深度学习训练营

案例 6：图像自然语言描述生成（让计算机“看图说话”）

1 任务和数据简介

本次案例将使用深度学习技术来完成图像自然语言描述生成任务，输入一张图片，模型会给出关于图片内容的语言描述。本案例使用 coco2014 数据集[1]，包含 82,783 张训练图片，40,504 张验证图片，40,775 张测试图片。案例使用 Andrej Karpathy[2]提供的数据集划分方式和图片标注信息，案例已提供数据处理的脚本，只需下载数据集和划分方式即可。

图像自然语言描述生成任务一般采用 Encoder-Decoder 的网络结构，Encoder 采用 CNN 结构，对输入图片进行编码，Decoder 采用 RNN 结构，利用 Encoder 编码信息，逐个单词的解码文字描述输出。模型评估指标采用 BLEU 分数[3]，用来衡量预测和标签两句话的一致程度，具体计算方法可自行学习，案例已提供计算代码。

2 方法描述

模型输入

图像统一到 256×256 大小，并且归一化到$[-1,1]$后还要对图像进行 RGB 三通道均值和标准差的标准化。语言描述标签信息既要作为目标标签，也要作为 Decoder 的输入，以<start>开始，<end>结束并且需要拓展到统一长度，例如:

$$< start > a\ table\ topped\ with\ plates\ of\ food\ and\ drinks < end >< pad >< pad >< pad > \cdots$$

每个 token 按照词汇表转为相应的整数。同时还需要输入描述语言的长度，具体为单词数加 2 (<start>，<end>)，目的是为了节省在<pad>上的计算时间。

Encoder

案例使用 ResNet101 网络作为编码器，去除最后 Pooling 和 Fc 两层，并添加了 AdaptiveAvgPool2d()层来得到固定大小的编码结果。编码器已在 ImageNet 上预训练好，在本案例中可以选择对其进行微调以得到更好的结果。

Decoder

Decoder 是本案例中着重要求的内容。案例要求实现两种 Decoder 方式，分别对应这两篇文章[4][5]。在此简要阐述两种 Decoder 方法，进一步学习可参考原文章。

第一种 Decoder 是用 RNN 结构来进行解码，解码单元可选择 RNN、LSTM、GRU 中的一种，初始的隐藏状态和单元状态可以由编码结果经过一层全连接层并做批归一化 (Batch Normalization) 后作为解码单元输入得到，后续的每个解码单元的输入为单词经过 word embedding 后的编码结果、上一层的隐藏状态和单元状态，解码输出经过全连接层和 Softmax 后得到一个在所有词汇上的概率分布，并由此得到下一个单词。Decoder 解码使用到了 teacher forcing 机制，每一时间步解码时的输入单词为标签单词，而非上一步解码出来的预测单词。训练时，经过与输入相同步长的解码之后，计算预测和标签之间的交叉熵损失，进行 BP 反传更新参数即可。测试时由于不提供标签信息，解码单元每一时间步输入单词为上一步解码预测的单词，直到解码出<end>信息。测试时可以采用 beam search 解码方法来得到更准确的语言描述，具体方法可自行学习。

第二种 Decoder 是用 RNN 加上 Attention 机制来进行解码，Attention 机制做的是生成一组权重，对需要关注的部分给予较高的权重，对不需要关注的部分给予较低的权重。当生成某个特定的单词时，Attention 给出的权重较高的部分会在

图像中该单词对应的特定区域，即该单词主要是由这片区域对应的特征生成的。Attention 权重的计算方法为：

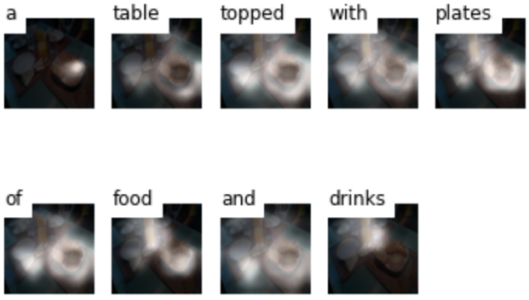
$$\alpha = softmax\left(fc\left(relu\left(fc(encoder\_output) + fc(h)\right)\right)\right)$$

其中 softmax()表示 Softmax 函数，fc()表示全连接层，relu()表示 ReLU 激活函数，encoder_output 是编码器的编码结果，h 是上一步的隐藏状态。初始的隐藏状态和单元状态由编码结果分别经过两个全连接层得到。每一时间步解码单元的输入除了上一步的隐藏状态和单元状态外，还有一个向量，该向量由单词经过 word embedding 后的结果和编码器编码结果乘上注意力权重再经过一层全连接层后的结果拼接而成。解码器同样使用 teacher forcing 机制，训练和测试时的流程与第一种 Decoder 描述的一致。

样例输出

第一种 Decoder 得到的结果仅包含图像的文字描述，如下图：

第二种 Decoder 由于有 Attention 机制的存在，可以得到每个单词对应的图片区域，如下图：



3 参考程序及使用说明

本次案例提供了完整、可供运行的参考程序，各程序简介如下：

create_input_files.py：下载好数据集和划分方式后需要运行该脚本文件，会生成案例需要的 json 和 hdf5 文件，注意指定输入和输出数据存放的位置。

datasets.py：定义符合 pytorch 标准的 Dataset 类，供数据按 Batch 读入。

models.py：定义 Encoder 和 Decoder 网络结构，其中 Encoder 已提前定义好，无需自己实现。两种 Decoder 方法需要自行实现，已提供部分代码，只需将 #To Do 部分补充完全即可。

solver.py：定义了训练和验证函数，供模型训练使用。

train.ipynb：用于训练的 jupyter 文件，其中超参数需要自行调节，训练过程中可以看到模型准确率和损失的变化，并可以得到每个 epoch 后模型在验证集上的 BLEU 分数，保存最优的验证结果对应的模型用于测试。

test.ipynb：用于测试的 jupyter 文件，加载指定的模型，解码时不使用 teacher forcing，并使用 beam search 的解码方法，最终会得到模型在测试集上的 BLEU 分数。

caption.ipynb：加载指定模型，对单张输入图片进行语言描述，第一种 Decoder 方法只能得到用于描述的语句，第二种 Decoder 方法同时可以获取每个单词对应的注意力权重，最后对结果进行可视化。

utils.py：定义一些可能需要用到的函数，如计算准确率、图像可视化等。

环境要求：python 包 pytorch, torchvision, numpy, nltk, tqdm, h5py, json, PIL, matplotlib, scikit-image, scipy=1.1.0 等。

4 要求与建议

- 完成 models.py 文件中的 #To Do 部分，可参考第 2 部分中的介绍或原论文；
- 调节超参数，运行 train.ipynb，其中 attention 参数指示使用哪种 Decoder，分别训练使用两种不同 Decoder 的模型，可以分两个 jupyter 文件保存最佳参数和训练记录，如 train1.ipynb, train2.ipynb；
- 运行 test.ipynb 得到两个模型在测试集上的 BLEU 分数，分别保留结果；
- 选择一张图片，可以是测试集中的，也可以是自行挑选的，对图片进行语言描述自动生成，分别保留可视化结果；
- 在参考程序的基础上，综合使用深度学习各项技术，尝试提升该模型在图像自然语言描述生成任务上的效果，如使用更好的预训练模型作为 Encoder，或者提出更好的 Decoder 结构，如 Adaptive Attention 等；
- 完成一个实验报告，内容包括基础两个模型的实现原理说明、两个模型的最佳参数和对应测试集 BLEU 分数、两个模型在单个图片上的表现效果、自己所做的改进、对比分析两个基础模型结果的不同优劣。
- 禁止任何形式的抄袭，借鉴开源程序务必加以说明。

5 参考资料

[1] MS-COCO 数据集: https://cocodataset.org/

[2] 划分方式与 caption 信息:

http://cs.stanford.edu/people/karpathy/deepimagesent/caption_datasets.zip

[3] https://en.wikipedia.org/wiki/BLEU

[4] Vinyals O, Toshev A, Bengio S, et al. Show and tell: A neural image caption generator[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3156-3164.

[5] Xu K, Ba J, Kiros R, et al. Show, attend and tell: Neural image caption generation with visual attention[C]//International conference on machine learning. 2015: 2048-2057.