



# 看图说话机器人

Image Captioning



余杨、冯宇琨、王磊



# 任务规划

---

1. 了解任务背景、数据集、评价指标
  2. 经典论文通读
  3. 了解经典网络架构
  4. NIC2架构(im2txt)训练 & fine-tuning , 搭建v0.1版本可用系统
  5. Im2txt改进:encoder & Attention , 训练至收敛
  6. 整合改进模型到系统后端
  7. 文档整理
- 代码仓库地址 : <https://github.com/stoensin/IC> (包含分支对应工作中不同部分)

- 1.1) COCO数据准备  
数据下载和数据处理
- 1.2) 数据集了解  
清楚数据和数据格式
- 1.3) 评价指标  
了解几种评价机制
- 1.4) 论文阅读  
方向选型
- 1.5) 模型规划设计  
基于im2txt框架进行
- 1.6) 系统架构规划  
采用web server&app
- 1.7) 开发环境准备  
用git协作和版本控制
- 1.8) 运行im2txt  
作为初始版本 v0.0  
基于此基础先行搭建系统

## ▼ 2) 开发&训练

### ▼ 2.1) 搭建 web server 和web前端

- 2.1.1) web server
- 2.1.2) web 前端

### • 2.2) 模型改进

attention机制  
encoder改进

### • 2.3) 训练模型

可视化查看loss

### • 2.4) 评估模型

基于几种评价机制对模型  
进行评估

## ▼ 3) 测试&发布

- 3.1) 测试系统

# 任务甘特图



# Image caption任务

a woman standing in front of a wall holding a cell phone .



图像描述生成(Image Caption)是一个融合 计算机视觉(cv)、自然语言处理(nlp)的综合问题, 简单来说就是翻译一副图片为一段描述文字。该任务对于人类来说非常容易, 但是对于机器却非常具有挑战性, 它不仅需要利用模型去理解 图片的内容并且还需要用自然语言去表达它们之间的关系。除此之外, 模型 还需要能够抓住图像的语义信息, 并且生成人类可读的句子。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。

# 传统做法

---

图像描述生成可以认为是一种动态的目标检测，由全局信息生成图像摘要。早先的做法例如《Baby Talk》, 《Every picture tell a story》等都是利用图像处理的一些算子提取出图像的特征，经过SVM分类等等得到图像中可能存在的目标object。根据提取出的object以及它们的属性利用CRF或者是一些人为制定的规则来恢复成对图像的描述。这种做法非常依赖(图像特征的提取)和(生成句子时所需要的规则)。

效率不高，且耦合高，难以达到端对端。

# 相关数据集

---

数据集名称	数据量 (train-val-test)	训练用时	用途
PASCAL	1k	One hour	Debugging and testing
Flickr 8k	6k-1k-1k	Few hours	
Flickr 30k	28k-1k-1k	Less than a day	Training
MS COCO	82k-40k-40k	A couple of days	
SBU	1M	Very long	
Visual Genome	100K	Weeks	

# MS-COCO数据集

COCO的全称是Common Objects in COntext, 是微软团队提供的一个可以用来进行图像识别的数据集。MS COCO数据集中的图像分为训练、验证和测试集。COCO通过在Flickr上搜索80个对象类别和各种场景类型来收集图像, 其使用了亚马逊的Mechanical Turk (AMT), COCO数据集现在有3种标注类型: object instances (目标实例), object keypoints (目标上的关键点), 和image captions (图像标注), 均使用JSON文件存储。本文使用的数据为 image captions (2014)。

基本数据格式如:

Info、image、  
licenses  
为三类标注数据的  
共享字段:

```
{
  "info": info,
  "licenses": [license],
  "images": [image],
  "annotations": [annotation],
}
```

```
info{
  "year": int,
  "version": str,
  "description": str,
  "contributor": str,
  "url": str,
  "date_created": datetime,
}
image{
  "id": int,
  "width": int,
  "height": int,
  "file_name": str,
  "license": int,
  "flickr_url": str,
  "coco_url": str,
  "date_captured": datetime,
}
license{
  "id": int,
  "name": str,
  "url": str,
}
```

# MS-COCO数据集-Image Caption

Image caption 类型的数据中带有的annotation字段用来存储描述图片的语句。每个语句描述了对应图片的内容，而每个图片至少有5个描述语句(有的图片多于这个数量)。

annotation定义如下：

```
annotation{
  "id": int,
  "image_id": int,
  "caption": str
}

{
  "image_id": 32117,
  "id": 133,
  "caption": "Several metal balls sit in the sand near a group of people."
}
```

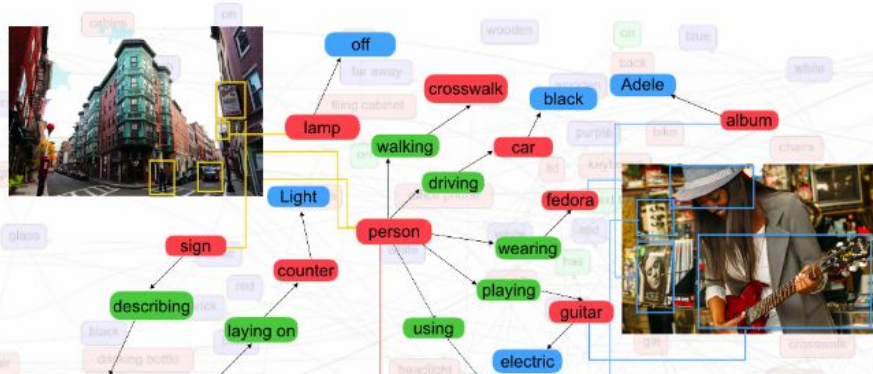


# Visual-Genome 数据集

Li Fei-Fei团队设计了这样一个数据集:它不但包括了图像本身,更包括了图像内对象之间的关系等众多数据(包括objects、attributes、relationship等)。并希望通过这些数据能够推动“认知”这一问题在CV领域的发展。

Visual Genome一共包括了 **108K张图片**，平均每张图片内包含了 **35个object**，和 **26个attributes**，以及 **21对object之间的relationship pair**。本数据集的图片取自MS COCO 和 YFCC100M。

作者们还将其中所有的object、attributes、relationships和在region descriptions与question answer pairs中的名词短语都映射到了WordNet synset上。从而让打通了从CV到Knowledge乃至NLP之间的连接通道。



108,077 Images

## 5.4 Million Region Descriptions

## 1.7 Million Visual Question Answers

### 3.8 Million Object Instances

## 2.8 Million Attributes

## 2.3 Million Relationships

## Everything Mapped to Wordnet Synsets

100

## 摘自COCO Captioning Leaderboard

	CIDEr-D	METEOR	Rouge-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4	SPICE	date
Tencent/Vision	1.196	0.277	0.573	0.795	0.635	0.485	0.363	0.213	2017-08-07
Team: Description: multi-attention and RL Link:									
panderson@MSR/ACRV	1.179	0.276	0.571	0.802	0.641	0.491	0.369	0.215	2017-07-22
Team: Description: Bottom-Up and Top-Down Attention Link:									
DEEPAI	1.173	0.275	0.572	0.786	0.629	0.485	0.368	0.213	2017-07-22
CASIA_IVA	1.170	0.274	0.572	0.786	0.629	0.484	0.368	0.213	2017-07-22
Watson Multimodal	1.147	0.270	0.563	0.781	0.619	0.470	0.352	0.207	2017-03-17
Team: Description: Attention models trained with reinforcement learning. Link: <a href="https://arxiv.org/abs/1612.00563">https://arxiv.org/abs/1612.00563</a>									
QMUL-VISION	1.102	0.264	0.554	0.778	0.612	0.459	0.337	0.203	2017-06-27
SenmaoYe	1.053	0.270	0.552	0.742	0.577	0.443	0.341	0.200	2017-04-29
Team: Description: attentive linear transformation Link:									
MSM@MSRA	1.049	0.266	0.552	0.751	0.588	0.449	0.343	0.197	2016-10-25
Team: Description: Image captioning by exploiting image attributes.									

# 评价指标

名称	描述	补充
CIDEr-D	CIDEr 是专门设计出来用于图像标注问题的。这个指标将每个句子都看作“文档”，将其表示成 Term Frequency Inverse Document Frequency(tf-idf)向量的形式，通过对每个n元组进行(TF-IDF) 权重计算，计算参考 caption 与模型生成的 caption 的余弦相似度，来衡量图像标注的一致性的。	CIDEr-D 是修改版本，为的是 对于 gaming 问题更加鲁棒。什么是 Gaming 问题：就是一个句子经过人工判断得分很低，但是在自动计算标准中却得分很高的情况。为了避免这种情况，CIDEr-D 增加了截断(clipping)和基于长度的高斯惩罚。
METEOR	METEOR 是基于BLEU进行了一些改进，其目的是解决一些 BLEU 标准中固有的缺陷。使用 WordNet 计算特定的序列匹配，同义词，词根和词缀，释义之间的匹配关系，改善了BLEU的效果，使其跟人工判别共更强的相关性。	METEOR 也包括其他指标没有发现一些其他功能，如同义词匹配等
Rouge-L	ROUGE 是出于召回率来计算，所以是自动摘要任务的评价标准，	Rouge-L基于最长公共子序列(LCS)的度量方法。LCS 是同时出现在两个句子中且顺序相同的一组词。将两个句子的 LCS 长度记为 $l(c_i, s_{ij})$ ，ROUGE-L 通过计算 F-measure (F1-score)来生成。
BLEU	BLEU是Bilingual Evaluation Understudy的缩写。这个计算标准在图像标注结果评价中使用是很广泛的，但是它的设计初衷并不是针对图像标注问题，而是针对机器翻译问题，它是用于分析待评价的翻译语句和参考翻译语句之间n元组的相关性的。直白地说，它的核心思想就是：机器翻译语句与人类的专业翻译语句越接近就越好。	引入一个简洁性惩罚呢？这是因为BLEU倾向于更短的句子，这样精度分数就会很高。为了解决这个问题，使用了乘以一个简洁性惩罚来防止很短的句子获得很高的分数。
SPICE	SPICE 也是专门设计出来用于 image caption 问题的。全称是 Semantic Propositional Image Caption Evaluation。前面四个方法都是基于 n-gram 计算的，所以 SPICE 设计出来解决这个问题。PICE 使用基于图的语义表示来编码 caption 中的 objects, attributes 和 relationships。	它先将待评价 caption 和参考 captions 用 Probabilistic Context-Free Grammar (PCFG) dependency parser parse 成 syntactic dependencies trees然后用基于规则的方法把 dependency tree 映射成 scene graphs。最后计算待评价的 caption 中 objects, attributes 和 relationships 的 F-score 值。

# 模型篇

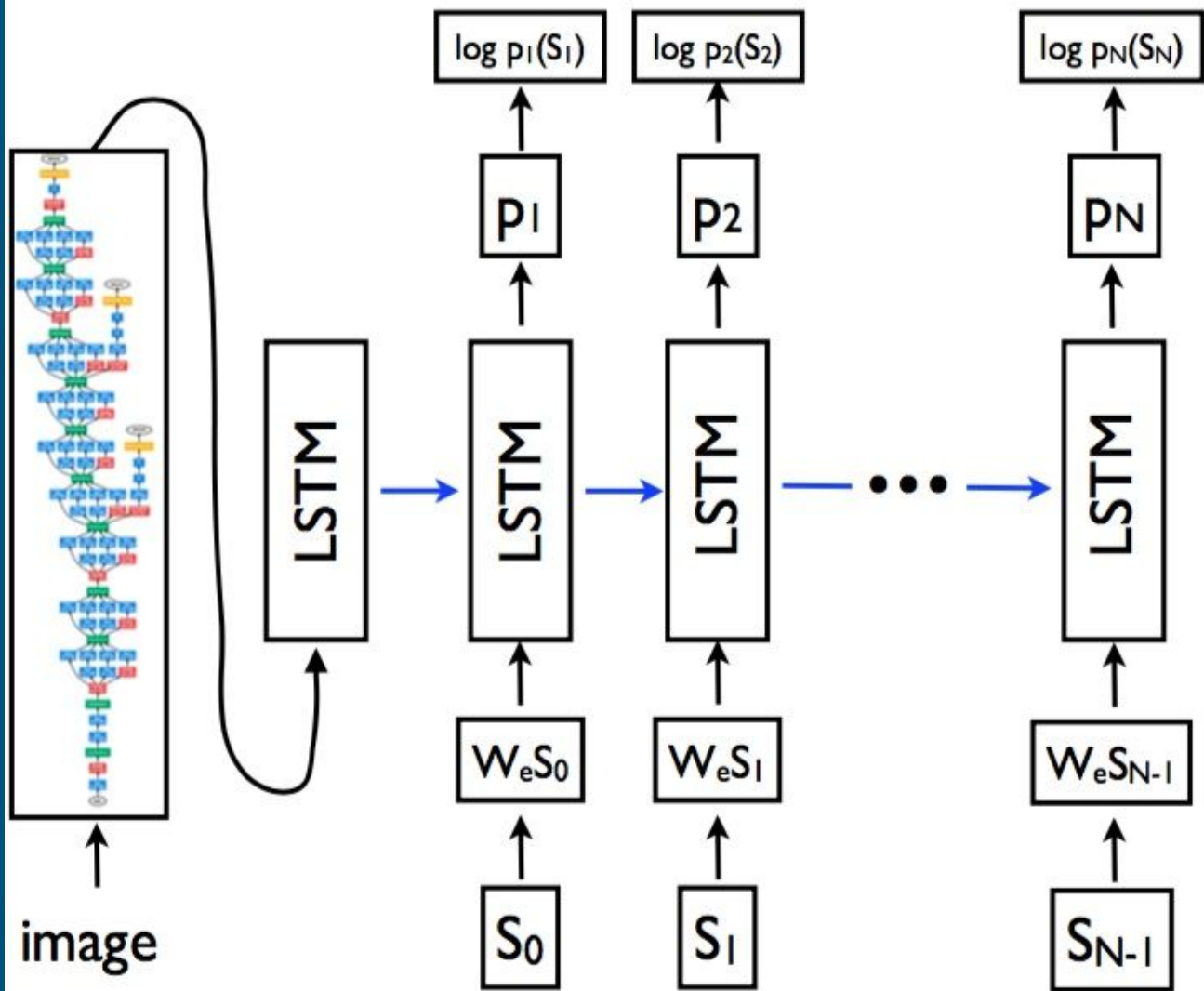


# 经典模型架构

NIC(2014) : Show and Tell: A Neural Image Caption Generator

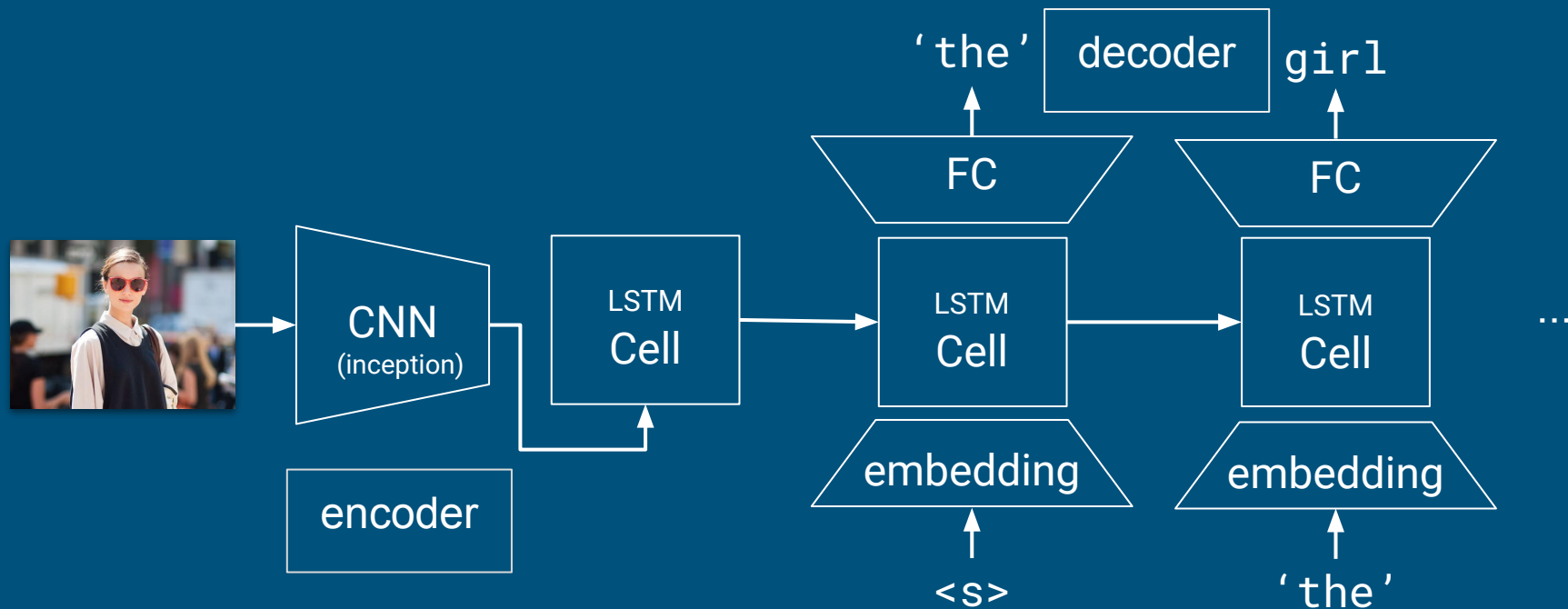
论文采用encoder-decoder结构, encoder部分用CNN网络提取到图片的特征向量, decoder基于前者的输出用LSTM逐个输出字符, 论文采用beam search来减小搜索空间获取最佳语句。

2015年Google团队携NIC参加MSCOCO Image Captioning Challenge ,赛后总结了一些经验并提出了对NIC的改进 ( 使用得分更高的特征提取网络,并直接载入预训练的网络, 使用batch norm,提升LSTM层数和CELL深度,LSTM随机初始化,使用dropout, 用更少的Beamsearch参数预测 ),另外他们也从实验中得出大语料训练出来的word embedding对模型作用极小, 在不同数据集下模型得分差异较大;Google随后把代码开源到了Tensorflow下的Im2txt。



# im2txt模型架构

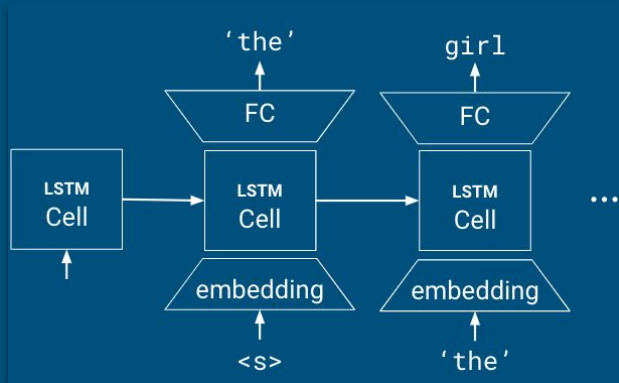
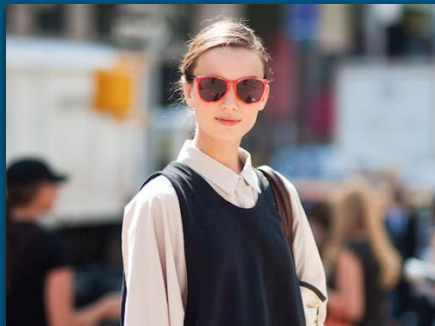
Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge[arxiv:1609.06647]



# im2txt之模型细节

encoder结构中使用的了InceptionV3网络作特征提取的, 网络已经在LSVRC-2012-CLS图像分类数据集上进行了多轮训练, 这里只需加载预训练好的模型即可, 并且使用了batch normalization 提高网络训练的性能。

训练网络期间, CNN网络参数在一开始是固定不 进行训练, 在LSTM网络训练得到较平稳的loss曲线后, 取消CNN参数的固定, 让CNN与LSTM一起训练调参, 论文中提到这样做带来的效果是 颜色可以被正确描述出来。



decoder结构中的LSTM, 它的开始时刻的输入是在encoder中的提取的特征向量(8, 8, 2048), 它的常规输入是word embedding vectors, 每步的输出是词汇表中所有单词的概率。训练目标就是最大化全部 训练样本的对数似然之和。

$$\log P(S|I; \theta) = \sum_{t=0}^N \log P(S_t | S_0, S_1, \dots, S_{t-1}, I; \theta)$$

$$\theta^* = \arg \max_{\theta} \sum_{(I, S)} \log P(S|I; \theta)$$

# im2txt之beam search

---

在seq2seq中获取decoder的输出结果, 常用方法有贪心搜索(Greedy Search)集束搜索(Beam Search)。贪心搜索只选择了概率最大的一个, 而集束搜索则选择了概率最大的前k个。这个k值也叫做集束宽度(Beam Width)。集束搜索本质上也是贪心的思想, 当k等于1的时候就是贪心搜索, 只不过它考虑了更多的候选搜索空间, 因此可以得到更多的输出结果。

在im2txt模型的测试生成过程中, 根据一张图片P, 推断联合概率最大的序列作为输出, 但要计算全部序列的概率然后选出概率值最大的序列这么做却不可行的, 因为每个位置都有整个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, 所以这里就用到了beam search来缩小搜索空间, 获取最佳的输出语句。

google在[Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge](#) 中提到了对他们的beam width 取了从1至20范围内多组值进行训练, 后来发现值为3效果最佳, 所以在im2txt模型中caption\_generator.py设置了beam\_size=3 。

在我们的实验中对beam width 的值修改了1~20内的多组值, 发现值3的时候生成语句效果的确比较好。



TABLE 1

Scores on the MSCOCO development set for two models: NIC, which was the model which we developed in [46], and NICv2, which was the model after we tuned and refined our system for the MSCOCO competition.

Metric	BLEU-4	METEOR	CIDER
NIC	27.7	23.7	85.5
NICv2	<b>32.1</b>	<b>25.7</b>	<b>99.8</b>
Random	4.6	9.0	5.1
Nearest Neighbor	9.9	15.7	36.5
Human	21.7	25.2	85.4

TABLE 2

BLEU-1 scores. We only report previous work results when available. SOTA stands for the current state-of-the-art.

Approach	PASCAL (xfer)	Flickr 30k	Flickr 8k	SBU
Im2Text [18]	25	55	48	11
TreeTalk [14]				19
BabyTalk [3]				
Tri5Sem [16]				
m-RNN [27]				
MNLM [29] <sup>5</sup>		56	51	
SOTA	25	56	58	19
NIC	<b>59</b>	<b>66</b>	<b>63</b>	<b>28</b>
Human	69	68	70	

# im2txt之模型成果

数据摘自 Show and Tell [arxiv:1609.06647]

# im2txt之输入数据处理

*im2txt/data/build\_mscoco\_data.py*

处理流程：根据字幕文件提取图片ID和对应语句 => 分配比例 => 建立词汇表 & 图片数据预处理 => 生成TFrecord  
mscoco数据集分配如下：


















train\_dataset = 100% of mscoco\_train\_dataset + 85% of mscoco\_val\_dataset.

val\_dataset = 5% of mscoco\_val\_dataset (for validation during training).

test\_dataset = 10% of mscoco\_val\_dataset (for final evaluation).

数据集中annotations下的json文件生成训练RNN需要的词汇表word\_counts

生成的TFrecords文件包含256个训练集、4个验证集和8个测试集：

 train-00253-of-00256	 test-00000-of-00008
 train-00254-of-00256	 test-00001-of-00008
 train-00255-of-00256	 test-00002-of-00008
 val-00000-of-00004	 test-00003-of-00008
 val-00001-of-00004	 test-00004-of-00008
 val-00002-of-00004	 test-00005-of-00008
 val-00003-of-00004	 test-00006-of-00008
 word_counts.txt	 test-00007-of-00008
	 train-00000-of-00256

其中，每个tfrecord文件中包含约2300条记录，每条记录都是一个序列化的SequenceExample协议，其格式如下：

Context:

image/image\_id: MSCOCO图像ID

image/data: 关于图像完整RGB信息的字符串

feature\_lists:

image/caption: 包含带标记的字幕语句的字符串列表

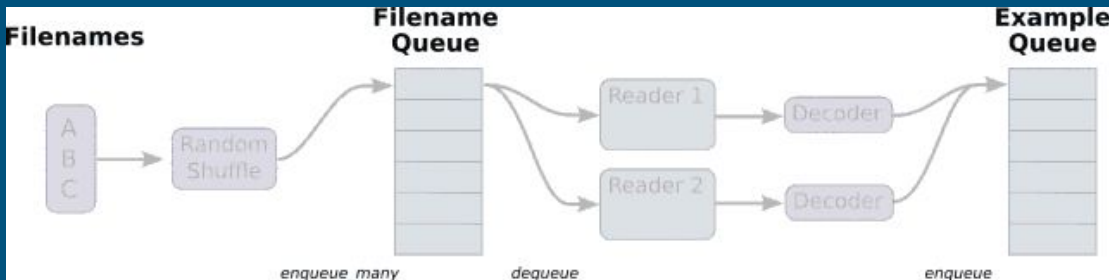
image/caption\_ids: 与字幕语句对应的id列表

至于训练数据这么处理的好处：用硬盘空间换内存，减少内存开销，加载速度也更快；可以随意改组调用数据；数据在Tensorflow中异步预处理更加友好。

# im2txt训练

对于训练数据的读取, im2txt在`show_and_tell_model.py`的`build_inputs`方法中调用了基于Tensorflow框架的文件队列功能, 实现了数据pipeline, 在模型训练过程中, tensorflow会按照队列内的文件自动调度, 前面我们进行预处理的TFrecords文件会被框架自动载入内存, 性能也很也非常不错。

这个流程分为两个阶段: 第一个阶段将输入文件打乱, 并在文件队列入列, 然后Reader从文件队列中读取一个文件, 同时文件队列出列这个文件, Reader同时对文件进行解码, 然后生产数据样本, 并将样本在样本队列中入列, 可以定义多个Reader并发地从多个文件同时读取数据。从样本队列中的出列一定量的样本数据即可以用于一个训练过程。如图:



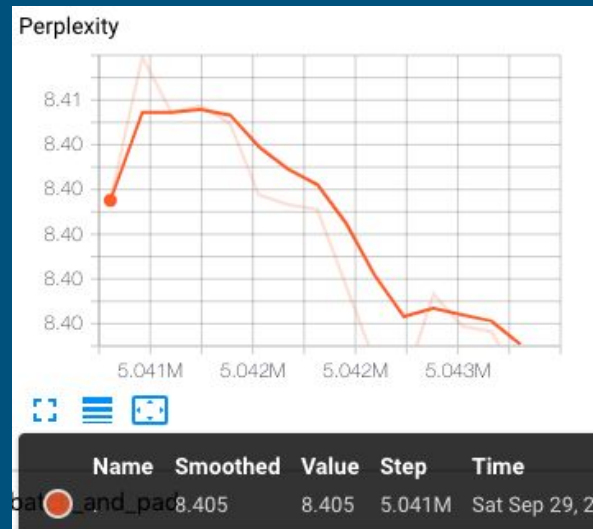
只对decoder部分的训练

```
INFO:tensorflow:Starting Queues.
INFO:tensorflow:global_step/sec: 0
INFO:tensorflow:global step 1: loss = 9.4074 (14.688 sec/step)
INFO:tensorflow:Recording summary at step 1.
INFO:tensorflow:global step 2: loss = 9.0141 (3.290 sec/step)
INFO:tensorflow:global step 3: loss = 7.5687 (0.950 sec/step)
INFO:tensorflow:global step 4: loss = 13.8843 (1.015 sec/step)
INFO:tensorflow:global step 5: loss = 9.9048 (1.024 sec/step)
INFO:tensorflow:global step 6: loss = 9.4803 (0.940 sec/step)
INFO:tensorflow:global step 7: loss = 9.3944 (0.918 sec/step)
INFO:tensorflow:global step 8: loss = 8.1105 (0.947 sec/step)
INFO:tensorflow:global step 9: loss = 8.0235 (0.939 sec/step)
INFO:tensorflow:global step 10: loss = 16.0693 (0.972 sec/step)
INFO:tensorflow:global step 11: loss = 12.1528 (0.920 sec/step)
INFO:tensorflow:global step 12: loss = 11.2044 (0.883 sec/step)
INFO:tensorflow:global step 13: loss = 8.3027 (0.958 sec/step)
INFO:tensorflow:global step 14: loss = 6.9101 (0.971 sec/step)
INFO:tensorflow:global step 15: loss = 9.5765 (0.905 sec/step)
INFO:tensorflow:global step 16: loss = 9.2846 (0.955 sec/step)
INFO:tensorflow:global step 17: loss = 8.5414 (0.918 sec/step)
INFO:tensorflow:global step 18: loss = 7.9638 (0.986 sec/step)
INFO:tensorflow:global step 19: loss = 8.1441 (0.838 sec/step)
INFO:tensorflow:global step 20: loss = 7.1030 (0.836 sec/step)
INFO:tensorflow:global step 21: loss = 6.4069 (0.827 sec/step)
INFO:tensorflow:global step 22: loss = 7.7179 (0.834 sec/step)
INFO:tensorflow:global step 23: loss = 7.4246 (0.854 sec/step)
INFO:tensorflow:global step 24: loss = 7.2192 (0.834 sec/step)
INFO:tensorflow:global step 25: loss = 8.7324 (0.865 sec/step)
INFO:tensorflow:global step 26: loss = 9.2637 (0.897 sec/step)
INFO:tensorflow:global step 27: loss = 8.0228 (0.818 sec/step)
INFO:tensorflow:global step 28: loss = 7.3397 (0.861 sec/step)
INFO:tensorflow:global step 29: loss = 6.7069 (0.856 sec/step)
INFO:tensorflow:global step 30: loss = 6.4322 (0.898 sec/step)
INFO:tensorflow:global step 31: loss = 6.3612 (0.866 sec/step)
INFO:tensorflow:global step 32: loss = 6.1956 (0.857 sec/step)
INFO:tensorflow:global step 33: loss = 6.6166 (0.856 sec/step)
```

# im2txt训练和评估

由于训练该模型耗时较长，所以本文对于模型的训练过程采用了github上共享出来的5M steps的预训练模型，在此基础上fine-tune。

如下图所示，在step=5.041M这点上，我们进行了一个调整：取消CNN网络权重的固定，使其开始和RNN一起被训练更新，所以我们可以看到loss曲线从一个点陡然上升，然后开始下降，Perplexity(困惑度)也是反应了网络性能的改变，并且在此之前，loss曲线的平均值在1.5浮动，而开始fine-tune后，loss逐渐升高，在网络训练了一段时间后才开始下降，并且有一点也出现很大变化，此时网络训练1step耗时从0.8 second 开始上升，逐渐变化至6 sec/step，在经过很长时间训练后才下降。



# im2txt之推断

其中im2txt的推断实现(im2txt/run\_inference.py)流程如下:

从checkpoint路径restore保存的模型

=> 从输入图片提取"lstm/initial\_state:0"的state值 & 加载词汇表

=> 输入initial\_state到LSTM, 从生成第一个字符开始, 获取到"softmax:0"的值 和 "lstm/state:0"的值, 在限制字幕maxlength下, 执行循环操作: 以LSTM上一步产生的output 和 state作为输入; 使用 beam search(size=3)对每一个输出的向量进行范围搜索, 根据概率大小排序选取TOP-3的值产生最佳的字符的输出, 直至生成完整语句向量

=> 根据词汇表, 把上一步输出的向量转为可读语句

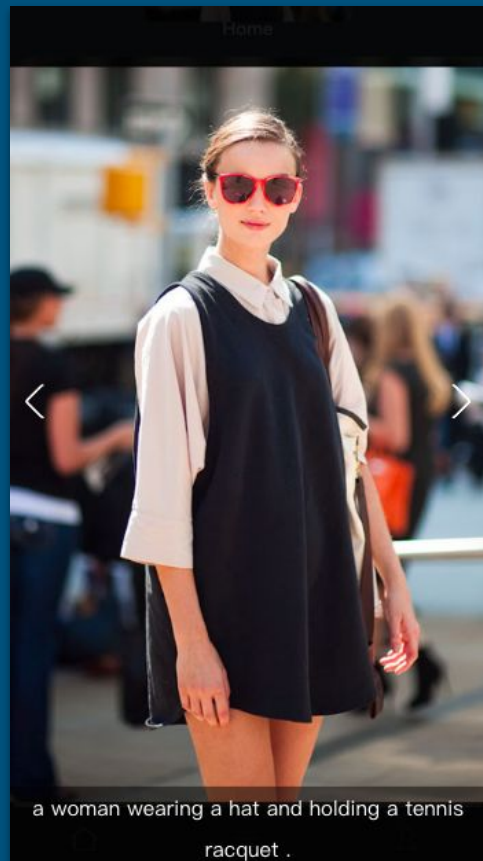
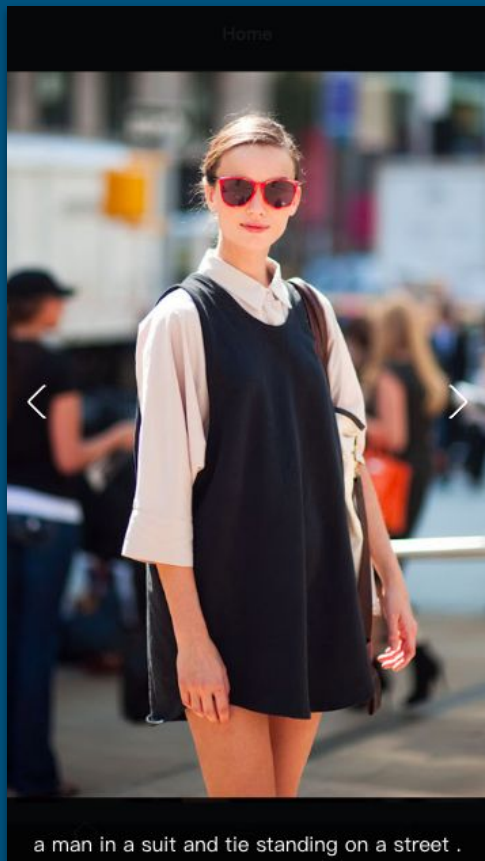
以上为im2txt流程中的推断实现流程; 我们的模型作为API服务的实现逻辑也和推断的逻辑一样, 所以在我们的系统API服务中就采取了以上流程。

# im2txt实验对比

左图为模型训练2M steps Perplexity $\approx$ 8.9

右图为模型训练5M steps Perplexity $\approx$ 8.4

右图对比左图会有较好的识别效果, 但整体还是有待改进



# im2txt 改进之Encoder

---

1. 采用图片分类任务中效果更好的Inception\_V4、Inception\_resnet\_V2这样的网络代替encoder;
2. 采用目标检测任务中的Faster R-CNN网络作为encoder, 把RPN产生的结果输出到decoder;

Faster R-CNN可以对图像进行多标签分类, 提取多区域特征, 提取颜色、材质等属性特征, 结合这些特征, 可以训练出一个semantic model, 比较第一种方案, 这样获取到的“图像的高层次的抽象信息”, 对于Image caption任务或是VQA 任务来说更有价值;

3. 替代网络均采用在分类任务、检测任务中训练后的预训练模型以节省时间;

# im2txt 改进之Attention

---

论文依据:

[1][Show, Attend and Tell: Neural Image Caption Generation with Visual Attention](#)

[2][Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering](#)

[3][What value do explicit high level concepts have in vision to language problems?](#)

改进方案:

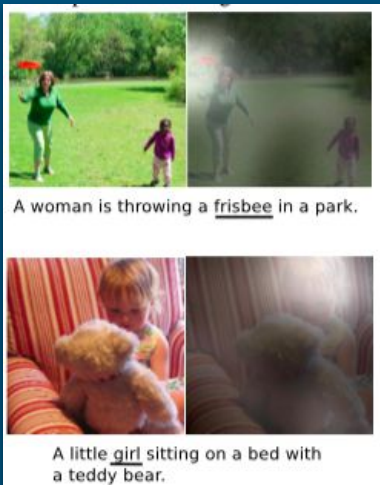
soft attention , self-attention, bottom-up attention;



# im2txt 改进之Attention[1]

*[Show, Attend and Tell: Neural Image Caption Generation with Visual Attention]*

论文中介绍了两种Attention机制, Hard Attention和Soft Attention, Attention的加入带来了模型选择特征的能力, 通过这种机制我们在模型运行时已经可以可视化出当生成对应的caption时, 模型所关注的图像区域, 如:



并且, attention 机制解决了传统encoder-decoder架构中对于长输入序列的支持(当我们将输入序列中的必要信息表示为一个固定长度的向量, 而当输入序列很长时则难以保留全部的必要信息, 尤其是当输入序列的长度比训练集中的更长时), 模型对信息进行有选择的“挑选”, 而不再依赖固定长度的信息, 所以attention机制改善了以前的模型在较长输入时性能不佳的情况。

soft attention如何实现的:

在decoder结构中通过保留LSTM编码器对输入序列的中间输出结果, 然后训练一个atend层来对这些输入进行选择性的学习并且在模型输出时将输出序列与之进行关联。换一个角度而言, 输出序列中的每一项的生成概率取决于在输入序列中选择了哪些项。

# im2txt 改进之Attention[1]-soft\_attention

具体怎么来做呢？之前是用CNN提取了固定长度的向量特征 $l$ ，实际上，我们可以利用CNN的空间特性，使用卷积层的feature map，给图片的不同位置都提取一个特征。举例来说，用

$a = \{a_1, a_2, \dots, a_L\}, a_i \in R^D$  表示我们提取的图片特征，一共  $L$  个位置，每个位置的特征为一个  $D$  维的向量，对于一个高、宽为14，通道数为256的feature map，对应的  $L = 14 \times 14 = 196$ ， $D = 256$ 。

设在第 $t$ 阶段（通俗的讲，就是生成第 $t$ 个单词时）传入Decoder RNN的上下文向量为  $z_t$ ，RNN前一阶段的隐层状态为  $h_{t-1}$ 。这个上下文向量  $z_t$  就是  $a = \{a_1, a_2, \dots, a_L\}$  的一个加权平均，具体地， $z_t$  和  $a = \{a_1, a_2, \dots, a_L\}$  的关系用下面的式子表达：

$$z_t = \sum_{i=1}^L \alpha_{t,i} a_i$$

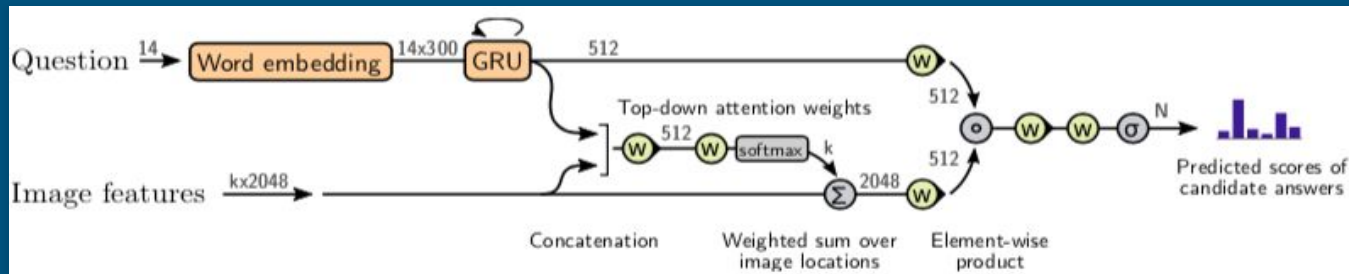
$\alpha_{t,i}$  就是衡量生成第 $t$ 个单词时，第 $i$ 个位置的图像特征所占的权重。这个权重实际是前一个隐层状态  $h_{t-1}$  和第 $i$ 个位置图像特征  $a_i$  的函数。具体的表达式为：

$$e_{ti} = f_{\text{att}}(a_i, h_{t-1})$$
$$\alpha_{t,i} = \frac{\exp(e_{ti})}{\sum_{k=1}^L \exp(e_{tk})}$$

由于  $\alpha_{t,i}$  只和已有的信息  $h_{t-1}, a_i$  有关，因此这些参数也是可以从数据中进行端到端的自动学习

# im2txt 改进之Attention[2]

[Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering]



文章提出一种自上而下与自下而上相结合的注意力模型方法，应用于视觉场景理解Image Caption和视觉问答VQA系统的相关问题。

**Bottom-Up:** 基于自下而上的注意力模型，采用 Faster R-CNN提取图像中的兴趣区域，获取对象特征；该模型的任务是获取图像兴趣区域提取图像特征，类似于对图像进行特征编码，实现 encoder阶段任务；

**Top-Down:** 基于自上而下的注意力模型（LSTM）用于学习特征所对应的权重，以此实现对视觉图像的深入理解。该模型用于学习调整特征权重，实现了图像内容的“attention”，逐词生成描述，相当于 decoder解码阶段。

# im2txt 改进之Attention[2]

*[Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering]*

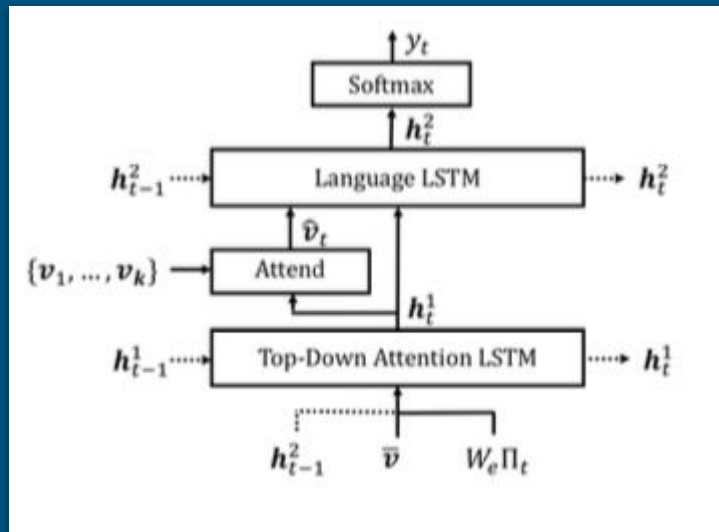
Caption model结构如图所示, 共有2个LSTM模块,

一个是Language LSTM, 另一个是Top-Down Attention LSTM,

他们都是常规标准的LSTM结构, 且互相使用对方前一个状态的

hidden state值作为输入, 其中Top-Down Attention用来

确定image feature  $v_i$  的权重, 属于soft attention机制。

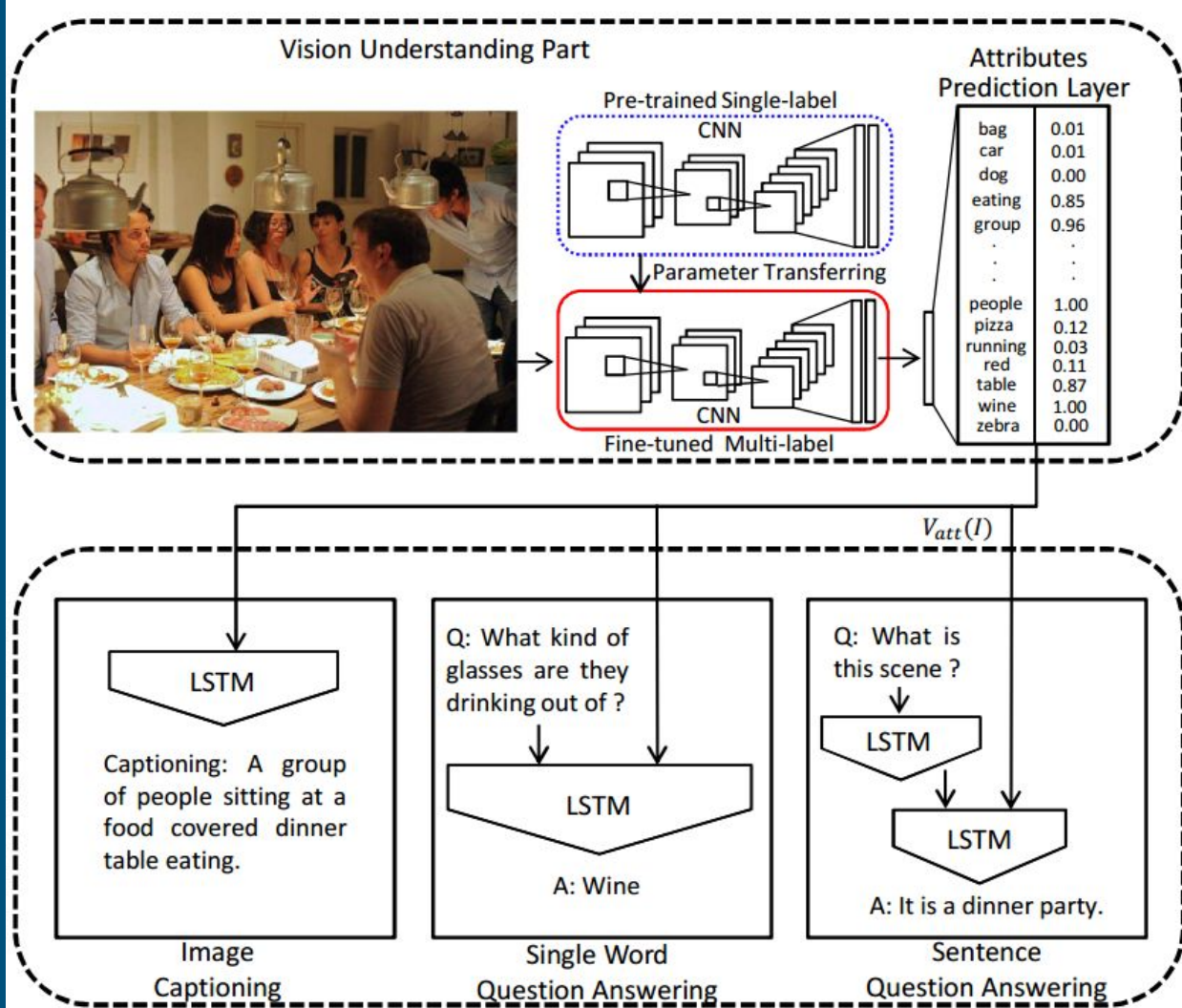


# im2txt改进之 Attention[3]

[What Value Do Explicit High Level Concepts Have in Vision to Language Problems?]

本文在CNN-RNN的基础上提出了一种利用图像高层次语义的方法，并在多个任务上验证了其有效性。用attribute vector进行finetune，用detections进行预测再综合输出，得到具有较高层语义信息的RNN输入。根据原文的流程图，可以分三个演化过程：

1. Pretrained VGG on imagenet.
2. Finetune on MSCOCO to make it a semantic model.
3. Using a detector.



# im2txt改进之Attention[3]

## 1. Finetune on MSCOCO to make it a semantic model

### 1) Attribute Vector

使提取出来的特征具有语义信息，也就是说让特征图和句子挂钩，文中使用了attribute vector。在训练集中的标注语句中提取了最常出现(至少5次)的256个词作为最具代表性的属性，这些词可以是任何词性，但是不区分时态和单复数，因为只是在标注训练集上，虽然非常可以代表训练图片的属性，缺点是词太少了，文章后面有用wordnet扩充。

### 2) Finetune multi-label

将pretrained VGG最后输出改成256-D的，进行多标签任务的分类(因为一个图片可能对应好几个属性)进行微调。这里对FC层使用xavier初始化，不同的层lr策略不同，并且使用element wise logistic loss function而不是MSE。通过上面的流程，每张图片经过网络之后输出的就已经是带有语义属性了。

## 2. Using a detector

作者也考虑了这个问题，给一张图过一次网络固然可以，很多以前的模型也是这么做的，但是语义就不完整了。比如说，我的图里有很多不同的物体和动作，如果只用一幅图直接过了那可能出来的属性就有一些预测不到，精度不够，但如果先做image detection，将BBoxes都过一遍，再结合起来，更细的粒度可能会带来更好的效果。为了computational efficiency，在经过检测网络后，将BBoxes聚类成m个，再取每一类IOU最大的k个框框，加上原图一共 $m*k+1$ 将BBoxe。这样会出现 $m*k+1$ 盒预测得到的attribute vector，对每一个属性取最大值做max pooling得到一个vector就是原文中的 $V_{att}(I)$ 了，这个向量就是将要输入到RNN中进行接下来的流程。



# im2txt 进阶 Dense captioning

- [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模型则采用single LSTM 或 mult LSTM。

# 其他模型架构

---

以下为其他效果不错的网络：

Neural Baby Talk - Lu J et al, CVPR 2018.

本文结合了image captioning的两种做法：1. 以前基于template的生成方法(baby talk)；2. 近年来主流的encoder-decoder方法(neural talk)。主要做法其实跟作者以前的工作"Knowing When to Look: Adaptive Attention via A Visual Sentinel for Image Captioning"类似：在每一个timestep，模型决定生成到底是生成extual word (不包含视觉信息的连接词)，还是生成visual word。其中visual word的生成是一个自由的接口，可以与不同的object detector对接。

Mind's Eye: A Recurrent Visual Representation for Image Caption Generation - Chen X et al, CVPR 2015.

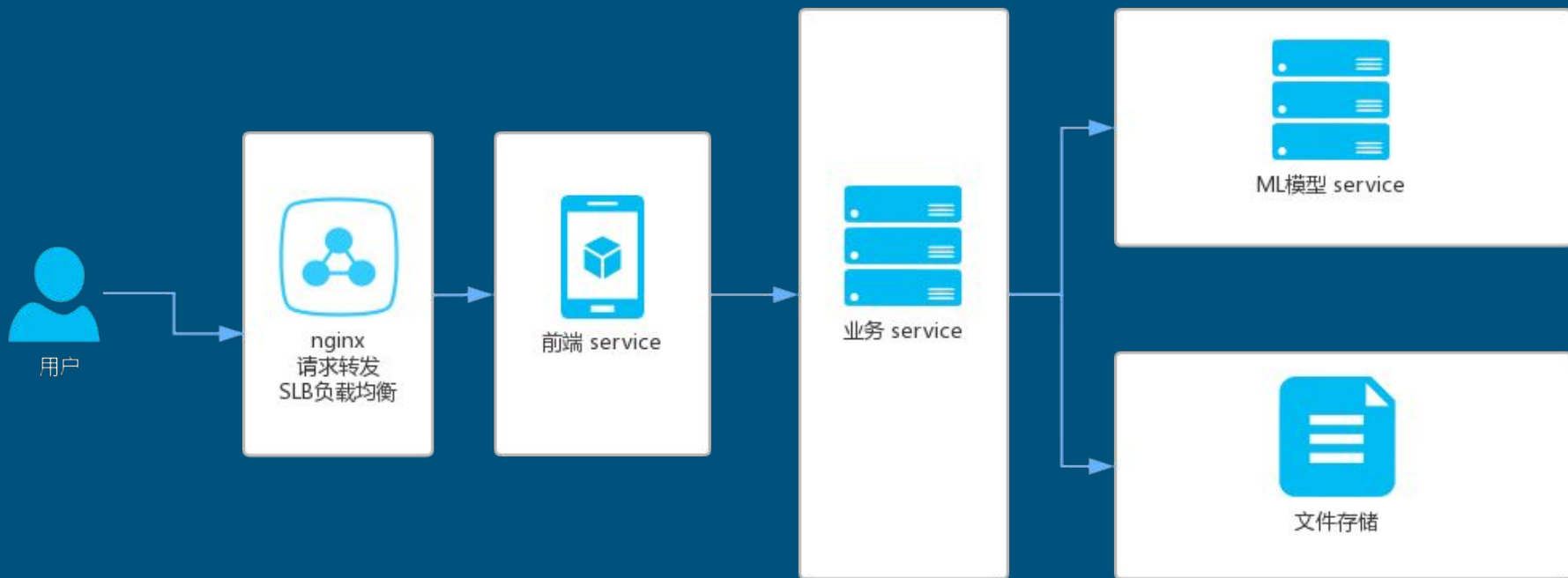
这篇论文较多地改动了Decoder部分RNN的本身的结构，使得RNN网络不仅能将图像特征翻译为文字，还能反过来从文字得到图像特征，此外还顺带提高了性能。



# 系统篇



# 系统架构



# 系统架构描述

---

1. 前后端分离、模型 业务分离；
2. 前端服务:基于VUE搭建, 提供数据展示, 数据 输入, 用户交互;功能有:列表、gallery、收藏
3. 业务服务:基于Tornado搭建, 提供API服务:用户登录登出、前端数据返回、处理输入数据、清除数据, 以及与ML模型服务通信；
4. ML模型服务:基于Tornado搭建, 提供image captioning api服务, 对输入的图片数据返回对应的 captions；
5. 服务间通信统一采用json数据格式；

目前系统源码位于仓库[https://github.com/stoensin/IC backendapi](https://github.com/stoensin/IC_backendapi)分支下