片的中间，左上角，右下角等等)，必须强调下，**一个窗口只能学到一种特征**！另外，**窗口还有其他叫法：卷积核(kernal),过滤器(filter)**。我们在做图像识别时光学习一个特征肯定是不够的，我们想要学习更多的特征，就需要更多的窗口。如果用三个窗口的话如下图



窗口与窗口间的w和b是不共享的，三个窗口就表示有三个w矩阵和三个偏移值b，结果是从整张图片的各个位置学到三种不同的特征。到这里肯定有人会问，你说学到特征了，怎么证明学到了呀？现在我们用20个窗口来学习MNIST里的图片特征，我们只看20个窗口里的权值矩阵w，如果把这20个w画成20张黑白图，每张图片都是5\*5(一个权值代表一个像素点)，如下图所示

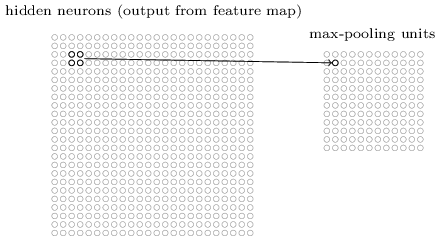


盯着其中的一张看，白色区域表示权值比较小，说明窗口的这部分对输入层的神经元不敏感(responds less)，相反黑色部分表示权值比较大，说明窗口的这部分对输入层的神经元敏感(responds more).每张图片都有明显的黑白区域，这也能够说明CNN确实学到一些和空间结构相关的特征。究竟学的是什么特征呢？这个很难回答清楚，此处暂不深究，更好理解的话可以参考 Visualizing and Understanding Convolutional Networks

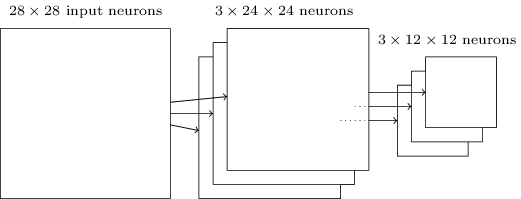
**权值共享还有一个很大的好处，就是可以大大减少模型参数的个数**。我们的例子中，一个窗口参数个数是26(5\*5+1),20个窗口就是520个参数，如果换成全连接的话就是785(28\*28+1)个参数，比CNN多了265个参数。可能你觉得265嘛，对计算机来说完全不算什么。如果我们是30个隐藏层的DNN的话(深度学习里很常见的)，需要23550(785\*30)个参数，是CNN的45倍多。。当然我们也不能光光去比较它们参数的个数，毕竟两个模型本质原理上就相差甚远，但是直觉上我们可以感受到，CNN可以依靠更少的参数来获得和DNN相同的效果，更少的参数就意味着更快的训练速度，这可是谁都想要的。

### 池化

CNN还有一个重要思想就是池化，池化层通常接在卷积层后面。池化这个词听着就很有学问，其实引入它的目的就是为了简化卷积层的输出。通俗地理解，池化层也在卷积层上架了一个窗口，但这个窗口比卷积层的窗口简单许多，不需要w，b这些参数，它只是对窗口范围内的神经元做简单的操作，如求和，求最大值，把求得的值作为池化层神经元的输入值，如下图，这是一个2\*2的窗口



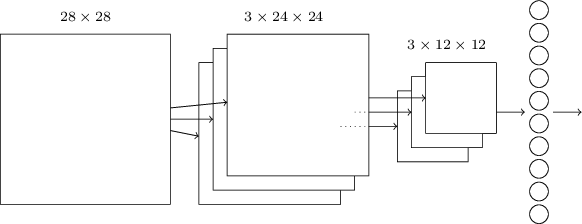
值得注意的是，我们此时的窗口每次移动两步，采用的是求最大值的方法，所有称之为max-pooling，刚刚卷积层含有24\*24个神经元，经过池化后到池化层就是12\*12个神经元。通常卷积层的窗口是多个的，池化层的窗口也是多个的。简单来说，卷积层用一个窗口去对输入层做卷积操作，池化层也用一个窗口去对卷积层做池化操作。但是注意这两个操作的本质区别。下面来看一个用三个卷积窗口和跟随其后的池化窗口长啥样。



怎么理解max-pooling呢？由于经过了卷积操作，模型从输入层学到的特征反映在卷积层上，max-pooling做的事就是去检测这个特征是否在窗口覆盖范围的区域内。这也导致了，它会丢失这种特征所在的精准位置信息，所幸的是池化层可以保留相对位置信息。而后者相比而言比前者更重要。不理解上面的话也没关系，但是需要记住池化层一个最大的好处：经过池化后，大大减少了我们学到的特征值，也就大大减少了后面网络层的参数(上图可以看出池化层的神经元数明显少于卷积层神经元数)。

max-pooling技术只是池化技术的一种，还有一种比较常用的是L2-pooling,与max-pooling唯一的区别就是在池化窗口扫过的区域里做的操作不是求最大值，而是所有神经元平方后求和再开根号，这和我们L2正则对权值参数的操作是一样的。实际操作中，这两种方式都是比较常用的。池化操作方式的选择也是我们调参工作的一部分，我们可以根据validation data集来调节，选择更好的池化操作。

介绍完CNN的三个几本思想概念后我们把它串起来看下。



从左往右依次是输入层，卷积层，池化层，输出层。

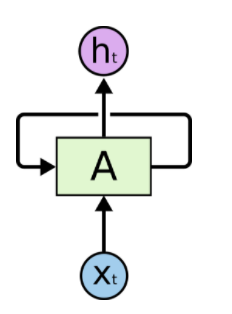
## LSTM

### RNN循环神经网络

人类针对每个问题的思考，一般不会是完全的从头开始思考。正如当你阅读这篇译文的时候，你会根据已经阅读过的内容来对后面的内容进行理解，你不会把之前的东西都丢掉从头进行思考，你对内容的理解是贯穿的。

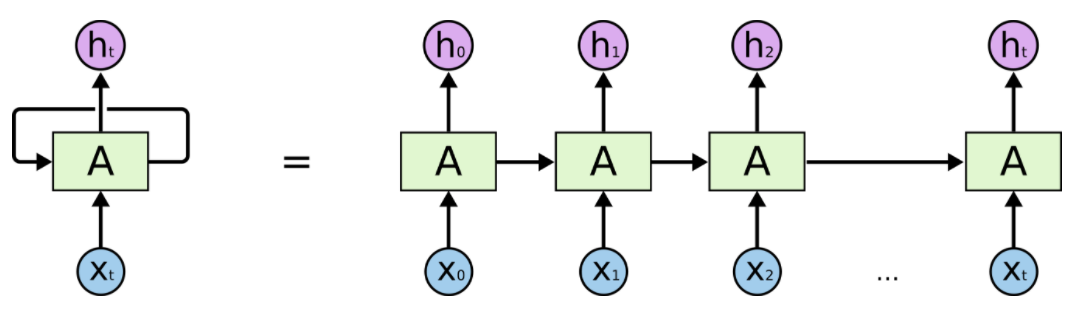
传统的神经网络做不到这一点，而这似乎是一个主要的缺点。 例如，假设您想对电影中的每个事件进行分类。我们无法想象传统神经网络如何能够利用前面的场景去干预后面的预测。

幸好循环神经网络解决了这个问题， 它们是具有循环的网络，允许信息持续存在，示意图如下。



在上图中，一组神经网络 A接收某些输入xt，并输出一个值ht。 循环允许信息从网络的一个步骤传递到下一个。

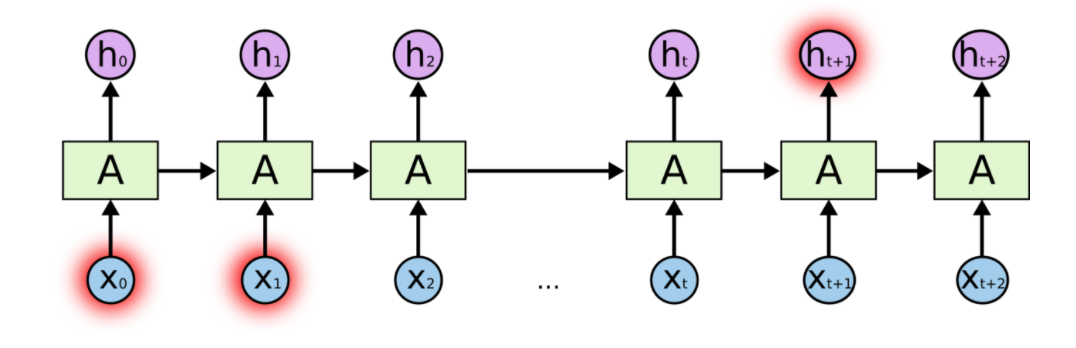
这些循环使得循环神经网络看起来很神秘。 然而，如果你再多考虑一下，你就会发现其实它和传统的神经网络并不是完全不同。 一个循环神经网络可以被认为是同一个网络的多个副本，每一个都传递一个消息给后继者。 我们考虑一下如果将循环展开会发生什么：



这种链状特征揭示了循环神经网络与序列和列表密切相关。 它们是用于此类数据的自然神经网络结构。

考虑尝试预测文本中的最后一个单词“I grew up in France… I speak fluent French.”。最近的信息表明，下一个单词可能是一种语言的名称，但如果我们想缩范围确定那种语言，我们需要从前面获取法国的背景。 相关信息和需要该信息的地方的距离变得非常大的情况是很可能的。

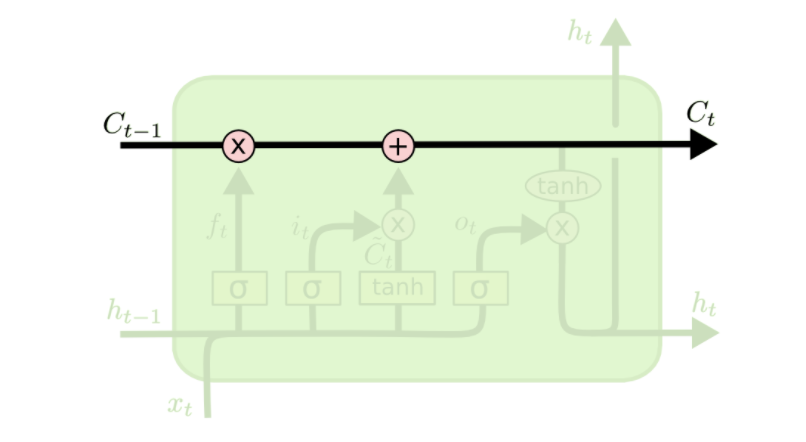
不幸的是，随着距离的增加，RNN无法有效的利用历史信息。



### LSTM背后的核心理念

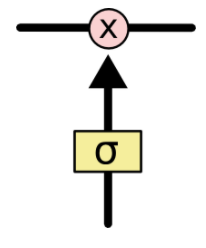
LSTM的关键是细胞状态，表示细胞状态的这条线水平的穿过图的顶部。

细胞的状态类似于输送带，细胞的状态在整个链上运行，只有一些小的线性操作作用其上，信息很容易保持不变的流过整个链。



LSTM确实具有删除或添加信息到细胞状态的能力，这个能力是由被称为门(Gate)的结构所赋予的。

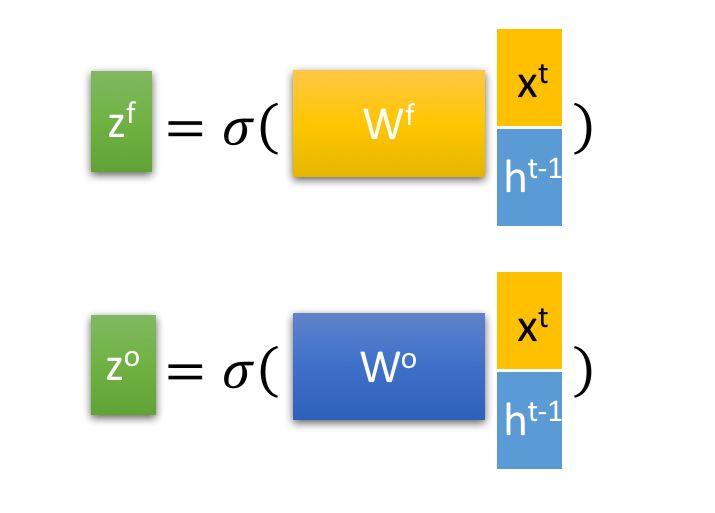
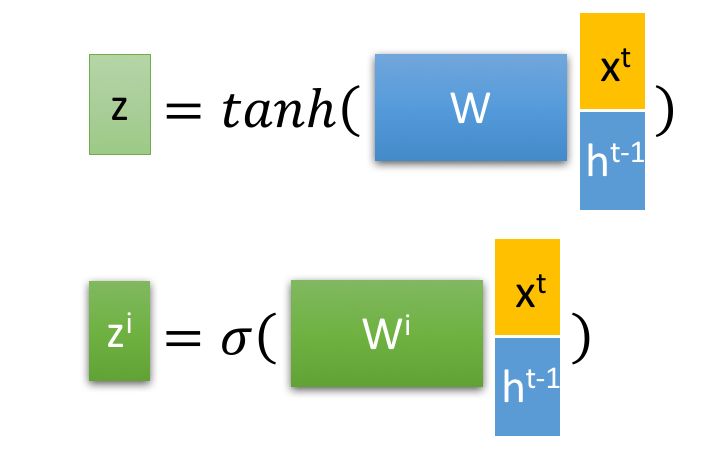
门(Gate)是一种可选地让信息通过的方式。 它由一个Sigmoid神经网络层和一个点乘法运算组成。



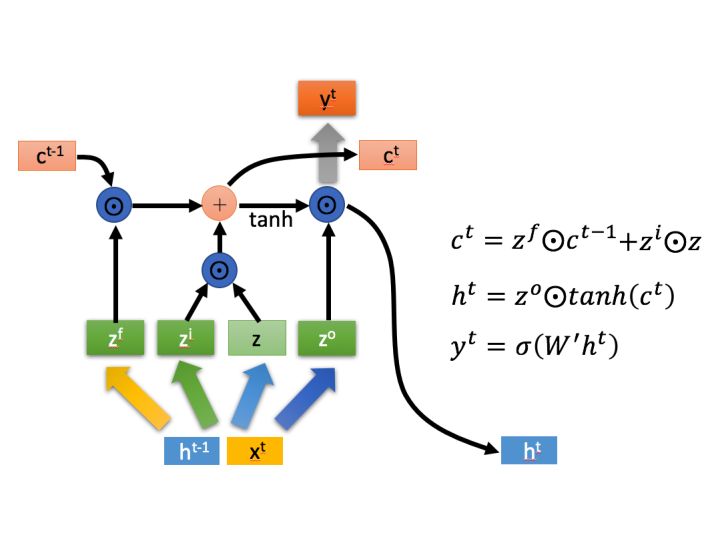
Sigmoid神经网络层输出0和1之间的数字，这个数字描述每个组件有多少信息可以通过， 0表示不通过任何信息，1表示全部通过

LSTM有三个门，用于保护和控制细胞的状态。

首先使用LSTM的当前输入和上一个状态传递下来的拼接训练得到四个状态。



其中，，，是由拼接向量乘以权重矩阵之后，再通过一个sigmoid激活函数转换成0到1之间的数值，来作为一种门控状态。而z则是将结果通过一个tanh激活函数将转换成-1到1之间的值（这里使用 tanh 是因为这里是将其做为输入数据，而不是门控信号）。



LSTM内部主要有三个阶段：

1. 忘记阶段。这个阶段主要是对上一个节点传进来的输入进行选择性忘记。简单来说就是会 “忘记不重要的，记住重要的”。

具体来说是通过计算得到的（f表示forget）来作为忘记门控，来控制上一个状态的 哪些需要留哪些需要忘。

2. 选择记忆阶段。这个阶段将这个阶段的输入有选择性地进行“记忆”。主要是会对输入进行选择记忆。哪些重要则着重记录下来，哪些不重要，则少记一些。当前的输入内容由前面计算得到的z表示。而选择的门控信号则是由（i代表information）来进行控制。

将上面两步得到的结果相加，即可得到传输给下一个状态的。也就是上图中的第一个公式。

3. 输出阶段。这个阶段将决定哪些将会被当成当前状态的输出。主要是通过来进行控制的。并且还对上一阶段得到的进行了放缩（通过一个tanh激活函数进行变化）。

与普通RNN类似，输出往往最终也是通过变化得到。

## beam search

Beam Search（集束搜索）是一种启发式图搜索算法，通常用在图的解空间比较大的情况下，为了减少搜索所占用的空间和时间，在每一步深度扩展的时候，剪掉一些质量比较差的结点，保留下一些质量较高的结点。这样减少了空间消耗，并提高了时间效率，但缺点就是有可能存在潜在的最佳方案被丢弃，因此，Beam Search算法是不完全的，一般用于解空间较大的系统中。

该算法常用的场景如：机器翻译，语音识别，当系统的数据集比较大，计算资源受限，而且没有唯一最优解时，该算法能够较快的找到接近最正确的解。

beam search可以看做是做了约束优化的广度优先搜索，首先使用广度优先策略建立搜索树，树的每层，按照启发代价对节点进行排序，然后仅留下预先确定的个数（Beam width-集束宽度）的节点，仅这些节点在下一层次继续扩展，其他节点被剪切掉。

1、将初始节点插入到list中，

2、将给节点出堆，如果该节点是目标节点，则算法结束；

3、否则扩展该节点，取集束宽度的节点入堆。然后到第二步继续循环。

4、算法结束的条件是找到最优解或者堆为空。

在使用上，集束宽度可以是预先约定的，也可以是变化的，具体可以根据实际场景调整设定。

比如，对于MLE算法训练的模型，beam search只在预测的时候需要。训练的时候因为知道正确答案，并不需要再进行这个搜索。

预测的时候，假设词表大小为3，内容为a，b，c。beam size是2，decoder解码的时候：

1： 生成第1个词的时候，选择概率最大的2个词，假设为a,c,那么当前的2个序列就是a和c。

2：生成第2个词的时候，我们将当前序列a和c，分别与词表中的所有词进行组合，得到新的6个序列aa ab ac ca cb cc，计算每个序列的得分并选择得分最高2个序列，作为新的当前序列，假如为aa cb。

3：后面会不断重复这个过程，直到遇到结束符或者达到最大长度为止。最终输出得分最高的2个序列。

# NIC模型

## 模型结构

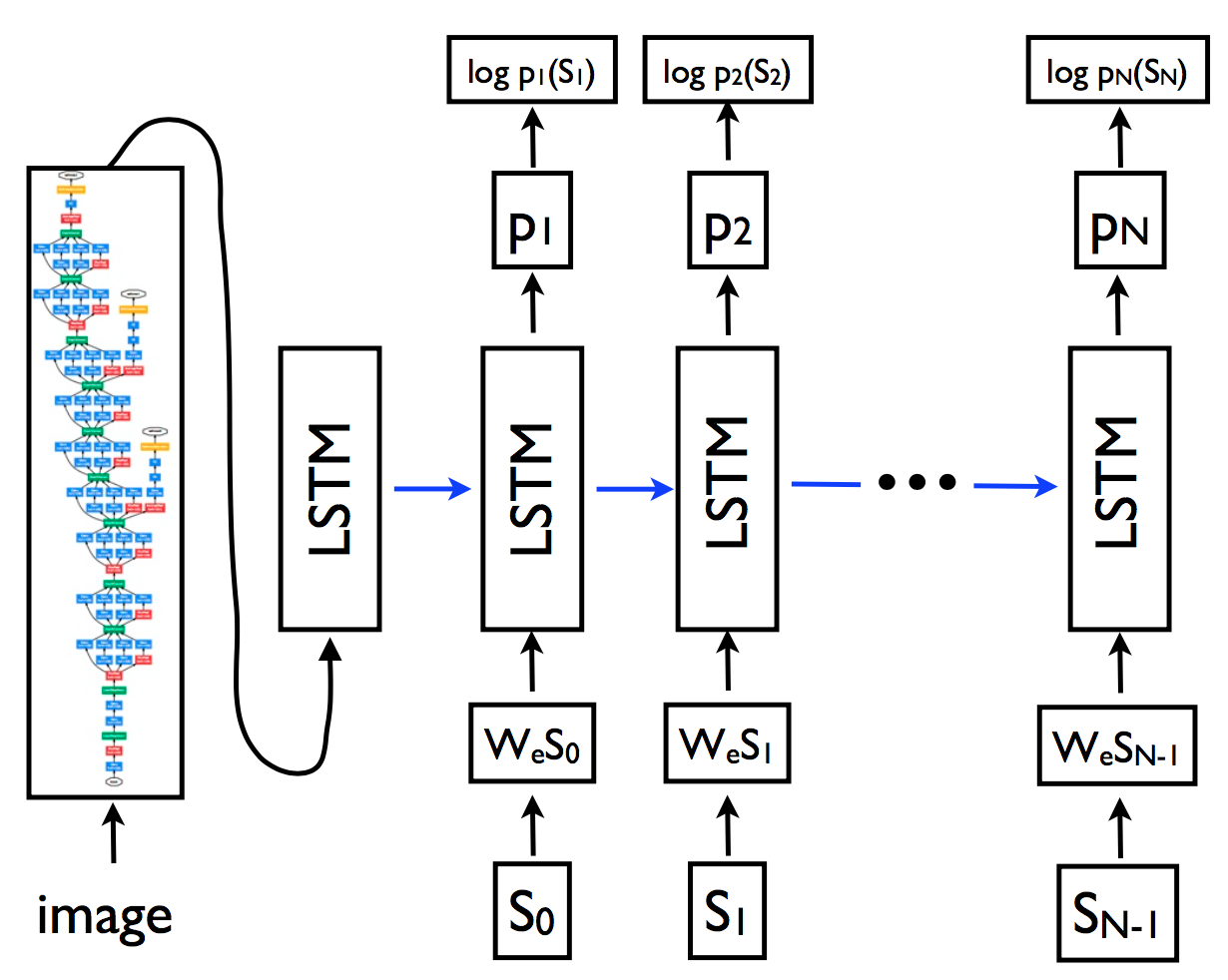
NIC模型的结构非常“简单”：就是利用encoder-decoder框架，首先利用CNN（这里是GoogLeNet）作为encoder，将 Softmax 之前的那一层固定维数的向量作为图像特征；再使用LSTM作为decoder，其中图像特征输入decoder（图像特征仅在开始时刻输入，后面就没用了）。模型的训练就是和任务描述那里介绍的一样，使用最大化对数似然来训练，然后在测试阶段采用beam search来减小搜索空间。

相较于百度的m-RNN模型，NIC模型的主要不同点在于：

抛弃RNN，使用了LSTM；

CNN部分使用了一个比AlexNet更好的卷积神经网络；

CNN提取的图像特征数据只在开始输入一次。



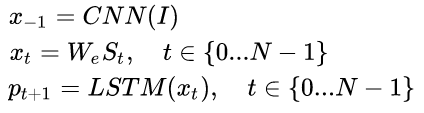
·图像特征部分是换汤不换药：我们可以看见，图像经过卷积神经网络，最终还是变成了特征数据（就是特征向量）出来了。唯一的不同就是这次试用的CNN不一样了，取得第几层的激活数据不一样了，归根结底，出来的还是特征向量；

·但是！图像特征只在刚开始的时候输入了LSTM，后续没有输入，这点和m-RNN模型是不同的！

·单词输入部分还是老思路：和m-RNN模型一样，每个单词采取了独热（one-hot）编码，用来表示单词的是一个维度是词汇表数量的向量。向量和矩阵We相乘后，作为输入进入到LSTM中。

·使用LSTM来替换了RNN。

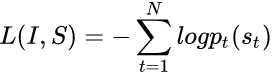
以上模型所示的流程，可以用下列公式来概括：



需要注意的是，这个结构与传统的encoder-decoder有点区别。

NIC 模型仅在decoder的开始时刻输入了图像特征，而不是在每个解码时刻都输入了图像特征。作者给出的理由是，如果在每个时刻都输入图像特征，那么模型会把图像的噪声放大，并且容易过拟合。实际上，后面的一些文章在基于attention来做caption，那么就必然要每个时刻都输入。因此本文的模型是非常简单的。

模型的训练：NIC模型的损失函数和m-RNN模型却有不同，但基本思路还是一样的：一个可求导的损失函数，利用反向传播来求梯度，然后利用随机梯度下降来学习到最优的参数。其损失函数为：



·综上：NIC模型相较于m-RNN模型，其重要的改进在于：

首先，在语言模型部分将RNN替换为了实践证明在NLP方面效果更好的LSTM。其次，在图像模型部分使用了效果更好的卷积神经网络模型来做图像特征数据的提取。最后，改变了图像特征数据的输入方式，从m-RNN的每个时间点都输入变成了只在初始时输入1次。

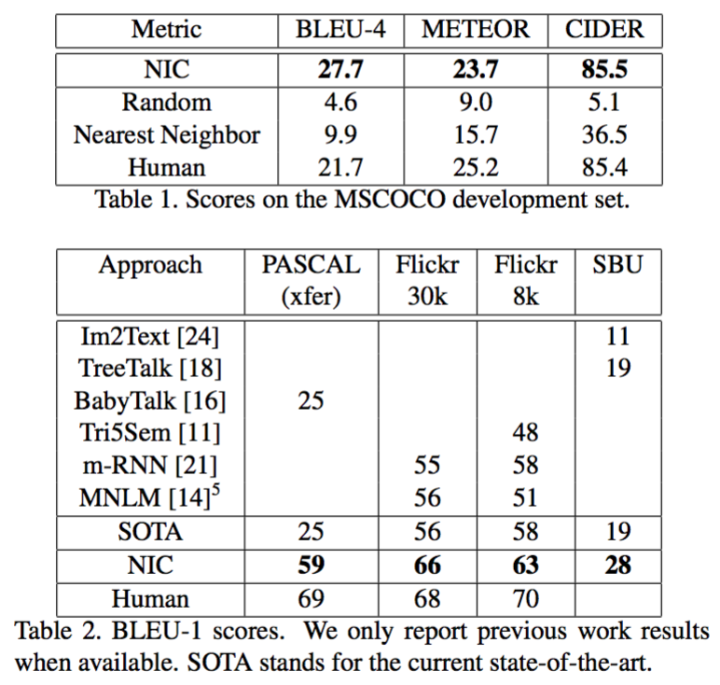
不难发现encoder-decoder的优点：非常灵活，不限制输入和输出的模态，也不限制两个网络的类型；encoder-decoder的缺点：当输入端是文本时，将变长序列表达为固定维数向量，存在信息损失，并且目前还不能处理太长的文本。

## 训练细节

在训练过程中，固定学习率且不加动量（momentum）；词条化后去掉了词频小于5的词；在ImageNet上预训练GoogLeNet，并且在训练caption模型时这部分的参数保持不变；在大型新闻语料上预训练词向量，但是效果并没有明显提升；使用dropout和模型ensemble，并权衡模型的容量：隐层单元个数与网络深度；512维词向量；使用困惑度（perplexity）来指导调参。

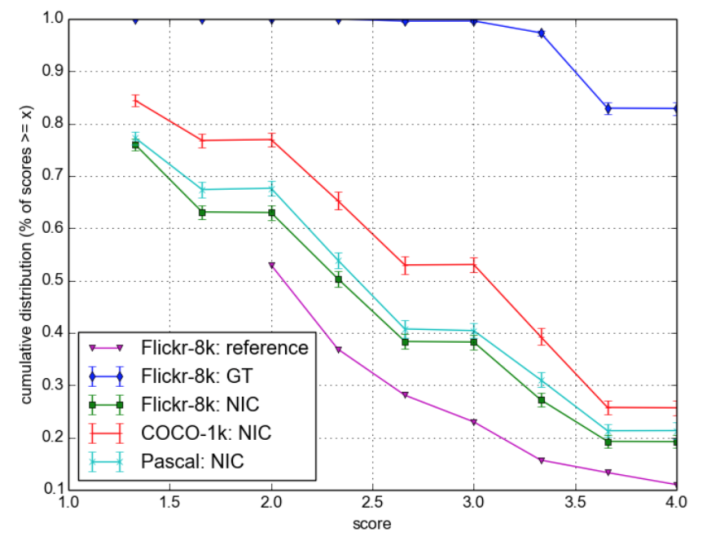
## 实验结果

作者在论文中多次强调，需要更好的自动评价计算方式。因为以自动评价指标来评测的话，模型的评测结果在很多时候要比人写的caption的评测结果还要好，但人工评价的结果显示，实际上模型的caption相比于人写的caption还有很大差距。



我们可以看到：在表1中，实验使用了微软的COCO数据集，3中评价标准的得分，谷歌NIC模型的得分和人类（Human）得分是不相仲伯的。在表2中，基于不同数据集统一计算BLEU-1得分，NIC的得分和人类得分也比较接近。

但谷歌补充了一个基于人类判断的实验，邀请人类对于自己生成的标准语句进行评级，一共分成4个等级，分成“描述没有错误”、“描述中有点小错误”、“多少还是和图像相关”和“和图像无关”4个等级，分别得分从4到1。那么，真实的对比就来了：



上图中，x坐标是BLEU得分，y坐标是表示积累分布（也就是说，输出的描述语句集合中，有百分只多少的得分大于当前的x）。其中：

Flickr-8k：NIC 表示的是使用NIC模型在Flick8k测试集上跑的结果的得分曲线；

Pascal：NIC 表示的是是使用NIC模型在Pascal测试集上跑的结果的得分曲线；COCO-1k：NIC 表示的是是使用NIC模型在COCO-1k测试集上跑的结果的得分曲线；

Flickr-8k：ref 表示的是另一篇论文的结果的得分曲线，这里作为一个基准；

Flickr-8k：GT 表示的是对Flickr-8k图像的人工标注语句同样进项人工分等级评价的结果。

由此可见：虽然自动裁判员BLEU-4认为NIC模型的得分超出了人类得分，但是如果让人类来当裁判员，NIC还差得远。

# 参考文献

CNN：<https://blog.csdn.net/m0_37490039/article/details/79378143>

n-gram：<https://zhuanlan.zhihu.com/p/32829048>

m-RNN：<https://zhuanlan.zhihu.com/p/22520434?refer=intelligentunit>