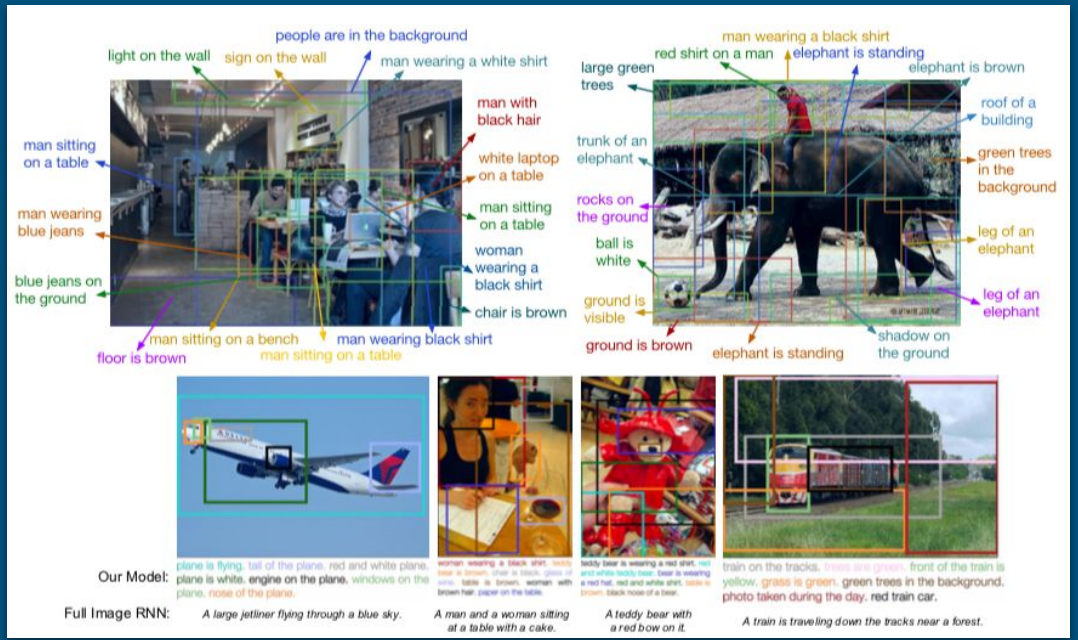
densecap实施

Dense caption-经典论文

[1]DenseCap: Fully Convolutional Localization Networks for Dense Captioning: *Justin Johnson, Andrej Karpathy, Li Fei-Fei*

[2]Dense Captioning with Joint Inference and Visual Context：Linjie Yang Kevin Tang Jianchao Yang Li-Jia Li

[3]A Hierarchical Approach for Generating Descriptive Image Paragraphs: Justin Johnson, Andrej Karpathy, Li Fei-Fei



Dense caption任务是image caption和object detection任务相结合。

其中object detection任务一般采用的是Faster R-CNN 网络，Caption Model则采用single LSTM 或 muitl LSTM进行构建，这两个部分间也有多种结合的方式，这也是不同论文探索的各种方向上采取的一系列策略决定的。

Dense caption[1]

[1]DenseCap: Fully Convolutional Localization Networks for Dense Captioning: *Justin Johnson, Andrej Karpathy, Li Fei-Fei*

densecap模型的结构如图，网络由4部分构成：

- CNN 特征提取网络
- Localization Layer
- Recognition Netwok
- RNN caption Model

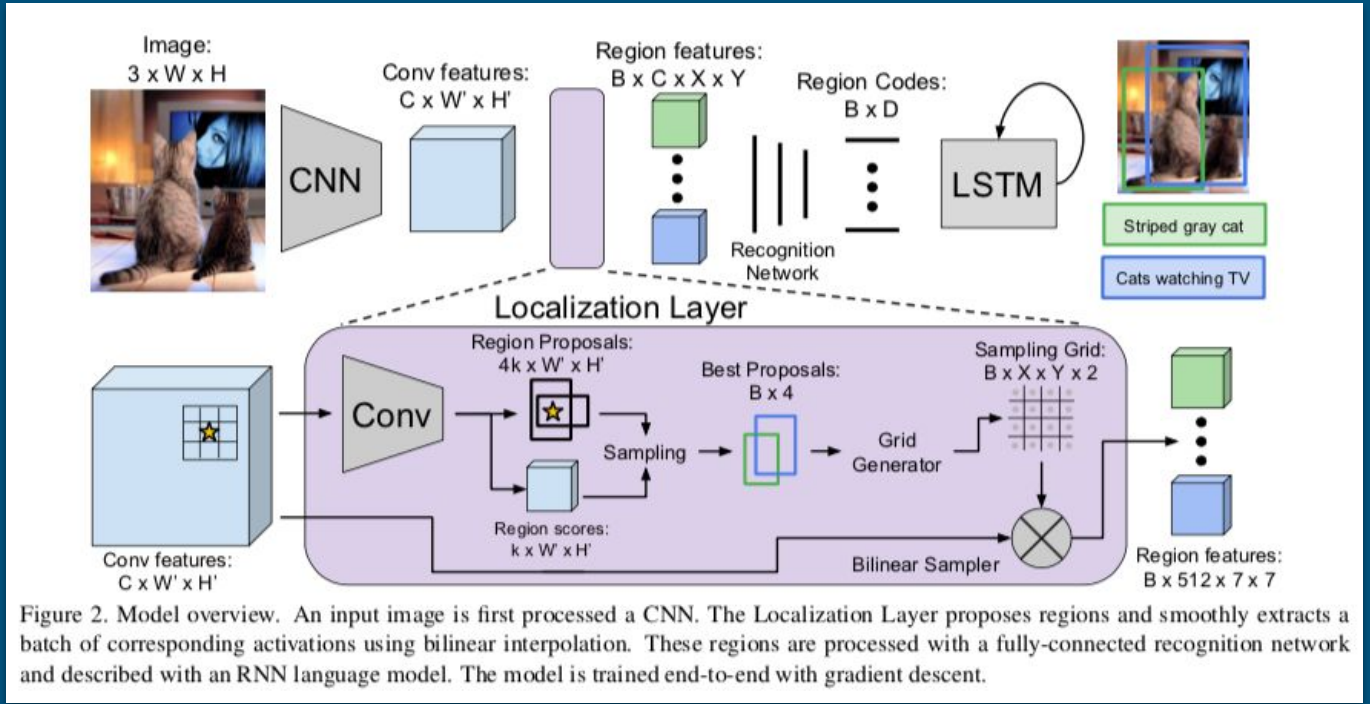
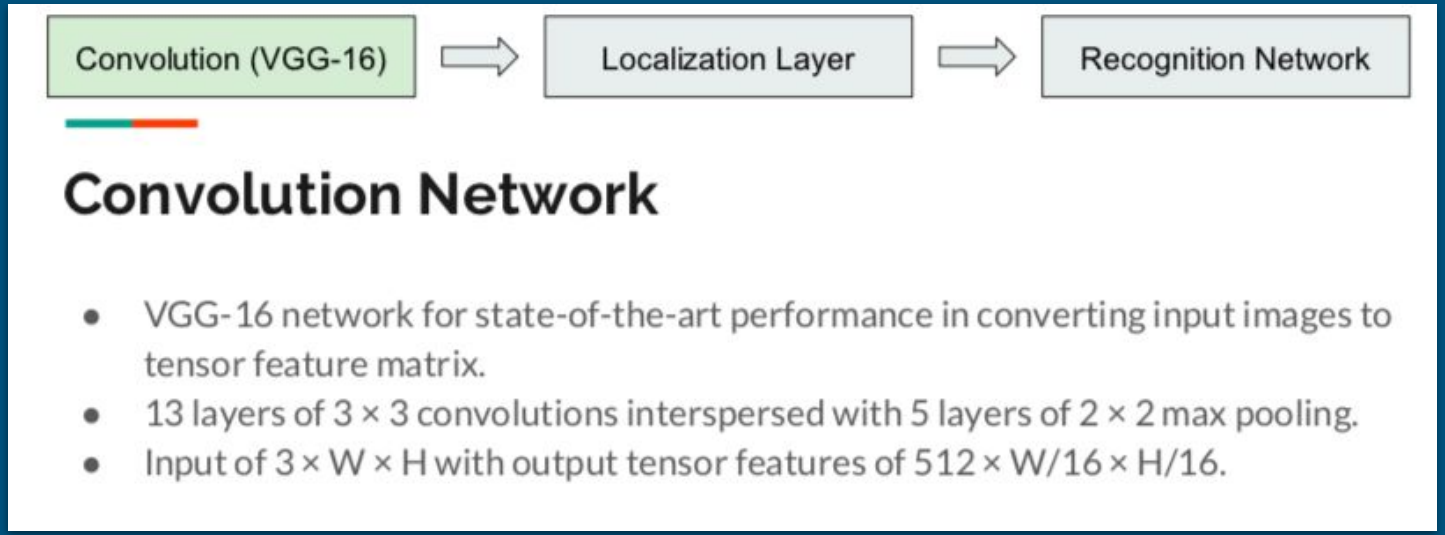


Figure 2. Model overview. An input image is first processed a CNN. The Localization Layer proposes regions and smoothly extracts a batch of corresponding activations using bilinear interpolation. These regions are processed with a fully-connected recognition network and described with an RNN language model. The model is trained end-to-end with gradient descent.

Dense caption[1]

图像特征提取网络论文中选取的是VGG-16,对输入图片提取出其特征向量，输入到**Localization layer**



Dense caption[1]

Localization layer 识别感兴趣的区域并且从每一个区域平滑得提取一个固定尺寸的图像向量表示。本文基于Faster R-CNN但是用双线性插值代替了 RoI pooling 机制。这一变化的好处在于打开了预测仿射或者变形候选区域的可能性而不仅仅是bounding box。

输入 => 来自VGG16卷积网络的特征图谱C×W′×H′C×W′×H′(任意size)
输出 => 输出B个候选区域的表征向量(定长)，在每个特征向量都包含下面三个关键信息：

1. 候选区域的坐标：输出形式是一个B×4B×4的矩阵，每行代表一个候选区域的坐标
2. 候选区域的置信分数：一个长度为BB的一维列向量，向量内每个元素都给出了候选区域的得分。得分越高说明越可能是真实区域
3. 候选区域的特征：输出形式为B×C×X×YB×C×X×Y的特征集合，这里B代表区域个数，X×YX×Y表示特征图谱的大小(注意，这里的size已经是固定的)，CC代表特征的维度

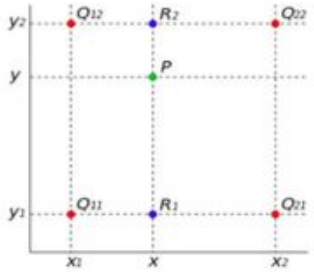
Dense caption[1]

Localization layer 中的 Faster R-CNN 用双线性插值代替了原有的RoI pooling机制，但在结构中还是称其为pooling层

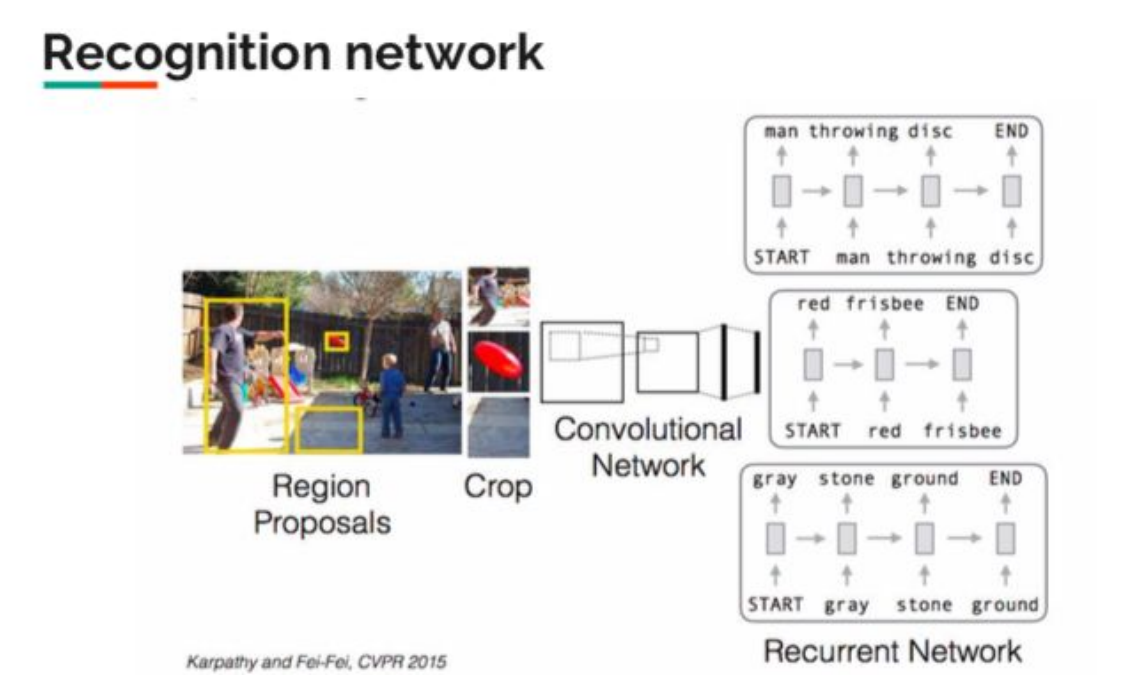
Bilinear interpolation

- Allows for backpropagation of error to previous layers.

$$f(x,y) = \frac{1}{(x_2-x_1)(y_2-y_1)}\big(f(Q_{11})(x_2-x)(y_2-y) + f(Q_{21})(x-x_1)(y_2-y) + f(Q_{12})(x_2-x)(y-y_1) + f(Q_{22})(x-x_1)(y-y_1)\big)$$



Dense caption[1]

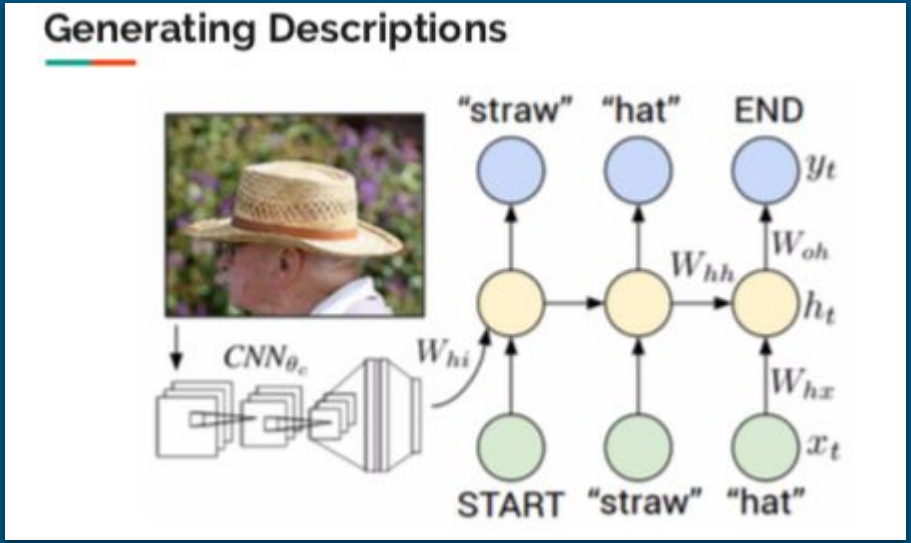


Recognition Network网络的作用就是将来自定位层的每个region的特征faltten为一个向量并将其传至两个全连接层，这两层都是使用了ReLU单元并且使用了Dropout。每一个region从而能够产生一个维数为D = 4096的code，里面包含了这一个region中的visual appearance。所有positive region的codes被收集到了一个大小为B × D的矩阵中，然后被传送至RNN语言模型。

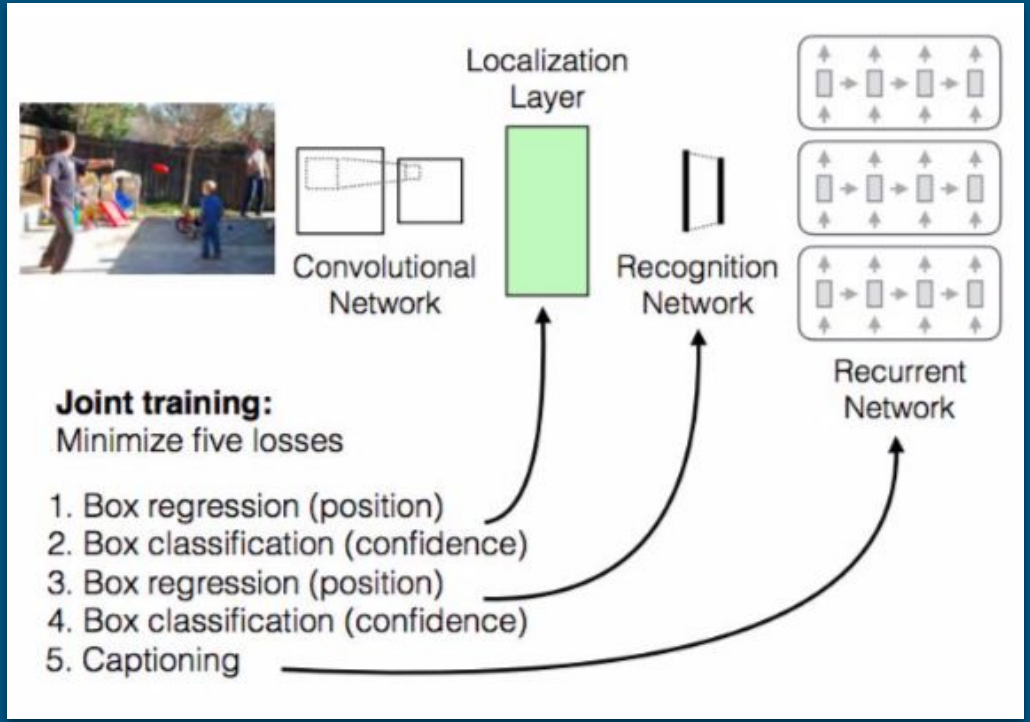
另外，Recognition Network也可以再次完善每个region proposal的置信度和位置。这两者来自于每个region所对应的D维code的线性变换。

Dense caption[1]

caption model将得到的图像区域特征向量 输入到我们的RNN模型中并生成描述 这个过程和我们在show and tell的im2txt模型中是一样的



Dense caption[1]



整个网络模型的Loss function：

采样后的positive和negative regions里，对于置信度，文中使用的是binary logistic loss（被使用两次）；

于box regression，使用的是smooth L1 loss来变换坐标空间（被使用两次）；

语言模型中每个时刻用的是 cross-entropy 。

有的损失函数都是通过RNN中的batch size和sequence length来正则化的。

Dense caption[1]

(binary logistic loss)

$$l(w,b) = -\sum_{i=1}^{m} lnP(y_i|x_i;w,b)$$

$$P(y=1|x) = \frac{e^{w^T x+b}}{1+e^{w^T x+b}}$$

$$P(y=0|x) = \frac{1}{1+e^{w^T x+b}}$$

(cross-entropy)

$$L(I,S) = -\frac{1}{T}\sum_{t=1}^{T} logp_t(x_t)$$

(smooth L1 loss)

$$L_{loc}(t^u,v) = \sum_{i\in\{x,y,w,h\}} smooth_{L1}(t_i^u - v_i)$$

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x|<1 \\ |x|-0.5 & \text{otherwise} \end{cases}$$

各loss的计算公式

Dense caption[1]

对于densecap的训练: 用预训练ImageNet的权重来初始化CNN，其他权重的取值来源于标准差为0.01的高斯分布。CNN中权重的训练使用了momentum为0.9的SGD，其他的权重使用了Adam算法。学习速率分别为β1=0.9β1=0.9和β2=0.99β2=0.99。CNN的fune-tuning在迭代一轮后就开始了，但是不包含 CNN中的前四个卷积层的微调。

Results (Image Retrieval)

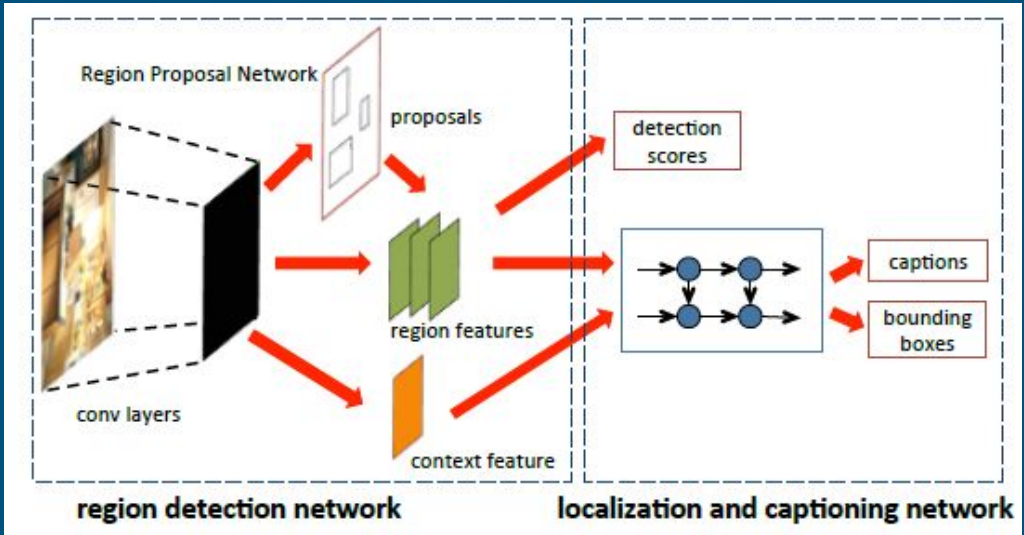
	Ranking				Localization			
	R@1	R@5	R@10	Med. rank	IoU@0.1	IoU@0.3	IoU@0.5	Med. IoU
Full Image RNN [21]	0.10	0.30	0.43	13	-	-	-	-
EB + Full Image RNN [21]	0.11	0.40	0.55	9	0.348	0.156	0.053	0.020
Region RNN [21]	0.18	0.43	0.59	7	0.460	0.273	0.108	0.077
Our model (FCLN)	**0.27**	**0.53**	**0.67**	**5**	**0.560**	**0.345**	**0.153**	**0.137**

- Performs much better than most other models when doing natural language queries to retrieve images.

Dense caption[2]

Dense Captioning with Joint Inference and Visual Context：Linjie Yang Kevin Tang Jianchao Yang Li-Jia Li

该论文的模型结构如下图：



模型由两个部分组成：
a region detection network 和
a localization and captioning network。
前者为采用Faster-RCNN的区域监测网络，后者进行关键区域定位和描述生成任务。

Dense caption[2]

Dense Captioning with Joint Inference and Visual Context：Linjie Yang Kevin Tang Jianchao Yang Li-Jia L

这篇论文是从李飞飞团队的densecap出发，在其基础上进行创新，论文提出了densecap任务的两个难点：

1、object detection获得的bounding box往往会发生堆叠重合的情况，因此要根据描述来准确定位区域；

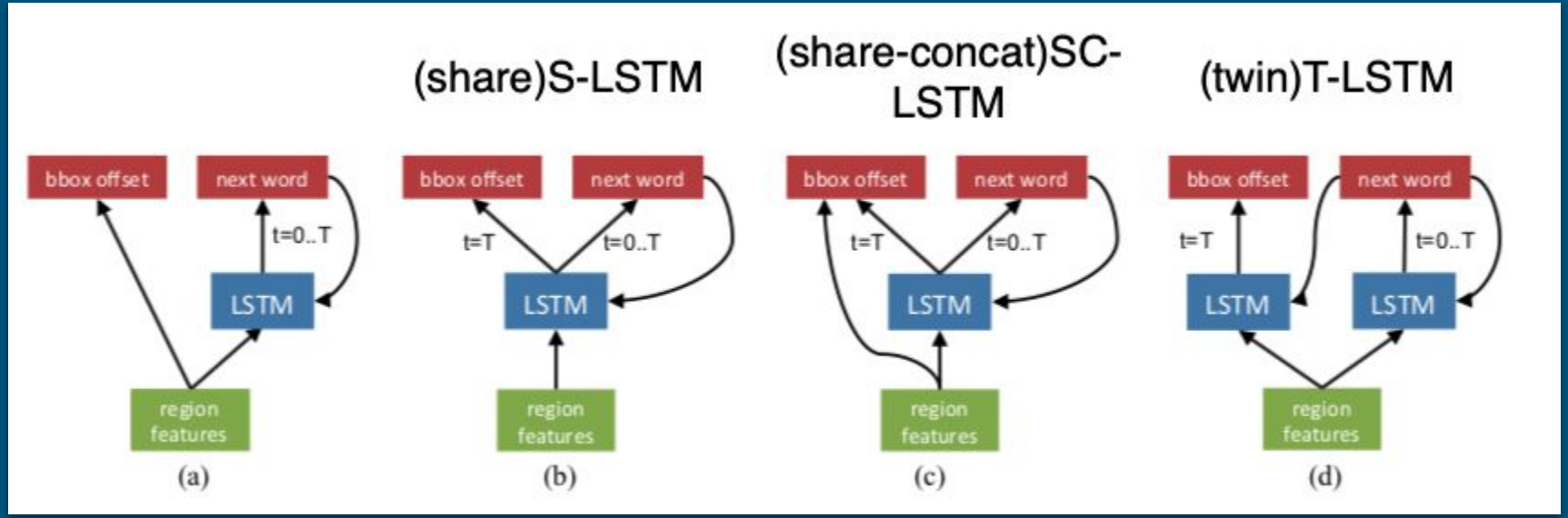
2、很多区域被描述，由于有多种描述,导致部分区域的内容变得模糊不清。

本论文中提出两个策略:**joint inference** 和 **context fusion**来对以上问题进行改进。以上描述的2种模式在效果上，同encoder-decoder结构中的attention机制有一定程度的相似，但计算方式是完全不同的，接下来我们对这2个方法进一步的了解。

Dense caption[2]

joint inference:

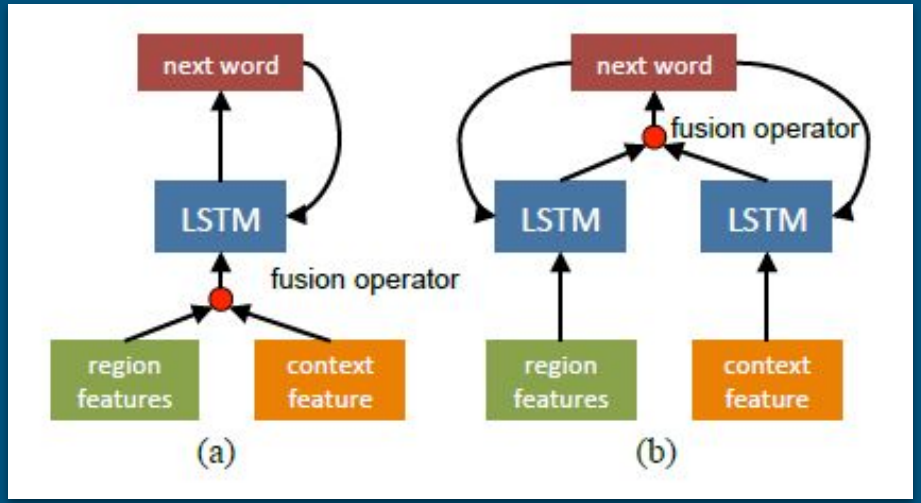
为了让bounding box的定位受相关联图片区域的描述的影响，提高定位的准确性，我们利用LSTM不仅用来生成描述，还用来对bounding box的定位进行修饰，论文列出以下很多种方式，如下所示：



im2txt 进阶 Dense captioning[2]

context fusion:

为了将context feature和local feature结合起来，论文选用RoI pooling feature作为图片的visual context feature较好的形式，但是context feature并没有直接和bounding box直接关联，仅仅使用context feature协助生成caption，再利用上面的joint inference来影响local feature。本文为了实现此功能，有如下两种形式ealy-fusion和late-fusion：



im2txt 进阶 Dense captioning[2]

有了以上两种创新结构，我们可以有多种组合形式。最后模型的损失函数如下:

$$L = L_{cap} + \alpha L_{det} + \beta L_{bbox}$$

L-caption表示caption model的交叉熵损失函数，计算的是预测下一个单词的出现概率的损失

L-det表示detection的交叉熵损失函数，计算的是前景和背景的二分类交叉熵损失

L-bbox 表示bounding box的L1正则损失函数

α、β为加权系数

Dense caption[2]

Table 3: The mAP performance of integrated models with combinations of joint inference models and context fusion structures on Visual Genome V1.0.

model		S-LSTM	SC-LSTM	T-LSTM
early-fusion	$[\cdot,\cdot]$	6.74	7.18	8.24
	$\oplus$	6.54	7.29	8.16
	$\otimes$	6.69	7.04	8.19
late-fusion	$[\cdot,\cdot]$	7.50	7.72	8.49
	$\oplus$	7.19	7.47	8.53
	$\otimes$	7.57	7.64	**8.60**

Table 4: The mAP performance of different dense captioning models on Visual Genome V1.2.

model		baseline	S-LSTM	T-LSTM
no context		6.98	6.44	8.16
late-fusion	$[\cdot,\cdot]$		7.76	**9.03**
	$\oplus$		7.06	8.71
	$\otimes$		7.63	8.52

同时论文中列出了网络结构中的joint inference和context fusion不同类型的结果对比，可以看出T-LSTM和Late-fusion是性能较高的一组。

所以在我们的网络结构中实现的是T-LSTM和Late-fusion，其中Late-fusion采用的sum的方式进行实现的。

在和densecap论文的模型的性能对比上，本论文实现的效果达到了73%的提高。

	Visual Genome V1.0			V1.2
Model	Johnson *et al.* [20]	Ours	Gain	Ours
mAP	5.39	**9.31**	73%	**9.96**

Visual Genome数据集

和MS CoCo数据集最大的不同是 标签数据在图像文字标注部分进行了大量的工作，其格式如 图：


```
{'id': 1223,
 'regions': [{'height': 182,
   'image_id': 1223,
   'phrase': 'white backdrop',
   'region_id': 4989937,
   'width': 311,
   'x': 151,
   'y': 7},
  {'height': 97,
   'image_id': 1223,
   'phrase': 'black cords on the floor',
   'region_id': 4989938,
   'width': 236,
   'x': 5,
   'y': 202},
  {'height': 127,
   'image_id': 1223,
   'phrase': 'floor of the studio',
   'region_id': 4989939,
```


标签中对于每一幅图片都进行了区域标注以及对应的描述信息，图片的regions数量从40-300不等，为我们的任务提供了丰富的标签，很好的驱动了densecap任务，并且李飞飞团队在17年提出的论文中又进一步对该数据集进行丰富，加入了paragraphs属性，即段落描述。

Dense caption工程实现

- 代码参考
  - The Faster-RCNN framework: tf-faster-rcnn by endernewton
  - Py-faster-rcnn framework: py-faster-rcnn by Ross Girshick
  - Dense Captioning with Joint Inference and Visual Context‘s code: densecap
  - Dense Cap offical code: densecap by Justin Johnson
  - Densecap by InnerPeace-Wu

我们的代码最终实现的模型为论文 Dense Captioning with Joint Inference and Visual Context 中提出的模型。Joint Inference部分实现的是T-LSTM结构context fusion则对late fusion中的sum方式进行的实现。

Dense caption工程实现

代码实现过程中，对于模型最重要的部分Faster R-CNN的代码我们选用了github上star较多的采用了tensorflow进行实现的仓库，并且在做的过程中也发现有部分代码需要从py-faster-rcnn仓库中进行引用，这部分代码主要是对RPN结构进行处理的部分。论文[2]的官方实现是caffe下实现的代码，其中模型部分的代码需要我们在tensorflow下重构。对于caption model在进行test or inference时，要采取beam search对生成的语句进行筛选，这里引用了im2txt中的代码。

由于实现的模型由几个部分组成，而这几个部分在网络训练过程中要进行调参，所以模型调参的参数是比较多的，所以必须采取统一的配置文件对参数进行设置，训练过程中对该配置文件进行调整来适应不同阶段的调参需求。参数虽然较多，但大部分可以不用调整，保持官方代码中对应的值。

由于引用了ms coco的evaluate模块(pycocoevalcap)，所以在进行val的时候需要python2.7的环境，train和inference则可以跑在任意版本下。

我们的运算设备有足够的内存和GPU，代码中设置数据集batchsize为256，在一些内存有限制的设备上则需要调小。

Dense caption工程实现

在工程实现过程中，不论简单或是复杂的代码结构，为确保每个组成部分按我们需要的dimension设置，在构建代码时候我们按照pipline由进到出对每块结构进行打印，如图：

```
name: target_sentence                ==> (256, 12)
name: rpn_bbox_outside_weights                ==> (1, 34, 45, 48)
name: pool5                ==> (256, 7, 7, 1024)
name: rpn_bbox_inside_weights                ==> (1, 34, 45, 48)
name: proposal_rois                ==> (9, 5)
name: head                ==> (1, 34, 45, 1024)
name: clss                ==> (256,)
name: rpn_cls_score_reshape                ==> (1, 408, 45, 2)
name: anchors                ==> (18360, 4)
name: cont_sentence                ==> (256, 12)
name: cls_prob                ==> (256, 2)
name: gt_boxes                ==> (262, 5)
name: rpn_bbox_pred                ==> (1, 34, 45, 48)
name: rpn_cls_score                ==> (1, 34, 45, 24)
```

这样通过查看我们打印的结构，能利于我们在对模型进行优化带来便利。

Dense caption训练调参

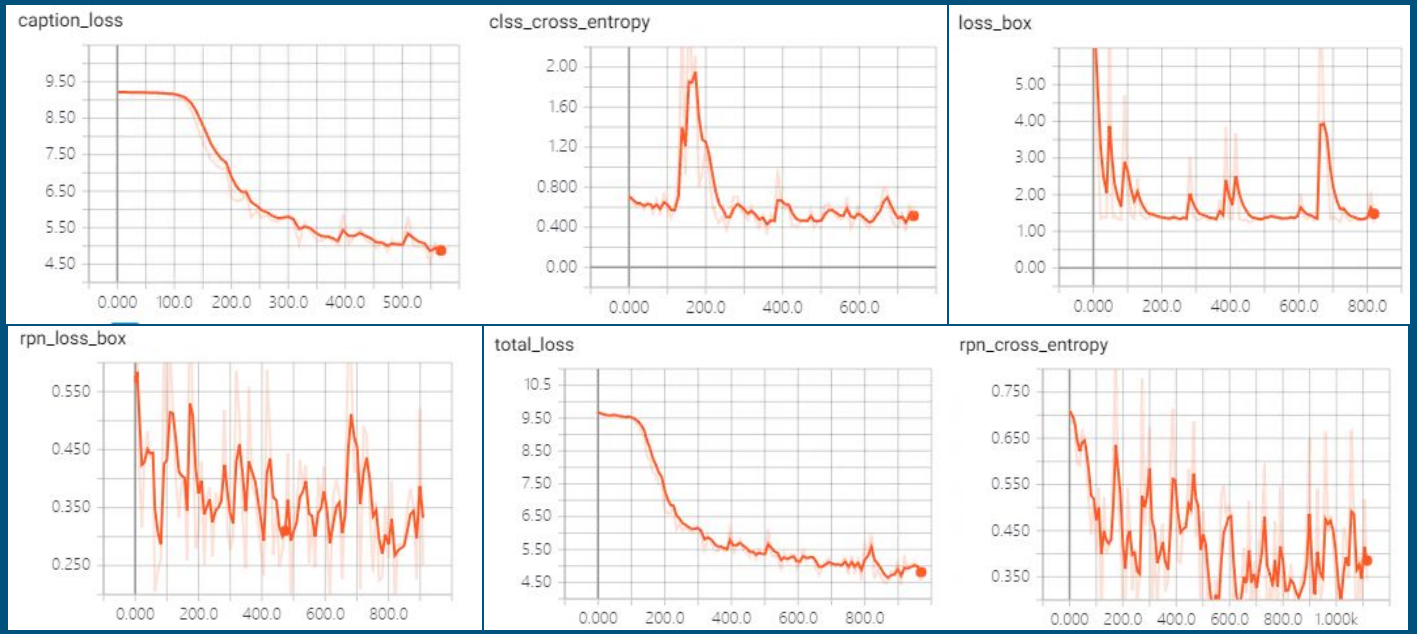
模型训练过程中的可调整的参数如下：

参数	值	备注
OPTIMIZER	sgd_m、adam	优化器
LEARNING_RATE	default:0.001	学习率
LR_DIY_DECAY	default:False	是否手动调整学习率衰减
RESNET.FIXED_BLOCKS	default:3	固定detection权重
CONTEXT_FUSION	default:False	是否开启CONTEXT_FUSION模式

其中优化器经过我们2组实验同时进行发现，使用任意一个都是可行，所以后面默 认使用sgd_m

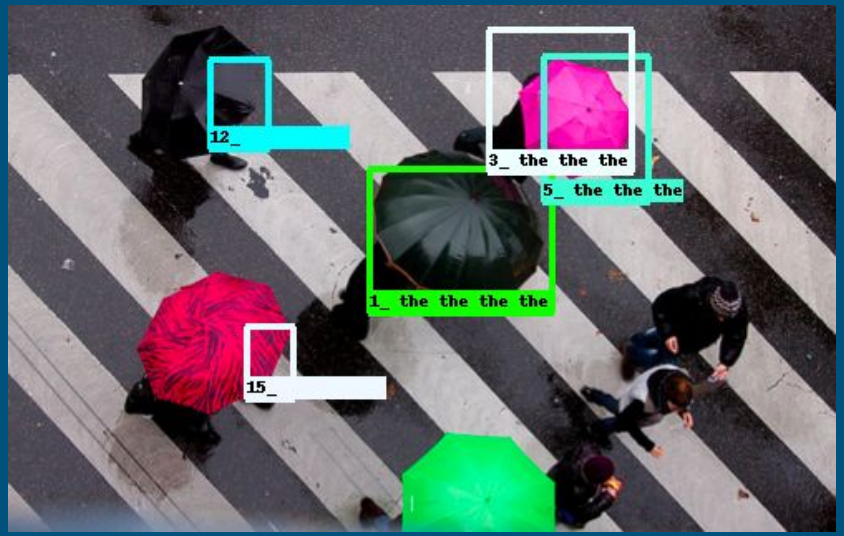
Dense caption训练调参

训练的过程分几个阶段进行，开始阶段会引入Faster R-CNN的预训练模型，第一阶段不对detection结构的参数进行调整，调整其他结构的参数，训练的收敛非常快（初始学习率为0.001）：



Dense caption训练调参

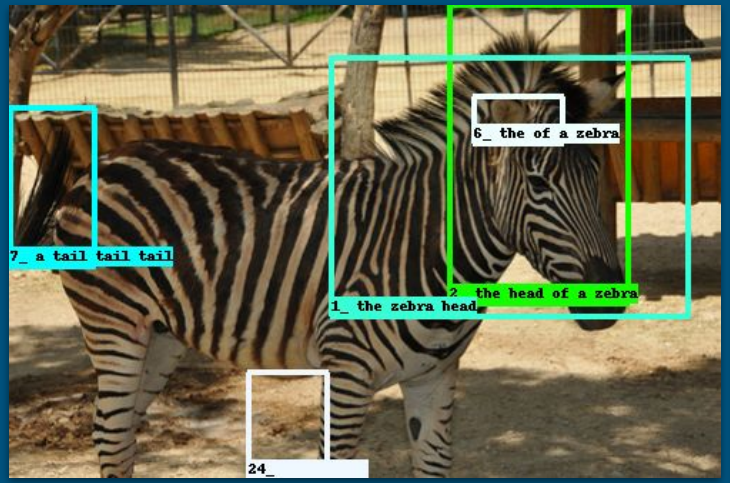
在第一个阶段模型各项loss收敛至平稳后，则开始对detection model的参数进行训练调整，同时需要把学习率调¼，继续训练至loss收敛。前2个阶段中我们用tensorboard可视化训练中的图像可以看到，模型在detection部分的提升是较为明显的。



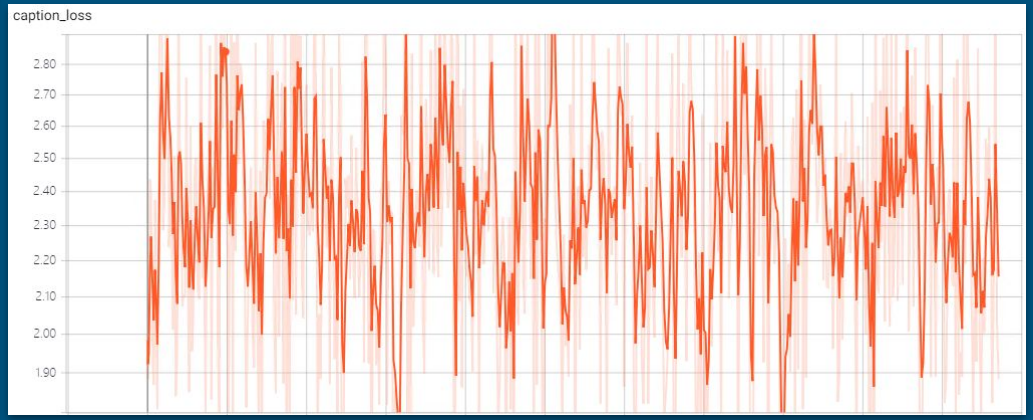
Dense caption训练调参

在对detection model进行调参至收敛后，则开始对caption model进行调参数，这里调整的设置主要是context fusion的开启，并锁定 cnn网络的参数，学习率继续衰减¼,同样训练至loss收敛。

最后阶段的训练则在上阶段的基础上取消对cnn网络参数的锁定，学习率继续衰减¼，开启训练。



网络50K迭代训练captionloss如图，值在2.4起伏不定，没有收敛



Dense caption 模型Evaluate

我们模型的evaluate采用了coco-caption: https://github.com/tylin/coco-caption 提供的评价机制，由于我们的模型是基于Python3搭建，而evaluate部分的代码需要运行在python2环境下，并且需要安装java-1.8.0+。

在工程实施中发现即便跑在python2环境下也容易出现IOError: [Errno 32] Broken pipe: Python错误这是因为:这个错误通常是在Linux下使用管道操作造成的;当管道的另一头输出部分提前关闭，而管道输入部分的内容没有完全输出，并且输入部分的内容很大，超过默认缓冲区大小，那么就会出这种错误。

解决这个问题的办法我们采用的是在pycocoevalcap/meteor 目录下创建data/paraphrase-en.gz文件来缓存输入。

右图是我们的模型20K训练时的mAP=7.9

```
im_detect: 995/1000 0.691s 0.002s
im_detect: 996/1000 0.691s 0.002s
im_detect: 997/1000 0.690s 0.002s
im_detect: 998/1000 0.690s 0.002s
im_detect: 999/1000 0.690s 0.002s
im_detect: 1000/1000 0.690s 0.002s
write to result.json
Evaluating detections
tokenization...
computing Meteor score...
caption scoring finished, takes 22.980038 seconds
mean ap computing finished, takes 0.841872 seconds
mean match ratio is 0.130
ap matrix
[[0.25425104 0.19542625 0.14363081 0.08075681 0.04145642 0.01956658]
 [0.21976282 0.17106741 0.12773903 0.07308363 0.03823357 0.0184915 ]
 [0.16631899 0.13158726 0.10135024 0.06087363 0.03299479 0.0160083 ]
 [0.10166215 0.08352587 0.06610737 0.04204854 0.0249769  0.0122456 ]
 [0.04013165 0.03469341 0.02922575 0.02105684 0.01501212 0.0081083 ]]
mean average precision is 7.905
```

Dense caption成果

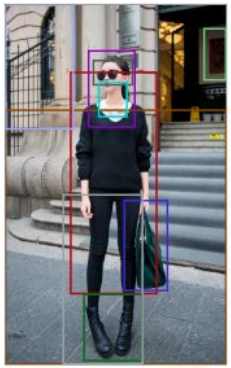
由于模型中由多个不同的结构构成，所以模型的参数较多，并且在调参过程中分了几个不同的阶段对参数进行调试，较im2txt调参过程更加复杂。(右图为训练50K的模型效果。)

另外目前虽然loss没有达到较小的收敛区间，一直在2.X左右，但网络效果已经呈现不错的性能，mAP的得分为8.05。

生成的句子比较丰富，但是我们希望能生成比较流畅完整的句子。

部署的环境：Centos7.1 / 1G/ 1CPU

a man wearing a blue shirt. a woman wearing sunglasses. a pair of sunglasses. the head of a man. the man is wearing a black shirt. a white shirt on a woman. a woman wearing sunglasses. a yellow sign on the wall. the hand of a person. the shirt is black. a man in a black shirt. a black chair in the background. the woman has blonde hair. a building in the background. the hand of a man. people in the background. the arm of a man.



实验效果

a woman wearing a black jacket. the bag is green. black sunglasses on a woman. a pair of black boots. a woman walking on the sidewalk. a woman wearing sunglasses. the woman is wearing black pants. picture on the wall. black metal railing on the side of the building. the woman is wearing a necklace.

Dense caption进阶

A Hierarchical Approach for Generating Descriptive Image Paragraphs: Justin Johnson, Andrej Karpathy, Li Fei-Fei

这是李飞飞团队对densecap领域提出的第二篇论文，文中提出的模型对输入图像可以生成它的自然语言段落描述，并且旨在利用图像和Paragraphs的组合结构，起结构如下。

Region Detector：通过目标检测网络对输入图像提取出图像中的propose regions of interest区域向量。

Region Pooling：然后在这些区域向量中聚合特征(max pooling)以产生富有表达图像语义的合并表示。（其中论文提出了相关同期一些工作，表示max pooling不会丢失需要表达的重要信息）

Hierarchical Recurrent Network，该结构包含Sentence RNN和Word RNN

Sentence RNN：每个timestep都接收池化区域特征向量作为输入，其初始状态值为0，该结构决定在结果段落中生成多少句子，并为每个句子产生输入主题向量。

Word RNN：根据输入的句子主题向量，生成每个句子的单词。最终生成一个段落描述。

Dense caption进阶

论文中提出的模型结构如图:

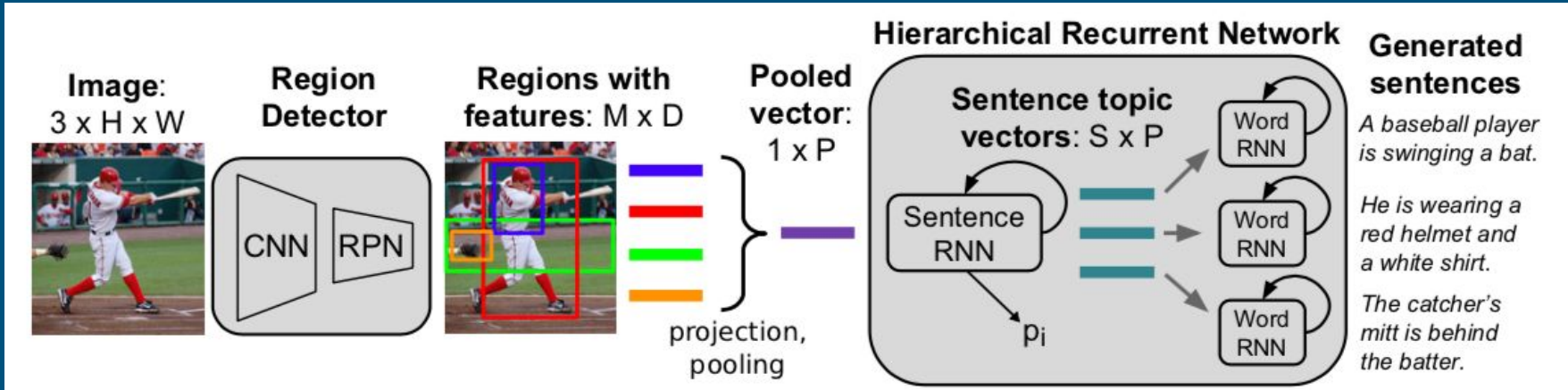


Figure 2. Overview of our model. Given an image (left), a region detector (comprising a convolutional network and a region proposal network) detects regions of interest and produces features for each. Region features are projected to $\mathbb{R}^P$, pooled to give a compact image representation, and passed to a hierarchical recurrent neural network language model comprising a sentence RNN and a word RNN. The sentence RNN determines the number of sentences to generate based on the halting distribution p_i and also generates sentence topic vectors, which are consumed by each word RNN to generate sentences.

Dense caption总结

我们当前的模型，也有很多需要进行优化的地方比如loss收敛，生成了重复的句子，生成句子的时间较长导致的服务响应较慢。产生这些问题的原因既有模型方面的问题，也有系统架构的问题导致，如果模型要进一步用于生产环境，要在架构上的改进较多，不过从整体性能上提高来看这些改进对性能提升的比重较低，还是要把重心放在模型改进上；模型方面由于我们采用的论文的模型使用的部分结构计算上较为耗时，所以对内存和CPU的占用较高，这也是我们目前Demo较慢的原因。

“A Hierarchical Approach for Generating Descriptive Image Paragraphs: Justin Johnson, Andrej Karpathy, Li Fei-Fei ”这篇论文为我们提供了后续改进的方向，同时论文中也提出李飞飞团队为该任务制作的数据集，他们在visual genome基础上补充了图片的paragraphs段落描述标签，为我们进行下阶段的任务提供了数据保障，同时论文采用的目标检测网络在目前看来，还可以继续改进，如Mask R-CNN能获得更好的目标区域检测性能，对于