NJU_NLP_SummerCamp_2019_report_week5

> 2019.07.29-2019.08.04
> @author Eric ZHU

本周学习内容

1. 完成了上周模型的训练
2. 论文阅读：Dense Captioning with Joint Inference and Visual Context
3. 论文阅读：Actor-Critic Sequence Training for Image Captioning

> 注：两篇论文的阅读报告可以直接点击超链接进入博客阅读

其他说明

1. 关于选方向的事可能还要再慎重一点。虽然本周看的比较偏RL方向，但是自己本身可能不是非常想选RL+NLP作为拓展的方向。一方面是因为自己对RL其实了解非常肤浅，如果独立实现一个项目难度可能会比较大。另一方面，当前RL+NLP受到的评价比较两极分化。所以最好还是在对RL有更深理解后再研究这个话题。
2. 好像自己还是对CV更感兴趣一点。最近看的文章都是对Decoder进行修改。下周要不看看和Encoder相关的?
3. 下周可能会离开学校一段时间。后两周应该是看论文+写模型。预计22号左右返校，到时候再进行模型的训练工作。

附录

1. "Dense Captioning with Joint Inference and Visual Context" 阅读报告（见下页）
2. ”Actor-Critic Sequence Training for Image Captioning“ 阅读报告（见下页）
3. 7.22-7.28_Show&Tell 实验记录（见下页）

论文信息

类别	内容
论文名称	Dense Captioning with Joint Inference and Visual Context
发表时间	CVPR 2017
作者	Linjie Yang, Kevin Tang, Jianchao Yang, Li-Jia Li
关键词	Image Captioning, Dense Captioning

核心信息

文章领域及方向

CV & NLP，Dense Captioning 问题

文章概要

本文提出了Joint Inference与Context Fusion这两种新方法来解决在Dense Captioning任务中出现的问题。这两种方法利用图片的整体环境信息与物体周围的局部信息，生成对感兴趣区域（Region of Interest, ROI）的描述与检测框的偏移值，在综合得分（mean Average Precision, mAP）指标上达到了当时的state-of-the-art水平。

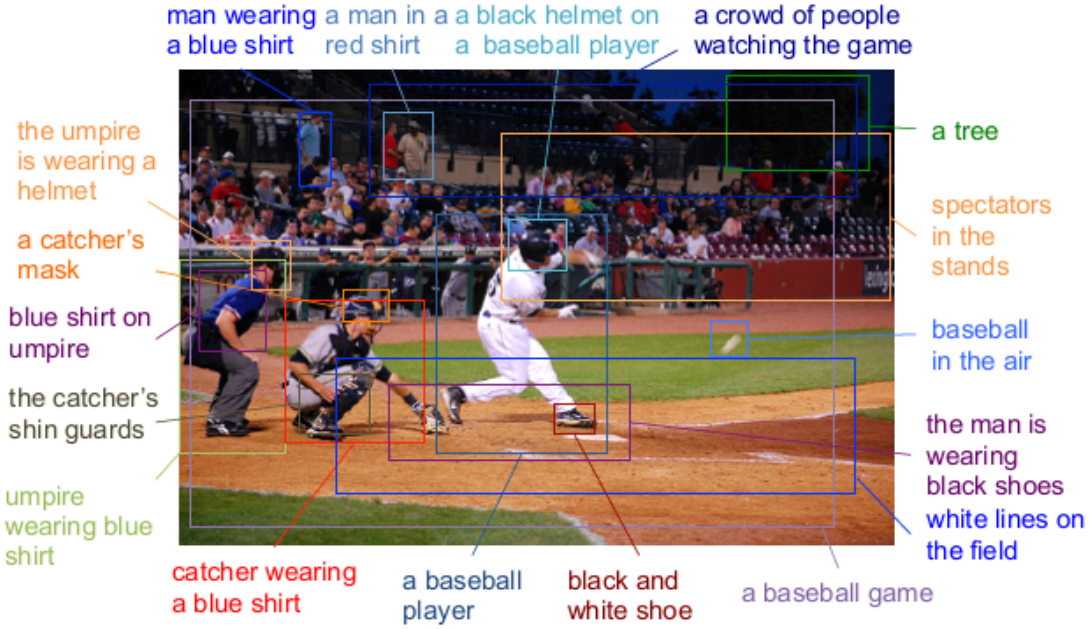
引言

数据集内对图像的描述存在的问题

1. 只关注图像中**显著的物体**，或倾向于描绘**画面整体**，不能完整地反映**对图像的完整理解。**
2. 图像描述较为**主观**，使得captioning任务的结果难以进行评价。

什么是Dense Captioning？

1. Dense Captioning是针对上述问题所提出的图像标注任务。
2. 在Dense Captioning任务中，标注者或机器将会**尽可能多地**框出图像中出现的**视觉概念物**（如：物体，物体的部分，物体间的交互情况）。
3. 与传统Captioning任务不同的是，Dense Captioning任务中物体的集合是开放的，即对物体的标注不受**有效物体集合**的限制。此外，它也关注**物体的部分**和**多物体间的交互**情况。
4. Dense Captioning相较于传统的captioning方法更加**客观**，更难受到标注者偏见的影响。下图为Dense Captioning的一个例子。



当前Dense Captioning任务的解决步骤

1. Region Proposal：通过RPN （Region Proposal Network） 生成潜在目标所在的区域框，划分出**感兴趣区域** （Region of Interest，**RoI**）。
2. 通过预测网络，对每个RoI生成以下**三个指标**：
    1. 检测得分（与物体检测任务中相似）
    2. 描述RoI内容的短语
    3. bounding-box offset（检测框偏移量，对检测框的调整参数，一般包括平移和缩放两部分）

Dense Captioning任务存在的两个难题

1. 物体检测框的**数量多**，并且常常**高度重叠。**
2. 部分物体脱离**环境**难以正确描述。

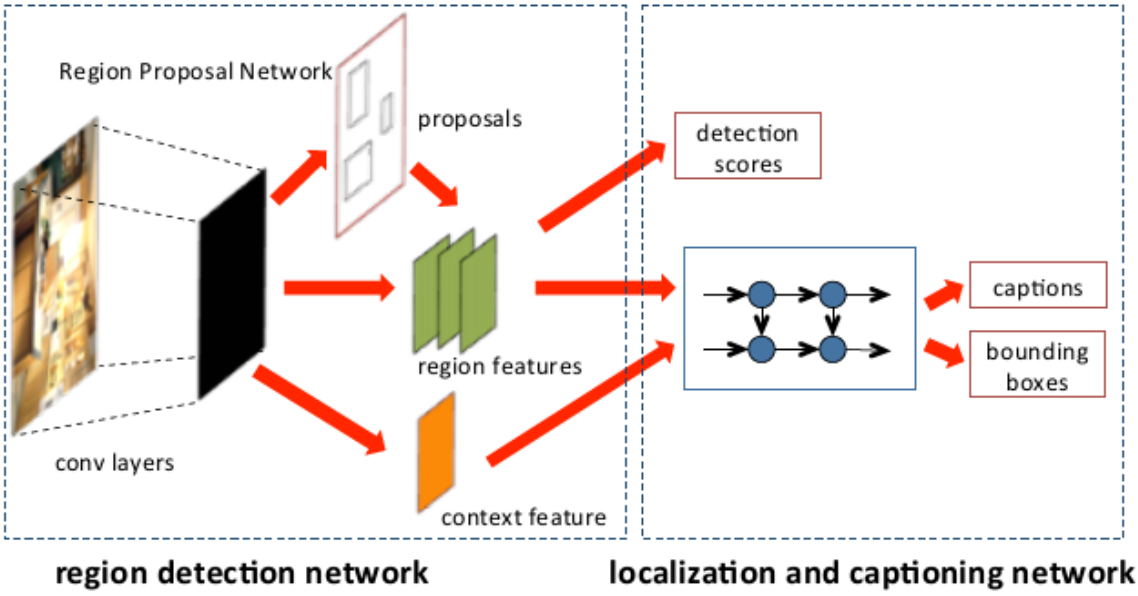
解决上述难题的两个方法

1. **Joint Inference** （联合推理，通过综合从RoI与环境中提取出的特征，对RoI的检测框偏移量进行生成与调整）
2. **Context Fusion**（环境融合，通过综合从RoI与环境中提取出的特征，来生成更好的RoI描述）

本文的贡献

1. 设计了包含两个关键方法的网络结构来解决Dense Captioning中遇到的问题。
2. 在模型结构方面进行了广泛实验，并对他们背后的机理进行了分析，最终得到了紧凑且有效的state-of-the-art 模型。

模型信息

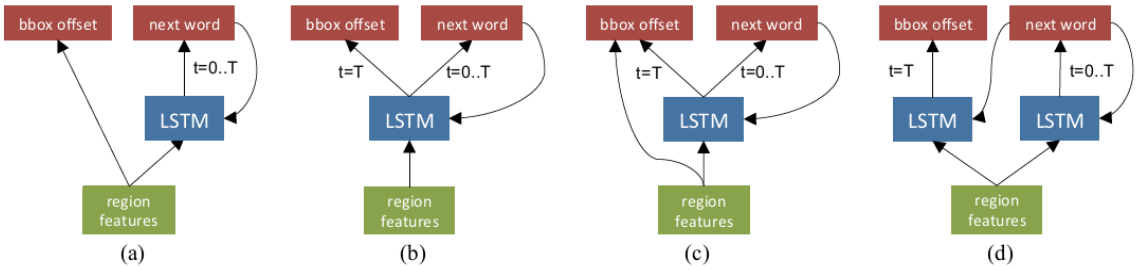


解决该问题的模型由**两部分**组成：Region Detection Network（生成RoI，提取RoI特征与环境特征）与Localization and Captioning Network（生成检测得分，描述短语和偏移量）

对于模型的第一部分，我们使用基于CNN的**RPN**（受到 **faster R-CNN** 的启发）。以下将对模型的第二部分进行详细描述。

Joint Inference：获取准确的localization

Localization包括生成RoI与bbox offset两部分，而在此部分**bbox offset**是我们的主要关注对象。本文共提出了4种Joint Inference方法来生成bbox offset：



(a) : Baseline model —— bbox offset仅由RoI特征生成

(b) : Shared-LSTM (S-LSTM) —— 用已有LSTM层的最后一步输出生成offset

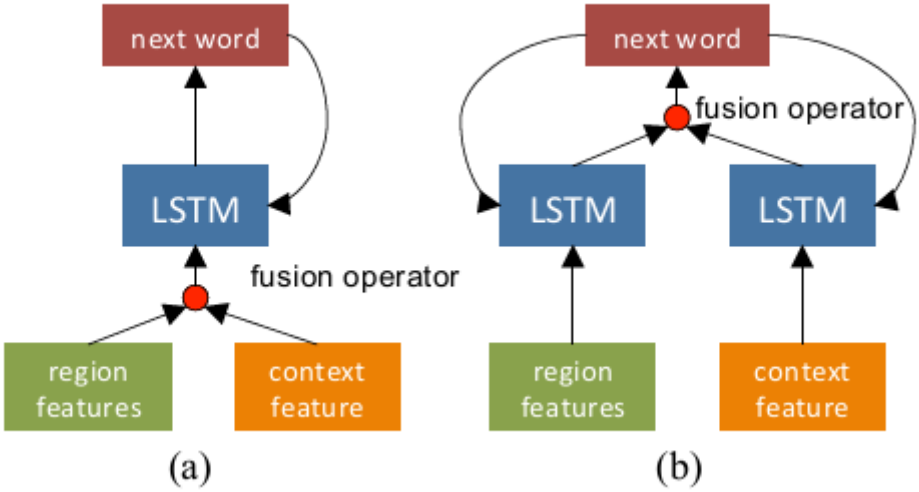
(c) : Shared-Concatenation-LSTM (SC-LSTM) —— 合并LSTM的输出与RoI特征生成offset（类似于ResNet)

(d) : Twin-LSTM (T-LSTM) —— 用两个LSTM网络分别生成caption与location。当caption完成时，location-LSTM此时收到完整的图像描述信息，并用整个caption作为输入生成offset。

Context Fusion：获取准确的description

当前利用环境信息完成Sequential Prediction任务（如Image Captioning）的尝试较为有限。本文着重于**更好地结合**整体环境特征与局部RoI特征完成Dense Captioning任务，而不是寻找环境特征的更好表示方法。因此，此处采用**将图片内所有的ROI的特征整合**后作为环境特征的简单方法。

本文共提出了两种Context Fusion方法来生成caption文本：

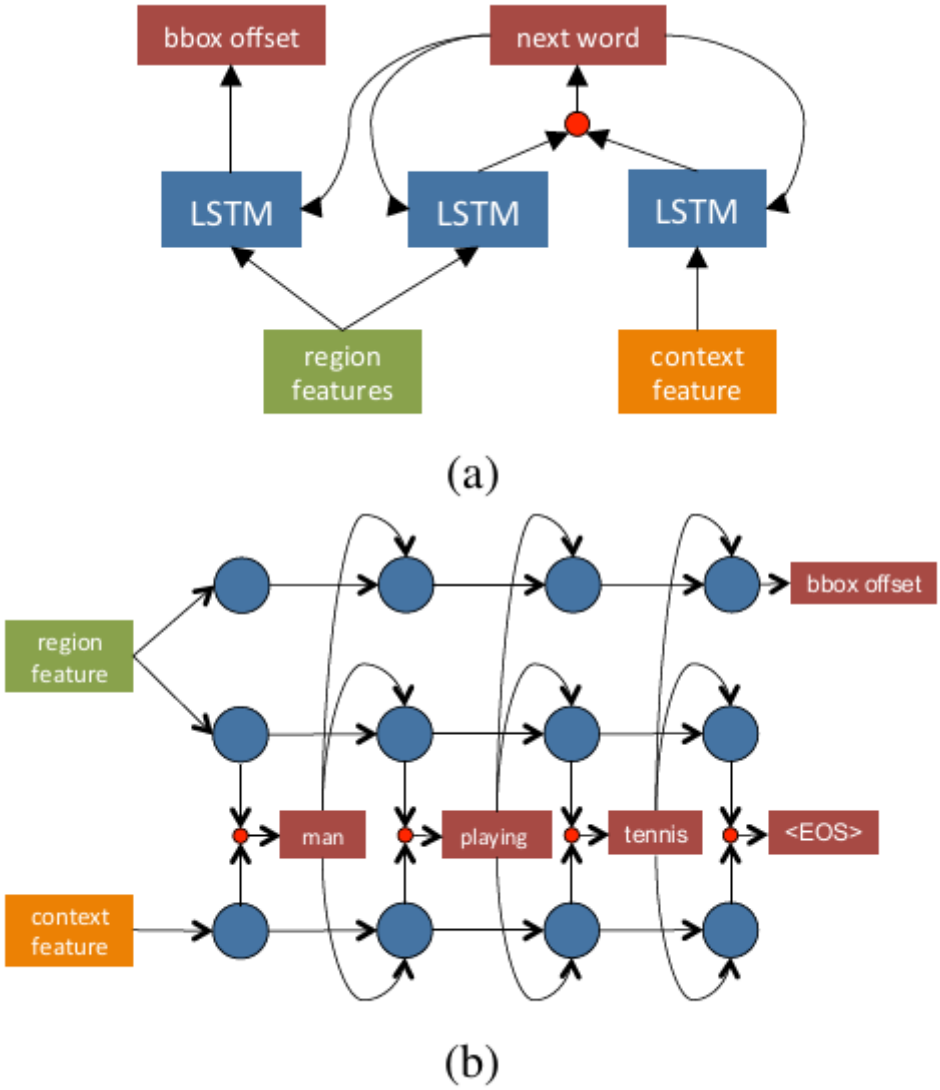


(a) Early-fusion —— 将局部RoI特征与整体环境特征通过fusion-operator**处理后送入**LSTM，生成描述文本。

(b) Late-fusion —— 将两种特征分别送入**独立的LSTM**进行处理，将**处理的结果**通过fusion-operator生成描述文本。

其中，**fusion-operator**共有三种：合并，相乘，求和。

模型整合



上文所提到的组件可以任意选择进行组合，产生不同的模型结构。以上为使用T-LSTM与Late-fusion的整合模型示意图。其中图 (b) 为图 (a) 中LSTM单元展开的形式。

> 注意：在LSTM单元的每一步生成Caption中的一个单词，在LSTM单元的最后一步生成offset。

模型训练与评估

数据集与评价指标

数据集：Visual Genome V1.0 & V1.2 （77K train, 5K val, 5K test）

评价指标：

1. Localization部分： IoU（Intersection-over-Union），阈值分别为 0.3, 0.4, 0.5, 0.6, 0.7。
2. Description部分： Meteor，阈值分别为 0, 0.05, 0.1, 0.15, 0.2, 0.25。
3. 总体评价指标：mAP（mean Average Precision），与本文相同。

训练过程

类别	描述
优化方法	随机梯度下降（SGD）
mini-batch size	采用的值为1
输入图片维度	图像调整为 600 x 720 像素
学习率	初始0.001，每100K次迭代后减半
momentum	0.98
权重衰减	未使用
fine-tuning	200K次迭代（~3 epochs）后开始调整CNN，600K次迭代（~9 epochs）后停止训练。CNN的前七层不参加调整。

其他细节：

1. 融合了语境信息的模型直接训练难以收敛，因此先训练不融合语境信息的模型（600K次迭代），再在此基础上进行fine-tuning。
2. 词库大小为10K，最大描述长度为10词。
3. 为提高效率，在描述的生成阶段使用了beam-size为1的集束搜索（即贪婪算法）。

训练结果

1. Joint Inference方面，T-LSTM表现最佳（mAP超出其他方法约30%）。
   原因：S-LSTM同时生成caption与offset，不属于同一类别的信息，效果不佳。
2. Context fusion方面，Late-fusion优于Early-fusion。
   原因：Early-fusion将局部信息与全局信息直接整合，而这两种信息存在显著的视觉差异，在后续的阶段难以分离。
3. 整体模型方面，环境信息对结果产生了较大提升（mAP 1.10）。其中，fusion-operator采用合并或相乘取得了相似的更好结果（较求和而言）。
4. 最好的模型：T-LSTM + Late-fusion + multiplication（mAP == 8.60）。

论文信息

类别	内容
论文名称	Actor-Critic Sequence Training for Image Captioning
发表时间	CVPR 2017
作者	Li Zhang, Flood Sung, Feng Liu, Tao Xiang, Shaogang Gong, Yongxin Yang, Timothy M. Hospedales
关键词	Image Captioning, Reinforcenemt Learning

核心信息

文章领域及方向

CV & NLP & RL，Image Captioning 问题

文章概要

本文通过构造Actor-Critic结构，利用强化学习解决了Encoder-Decoder结构在Image Captioning任务上存在的两个问题。本文采用Actor网络选择生成的词语序列，用Critic网络对Actor的决策加以评价，通过训练对Actor-Critic进行优化，最终在CIDERr等指标上达到了较高水平。

引言

本文重点

通过强化学习（Reinforcement Learning, RL）方法训练出在Captioning任务中更好的**语言生成模型**（Decoder），而不是**对图片的理解模型**（Encoder）。

当前Encoder-Decoder模型存在的问题

1. 因为模型在训练阶段**不直接接触自己生成的文本描述**，故在测试阶段由于样本不完全匹配，会导致**误差的累积。**
2. 模型训练常常采用**交叉熵**（**可微**）而不是CIDEr等**实际反映生成语言质量的指标**（**不可微**）作为损失函数。在训练阶段最优化交叉熵常常**并不能**最优化CIDEr等指标，这造成了训练的**偏差**与**低效。**

针对上述两个问题提出的解决方案

1. 通过**在训练过程中对模型的reward进行采样并优化**，解决训练阶段与测试阶段的不匹配问题。
2. 通过**将CIDEr等相关指标直接作为reward**并进行优化的方法，解决损失函数不高效的问题。

模型设计思路

本文通过采用**Actor-Critic**模型来解决Image Captioning问题。模型由如下两个部分组成：

1. **策略网络**（actor）：将Captioning视为给定图像的**序列决策**问题，根据图像**预测对应的Caption**（决策行为对应于单词的选择过程）。
2. **价值网络**（critic）：对于每一个**状态**（包含图像与当前已经生成的部分序列），根据语言指标（如CIDEr）**预测当前状态的value**（对于尚未生成的序列部分，基于当前的概率分布对其继续采样至序列完整）。

价值网络（critic）预测得到的value将被用来训练策略网络（actor）。假设actor预测的value值是准确的，那么actor使用value进行训练时将不会产生上述问题中提到的偏差。

与大多数强化学习任务相比，Image Captioning任务中**动作的数量更多**（如词典内词语的数量超过10K），但**训练的周期更短**（每一个序列生成完毕即结束）。

相关工作

1. **Image Captioning**：当前主要采用Encoder-Decoder模型，其中Encoder主要采用CNN，Decoder主要采用RNN。模型的细节在上文中已经提到，此处不再赘述。
2. **Image Captioning with RL**：与先前的一些尝试不同的是，本文将state作为RNN的输入而不是输出，因此本文可以构建一个独立的价值网络，而不是使用同一个RNN来构建actor与critic。
3. **Actor-Critic**：该模型通过critic生成决策的reward，并借此得到梯度来训练actor网络。AlpahGO即是该方法的典型应用之一。
4. **Sequence Generation**：当前有使用actor-critic结构解决机器翻译任务的尝试（同样是序列生成问题）。值得注意的是，在训练过程中常常需要引入多种技巧来**减小crtic网络结果的方差**，否则一些**罕见动作的reward往往会被大幅高估**，导致模型收敛的困难。

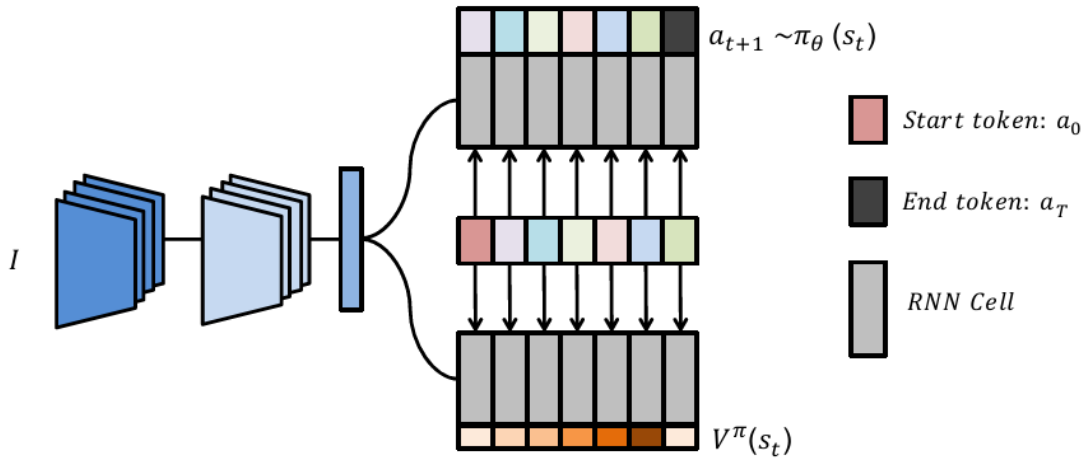
模型构造

问题公式化

我们将任务视为：对于给定的图像 I，生成Caption序列 $Y = \{y_1, \ldots, y_T\}, y_t \in D$，其中 D 为词典。为了简化公式，我们认为Caption序列的长度始终为定值 T.

对于训练与测试过程中的每一步，我们有输入-输出对 (I, Y) . 训练得到的序列生成模型对于给定的图像 I 将预测Caption序列 $\hat{Y}$，并计算得分 $R(\hat{Y}, Y)$ 来对模型进行优化与评价。此处的 R 函数依据任务不同，可以有BLEU或CIDEr等多种选择。

在本任务中，我们采用经典的Encoder-Decoder结构（见下图）



为了将Image Captioning任务转变成RL任务，我们考虑将Caption的生成过程视为**有限Markov决策过程**（MDP）$\{S, A, P, R, \gamma\}$. 在MDP中，状态 S 由图像 I 经过CNN的编码结果 I_e 与当前生成的单词序列（决策序列）$\{a_0, a_1, \ldots, a_t\}$ 两部分组成。因此，状态转移函数 P 为：$s_{t+1} = \{s_t, a_{t+1}\}$，即 a_{t+1} 成为新状态 s_{t+1}的一部分，并且同时获得reward r_{t+1}. 此外，$\gamma \in [0, 1]$ 为discount factor. 因

此，我们可以将该任务视为优化reward的标准RL任务。

模型组成部分

本模型采用Inception-V3作为CNN部分，LSTM作为RNN部分。策略网络（actor）与价值网络（critic）**都基于LSTM进行序列动作的决策和value的生成。**

策略网络（actor）

基于对任务的描述，我们将 I_e 和序列开始标记<start> a_0 进行组合得到起始状态 $s_0 = \{I_e, a_0\}$. 因此，我们可以递归地得到**每个状态的表示方法**：

$$s_t = \{I_e, a_0, a_1, \ldots, a_t\}$$

我们将状态 s_t 送入LSTM中获得隐藏层 h_{t+1}. 为了获得词语生成的概率模型，我们增加一个**随机输出层** f （常常用softmax作为激活函数），其结果为 $a_{t+1} \in D$.

$$h_{t+1} = LSTM(s_t),$$

$$a_{t+1} \sim f(h_{t+1})$$

因此，策略网络得出了在当前状态 s_t 下，动作 a_{t+1} 的概率分布 $p(a_{t+1}|s_t)$.

价值网络（critic）

在给定的策略 π，采样获得的动作与奖励函数下，value表示**期望的未来收益**（为 s_t 的函数）。我们约定 V 作为估计的state-value函数。

$$V^{\pi}(s_t) = \mathbb{E}[\sum_{l=0}^{T-t-1} \gamma^l r_{t+l+1} \mid a_{t+1}, \ldots, a_T \sim \pi, I]$$

其中discount factor γ 使得我们能够**以增加偏差为代价来减小方差。**它与MDP中出现的 γ 是一致的。

本模型中的价值网络可以被视为一个**Encoder**。我们使用一个独立的LSTM层（参数记为 ϕ ），接受状态 s_t 为输入，用其单一的输出值来预测 **TD 目标值。**

Advantage 函数

我们使用时序差分（TD）学习来获得Advantage. 我们定义Q值如下：

$$Q^{\pi}(s_t, a_{t+1}) = (1-\lambda)\sum_{n=1}^{\infty} G_t^n$$

其中 G_t^n 为n步的期望回报：

$$G_t^n = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V^{\pi}(s_t)$$

因此，我们得到Adavantage函数的表示形式：

$$A^{\pi}(s_t, a_{t+1}) = Q^{\pi}(s_t, a_{t+1}) - V^{\pi}(s_t) = (1-\lambda)\sum_{n=1}^{\infty} G_t^n - V^{\pi}(s_t)$$

Value 函数

我们使用一个非线性函数来表示Value，最简单的方法就是解决一个非线性回归问题：

$$argmin\ \phi\ ||Q^{\pi}(s_t, a_{t+1}) - V_{\phi}^t(s_t)||^2$$

其中 $Q^{\pi}(s_t, a_{t+1})$ 的定义已经在上文说明

λ 的选择

λ 在整个算法中起到重要作用。如果 $\lambda = 0$, 那么Advantage与Value的估计**只考虑单步TD**；如果 $\lambda = 1$, 那么估计则转变成**蒙特卡洛方法。**因为Image Captioning任务中的序列长度较短，我们需要对整个序列进行采样来获得reward，所以我们令 $\lambda = 1$, 这使得Advantage与Value的估计不会产生偏差，而有限的序列长度也限制了估计结果方差的大小。此时我们有：

$$Q^{\pi}(s_t, a_{t+1}) = \sum_{l=0}^{T-t-1} \gamma^l r_{t+l+1}$$

Reward

在本任务中，我们仅能在Caption序列完全生成时才能得到评价指标（如：CIDEr分数）。因此，我们定义reward如下：

$$r_t = \begin{cases} 0 & \text{if } t < T \\ score & \text{if } t = T \end{cases}$$

因此有：

$$Q^{\pi}(s_t, a_{t+1}) = \gamma^{T-t-1} r_T$$

模型训练与评估

训练过程

1. CNN部分采用Inception-v3，RNN部分采用深度为512的LSTM层（与embedding维度相同）
2. 词典大小为12K（注意上述参数均与NICv2模型相同，便于对比）
3. actor-critic模型难以从头开始训练。
   原因：两者都**无法给对方提供有效的指示信号。**actor将会完全随机选择单词，并在critic网络中得到非常低的reward。
4. 训练步骤：
   1. （**预训练actor**）以**交叉熵**为标准，训练一个新的actor网络。
   2. （**预训练critic**）对经过预训练的actor网络（参数固定）的决策进行采样，用采样的结果训练critic网络。（2K次迭代，使用Adam优化器，学习率5e-5）
   3. （**联合训练**）使用Adam优化器，初始学习率5e-5，1M次迭代后降为5e-6，mini-batch大小为16，discount factor λ = 1（蒙特卡洛方法）
5. 词语选择方面，为了提高效率，采用简单的**贪婪算法**选择生成的词语。

训练结果

1. 本模型的表现相较于同时期的其他模型有较大提升，详情见下图。

Metric	CIDEr-D	BLEU-4	METEOR	ROUGE-L
NIC [22]	0.855	0.277	0.237	-
NICv2 [23]	0.998	0.321	0.257	-
Semantic [10]	1.007	0.302	0.256	0.539
Semantic [10]+Attention [26]	1.042	0.311	0.263	0.543
Semantic [10]+Attention [26]+Memory [7]	1.057	0.318	0.266	0.547
Semantic [10]+Self-critical [17]	1.140	0.323	0.266	0.554
Ours	**1.162**	**0.344**	**0.267**	**0.558**

2. 在计算成本方面，本模型是效果相似的模型中**效率最高**的一个。这主要是因为本模型**不包含attention单元。**另外，self-critical模型需要在每次迭代中**采样两次**（随机采样 + 贪婪算法解

码），提高了算法的复杂度。详情见下图。

Model	Time
Semantic [10]+Attention [26]	0.10
Semantic [10]+Self-critical [17]	0.13
Semantic [10]+Self-critical [17]+Attention [26]	0.18
Ours	**0.07**

3. 以下为人工标注，优化交叉熵（XE）模型与本模型的结果示意图。可以看出本模型在结果的准确度上有较大的优越性。

Human : A man doing a trick on this skateboard.
XE : a man riding a skateboard on a cement ledge.
Ours : a man doing a trick on a skateboard.

Human : A motorcycle carrying many wheels is parked.
XE : a motorcycle parked next to a yellow wall.
Ours : a yellow motorcycle parked in front of a street.

Human : Large black motorcycle sitting next to a white building.
XE : a motorcycle parked next to a building.
Ours : a black motorcycle parked next to a building.

Human : A small personal pizza sits in a pizza box.
XE : a pizza is sitting on a box with a box of pizza.
Ours : a person holding a box of pizza.

Human : A group of skiers in the mountains reach a sign.
XE : a group of people standing on top of slope.
Ours : a group of people skiing on a snow covered slope.

Human : Old and new trains navigating a rail yard.
XE : a train is on the tracks in a city.
Ours : a black and white photo of trains on a train yard.

Show and Tell : Image Caption based on NIC

> 时间：2019-07-22
> 本项目是对 *Show and Tell: A Neural Image Caption Generator* 论文的复现
> 以下代码参考了 此代码库 与 此代码库

文件结构

名称	描述
temp\	自己实现的部分功能脚本
log\	训练日志与模型的保存文件夹
utils\	部分辅助脚本的保存文件夹
pic\	文档所需的部分图片的保存文件夹
process.py	COCO数据集的预处理脚本
data_loader.py	COCO数据集的Dataset定义脚本
model.py	模型的定义脚本
train_nic.py	模型的训练脚本

数据集

本项目采用的数据集为 Common Objects in Context, 具体信息如下：

- Training: 2014 Contest Train images [83K images/13GB]
- Validation: 2014 Contest Val images [41K images/6GB]
- Test: 2014 Contest Test images [41K images/6GB]

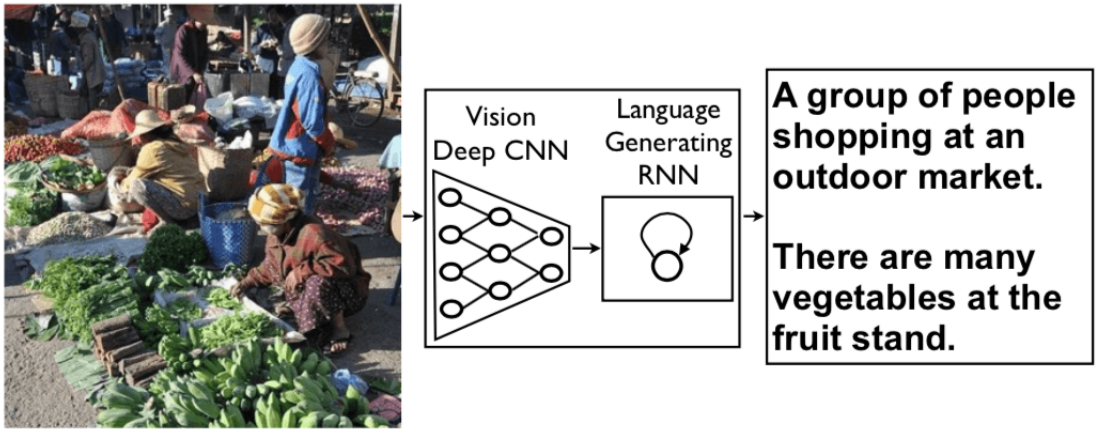
运行环境

> 使用的运行环境如下：
>
> torch == 1.0.1
> torchvision == 0.2.2
> pycocotools == 2.0.0
> pycocoevalcap == 0.0
>
> 训练机配置如下：
>
> GPU : GTX 1080
> CUDA == 8.0.44

模型结构

NIC结构由两个模型构成：Encoder与Decoder.

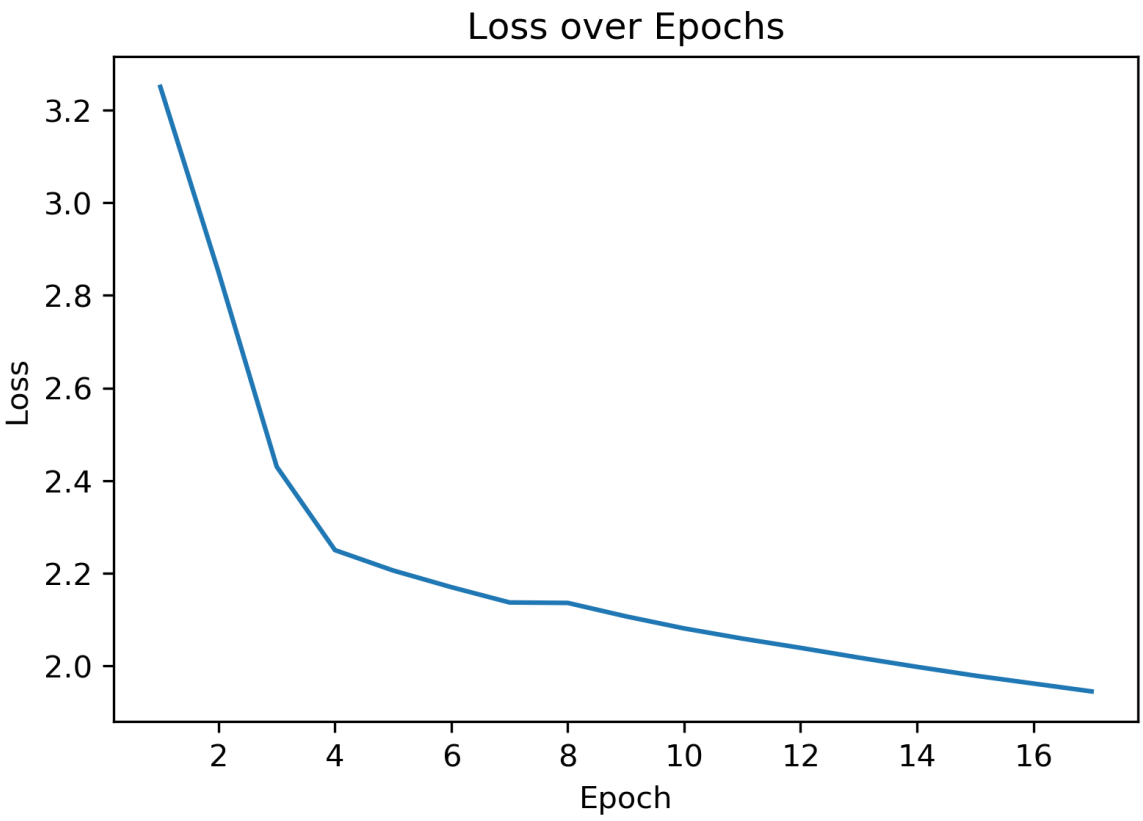
- Encoder：卷积神经网络，采用经过在 `ImageNet` 上预训练的 `ResNet152` 模型（`torchvision.models.resnet152`）并进行微调（*fine tuning*），目的是提取图片特征，创建对图片特征进行语义描述的定长向量（Feature Vector）
- Decoder：循环神经网络，采用LSTM/GRU作为结构单元，目的是在 Feature Vector 的基础上生成对图片的自然语言描述文本（Caption）



> 以上图片来源于此处

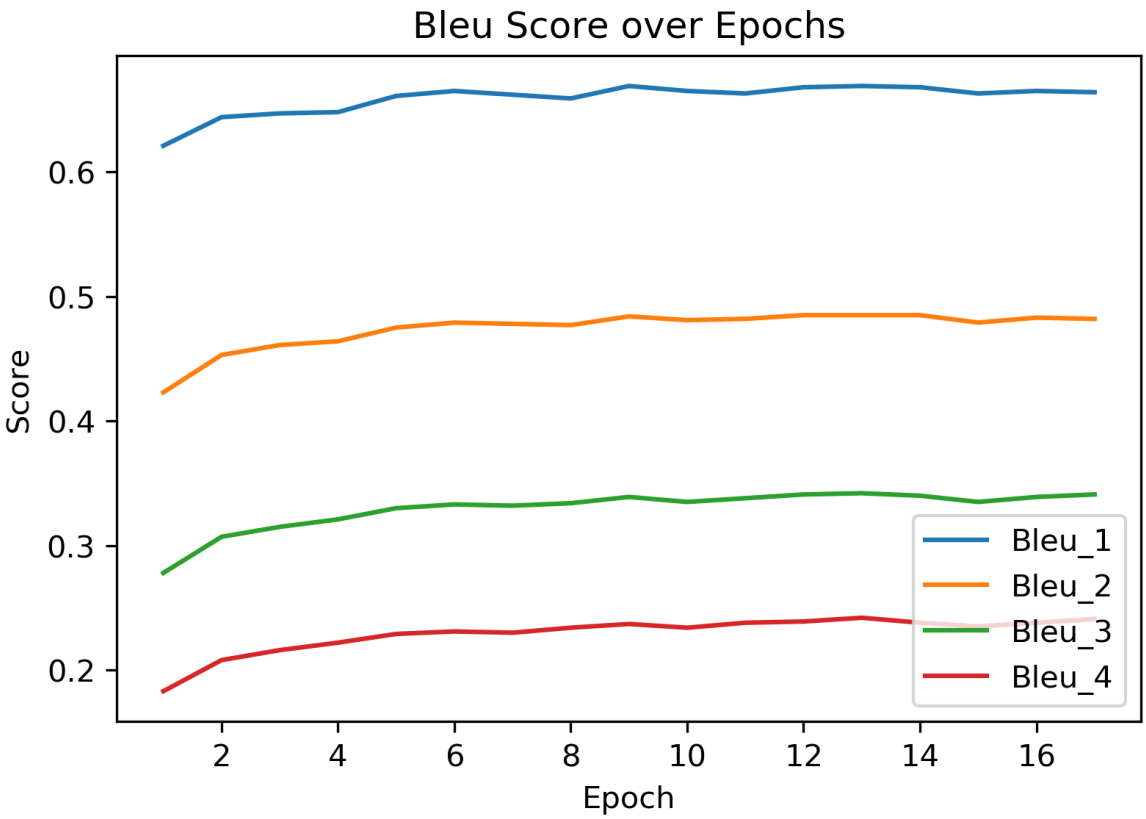
模型训练与结果

Loss



在训练过程中，模型的Loss基本呈稳定下降趋势。

BIEU指标



在训练过程中，模型的各级BLEU分数均呈现上涨趋势，并在10个Epoch左右出现收敛的倾向。

多指标评价

以下给出训练过程中奇数次Epoch的训练评估结果作为参考

Epoch	Bleu_1	Bleu_4	METEOR	ROUGE_L	CIDEr	SPICE
1	0.621	0.183	0.190	0.449	0.539	0.116
3	0.647	0.216	0.209	0.472	0.668	0.138
5	0.661	0.229	0.214	0.481	0.713	0.146
7	0.662	0.230	0.218	0.483	0.725	0.146
9	0.669	0.237	0.222	0.489	0.747	0.151
11	0.663	0.238	0.224	0.489	0.757	0.153
13	0.669	0.242	0.225	0.491	0.762	0.153
15	0.663	0.235	0.224	0.489	0.754	0.153
17	0.664	0.241	0.226	0.491	0.765	0.155

存在问题：

原论文中，在词语的生成选择阶段采用了Beam-size为20的Beam search。这提升了模型的BLEU指标约0.03。然而，对于自己实现的Beam search（在 `model.py\test_generate` 中实现），其计算速度远低于简单的贪婪算法。

根据分析可知，对每一组输入数据，贪婪算法仅需通过LSTM层，计算最多**20次**词语的概率分布（O(N)）。但对于Beam-size为20的Beam search，序列长度每增加1，都需要对当前所存储的所有状态计算一次概率分布（O(N^2)）。此外，还有大量的保存/恢复state，对概率序列进行排序（通过heapq实现）等操作。根据检验，使用个人实现的Beam search使评估的时间增加了约10倍。因此没有对其表现进行实际的评价。