在玩图像分类和图像分割？来挑战基于 TensorFlow 的图像注解生成！

本文作者：三川 2017-05-02 18:44

导语：深度学习在 CV 和 NLP 领域的交叉应用。

雷锋网(公众号：雷锋网)按：本文刊载于 Oreilly，雷锋网编译。阅读原文地址请点此。

玩过图像分类的开发者不少，许多人或许对图像分割（image segmentation）也不陌生，但图像注解（image caption）的难度，无疑比前两者更进一步。

原因无他：**利用神经网络来生成贴合实际的图像注释，需要结合最新的计算机视觉和机器翻译技术，缺一不可。**对于为输入图像生成文字注解，训练神经图像注解模型能使其成功几率最大化，并能生成新奇的图

像描述。举个例子，下图便是在 MS COCO 数据集上训练的神经图像注解生成器，所输出的潜在注解。

左图注解：一个灰衣男子挥舞棒子，黑衣男子旁观；右图注解：一辆大巴车“坐”在一个人旁边

本文是一篇中级教程，旨在教给大家如何在 Flickr30k 数据集上训练图像注解生成模型，使用的是谷歌 Show and Tell 模型的变体。我们使用 TensorFlow 框架来创建、训练、测试模型，因为 TensorFlow 相对容易使用，并且有不断增长的庞大用户社群。

图像注解技术的价值

近来深度学习在 CV（计算机视觉）和 NLP（自然语言处理）领域的成功，激发了 AI 研究人员在这两者的交叉领域探索新的应用。而作为结果的注解生成模型，需要平衡对视觉线索和自然语言的理解。

这两门传统上泾渭分明、并不相关的领域之间所产生的交集，有潜力在业内产生广泛的影响。该技术有一些直接应用场景，比如为 YouTube 视频生成简介，又比如为无标签图像做注解，但其价值远不止于此。类似于传统的 CV 技术试图让现实世界变得让计算机更容易接触更容易理解，该技术有潜力让现实世界变得对于我们来说更容易理解。它可以作为我们的导游、日常生活的视觉辅助，比如意大利 AI 公司 Eyra 的为视觉障碍患者开发的可穿戴设备 Horus（古埃及神话中的荷鲁斯之眼）。

准备工作

迁移学习怎么做？迁移成分分析 (TCA) 方法简介

热门搜索

腾讯 虚拟现实 机器学习 锤子科技 OPPO 淘宝 处理器 Galaxy Galaxy S7 监管 高德

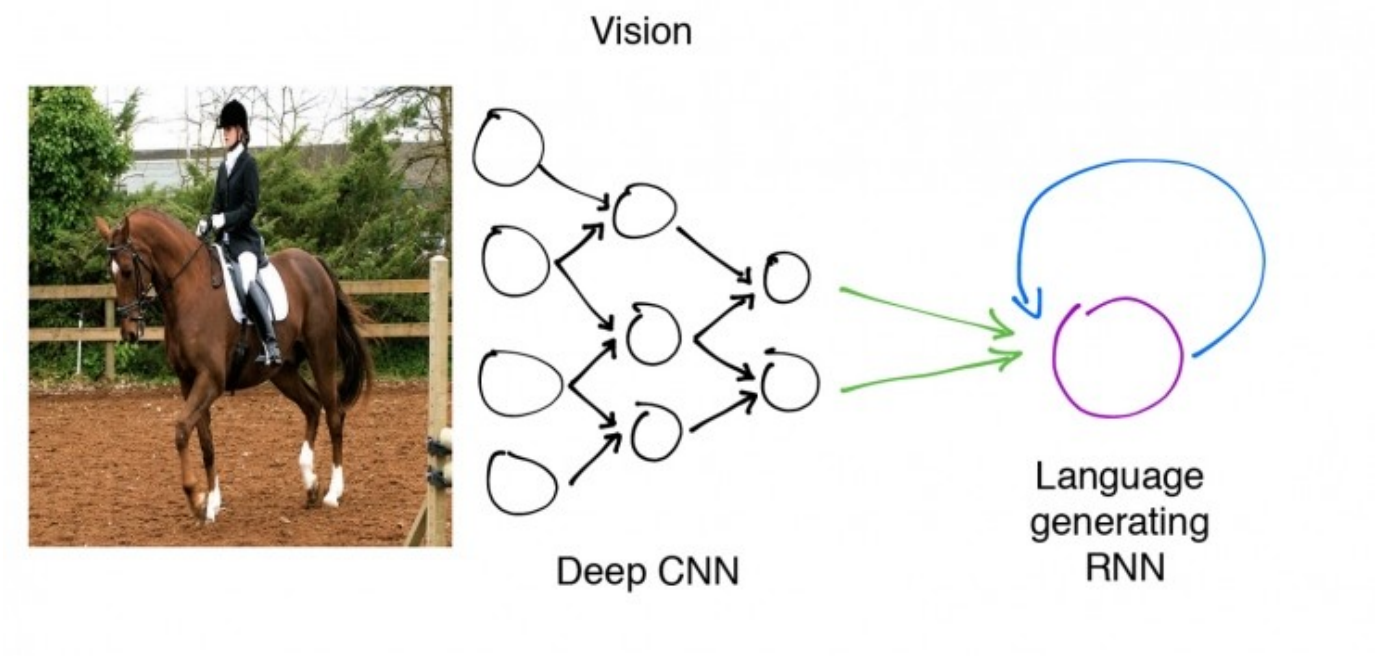
首先，你需要安装 TensorFlow。

其次，你需要 pandas, opencv2 以及 Jupyter 来跑相关代码。但是，为了简化安装过程，我们强烈推荐你在我们的 GitHub 资源库里跟随 Docker 的安装指南。

你还需要下载 Flickr30k 数据集的图像注解和 image embeddings。下载链接也在 GitHub 资源库里。

现在教程开始。

图像注解生成模型



在高层级，这就是我们将要训练的模型。每一幅图像将会用深度 CNN 编码成 4,096 维的矢量表示。一个语言生成 RNN 会随后对其按次序解码，成为自然语言描述。

注解生成——作为图像分类的延伸

作为一个历史悠久的 CV 任务，图像分类背后有许多强大模型。图像分类能把图像中相关联的形状、物体的视觉信息拼凑到一起，把图像放入物体类别中。针对其他 CV 任务的机器学习模型，建立在图像分类的基础之上，比如物体识别和图像分割。它们不仅能对提供的信息进行识别，还能学习如何解读 2D 空间，调和两种理解，并决定图像中物体信息的位置分布。

对于注释生成，这引发了两个问题：

1. 我们如何充分利用图像分类模型的成功，从图像提取重要信息？
2. 我们的模型，该如何调和对语言和图像的理解？

利用迁移学习

我们可以利用已有的模型，推动图像注解。迁移学习使得——在不同任务上训练神经网络而学习到的数据变形，能用于我们的数据。在我们的例子中，VGG-16 图像分类模型导入 224x224 分辨率的图像，生成对分类图像非常有用的 4,096 维特征矢量。

我们可以把 VGG-16 模型的表示（即 image embedding）用于训练其他模型上面。受篇幅限制，本文对 VGG-16 的架构不欲详述，并预先计算好了 4,096 维特征，来加速训练。

载入 VGG 图像特征和图像注解比较直接：

```
def get_data(annotation_path, feature_path):
    annotations = pd.read_table(annotation_path, sep='\t', header=None, names=['image', 'caption'])
    return np.load(feature_path,'r'), annotations['caption'].values
```

理解注解

有了图像表示，我们需要模型来学习把它解码为可理解的注解。由于文本的序列本质，我们需利用 RNN/LSTM 中的循环。对于序列中的给定词语，这些网络被训练，用以预测下一个词语以及图像表示。

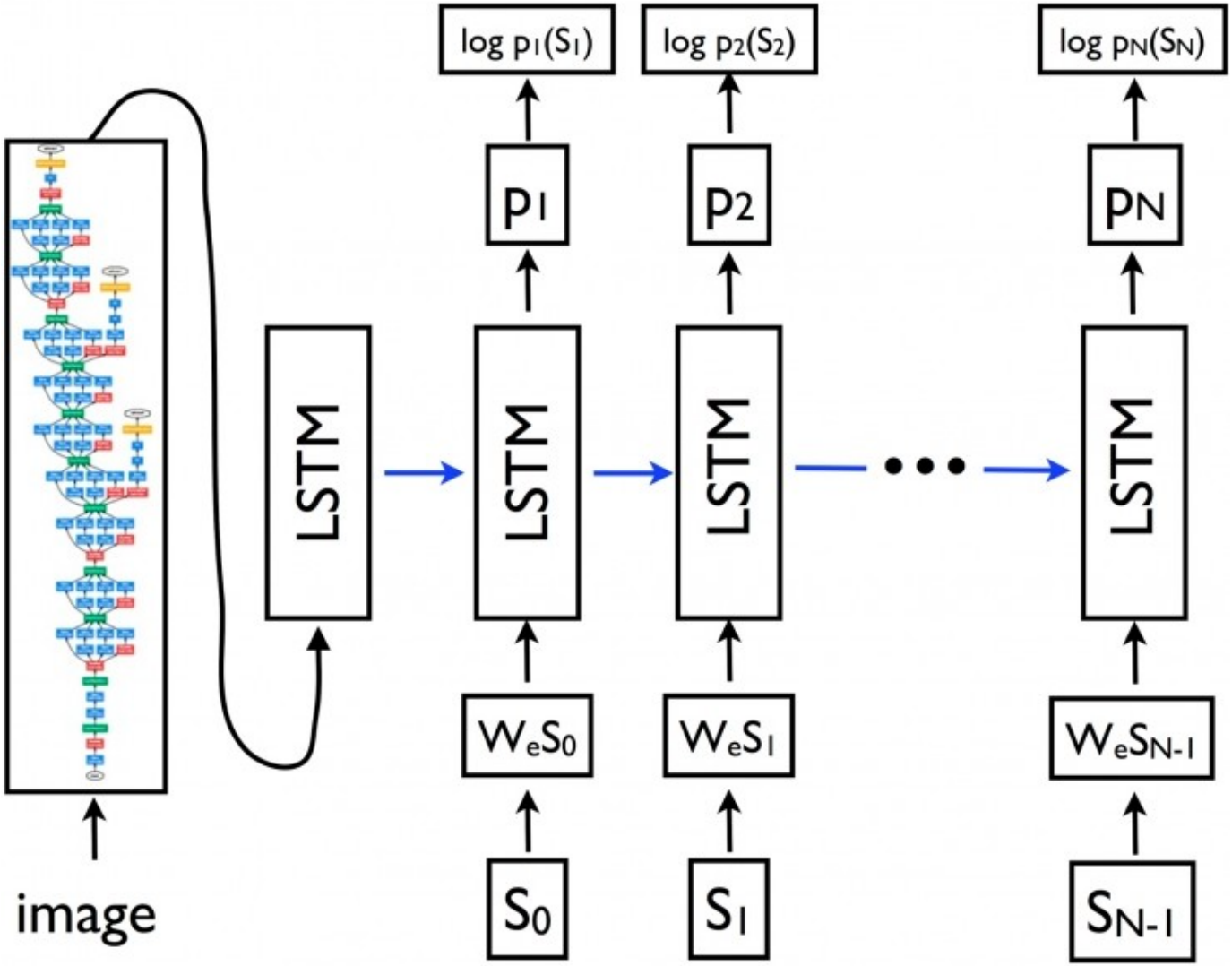
LSTM 单元允许模型在注解词语序列中，更好地选择使用哪条信息、记忆什么、又要忘记什么。TensorFlow 提供了一个 wrapper 函数，来对给定输入、输出维度生成 LSTM 层。

为了把词语转换成适合 LSTM 输入的固定长度表示，我们使用了一个 embedding 层，它能学习把词语映射到 256 维的特征（或 word-embedding）。Word-embedding 帮助我们把词语表示为矢量，相近的词语矢量在语义上也是近似的。

在 VGG-16 图像分类器里，卷积层提取了 4,096 维表示，传入最终的 softmax 层进行分类。由于 LSTM 单元需要 256 维文本特征作为输入，我们需要把图像表示转译为针对目标注解的表示。为实现这一点，我们使用了另一个 embedding 层，学习把 4,096 维图像特征映射到 256 维文本特征的空间。

创建和训练模型

整体上，这就是谷歌 Show and Tell 模型看起来的样子：



上图中，{s0, s1, ..., sN}代表了注解中我们想要预测的词语，{wes0, wes1, ..., wesN-1} 是每个词的 word embedding 矢量。LSTM 的输出 {p1, p2, ..., pN} 是模型生成的概率分布，针对句子中的下一个词。该模型被训练来最小化每个词语对数概率（log probabilities ）的负数总和。

```
def build_model(self):
```

```
        # declaring the placeholders for our extracted image feature vectors, our caption,
and our mask
        # (describes how long our caption is with an array of 0/1 values of length `maxlen`
        img = tf.placeholder(tf.float32, [self.batch_size, self.dim_in])
        caption_placeholder = tf.placeholder(tf.int32, [self.batch_size, self.n_lstm_steps])
        mask = tf.placeholder(tf.float32, [self.batch_size, self.n_lstm_steps])

        # getting an initial LSTM embedding from our image_imbedding
        image_embedding = tf.matmul(img, self.img_embedding) + self.img_embedding_bias

        # setting initial state of our LSTM
        state = self.lstm.zero_state(self.batch_size, dtype=tf.float32)

       total_ loss = 0.0
        with tf.variable_scope("RNN"):
            for i in range(self.n_lstm_steps):
                if i > 0:
                   #if this isn't the first iteration of our LSTM we need to get the
word_embedding corresponding
                   # to the (i-1)th word in our caption
                    with tf.device("/cpu:0"):
                        current_embedding = tf.nn.embedding_lookup(self.word_embedding,
caption_placeholder[:,i-1]) + self.embedding_bias
                else:
                     #if this is the first iteration of our LSTM we utilize the embedded
image as our input
                     current_embedding = image_embedding
                if i > 0:
                    # allows us to reuse the LSTM tensor variable on each iteration
                    tf.get_variable_scope().reuse_variables()

                out, state = self.lstm(current_embedding, state)

                print (out,self.word_encoding,self.word_encoding_bias)
```

```
            if i > 0:
                #get the one-hot representation of the next word in our caption
                labels = tf.expand_dims(caption_placeholder[:, i], 1)
                ix_range=tf.range(0, self.batch_size, 1)
                ixs = tf.expand_dims(ix_range, 1)
                concat = tf.concat([ixs, labels],1)
                onehot = tf.sparse_to_dense(
                        concat, tf.stack([self.batch_size, self.n_words]), 1.0, 0.0)

                #perform a softmax classification to generate the next word in the
caption
                logit = tf.matmul(out, self.word_encoding) + self.word_encoding_bias
                xentropy = tf.nn.softmax_cross_entropy_with_logits(logits=logit,
labels=onehot)
                xentropy = xentropy * mask[:,i]

                loss = tf.reduce_sum(xentropy)
                total_loss += loss

        total_loss = total_loss / tf.reduce_sum(mask[:,1:])
        return total_loss, img,  caption_placeholder, mask
```

使用推理生成注解

训练之后，我们有了一个模型。给定图像和所有此前的词语，它能给出下一步某个词出现在注解中的概率。如何用它来生成新注解呢？

最简单的办法，是拿来一个输入图像，输出下一个可能性最高的词语，创建一个简单的图像注解。

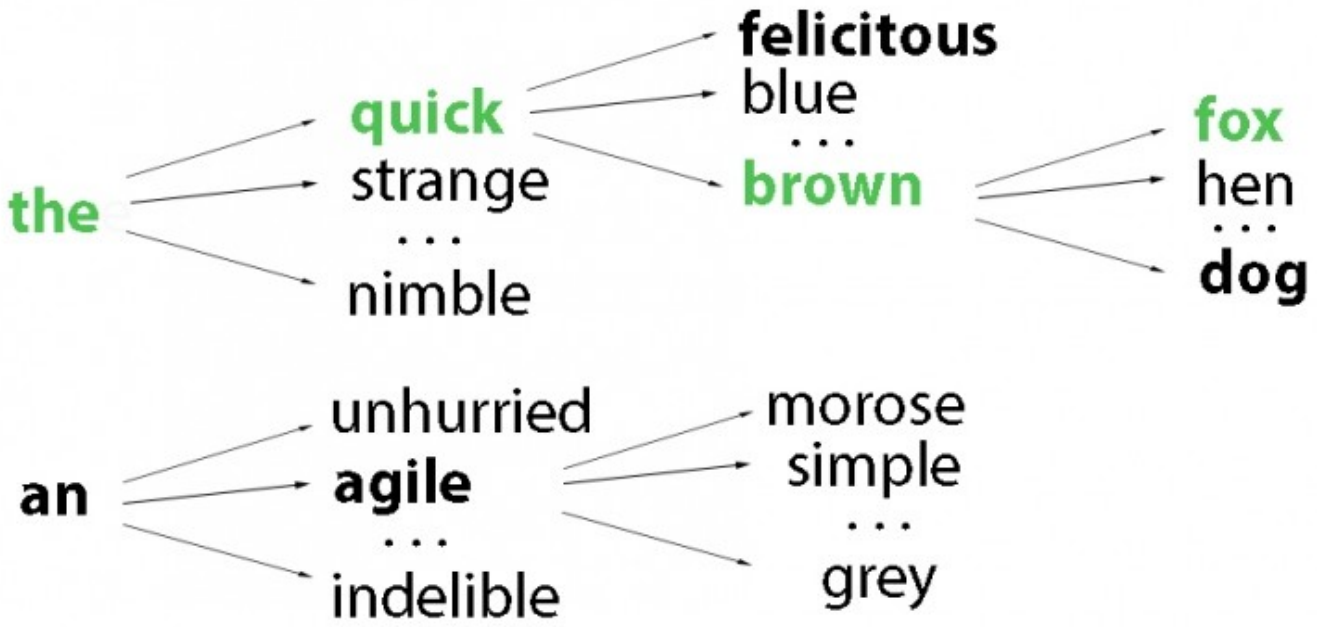
```
def build_generator(self, maxlen, batchsize=1):
```

```
        #same setup as `build_model` function
        img = tf.placeholder(tf.float32, [self.batch_size, self.dim_in])
        image_embedding = tf.matmul(img, self.img_embedding) + self.img_embedding_bias
        state = self.lstm.zero_state(batchsize,dtype=tf.float32)

       #declare list to hold the words of our generated captions
        all_words = []
        print (state,image_embedding,img)
        with tf.variable_scope("RNN"):
           # in the first iteration we have no previous word, so we directly pass in the
image embedding
           # and set the `previous_word` to the embedding of the start token ([0]) for the
future iterations
            output, state = self.lstm(image_embedding, state)
            previous_word = tf.nn.embedding_lookup(self.word_embedding, [0]) +
self.embedding_bias

            for i in range(maxlen):
                tf.get_variable_scope().reuse_variables()

                out, state = self.lstm(previous_word, state)


               # get a one-hot word encoding from the output of the LSTM
                logit = tf.matmul(out, self.word_encoding) + self.word_encoding_bias
                best_word = tf.argmax(logit, 1)

                with tf.device("/cpu:0"):
                   # get the embedding of the best_word to use as input to the next iteration
of our LSTM
                    previous_word = tf.nn.embedding_lookup(self.word_embedding, best_word)

                previous_word += self.embedding_bias

                all_words.append(best_word)
```

```
        return img, all_words
```

这在许多情况下会奏，但“贪婪”地使用每一个概率最大的词语，未必能获得整体上概率最大的注解。

绕过这个难题的一个潜在方案，是采用一个名为"Beam Search"的方法。该算法会对长度 t 以内的 k 个最佳语句集反复考量，作为候选来生成 t + 1 大小的句子，只保留结果中的 k 个最佳选择。这允许开发者探索一个较大的优质注解空间，同时让推理在计算上可追踪。在下面的例子里，算法保持了一个 k = 2 的候选句子列表，即每个垂直时间步到每个加粗词语的路线。



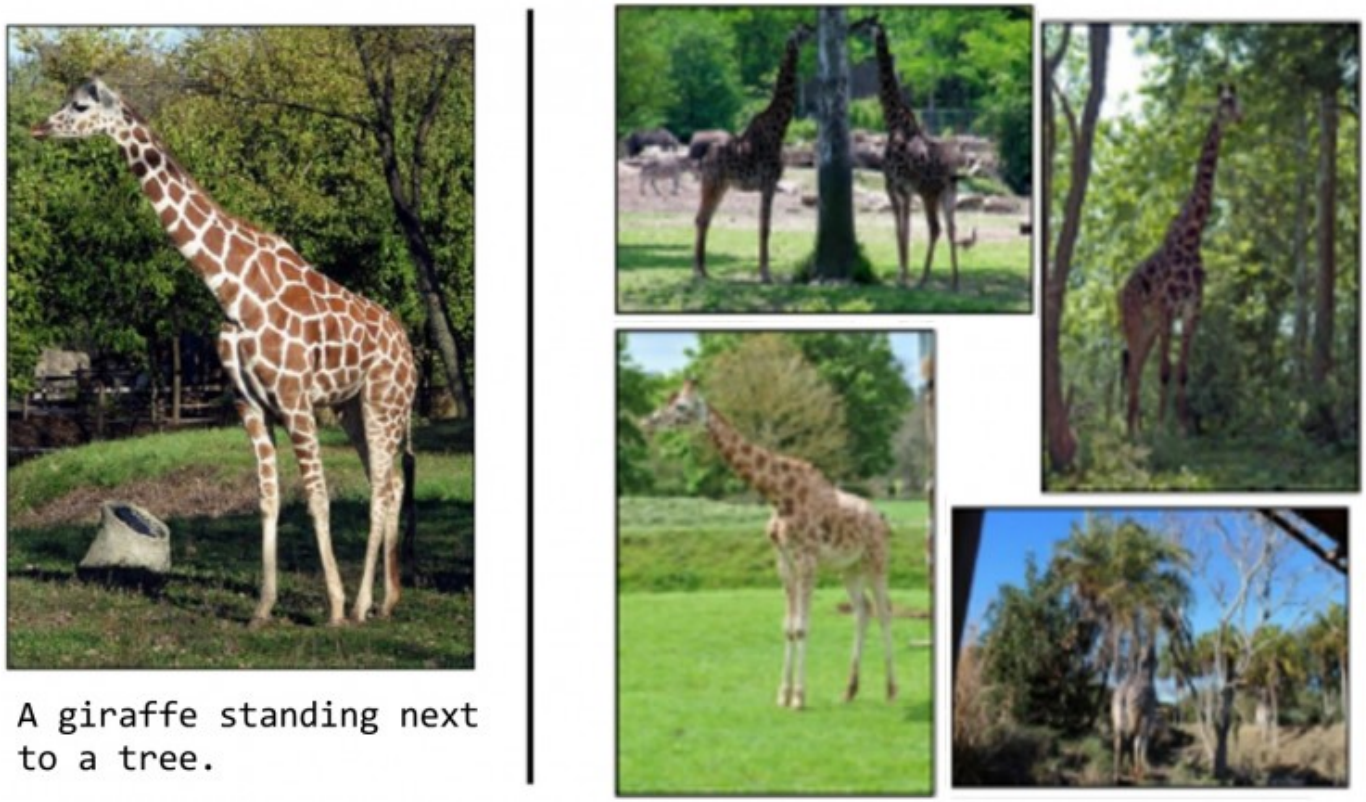
局限性

对于学习把图像映射到人类级别的文字注解，该神经图像注解生成器提供了一个十分有用的框架。铜鼓偶在大量图像—注解成对数据上训练，该模型学会了通过视觉特征抓取相关语义信息。

但对于静态图片而言，嵌入我们的注解生成器，将会聚焦于图像中对分类有用的特征，而不是对注解生成有用的特征。为提升每个特征里涵盖的与任务相关的信息，我们可以训练图像嵌入模型（用来对特征进行

编码的 VGG-16 网络）作为注解生成模型的一部分。这使得我们能为图像编码器调参，以更符合注解生成器的角色。

另外，如果我们仔细观察生成的注解，会发现它们基本是日常、常见的情形。下图便是例子：



几乎可以肯定，这图片是“长颈鹿站在一棵树旁边”。但是，如果我们看看其他图片，会立刻发现无论对于哪一张生成的注解都是“长颈鹿站在一棵树旁边”。这是因为训练集中的长颈鹿都出现在树旁边。

下一步

首先，如果你想要提升该模型，你需要看看谷歌的开源 Show and Tell 神经网络。它用 MS COCO 数据集和 Inception-v3 图像嵌入训练。

现有的前沿图像注解模型会包含一个视觉注意力机制（visual attention mechanism），使得模型能发现图像中感兴趣的部分，因而能在生成注解时选择性地聚焦。

另外，如果你对注解生成的前沿执行非常好奇，不妨看看这一篇论文：Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention。

雷锋网提醒：作者已将本文所有 Python 代码上传于 GitHub，请戳这里。

本课程面向深度学习开发者，讲授如何利用 TensorFlow 解决图像识别、文本分析等具体问题。课程跨度为 10 周，将从 TensorFlow 的原理与基础实战技巧开始，一步步教授学员如何在 TensorFlow 上搭建 CNN、自编码、RNN、GAN 等模型，并最终掌握一整套基于 TensorFlow 做深度学习开发的专业技能。

两名授课老师佟达、白发川身为 ThoughtWorks 的资深技术专家，具有丰富的大数据平台搭建、深度学习系统开发项目经验。

> 时间：每周二、四晚 20：00-21：00
> 开课时长：总学时 20 小时，分 10 周完成，每周2次，每次 1 小时
> 线上授课地址：http://www.mooc.ai/

相关文章：

教你从零开始在 TensorFlow 上搭建 RNN（完整代码）！

相关文章

图像注解 TensorFlow

如何利用微信监管你的TF训练

如何实现Tensorflow多机并行线性加速？

TensorFlow新功能解锁：可在TensorBoard中增加自定

TensorFlow Agents日前开源，轻松在TF中构建并行

文章点评：

我有话要说……

同步到新浪微博

提交

0

热门关键字

热门标签 微信小程序平台 微信小程序在哪 CES 2017 CES 2016年最值得购买的智能硬件 2016 互联网 小程序 微信朋友圈 抢票软件 智能手机 智能家居 智能手环 智能机器人 智能电视 360智能硬件 智能摄像机 智能硬件产品 智能硬件发展 智能硬件创业 黑客 白帽子 大数据 云计算 新能源汽车 无人驾驶 无人机 大疆 小米无人机 特斯拉 VR游戏 VR电影 VR视频 VR眼镜 VR购物 AR 直播 扫地机器人 医疗机器人 工业机器人 类人机器人 聊天机器人 微信机器人 微信小程序 移动支付 支付宝 P2P 区块链 比特币 风控

高盛 人脸识别 指纹识别 黑科技 谷歌地图 谷歌 IBM 微软 乐视 百度 三星s8 腾讯 三星Note8 小米MIX 小米Note 华为 小米 阿里巴巴 苹果 MacBook Pro iPhone Facebook GAIR IROS 双创周 云栖大会 智能硬件公司 智能硬件 QQ红包 支付宝红包 敬业福 支付宝敬业福 支付宝集五福 Waymo 虚拟现实 深度学习 人工智能 中国银联 蚂蚁金服 WRC CNCC prisma 索尼z3 cvpr 苹果换电池 锤子t3 mx4 pro 腾讯qrobot q影 智能灯 2015淘宝双十一销售额 双11总额 powerbeats 3 万圣节 科技 vibe z2 devonthink 微信小程序开发者文档 更多

联系我们 关于我们 加入我们 意见反馈 投稿

0