自然语言处理大作业实验报告

张博源 刘冬阳 梁浩 李汉炫 李鑫

December 2021

1 小组成员信息及分工

- 张博源 (组长)：模型调研 Baseline 代码 文档撰写
- 刘冬阳：网页端 Demo 文档撰写
- 梁浩：模型调研 代码改进 文档撰写
- 李汉炫：模型调研 文档撰写与整理
- 李鑫：文档撰写

---

2 任务定义

Image Caption 任务是一个需要综合计算机视觉和自然语言处理的任务，需要使用计算机建立某种映射方式，将处于视觉模态当中的数据映射到文本模态当中。

总的来说，这样的映射任务需要如下两个基本需求：

1) 语法的正确性，映射的过程当中需要遵循自然语言的语法，使得结果具有可读性.
2) 描述的丰富程度，生成的描述需要能够准确描述对应图片的细节，产生足够复杂的描述。

3 方法描述

本方法使用了 Encoder-Decoder 结构，Encoder 部分通过 Faster-RCNN 提取图片中的物体区域，并且将原图划分为 5x5 的区域，将这些区域额外添加到检测结果中，然后

通过 ROI Pooling 从 Faster-RCNN 的 FPN 输出特征中提取上述每个区域的特征，最后通过一个全连接层得到 Encoder 部分的输出，示意图见图1。设 I 为输入图像，FPN 为 Faster-RCNN 的 FPN 网络，R 为 Resnet-101 网络，B 为检测得到的有物体的区域，P 为将图像划分后得到的区域加上整张图片，共 26 个额外区域，Encoder 部分的输出为 $FC(ROIPooling(FPN(R(I)), B \cup P))$。

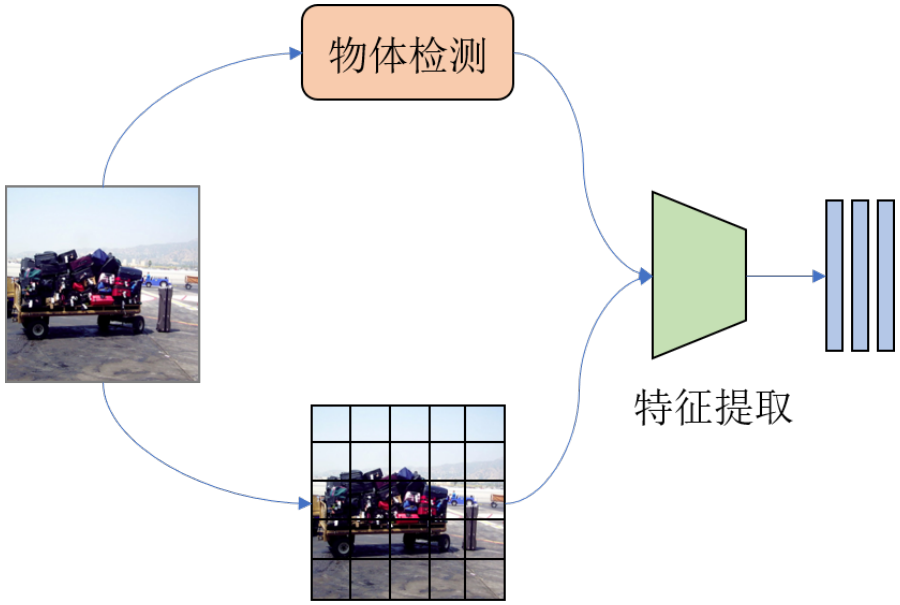

图 1: Encoder 示意图

Decoder 是一个带注意力的单层 LSTM 网络，使用上一步的状态和全部区域特征作为输入，通过全连接网络计算出每一个区域的权重，将区域特征加权求和，和上一步的输出一起作为下一步的输入，示意图见图2。记注意力中状态隐藏层权重为 $W_s \in \mathbb{R}^{h \times n}$ 、区域隐藏层权重为 $w_r \in \mathbb{R}^{h \times m}$、概率计算层权重为 $W_f \in \mathbb{R}^{1 \times h}$ ，记 LSTM 单元为 LSTM，记区域特征为 $F \in \mathbb{R}^{m \times k}$ 、上一步的 LSTM 隐状态为 $H \in \mathbb{R}^{n \times 1}$，则注意力计算结果为 $softmax(W_f \times relu(W_r \times F + repeat(W_s \times H, k)))$。后续将按照注意力对区域特征加权求和，并与上一步输出单词的词向量拼接起来，作为 LSTM 的输入。图 2 中的 gate 为 LSTM 中的输入门，故不再详细说明。

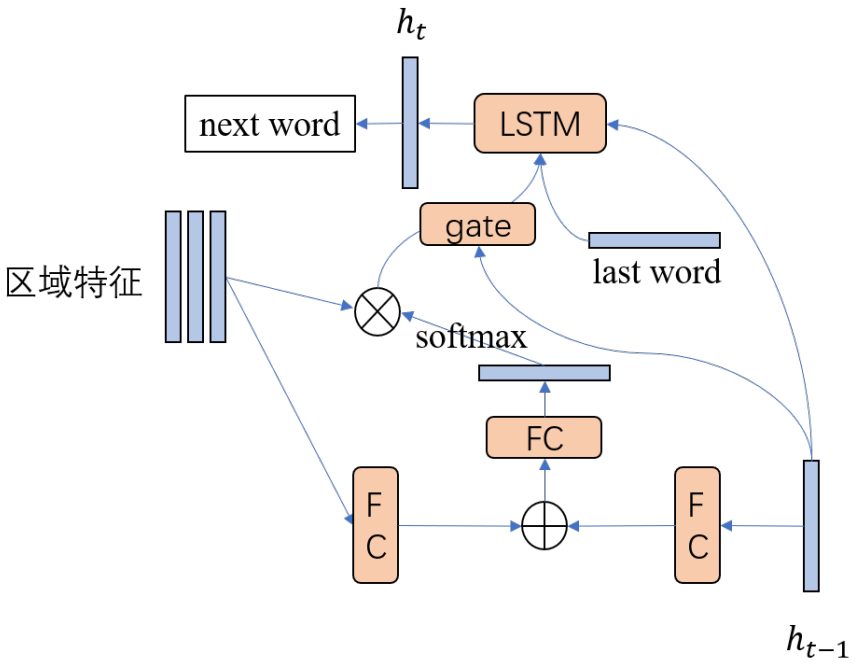

图 2: Decoder 示意图

4 系统框架

本部分将分别介绍本项目系统的网页端 Demo、使用指南以及系统结构：

- 网页端 Demo

  我们的网页端 Demo 部署在了腾讯云服务器上，其页面如图3所示，用户可以点击该网址直接访问。

- 使用指南

  我们的 Demo 包含两个页面，用户可以根据需求通过页面左侧的导航栏进行跳转。

  - 第一个页面（Live Demo）允许用户从本地选择图片并向后端服务器上传，后端接收到图片后会使用我们训练好的 PyTorch 模型对图片进行前向传播，最终得到生成的图像标题。对应对一张图片，基于 beam search 策略我们可以生成多条标题。在 Demo 中展示的标题数量为三条，其内容和置信度会展示在网页下方的表格中。在默认情况下（网页中 Local Server 开关为关闭状态时），图片会上传到部署在腾讯云上的后端服务器上。同时 Demo 也支持向本地的后端服务器发送请求。

– 第二个页面展示和对比了我们选择的 Baseline 以及我们改进后的模型在一些图片上的预测结果。

- 系统结构

  我们的 demo 系统为前后端分离开发。前端使用 Vue 编写，通过 nginx 部署在服务器上。后端使用 Flask 响应 http 请求，并维持了一个训练好的 PyTorch 的实例，该实例对用户上传的图片进行处理，得到图像对应的标题。

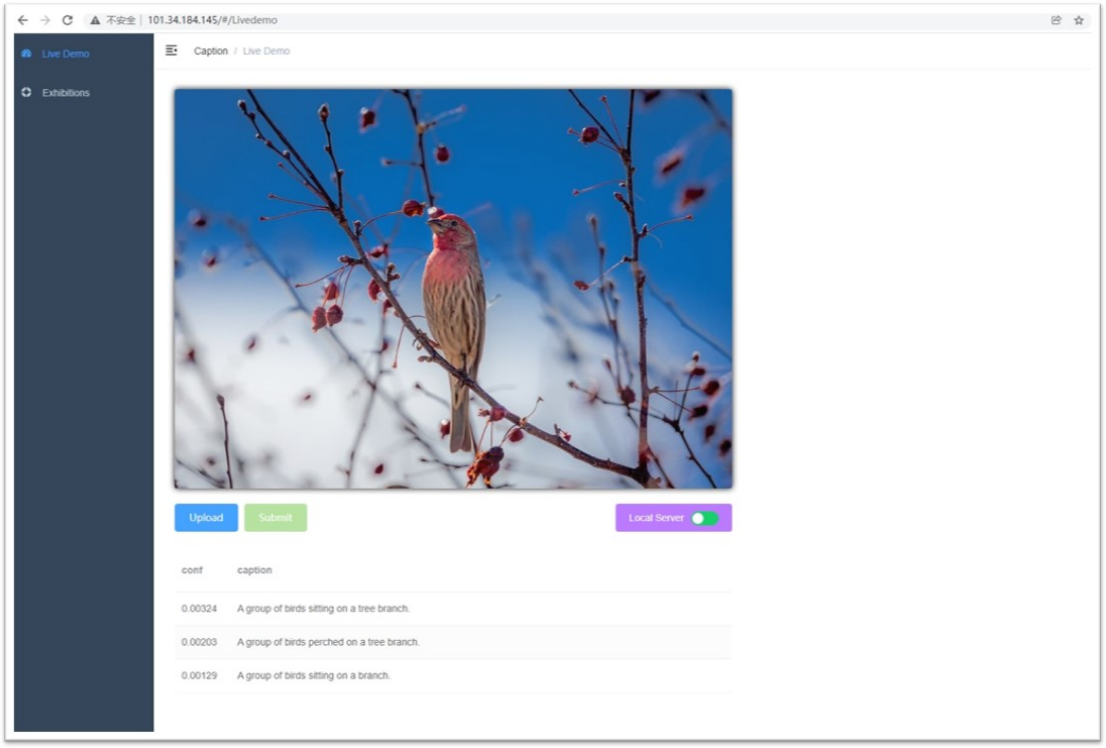

图 3: 网页端 Demo

5 实现细节

对于 Encoder 部分提取的区域，我们会将区域数量限制在 64，多余的舍弃，不足的补 0。对于 Faster-RCNN，我们使用了在 MSCOCO 数据集上预训练过的以 Resnet-101 为骨架的模型。Decoder 中的隐藏层维度均设置为 512，Encoder 中的全连接层输出同样为 512。优化器使用了 Adam，学习率设置为 0.001。

6 数据集

我们使用了 MSCOCO2014 数据集，其中训练集包含 113287 张图片，我们随机选取了 5000 张图片作为测试集。

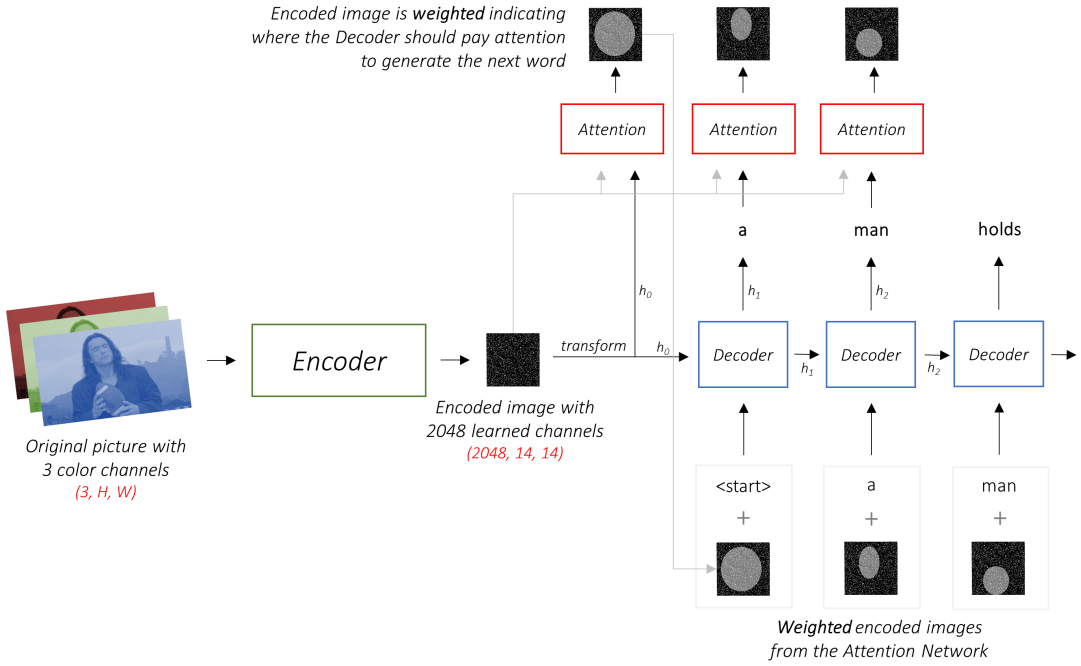

图 4: 基准模型结构

7 基准系统

本项目的基准模型采用了 show attend and tell 一文中所提出的结构。为了 Decoder 在序列的各个位置都能考虑到图片的不同部分，该论文在模型的解码阶段加入了 Attention 机制。完整的结构示意图如图4所示，下面将对基准模型的 Encoder、Deecoder 和 Attention 结构分别进行介绍：

- Encoder

  基准模型的 Encoder 部分直接采用了 PyTorch 的 torchvision 模块中已有的预训练模型——ResNet-101。具体地，由于 Encoder 只需要对图像进行编码，而不对其进行分类，我们丢弃 ResNet-101 的最后两层（池化层和线性层）。此外，我们添加了一个池化层来固定编码的长度以处理不同尺寸的图片输入。

- Decoder

  Decoder 的工作是查看编码后的图像并逐字生成标题。接收 Encoder 的输出后，将其平展为尺寸 $N, 14 \times 14, 2048$。我们使用编码后的全局图像特征初始化 LSTM 的隐藏状态和单元格状态。首先，通过减少标题长度对 N 幅图像和标题进行排序。这是为了只能处理有效的时间步长。

  遍历每个时间步，只处理有颜色的区域，它们是该时间步的有效批处理大小 N_t。排序允许顶部 N_t 在任何时间步上与前一步的输出对齐。利用注意网络计算每个时间步的权重和注意加权编码。将注意力加权编码通过一个滤波器或门。这个门是解码器先前隐藏状态的一个 s 形激活线性变换。

  将这种经过过滤的注意加权编码与 <start> token 连接起来，并运行 LSTMCell 来生成新的隐藏状态 (或输出)。一个线性层将这个新的隐藏状态转换为存储在词汇表中的每个单词的分数。同时还存储 Attention 网络在每个时间步返回的权重。

图 5: 结果展示

- Attention

  基准模型中的 Attention 结构比较简单，仅由线性层和几个激活组成。具体地，用两个线性层分别将来自 Encoder 的图像编码和来自 Decoder 的隐藏状态（即：输出）转换为相同的维度。之后将两组编码相加并使用 ReLU 函数激活。接着，第三个线性层再将激活结果转换为 1 维。最后使用 softmax 函数来生成权重 α。形式化地，对于具

有 P 个像素的被编码图像，在每一个时间步 t 上都有：

$$\sum_{p}^{P} \alpha_{p.t} = 1$$

8 实验设置与结果分析

量化实验部分，我们比较了两种方法在 MSCOCO2014 上的 BLEU-4，对比如表1所示。可以看出我们的方法相较于 Baseline 有一定的提升。对于更加复杂的模型，我们尝试了使用 LSTM 更换 Attention 部分，但结果并不理想（表1中 Complex），考虑到 Faster-RCNN 本身没有预测物体的属性，从它提取出的特征很可能无法支持更加复杂的网络结构，如果想要进一步提升性能很可能需要更换特征提取方式。

模型	Baseline	Faster-RCNN+LSTM	Complex
BLEU-4	0.225	0.239	0.223

表 1: 结果对比

主观实验部分，我们随机抽取了一些图片并进行了对比，见图5。得益于 Faster-RCNN 的物体检测能力，我们的模型可以更好地描述图片中的物体。