中国矿业大学

CHINA UNIVERSITY OF MINING AND TECHNOLOGY

本科生毕业设计（论文）

基于 LSTM 的图像描述方法研究与实现

Research and Implementation of Image Captioning Based on LSTM

作　者：刘奇

导　师：刘兵　副教授

中国矿业大学

2018 年 6 月

中国矿业大学

本科生毕业设计（论文）

基于 LSTM 的图像描述方法研究与实现

Research and Implementation of Image Captioning Based on LSTM

作　　者　刘奇　　学　　号　08143258

导　　师　刘兵　　职　　称　副教授

学　　院　计算机学院　　专　　业　电子信息科学与技术

二〇一八年六月

学位论文原创性声明

本人郑重声明:所呈交的学位论文《基于LSTM的图像描述方法研究与实现》,是本人在指导教师指导下,在中国矿业大学攻读学位期间进行的研究工作所取得的成果。据我所知,除文中已经标明引用的内容外,本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。本人完全意识到本声明的法律结果由本人承担。

作者签名:

2018 年6月4日

致谢

对于本毕业设计，最要感谢的是我的指导老师，中国矿业大学计算机学院的刘兵副教授。感谢老师在我的毕设期间不定期地监督和指导我的毕业设计工作，在现场和电话等方式进行交流中，总是能指出我研究和实践中的痛点和值得深入和改进的地方，专业而负责地 push 我进行毕设工作进度，并在我几次跑偏了不是太远的时候，对我以醍醐灌顶，使我重回正轨。从而，最终我得以在毕设课题所属的领域，得以窥见研究热点和前沿之一斑。佶屈聱牙，磕磕绊绊，“故余虽愚，卒或有所闻”。

感谢我校计算机学院，特别感谢 14 级的辅导员劳模李琳老师以及辅导员助理们，为我们付出了很多。感谢学院秦峰书记，在学院学术学风建设，学生科技创新，学科竞赛等活动上提供了大量的支持。感谢我系的陈岱，王冠军，王重秋，江海峰，徐新征，周勇老师，在我和队友参加科研竞赛和课程设计过程中给予我们悉心的指导和大力的支持。感谢陈岱，刘兵，杨小东老师为我在推荐免试研究生中给予我的帮助。同时感谢机电学院李伟，王建老师，为筹建大学生创新训练基地，组织管理学生创新创业活动等方面做出了巨大的贡献。感谢慕课网，GitHub，CSDN 等计算机技术学习网站和社区，让我从四年前对编程一无所知的小白，成为略懂一二的小白 plus。

特别感谢曾经和我并肩作战的队友李小硕，马志宁，刘灿斌，李博雅，吴孙阳，皮金勇，刘云飞，李天昊，秦铁鑫等，感谢计算机 14-7 班的同学们，感谢梅 2A408 的我的室友，以及其他给予我帮助和支持认识和不认识的人。另外，感谢我的爸爸妈妈还有姐姐对我全心全意的爱和支持，感谢我的女朋友她迟迟不出现使我能潜心学习和研究。

感谢在矿大，在徐州度过的四年，“风起楚汉天子地，梦醉鉴中放鹤亭”。回头望去，该走的弯路一点都没少走，失去了很多，收获的也同样多。

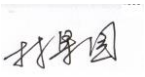
本文写完，表明我大学本科四年即将结束。何谓“大学”？何谓“大学之意义”？何谓“我的大学”？是“脱心志于俗谛之桎梏，真理得以发扬”，还是“为往圣继绝学，为万世开太平”，抑或是“忘记所有书本上所学的后所留下的东西”呢？

“溯洄从之，道阻且长。溯游从之，宛在水中央”，前路依然未知，只有不自欺，不相欺，且行且歌，成为问题的解决者，才能对未知进行把控，在极大的悲哀中体味极大的快慰。

“生存还是毁灭，这是一个问题”。

中国矿业大学本科毕业设计（论文）任务书

<table>
<tr><td colspan="4">设计（论文）题目：基于 LSTM 的图像描述方法研究与实现</td></tr>
<tr><td>学院</td><td>计算机科学与技术学院</td><td>专业年级</td><td>电子信息科学与技术</td></tr>
<tr><td>学生姓名</td><td>刘奇</td><td>学号</td><td>08143258</td></tr>
<tr><td>下达日期</td><td>2018-01-08</td><td>毕业设计（论文）日期</td><td></td></tr>
<tr><td colspan="4">1、设计（论文）的主要内容
Image Captioning 是一个融合计算机视觉、自然语言处理和机器学习的综合问题，它类似于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。本毕业设计需要设计实现基于 LSTM 的图像描述算法，并应用于实际场景。</td></tr>
<tr><td colspan="4">2、设计（论文）的基本要求
研究 Image Captioning 方法基于 encoder-decoder 模型。其中 encoder 一般为卷积神经网络，利用最后全连接层或者卷积层的特征作作为图像的特征，decoder 一般为递归神经网络，主要用于图像描述的生成。由于普通 RNN 存在梯度下降的问题，RNN 只能记忆之前有限的时间单元的内容，而 LSTM 是一种特殊的 RNN 架构，能够解决梯度消失等问题，并且其具有长期记忆，研究在 decoder 阶段采用 LSTM 模型的方法。设计实现基于 LSTM 的图像描述算法和软件。课题难度较大，工作量饱满，实现 encoder-decoder 模型，并开发设计相应软件，撰写毕业设计论文。
论文应有正确，清晰的图表说明，理论论述严谨，软件设计合理，功能设计齐全。最终需交纳论文电子稿以及软件设计源程序。还需交纳 3000 字的英文翻译。</td></tr>
</table>

指导教师签字：  　　教学院长签字： 

中国矿业大学毕业设计指导教师评阅书

学生姓名	刘奇	学号	08143258
设计（论文）题目	基于 LSTM 的图像描述方法研究与实现		

指导教师评语（①基础理论及基本技能的掌握；②独立解决实际问题的能力；③研究内容的理论依据和技术方法；④取得的主要成果及创新点；⑤工作态度及工作量；⑥总体评价及建议成绩；⑦存在问题；⑧是否同意答辩等）：

毕业选题具有重要理论研究和应用价值，该生能够通过图书馆、互联网及其他资源或信息检索工具，进行资料查询、文献检索，梳理总结图像描述方法的国内外研究现状。基于计算机工程领域相关背景知识进行毕业设计课题的需求分析，能够在毕业设计过程中学习 TensorFlow、卷积神经网络、LSTM 循环网络等新技术、新知识，基于学习的模型和理论，设计并实现了基于 LSTM 的图像描述算法和原型系统，说明该生具有不断学习和适应计算机技术快速发展的能力。工作量饱满，态度端正。论文结构合理，条理清晰，论述充分，达到了本科生毕业设计要求。同意答辩！

成 绩： 97　　指导教师签字：

2018 年　6 月　1 日

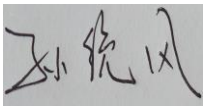
中国矿业大学毕业设计评阅教师评阅书

学生姓名	刘奇	学号	08143258
设计（论文）题目	基于 LSTM 的图像描述方法研究与实现		

评阅教师评语（①选题的意义；②基础理论及基本技能的掌握；③综合运用所学知识解决实际问题的能力；④工作量的大小；⑤取得的主要成果及创新点；⑥写作的规范程度；⑦总体评价及建议成绩；⑧存在问题；⑨是否同意答辩等）：

选题具有现实的研究意义。课题较难，工作量较大，符合毕业设计的要求。该生具有查阅文献资料的能力，具有扎实的理论基础和自学能力。构建了基于 LSTM 和多种深度网络模型组合的以 RNN 和 VGG 组合的图像描述，并通过实验进行了验证，具有综合利用所学知识解决实际问题的能力和一定的科研能力。论文叙述完整，思路清晰，但论文格式还需改进。同意答辩。

成 绩： 92

评阅教师签字： 

2018 年 6 月 4 日

中国矿业大学毕业设计答辩及综合成绩

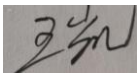
答　　辩　　情　　况

提　出　问　题	回　答　问　题				
	正确	基本正确	有一般性错误	有原则性错误	没有回答
应用场景是什么？	√				
索引序列如何映射？	√				
词库是如何定义的？	√				
6662 如何进行编码？用了多少二进制编码？	√				
测试图像如何设置输出？	√				

答辩委员会评语及建议成绩：

96 按期圆满完成任务书规定的任务；能熟练地综合运用所学理论和专业知识；立论正确，计算、分析、实验等正确严密，结论合理；独立工作能力较强，科学作风严谨；设计（论文）有自己的独到见解，水平较高。设计的软件在答辩演示中能够正常运行，功能设计合理，达到了设计方案中的要求，设计工作量饱满；具备较强的本专业工程项目管理能力。说明书条理清楚，论述充分，文字通顺，符合技术用语要求，符号统一，编号齐全，书写工整，图纸完备、整洁、正确。

答辩时，思路清晰，论点正确，回答问题有理论根据，基本概念清楚，对主要问题回答正确、深入。

答辩委员会主任签字： 

2018 年　6 月　7　日

学院领导小组综合评定成绩：优秀

学院领导小组负责人：

年　　月　　日

摘　要

本文旨在解决让计算机根据图像内容来自动生成对应的自然语言描述的问题，即让计算机学会“看图说话”，其属于人工智能领域连接计算机视觉和自然语言处理的基础性问题。本文研究和构建了基于 RNN 特别是 LSTM 和多种深度卷积神经网络的端到端图像描述模型，并比较分析了模型间的描述性能。模型整体为 CNN+RNN 的架构，通过卷积神经网络来提取输入图像的特征编码，将其作为输入传达到 RNN 解码器，模型通过随机梯度下降算法最小化训练图像描述的损失函数来端到端地训练。在多种数据集上的结果表明，模型可以通过学习来使得描述的精确度逐渐上升。最终的模型具备了对于图像内容产生自然语言描述的能力。其中在 Flisk8k 数据集上，模型的 BLEU-1 达到了 0.508，BLEU-2 达到了 0.300，超过了部分现有研究模型的结果。此外，对图像描述模型细节进行了可视化，并建立了完整的应用系统。未来将在模型性能，评价标准多语言支持等方面做进一步的研究。

本论文有图 108 幅，表 13 个，参考文献 46 篇。

关键词：图像描述;长短期记忆网络；循环神经网络；

Abstract

This paper aims to solve the problem of letting the computer automatically generate natural language descriptions corresponding to the content of an image, that is, letting the computer learn to “see and speak”, which is a fundamental problem in artificial intelligence that connects computer vision and natural language processing. This paper builds several end-to-end image captioning models by combining RNN especially LSTM with several deep convolutional neural networks, and performances are compared among models. All the models are belonged to the CNN+RNN architecture. The front convolutional neural network is used to extract the feature code from the input image, which is passed as input to the RNN decoder. All the models use the stochastic gradient descent algorithm to minimize the loss function and are trained end-to-end. The results on several datasets show that the models can learn to make the description more accurate. The final models are able to generate natural language descriptions according to the image content. Results on the Flisk8k dataset shows that the BLEU-1 of the one model reached 0.508, and the BLEU-2 reached 0.300, which exceeded some of the existing models. In addition, an application system is established Based on the image description models and details of models are shown by several data virtualization technologies.

There are 108 figures, 13 tables and 46 references.

Keywords: Image Captioning; LSTM; RNN;

目录

Contents

1 绪论
1 Introduction

1.1问题概览（Overview）

本文研究的图像描述（Image Captioning）问题，是让计算机根据图像内容自动生成自然语言的描述[3]，是人工智能领域中将计算机视觉和自然语言处理相结合的基础性问题[1]。

早在现代电子计算机发明前一个多世纪的机械式计算机时代，Ada Lovelace等先驱就已经思考能否让计算机进行纯粹的计算之外的任务 [13]。Alan Turing认为机器最终能在所有纯智力的领域同人类竞争，但是对于如何开始这一过程，一种途径是让计算机能够从事人类认为非常抽象的活动，例如下棋；另一种途径是赋予机器以类似人的感官，然后让计算机学会使用这些感官来认知世界并和人类正常交流，类似于对幼儿进行教授。Turing 认为两种途径都是可取的[14]。

“人工智能”概念提出者 John McCarthy 等人在上世纪 50 年代的机器国际象棋项目[15] ，以及近几年 Google Deepmind 团队的机器围棋 Alpha Go 项目[16] ，属于前一种途径取得的突破。如今，人工智能已经成为一个拥有众多活跃研究课题并且仍在蓬勃发展的领域，其目标之一和后一种途径一脉相承：赋予计算机观察并理解纷杂的世界，并通过自然语言和人类交流的能力[17]。

和现有的已经被充分研究的图像分类，物体检测任务相比，图像描述是目前计算机视觉领域研究的热点[4]，同时也具有相当大的难度和挑战性。

首先，生成的描述必须捕捉到图像中出现的物体。其次是还描述这些物体之间的相互联系以及对象的属性和涉及到的活动。在此，生成的语义上的认知（knowledge）必须被以自然语言（比如说英语）的形式表达出来，描述必须是看上去能够被读懂的（visual understanding）。用通俗的话说，就是让计算机学会“看图说话”，并且说的还要是“人话”。

从图像中自动生成描述也有实际应用的意义，例如对视力障碍人士提供帮扶，自动标记每天上传到互联网上海量的网络视频节省以人工标记成本[2], 此外还可以用于根据内容对图像和视频进行追溯，以及机器人和人的交互等。

1.2研究现状和分析（Related Work）

“自底而上”（bottom-up）和“自顶而下”（top-down）是进行图像描述的两种主要的途径[2]。早期的研究始于“自底而上”，最新的研究结果大多采用“自顶而下”的方法。

对于“自底而上”的方法，特点是先从一张图片中获取观察到的对象，然后将这些确定的对象组合成一段描述。其缺点在于，整个过程并非是“端到端（end-to-end）”的，即中间结果需要人为的干预和设计。[6]首次提出通过了建立在人工设定的特征的图形模型来建立图像和描述之间的映射关系。[10]首次将神经网络应用到了图像描述任务中，提出了多模态（multi-model）的处理方法，同时指出了神经网络可以用来解码（encode）处理从经过卷积神经网络（Convolution Neural Network，CNN）作为编码器（encoder）从图像中得到的表示（representation），并且在结果的隐藏维度和词汇嵌入（word embedding）包含了图像的语义信息。[38]将图像描述问题拆解成两个更细的问题。首先，训练一个 CNN 和一个双向循环神经网络（bi-directional Recurrent Neural Network, BRNN）来学习图像分割（fragment）和对应的描述的多模态嵌入，并在信息检索任务中取得了当时的最好结果。其次，训练一个 RNN 来学习从原始图像中检测到的物体分割来组成一段描述文字。和[38]的方法类似，[39]是先训练一个物体检测器来确定图像中的分割并检测词汇，然后经过语言模型和多模态相似度模型处理最终生成描述。

对于“自顶而下”的方法，总的来说，是生成一张图像的语义表示（semantic representation），然后使用其他不同的结构，例如循环神经网络的结构将其编码（decode）进描述中。由于不是分别检测物体再生成描述，而是从图像到描述“一条龙”，即端到端地训练出基于循环神经网络的模型[2]，所以称为“自顶而下”。[1]和[11]二者均使用了性能更加优越的卷积神经网络对图像进行编码，并且使用长短期记忆网络（Long Short Term Memory, LSTM）[12]取代了[10]中所用的前馈网络。 [1]同时指出了 LSTM 不需要在每个时间步都接收图像表示依然能够获得当时最好的结果。 [11]提出了模型在视频描述任务上的应用。

除了以上两种方式外，还有方法在神经网络中引入注意力（attention）机制，使得模型能够学习到应当注意的方面而忽视其他的方面。这在[41]中的强化学习技术和变分自编码[40]中得到了验证，理论上同样适用于的图像描述中编码器解码器的模型结构。而本文采用的是相对结构更简单，类似于[1]和[11]的“自顶而下”的方式。

就目前而言，虽然图像描述模型的性能相较于之前有了一定提升甚至在部分指标上和人类平齐， [1]同时指出现有的评价指标无法评价反映机器描述和人类的描述的本质差别，现有的方法产生的结果和人类的平均水平还有很大的差距。对于中文图像描述的研究目前比较少，[5]将机器翻译和图像描述结合，基于现有的英文数据集构建了中文数据集，并构建了相应的中文描述模型，为

中文图像描述奠定了初步基础。图像自动描述的研究还任重而道远，需要在理论和方法创新，模型精确度提升，乃至评价指标，多语言的支持等等方面进行改进。

1.3主要工作（Contributions）

就图像描述任务，本设计首先基于普通 RNN 和 VGG16 网络共同构建了图像描述模型，其中包括对于模型架构的设计和实现，对于 CNN 网络的预训练后对图像进行编码表示，构建数据集词汇表和词汇嵌入，描述的标记化等。随后在 MS COCO 数据集上分别基于小样本和大样本进行了训练，得到了能够根据图像内容产生自然语言描述的描述模型。其次是本文的工作重点在于，基于 LSTM 结合几种经典的卷积网络 VGG16，InceptionV3, ResNet50 共同构建了数种图像描述模型，在 MS COCO，Flickr8k, Flick30k 数据集上进行训练和测试，采用机器翻译中的评价指标对模型的描述性能进行了分析对比，还将模型应用于“开放式”检测任务中并对生成的结果进行了定性评价。另外对于模型的中间结构中间过程进行了数据可视化处理，还将模型推广到实际的应用中，构建了 B/S 端的完整在线图像描述应用，将研究和应用紧密地结合。

1.4章节安排（Outline）

论文的结构安排如下：

第 1 章，绪论部分，概览本设计的研究对象、研究背景和意义，目前对于图像描述问题的研究现状，简述本文的主体组织结构；

第 2 章，相关理论概述。主要阐述了图像描述任务作为监督学习任务的目标，所采用的基于批处理随机梯度下降优化方法和反向传播算法。在神经网络章节，阐述了图像描述模型所使用到的卷积神经网络，循环神经网络的基本结构和原理；

第 3 章，本设计的研究点之一，基于标准 RNN 构建图像描述模型。包括模型的整体设计和构建，CNN 的预训练来对输入图像进行表示，输入描述表示方法，实验和相关结果；

第 4 章，本设计最主要的研究点，基于 LSTM 的图像描述模型。包括将 LSTM 作为描述模型的处理核心，分别结合几种经典的卷积网络构建了相应几种的图像描述模型，模型构建和训练细节，评价指标，实验以及相关结果。

第 5 章，模型和数据可视化。使用工程的方法，对模型结构和运行，中间的数据结果等分别进行了可视化；

第 6 章，将研究扩展到应用。以 B/S 体系结构的在线图像描述应用系统为载体，对研究理论和前期成果提出了扩展到实际应用的要求，以工程实践内容倒逼模型细节的优化，增加了很多额外的需求，桥接了研究和工程；

第 7 章，总结和展望。对本文的研究内容和工作进行了总结，并对未来的可能的工作指出了值得进一步探索和研究的方向。

2 深度学习背景和原理
2 Deeplearning Background

2.1引言（Introduction）

图像描述任务作为连接人工智能领域两大研究范围 CV 和 NLP 的综合性问题，传统的基于自底而上且人工干预和设计中间层的方法正在被越来越少应用到，而使用深度卷积网络和循环神经网络自顶而下构建图像描述端到端模型的方法将被在本设计中采用。本章将介绍本设计所涉及到的最为核心的背景知识，包含任务的属性和核心优化算法以及所用神经网络的基本构造和内部工作原理。

2.2监督学习（Supervised Learning）

深度学习是机器学习的一种，是通向人工智能的一种途径，如图 2-1 所示。现代术语“深度学习”通常上是深层人工神经网络代名词[18]，关于“深度”是因为其诉诸于多层次组合原理，旨在让计算机通过构建简单概念来学习复杂的概念，如果绘制出表示这些概念如何建立在彼此之上的图，就得到一个层次很多（很“深”）的图，通常直观表现在在人工神经网络的层数相对比较多。而对于“学习”，Mitchell 在[19]给出了普遍为学术界接受的形式化定义，即表现为程序能够通过数据来进行自我提升。

图 2-1 深度学习、机器学习和人工智能的关系

Figure 2-1 Venn diagram of Deep Learning, Machine Learning and Artificial Intelligence

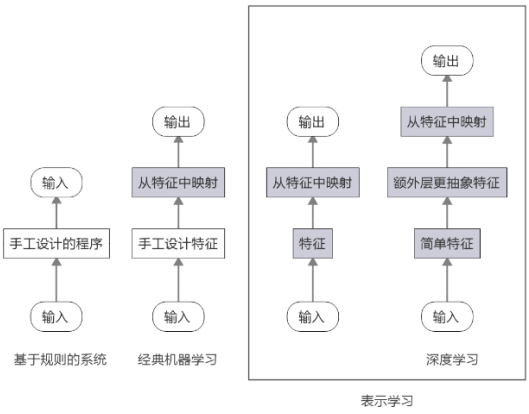


图 2-2 深度学习系统和其他 AI 系统处理流程比较

Figure 2-2 Flowchats showing the differences among different AI Systems

计算机从数据中习得结果称为“模型（model）”，学得的模型反映了关于数据的潜在规律。学得模型的过程称为“训练（training）”，训练使用的数据为训

练数据（training data），借助模型对新的情况进行预测的过程被称为测试（testing）。潜在规律本身，被称为“真实（ground truth）”，训练的目的就是为了找到或者逼近真实。

若提供的训练数据都是有标记，那么该学习任务就属于“监督学习（supervised learning）”。很多要求计算机解决的现实问题可以形式化地表述为寻找一个映射$f: X \to Y$，其中的X是输入空间，Y是输出空间。但是通常对于此类问题，我们很难找到f，而获得样例$(\mathrm{x}, \mathrm{y}) \in (\mathrm{X}, \mathrm{Y})$是相对简单的。

假设训练数据是从数据生成分布D中获得的 n 个样例$\{(x_1, y_1), \ldots, (x_n, y_n)\}$即$(x_i, y_i) \sim D$，样例彼此互为独立同分布（independent and identically distributed, iid）。那么，要学习得到的映射$f: X \to Y$即是从一个候选函数集合中找到和训练样本尽可能一致的。我们的目标就是找到一个$f^* \in \mathcal{F}$使得其满足：

$$f^* = arg \min_{f \in \mathcal{F}} E_{(x,y)\sim D}\, L(f(x), y) \tag{2.1}$$

其中损失函数（loss function）$L(\hat{y}, y)$用于衡量$\mathrm{f} \in \mathcal{F}$预测的标记$\hat{y_i} = f(x_i)$和真实标记$y_i$之前的差距。也就是寻找一个函数$f^*$在数据生成分布D上使得整体损失最小。但是很显然，不可能得到D上所有的元素。因此，在独立同分布的假设条件下，对于有限的训练数据而言，我们希望得到：

$$f^* \approx arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) \tag{2.2}$$

换句话说就是在有限的训练样例上对损失进行优化。

由于训练数据是有限的，我们不单单希望f能在训练数据上拟合得好，更加希望对于其他不在训练数据中的样例$(\mathrm{x}, \mathrm{y}) \sim \mathrm{D}$依然能够产生很好的预测结果，也就是泛化。对于在整个$f \in \mathcal{F}$集合中选择泛化能力较好的f，我们通常引入正则化项（regularization term）R：

$$f^* \approx arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) + R(f) \tag{2.3}$$

其中R为一个标量函数，体现了对于不同学习算法的在某些方面的性能偏好，而无视它们在训练数据上的拟合效果的强弱。关于正则化项的选择，“奥卡姆剃刀（Occam’s razor）”是一种常用的原则[20]，即“若有多个假设和观察一致，则选择最简单的那个（*Suppose there exist two explanations for an occurrence. In this case the simpler one is usually better.*）”。换句话说，正则化可以视作对于选取函数复杂性的度量，正则化项的引入，使得式(2.3)得到在训练数据上拟合得很好并且更简单的f，同时也旨在弥补式(2.1)和式(2.2)间的差异性[17]。

2.3优化算法（Optimization）

深度学习背景下的优化是指改变x以最小化或者最大化某个函数$f(x)$。通常可以最小化$f(x)$来指代大多数的优化问题，最大化可以经过最小化$-f(x)$来实现，需要最小化或者最大化的函数被称为目标函数或者准则，当其进行最小化时，也称目标函数为代价函数（cost function）、损失函数（loss function）或者误差函数（error function）。

2.2.1梯度下降

梯度下降法（gradient descent, GD）是一种常用的采用梯度下降最小化目标函数的一阶优化方法，也被称为“最速下降法”。使用梯度下降法找到一个函数的局部最小值。若相反的，向梯度的正方向移动，会接近函数的局部极大值点，这个过程称之为“梯度上升法”。

梯度下降法针对多维输入函数$f: \mathbb{R}^n \to \mathbb{R}$输出为一维（标量）“最小化”的概念才有意义。梯度（gradient）是相对一个向量求导的导数：f的导数是包含所有偏导数的向量，记作$\nabla_x f(x)$，在负梯度方向上移动可以减小f，因此这种方法被称作“梯度下降法”。根据梯度下降法，建议新的点为:

$$x' = x - \epsilon \nabla_x f(x) \tag{2.4}$$

其中ϵ称为学习率（learning rate）为一个用来确定步长大小正标量，普遍的方式是选择一个小常数，也可以通过一些原则来设定。梯度下降法在梯度的每一个元素为零时（在实践中，很接近零时）收敛。在某些情况下可通过解方程$\nabla_x f(x) = 0$直接找到临界点。因此，总体上讲梯度下降算法可以分成两步：1)反向传播（backpropagation，见章节 4.2）计算梯度；2)将参数向梯度相反的方向移动一小步来更新参数。

2.2.2随机梯度下降

梯度下降算法（GD）要沿着整个训练集的梯度方向下降，当训练样本很大，输入维数也很大的时候，每次迭代都要处理所有样本，效率会比较低下。出于实际的考虑，往往采用随机梯度下降（stochastic gradient descent, SGD），每次计算一个样本来对模型参数进行更新。但是，随机梯度下降由于每次计算只考虑一个样本，使得每次迭代不一定都是朝着模型整体最优的方向。如果样本噪声（noise，无用和干扰的数据）较多，基于随机梯度下降的模型容易陷入局部最优解而收敛。深度学习中的 SGD 按照数据生成分布抽取 m 个样本组成“一批”，利用这批样例本上的梯度信息完成一次模型更新，因此基于“批处理”的 SGD 称之为 mini-batch SGD。遍历一次训练样本成为“一轮（epoch）”

[24]。批处理 SGD 能够获得相对单个样本更鲁棒的梯度信息[21]。目前对于深度神经网络，如卷积神经网络、循环神经网络等，均采用批处理的 SGD。

表 2-1 基于批处理的随机梯度下降算法

Table 2-1 mini-batch SGD

算法：基于“批处理”的随机梯度下降在第k个训练迭代的参数更新
Given: 初始参数θ *Given: 学习率ϵ_k* repeat: *从训练集中采样m个样本$\{(x_1, y_1), \dots, (x_m, y_m)\}$组成一个小批* *通过反向传播计算梯度估计：$\nabla_\theta g(x) \approx \nabla_\theta \left[\frac{1}{m}\sum_{i=1}^{m} L\left(f_\theta(x_i, y_i)\right) + R(f_\theta)\right]$* *计算更新的方向：$\Delta\theta := -\epsilon_k \nabla_\theta g(x)$* *进行参数更新：$\theta := \theta + \Delta\theta$* until *收敛*

2.4反向传播（Backpropagation）

深度学习背景下，前向传播是指输入x，经过各层隐藏单元，最终产生输出$\hat{y}$的过程。训练过程中，前向传播持续向前直到产生一个标量代价函数$\mathcal{J}(\theta)$，反向传播（back propagation）则是指计算该标量函数关于输入的梯度的方法，通过递归地使用链式法则来实现。原则上反向传播可以计算任何函数的导数（对于某些函数，报告该函数的导数是未定义的）[13]，只是对于学习算法而言，通常最需要的梯度是上述代价函数关于参数的梯度即$\nabla_\theta \mathcal{J}(\theta)$，使用相同的过程同样可以求得关于输入$x$的梯度。

表 2-2 反向传播算法

Table 2-2 Back Propagation

算法：反向传播算法用于训练
Given: 训练集$\{(x_1, y_1), \dots, (x_N, y_N)\}$ *Given: 迭代次数（即训练轮数，epoch）：T* *for t in T:* *while 训练数据未遍历完 do:* *for l in L:* *前馈计算每一层的状态和激活值x^l，直到计算出最终误差z* *for $l = L, L-1, \dots, 1$:* *反向传播计算出第l层误差对该层参数的导数：$\frac{\partial z}{\partial\left(vec(w^l)^T\right)}$；* *以及第$l$层误差对该层输入数据的导数：$\frac{\partial z}{\partial\left(vec(x^l)^T\right)}$；*

更新参数： $w^l \leftarrow w^l - \eta \frac{\partial z}{\partial (w^l)}$ *（*η*是每次随机梯度下降的步长）*

Return: $w^l, l = 1,2, \ldots L$

2.5 神经网络（Neural Networks）

2.5.1 基本神经网络

构成人工神经网络的基本单元是人工神经元如图 2-3 所示，简称神经元（neuron），其设计灵感来源于生物神经元。生物神经元有一个阈值，当接受的输入信号的积累效果超过阈值时，就处于兴奋状态；否则处于抑制状态。人工神经元使用一个非线性激活函数，输出一个激活值。假设一个神经元接受n个输入$x = (x_1, x_2, \ldots x_n)$，z表示神经元获得输入信号x的加权和，输出a为该神经元的激活值：

$$z = w^T x + b \tag{2.5}$$

$$a = f(z) \tag{2.6}$$

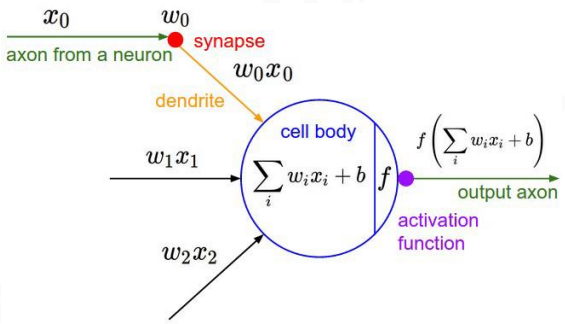


图 2-3 人工神经元

Figure 2-3 a single Neuron

其中w为 n 维权重向量，b 是偏置，f是激活函数（activation function）。当激活函数为 0 或 1 的阶跃函数，神经元就是感知器，即最简单的人工神经网络[22]。但是阶跃函数是不可导的，为了使用最优化的方法来求解，需要引入连续的非线性函数作为激活函数。

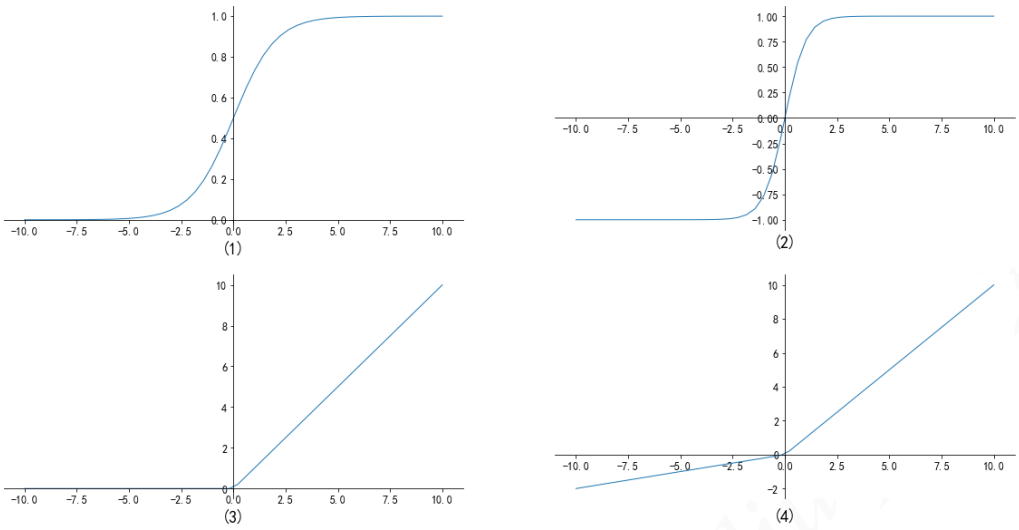


图 2-4 常见非线性激活函数。从(1)到(4)分别为 Sigmoid, tanh, ReLu, Leaky ReLu 激活函数

Figure 2-4 Activation Functions. (1) Sigmoid, (2) tanh, (3) ReLu, (4) Leaky Relu

常见的激活函数如图 2-4 所示有：S 形（Sigmoid）函数：$\sigma(x) = 1/(1+e^{x})$；双曲正弦（tanh）函数：$tanh(x) = (e^{-x} - e^{x})/(e^{-x} + e^{x})$；线性整流（ReLu）函数：$relu(x) = max(0,x)$；带滞露的线性整流（*Leaky Relu*）函数：$f(x) = max(ax,x), a \in (0,1)$，等。

单个神经元的作用十分有限，通过增加神经元形成组，重复的矩阵相乘和逐元素的这种非线性就可以构建起基本的神经网络[17]。

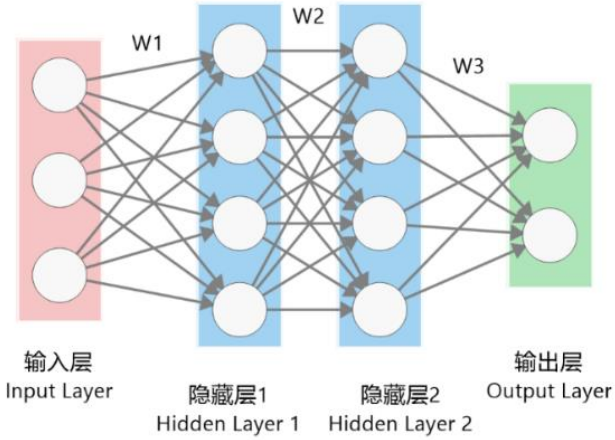


图 2-5 一个 3 层神经网络示意

Figure 2-5 Illustration of a 3-layer Neural Network

如图 2-5 所示，为一个 3 层的神经网络（包含两个隐藏层和一个输出层，并且层与层之间是全连接。由于输入层没有可调的权重参数故不计入其中）,由此可以得到$f(x) = w_3\sigma(w_2\sigma(w_1x))$，其中$w_1, w_2, w_3$为权重矩阵，$\sigma$为逐元素的非线性激活函数处理。

2.5.2 卷积神经网络

卷积神经网络 [26]，是一种专门用来处理具有类似网格结构数据的神经网络。卷积神经网络的特点在于隐藏层分别为卷积层和汇聚层（pooling layer，又叫做下采样层）[18]。

卷积层通过一块块卷积核（convolutional kernel）在原始数据上平移来提取特征。换句话说，每个卷积核都是一个特征提取器。在卷积神经网络中只涉及离散卷积的情形。把一张二维图像I作为输入，需要使用一个二维的核K，则对于位置i,j：

$$H(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i,j)K(i-m, j-n) \tag{2.7}$$

卷积是可交换的，因此也等价于：

$$H(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(i,j) \tag{2.8}$$

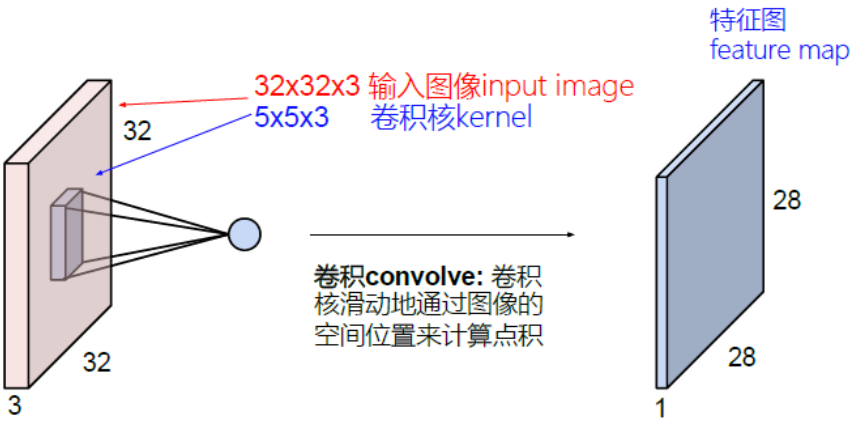


图 2-6 使用 5x5 的卷积核对$32 \times 32 \times 32$的输入进行卷积示意图

Figure 2-6 Illustration of a 5×5 kernel over a $32 \times 32 \times 3$ input array

如图 2-6 是卷积操作的示意图，输入图像存储为$32 \times 32 \times 3$的多维数组，使用5×5的卷积核，深度总是和输入的深度保持一致，不进行输入填充（padding），移动步幅（stride）为 1，则一个这样的卷积核就得到了一个$28 \times 28 \times 1$的特征图，可以理解为图像经过滤波之后的一个通道，通常一个卷积层有很多个这样的卷积核假设为k，通过堆叠卷积后的输出，从而得到深度为k输出层。

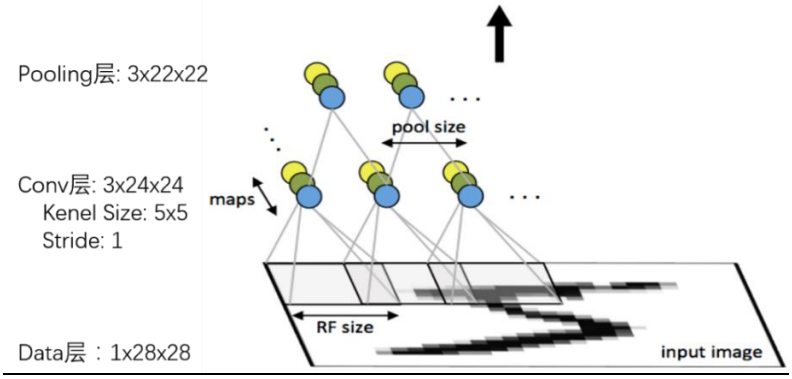


图 2-7 卷积网络中的参数共享

Figure 2-7 Illustration of weight sharing in CNN

使用卷积来替代全连接，第l层的每个神经元都只和第$l-1$层的一个局部窗口内的神经元相连，构成一个局部连接网络，相同特征图内，不同空间位置的神经元共享卷积核权值，如图 2-7，因此卷积层显著降低了要学习的参数数量。

汇聚层通过汇聚特征即特征降维，而后稀疏参数来较少要学习的参数，降低网络复杂度。汇聚层最常见的包括最大汇聚（max pooling）和平均值汇聚（average pooling），使得特征提取具有“平移不变性”，即对于输入进行少量平移后，经过汇聚处理后的大部分特征不会发生变化[13]。

通常卷积神经网络通过堆叠卷积层以及增加汇聚层来控制计算精确度来构建[17]。CNN 还有其余组件如全连接层（full-connected layer，简称 fc layer），Dropout[31]等，这些组件进行组合从而构建起 CNN 的架构。 另外[27]表明，汇聚操作并非是卷积神经网络所必须的，使用一种 stride convolutional layer（步幅全卷积层）来代替汇聚层进而构建出只含卷积操作的网络即全卷积网络，其实验结果显示这种改造后的卷积网络在某些性能上能够超过传统的卷积网络。

2.5.3 循环神经网络

循环神经网络，是一类专门用来处理序列数据的神经网络模型。循环神经网络的特点在于模型的不同部分共享参数，使得模型能扩展到不同形式（通常是序列长度不同）的序列并进行泛化。对于 RNN 模型而言，输入序列$x^{(1)}, x^{(1)}, \ldots, x^{(l)}$被看作随着时间步（time step，指序列的位置而非现实世界的时间）而递进的时间序列，RNN 对于当前时间步的计算还依赖于上一时间步的计算结果。对于循环神经网络，时间步t的状态：

$$h^{(t)} = f_\theta\left(h^{(t-1)}, x^{(t)}\right) \tag{2.9}$$

显然从递推公式可知，在每个时间步，用于参数化的f使用相同的参数。对应于最基本的循环神经网络的结构如图 2-8，表示没有输出的循环神经网络。

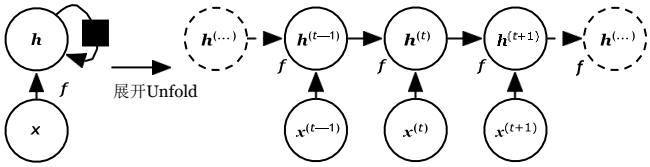


图 2-8 没有输出的循环神经网络结构折叠和展开示意图

Figure 2-8 A recurrent network with on outputs

通常在某些时间步，我们希望能够产生相应的输出或者预测。标准的循环网络如图 2-9 所示:

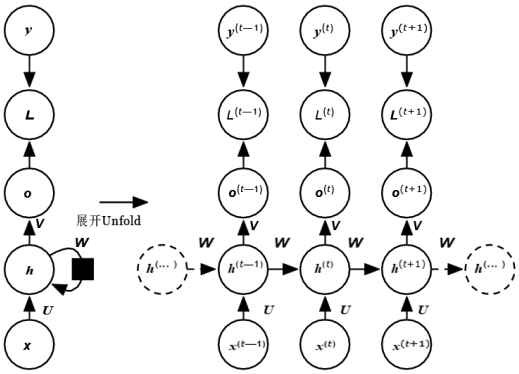


图 2-9 标准的循环神经网络的折叠和展开形式

Figure 2-9 A standard RNN

其中隐藏状态层，是由如图 2-10 所示的处理单元 A 在随时间步重复构建而成的，标准的 RNN 处理单元 A 中只有单一的一个层：

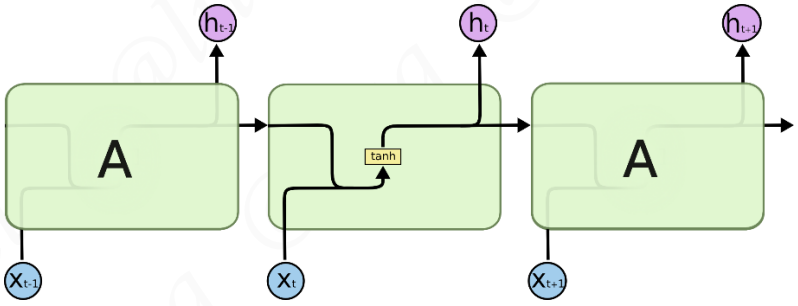


图 2-10 标准循环神经网络

Figure 2-10 a standard RNN

对于第t个时间步的状态：

$$h^{(t)} = tanh W \begin{bmatrix} h^{(t-1)} \\ x^{(t)} \end{bmatrix} \tag{2.10}$$

其中$\mathrm{h} \in \mathbb{R}^{\mathrm{n}}$，权重W形状为$n\ x\ 2n$。

[12]提出的长短期记忆网络（Long Short Term Memory, LSTM）是标准循环神经网络的改进版本。其中的隐藏状态层，通过不断重复如图 2-11 所示的处理

单元 A 来构建，取而代之的是每个处理单元中有四个不同的网络层，即所谓的“门”：

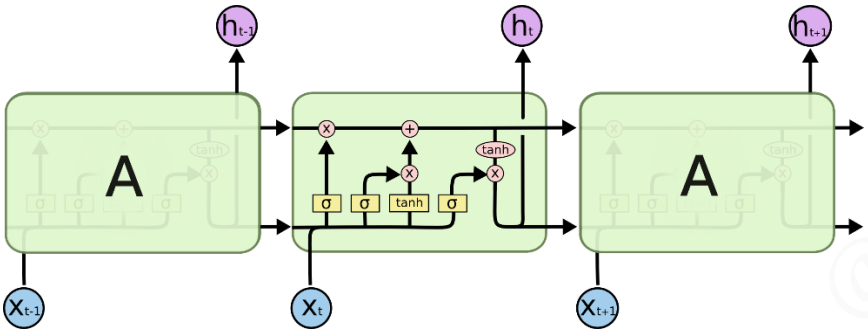


图 2-11 长短期记忆网络

Figure 2-11 an LSTM

LSTM 的门机制的引入，解决了标准 RNN 中存在的梯度消失和梯度爆炸的问题，从而使得信息能够在序列中较长距离也可以保留和传递，实现了信息的“长期依赖（long-term dependencies）”。遗忘门（forget gate）、输入门（input gate）、输出门（output gate）结构都相同，主要由 Sigmoid 激活函数和点积操作构成。Sigmoid 的值域在[0, 1]，所以门控制器描述了信息通过的比例，例如取值为 0 代表这部分信息不能通过，理解为“全部遗忘”，而取值为 1 代表所有信息均通过，理解为完全保留该分支的记忆。在第t个时间步，更新的机制如下：

$$\begin{bmatrix} i \\ f \\ o \\ g \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W \begin{bmatrix} x^{(t)} \\ h^{(t-1)} \end{bmatrix} \tag{2.11}$$

$$C^{(t)} = f \odot C^{(t-1)} + i \odot g \tag{2.12}$$

$$h^{(t)} = o \odot tanh\left(C^{(t)}\right) \tag{2.13}$$

其中σ为 Sigmoid 激活函数，tanh表示前面提到的双曲正弦激活函数，⊙表示元素积。其中g（又是被称为“屏蔽门（block gate）”，取值范围在-1 到 1 之间），用来调整“记忆单元（memory cell）”，即C能够在很长的时间上随时间进行反向传播（backpropagation through time）而不中断[17]，因此是 LSTM 能够保持长期依赖特征的体现。

2.6本章小结（Summary）

本章介绍了本设计涉及到的深度学习部分相关概念和原理，指出了本设计作为监督学习要达到的目标即要通过优化来最小化目标函数。介绍了梯度下降和本设计采用的基于批处理的随机梯度下降算法，来实现优化中对于参数的迭代更新，已经优化过程内部所使用的反向传播算法。由于深度卷积网络和循环神经网及其特例 LSTM 是本

设计使用模型的最重要构成，所以也从构成神经网络的人造神经元，到基本神经网络，再到卷积网络以及循环神经网络的基本构成做了简要介绍。

3 基于标准 RNN 和 VGG 的图像描述模型
3 Image Captioning Model Based on Standard RNN and VGG

3.1引言（Introduction）

在上一章节中，我们介绍了本设计的部分深度学习背景和原理，接下来，我们需要针对图像描述任务开始着手设计和构建我们的模型。这一章将采用最为经典的卷积神经网络之一的 VGG，并采用标准的 RNN 作为描述生成模型，共同构建端到端的图像描述模型。

3.2相关工作（Related Work）

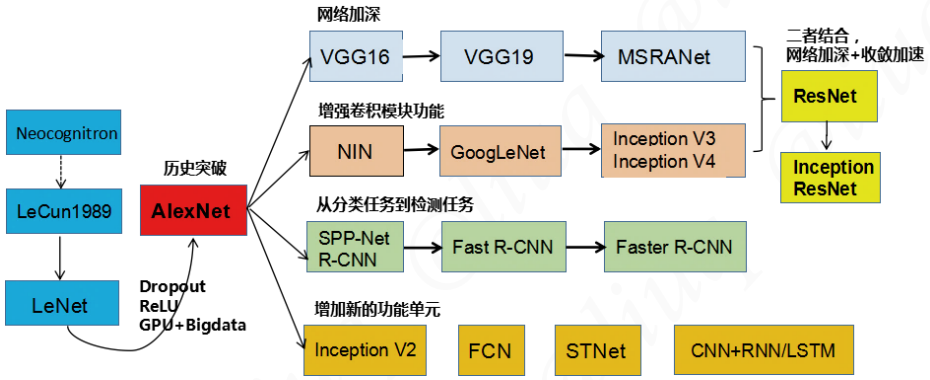


图 3-1 卷积网络简明演化历史

Figure 3-1 a Brief History of CNN

本设计使用卷积神经网络来提取图像中的特征，如图 3-1 是卷积神经网络的大致发展历程[25]：LeCun 在[26]首次提出 CNN 模型，并在神经网络中引入卷积层，Hinton 在[28]中设计出 AlexNet 在 ImageNet[29]图像分类比赛中实现突破，大大降低了图像分类的错误率。随后各种新的更好的卷积神经网络层出不穷，主要朝着 4 个方向演化[18]：1)网络加深；2)卷积层功能增强；3)从分类任务到检测任务；4)新的功能单元的出现和添加。

VGG 模型在[32]中首次被提出，并取得了 2014 年 ImageNet ILSVRC 竞赛物体检测任务的第一名和物体分类任务的第二名。VGG 有 5 个卷机组，2 层全连接图像特征，1 层全连接分类特征，其中带权重的有 16~19 层。由于其优异的性能，相对简单的结构所以本设计采用其和标准 RNN 一起来构建图像描述基准模型。

3.3模型和输入表示（Model and Inputs Representation）

3.3.1 模型构架

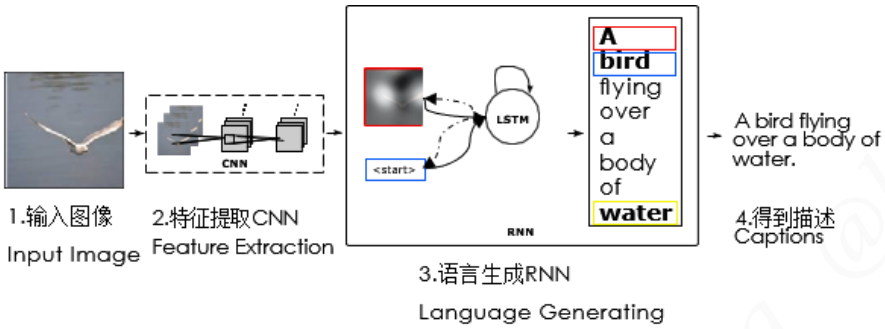


图 3-2 端到端的图像描述网络

Figure 3-2 end-to-end neural network

描述模型整体采用 CNN+RNN 的构架，端到端（end to end）从原始输入图像数据，最终产生对于图像的自然语言的描述，换句话说从原始图像数据到最终的高层语义，中间的映射不经过任何人为的干预[21]而只经过神经网络的处理如图 3-2。总体流程即 CNN 作为编码器（encoder）获取图像的特征（转换成固定长度的特征向量），输入到基于 RNN 的描述生成器或者叫语言模型，然后和嵌入的词汇在时间步进行训练，从而得到用以描述的模型和生成对应的描述。随后，LSTM 网络按照时间步进行前向和反向传播，从特殊的开始标记开始，每个时间步上对词汇进行采样，并得到相应时间步上候选词汇的概率分布，将概率最大的词汇取出作为下一个时间步的输入，依此类推，直到遇到特殊的结束标记，描述被生成。

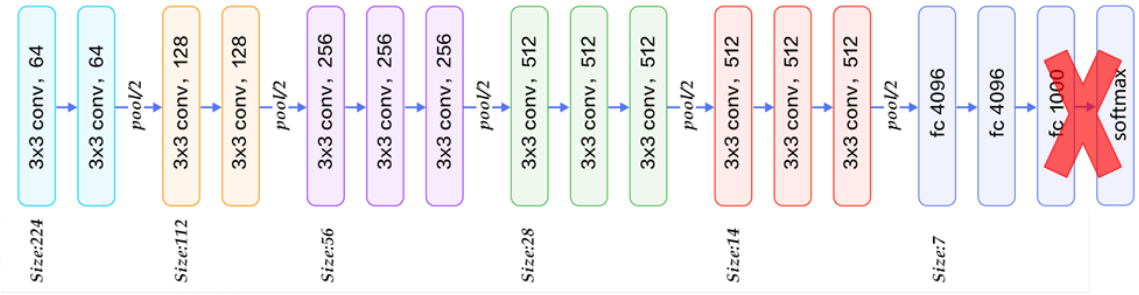


图 3-3 VGG16 网络结构和处理

Figure 3-3 Architecture of VGG16 and operation in this paper

本章中使用的 VGG16 预训练模型，将 VGG16 最后的全连接层和 Softmax 层去掉，从而将特（图中为 fc4096 层的输出）输入到后面的 RNN 中。

本设计中构造的处理后的 VGG16 模型详情如表 3-1，其中的 None 在训练中将为 batch 的大小所替换。

表 3-1 VGG16 详情参数

Table 3-1 Parameters of VGG16

层 Layer	类型 Type	输出尺寸 OutputShape	参数规模 Param #
input_1	(InputLayer)	(None, 224, 224, 3)	0
block1_conv1	(InputLayer)	(None, 224, 224, 64)	1792
block1_conv2	(Conv2D)	(None, 224, 224, 64)	36928
block1_pool	(MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1	(Conv2D)	(None, 112, 112, 128)	73856
block2_conv2	(Conv2D)	(None, 112, 112, 128)	147584
block2_pool	(MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1	(Conv2D)	(None, 56, 56, 256)	295168
block3_conv2	(Conv2D)	(None, 56, 56, 256)	590080
block3_conv3	(Conv2D)	(None, 56, 56, 256)	590080
block3_pool	(MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1	(Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2	(Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3	(Conv2D)	(None, 28, 28, 512)	2359808
block4_pool	(MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1	(Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2	(Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3	(Conv2D)	(None, 14, 14, 512)	2359808
block5_pool	(MaxPooling2D)	(None, 14, 14, 512)	0
flatten	(Flatten)	(None, 25088)	0
fc1	(Dense)	(None, 4096)	102764544
fc2	(Dense)	(None, 4096)	16781312
总参数规模 Total params:	134,260,544		
可训练参数 Trainable params:	134,260,544		
非可训练参数 Non-trainable	0		

3.3.2 图像表示

首先对原始图像进行预处理，包含对图像进行重塑（reshape）为以统一的规格，以及将图像各通道像素值范围映射到指定区间等。接着使用上述在

ImageNet 预训练后权重的 CNN，对预处理后的原始图像做进一步处理，处理处理流程如图 3-4 所示，从而提取到固定长度向量表示的图像特征编码：

$$x_{-1} = CNN(I) \tag{3.1}$$

其中I表示输入图像。图 3-9 末尾的箭头表示，得到的特征编码x_{-1}将作为初始化输入传递到后面的 RNN 语言模型当中。

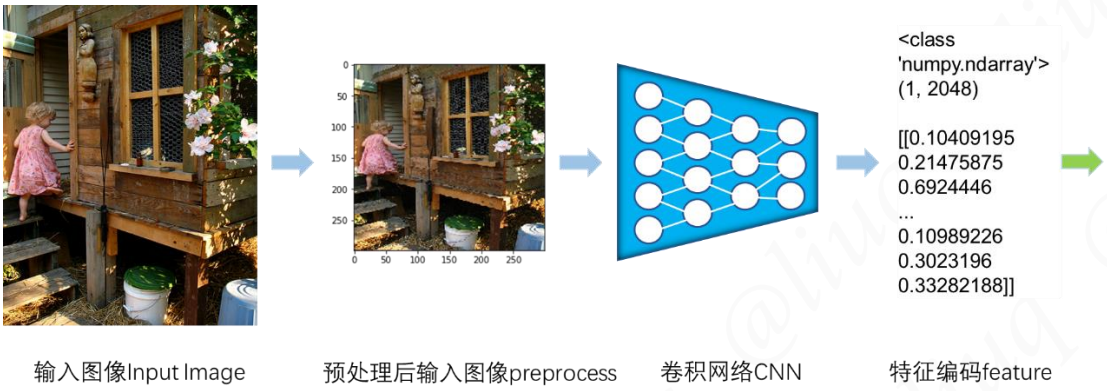


图 3-4 图像预处理

Figure 3-4 Preprocessing Image

3.3.3 构建词汇表和描述标记化

对于图像的描述可以视作由词汇（英文等符号化的语言为单词，中文为词语）组成的字符串，显然神经网络不能直接处理字符串，因此需要经过序列化处理为能够处理的序列数据。首先来构建整体词汇表，词汇表的构建和大小很灵活，也可以使用现有学术界和工业界的常用词汇表，理论上词汇表的规模越大，就越能够对现实世界进行反映，然而对应的模型计算成本也越巨大，原因见后文。同样加入词汇表中还有几个关键的标记（token），对于每个描述，在描述的首尾分别加入<start>和<end>，来指示描述的开始和结尾；对于描述中不在词汇表中的词，使用<unk>标记来替代；由于对于图像的描述长短不一，为了使 LSTM 处理向量方便，将所有描述处理成定长即对应的时间步长度相等，这个长度取数据集中最长的描述的长度，其余长度不够的在<end>标记后补上若干个<null>标记，在反向传播时，不计算该标记的梯度。给予词汇表中每个词汇（包含标记）一个唯一的整数索引值，这样就有了索引到词汇的映射即 index to word。

反过来，将描述中每个位置上的词汇映射为其在词汇表中对应的索引值，即 word to index。从而将一个描述转换成一个索引值序列，举例如表 4-2。从而为进一步的词汇嵌入（word embedding）做准备。

表 3-2 描述标记化示意图

Table 3-2 Illustration of Caption Tokenization

图像	描述和单词切分	映射为（定长）索引序列
 Flickr8k: 1000268201_693b08cb0e.jpg	'A child in a pink dress is climbing up a set of stairs in an entry way . ' <class 'str'> 72	['<start>', 'A', 'child', 'in', 'a', 'pink', 'dress', 'is', 'climbing', 'up', 'a', 'set', 'of', 'stairs', 'in', 'an', 'entry', 'way', '.', '<end>'] <class 'list'> 20 [1, 1512, 6662, 5512, 1512, 1460, 7510, 1109, 5284, 491, 1512, 7056, 5671, 3090, 5512, 5948, 1153, 470, 2467, 2] <class 'list'> 20
	['A', 'child', 'in', 'a', 'pink', 'dress', 'is', 'climbing', 'up', 'a', 'set', 'of', 'stairs', 'in', 'an', 'entry', 'way', '.'] <class 'list'> 18	[[1 1512 6662 5512 1512 1460 7510 1109 5284 491 1512 7056 5671 3090 5512 5948 1153 470 2467 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]] <class 'numpy.ndarray'> (1, 40) int32

3.3.4 词汇嵌入

前文中我们将描述转换成为了由索引值构成的序列，这些索引值只是特定表示词汇在词汇表中的具体位置，对于每个词汇还需要使用词汇嵌入处理成 LSTM 正向和反向传播过程中处理的向量，因此使用词汇嵌入。词汇嵌入，是一种将词汇映射成高维向量的处理方法，即$word \longrightarrow \mathbb{R}^n$。最自然和最简单的方式，是将每个词汇处理成一个长度和词汇表规模相同的 one hot（独热编码，或一位有效编码）向量，以表 3-3 为例，若词汇表大小为 7709，因此描述中的每个单词都转换成了一个和词汇表规模一致的 one hot 向量，只在对应索引位置上为 1 其余位均为 0 表示:

表 3-3 词汇嵌入示意图

Table 3-3 Illustration of Word Embedding

索引 Index	词汇 word	one hot 向量
		['<start>', 'A', 'child', 'in', 'a', 'pink', 'dress', 'is', 'climbing', 'up', 'a', 'set', 'of', 'stairs', 'in', 'an', 'entry', 'way', '.', '<end>']
		[1, 1512, 6662, 5512, 1512, 1460, 7510, 1109, 5284, 491, 1512, 7056, 5671, 3090, 5512, 5948, 1153, 470, 2467, 2]
0	<null>	[[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]]
1	<start>	[1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [2.]
2	<end>	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
470	way	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
491	up	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
1109	is	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
1153	entry	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
1460	pink	[0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
1512	a	[0.] [1.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
2467	.	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
3090	stairs	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
5284	climbing	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
5512	in	[0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
5671	of	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
5948	an	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
6662	child	[0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
7056	set	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	... ...
7510	dress	[0.] [0.] [0.] [0.] [0.] [0.] [1.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
...	...	[0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.] [0.]
7708	zippered	[0.]] [0.]]

直接处理代表词汇的 one hot 向量显然效率不高，对于描述序列$\mathrm{S}=(\mathrm{S}_0,\mathrm{S}_1,\ldots,\mathrm{S}_\mathrm{N})$，有：

$$x_t = W_e S_t, t \in 0,1 \ldots, N-1 \tag{3.2}$$

其中S_t即经过处理得到的词汇的 one hot 向量。W_e是词汇嵌入的权重矩阵，即将对应词汇的 one hot 向量映射成为另外维度和隐藏意义的向量表示，随着网络的训练进行改变和最终确定。自此词汇嵌入完成。

3.4模型训练（Model Training）

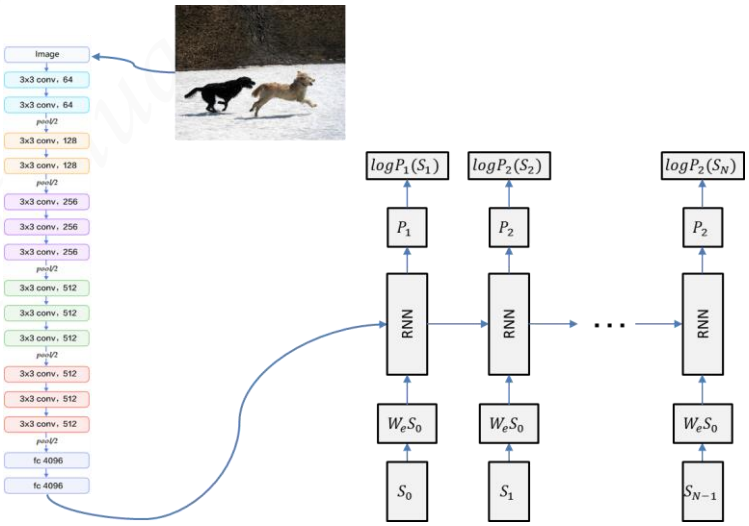


图 3-5 RNN 语言模型构架

Figure 3-5 Architecture of RNN Language Model

具体而言模型结构如，标准 RNN 在每个时间步上接收输入，并得到词汇表中所有词汇的概率分布：

$$p_{t+1} = RNN(x_t), t \in 0,1 \ldots, T-1 \tag{3.3}$$

给定一个标记化的序列$\mathrm{s}_1,\ldots,\mathrm{s}_\mathrm{T}$，传递给 RNN 的是长度为$\mathrm{T}+2$的词汇向量$x_{-1}, x_0, \ldots, x_T$，其中$x_{-1}$即式 3.1 中的结果，表示经过 CNN 处理得到的特征编码，x_0对应于一个特殊的<start>标记，x_t对应于标记化后的序列中的s_t，其中$\mathrm{t}=1,\ldots,\mathrm{T}$。RNN 计算一个隐藏状态$h_t$组成的序列，输出$p_t$组成的向量，计算方法如式 3.3 到 3.5 所示。输出向量p_t的大小为$|V|+1$，其中 V 表示标记化的词汇表，额外的这个 1 表示一个特殊的<end>标记，若我们每次选择p_t概率分布最大的那个，则在$\mathrm{t}=0,\ldots,\mathrm{T}-1$上的目标输出是$s_{t+1}$，在$\mathrm{t}=\mathrm{T}$上的目标输出是<END>标记。向量$p_{-1}$是缺省的。

$$h_t = tanh W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \tag{3.4}$$

$$p_{t+1} = Softmax\left(h_t\right) \tag{3.5}$$

对应于实现上，需要提供有如下方法接口：

在单个时间步上的前向传播: 计算在单个时间步上隐藏状态值；整个时间步上标准 RNN 处理单元的前向传播: 标准 RNN 完成在所有时间步上的前向传播，返回得到整个时间步上的所有隐藏状态；在单个时间步上标准 RNN 处理单元的反向传播：计算在单个时间步上损失函数关于输入，RNN 隐藏状态，权重矩阵等的梯度；整个时间步上标准 RNN 处理单元的反向传播：计算在整个时间步上损失函数关于输入，RNN 隐藏状态，权重矩阵等的梯度；整个时间步上的词汇嵌入前向传播和整个时间步上的词汇嵌入反向传播 ：由于词汇经过了标记化，表示为整形索引值，所以不能反向反向传播到单词那一级。所以计算和返回损失函数对于词汇嵌入矩阵的梯度；临时的仿射变换层：用以将 RNN 隐藏状态向量转换为每个时间步上词汇表中相应词汇的得分；临时的 Softmax 损失函数：在每个时间步上得到词汇表中每个单词的得分，并且已知每个时间步的 ground truth 词汇，因此需要使用一个临时的 softmax 损失函数来计算每个时间步上的 loss 和梯度。然后将整个时间上的 softmax loss 值求和，并计算在整个 minibatch 上的平均值。

3.5 实验（Experiments）

3.4.1 数据集

MS COCO（Microsoft Common Objection Context）数据集市一个大规模对象识别，语义分割和描述的数据集。本章节使用的数据集中包含 82783 张训练图像，40504 张验证图像，40775 张测试图像。对应于 400135 段训练图像的描述数据，195954 段验证图像的描述，203800 段测试集描述数据。其中最长的描述长度为 17。

3.4.2 实验设置

本章中完成了基于原生 Python 和 NumPy 关于标准 RNN 描述模型的构建。

处理器：Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz (8 CPUs), ~2.8GHz。

实验平台：Windows 10 专业版 64-bit (10.0, Build 16299)。8192MB RAM。

开发工具和语言：Anaconda, Jupyter notebook, PyCharm，Python（基于 NumPy）。

超参数设置：学习率：5×10^{-3}；学习衰减率：0.95；词汇表大小：1004。最长描述长度：17。对于小样本训练：训练样本大小 100，批大小：50；RNN 隐藏状态维数：512；单词嵌入维数：256。对于大样本训练：训练样本大小 10000，批大小 50，RNN 隐藏状态维数：512；单词嵌入维数：256。

3.6 实验结果（Results）

3.6.1 基于小样本训练的结果

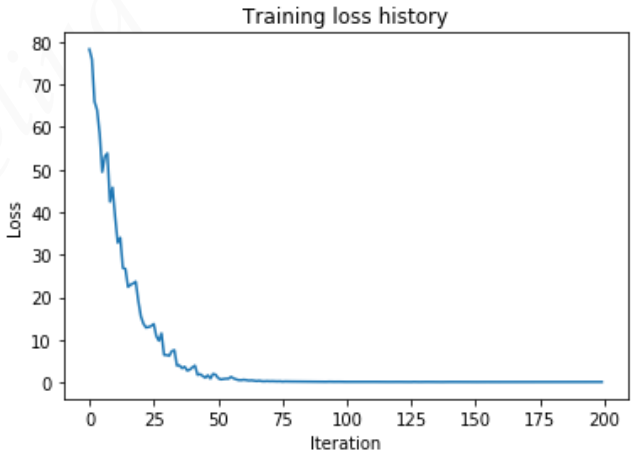


图 3-6 基于小样本的训练损失变化

Figure 3-6 Training loss history Based on small sample

由于数据集总体规模较大，为了验证模型和算法正确性，我们先在小样本上（从数据集中随机抽取 100 张图像和对应的 100 段描述，一张图像只取其中的一段 ground truth 描述）进行训练。如图 3-7 所示为采用章节 3.4.2 实验设置，训练 50 轮总共迭代 200 次后的实验结果。可以看出，在样本比较小的情况下，训练的损失函数随着迭代次数的增加在不断降低直至为 0 并保持不变，其中前 1/8 的迭代周期损失值下降尤为明显。证明了模型的有效性，即模型在小样本的条件下具有良好的“学习能力”。

图 3-7 基于小样本的训练和验证集描述结果

Figure 3-7 Captions of train and validation images Based on small sample

但同时，模型出现了过拟合，在训练集上正确率为 100%。但在验证集上验证模型，并和训练样本对比，部分结果如图 3-7。很明显，训练样本上的模型生成了很好的结果，生成的描述和 ground truth 几乎完全一致。而在验证数据上的描述则明显效果较于前者差得多。

3.6.2 基于大样本的训练结果

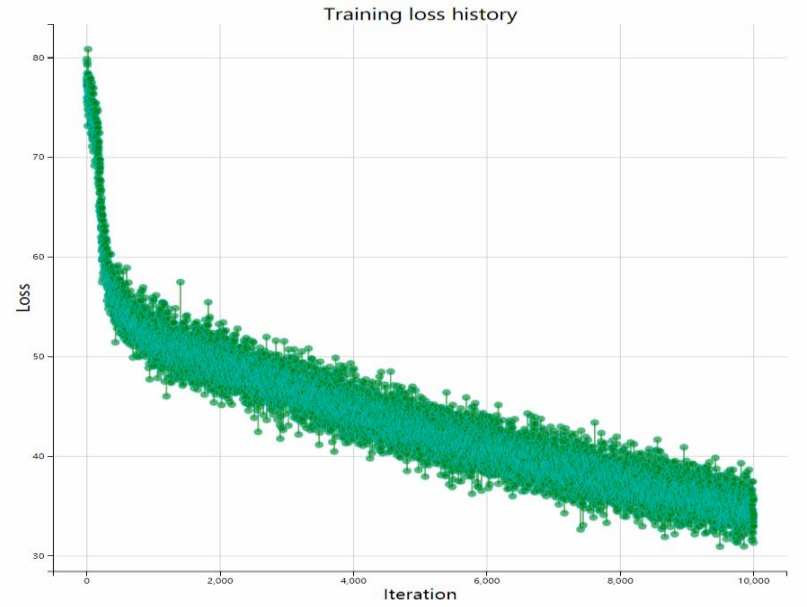
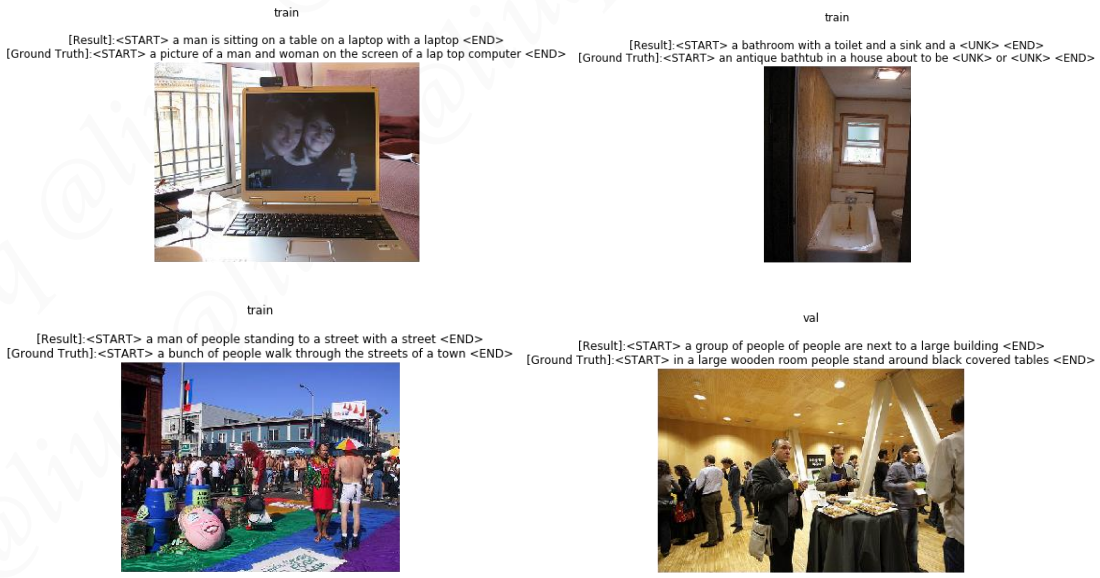


图 3-8 基于大样本的训练损失变化

Figure 3-8 Training loss history Based on large sample

如图 3-8 所示为采用章节 3.4.2 实验设置中的大样本实验的设置训练 50 轮总共迭代 10000 次后的实验结果。和小样本上训练的结果相比，共同点在于损失随着迭代次数和训练轮数的增加呈现下降的趋势；区别在于：由于训练样本的规模较大，所以整个过程中的损失值范围都比下样本上的高得多；另外，大样本的振荡情况比小样本下明显得多，相邻迭代次数乃至有的相隔数百次迭代之间损失值并非后来者总是低于前面的损失值。



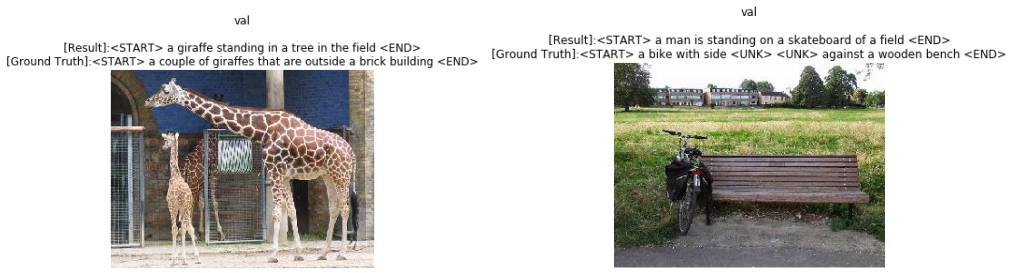


图 3-9 基于大样本的训练和验证图像描述结果

Figure 3-9 Captions of train and validation images Based on large sample

对训练样本进行随机抽样检查，训练样本的和结果没有出现在小样本情况下的过拟合情况。结合损失值的变化趋势以及在训练集和验证集上的抽样结果，表明基于标准 RNN 和 VGG 的描述模型能够接受输入图像端到端地生成与图像内容相关的自然语言的描述，且在验证集上的描述结果表明模型在大样本的情况下，同时具有一定的泛化能力。

3.7本章小结（Summary）

本章基于标准 RNN 和 VGG16，完成了从模型架构设计，输入图像信息表示，词汇表构建、词汇嵌入、描述标记化以及描述语言生成模型的构建。并使用 MS COCO 数据集，非别在大样本的和小样本的情况下， 对模型进行了训练以及在验证集上进行了杨峥。小样本下训练得到了过拟合的模型，在大样本下获得了具有一定泛化能力的模型，结果显示模型无论对于训练数据还是验证数据均能产生能够被读懂的自然语言的描述，证明了模型的正确性。作为设计的基准模型，为之后的方法研究和模型间对比试验奠定了一定基础。

4 基于 LSTM 和 VGG、GoogLeNet、ResNet 的图像描述模型
4 Image Captioning Models Based on LSTM and pluggable VGG, GoogLeNet, ResNet

4.1 引言（Introduction）

前一章基于标准 RNN 和 VGG 网络构建了端到端的图像描述模型并能够对图像进行相关的描述。本章将以 LSTM 作为描述生成的核心组件，结合除了包括 VGG 在内的 GoogLeNet、ResNet 图像特征提取模型，基于深度学习框架，构建新的端到端的图像描述模型，并介绍部分训练细节。对于模型测试，将原始图像对应描述构建为描述的参考包（bag），将生成的描述构建为假设包，使用机器翻译中的 BLEU 评价指标对描述生成模型整体性能进行了定量的测评，此外还测试对比了模型间的运行性能，并将模型推广到开放式的图像生成任务之中，通过对生成结果的抽样对描述生成结果给出了定性的评价。

4.2 相关工作（Related Work）

4.2.1 GoogLeNet

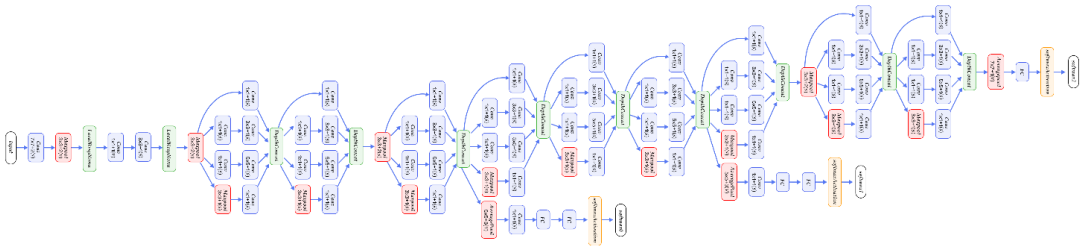


图 4-1 GoogLeNet Inception-V1

Figure 4-1 GoogLeNet Inception-V1

GoogLeNet 是 ImageNet ILSVRC 物体分类任务的冠军，图 3-5 是 GoogLeNet 的第一版（GoogLeNet-InceptionV1）,共有 22 层：最开始由三层普通卷积构成，接下来由三组子网络组成，子网络为 Inception（嵌套式的卷积结构）模块的叠加。如图 4-2 为 Inception 模块的原始设计和改进，从而如前文所讲增强了卷积模块的功能，可以提取高度非线性特征同时减少参数[33]。

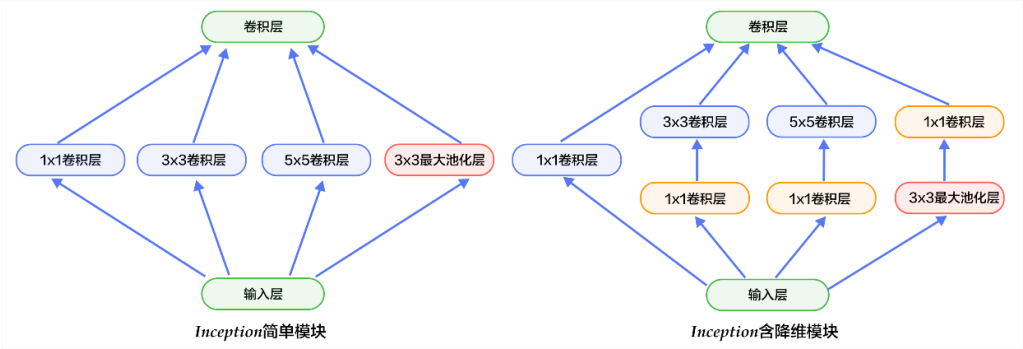


图 4-2 Inception 模块原始版本和改进

Figure 4-2 Inception Module and its Improved Version

InceptionV2[34]引入了 BN（batch normalization，批量正则化）层，Inception V3[36]对卷积层做了分解，进一步提高了网络的非线性能力且进一步加深了网络，Inception V4[37]引入了后文的 ResNet 设计思路，从 V1~V4 的每次改进都带来了准确度的提升。本设计中选用 GoogLeNet Inception V3，各层详细信息见附录 B.1，也作为预训练网络之一，处理方法同上文的 VGG。

4.2.2 ResNet

ResNet（Residual Network，残差神经网络）[35]在 2015 年获得了 ImageNet 图像分类、图像物体定位和图像物体检测比赛的冠军。针对训练卷积网络随着网络的加深导致精确度下降（即“退化”）的问题，ResNet 提出了残差学习的概念，和传统直连的卷积网络相比，ResNet 有很多旁路的支线将输入直接连接到后面的层，使得后面的层可以直接学习残差（目标输出减去输入），在已有设计（BN，小卷积核，全卷积网络）的基础上，引入了残差学习单元，如图 4-3 所示，有两层的学习单元和三层的学习单元（瓶颈 Bottleneck 模式连接，先降维再升维）。

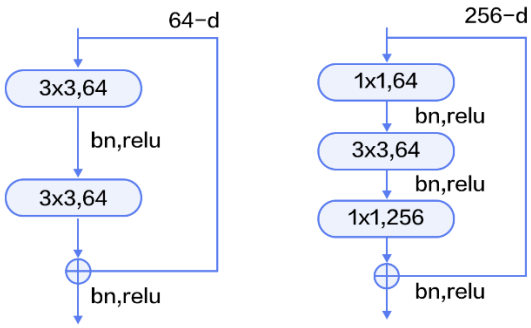


图 4-3 残差模块

Figure 4-3 Residual Modules

图 3-8 为 50、101、152 层 ResNet 网络示意图，区别在于每组中残差学习单元重复的次数。本设计采用的是其中的 50 层结构，各层详细信息见附录 B.2。

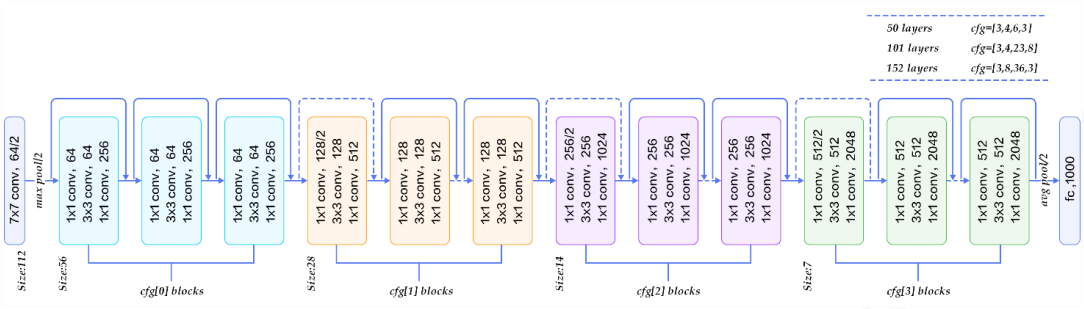


图 4-4 基于 ImageNet 的 ResNet 模型

Figure 4-4 ResNets Based on ImageNet

4.3模型（Model）

4.3.1 LSTM 语言模型

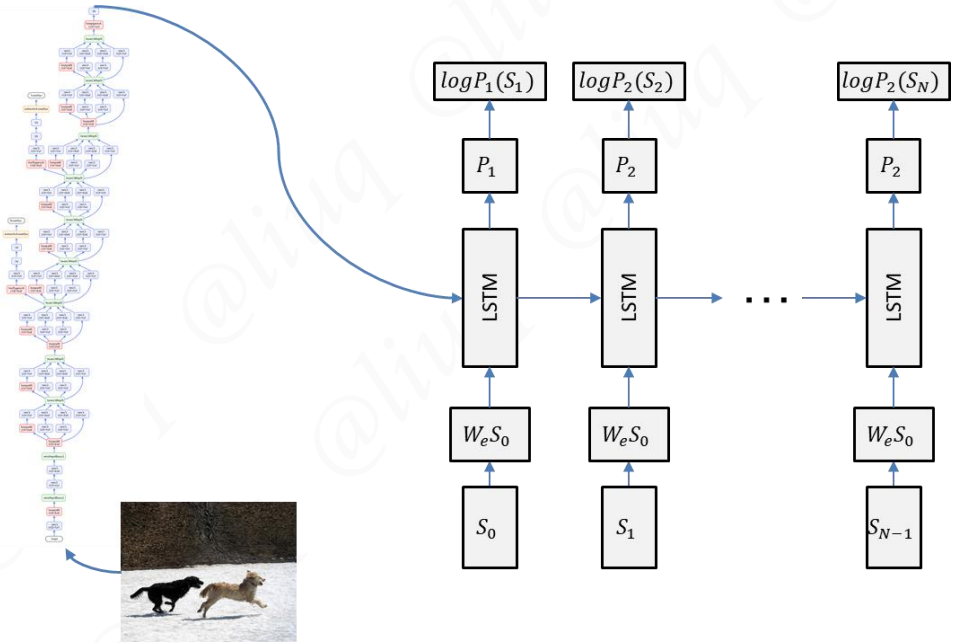


图 4-5 LSTM 语言模型构架

Figure 4-5 Architecture of the LSTM Language Model

用来对图像编码的 CNN 是在 ImageNet 图像分类比赛中预训练的 GoogLeNet Inception V3 去掉最后的部分隐藏层和输出层，进而将原本为隐藏层的全连接层即图像的特征，来初始化 LSTM 网络。随后，LSTM 网络按照时间步进行前向和反向传播，从特殊的开始标记开始，每个时间步上对词汇进行采样，并得到相

应时间步上候选词汇的概率分布，将概率最大的词汇取出作为下一个时间步的输入，依此类推，直到遇到特殊的结束标记，描述被生成。

具体而言，LSTM 在每个时间步上接收输入，并得到词汇表中所有词汇的概率分布：

$$p_{t+1} = LSTM(x_t), t \in 0,1 \ldots, T-1 \tag{4.1}$$

LSTM 处理过程如图 4-6 所示：

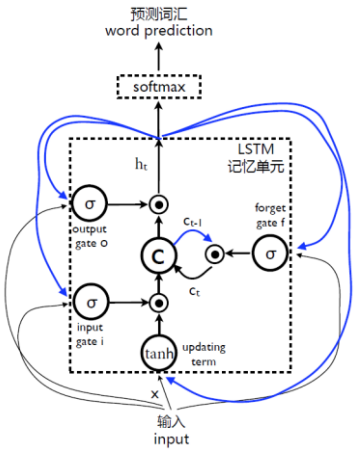


图 4-6 LSTM

Figure 4-6 LSTM

给定一个标记化（见章节 4.2）的序列$S_1, \ldots, S_T$，传递给 RNN 的是长度为$T+2$的词汇向量$x_{-1}, x_0, \ldots, x_T$，其中x_{-1}即式 3.1 中的结果，表示经过 CNN 处理得到的特征编码，x_0对应于一个特殊的<START>标记，x_t对应于标记化后的序列中的s_t，其中$t = 1, \ldots, T$。LSTM 计算一个隐藏状态h_t组成的序列，输出p_t组成的向量，计算方法如式 4.2 到 4.5 所示。输出向量p_t的大小为$|V|+1$，其中 V 表示标记化的词汇表，额外的这个 1 表示一个特殊的<END>标记，若我们每次选择p_t概率分布最大的那个，则在$t = 0, \ldots, T-1$上的目标输出是s_{t+1}，在$t = T$上的目标输出是<END>标记。向量p_{-1}是缺省的。对应的，形式化表示为：

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ g_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} \begin{bmatrix} W_{ix} & W_{ih} \\ W_{fx} & W_{fh} \\ W_{ox} & W_{oh} \end{bmatrix} \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} \tag{4.2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{4.3}$$

$$h_t = o_t \odot tanh(c_t) \tag{4.4}$$

$$p_{t+1} = Softmax(h_t) \tag{4.5}$$

以接受 InceptionV3 处理图像后得到的 2048 维特征为例，其经过仿射变换，转换为 512 维的图像嵌入特征向量，来初始化如前文所述 LSTM 中的隐藏状态（hidden state）h和细胞状态（cell state）c，其维度为图中所示为 512 维。比如在 Flickr8k 数据集上，则图中所示描述最大长度为 40，用于将所有描述处理为定长序列，如前文所述不够长度的进行填充。词汇嵌入向量同样为 512 维，原因见章节 4.2。去掉预训练模型后的完整的生成器（或者称为语言模型）结构如图 4-7 所示：

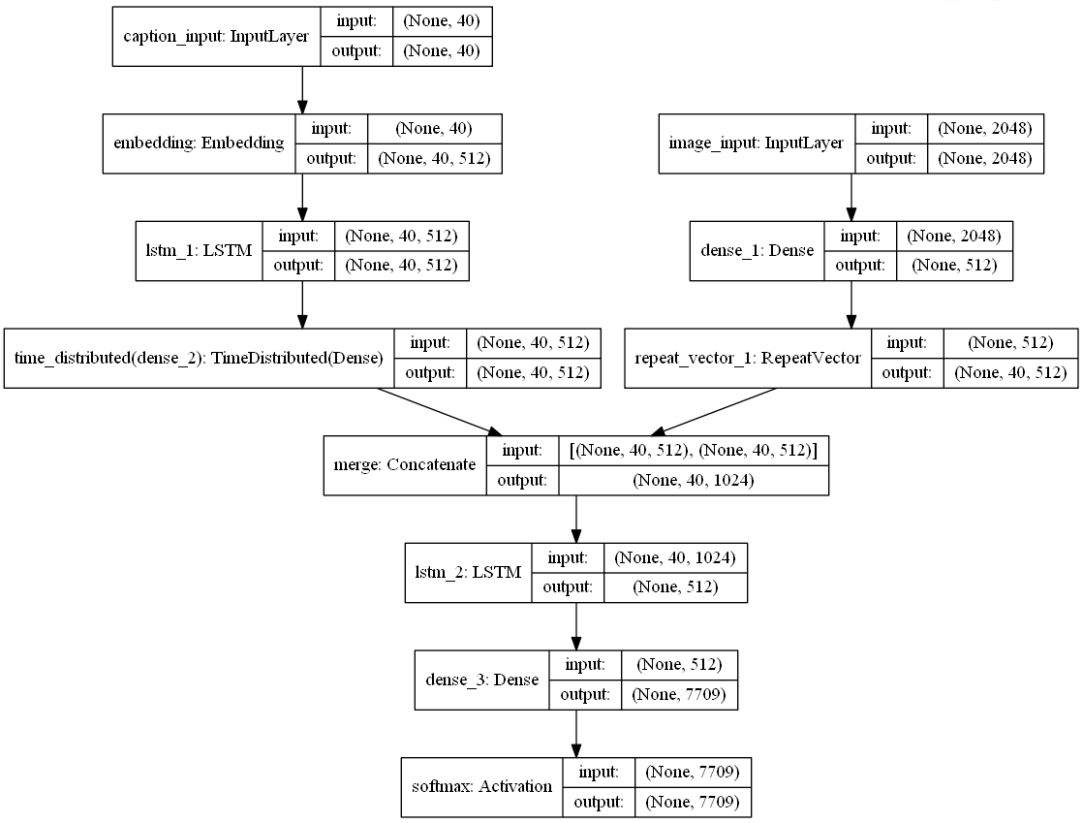


图 4-7 描述生成器（语言模型）网络结构

Figure 4-7 Architecture of the Caption Generator (Language Model)

模型详情如表 3-1, 其中的 None 在训练中将为 batch 的大小所替换。

表 4-1 描述生成器（语言模型）各层参数详情

Table 4-1 Parameters of the Caption Generator (Language Model)

层 Layer	类型 Type	输出尺寸 OutputShape	参数规模 Param #	相连于 Connected to
caption_input	(InputLayer)	(None, 40)	0	
image_input	(InputLayer)	(None, 2048)	0	CNN 的 fc 层
embedding	(Embedding)	(None, 40, 512)	3947008	caption_input[0][0]
dense_1	(Dense)	(None, 512)	1049088	image_input[0][0]
lstm_1	(LSTM)	(None, 40, 512)	2099200	embedding[0][0]
repeat_vector_1	(RepeatVector)	(None, 40, 512)	0	dense_1[0][0]
time_distributed	(TimeDistributed)	(None, 40, 512)	262656	lstm_1[0][0]
merge	(Concatenate)	(None, 40, 1024)	0	repeat_vector_1[0][0] time_distributed[0][0]
lstm_2	(LSTM)	(None, 512)	3147776	merge[0][0]
dense_3	(Dense)	(None, 7709)	3954717	lstm_2[0][0]
softmax	(Activation)	(None, 7709)	0	dense3[0][0]
总参数规模 Total params:	14,460,445			
可训练参数 Trainable params:	14,460,445			
非可训练参数 Non-trainable params:	0			

4.3.2 损失函数

这里损失函数为整个时间步上的每个时间步对应正确词汇的负对数似然（negtive log likelihood，即上文中所讲已知p_t前提下，对应时间步的正确词汇的条件概率取值）进行求和，即:

$$L(I,S) = -\sum_{t=1}^{N} log\, p_t(S_t) \tag{4.6}$$

用前文所述的基于批处理的 SGD 算法来最小化模型的损失函数。

4.4 实验（Experiments）

4.4.1 数据集

本章节使用的是 Flick8k，Flickr30k 和微软的 COCO 数据集，如表 4-1 所示，除了数据规模和图像类型、规格等有区别之外，其共同之处在于每张图片都提供 5 段人工的描述，可以满足本设计的需要。

表 4-2 数据集

Table 4-2 Datesets

数据集名称	规模		
	训练集 train	验证集 val	测试集 test
Flickr8k	**6000**	**1000**	**1000**
Flickr30k	**28000**	**1000**	**1000**
MS COCO	**82783**	**40504**	**40775**

4.4.2 实验设置

处理器：Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz (8 CPUs), ~2.8GHz。NVIDIA GeForce GTX 1050。

实验平台：Windows 10 专业版 64-bit (10.0, Build 16299)。8192MB RAM。

开发工具和语言：Anaconda, Jupyter notebook, PyCharm/PyCharm，Python（基于深度学习框架 Keras 和 TensorFlow 后端）。

超参数设置：学习率：1×10^{-2}；学习衰减率：0.0（注，这里根据[1]的研究结果：设置固定的学习率并且学习衰减率为 0 的情况下有利于模型的训练，来设置）；词汇表大小：7709；批大小：256；LSTM 隐藏状态维数：512；单词嵌入维数：512。最长描述长度（最长时间步）：40。

4.4.3 模型训练细节

由前文可知，由于问题和模型结构的复杂性，数据集的庞大，计算量和需要训练得到的参数规模非常庞大。本章基于 Keras（见附录 A）结合 Tensor 后端的模型实现调用 GPU（graphics processing unit，图像处理器）结合 CUDA 和 cuDNN 来加速网络训练，这也是深度学习中常用的训练策略之一。虽然有 GPU 加速计算，若每次都在全部的数据上进行迭代，时间成本同样很巨大（以小

时、天乃至月为计量），所以采用递进的方式，先在全部数据中的一部分上进行训练，验证模型的正确性；之后再在足够大以及全部数据集上训练和调试。

前一章中模型损失函数总体下降的趋势和描述结果的产生证明了模型和算法的正确性，由于深度学习的强大，模型在训练中（特别是小数据集上）的正确率很容易就能达到甚至接近 100%，同时过拟合的现象几乎是无法避免的，如图 4-8 所示是某次训练最终得到的损失函数曲线，从较大值直至减小接近于 0，表明在训练集上模型的效果很好。通过采样查看生成的描述，发现训练集图像，生成的描述大多数都和真实的描述即 ground truth 一致，证明模型能够经过训练产生自然语言的描述；而验证集上生成的描述，效果却很差，很多都和图像内容不一致，由此可以很直观地看出模型的过拟合。因此，训练集和验证集上得到的损失值和正确率，对于衡量模型的好坏只能起到参考作用，更重要的是在测试集上的结果和由此反映的模型的泛化能力。

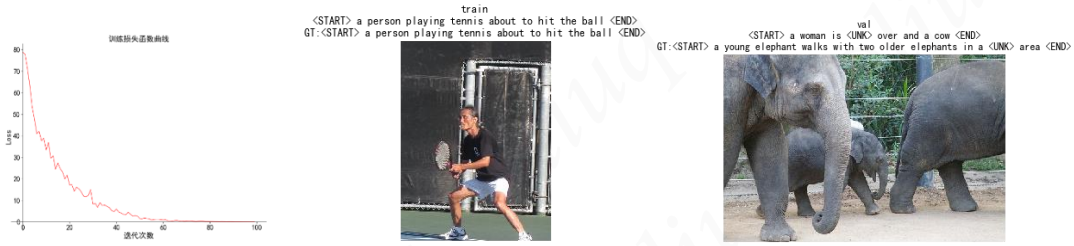


图 4-8 过拟合训练损失曲线和描述结果

Figure 4-8 Train loss history and captions of over-fitting model

在模型训练的细节上，对于 CNN 而言，没有采用通常的随机初始化参数的方式，而是使用在 ImageNet 上训练得到最佳权重参数并在时候的描述生成中保持不变，而其余所有权重参数均随机初始化。还根据研究[1]的成果，采用了 512 维的词汇嵌入和隐藏维度设置。对于正则化来防止过拟合尝试了[37]的 dropout 方法。

另外的情况就是与过拟合相反的“欠拟合”，如图 4-9 所示为训练中某次欠拟合的模型：

图 4-9 欠拟合训练损失和描述结果

Figure 4-9 Train loss history and captions of under-fitting model

表现为，无论是训练集还是验证集上正确率和取得预测的结果和真实值均有较大差异。因此需要在大的数据上过多轮的训练，并保证所有词汇在训练中出现不少于一定次数。

4.5 实验结果（Results）

4.5.1 训练过程结果

如图 4-10 是其中使用 InceptionV3+LSTM 模型在训练集上迭代 50 轮的训练曲线：

图 4-10 基于 LSTM 和 InceptinV3 模型的训练曲线

Figure 4-10 Training history of LSTM+InceptionV3 model

可以直观地看到训练过程中损失函数值 Loss 在局部有震荡，但总体呈现下降趋势，而生成的描述和训练的描述完全一样的比率即正确率 Acc 在稳步提升。通过对训练的描述进行抽样如表 4-3 所示，直觉上看，生成的描述，从一开始的非常糟糕，慢慢变得和图片中的主要对象相关或者接近，在某些局部位置会有语法上的错误，再然后语法上的错误几乎消失,描述的质量并非持续上升。

表 4-3 损失和描述随训练轮数变化示例

Table 4-3 Illustration of loss history and captioning performance of a image

Ground Truth:
a black dog is running after a white dog in the snow.
black dog chasing brown dog through snow
two dogs chase each other across the snowy ground.
two dogs play together in the snow.
two dogs running through a low lying body of water.

轮（epoch）	Loss（损失函数）	Caption（描述生成）
1	4.24642	a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a
4	3.3340	a black dog and a black dog play with a ball in its mouth.
6	3.3160	a black dog and a black dog are running in the snow.
8	3.1621	a black and white dog is playing with a tennis ball in the snow.
12	3.1406	a black dog and a black dog are in the snow.
24	3.2616	two dogs in the snow.
50	3.2717	a black dog is running through the grass with a toy in its mouth.

通过更多轮的迭代提高了在训练集上的精确度，减小损失值。但最重要的是在测试集上测试才能更好地评价模型的描述能力的好坏。

4.5.2 测试集描述生成结果

训练好模型之后，对于测试图像的描述生成，第一种是基于采样的方法如图 4-11。测试图像输入预训练 CNN 模型得到特征来初始化 LSTM 网络，若沿用之前模型在训练和验证时的描述生成方法，从<start>标记开始，产生的词汇表上的概率分布，采样条件概率最大的词汇作为预测的第一个词汇 y0，然后将

其和隐藏状态一起作为下一个时间步上的输入，从而产生第二个预测词汇 y1，依此类推，直到预测到<end>标记或者预测达到最大描述长度。由此即完成对测试图像生成描述。

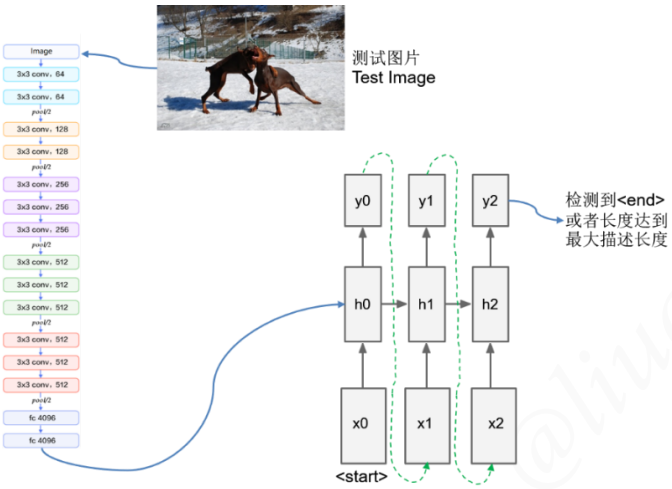


图 4-11 描述生成示意图

Figure 4-11 Illustration of image captioning

这种方法很直观地体现了贪婪（greedy，或称贪心）的策略，但是明显的问题是，我们希望能够得到能使产生描述序列整体条件概率最大，而上述基于采样的方法通常情况下不能很好地做到这点。

第二种方法是 beam search（集束搜索）的方法，定义 beam width（集束宽）为k，即在在任意时间步t，我们使用已经生成的长度为t的句子来生成新的长度为$t+1$的句子，每次只取所有组合中在条件概率分布最大的前k个，直到出现<end>结束标志或者长度达到最大描述长度。从最终的k个描述集合中选择条件概率最大作为最终描述，即：

$$S = \underset{S'}{argmax}\, p(S'|I) \tag{4.7}$$

4.5.3 评价指标结果

但是对于模型整体性能的描述，使用人工打分的方式成本较高。为了使用单一值来评价模型整体的描述好坏，本设计中使用了机器翻译中的 BLEU（Bilingual evaluation understudy，双语评估替补）[45]作为评价指标，精度采用修正 n-单位精确度（modified n-gram recession），其中的 n-gram 指一个语句中连续 n 个词组成的片段。

将测试集上全部生成的描述组成的语料库作为待评价（candidate）设为C，图像的 ground truth 组成的语料库作为标准参考（reference）设为S。

对于一个生成的描述C_i，与之对应的 ground truth 组成了一组参考描述$\mathrm{S_i} = \{s_{i1}, s_{i2}, \ldots, s_{im}\} \in S$，令 n-grams 表示 n 个词汇组成的词组集合，w_k为第 k 组可能的 n-gram，$\mathrm{Count_k}$表示w_k在待评价描述中出现的次数，同理$\mathrm{Count_k(s_{ij})}$表示w_k在参考描述中出现的次数，因此整个语料库层面上的重合精度为：

$$CP_n(C,S) = \frac{\sum_i \sum_k min\left(Count_k(c_i), max_{j \in m} Count_k(s_{ij})\right)}{\sum_i \sum_k Count_k(c_i)} \tag{4.8}$$

进一步有：

$$BLEU_N(C,S) = BP * exp\left(\sum_{n=1}^{N} w_n log\left(CP_n\right)\right) \tag{4.9}$$

其中 BP（Brevity Penalty）是用于对生成描述长度比较短时对于式（4.8）影响较大的惩罚项[45]，

$$BP = \begin{cases} 1 & l_s < l_c \\ e^{1-\frac{l_s}{l_c}} & l_s \geq l_c \end{cases} \tag{4.10}$$

l_c表示C_i的长度，l_s表示S_i中长度和l_c最接近的，其中 N 取 1，2，3，4，常数w_n取$\frac{1}{N}$。

在 Flick8k 上的训练得到的模型评价结果如下：

表 4-4 在 Flickr8k 的评价结果

Table 4-4 Results of metrics on the Flickr8k

组	模型/方法	BLEU-1	BLEU-2	BLEU-3	BLEU-4
实验组	VGG+LSTM	0.2725	0.1539	0.0860	0.0458
	Inception V3+LSTM	0.4812	0.2949	0.1789	0.1060
	ResNet+LSTM	0.5075	0.3000	0.1757	0.0986
	综合最好	0.5075	0.3000	0.1789	0.1060
对照组	Tri5Sem[42]	0.48	-	-	-
	m-RNN[43]	0.5778	0.2751	0.2307	-
	MNLM[44][1]	0.51	-	-	-
	当前最好[1]	0.63	-	-	0.277
	人类水平	0.70	-	-	0.217

其中 BLEU 的数值范围为 0 到 1 之间，越接近于 1 表明生成的描述和原本的描述的契合度越高，模型的效果越好。在 BLEU 来看，实验组模型总体上达到了中等的水平，在部分指标上超过了部分现有模型的效果，例如在 BLEU-1 上超过和接近了对照组研究中 Tri5Sem[42]和 m-RNN[43]方法得到的结果，在 BLEU-2 上超过了 m-RNN[43]。实验组模型横向比较，总体差距不是很大，但是在部分参数上，例如实验组 2 和 3 的结果，虽然在 BLEU-1 和 BLEU-2 上，3

均超过了 2，但 BLEU-3 和 BLEU-4，2 反而高过了 3，说明 BLEU 评价指标得分 n-gram 取值之间，并非当模型在 n-gram 取值较小时的得分较高者，在 n-gram 增大时得分依然较高。但所有模型的 BLEU 得分均与 n-gram 呈正相关，原因在于 n-gram 越大时，对应的对句子的判定就越需要注重前后词汇的关联，这也是和 BLEU 原理相契合的地方。此外，实验组模型距离当前最好水平以及人类平均水平还有较大差距，说明模型还有较大改进的空间。

运行时间效率对比结果：

表 4-5 在 Flickr8r 模型时间效率评价结果

Table 4-5 Runtime performance of models on the flickr8k

时间效率	**VGG16+LSTM**	**InceptionV3+LSTM**	**ResNet50+LSTM**
平均提取一张图像特征用时	41.8ms	46.1ms	37.4ms
平均训练一轮用时（不包含特征提取）	981s	960s	976s
平均描述一张图像时间（不包括提取特征，一张图生成五段描述）	56.5s	56.2s	55.4s
平均对一张图像生成一段描述时间（不包括提取特征	11.4s	11.2s	11.1s

对于从图像中提取特征，模型间的效率相差无几，并且都是毫秒级别的。整个描述模型无论是训练还是测试时间效率的主要影响因素都在后面的语言模型上。都采用 LSTM 作为语言模型的情况下，之所以训练时间结果相差较大，可能的原因在于提取的特征维度不同，比如 VGG16 的是图像 4096 维的特征，而 InceptionV3 和 ResNet50 提取的是 2048 维的特征。还有可能的原因在于 CPU 和 GPU 的状态，测试平台的状态同样对模型的时间效率有影响，所以应当尽量保证在 CPU 和 GPU 没有其他多余的任务的情况下来测试和对比分析性能。而对于图像生成描述而言，都是属于秒的量级，彼此之间的性能差异不算很大，要提高图像描述模型的运算时间效率，可以通过提升实验平台运算性能，例如采用多 GPU 并行计算；或者缩减图像特征量来减小运算量，例如将 4096 维特征缩减为 1024 维甚至更小；以及对网络层级和隐藏状态层，词汇嵌入层等参数进行调整，这些都是理论上可行的方案。

4.5.4 定性结果

对于生成的描述，虽然我们可以通过抽样查看其语句是否和定义图像的内容一致，对于单个句子而言，也可以通过比较生成的描述和其 ground truth 之间

的相似度，例如余弦相似度来衡量两者之间的相关程度，但是对于一幅图像生成的描述，可以称之为描述得很好的句子绝对不只有一种，也就是没有标准答案。

由于对于此类任务，人类具有天然的优势，采取人工来对生成的描述进行打分。是可取的，通过部分抽样来人工打分，由此在测试集上生成的描述大体上分成以下几个等级：

1) 描述和图像内容几乎一致，且没有语法语义上的错误——优；

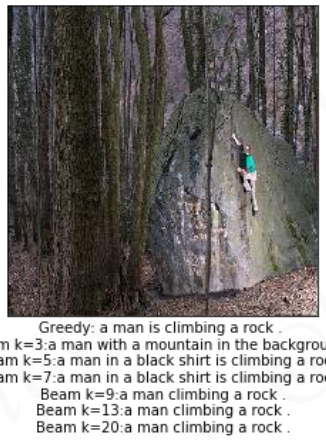


图 4-12 描述为优

Figure 4-12 Outstanding captioning

2) 描述和图像主要内容大体一致，只在语法语义上有一些错误——良；

图 4-13 描述为良

Figure 4-13 Exceeds expectations captioning

3) 描述和图像主要有明显不相符之处，语法语义上错误较少——及格；

Greedy: a man in a black shirt and black shorts is standing on a rock 's face .
Beam k=3: a man in a black shirt is standing on a mountain .
Beam k=5: a man and a woman sit on a rock .
Beam k=7: a group of people sit on a rock .
Beam k=9: a group of people sit on a rock .
Beam k=13: a group of people sit on a rock .
Beam k=20: a crowd of people playing a game .

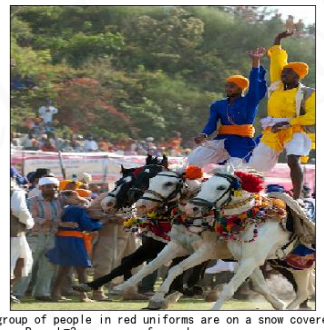
Greedy: a man in a red shirt and black helmet riding a bike .
Beam k=3: a man in a red shirt and black helmet riding a bike .
Beam k=5: a man in a red jacket and black helmet in a race .
Beam k=7: a man and a woman in a helmet are riding a dirt bike .
Beam k=9: a man does a jump in a race .
Beam k=13: a man doing a race .
Beam k=20: a man doing a race .

图 4-14 描述为及格

Figure 4-14 Acceptable captioning

4) 描述和图像内容有很多不符之处，但在语义语法上错误较少或者没有；或语法语义错误较多，内容略微相关——差；

Greedy: a man in a black jacket is in the snow
Beam k=3: a group of people standing on a snow covered mountain .
Beam k=5: a group of people on a snowy mountain .
Beam k=7: a group of people on a snowy mountain .
Beam k=9: a group of people on a snowy mountain .
Beam k=13: a group of people on a snowy mountain .
Beam k=20: a group of people .



Greedy: a group of people in red uniforms are on a snow covered field .
Beam k=3: a group of people are on a race .
Beam k=5: a group of people on a dirt track .
Beam k=7: a group of greyhound dogs in a race .
Beam k=9: a group of greyhound dogs in a race .
Beam k=13: a group of racing dogs in a race .
Beam k=20: a group of racing dogs in a race .

图 4-15 描述为差

Figure 4-15 Poor captioning

5) 描述和主体图像内容完全不符，或者还有语法语义上的错误——极差。

Greedy: a black and white dog is running along the beach .
Beam k=3: a black and white haired dog is playing with a red toy .
Beam k=5: a black and white dog with a toy in its mouth in the water .
Beam k=7: a black and white dog with a toy in its mouth in the water .
Beam k=9: a black and white dog with a toy in its mouth plays with a toy .
Beam k=13: a black and white dog with a ball in its mouth plays with a toy .
Beam k=20: a group of black dogs play in the water .

Greedy: a man in a red shirt and black shorts is jumping in the air in front of a white mountain .
Beam k=3: a black and white dog jumps in the air to catch a ball .
Beam k=5: a group of people are jumping in front of a mountain .
Beam k=7: a group of people are jumping in the air in front of a mountain .
Beam k=9: a group of black dogs playing in the snow
Beam k=13: a group of black dogs playing in the snow
Beam k=20: a group of black dogs playing in the mountains .

图 4-16 描述为极差

Figure 4-16 Dreadful captioning

4.5.5 开放式检测结果

由此，使得“开放式”检测成为可能。

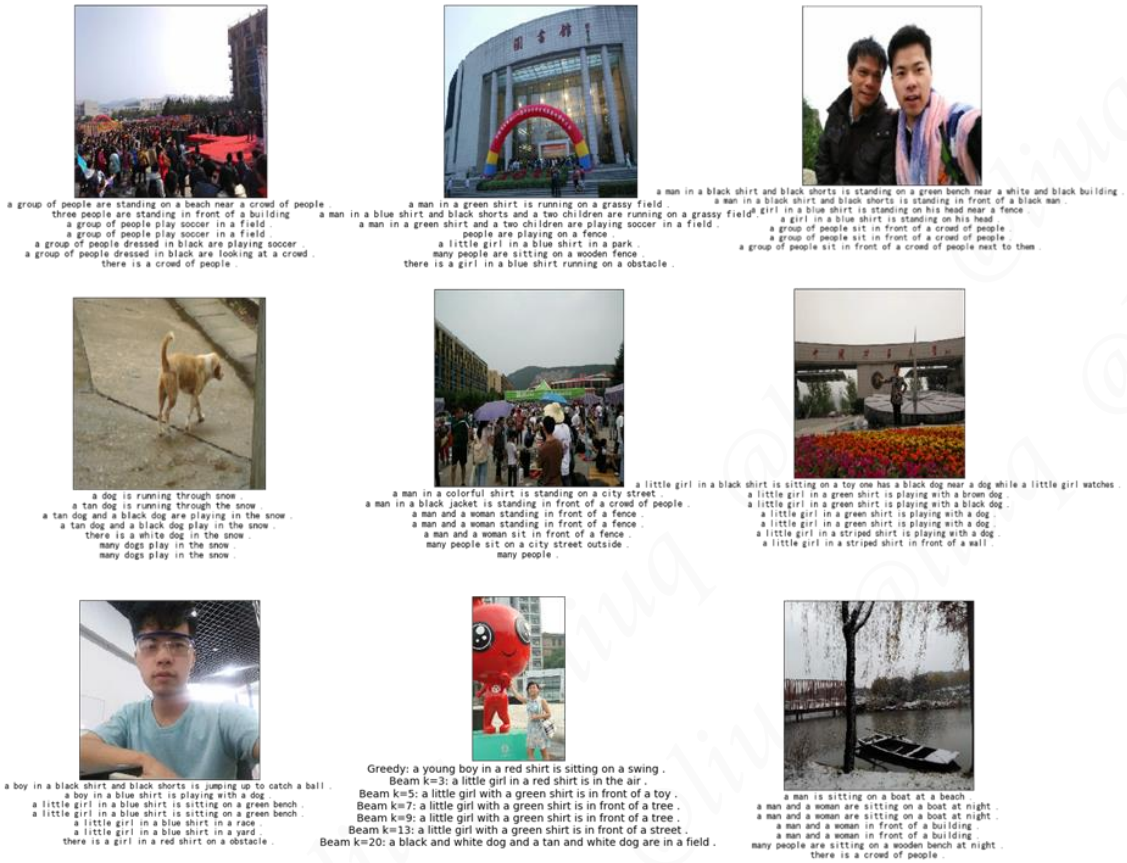


图 4-17 “开放式”描述结果

Figure 4-17 Open-world captioning

对于数据集以外的图像，使用前文所述的处理方法，也能够得到预期的和图像内容相关且可读懂的描述，但直观看来有些地方仍不够准确，比如男女性区分不清，颜色表述失误，对象的动作表述失误等。有的描述结果则“匪夷所思”，比如图 4-17 右下方的描述则“无中生有”认为空船上坐了一个人等等。证明模型在泛化上还有很多提升的空间。

4.6 本章小结（Summary）

本章是本设计的研究重点。主要工作在于基于 LSTM 和几种经典的深度卷积神经网络分别构建了几种图像描述模型，介绍了训练的细节，并对实验方法软硬件平台，损失函数正则化方式方面进行了改进。对于测试，使用贪婪搜索和 beam search 的方法来生成描述，采用机器翻译中的 BLEU 评价指标来定量评价模型，结果显示部分模型指标超过了现有的某些模型，但在距离现有最好水平以及人类水平还有一定差距。另外，还对比分析了模型之间的时间性能，得

出了模型间的时间效率主要在于语言模型上的结论，并给出了影响假设和改进措施。通过抽样还对对模型描述结果给出了定性评价，以及将模型应用于开放式检测任务中，结果能够生成和目标任务相符的图像描述，部分结果差强人意或者偏差很大，说明模型在泛化能力上还有相对较多的。

5 模型可视化和交互设计
5 Visulization and Interations Design of Models

5.1引言（Introduction）

上一章中，完成了对基于 LSTM 的多种描述模型从构建到训练再到测试分析整个完整流程。本章将对模型的计算图可视化，交互式结果展示，特征可视化等开展进一步的工作。

5.2基于 matplotlib 和 d3.js 的模型可视化（Model Visualization Based on matplotlib and d3.js）

由于构建的模型本身的抽象性，不利于人的理解和对于其运行机制有直观的理解，可视化毫无疑问可以是解决该问题的途径之一。用来可视化的方法有很多，前几章中对于模型训练和生成的结果，在工程上使用 matplotlib 和 d3.js（见附录 A）进行了结果的可视化以及初步的交互。

例如使用 matplotlib 绘制训练 loss 曲线和精确度曲线等数学函数图像，以及对图像经过卷积网络处理得到的中间各层的卷积特征可视化展示，例如，将图 5-1 中经过 VGG 网络处理得到的各层特征图可视化展示为：

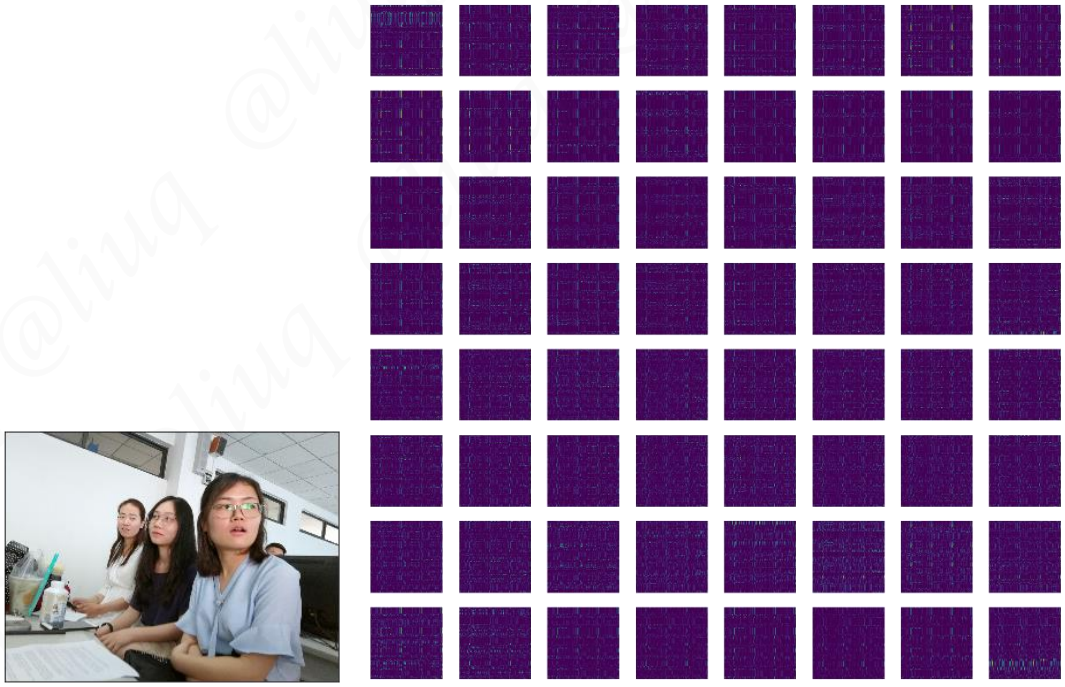


图 5-1 基于 matplotlib 的卷积特征可视化

Figure 5-1 Feature maps of a images shown by matplotlib

另外结合 d3.js，使得模型产生的结果不在局限于在 IDE 中打开和展示而可以在浏览器中展示，同时增加了图表交互式的效果，以图 5-2 为例，为散点图曲线中的每个端点增加了气泡标注来显示当前点迭代次数以及损失指，同时具有还有图标的移动和缩放等功能。

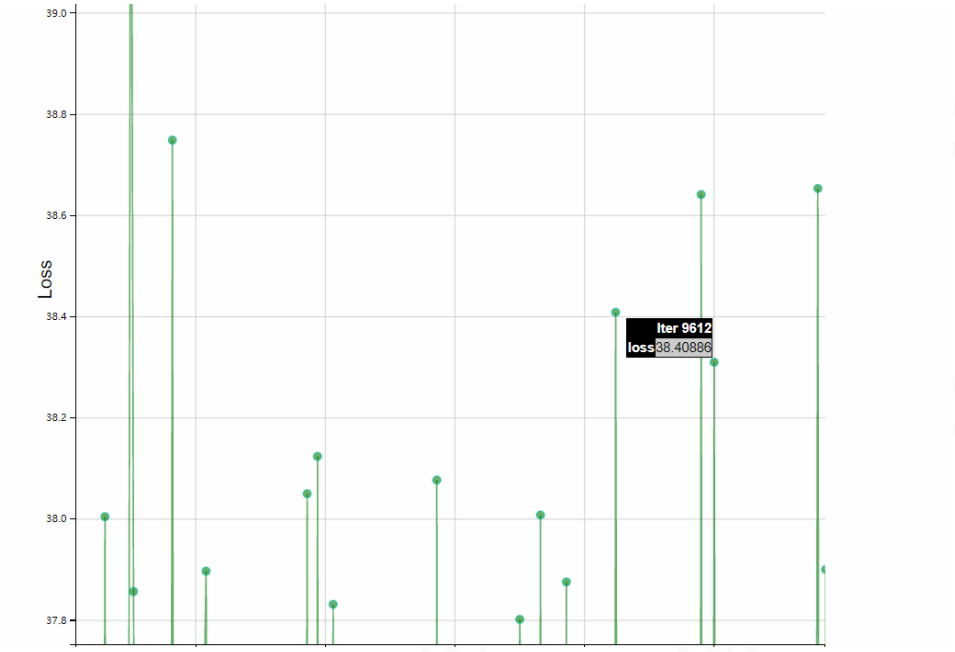


图 5-2 基于 d3.js 的图表可视化示例

Figure 5-2 Illustration of the visualization of d3.js

5.3基于 Tensorboard 的模型可视化（Model Visualization Based on Tensorboard）

对于描述模型结构的可视化，这里采用 Tesorflow[46]（见附录 A）中的 Tensorboard，进行了构建。得到的模型整体计算图如下：

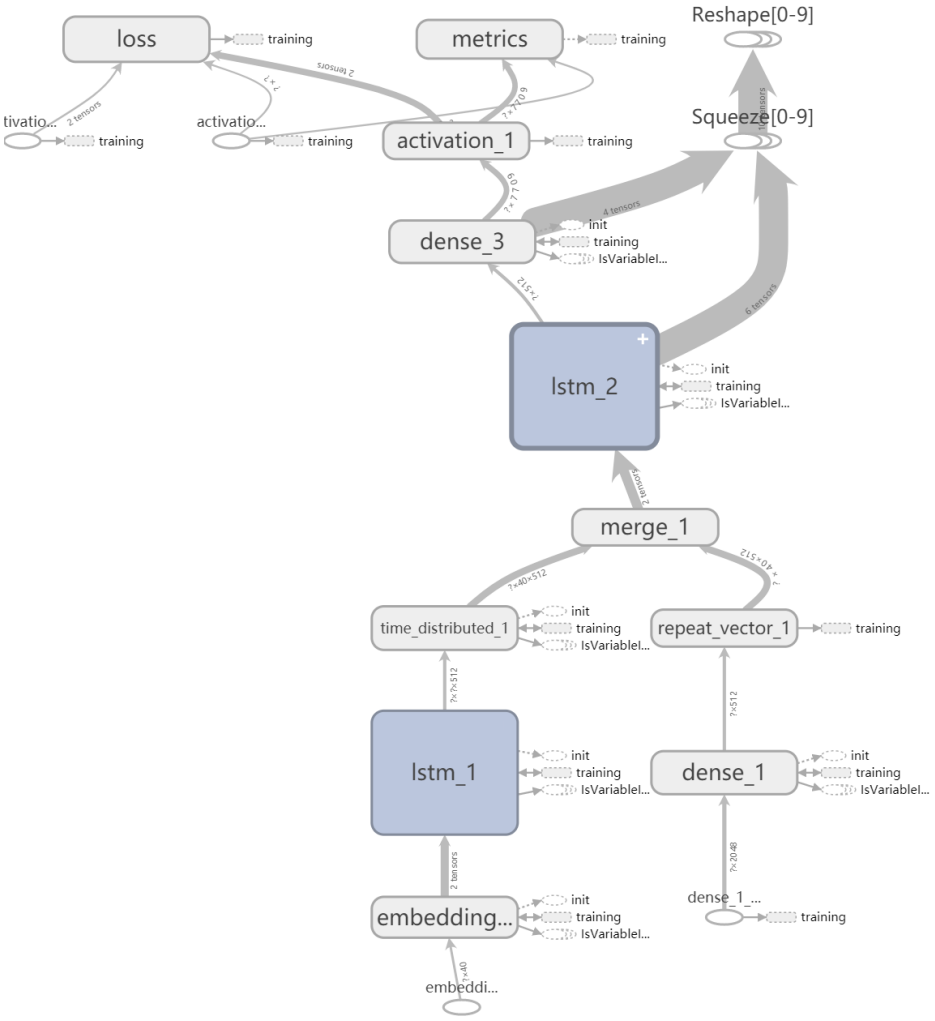


图 5-3 基于 TensorBoard 的模型可视化

Figure 5-3 Model virtualization Based on Tensorboard

图中带箭头的线条代表了数据的流动，线条上的数字表明了数据的规模，越宽的地方代表数据规模越大，箭头的方向代表了数据的流向。对应于第四章中的 LSTM 描述模型的结构，其中的词汇嵌入层可视化如下：

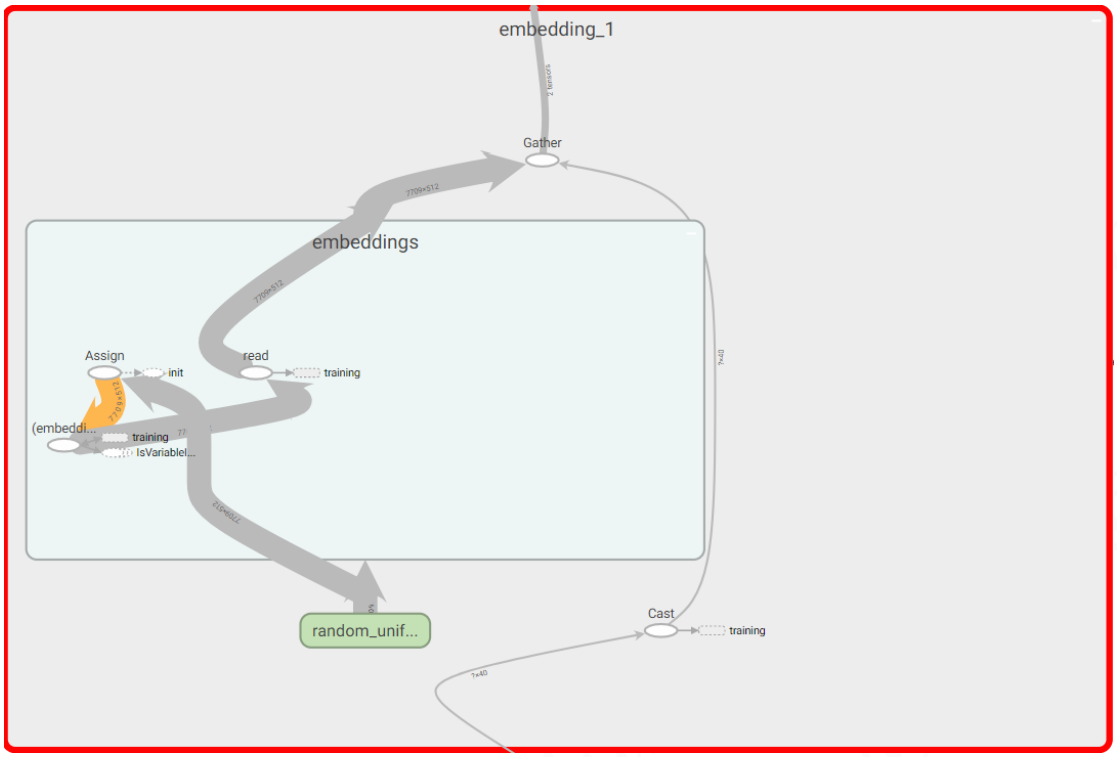


图 5-4 词汇潜入层可视化结果

Figure 5-4 Visualization of word embedding

对于全连接层的可视化结果：

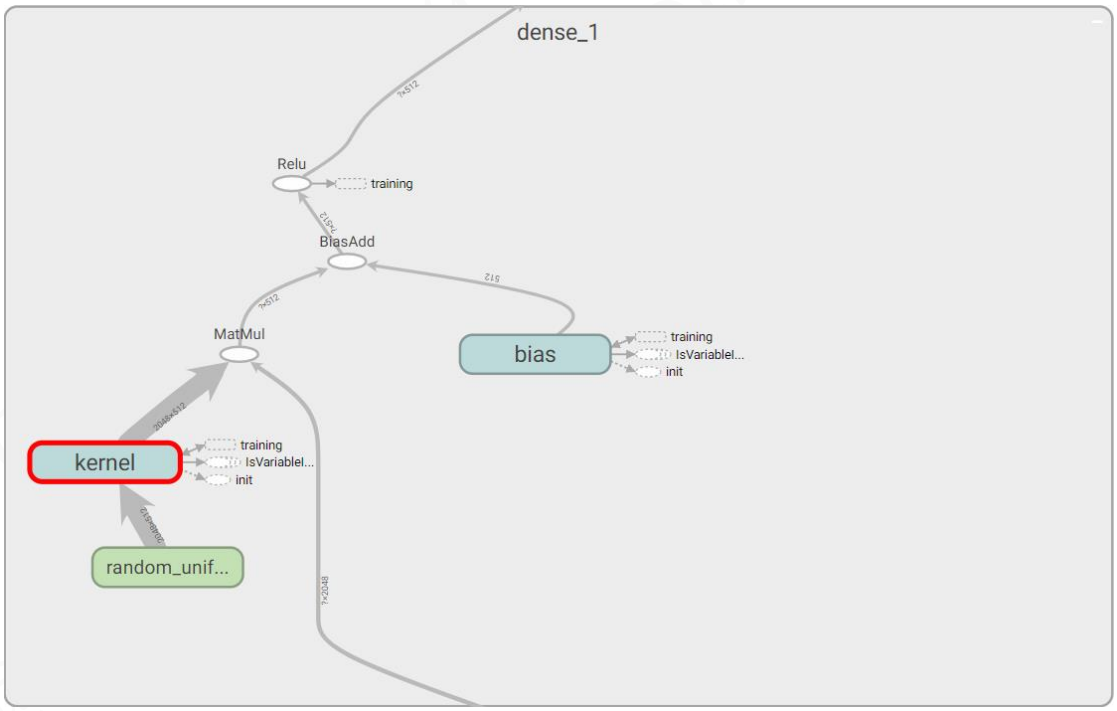


图 5-5 全连接层的可视化结果

Figure 5-5 Virtualization of dense layer

对于其中的 LSTM 层的输入来源显示：

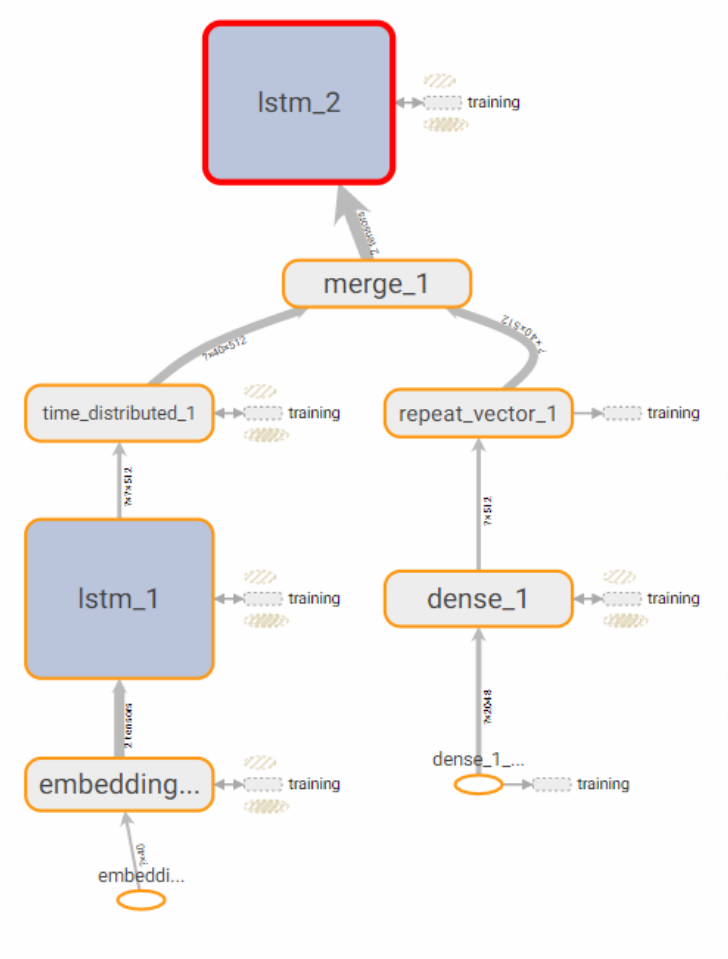


图 5-6 LSTM 输入可视化结果

Figure 5-6 Visualization of showing inputs of LSTM

整个训练过程的示意图：

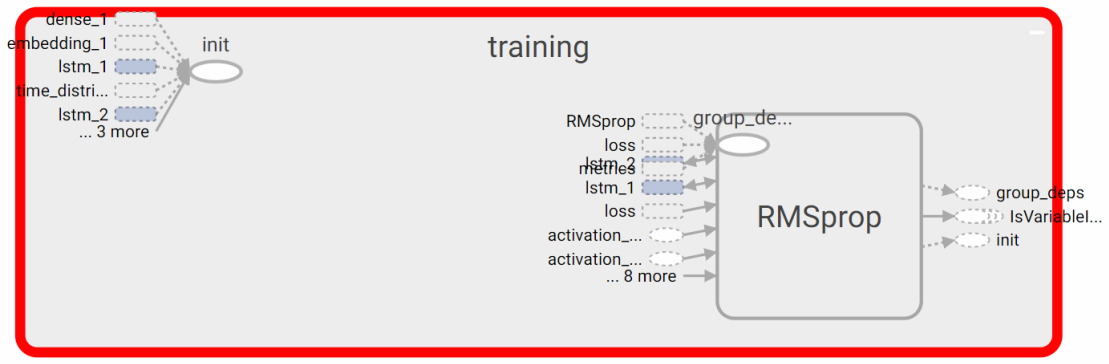


图 5-7 训练过程可视化示意图

Figure 5-7 Illustration of training

训练过程中 loss 的来源：

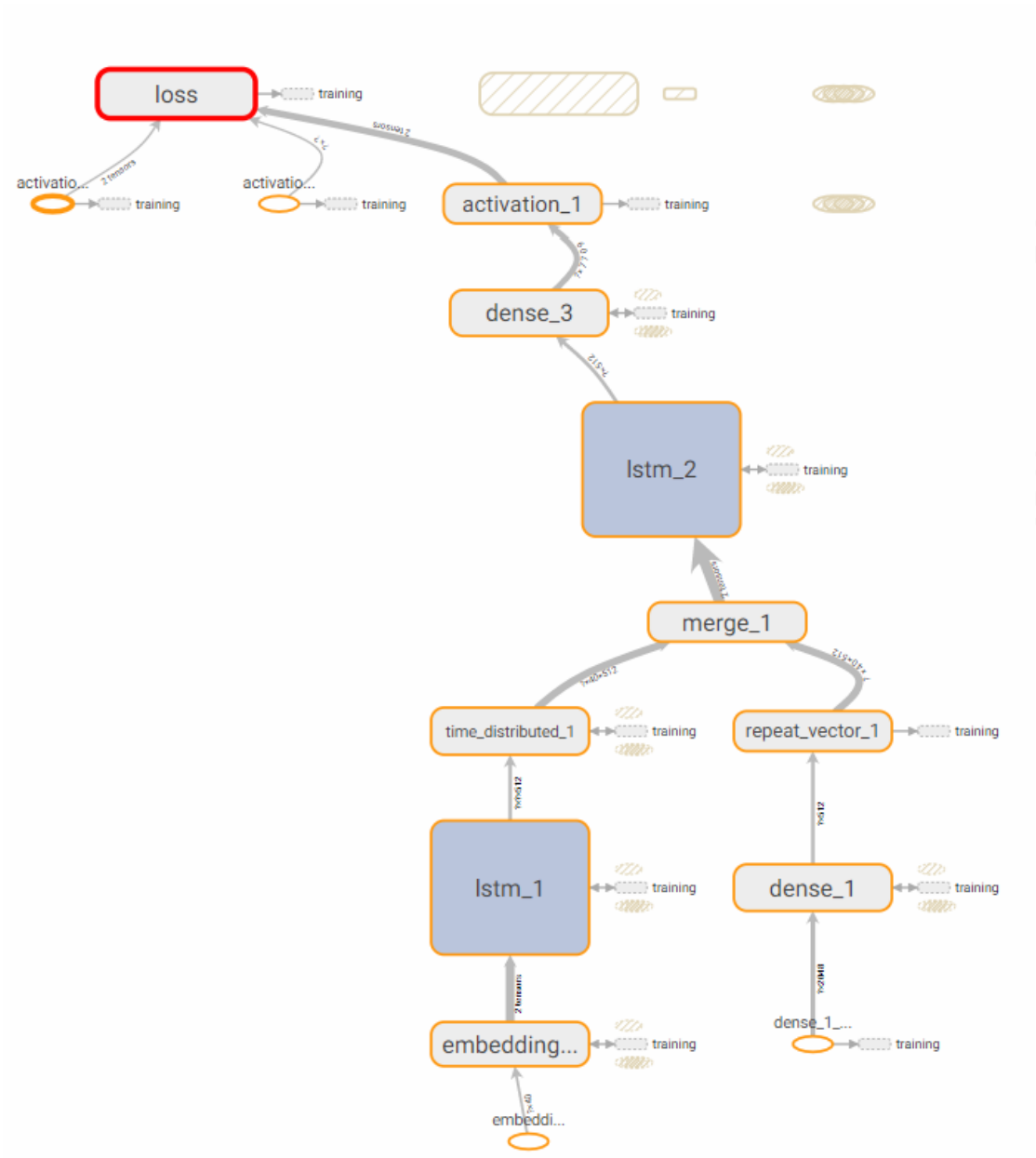


图 5-8 训练过程损失计算可视化示意图

Figure 5-8 Illustration of showing inputs of loss

训练过程中计算精确度的过程：

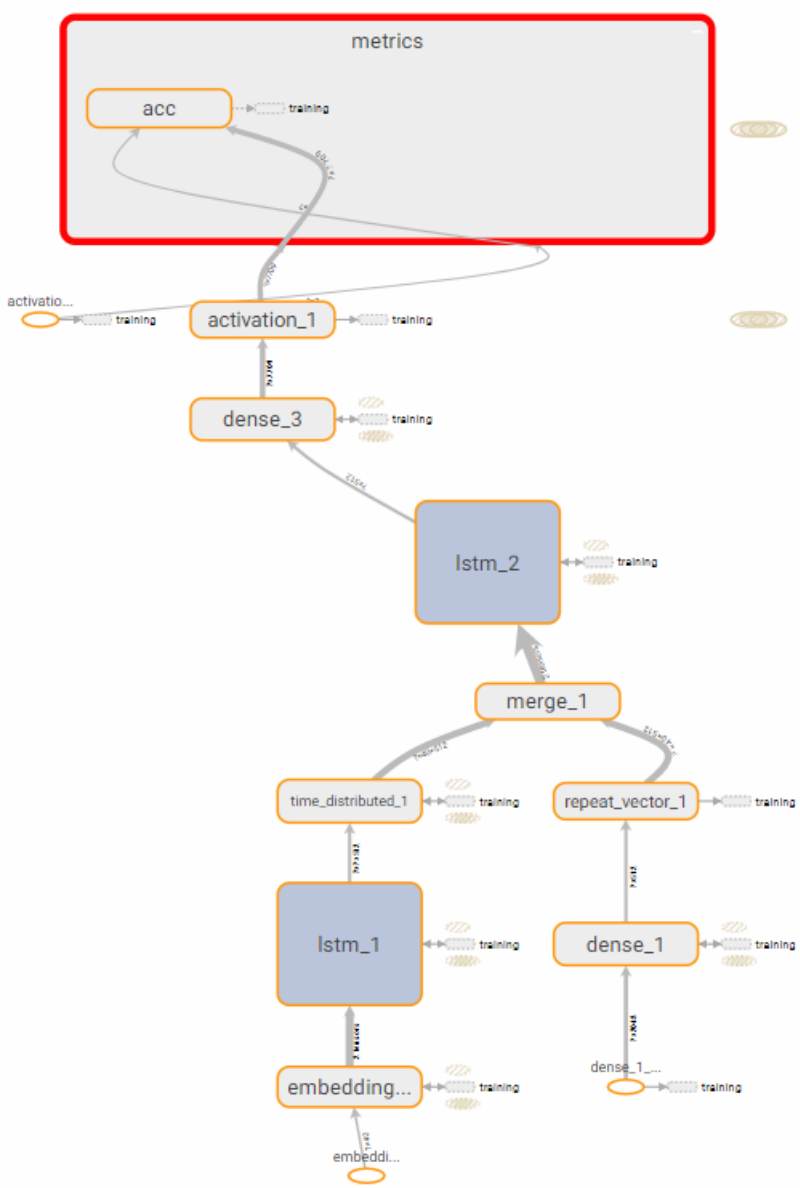


图 5-9 训练过程精确度计算可视化示意图

Figure 5-9 Illustration of showing inputs of the calculation of accuracy

5.4本章小结（Summary）

本章针对模型结构复杂抽象，不易理解和演示的问题。结合了若干种工程技术和方法，对图像的训练和测试中间结果，模型的结构运行机理进行了可视化方法的研究和实践。

6 将模型应用于工程之中以基于 Django 的 B/S 系统为例
6 Application of Image Captioning Model

6.1 引言（Introduction）

上一章节中完成了对于模型的可视化，本章中将以工程项目为驱动，来构建基于图像描述原理和模型的应用系统，做到理论和实践相结合，以软件工程的方法，构建基于 Django 的 B/S 端图像描述系统。

6.2 基于 Django 的描述生成系统（Image Captioning Application Based on Django）

6.2.1 需求分析和设计模式

基于之前的工作，下面来构建图像描述生成应用系统，来使得模型能够被更多地使用和访问，同时也相当提供了一种数据可视化的方案。整个 Image Captioning 系统的主要功能需求如下：

数据集管理：对模型采用的数据集信息进行管理，包含数据集图像保存，记录的增添，删除，修改，排序，搜索等功能。提供用户访问，交互友好的接口；

模型管理：包含模型的创建，保存，读取，训练，测试等。

用户管理：一般应用系统必须的功能，包含用户的登录，注册，账户信息维护增添，删除，修改，排序，搜索，找回，激活等功能。

基础功能：富文本编辑，验证码生成，文件导入导出，安全性，等等。

扩展功能：兼容性，移动设备支持，等等。

基于以上需求，本设计构建了基于 Django（见附录 A.4）的在线描述生成系统，将功能需求分解成为一个个的 APP 应用：

Admin/xadmin: 后台管理子应用。

Show2tell：ImageCaptioning 模型可视化展示和测试。

Playground：ImageCaptioning 模型的在线训练和调试。

Notebook：开发文档展示，程序文件下载，模型参数文件和源代码下载管理等，可视作一个静态元素的博客子系统。

Users：用户和用户操作管理，预留一些功能接口。

具体来说，系统总体采用 MVT（model-view-template，模型-视图-模板）分离的设计模式，Model 层负责所有数据的管理，包括数据集文件和信息，以

及其余对象和操作的数据。View 层负责和用户交互，获取用户的请求以及和后端模型层沟通。Template 即用以管理和返回用户可视的界面或提示。三层相互独立，通过层间的接口函数相互调用和进行通信。

6.2.2 Template 层设计

Template 层具体到实现上，基于 HTML+JavaScript+Bootstrap 前端响应式框架（见附录 A.4）创建出系统的若干功能界面。

数据集查看：查看数据集的图像和描述信息。

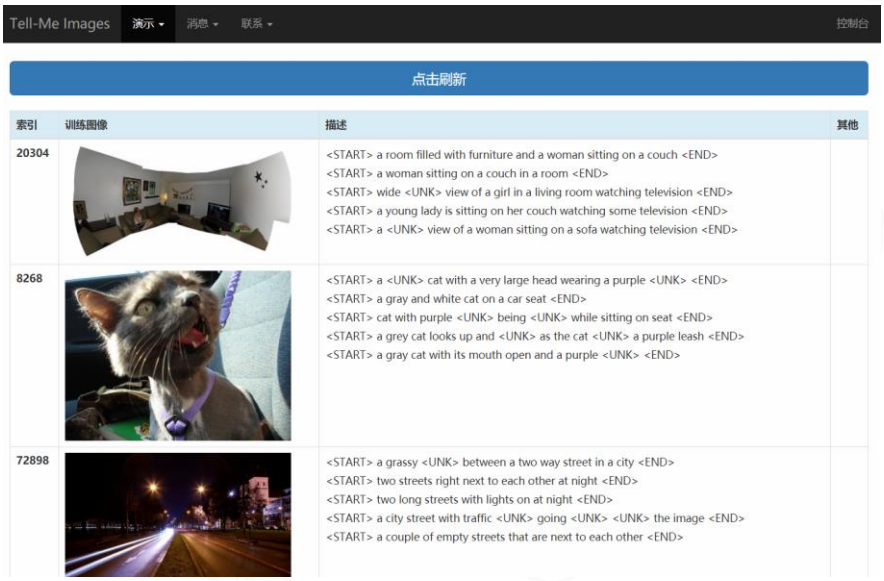


图 6-1 数据集前端展示示例

Figure 6-1 Illustration of showing dataset to the frontend

功能演示界面，用于对图像描述模型描述效果的呈现，以及提供请求的 API 接口，为未来可能的商业化应用打下一定基础。

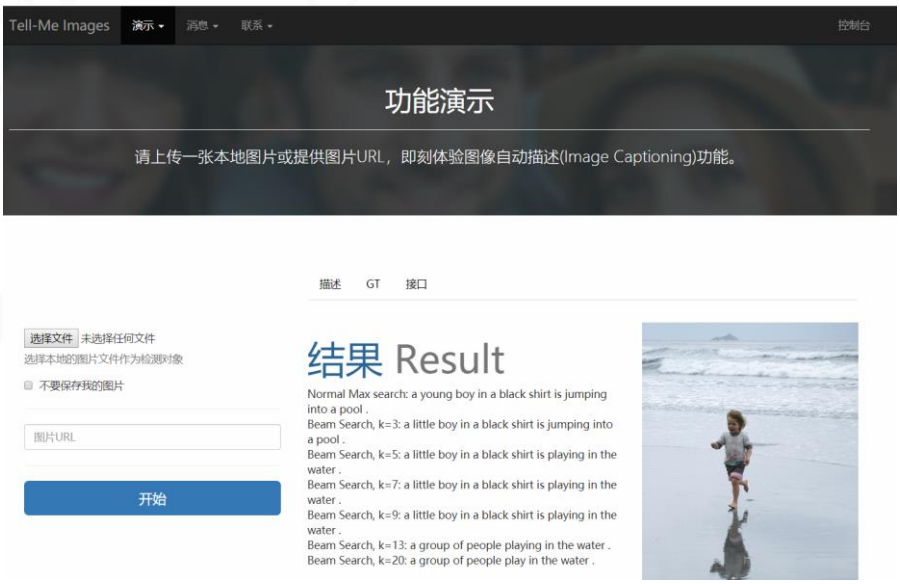


图 6-2 图像描述结果前端展示

Figure 6-2 Illustration of image captioning shown to the frontend

原理呈现即需求中的 notebook 子应用：本质上是一个小型的博客子系统，用于对于模型构建源代码，相关原理功能，用户文档手册，发布更新信息，提供软件下载等功能进行呈现。

图 6-3 子应用之 Notebook

Figure 6-3 Notebook app of the project

用户管理：包含用户登陆注册，验证激活，密码找回，留言板等相关功能进行呈现。

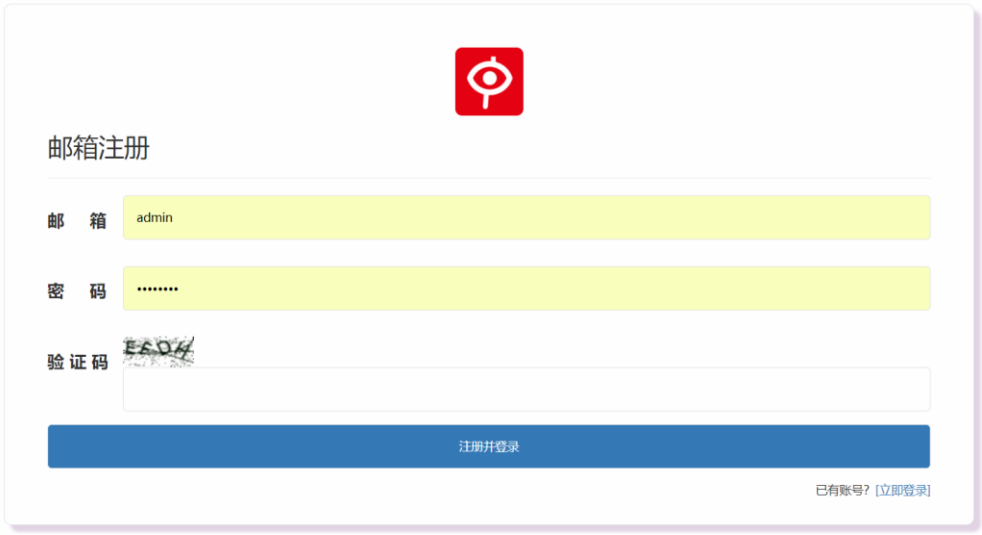


图 6-4 用户管理子应用示例之一的用户注册

Figure 6-4 Illustration of user register of the User app

后台管理子应用：其属于前后端的结合，如图 6-5 所示，管理员可以对于所有用户，后端数据，前端页面元素，文件等进行包含增加删除，修改，筛选，排序，导入，导出等一系列操作。

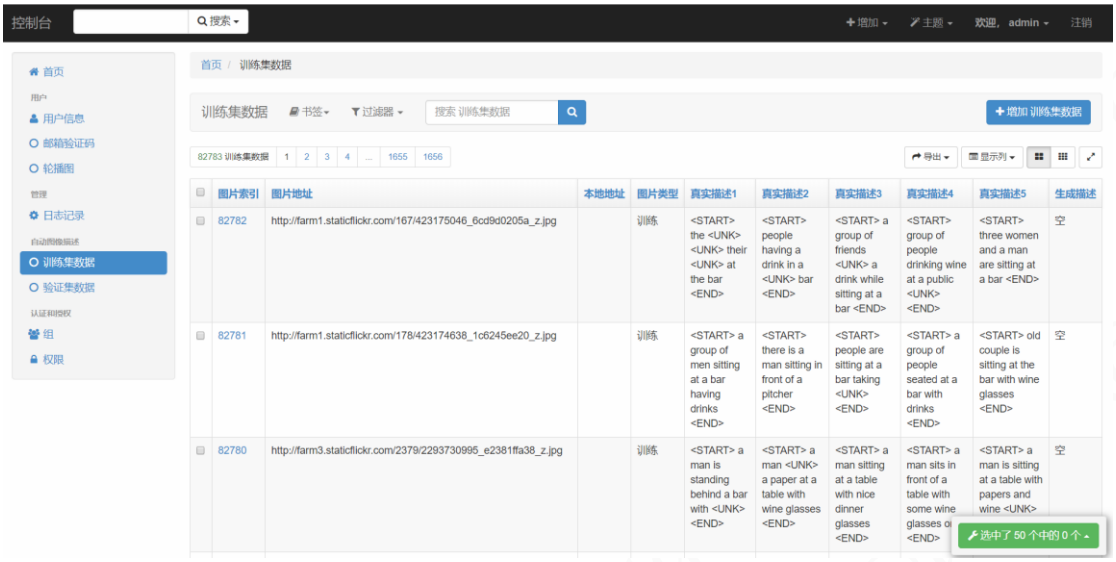


图 6-5 控制台示意图 1

Figure 6-5 Illustration 1 of the console

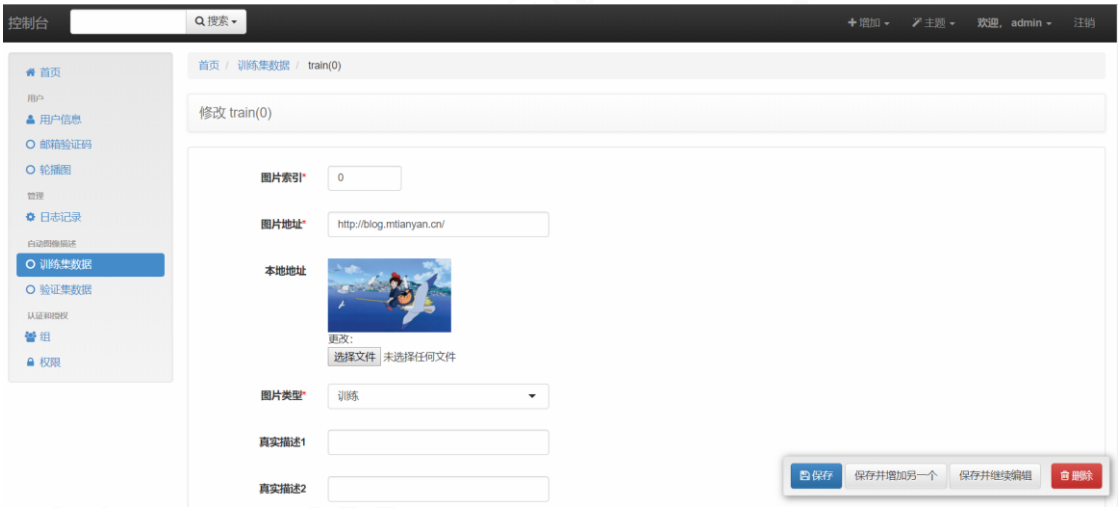


图 6-6 控制台示意图 2

Figure 6-6 Illustration 2 of the console

6.2.3 数据表设计

Model 层体现在本设计中体现为使用 MySQL 关系数据库来构建数据表，定义数据表关键字段，键值间关系等，图 6-7 为应用系统部分数据表以及训练数据存储记录。

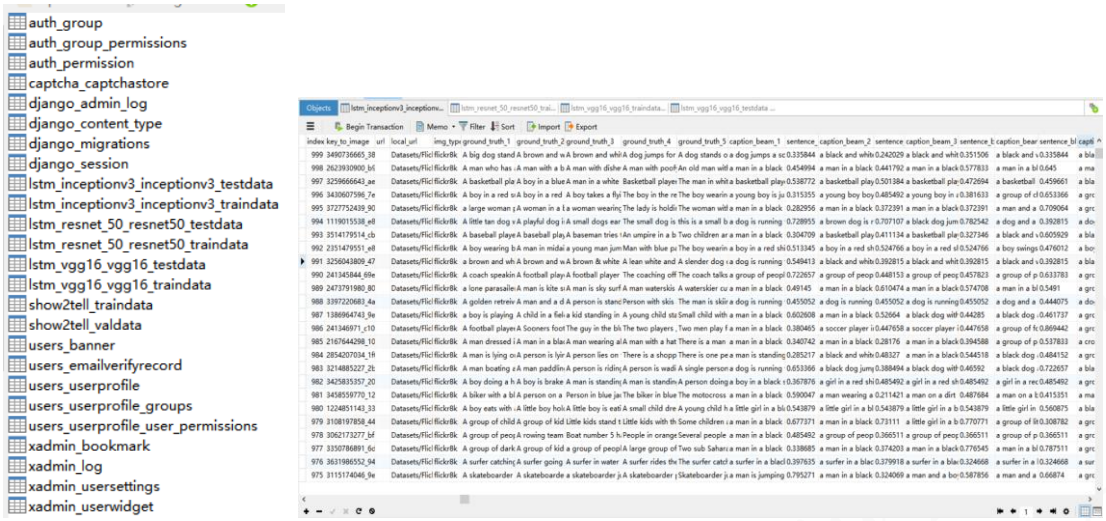


图 6-7 项目模型层中的部分数据表

Figure 6-7 Some tables of the Model in the project

其中的代表性数据表如下：

表 6-1 图像描述关系表

Table 6-1Table of image captioning

字段名称	字段类型	字段长度	字段意义
id	int	11	记录 ID
index_to_image	int	11	图片索引
url	varchar	250	远程链接
local_url	varchar	200	本地链接
img_type	varchar	20	图像类型
ground_truth_1	varchar	200	真实描述 1
ground_truth_2	varchar	200	真实描述 2
ground_truth_3	varchar	200	真实描述 3
ground_truth_4	varchar	200	真实描述 4
ground_truth_5	varchar	200	真实描述 5
caption_beam_1	varchar	200	生成描述 1
sentence_bleu_1	double	0	描述 1 对于 gt1~5 的 blue-1 得分
caption_beam_2	varchar	200	生成描述 2
sentence_bleu_2	double	0	描述 2 对于 gt1~5 的 blue-1 得分
caption_beam_3	varchar	200	生成描述 3

sentence_bleu_3	double	0	描述 3 对于 gt1~5 的 blue-1 得分
caption_beam_4	varchar	200	生成描述 4
sentence_bleu_4	double	0	描述 4 对于 gt1~5 的 blue-1 得分
caption_beam_5	varchar	200	生成描述 5
sentence_bleu_5	double	0	描述 5 对于 gt1~5 的 blue-1 得分

6.2.4 View 层重构和移植

所有的代码逻辑都是在 View 层中进行的。将前文所述图像描述模型的 Python 逻辑封装重构然到 Django 项目中创建的函数接口中，设置不同的 url 即对应的 view，在接口中处理包含 ImageCaptioning 模型的构建和训练，数据库存取等全部操作，通过返回 render 对象和 json 或者字典数据，从而使得前端更新显示。

6.2.5 系统部署运行

系统部署即让整个系统运行在远程主机或者云服务器上，从而实现随时随地访问，这也是本模型系统使用 B/S 结构的原因，其具体原理和技术如图 6-8 所示，不属于本文的主体内容，参见附录 A.4。

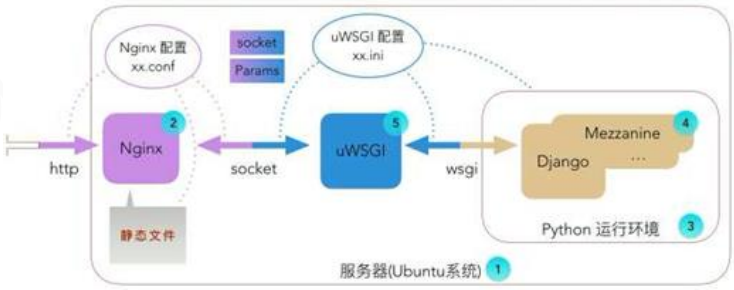


图 6-8 项目部署上线

Figure 6-8 Deployment of the project

部署成功后的系统如图 6-9 所示。

图 6-9 项目在线运行

Figure 6-9 Project on the cloud

6.2.6 系统发布

系统部署上线之后，经过运行测试，功能调试达到足够可用，随后对系统进行发布上线。部分结果如下：

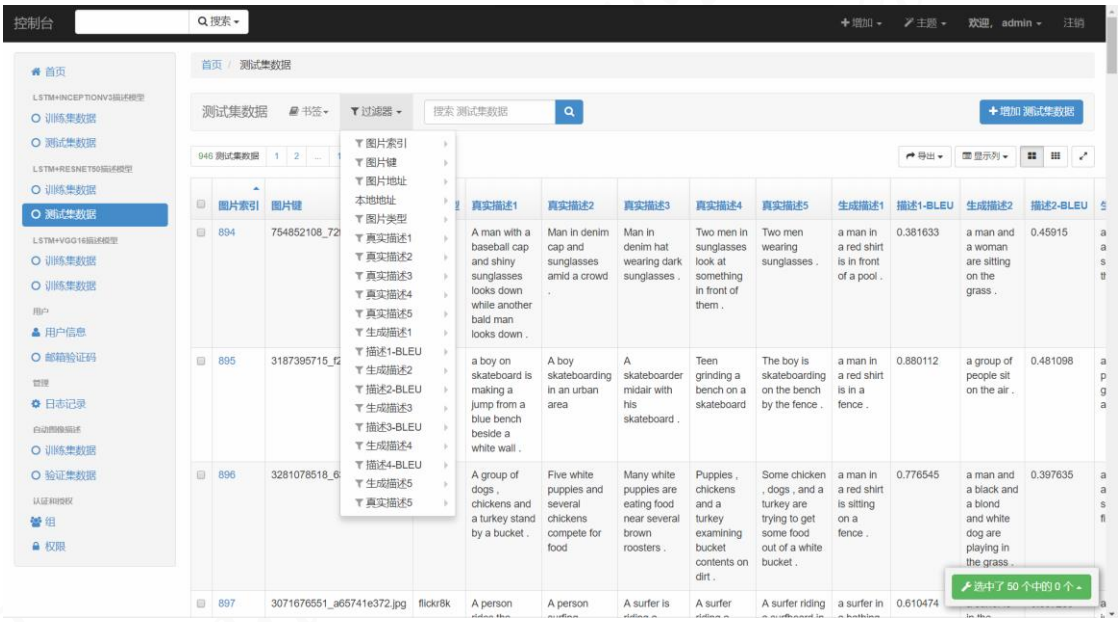


图 6-10 部分数据集和结果

Figure 6-10 Part of datasets and results

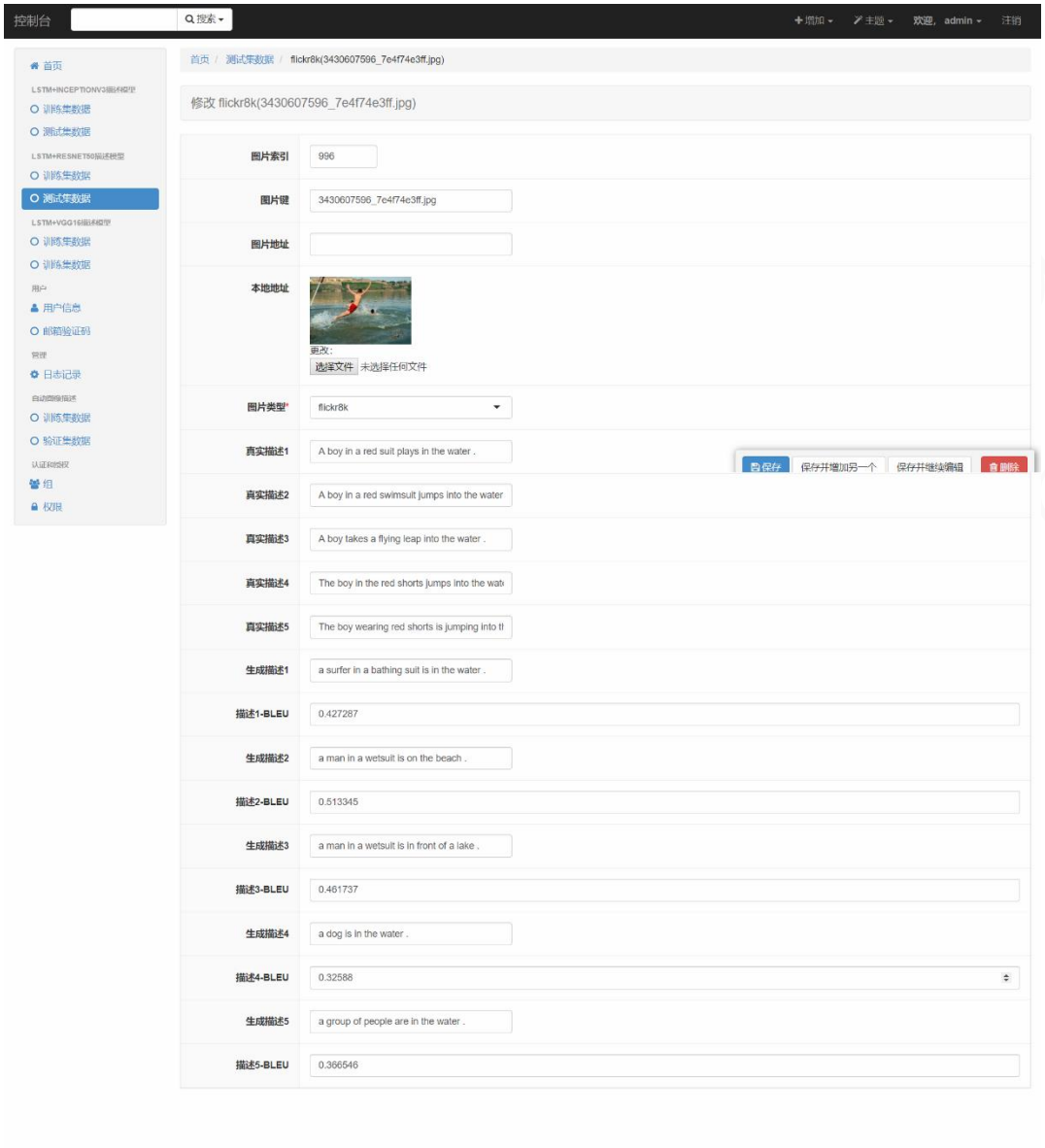


图 6-11 数据集和结果记录示意

Figure 6-11 Illustration of one image record

6.3本章小结（Summary）

本章对于图像描述在工程中的需求进行了详细分析，基于多种工程技术以将原理和模型应用于在线 B/S 架构的图像描述应用为目标，采用多种工程技术，如 Django, Bootstrap, HTML, JS 等，采用 MVT 设计模式，构建起了完整的图像描述应用系统部署上线并成功发布，将理论和模型推进到具体的应用中。并且以应用来反推和倒逼模型和原理的改进。

7 结论和展望
7 Conclusions and future works

7.1结论（Conclusions）

本设计的重点工作在于采用自顶而下的方法，基于 LSTM 和深度卷积神经网络，结合几种经典的卷积网络（VGG, GoogLeNet 和 ResNet）构建了的端到端的神经网络图像描述模型，能够从输入图像中生成和图像内容相关且可读懂的自然语言描述。模型采用深度卷积神经网络作为编码器从图像中提取特征，传递到随后的 LSTM 解码器中来生成描述，属于 CNN+RNN 的 NIC（Neural Image Caption）结构。使用随机梯度下降算法，通过最小化负对数似然损失函数来训练模型。BLEU 作为模型生成描述的评价指标的实验结果表明本设计的模型的描述水平在部分指标上超过部分现有模型，描述能力总体处于中等水平，但是在部分指标上距离目前最新研究成果以及人类水平还有较大差距。以上述多种模型的对比实验结果表明，模型间的主要时间性能差异来自于后者的语言模型，而描述性能的对比评价，由于实验数据集很运算量较为巨大，实验平台计算能力有限，部分结果没出算出，期待后期能够进一步对已完成的部分加以改进和测试，以获得更好的描述结果。此外，也基于标准 RNN 构建了图像描述，在小样本和大样本上测试的结果显示后者同样可以产生自然语言的描述。另外，还搭建基于 B/S 的在线图像自动描述系统。

7.2展望（Future works）

虽然本设计取得了一些工作和结果但仍任存在许多不足之处，例如在本设计中模型评价指标过于单一，后期将采用更多的评价，并对构建的多种模型进行对比，发现性能差别和影响因素之所在。对于模型防止过拟合正则化以及欠拟合的训练调试方法还需要进一步研究和改进。此外还有很重要的一点，由于使用的数据集均是基于英文的，而中文作为世界使用人口最多的语言，毫无疑问是英文之外值得去扩展的维度。

参考文献

[1] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. CoRR, abs/1411.4555, 2014.

[2] M.Soh, “Learning CNN-LSTM architectures for image caption generation,” http://cs224d.stanford.edu/reports/msoh.pdf,2016.

[3] M. Pedersoli, T. Lucas, C. Schmid, and J. Verbeek. Areas of attention for image captioning. In The IEEE International Conference on Computer Vision (ICCV), Oct 2017. 2, 3.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet large scale visual recognition challenge,” Int. J. Comput. Vis. (IJCV), vol. 115, no. 3, pp. 211–252, 2015.

[5] Xirong Li, Weiyu Lan, Jianfeng Dong, and Hailong Liu. 2016. Adding chinese captions to images. In Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval. pages 271–275

[6] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. In Proceedings of the 11th European Conference on Computer Vision: Part IV, ECCV’10, pages 15–29, Berlin, Heidelberg, 2010. Springer-Verlag.

[7] Polina Kuznetsova, Vicente Ordonez, Alexander C. Berg, Tamara L. Berg, and Yejin Choi. Collective generation of natural image descriptions. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1, ACL ’12, pages 359–368, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[8] Xinlei Chen and C. Lawrence Zitnick. Learning a recurrent visual representation for image caption generation. CoRR, abs/1411.5654, 2014.

[9] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, and Alan L. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). CoRR, abs/1412.6632, 2014.

[10] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. CoRR, abs/1411.2539, 2014.

[11] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. CoRR, abs/1411.4389, 2014.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[13] Ian Goodfellow Yoshua Bengio and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[14] Alan Turing.Computing Machinery and Intelligence.1950

[15] Teven Levy.Hackers: Heroes of the Computer Revolution. Anchor Press/Doubleday. 1984

[16] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of Go without human knowledge. Nature 550, 354–359(2017).

[17] Karpathy, A. 2016. Connecting Images and Natural Language. PhD Thesis, Department of Computer Science, Stanford University.

[18] 李嘉璇. TensorFlow 技术解析与实践.人民邮电出版社.2017.

[19] Mitchell TM, Carbonell JG, Michalski RS. Machine Learning.McGraw-Hill.2003

[20] 周志华.机器学习.清华大学出版社.2016

[21] Xiu-Shen Wei.CNN_Book. http://lamda.nju.edu.cn/weixs.2017

[22] 邱锡鹏.神经网络与深度学习讲义.http://nlp.fudan.edu.cn/dl-book.2015.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.Gradient-Based learning applied to document recognition. Procedings of the IEEE, pages 1-46,1998

[24] Yoshua Bengio, Aaron Conville, and Pascal Vincent. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8):1798-1828,2013.

[25] 山世光, 阚美娜, 刘昕, 刘梦怡, 邬书哲.深度学习：多层神经网络的复兴与变革.科技导报.34(14),60-70.2016.

[26] LeCun, Y. Generalization and network design strategies. Technical Report CRG-TR-89-4. University of Toronto.1989

[27] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. arXiv:1412.6806 [cs], December 2014.

[28] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems (NIPS). Cambridge MA: MIT Press, 2012: 1097-1105.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.

[30] Olah, C. Understanding LSTM Networks. Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/.2015.

[31] Perronnin, F., Sánchez, J., & Mensink, T.. Improving the fisher kernel for large-scale image classification. In ECCV (4). 2010

[32] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. BMVC, 2014

[33] Lin, M., Chen, Q., and Yan, S. Network in network. In Proc. ICLR, 2014.

[34] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

[35] K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition. CVPR 2016.

[36] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. Rethinking the incep-tion architecture for computer vision. In: CVPR. (2016).

[37] Szegedy, C., Ioffe, S., Vanhoucke, V. Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv:1602.07261 (2016).

[38] Andrej Karpathy and Fei-Fei Li. Deep visual-semantic alignments for generating image descriptions.

CoRR, abs/1412.2306, 2014.

[39] Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Kumar Srivastava, Li Deng, Piotr Doll´ar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. From captions to visual concepts and back. CoRR, abs/1411.4952, 2014.

[40] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. DRAW: A recurrent neural network for image generation. CoRR, abs/1502.04623, 2015.

[41] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. CoRR, abs/1406.6247, 2014.

[42] M. Hodosh, P. Young, and J. Hockenmaier. Framing image description as a ranking task: Data, models and evaluation metrics. JAIR, 47, 2013.

[43] J. Mao, W. Xu, Y. Yang, J. Wang, and A. Yuille. Explain images with multimodal recurrent neural networks. In arXiv:1410.1090, 2014.

[44] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. In arXiv:1411.2539, 2014.

[45] Papineni, Kishore, Salim Roukos, Todd Ward, and WeiJing Zhu. "BLEU: A Method for Automatic Evaluation of Machine Translation." ACL, 2002.

[46] P Barham, J Chen, Z Chen, A Davis, J Dean TensorFlow: A System for Large-Scale Machine Learning.M Abadi, OSDI, 2016

翻译部分

翻译原文

DenseCap: Fully Convolutional Localization Networks for Dense Captioning

Justin Johnson* Andrej Karpathy* Li Fei-Fei
Department of Computer Science, Stanford University
{jcjohns,karpathy,feifeili}@cs.stanford.edu

Abstract

We introduce the dense captioning task, which requires a computer vision system to both localize and describe salient regions in images in natural language. The dense captioning task generalizes object detection when the descriptions consist of a single word, and Image Captioning when one predicted region covers the full image. To address the localization and description task jointly we propose a Fully Convolutional Localization Network (FCLN) architecture that processes an image with a single, efficient forward pass, requires no external regions proposals, and can be trained end-to-end with a single round of optimization. The architecture is composed of a Convolutional Network, a novel dense localization layer, and Recurrent Neural Network language model that generates the label sequences. We evaluate our network on the Visual Genome dataset, which comprises 94,000 images and 4,100,000 region-grounded captions. We observe both speed and accuracy improvements over baselines based on current state of the art approaches in both generation and retrieval settings.

1. Introduction

Our ability to effortlessly point out and describe all aspects of an image relies on a strong semantic understanding of a visual scene and all of its elements. However, despite numerous potential applications, this ability remains a challenge for our state of the art visual recognition systems. In the last few years there has been significant progress in image classification [39, 26, 53, 45], where the task is to assign one label to an image. Further work has pushed these advances along two orthogonal directions: First, rapid progress in object detection [40, 14, 46] has identified models that efficiently identify and label multiple salient regions of an image. Second, recent advances in image captioning [3, 32, 21, 49, 51, 8, 4] have expanded the complexity of the label space from a fixed set of categories to sequence of words able to express significantly richer concepts.

However, despite encouraging progress along the label density and label complexity axes, these two directions have remained separate. In this work we take a step towards unifying these two inter-connected tasks into one joint framework. First, we introduce the dense captioning task (see Figure 1), which requires a model to predict a set of descriptions across regions of an image. Object detection is hence recovered as a special case when the target labels consist of one word, and image captioning is recovered when all images consist of one region that spans the full image.

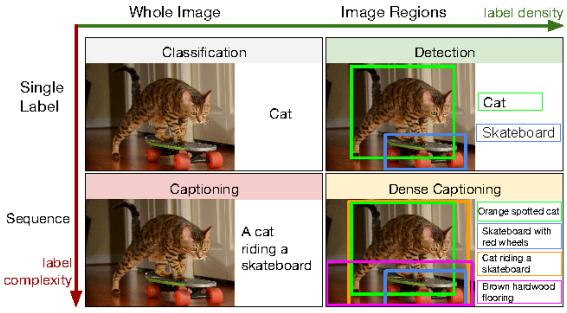


Figure 1. We address the Dense Captioning task (bottom right) with a model that jointly generates both dense and rich annotations in a single forward pass.

Additionally, we develop a Fully Convolutional Localization Network (FCLN) for the dense captioning task. Our model is inspired by recent work in image captioning [49, 21, 32, 8, 4] in that it is composed of a Convolutional Neural Network and a Recurrent Neural Network language model. However, drawing on work in object detection [38], our second core contribution is to introduce a new dense localization layer. This layer is fully differentiable and can be inserted into any neural network that processes images to enable region-level training and predictions. Internally, the localization layer predicts a set of regions of interest in the image and then uses bilinear interpolation [19, 16] to smoothly crop the activations in each region.

We evaluate the model on the large-scale Visual Genome dataset, which contains 94,000 images and 4,100,000 region captions. Our results show both performance and speed improvements over approaches based on previous state of the art. We make our code and data publicly available to support further progress on the dense captioning task.

*Both authors contributed equally to this work.

2. Related Work

Our work draws on recent work in object detection, image captioning, and soft spatial attention that allows downstream processing of particular regions in the image.

Object Detection. Our core visual processing module is a Convolutional Neural Network (CNN) [29, 26], which has emerged as a powerful model for visual recognition tasks [39]. The first application of these models to dense prediction tasks was introduced in R-CNN [14], where each region of interest was processed independently. Further work has focused on processing all regions with only single forward pass of the CNN [17, 13], and on eliminating explicit region proposal methods by directly predicting the bounding boxes either in the image coordinate system [46, 9], or in a fully convolutional [31] and hence position-invariant settings [40, 38, 37]. Most related to our approach is the work of Ren *et al*. [38] who develop a region proposal network (RPN) that regresses from anchors to regions of interest. However, they adopt a 4-step optimization process, while our approach does not require training pipelines. Additionally, we replace their RoI pooling mechanism with a differentiable, spatial soft attention mechanism [19, 16]. In particular, this change allows us to backpropagate through the region proposal network and train the whole model jointly.

Image Captioning. Several pioneering approaches have explored the task of describing images with natural language [1, 27, 12, 34, 42, 43, 28, 20]. More recent approaches based on neural networks have adopted Recurrent Neural Networks (RNNs) [50, 18] as the core architectural element for generating captions. These models have previously been used in language modeling [2, 15, 33, 44], where they are known to learn powerful long-term interactions [22]. Several recent approaches to Image Captioning [32, 21, 49, 8, 4, 24, 11] rely on a combination of RNN language model conditioned on image information, possibly with soft attention mechanisms [51, 5]. Similar to our work, Karpathy and Fei-Fei [21] run an image captioning model on regions but they do not tackle the joint task of detection of description in one model. Our model is end-to-end and designed in such way that the prediction for each region is a function of the global image context, which we show also ultimately leads to stronger performance. Finally, the metrics we develop for the dense captioning task are inspired by metrics developed for image captioning [48, 7, 3].

3. Model

Overview. Our goal is to design an architecture that jointly localizes regions of interest and then describes each with natural language. The primary challenge is to develop a model that supports end-to-end training with a single step of optimization, and both efficient and effective inference. Our proposed architecture (see Figure 2) draws on architectural elements present in recent work on object detection, image captioning and soft spatial attention to simultaneously address these design constraints.

In Section 3.1 we first describe the components of our model. Then in Sections 3.2 and 3.3 we address the loss function and the details of training and inference.

3.1. Model Architecture

3.1.1 Convolutional Network

We use the VGG-16 architecture [41] for its state-of-the-art performance [39]. It consists of 13 layers of 3×3 convolutions interspersed with 5 layers of 2×2 max pooling. We remove the final pooling layer, so an input image of shape $3 \times W \times H$ gives rise to a tensor of features of shape $C \times W' \times H'$ where $C = 512$, $W' = \lfloor \frac{W}{16} \rfloor$, and $H' = \lfloor \frac{H}{16} \rfloor$. The output of this network encodes the appearance of the image at a set of uniformly sampled image locations, and forms the input to the localization layer.

3.1.2 Fully Convolutional Localization Layer

The localization layer receives an input tensor of activations, identifies spatial regions of interest and smoothly extracts a fixed-sized representation from each region. Our approach is based on that of Faster R-CNN [38], but we replace their RoI pooling mechanism [13] with bilinear interpolation [19], allowing our model to propagate gradients backward through the coordinates of predicted regions. This modification opens up the possibility of predicting affine or morphed region proposals instead of bounding boxes [19], but we leave these extensions to future work.

Inputs/outputs. The localization layer accepts a tensor of activations of size $C \times W' \times H'$. It then internally selects B regions of interest and returns three output tensors giving information about these regions:

1. **Region Coordinates**: A matrix of shape $B \times 4$ giving bounding box coordinates for each output region.
2. **Region Scores**: A vector of length B giving a confidence score for each output region. Regions with high confidence scores are more likely to correspond to ground-truth regions of interest.
3. **Region Features**: A tensor of shape $B \times C \times X \times Y$ giving features for output regions; is represented by an $X \times Y$ grid of C-dimensional features.

Convolutional Anchors. Similar to Faster R-CNN [38], our localization layer predicts region proposals by regressing offsets from a set of translation-invariant anchors. In particular, we project each point in the $W' \times H'$ grid of input features back into the $W \times H$ image plane, and consider k anchor boxes of different aspect ratios centered at this projected point. For each of these k anchor boxes, the localization layer predicts a confidence score and four

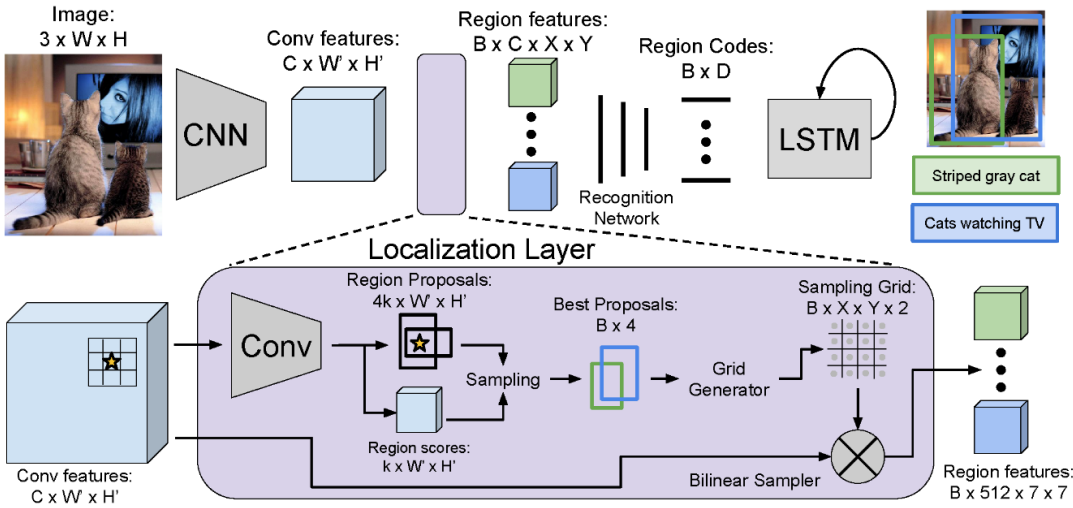


Figure 2. Model overview. An input image is first processed a CNN. The Localization Layer proposes regions and smoothly extracts a batch of corresponding activations using bilinear interpolation. These regions are processed with a fully-connected recognition network and described with an RNN language model. The model is trained end-to-end with gradient descent.

scalars regressing from the anchor to the predicted box coordinates. These are computed by passing the input feature map through a 3×3 convolution with 256 filters, a rectified linear nonlinearity, and a 1×1 convolution with $5k$ filters. This results in a tensor of shape $5k \times W' \times H'$ containing scores and offsets for all anchors.

Box Regression. We adopt the parameterization of [13] to regress from anchors to the region proposals. Given an anchor box with center (x_a, y_a), width w_a, and height h_a, our model predicts scalars (t_x, t_y, t_w, t_h) giving normalized offsets and log-space scaling transforms, so that the output region has center (x, y) and shape (w, h) given by

$$x = x_a + t_x w_a \qquad y = y_a + t_y h_a \tag{1}$$

$$w = w_a \exp(t_w) \qquad h = h_a \exp(h_w) \tag{2}$$

Note that the boxes are discouraged from drifting too far from their anchors due to L2 regularization.

Box Sampling. Processing a typical image of size $W = 720, H = 540$ with $k = 12$ anchor boxes gives rise to 17,280 region proposals. Since running the recognition network and the language model for all proposals would be prohibitively expensive, it is necessary to subsample them.

At training time, we follow the approach of [38] and sample a minibatch containing $B = 256$ boxes with at most $B/2$ positive regions and the rest negatives. A region is positive if it has an intersection over union (IoU) of at least 0.7 with some ground-truth region; in addition, the predicted region of maximal IoU with each ground-truth region is positive. A region is negative if it has IoU < 0.3 with all ground-truth regions. Our sampled minibatch contains $B_P \leq B/2$ positive regions and $B_N = B - B_P$ negative regions, sampled uniformly without replacement from the set of all positive and all negative regions respectively.

At test time we subsample using greedy non-maximum suppression (NMS) based on the predicted proposal confidences to select the $B = 300$ most confident proposals.

The coordinates and confidences of the sampled proposals are collected into tensors of shape $B \times 4$ and B respectively, and are output from the localization layer.

Bilinear Interpolation. After sampling, we are left with region proposals of varying sizes and aspect ratios. In order to interface with the full-connected recognition network and the RNN language model, we must extract a fixed-size feature representation for each variably sized region proposal.

To solve this problem, Fast R-CNN [13] proposes an RoI pooling layer where each region proposal is projected onto the $W' \times H'$ grid of convolutional features and divided into a coarse $X \times Y$ grid aligned to pixel boundaries by rounding. Features are max-pooled within each grid cell, resulting in an $X \times Y$ grid of output features.

The RoI pooling layer is a function of two inputs: convolutional features and region proposal coordinates. Gradients can be propagated backward from the output features to the input features, but not to the input proposal coordinates. To overcome this limitation, we replace the RoI pooling layer with with bilinear interpolation [16, 19].

Concretely, given an input feature map U of shape $C \times W' \times H'$ and a region proposal, we interpolate the features of U to produce an output feature map V of shape $C \times X \times Y$. After projecting the region proposal onto U we follow [19] and compute a *sampling grid* G of shape $X \times Y \times 2$ associating each element of V with real-valued coordinates into U. If $G_{i,j} = (x_{i,j}, y_{i,j})$ then $V_{c,i,j}$ should be equal to U

at $(c, x_{i,j}, y_{i,j})$; however since $(x_{i,j}, y_{i,j})$ are real-valued, we convolve with a sampling kernel k and set

$$V_{c,i,j} = \sum_{i'=1}^{W} \sum_{j'=1}^{H} U_{c,i',j'} k(i' - x_{i,j}) k(j' - y_{i,j}). \quad (3)$$

We use bilinear sampling, corresponding to the kernel $k(d) = \max(0, 1 - |d|)$. The sampling grid is a linear function of the proposal coordinates, so gradients can be propagated backward into predicted region proposal coordinates. Running bilinear interpolation to extract features for all sampled regions gives a tensor of shape $B \times C \times X \times Y$, forming the final output from the localization layer.

3.1.3 Recognition Network

The recognition network is a fully-connected neural network that processes region features from the localization layer. The features from each region are flattened into a vector and passed through two full-connected layers, each using rectified linear units and regularized using Dropout. For each region this produces a code of dimension $D = 4096$ that compactly encodes its visual appearance. The codes for all positive regions are collected into a matrix of shape $B \times D$ and passed to the RNN language model.

In addition, we allow the recognition network one more chance to refine the confidence and position of each proposal region. It outputs a final scalar confidence of each proposed region and four scalars encoding a final spatial offset to be applied to the region proposal. These two outputs are computed as a linear transform from the D-dimensional code for each region. The final box regression uses the same parameterization as Section 3.1.2.

3.1.4 RNN Language Model

Following previous work [32, 21, 49, 8, 4], we use the region codes to condition an RNN language model [15, 33, 44]. Concretely, given a training sequence of tokens $s_1, \ldots, s_T$, we feed the RNN $T + 2$ word vectors $x_{-1}, x_0, x_1, \ldots, x_T$, where $x_{-1} = \text{CNN}(I)$ is the region code encoded with a linear layer and followed by a ReLU non-linearity, x_0 corresponds to a special START token, and x_t encode each of the tokens s_t, $t = 1, \ldots, T$. The RNN computes a sequence of hidden states h_t and output vectors y_t using a recurrence formula $h_t, y_t = f(h_{t-1}, x_t)$ (we use the LSTM [18] recurrence). The vectors y_t have size $|V|+1$ where V is the token vocabulary, and where the additional one is for a special END token. The loss function on the vectors y_t is the average cross entropy, where the targets at times $t = 0, \ldots, T-1$ are the token indices for s_{t+1}, and the target at $t = T$ is the END token. The vector y_{-1} is ignored. Our tokens and hidden layers have size 512.

At test time we feed the visual information x_{-1} to the RNN. At each time step we sample the most likely next token and feed it to the RNN in the next time step, repeating the process until the special END token is sampled.

3.2. Loss function

During training our ground truth consists of positive boxes and descriptions. Our model predicts positions and confidences of sampled regions twice: in the localization layer and again in the recognition network. We use binary logistic losses for the confidences trained on sampled positive and negative regions. For box regression, we use a smooth L1 loss in transform coordinate space similar to [38]. The fifth term in our loss function is a cross-entropy term at every time-step of the language model.

We normalize all loss functions by the batch size and sequence length in the RNN. We searched over an effective setting of the weights between these contributions and found that a reasonable setting is to use a weight of 0.1 for the first four criterions, and a weight of 1.0 for captioning.

3.3. Training and optimization

We train the full model end-to-end in a single step of optimization. We initialize the CNN with weights pretrained on ImageNet [39] and all other weights from a gaussian with standard deviation of 0.01. We use stochastic gradient descent with momentum 0.9 to train the weights of the convolutional network, and Adam [23] to train the other components of the model. We use a learning rate of 1×10^{-6} and set $\beta_1 = 0.9, \beta_2 = 0.99$. We begin fine-tuning the layers of the CNN after 1 epoch, and for efficiency we do not fine-tune the first four convolutional layers of the network.

Our training batches consist of a single image that has been resized so that the longer side has 720 pixels. Our implementation uses Torch 7 [6] and [36]. One mini-batch runs in approximately 300ms on a Titan X GPU and it takes about three days of training for the model to converge.

4. Experiments

Dataset. Existing datasets that relate images and natural language either only include full image captions [3, 52], or ground words of image captions in regions but do not provide individual region captions [35]. We perform our experiments using the Visual Genome (VG) region captions dataset [25] [1]. Our version contained 94,313 images and 4,100,413 snippets of text (43.5 per image), each grounded to a region of an image. Images were taken from the intersection of MS COCO and YFCC100M [47], and annotations were collected on Amazon Mechanical Turk by asking workers to draw a bounding box on the image and describe its content in text. Example captions from the dataset include "cats play with toys hanging from a perch", "newspapers are scattered across a table", "woman pouring wine into a glass", "mane of a zebra", and "red light".

[1] Dataset can be downloaded at `http://visualgenome.org/`.

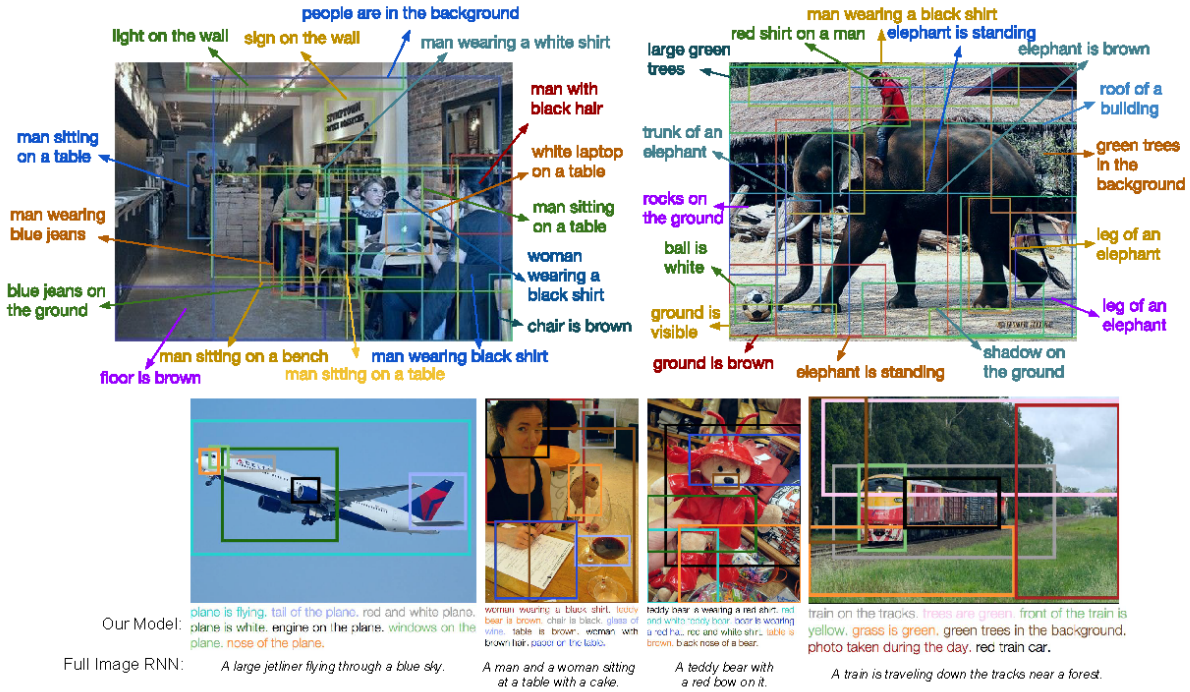


Figure 3. Example captions generated and localized by our model on test images. We render the top few most confident predictions. On the bottom row we additionally contrast the amount of information our model generates compared to the Full image RNN.

Preprocessing. We collapse words that appear less than 15 times into a special <UNK> token, giving a vocabulary of 10,497 words. We strip referring phrases such as "there is...", or "this seems to be a". For efficiency we discard all annotations with more than 10 words (7% of annotations). We also discard all images that have fewer than 20 or more than 50 annotations to reduce the variation in the number of regions per image. We are left with 87,398 images; we assign 5,000 each to val/test splits and the rest to train.

For test time evaluation we also preprocess the ground truth regions in the validation/test images by merging heavily overlapping boxes into single boxes with several reference captions. For each image we iteratively select the box with the highest number of overlapping boxes (based on IoU with threshold of 0.7), and merge these together (by taking the mean) into a single box with multiple reference captions. We then exclude this group and repeat the process.

4.1. Dense Captioning

In the dense captioning task the model receives a single image and produces a set of regions, each annotated with a confidence and a caption.

Evaluation metrics. Intuitively, we would like our model to produce both well-localized predictions (as in object detection) and accurate descriptions (as in image captioning).

Inspired by evaluation metrics in object detection [10, 30] and image captioning [48], we propose to measure the mean Average Precision (AP) across a range of thresholds for both localization and language accuracy. For localization we use intersection over union (IoU) thresholds .3, .4, .5, .6, .7. For language we use METEOR score thresholds 0, .05, .1, .15, .2, .25. We adopt METEOR since this metric was found to be most highly correlated with human judgments in settings with a low number of references [48]. We measure the average precision across all pairwise settings of these thresholds and report the mean AP.

To isolate the accuracy of language in the predicted captions without localization we also merge ground truth captions across each test image into a bag of references sentences and evaluate predicted captions with respect to these references without taking into account their spatial position.

Baseline models. Following Karpathy and Fei-Fei [21], we train only the Image Captioning model (excluding the localization layer) on individual, resized regions. We refer to this approach as a *Region RNN model*. To investigate the differences between captioning trained on full images or regions we also train the same model on full images and captions from MS COCO (*Full Image RNN model*).

At test time we consider three sources of region proposals. First, to establish an upper bound we evaluate the model on ground truth boxes (GT). Second, similar to [21] we use

Region source	Language (METEOR) EB	RPN	GT	Dense captioning (AP) EB	RPN	GT	Test runtime (ms) Proposals	CNN+Recog	RNN	Total
Full image RNN [21]	0.173	0.197	0.209	2.42	4.27	*14.11*	210ms	2950ms	**10ms**	3170ms
Region RNN [21]	0.221	0.244	0.272	1.07	4.26	*21.90*	210ms	2950ms	**10ms**	3170ms
FCLN on EB [13]	**0.264**	**0.296**	0.293	4.88	3.21	*26.84*	210ms	**140ms**	**10ms**	360ms
Our model (FCLN)	**0.264**	0.273	**0.305**	**5.24**	**5.39**	***27.03***	**90ms**	**140ms**	**10ms**	**240ms**

Table 1. Dense captioning evaluation on the test set of 5,000 images. The language metric is METEOR (high is good), our dense captioning metric is Average Precision (AP, high is good), and the test runtime performance for a 720×600 image with 300 proposals is given in milliseconds on a Titan X GPU (ms, low is good). EB, RPN, and GT correspond to EdgeBoxes [54], Region Proposal Network [38], and ground truth boxes respectively, used at test time. Numbers in GT columns (italic) serve as upper bounds assuming perfect localization.

an external region proposal method to extract 300 boxes for each test image. We use EdgeBoxes [54] (EB) due to their strong performance and speed. Finally, EdgeBoxes have been tuned to obtain high recall for objects, but our regions data contains a wide variety of annotations around groups of objects, stuff, etc. Therefore, as a third source of test time regions we follow Faster R-CNN [38] and train a separate Region Proposal Network (RPN) on the VG regions data. This corresponds to training our full model except without the RNN language model.

As the last baseline we reproduce the approach of Fast R-CNN [13], where the region proposals during training are fixed to EdgeBoxes instead of being predicted by the model (*FCLN on EB*). The results of this experiment can be found in Table 1. We now highlight the main takeaways.

Discrepancy between region and image level statistics. Focusing on the first two rows of Table 1, the Region RNN model obtains consistently stronger results on METEOR alone, supporting the difference in the language statistics present on the level of regions and images. Note that these models were trained on nearly the same images, but one on full image captions and the other on region captions. However, despite the differences in the language, the two models reach comparable performance on the final metric.

RPN outperforms external region proposals. In all cases we obtain performance improvements when using the RPN network instead of EB regions. The only exception is the FCLN model that was only trained on EB boxes. Our hypothesis is that this reflects people's tendency of annotating regions more general than those containing objects. The RPN network can learn these distributions from the raw data, while the EdgeBoxes method was designed for high recall on objects. In particular, note that this also allows our model (FCLN) to outperform the FCLN on EB baseline, which is constrained to EdgeBoxes during training (5.24 vs. 4.88 and 5.39 vs. 3.21). This is despite the fact that their localization-independent language scores are comparable, which suggests that our model achieves improvements specifically due to better localization. Finally, the noticeable drop in performance of the FCLN on EB model when evaluating on RPN boxes (5.39 down to 3.21) also suggests that the EB boxes have particular visual statistics, and that the RPN boxes are likely out of sample for the FCLN on EB model.

Our model outperforms individual region description. Our final model performance is listed under the RPN column as 5.39 AP. In particular, note that in this one cell of Table 1 we report the performance of our full joint model instead of our model evaluated on the boxes from the independently trained RPN network. Our performance is quite a bit higher than that of the Region RNN model, even when the region model is evaluated on the RPN proposals (5.93 vs. 4.26). We attribute this improvement to the fact that our model can take advantage of visual information from the context outside of the test regions.

Qualitative results. We show example predictions of the dense captioning model in Figure 3. The model generates rich snippet descriptions of regions and accurately grounds the captions in the images. For instance, note that several parts of the elephant are correctly grounded and described ("trunk of an elephant", "elephant is standing", and both "leg of an elephant"). The same is true for the airplane example, where the tail, engine, nose and windows are correctly localized. Common failure cases include repeated detections (e.g. the elephant is described as standing twice).

Runtime evaluation. Our model is efficient at test time: a 720×600 image is processed in 240ms. This includes running the CNN, computing $B = 300$ region proposals, and sampling from the language model for each region.

Table 1 (right) compares the test-time runtime performance of our model with baselines that rely on EdgeBoxes. Regions RNN is slowest since it processes each region with an independent forward pass of the CNN; with a runtime of 3170ms it is more than $13\times$ slower than our method.

FCLN on EB extracts features for all regions after a single forward pass of the CNN. Its runtime is dominated by EdgeBoxes, and it is $\approx 1.5\times$ slower than our method.

Our method takes 88ms to compute region proposals, of which nearly 80ms is spent running NMS to subsample regions in the Localization Layer. This time can be drastically reduced by using fewer proposals: using 100 region proposals reduces our total runtime to 166ms.

	Ranking				Localization			
	R@1	R@5	R@10	Med. rank	IoU@0.1	IoU@0.3	IoU@0.5	Med. IoU
Full Image RNN [21]	0.10	0.30	0.43	13	-	-	-	-
EB + Full Image RNN [21]	0.11	0.40	0.55	9	0.348	0.156	0.053	0.020
Region RNN [21]	0.18	0.43	0.59	7	0.460	0.273	0.108	0.077
Our model (FCLN)	**0.27**	**0.53**	**0.67**	**5**	**0.560**	**0.345**	**0.153**	**0.137**

Table 2. Results for image retrieval experiments. We evaluate ranking using recall at k (R@K, higher is better) and median rank of the target image (Med.rank, lower is better). We evaluate localization using ground-truth region recall at different IoU thresholds (IoU@t, higher is better) and median IoU (Med. IoU, higher is better). Our method outperforms baselines at both ranking and localization.

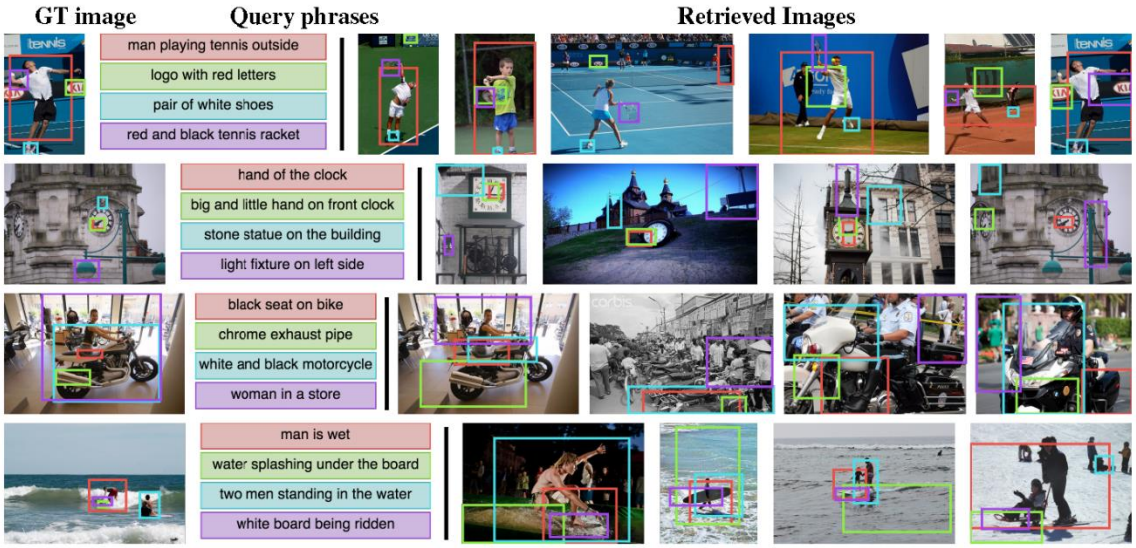


Figure 4. Example image retrieval results using our dense captioning model. From left to right, each row shows a ground-truth test image, ground-truth region captions describing the image, and the top images retrieved by our model using the text of the captions as a query. Our model is able to correctly retrieve and localize people, animals, and parts of both natural and man-made objects.

4.2. Image Retrieval using Regions and Captions

In addition to generating novel descriptions, our dense captioning model can support image retrieval using natural-language queries, and can localize these queries in retrieved images. We evaluate our model's ability to correctly retrieve images and accurately localize textual queries.

Experiment setup. We use 1000 random images from the VG test set for this experiment. We generate 100 test queries by repeatedly sampling four random captions from some image and then expect the model to correct retrieve the source image for each query.

Evaluation. To evaluate ranking, we report the fraction of queries for which the correct source image appears in the top k positions for $k \in \{1, 5, 10\}$ (recall at k) and the median rank of the correct image across all queries.

To evaluate localization, for each query caption we examine the image and ground-truth bounding box from which the caption was sampled. We compute IoU between this ground-truth box and the model's predicted grounding for the caption. We then report the fraction of query caption for which this overlap is greater than a threshold t for $t \in \{0.1, 0.3, 0.5\}$ (recall at t) and the median IoU across all query captions.

Models. We compare the ranking and localization performance of full model with baseline models from Section 4.1.

For the Full Image RNN model trained on MS COCO, we compute the probability of generating each query caption from the entire image and rank test images by mean probability across query captions. Since this does not localize captions we only evaluate its ranking performance.

The Full Image RNN and Region RNN methods are trained on full MS COCO images and ground-truth VG regions respectively. In either case, for each query and test image we generate 100 region proposals using EdgeBoxes and for each query caption and region proposal we compute the probability of generating the query caption from the region. Query captions are aligned to the proposal of maximal probability, and images are ranked by the mean probability

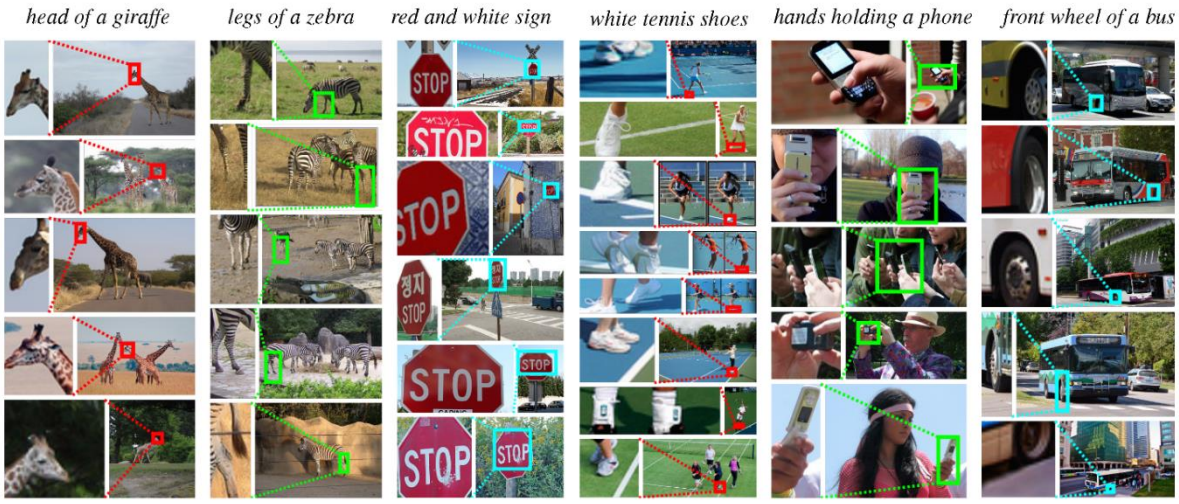


Figure 5. Example results for open world detection. We use our dense captioning model to localize arbitrary pieces of text in images, and display the top detections on the test set for several queries.

of aligned caption / region pairs.

The process for the full FCLN model is similar, but uses the top 100 proposals from the localization layer rather than EdgeBoxes proposals.

Discussion. Figure 4 shows examples of ground-truth images, query phrases describing those images, and images retrieved from these queries using our model. Our model is able to localize small objects ("hand of the clock", "logo with red letters"), object parts, ("black seat on bike", "chrome exhaust pipe"), people ("man is wet") and some actions ("man playing tennis outside").

Quantitative results comparing our model against the baseline methods is shown in Table 2. The relatively poor performance of the Full Image RNN model (Med. rank 13 vs. 9,7,5) may be due to mismatched statistics between its train and test distributions: the model was trained on full images, but in this experiment it must match region-level captions to whole images (Full Image RNN) or process image regions rather than full images (EB + Full Image RNN).

The Region RNN model does not suffer from a mismatch between train and test data, and outperforms the Full Image RNN model on both ranking and localization. Compared to Full Image RNN, it reduces the median rank from 9 to 7 and improves localization recall at 0.5 IoU from 0.053 to 0.108.

Our model outperforms the Region RNN baseline for both ranking and localization under all metrics, further reducing the median rank from 7 to 5 and increasing localization recall at 0.5 IoU from 0.108 to 0.153.

The baseline uses EdgeBoxes which was tuned to localize objects, but not all query phrases refer to objects. Our model achieves superior results since it learns to propose regions from the training data.

Open-world Object Detection Using the retrieval setup described above, our dense captioning model can also be used to localize arbitrary pieces of text in images. This enables "open-world" object detection, where instead of committing to a fixed set of object classes at training time we can specify object classes using natural language at test-time. We show example results for this task in Figure 5, where we display the top detections on the test set for several phrases.

Our model can detect animal parts ("head of a giraffe", "legs of a zebra") and also understands some object attributes ("red and white sign", "white tennis shoes") and interactions between objects ("hands holding a phone"). The phrase "front wheel of a bus" is a failure case: the model correctly identifies wheels of buses, but cannot distinguish between the front and back wheel.

5. Conclusion

We introduced the dense captioning task, which requires a model to simultaneously localize and describe regions of an image. To address this task we developed the FCLN architecture, which supports end-to-end training and efficient test-time performance. Our FCLN architecture is based on recent CNN-RNN models developed for image captioning but includes a novel, differentiable localization layer that can be inserted into any neural network to enable spatially-localized predictions. Our experiments in both generation and retrieval settings demonstrate the power and efficiency of our model with respect to baselines related tp previous work, and qualitative experiments show visually pleasing results. In future work we would like to relax the assumption of rectangular proposal regions and to discard test-time NMS in favor of a trainable spatial suppression layer.

6. Acknowledgments

Our work is partially funded by an ONR MURI grant and an Intel research grant. We thank Vignesh Ramanathan, Yuke Zhu, Ranjay Krishna, and Joseph Lim for helpful comments and discussion. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

References

[1] K. Barnard, P. Duygulu, D. Forsyth, N. De Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. *JMLR*, 2003. 2

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003. 2

[3] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollar, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 1, 2, 4

[4] X. Chen and C. L. Zitnick. Mind's eye: A recurrent visual representation for image caption generation. *CVPR*, 2015. 1, 2, 4

[5] K. Cho, A. C. Courville, and Y. Bengio. Describing multimedia content using attention-based encoder-decoder networks. *CoRR*, abs/1507.01053, 2015. 2

[6] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011. 4

[7] M. Denkowski and A. Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014. 2

[8] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv preprint arXiv:1411.4389*, 2014. 1, 2, 4

[9] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. *CVPR*, 2014. 2

[10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 5

[11] H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. Platt, et al. From captions to visual concepts and back. *CVPR*, 2015. 2

[12] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: Generating sentences from images. *ECCV*, 2010. 2

[13] R. Girshick. Fast R-CNN. *ICCV*, 2015. 2, 3, 6

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014. 1, 2

[15] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 2, 4

[16] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. DRAW: A recurrent neural network for image generation. *ICML*, 2015. 1, 2, 3

[17] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2015*, 2015. 2

[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2, 4

[19] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. *NIPS*, 2015. 1, 2, 3

[20] Y. Jia, M. Salzmann, and T. Darrell. Learning cross-modality similarity for multinomial data. *ICCV*, 2011. 2

[21] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *CVPR*, 2015. 1, 2, 4, 5, 6, 7

[22] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015. 2

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 4

[24] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *TACL*, 2015. 2

[25] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016. 4

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[27] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. Baby talk: Understanding and generating simple image descriptions. *CVPR*, 2011. 2

[28] P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi. Generalizing image captions for image-text parallel corpus. In *ACL (2)*, pages 790–796. Citeseer, 2013. 2

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. *ECCV*, 2014. 5

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015. 2

[32] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. Explain images with multimodal recurrent neural networks. *arXiv preprint arXiv:1410.1090*, 2014. 1, 2, 4

[33] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010. 2, 4

[34] V. Ordonez, X. Han, P. Kuznetsova, G. Kulkarni, M. Mitchell, K. Yamaguchi, K. Stratos, A. Goyal, J. Dodge, A. Mensch, et al. Large scale retrieval and generation of image descriptions. *International Journal of Computer Vision (IJCV)*, 2015. 2

[35] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *ICCV*, 2015. 4

[36] qassemoquab. stnbhwd. https://github.com/qassemoquab/stnbhwd, 2015. 4

[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015. 2

[38] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *NIPS*, 2015. 1, 2, 3, 4, 6

[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, April 2015. 1, 2, 4

[40] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. *ICLR*, 2014. 1, 2

[41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 2

[42] R. Socher and L. Fei-Fei. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. *CVPR*, 2010. 2

[43] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2014. 2

[44] I. Sutskever, J. Martens, and G. E. Hinton. Generating text with recurrent neural networks. *ICML*, 2011. 2, 4

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CVPR*, 2015. 1

[46] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*, 2014. 1, 2

[47] B. Thomee, B. Elizalde, D. A. Shamma, K. Ni, G. Friedland, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 4

[48] R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. *CVPR*, 2015. 2, 5

[49] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. *CVPR*, 2015. 1, 2, 4

[50] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988. 2

[51] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *ICML*, 2015. 1, 2

[52] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2014. 4

[53] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *ECCV*, 2014. 1

[54] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. *ECCV*, 2014. 6

翻译译文

DenseCap: 针对密集描述的全卷积定位网络

Justin Johnson*0, Andrej Karpathy*0, Li Fei-Fei
斯坦福大学，计算机科学系
{jcjohns, karpathy, feifeili}@cs.stanford.edu

摘要

我们引入了密集描述问题，其需要一个计算机视觉系统对图像中蕴含显著信息的多区域进行定位并使用自然语言来描述。密集描述问题包含对象检测*1 和图像描述，以往的对象检测只对检测到的对象生成一个单词，而图像描述则只针对完整的一幅图像生成描述。为了让定位和描述任务同时进行，我们提出了一种全卷积定位网络（Fully Convolutional Localization Network, FCLN）架构，它不需要额外的区域提议仅通过一个单一且有效的前向传播就能完成对一幅图像的处理，并通过单一的优化来端到端地训练。这种网络构架由卷积神经网络，一种全新的稠密定位层，和用来生成标记序列的循环神经网络语言模型共同构成。我们在包含 94,000 张图像，4,100,000 段标定区域上描述的视觉基因组数据集（Visual Genome dataset）上评价了我们的网络模型。通过对比发现，我们的模型无论是在描述生成还是图像检索，速度还是精确度，都比作为基准模型的目前最好的模型性能和结果还要优异。

1. 绪论

得益于人类对于视觉场景强大的语义理解能力，对于一幅图像我们可以轻而易举地指出和描述上面所有元素的各个方面。人们不知不觉或多或少都发挥了这种天赋，然而这项能力对于即使是目前最先进的计算机视觉识别系统而言也是一项挑战。在过去的几年中，图像分类问题取得了重大进展[39,26,53,45]，成功为每幅图像都分配了一个标记。随后的工作朝着两个垂直的方向快速推进：其一，对象检测进展迅猛，[40,14,46]的模型已能够有效地识别和标记图像中的多个显著区域；其二，[3,32,21,49,51,8,4]中图像描述问题的最新进展表明，已将标记空间从固定的类别集合扩展为能够表达更丰富含义的词汇序列。

虽然在标记的密集程度和复杂程度这两个方面进展令人振奋，但是它们是分立的。本文的工作向前迈进了一步，将这两个相互关联的方向整合到一个共同的构架之中。首先，我们引入了密集描述问题（如图 1），其需要一个模型来预测一幅图像不同的区域来产生一组描述。

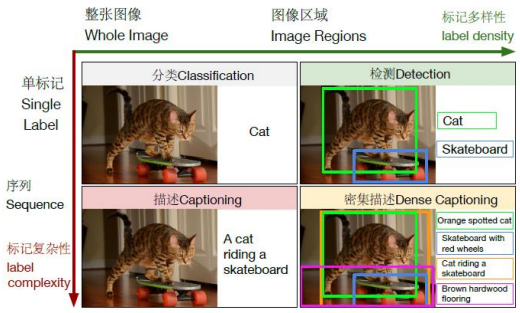


图 1 我们提出使用一个模型来解决密集描述问题（右下），通过一个单一的流程就能对图像生成兼具多样性和复杂性的注释。

当所有的目标（target）标记中一个词时，对象检测因而被恢复到一个特殊的状态。而当所有的图像区域中有一个延展（span）为完整图像大小的区域时，图像描述被恢复为一个特殊的状态。

此外，我们为密集描述问题创建了全卷积定位网络（Fully Convolutional Localization Network, FCLN）。我们的模型受到近期关于图像描述研究[49,21,32,8,4]中由卷积神经网络和循环神经网络语言模型组合和而成模型的启发。但就对象检测上的工作[38]，我们的第二个最重要的贡献是引入了一个稠密定位层。就稠密定位层整体，它完全可分离的，可以被嵌入到任何处理图像的神经网络中，以实现区域这一级上的训练和预测。就定位层内部，其首先来预测图像的一系列区域，然后使用双线性插值[19,16]平滑地裁剪每个区域的激活。

我们的模型在大规模视觉基因组数据集（large-scalle Visual Genome dataset）上进行评估，数据集包含 94,000 个图像和 4,100,000 个区域描述。结果表明和先前最好水平相比，我们模型无论在测试性能还是速度都更胜一筹。

我们公开我们的代码和数据，以支持密集描述任务的取得更进一步的发展。

2. 相关工作

我们的工作借鉴了近期在对象检测，图像描述，和软空间注意力（soft spatial attention）方面的工作，从而使得后续（downstream）处理图像特定区域成为可能。

对象检测[*1]（Object Detection）

我们的核心视觉处理模块是一个卷积神经网络（CNN）[29,26]，由于模型功能的强大，其俨然已经成为视觉识别问题中宠儿 [39]。R-CNN[14]首次被运用到密集预测问题上，其对于特定图像不同区域的处理是分别进行的。后续的研究集中解决如何让 CNN 在单一的前向传播来处理所有的边界[17,13]以及通过直接预测要么在图像坐标系中要么在全卷积中的边界方框来消除显式的求导，属于位置无关的设置[40,38,37]。[38]Ren 等人的研究工作和我们的方法最相关，其创建了一个区域提议网络（region proposal network, RPN），从锚定点回归到感兴趣的区域。但是，他们采用 4 步优化流程，而我们的方法不需要其中训练的流程。另外，我们用一种不同的空间软注意力机制[19,16]替代

他们的 RoI 池机制。特别是，这一改进使我们能够在区域提议网络进行反向传播并训练整个模型。

图像描述（Image Captioning）

针对用自然语言描述图像的问题，[1,27,12,34,42,43,28,20]提出了几种具有开创性的方法。更新的基于神经网络的方法已经采用循环神经网络（RNNs）[50,18]作为生成描述的核心构件。这些模型早先被用于语言建模[2,15,33,44]，它们以能很好地学习到长期交互而著称[22]。几种近期的图像描述方法[32,21,49,8,4,24,11]依赖于在图像信息条件下的 RNN 语言模型和可能软注意力机制[51,5]的组合。与我们的工作类似，Karpathy 和 Fei-Fei Li[21]在区域上运行图像描述模型，但他们没有处理在一个模型中同时检测和描述这个共同问题。我们的模型是端到端的，并且以这样的方式设计，即每个区域的预测都是全局图像上下文的函数，在后文可以看到这样会带来更好的性能和表现。最后，受[48,7,3]图像描述问题中指标的启发，我们为密集描述问题创建了评价指标。

3. 模型

概览

我们的目标是设计一个对处理图像每个有显著信息的区域进行定位同时用自然语言进行描述的体系结构。面临的首要挑战是开发支持通过单一步骤优化的端到端训练的模型，并要兼具训练和预测的高效性。我们提出的架构（参见图 2）借鉴了近期有关对象检测，图像描述和软空间注意力的工作中的构件，这些构建均满足自身的约束条件。

在第 3.1 节中。我们首先介绍我们模型的组成部分。然后在第 3.2 和 3.3 节，我们着重介绍损失函数，以及训练和预测中的细节。

3.1 模型架构

3.1.1 卷积网络

我们使用 VGG-16 架构[41]，因为其在[39]中代表当前的最好水平。VGG-16 由 13 层 3×3 卷积层和 5 个 2×2 最大汇聚层[*5] 组合而成。我们去掉了了最后的汇聚层，因此输入一张图像所表示为的$3 \times W \times H$的张量，处理得到一个$C \times W' \times H'$的特征张量，其中$\mathrm{C} = 512$，$W' = \left[\frac{W}{16}\right]$和$H' = \left[\frac{H}{16}\right]$。网络在图像上一组均匀的位置进行采样并被编码输出，来作为到定位层的输入。

3.1.2 全卷积定位层

定位层接收激活值形成的输入张量，确定感兴趣的空间区域并平滑地从每个区域提取固定大小的表示。我们的方法是基于 Faster R-CNN [38]的工作，不同之处在于我们用双线性插值[19]取代了它们的 RoI 汇聚机制[13]，从而使得我们的模型能通过预测区域的坐标反向传播梯度。这种修改意味着预测相互杂糅抑或是边界不固定区域而不是使用方框状区域的可

能性[19]，我们将在未来对这方面进行延伸。

输入/输出

定位层接受大小为$C \times W' \times H'$的激活张量。然后它在内部选择感兴趣的 B 区域并返回三个输出张量，从而给出关于这些区域的信息：

1.区域坐标：大小为 B×4 的矩阵，给出每个输出区域的边界方框坐标。

2.区域得分：长度为 B 的向量，给出每个输出区域的置信度分数。具有高置信度分数的区域更可能对应于图像的真相（ground truth）区域。

3.区域特征：形状$B \times C \times X \times Y$的张量，给出输出区域的特征;表示为 C 维$X \times Y$网格特征。

卷积锚定点

与 Fast R-CNN 类似[38]，我们的定位层通过回归来自一组平移不变锚定点的偏移来预测区域提议。特别是，我们将输入特征的$W' \times H$网格中的每个点投影回$W \times H$图像平面，并且以这个投影点为中心的不同长宽比的k个锚定方框[*2]（anchor box）。对于这些k个锚定方框中的每个，定位层都会预测一个置信度分数和四个从锚定点回归到预测方框坐标间的标量，这些是通过对输入特征图使用 256 个滤波器[*6]的 3×3 卷积进而通过校正的线性非线性激活函数[*4]和 5 千个滤波器的 1×1 卷积计算得到的。结果是规格为$5k \times W' \times H'$的张量，其中包含所有锚定点的分数和偏移量。

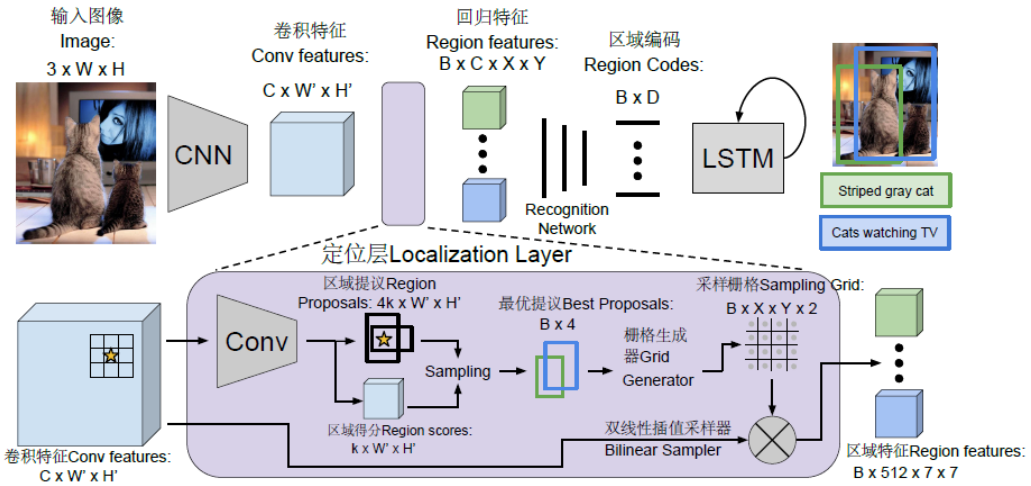


图 2 模型概览。输入图像首先经过 CNN 的处理。定位层来处理边界并使用双线性插值来平滑地提取一个批次的对应激活值。这些区域再经过全卷积定位网络的处理和一个 RNN 语言模型来生成描述。整个模型通过梯度下降来端到端的训练。

方框[*2]回归

我们采用[13]的参数化方法，从锚定点向提议区域回归。给定一个中心为(x_a, y_a)，宽度w_a和高度h_a的锚方框，给定归一化偏移和对数空间缩放变换，模型来预测标量(t_x, t_y, t_w, t_h)，设输出区域中心(x, y)和形状(w, h)，则有：

$$x = x_a + t_x w_a \quad y = y_a + t_y h_a \quad (1)$$

$$w = w_a exp(t_w) \quad h = h_a exp(h_w) (2)$$

注意，由于 L2 正则化项，可以防止方框漂离锚定点太远。

方框采样

处理一个典型的图像，$W = 720$，$H = 540$，有$k = 12$个锚定方框，进而产生17,280个区域提议。由于为所有提议运行识别网络和语言模型的成本过高，因此有必要对它们进行子采样。

在训练时，我们遵照[38]的方法并采样$B = 256$个方框的小批（minibatch），其中至多$B/2$个正区域其余为负区域。如果一个区域和某个 ground truth 区域重合相交（intersection over unoin, IoU）至少达到0.7就认为该区域为正的。相反，一个区域和所有的 ground truth 区域的$IoU < 0.3$，就认为该区域是负的。我们的采样的 minibatch 样本包含$BP \leq B/2$个正区域和$BN = B - BP$负片区域，要求均匀取样，不分别从全是正的区和全是负的片区域取代。

对于测试，我们基于预测的提议置信度使用贪婪非最大抑制（non-maximum, NMS）来选择$B = 300$个最可信的提议。

采样提议的坐标和置信度分别保存于到形状$B \times 4$和B的张量中，并作为定位层的输出。

双线性插值

采样后，我们留下了不同尺寸和长宽比的区域提议。为了与全连接的识别网络和 RNN 语言模型进行交互，我们必须为每个大小可变的区域提议提取一个固定大小的特征向量表示。

为了解决这个问题，Fast R-CNN [13]提出了一个 RoI 汇聚[*5]层，其中每个区域提议被投影到卷积特征的$W' \times H'$网格上，并且被划分成与像素边界对齐粗的$X \times Y$网格 。特征在每个网格单元内是最大汇聚的，从而输出特征的$X \times Y$网格。

RoI 汇聚层是一个两输入的函数，输入分别是卷积特征和区域提议坐标。梯度可以从输出特征反向传播到输入特征，但是无法反向传播到输入提议坐标。为了克服其这个局限性，我们用双线性插值替换了 RoI 汇聚层[16,19]。

具体而言，给定形状$C \times W' \times H'$的输入特征图U和一个区域提议，我们内插 U 的特征以产生形状$C \times X \times Y$的输出特征图V。在将区域提议投影到U之后，我们遵循[19]方法来计算形状$X \times Y \times 2$的采样网格 G，将 V 的每个元素与实值坐标关联到 U。如果$G_{i,j} = (x_{i,j}, y_{i,j})$，那么$V_{c,i,j}$应该等于在

$(c, x_{i,j}, y_i)$处的U。然而，由于$(x_{i,j}, y_{i,j})$是实值的，因此我们将采样核 k 与 set 进行卷积

$$V_{c,i,j} = \sum_{i'=1}^{W} \sum_{j'=1}^{H} U_{c,i',j'} k(i' - x_{i,j}) k(j' - y_{i,j}) \quad (3)$$

我们使用双线性采样，对应的核$k(d) + max(0, 1 - |d|)$。采样网格是一个关于提议坐标的线性函数，因此可以将梯度向后传播到预测区域提议坐标。运行双线性插值为所有采样区域提取特征，形成张量形状为$B \times C \times X \times Y$，从而作为定位层的最终输出。

3.1.3 识别网络

识别网络是一个全连接的神经网络，处理来自定位层的区域特征。将每个区域的特征展成一个向量并传入到两个全连接层，每个层都使用校正线性处理单元[*4]和 Dropout 正则化。对于每个区域，产生了维度$D = 4096$的编码，其紧凑地编码区域的视觉外观。将所有正区域的编码收集到形状为$B \times D$的矩阵中，并传递给 RNN 语言模型。

此外，我们允许识别网络再次提供一个改善每个提议区域的置信度和区域的机会。对每个区域提议的识别网络输出一个最终的标量置信度，以及四个来编码最终空间偏移量的标量。这两项输出是从对于每个区域 D 维编码线性变换计算得到的。最终的方框使用与第 3.1.2 节相同的参数化来回归。

3.1.4 RNN 语言模型

根据[32,21,49,8,4]所做的工作，我们使用区域编码来调节 RNN 语言模型[15,33,44]。具体而言，给定一个标记化的序列$s_1, \ldots, s_T$，传递给 RNN 的是长度为$T + 2$的词汇向量$x_{-1}, x_0, \ldots, x_T$，其中$x_{-1} = CNN(I)$是经过 CNN 处理的区域编码再通过一个线性层和紧接着的 ReLu 非线性层的处理结果，x_0对应于一个特殊的<START>标记，x_t对应于标记化后的序列中的s_t，其中$t = 1, \ldots, T$。RNN 计算一个隐藏状态h_t组成的序列，输出y_t组成的向量，计算方法为$y_t = f(h_{t-1}, x_t)$（我们使用的是长短期记忆网络 LSTM[18]的循环方式）。输出向量y_t，大小为$|V| + 1$，其中 V 表示标记化的词汇表，额外的这个 1 表示一个特殊的<END>标记。向量y_t上的损失函数使用的是平均交叉熵损失函数，其中在$t = 0, \ldots, T - 1$上的目标输出是s_{t+1}，在$t = T$上的目标输出是<END>标记。向量y_{-1}是缺省的。我们的标记和隐藏层的大小都是 512。

在测试时我们给予 RNN 以代表视觉信息的x_{-1}。在每个时间步，我们采样最有可能的下一个标记作为本时间步的输出，并在下一个时间步将其作为 RNN 的输入，重复该过程直到采样特殊的<END>标记。

3.2 损失函数

在训练期间我们的 ground truth 包括为正值的方框和描述。我们的模型预测了采样区域的位置和置信度两次：一次是在定位层，另一次在识别

网络中。我们对采样的正值和负值区域进行训练的置信度使用二元逻辑损失。对于方框回归，我们在变换坐标空间中使用类似于[38]平滑的 L1 损失。我们的损失函数中的第五项是语言模型的后续阶段的交叉熵项。

我们通过 RNN 中的批量大小和序列长度对所有损失函数进行归一化。我们尝试和找到了这些对结果有贡献项权重的有效设置，发现合理的设置是将前四个项权重设置为 0.1，用于描述的项权重设为 1.0。

3.3 训练和优化

我们通过单一流程的优化对整个模型进行端到端的训练。我们用 ImageNet [39]上预先训练的权重对 CNN 进行初始化，并且初始化的其他权重为设置为标准差为 0.01 的高斯分布。我们使用动量为 0.9 的随机梯度下降来训练卷积网络的权重，使用 Adam[23]来训练卷积网络的其他组件。我们使用1×10^{-6}的学习率。并设置$\beta_1 = 0.9$，$\beta_2 = 0.99$。在一轮的训练后，我们开始对 CNN 的层进行调试，为了提高效率，我们不调整网络的前四个卷积层。我们的训练批次包含一幅图像，经过了调整，更长的边有 720 个像素。我们的实现使用了 Torch7 [6]和[36]。一个小批次在 Titan X GPU 运行约为 300ms 在其上并对该模型进行为期约三天的训练达到收敛。

4. 实验

数据集

现有的图像和自然语言相关的数据集，要么只包括完整图像描述[3，52]，要么只有对于图像区域的简单词汇，不足以提供对于区域的描述[35]。我们在视觉基因组（VG）区域描述数据集[25][*8] 上开展了我们的实验。我们使用的版本包含 94,313 张图片和 4,100,413 片段文字（平均每张图像 43.5 段），每个描述都对应到图像的一个区域。图片来自 MS COCO 和 YFCC100M [47]数据集，在亚马逊众包平台由人工在图像上绘制边界方框并用文本描述其内容。来自数据集的示例描述如“cats play with toys hanging from a perch 猫玩悬挂在栖木上的玩具”，“新闻报纸散布在桌子上 newspapers are scattered across a table”，“ woman pouring wine into a glass 女人倒酒入玻璃杯“，“mane of a zebra 斑马的鬃毛”，“red light 红灯”等。

预处理

我们将出现次数少于 15 的词汇用一个特殊的<UNK>标记，得到一个有 10497 个词的词汇表。我们删除诸如“there is...”“this seems to be a”的引用短语。为了提高效率，我们丢弃超过 10 个单词的所有注释（占到总注释的 7%）。同时，我们还丢弃了所有少于 20 个或超过 50 个注释的图像，以减少每个图像区域数量的变化。剩下 87,398 张图片;作为验证和

测试集各分配到 5000 张，其余的用于训练。

为了测试评估，我们还通过将重叠的方框合并到具有多个参考描述的单个方框中对验证/ 测试中的 ground truth 区域进行预处理。对于每幅图像，我们迭代地选择具有最大数量的重叠方框的方框（基于阈值为 0.7 的 IoU），并将它们合并（通过取平均值）到具有多个参考描述的单个方框中。随后对着每组图像重复这个过程。

图 3.我们的模型在测试图像上生成的描述以及定位示例。我们返回最高置信度的前若干个预测。在底部我们增加了一些和完整图像 RNN 生成结果进行对比的信息。

4.1 密集描述

在密集描述问题中，模型会接收一幅图像并生成一组区域，每个区域都用一个置信度和一个描述进行注释。

评价指标

直观上，我们希望我们的模型既能产生良好的定位预测（如在对象检测中那样）又能进行准确的描述（如在图像描述中那样）。受对象检测[10,30]和图像描述[48]中的评估指标的启发，我们建议在定位和语言准确度评价上，将定位评价指标和语言查准率评价指标的各自的阈值范围二者交叉测量来平均查准率（AP）的平均值。

对于定位，我们使用交集（IoU）阈值为 0.3，0.4，0.5，0.6，0.7。对于语言，我们使用 METEOR 得分阈值为 0, 0.05, 0.1, 0.15, 0.2, 025。之所以采用 METEOR，是因为这个度量当用以作为参考的数量很少时被认为与人类的评判方式最为相关的 [48]。我们测量交叉组合这两种阈值设置的平均查准率（Average

Precesion），并得到平均查准率的平均值（mean AP）。

为了得出预测描述在关于语言生成上的精确度而不关心描述位置的所在，我们将每个测试图像上的 ground truth 描述整合为一个参考的包（bag, 之所以是包而不是集合 set，区别在于可以有重复元素，而集合中没有重复元素），从而得到了预测描述相对于参考的评价值，而不用考虑它们的空间位置。

基准模型

和 Karpathy 和 Fei-Fei Li 在[21]中类似，我们只在一个个相互独立，不定大小的区域上训练图像描述模型（不包括定位层）。我们把这种方法称之为区域 RNN 模型。为了分析在完整图像训练和在区域上进行对描述训练的不同之处，我们基于 MS COCO 数据集构建了相同的模型（完整图像 RNN 模型）对于完整图像和描述进行了的训练。

	Language (METEOR)			**Dense captioning** (AP)			**Test runtime** (ms)			
Region source	EB	RPN	GT	EB	RPN	GT	Proposals	CNN+Recog	RNN	Total
Full image RNN [21]	0.173	0.197	0.209	2.42	4.27	*14.11*	210ms	2950ms	**10ms**	3170ms
Region RNN [21]	0.221	0.244	0.272	1.07	4.26	*21.90*	210ms	2950ms	**10ms**	3170ms
FCLN on EB [13]	**0.264**	**0.296**	0.293	4.88	3.21	*26.84*	210ms	**140ms**	**10ms**	360ms
Our model (FCLN)	**0.264**	0.273	**0.305**	**5.24**	**5.39**	*27.03*	**90ms**	**140ms**	**10ms**	**240ms**

表 1.我们是在 5000 张图像上评价密集描述模型的。语言评价指标使用的是 METEOR(得分越高越好)，我们的密集描述评价指标是平均查准率（Average Precision，AP，越高越好），测试图像规格为 720x600，区域提议数量设定为 300 个，在 Titan X GPU 上处理结果单位为毫秒（ms, 越小越好）。EB, RPN, GT 分别代表 Edgeboxes[54], 区域提议网络 Region Proposal Network[38]和真相 ground truth 方框。GT 列下的斜体代表最佳位置上限。

在测试时，我们考虑三个区域提议的来源。首先，建立一个上界，对 ground truth 方框（GT）上的模型进行评估。其次，我们使用和[21]类似的额外区域提议的方法为每个测试图像提取 300 个方框。因其优异的性能和速度我们使用 EdgeBoxes[54]（EB）。最终 EdgeBoxes 已被调试到能对对于检测对象有较高的召回率，然而我们的区域数据中包含很多不同的注释，包括一堆各种的对象，等等，乱七八糟。因此，作为测试时区域的第三个来源，我们沿袭了 Faster R-CNN [38]的做法，在 VG 区域数据上训练了一个独立的区域提议网络（RPN）。和训练我们全部网络相同，只不过是没有 RNN 语言生成模型那部分。

作为最后一个基准对比模型，我们重现了 Fast R-CNN [13]模型，其中训练期间的区域提议被修改为 EdgeBoxes 而不是由模型（在 EB 上的 FCLN）来预测。实验结果如表 1。我们现在重点介绍我们主要的系列成果[*9]。

区域和图像级别统计之间的差异

观察表 1 的第一行,区域 RNN 模型在 METEOR 中获得了一致的更显

著的结果，表明区域和图像层面上语言统计数据的差异。注意，这些模型在几乎相同的图像上进行了训练，但其中一个模型是整幅图像描述，另一个模型是区域描述。不过，尽管语言模型有所不同，但这两种模式在最终指标上的表现相差无几。

RPN 优于外部区域提议

在所有情况下，当使用 RPN 网络而不是 EB 区域时，我们都会获得性能提升。唯一的例外出现在 FCLN 模型仅在 EB 方框上进行训练的情况。我们的假设是，这反映了人们对图像的区域进行注解的倾向超过了对于区域中包含物体进行注解的倾向。RPN 网络可以从原始数据中学习这些分布，而 EdgeBoxes 方法设计之初衷即在于检测对象上获得高召回率。尤其要注意的是，这也使得我们的模型（FCLN）优于在 EB 基准上的 FCLN，这是后者在训练期间（5.24 对比 4.88, 5.39 对比 3.21）仅限于 EdgeBoxes 造成的。尽管他们的与定位无关的得分和我们的结果势均力敌，但这表明我们的模型由于更好的定位从而获得了提升。最后值得注意的是，在 RPN 方框上评价指标显示，FCLN 在 EB 模型上的性能有明显下降，也表明 EB 方框具有特定的视觉统计数据，并且 RPN 框可对于 EB 模型上 FCLN 出现了超采样。

	Ranking				Localization			
	R@1	R@5	R@10	Med. rank	IoU@0.1	IoU@0.3	IoU@0.5	Med. IoU
Full Image RNN [21]	0.10	0.30	0.43	13	-	-	-	-
EB + Full Image RNN [21]	0.11	0.40	0.55	9	0.348	0.156	0.053	0.020
Region RNN [21]	0.18	0.43	0.59	7	0.460	0.273	0.108	0.077
Our model (FCLN)	**0.27**	**0.53**	**0.67**	**5**	**0.560**	**0.345**	**0.153**	**0.137**

图 2.图像检索实验结果

图 4.使用我们的密集描述模型得到的检索结果示例

我们的模型优于分别的区域描述

我们的最终模型性能如表 1 的 RPN 列下面所示，特别值得注意的是，其中 5.39 这一个单元格，我们的结果来源于我们完整的联合模型，而不是我们的为了评价而训练的独立的区域提议网络（RPN）。即使区域模型是在 RPN 提议中评价的，我们模型的表现也比区域 RNN 模型高很多（5.93 对比 4.26）。我们将这种改进归因于这样一个事实，即我们的模型可以利用测试区域之外的上下文中的视觉信息。

定性结果

图 3 中展示了对于密集描述我们模型的示例预测。该模型会准确地对区域生成丰富片段描述并进行标记。例如，如图 3 中，大象的几个部分正确接地并被描述（“trunk of an elephant 大象的鼻子”，“ elephant is standing 大象站着”，以及“leg of an elephant 大象的腿”）。对于示例中的飞机也是如此，例如尾翼（tail），引擎（engine,），雷达罩（nose）和舷窗（window）都被正确地标定和描述出来了。常见的失败情况包括重复检测（例如，大象被描述为站立两次）。

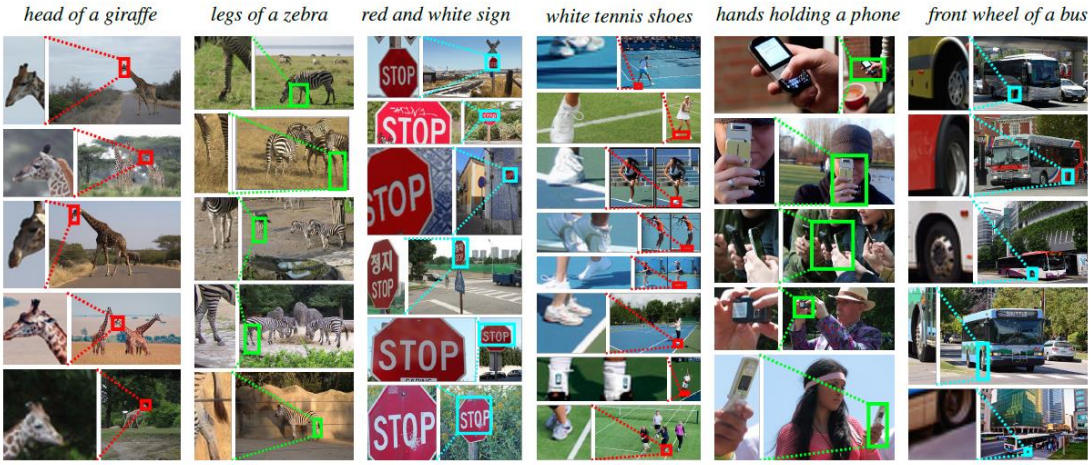


图 5“开放式”[*7]检测的结果示例

运行评价

我们的模型在测试的时间效率上很高：720×600 图像在 240 ms 内完成了处理。这包括运行 CNN，计算 B = 300 个区域提议，并从每个区域的语言模型中采样。

表 1（右）显示了我们模型的测试时运行时间性能和基于 Edge Box 的基准模型的比较结果。区域 RNN 是其中最慢的，因为它用 CNN 的在一个独立的前向传播来处理每个区域；运行时间为 3170ms，比我们的方法慢 13 倍以上。

在 CNN 的单向前向传播后，基于 EB 的 FCLN 提取所有区域的特征。它的运行时间受 EdgeBoxes 影响，并且它比我们的方法慢约 1.5 倍。

我们的方法需要 88ms 来计算区域提议，其中将近 80ms 用于运行

NMS 以在定位层中进行二次采样。通过减少提议数量可以将这个时间大大缩减：使用 100 个区域的提议将我们的总运行时间减少到 166ms。

4.2 使用区域和描述进行图像检索

除了生成新的描述之外，我们的密集描述模型支持使用自然语言查询来检索图像，并且可以将这些查询定位在检索的图像中。在正确检索图像，文本查询准确地定位问题上，我们的对模型进行了测评。

实验设计

我们使用来自 VG 测试集的 1000 个随机图像进行本实验。我们通过重复采样来自某个图像的四个随机描述生成 100 个测试查询，以期望该模型能够正确检索每个查询的源图像。

评价

为了评估检索得到的排名，我们显示正确的源图像出现在前 k, $k \in \{1,5,10\}$个位置的查询分数（即在 k 处得召回率）以及正确图像出现在所有查询的结果中的动态排名。

为了评价定位，对于每个查询描述，我们从得到的根据描述的采样结果中，检查图像和对应 ground truth 边界方框，计算该 ground truth 方框和模型预测的描述对应方框之间的 IoU。接着我们显示查询描述重叠超过阈值t，$t \in \{0.1,0.3,0.5\}$的分数（在 t 处的召回率）以及所有查询动态的 IoU。

模型

我们将整个模型得到检索排名结果和定位性能指标结果与第 4.1 节中的基准模型得到的结果进行比较。

对于在 MS COCO 上训练的完整图像 RNN 模型，我们计算从完整图像生成每个查询描述的概率，并按照查询描述中的平均概率对测试图像进行排名。由于不会定位描述，我们只评价其排名结果。

完整图像 RNN 和区域 RNN 方法分别在全部的 MS COCO 数据集图像和 VG 数据集 ground truth 区域上进行训练。在任何一种情况下，对于每个查询和测试图像，我们使用 EdgeBox 生成 100 个区域提议，并且为每个查询描述和区域提议计算从区域生成查询描述的概率。查询描述若和提议最大概率相符，则对应的一对区域-描述按照平均概率被排序。

完整的 FCLN 模型的处理过程和上述类似，但使用来自定位层前 100 个提议而不是 EdgeBoxes 提议。

讨论

图 4 显示了 ground truth 真实图像，描述这些图像的查询短语以及使用我们的模型从这些查询中检索到的图像的示例。

我们的模型能够定位小物体（“hand of the clock 时钟指针”，“ logo with red letters 带红色字母的标志”），物体某部分（“black seat on bike 自行车上的黑色座椅”，“ chrome exhaust pipe 镀铬排气管”），人（“man is wet 一个人身上打

湿了”）和一些行动（“man playing tennis outside 一个人在室外打网球”）。

表 2 显示了我们的模型与基准模型进行比较的定量结果。完整图像 RNN 模型的相对表现较差，可能是其训练和测试分布之间的统计数据不匹配造成的：该模型是在完整图像上进行训练的，但在测试实验中，它必须将区域级的描述与整个图像进行匹配（完整图像 RNN）或者对于图像的区域进行处理（EB+完整图像 RNN）而不是对完整图像进行处理。

区域 RNN 模型在训练和测试数据之间没有出现不匹配，在排名和定位方面优于完整图像 RNN 模型。和完整图像 RNN 相比，它将中位数从 9 减少到 7，并且提高了定位召回率在 IoU 为 0.5 的条件下，将结果从高 053 提升至了 0.108。

对于排名和定位，我们的模型在所有指标上都优于区域 RNN 基准模型，进一步将中位数从 7 减少到 5，并且将定位召回率在 IoU 为 0.5 的条件下，将结果从高 0.108 提升至了 0.153。

基准模型使用 EdgeBoxes，其被设定为检测物体对象，但是并非所有查询短语都对应于实际的对象。我们的模型通过学习从训练数据中得到提议区域，从而取得了优异的结果。

“开放式”对象检测[*7]

使用上述的检索设置，我们的密集描述模型也可以用于定位图像中任意文本片段。这使得“开放式（Open-World）”的对象检测成为可能，而不是在训练时提供一组固定的对象数据，在测试时使用自然语言描述上面这些指定的对象这样一个套路。图 5 显示了这个问题的处理结果，给出几个查询短语，得到在测试集上的认为最有可能的前几个结果。

我们的模型可以检测动物的部分（“head of a giraffe 长颈鹿的头部”，“ legs of a zebra 斑马的腿”），还可以了解一些物体的属性（“red and white sign 红色和白色标志”，“ white tennis shoes 白色网球鞋”）和物体之间的对比“hands holding a phone 拿着电话的手”）。短语“front wheel of a bus 巴士的前轮”是失败的情况：模型正确识别了车轮，但无法区分前轮和后轮。

5. 结论

我们引入了密集描述问题，其需要一个模型对图像的区域同时进行定位和描述。为了解决这个问题，我们创建了了 FCLN 架构，该架构支持端到端训练并且有良好的测试表现。我们的 FCLN 架构基于最近开发的用于图像描述的 CNN-RNN 模型，但包含一个全新的，完全可分离的定位层，其可内嵌到任何神经网络以实现局部空间的预测。我们在图像描述生成以及检索环境下的实验都证明了我们的模型与本文出现前的相关基准模型相比，无论在测试性能还是定性实验结

果上都显示出令人满意的结果。在未来的工作中，我们希望区域提议不局限于是矩形方框的假设，并放弃测试时使用 NMS，从而支持一个可训练的空间抑制层。

6. 致谢

感谢 ONR MURI 和 Intel Research 对我们部分工作的资助和支持。感谢 Vignesh Ramanathan，Yuke Zhu，Ranjay Krishna 和 Joseph Lim 提供的宝贵而又具建设性的点评和讨论意见。也非常感谢 NVIDIA 公司对此次研究所用 GPU 的支持。

译者按

*0 原文特别注明两位作者对于研究的贡献相当。

*1 由于 Object Detecton 中的 Object 直译为对象，而中文普遍翻译成目标检测，而目标的英文为 target，因此译者为了避免混淆，所以将前者翻译成对象检测。

*2 box 有的翻译成盒,有的翻译成箱，这里译者统一翻译成方框。

*4 即常见的 ReLu 函数。

*5 出于对层的实际作用考虑，将池化翻译为汇聚。

*6 用来卷积的方阵有很多种称呼，有的叫卷积核 kernel, 有的翻译为滤波器 filter，还有的地方称之为掩模 mask, 由于原文使用的是 filter 一词，所以在这里译者翻译为滤波器。

*7 这里译者将 Open-World 翻译为开放式，freestyle，意味着很任性地给你一张图像，来用上面模型进行测试。而不是在训练集上训练后，拿测试集图像来测试。

*8 原文中特别注明，数据集可以到 http://visualgenome.org/获取。

*9 原文中为 take-aways，直译为“外卖”，根据上下文翻译为“一系列的成果”。

参考文献

[1] K. Barnard, P. Duygulu, D. Forsyth, N. De Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. JMLR, 2003. 2
[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. The Journal of Machine Learning Research, 3:1137–1155, 2003. 2
[3] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollar, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325, 2015. 1, 2, 4
[4] X. Chen and C. L. Zitnick. Mind's eye: A recurrent visual representation for image caption generation. CVPR, 2015. 1, 2, 4
[5] K. Cho, A. C. Courville, and Y. Bengio. Describing multimedia content using attention-Based encoder-decoder networks. CoRR, abs/1507.01053, 2015. 2[6] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: Amatlab-like environment for machine learning. In BigLearn, NIPS Workshop, number EPFL-CONF-192376, 2011. 4
[7] M. Denkowski and A. Lavie. Meteor universal: Languagespecific translation evaluation for any target language. InProceedings of the EACL 2014 Workshop on Statistical MachineTranslation, 2014. 2
[8] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrentconvolutional networks for visual recognition and description.
arXiv preprint arXiv:1411.4389, 2014. 1, 2, 4
[9] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. CVPR, 2014. 2
[10] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes
(VOC) challenge. International journal of computer vision, 88(2):303–338, 2010. 5
[11] H. Fang, S. Gupta, F. Iandola, R. Srivastava, L. Deng, P. Doll´ar, J. Gao, X. He, M. Mitchell, J. Platt, et al. From captions to visual concepts and back. CVPR, 2015. 2
[12] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture
tells a story: Generating sentences from images. ECCV, 2010. 2
[13] R. Girshick. Fast R-CNN. ICCV, 2015. 2, 3, 6
[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014. 1, 2
[15] A. Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013. 2, 4
[16] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. DRAW: A recurrent neural network for image generation. ICML, 2015. 1, 2, 3
[17] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2015, 2015. 2
[18] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997. 2, 4
[19] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. NIPS, 2015. 1, 2, 3
[20] Y. Jia, M. Salzmann, and T. Darrell. Learning cross-modality similarity for multinomial data. ICCV, 2011. 2
[21] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. CVPR, 2015. 1,2, 4, 5, 6, 7
[22] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. arXiv preprint arXiv:1506.02078, 2015. 2
[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. ICLR, 2015. 4
[24] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. TACL, 2015. 2
[25] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016. 4
[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012. 1, 2
[27] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. Baby talk: Understanding and generating
simple image descriptions. CVPR, 2011. 2
[28] P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and
Y. Choi. Generalizing image captions for image-text parallel
corpus. In ACL (2), pages 790–796. Citeseer, 2013. 2
[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. GradientBased learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998. 2
[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll´ar, and C. L. Zitnick. Microsoft COCO: Common objects in context. ECCV, 2014. 5
[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional
networks for semantic segmentation. CVPR, 2015. 2
[32] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. Explain

images with multimodal recurrent neural networks. arXiv
preprint arXiv:1410.1090, 2014. 1, 2, 4
[33] T. Mikolov, M. Karafi´at, L. Burget, J. Cernock`y, and S. Khudanpur. Recurrent neural network Based language model. In INTERSPEECH, 2010. 2, 4
[34] V. Ordonez, X. Han, P. Kuznetsova, G. Kulkarni, M. Mitchell, K. Yamaguchi, K. Stratos, A. Goyal, J. Dodge, A. Mensch, et al. Large scale retrieval and generation of image
descriptions. International Journal of Computer Vision (IJCV), 2015. 2
[35] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo,
J. Hockenmaier, and S. Lazebnik. Flickr30k entities: Collecting
region-to-phrase correspondences for richer imageto-sentence models. ICCV, 2015. 4
[36] qassemoquab. stnbhwd. https://github.com/qassemoquab/stnbhwd, 2015. 4
[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You
only look once: Unified, real-time object detection. arXiv
preprint arXiv:1506.02640, 2015. 2
[38] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. NIPS, 2015. 1, 2, 3, 4, 6
[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh,
S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein,
A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual
Recognition Challenge. International Journal of Computer
Vision (IJCV), pages 1–42, April 2015. 1, 2, 4
[40] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. ICLR, 2014. 1, 2
[41] K. Simonyan and A. Zisserman. Very deep convolutional
networks for large-scale image recognition. ICLR, 2015. 2
[42] R. Socher and L. Fei-Fei. Connecting modalities: Semisupervised segmentation and annotation of images using unaligned text corpora. CVPR, 2010. 2
[43] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. TACL, 2014. 2
[44] I. Sutskever, J. Martens, and G. E. Hinton. Generating text
with recurrent neural networks. ICML, 2011. 2, 4
[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich.
Going deeper with convolutions. CVPR, 2015. 1
[46] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. arXiv preprint arXiv:1412.1441, 2014. 1, 2
[47] B. Thomee, B. Elizalde, D. A. Shamma, K. Ni, G. Friedland, D. Poland, D. Borth, and L.-J. Li.
Yfcc100m: The new data in multimedia research. Communications of the ACM, 59(2):64–73, 2016. 4
[48] R. Vedantam, C. Lawrence Zitnick, and D. Parikh. Cider:
Consensus-Based image description evaluation. CVPR,
2015. 2, 5
[49] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and
tell: A neural image caption generator. CVPR, 2015. 1, 2, 4
[50] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. Neural Networks,
1(4):339–356, 1988. 2
[51] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov,
R. Zemel, and Y. Bengio. Show, attend and tell: Neural image
caption generation with visual attention. ICML, 2015. 1,2
[52] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. TACL, 2014. 4
[53] M. D. Zeiler and R. Fergus. Visualizing and understanding
convolutional networks. ECCV, 2014. 1
[54] C. L. Zitnick and P. Doll´ar. Edge boxes: Locating object
proposals from edges. ECCV, 2014. 6

附录

A 本设计使用到的部分工程技术简介

A.1 Python+Numpy+Matplotlib+Pandas+nltk

Python 是一种脚本编程语言，支持面向过程和面向对象的程序设计，具有语法清晰简洁，被誉为“可执行的伪代码”，和众多其他语言能够结合起来使用，并且代码完全开源。能够高效地处理文本文件。另外由于活跃的社区和众多的支持库，所以在机器学习中的使用尤为突出。Python2 已经停止维护，目前只有 Python3 这个版本。本设计中使用到了 Python3（3.6）以及其部分相关支持库如，NumPy，Matplotlib，Pandas 等。

NumPy 是一个用于科学计算的 Python 函数库，提供了很多使用的科学计算函数，比如矩阵乘法，矩阵转置，逆运算以及其余线性代数，概率，随机数等科学运算。

Matplotlib 是一个 Python 绘制库，可以用于对图片常用操作，绘制专业的图表，以及其他的数据可视化任务。

Pandas 是一个 Python 数据分析库，基于 NumPy，能够对文本文件，计算数据进行高效地计算和分析。

Nltk 是一个 Python 自然语言处理开源工具库，提供了很多自然语言处理的重用函数接口。

A.2 Keras

Keras 由 Francois Chollect 开发的高层神经网络 API，基于 TensorFlow、Theano 以及 CNTK 后端，本设计使用的为 TensorFlow 后端。支持自动求导，支持 CNN 和 RNN 以及二者的结合。支持 CPU 和 GPU 训练的无缝切换。足以满足本设计的全部需求。

其和其他部分深度学习框架对比如下：

附表 1 Keras 和部分深度学习框架比较

Extra Table 1 Keras compared with several deeplearning frameworks

名称	支持平台	支持接口	自动求导	预训练模型	CNN 开发	RNN 开发	单机多 GPU
Keras	Windows Linux, OS X	Python	√	√	√	√	×
TensorFlow	Windows,	Python,	√	√	√	√	√

	Linux, OS X	C/C++, Java, Go					
PyTorch	Windows, Linux, OS X	Python	√	√	√	√	√

*注：*1 Caffe 和 Caffe2 已经完全并入 PyTorch 中。*

A.3 TensorFlow+Tensorboard

TensorFlow 是一个流行的开源机器学习计算平台（framework），可以使用相对简单的语言来实现比较复杂的算法模型，支持多种计算平台和语言接口，能提高项目开发效率。核心原理是其计算图和会话机制。其中 TensorBoard 使 TensorFlow 中的可视化工具组件，能够将计算图，运算参数和结果，图片等展示于浏览器之中。

A.4 Django+Nginx+uWsgi+Linux

Django 是由 Adian Holovaty 和 Simon Willison 使用 Python 开发的一个 Python Web 框架，采用 MVT 设计模式，即 Model 模型，View 视图和 Template 模板。支持 ORM（Object Relational Mapping, 对象关系映射）对数据库中记录进行操作。Nginx 是一个 HTTP 反向代理服务器，即获取来自客户端的请求，将请求传递到可能有的多个服务器中的一个目标服务器，本项目中使用其来支持多用户对服务器的同时访问，即实现了负载均衡。Uwsgi 是一个网络接口服务器，实现了 wsgi 协议，在本设计中的最后一章项目中作为 nginx 服务器和 Python Django 站点之间的通讯中间件。Linux 是一套开源的操作系统，本设计中将 Django 站点部署到远程的 Linux 服务器上。

A.5 Bootstrap+HTML+CSS+JavaScript+jQuery+D3.js

Bootstrap 是一套用于 HTML, CSS 和 jQuery 开发响应式布局移动设备优先的 Web 项目的工具集。HTML 是超文本标记语言，结合 CSS 层叠样式表和 JavaScript 语言常用于 web 开发。jQuery 是一个对 JavaScript 封装了的库，简化了后者的编程。D3.js 是一个用于数据可视化的 JavaScript 库，核心是基于 SVG 的操作。

A.6 MySQL

MySQL 是一种关系型数据库，内部使用 B+树来实现，免费开源，使用简单，适合小型项目和团队开发一般的应用项目。

B 本设计部分模型参数设置详情

B.1 InceptionV3

附表 2 描述模型中的 InceptionV3 图像模型参数详情

Extra Table 2 Parameters of the inceptionV3 image model

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, None, None, 3	0	
conv2d_1 (Conv2D)	(None, None, None, 3	864	input_1[0][0]
batch_normalization_1 (BatchNor	(None, None, None, 3	96	conv2d_1[0][0]
activation_1 (Activation)	(None, None, None, 3	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, None, None, 3	9216	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, None, None, 3	96	conv2d_2[0][0]
activation_2 (Activation)	(None, None, None, 3	0	batch_normalization_2[0][0]
conv2d_3 (Conv2D)	(None, None, None, 6	18432	activation_2[0][0]
batch_normalization_3 (BatchNor	(None, None, None, 6	192	conv2d_3[0][0]
activation_3 (Activation)	(None, None, None, 6	0	batch_normalization_3[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, None, None, 6	0	activation_3[0][0]
conv2d_4 (Conv2D)	(None, None, None, 8	5120	max_pooling2d_1[0][0]
batch_normalization_4 (BatchNor	(None, None, None, 8	240	conv2d_4[0][0]
activation_4 (Activation)	(None, None, None, 8	0	batch_normalization_4[0][0]
conv2d_5 (Conv2D)	(None, None, None, 1	138240	activation_4[0][0]
batch_normalization_5 (BatchNor	(None, None, None, 1	576	conv2d_5[0][0]
activation_5 (Activation)	(None, None, None, 1	0	batch_normalization_5[0][0]

```
max_pooling2d_2 (MaxPooling2D)  (None, None, None, 1 0           activation_5[0][0]
__________________________________________________________________________________________________
conv2d_9 (Conv2D)               (None, None, None, 6 12288       max_pooling2d_2[0][0]
__________________________________________________________________________________________________
batch_normalization_9 (BatchNor (None, None, None, 6 192         conv2d_9[0][0]
__________________________________________________________________________________________________
activation_9 (Activation)       (None, None, None, 6 0           batch_normalization_9[0][0]
__________________________________________________________________________________________________
conv2d_7 (Conv2D)               (None, None, None, 4 9216        max_pooling2d_2[0][0]
__________________________________________________________________________________________________
conv2d_10 (Conv2D)              (None, None, None, 9 55296       activation_9[0][0]
__________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, None, None, 4 144         conv2d_7[0][0]
__________________________________________________________________________________________________
batch_normalization_10 (BatchNo (None, None, None, 9 288         conv2d_10[0][0]
__________________________________________________________________________________________________
activation_7 (Activation)       (None, None, None, 4 0           batch_normalization_7[0][0]
__________________________________________________________________________________________________
activation_10 (Activation)      (None, None, None, 9 0           batch_normalization_10[0][0]
__________________________________________________________________________________________________
average_pooling2d_1 (AveragePoo (None, None, None, 1 0           max_pooling2d_2[0][0]
__________________________________________________________________________________________________
conv2d_6 (Conv2D)               (None, None, None, 6 12288       max_pooling2d_2[0][0]
__________________________________________________________________________________________________
conv2d_8 (Conv2D)               (None, None, None, 6 76800       activation_7[0][0]
__________________________________________________________________________________________________
conv2d_11 (Conv2D)              (None, None, None, 9 82944       activation_10[0][0]
__________________________________________________________________________________________________
conv2d_12 (Conv2D)              (None, None, None, 3 6144        average_pooling2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, None, None, 6 192         conv2d_6[0][0]
__________________________________________________________________________________________________
batch_normalization_8 (BatchNor (None, None, None, 6 192         conv2d_8[0][0]
__________________________________________________________________________________________________
batch_normalization_11 (BatchNo (None, None, None, 9 288         conv2d_11[0][0]
__________________________________________________________________________________________________
batch_normalization_12 (BatchNo (None, None, None, 3 96          conv2d_12[0][0]
__________________________________________________________________________________________________
activation_6 (Activation)       (None, None, None, 6 0           batch_normalization_6[0][0]
__________________________________________________________________________________________________
activation_8 (Activation)       (None, None, None, 6 0           batch_normalization_8[0][0]
__________________________________________________________________________________________________
```

```
activation_11 (Activation)      (None, None, None, 9 0           batch_normalization_11[0][0]
__________________________________________________________________________________________________
activation_12 (Activation)      (None, None, None, 3 0           batch_normalization_12[0][0]
__________________________________________________________________________________________________
mixed0 (Concatenate)            (None, None, None, 2 0           activation_6[0][0]
activation_8[0][0]
activation_11[0][0]
activation_12[0][0]
__________________________________________________________________________________________________
conv2d_16 (Conv2D)              (None, None, None, 6 16384       mixed0[0][0]
__________________________________________________________________________________________________
batch_normalization_16 (BatchNo (None, None, None, 6 192         conv2d_16[0][0]
__________________________________________________________________________________________________
activation_16 (Activation)      (None, None, None, 6 0           batch_normalization_16[0][0]
__________________________________________________________________________________________________
conv2d_14 (Conv2D)              (None, None, None, 4 12288       mixed0[0][0]
__________________________________________________________________________________________________
conv2d_17 (Conv2D)              (None, None, None, 9 55296       activation_16[0][0]
__________________________________________________________________________________________________
batch_normalization_14 (BatchNo (None, None, None, 4 144         conv2d_14[0][0]
__________________________________________________________________________________________________
batch_normalization_17 (BatchNo (None, None, None, 9 288         conv2d_17[0][0]
__________________________________________________________________________________________________
activation_14 (Activation)      (None, None, None, 4 0           batch_normalization_14[0][0]
__________________________________________________________________________________________________
activation_17 (Activation)      (None, None, None, 9 0           batch_normalization_17[0][0]
__________________________________________________________________________________________________
average_pooling2d_2 (AveragePoo (None, None, None, 2 0           mixed0[0][0]
__________________________________________________________________________________________________
conv2d_13 (Conv2D)              (None, None, None, 6 16384       mixed0[0][0]
__________________________________________________________________________________________________
conv2d_15 (Conv2D)              (None, None, None, 6 76800       activation_14[0][0]
__________________________________________________________________________________________________
conv2d_18 (Conv2D)              (None, None, None, 9 82944       activation_17[0][0]
__________________________________________________________________________________________________
conv2d_19 (Conv2D)              (None, None, None, 6 16384       average_pooling2d_2[0][0]
__________________________________________________________________________________________________
batch_normalization_13 (BatchNo (None, None, None, 6 192         conv2d_13[0][0]
__________________________________________________________________________________________________
batch_normalization_15 (BatchNo (None, None, None, 6 192         conv2d_15[0][0]
__________________________________________________________________________________________________
batch_normalization_18 (BatchNo (None, None, None, 9 288         conv2d_18[0][0]
```

```
__________________________________________________________________________________________________
batch_normalization_19 (BatchNo (None, None, None, 6 192         conv2d_19[0][0]
__________________________________________________________________________________________________
activation_13 (Activation)      (None, None, None, 6 0           batch_normalization_13[0][0]
__________________________________________________________________________________________________
activation_15 (Activation)      (None, None, None, 6 0           batch_normalization_15[0][0]
__________________________________________________________________________________________________
activation_18 (Activation)      (None, None, None, 9 0           batch_normalization_18[0][0]
__________________________________________________________________________________________________
activation_19 (Activation)      (None, None, None, 6 0           batch_normalization_19[0][0]
__________________________________________________________________________________________________
mixed1 (Concatenate)            (None, None, None, 2 0           activation_13[0][0]
activation_15[0][0]
activation_18[0][0]
activation_19[0][0]
__________________________________________________________________________________________________
conv2d_23 (Conv2D)              (None, None, None, 6 18432       mixed1[0][0]
__________________________________________________________________________________________________
batch_normalization_23 (BatchNo (None, None, None, 6 192         conv2d_23[0][0]
__________________________________________________________________________________________________
activation_23 (Activation)      (None, None, None, 6 0           batch_normalization_23[0][0]
__________________________________________________________________________________________________
conv2d_21 (Conv2D)              (None, None, None, 4 13824       mixed1[0][0]
__________________________________________________________________________________________________
conv2d_24 (Conv2D)              (None, None, None, 9 55296       activation_23[0][0]
__________________________________________________________________________________________________
batch_normalization_21 (BatchNo (None, None, None, 4 144         conv2d_21[0][0]
__________________________________________________________________________________________________
batch_normalization_24 (BatchNo (None, None, None, 9 288         conv2d_24[0][0]
__________________________________________________________________________________________________
activation_21 (Activation)      (None, None, None, 4 0           batch_normalization_21[0][0]
__________________________________________________________________________________________________
activation_24 (Activation)      (None, None, None, 9 0           batch_normalization_24[0][0]
__________________________________________________________________________________________________
average_pooling2d_3 (AveragePoo (None, None, None, 2 0           mixed1[0][0]
__________________________________________________________________________________________________
conv2d_20 (Conv2D)              (None, None, None, 6 18432       mixed1[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)              (None, None, None, 6 76800       activation_21[0][0]
__________________________________________________________________________________________________
conv2d_25 (Conv2D)              (None, None, None, 9 82944       activation_24[0][0]
__________________________________________________________________________________________________
```

```
conv2d_26 (Conv2D)              (None, None, None, 6 18432       average_pooling2d_3[0][0]
__________________________________________________________________________________________________
batch_normalization_20 (BatchNo (None, None, None, 6 192         conv2d_20[0][0]
__________________________________________________________________________________________________
batch_normalization_22 (BatchNo (None, None, None, 6 192         conv2d_22[0][0]
__________________________________________________________________________________________________
batch_normalization_25 (BatchNo (None, None, None, 9 288         conv2d_25[0][0]
__________________________________________________________________________________________________
batch_normalization_26 (BatchNo (None, None, None, 6 192         conv2d_26[0][0]
__________________________________________________________________________________________________
activation_20 (Activation)      (None, None, None, 6 0           batch_normalization_20[0][0]
__________________________________________________________________________________________________
activation_22 (Activation)      (None, None, None, 6 0           batch_normalization_22[0][0]
__________________________________________________________________________________________________
activation_25 (Activation)      (None, None, None, 9 0           batch_normalization_25[0][0]
__________________________________________________________________________________________________
activation_26 (Activation)      (None, None, None, 6 0           batch_normalization_26[0][0]
__________________________________________________________________________________________________
mixed2 (Concatenate)            (None, None, None, 2 0           activation_20[0][0]
                                                                 activation_22[0][0]
                                                                 activation_25[0][0]
                                                                 activation_26[0][0]
__________________________________________________________________________________________________
conv2d_28 (Conv2D)              (None, None, None, 6 18432       mixed2[0][0]
__________________________________________________________________________________________________
batch_normalization_28 (BatchNo (None, None, None, 6 192         conv2d_28[0][0]
__________________________________________________________________________________________________
activation_28 (Activation)      (None, None, None, 6 0           batch_normalization_28[0][0]
__________________________________________________________________________________________________
conv2d_29 (Conv2D)              (None, None, None, 9 55296       activation_28[0][0]
__________________________________________________________________________________________________
batch_normalization_29 (BatchNo (None, None, None, 9 288         conv2d_29[0][0]
__________________________________________________________________________________________________
activation_29 (Activation)      (None, None, None, 9 0           batch_normalization_29[0][0]
__________________________________________________________________________________________________
conv2d_27 (Conv2D)              (None, None, None, 3 995328      mixed2[0][0]
__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, None, None, 9 82944       activation_29[0][0]
__________________________________________________________________________________________________
batch_normalization_27 (BatchNo (None, None, None, 3 1152        conv2d_27[0][0]
__________________________________________________________________________________________________
batch_normalization_30 (BatchNo (None, None, None, 9 288         conv2d_30[0][0]
```

```
activation_27 (Activation)      (None, None, None, 3 0          batch_normalization_27[0][0]
__________________________________________________________________________________________
activation_30 (Activation)      (None, None, None, 9 0          batch_normalization_30[0][0]
__________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, None, None, 2 0          mixed2[0][0]
__________________________________________________________________________________________
mixed3 (Concatenate)            (None, None, None, 7 0          activation_27[0][0]
activation_30[0][0]
max_pooling2d_3[0][0]
__________________________________________________________________________________________
conv2d_35 (Conv2D)              (None, None, None, 1 98304      mixed3[0][0]
__________________________________________________________________________________________
batch_normalization_35 (BatchNo (None, None, None, 1 384        conv2d_35[0][0]
__________________________________________________________________________________________
activation_35 (Activation)      (None, None, None, 1 0          batch_normalization_35[0][0]
__________________________________________________________________________________________
conv2d_36 (Conv2D)              (None, None, None, 1 114688     activation_35[0][0]
__________________________________________________________________________________________
batch_normalization_36 (BatchNo (None, None, None, 1 384        conv2d_36[0][0]
__________________________________________________________________________________________
activation_36 (Activation)      (None, None, None, 1 0          batch_normalization_36[0][0]
__________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, None, None, 1 98304      mixed3[0][0]
__________________________________________________________________________________________
conv2d_37 (Conv2D)              (None, None, None, 1 114688     activation_36[0][0]
__________________________________________________________________________________________
batch_normalization_32 (BatchNo (None, None, None, 1 384        conv2d_32[0][0]
__________________________________________________________________________________________
batch_normalization_37 (BatchNo (None, None, None, 1 384        conv2d_37[0][0]
__________________________________________________________________________________________
activation_32 (Activation)      (None, None, None, 1 0          batch_normalization_32[0][0]
__________________________________________________________________________________________
activation_37 (Activation)      (None, None, None, 1 0          batch_normalization_37[0][0]
__________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, None, None, 1 114688     activation_32[0][0]
__________________________________________________________________________________________
conv2d_38 (Conv2D)              (None, None, None, 1 114688     activation_37[0][0]
__________________________________________________________________________________________
batch_normalization_33 (BatchNo (None, None, None, 1 384        conv2d_33[0][0]
__________________________________________________________________________________________
batch_normalization_38 (BatchNo (None, None, None, 1 384        conv2d_38[0][0]
```

```
activation_33 (Activation)      (None, None, None, 1 0           batch_normalization_33[0][0]
activation_38 (Activation)      (None, None, None, 1 0           batch_normalization_38[0][0]
average_pooling2d_4 (AveragePoo (None, None, None, 7 0           mixed3[0][0]
conv2d_31 (Conv2D)              (None, None, None, 1 147456      mixed3[0][0]
conv2d_34 (Conv2D)              (None, None, None, 1 172032      activation_33[0][0]
conv2d_39 (Conv2D)              (None, None, None, 1 172032      activation_38[0][0]
conv2d_40 (Conv2D)              (None, None, None, 1 147456      average_pooling2d_4[0][0]
batch_normalization_31 (BatchNo (None, None, None, 1 576         conv2d_31[0][0]
batch_normalization_34 (BatchNo (None, None, None, 1 576         conv2d_34[0][0]
batch_normalization_39 (BatchNo (None, None, None, 1 576         conv2d_39[0][0]
batch_normalization_40 (BatchNo (None, None, None, 1 576         conv2d_40[0][0]
activation_31 (Activation)      (None, None, None, 1 0           batch_normalization_31[0][0]
activation_34 (Activation)      (None, None, None, 1 0           batch_normalization_34[0][0]
activation_39 (Activation)      (None, None, None, 1 0           batch_normalization_39[0][0]
activation_40 (Activation)      (None, None, None, 1 0           batch_normalization_40[0][0]
mixed4 (Concatenate)            (None, None, None, 7 0           activation_31[0][0]
activation_34[0][0]
activation_39[0][0]
activation_40[0][0]
conv2d_45 (Conv2D)              (None, None, None, 1 122880      mixed4[0][0]
batch_normalization_45 (BatchNo (None, None, None, 1 480         conv2d_45[0][0]
activation_45 (Activation)      (None, None, None, 1 0           batch_normalization_45[0][0]
```

```
conv2d_46 (Conv2D)              (None, None, None, 1 179200      activation_45[0][0]
__________________________________________________________________________________________________
batch_normalization_46 (BatchNo (None, None, None, 1 480         conv2d_46[0][0]
__________________________________________________________________________________________________
activation_46 (Activation)      (None, None, None, 1 0           batch_normalization_46[0][0]
__________________________________________________________________________________________________
conv2d_42 (Conv2D)              (None, None, None, 1 122880      mixed4[0][0]
__________________________________________________________________________________________________
conv2d_47 (Conv2D)              (None, None, None, 1 179200      activation_46[0][0]
__________________________________________________________________________________________________
batch_normalization_42 (BatchNo (None, None, None, 1 480         conv2d_42[0][0]
__________________________________________________________________________________________________
batch_normalization_47 (BatchNo (None, None, None, 1 480         conv2d_47[0][0]
__________________________________________________________________________________________________
activation_42 (Activation)      (None, None, None, 1 0           batch_normalization_42[0][0]
__________________________________________________________________________________________________
activation_47 (Activation)      (None, None, None, 1 0           batch_normalization_47[0][0]
__________________________________________________________________________________________________
conv2d_43 (Conv2D)              (None, None, None, 1 179200      activation_42[0][0]
__________________________________________________________________________________________________
conv2d_48 (Conv2D)              (None, None, None, 1 179200      activation_47[0][0]
__________________________________________________________________________________________________
batch_normalization_43 (BatchNo (None, None, None, 1 480         conv2d_43[0][0]
__________________________________________________________________________________________________
batch_normalization_48 (BatchNo (None, None, None, 1 480         conv2d_48[0][0]
__________________________________________________________________________________________________
activation_43 (Activation)      (None, None, None, 1 0           batch_normalization_43[0][0]
__________________________________________________________________________________________________
activation_48 (Activation)      (None, None, None, 1 0           batch_normalization_48[0][0]
__________________________________________________________________________________________________
average_pooling2d_5 (AveragePoo (None, None, None, 7 0           mixed4[0][0]
__________________________________________________________________________________________________
conv2d_41 (Conv2D)              (None, None, None, 1 147456      mixed4[0][0]
__________________________________________________________________________________________________
conv2d_44 (Conv2D)              (None, None, None, 1 215040      activation_43[0][0]
__________________________________________________________________________________________________
conv2d_49 (Conv2D)              (None, None, None, 1 215040      activation_48[0][0]
__________________________________________________________________________________________________
conv2d_50 (Conv2D)              (None, None, None, 1 147456      average_pooling2d_5[0][0]
__________________________________________________________________________________________________
batch_normalization_41 (BatchNo (None, None, None, 1 576         conv2d_41[0][0]
__________________________________________________________________________________________________
```

```
batch_normalization_44 (BatchNo (None, None, None, 1 576         conv2d_44[0][0]
__________________________________________________________________________________________________
batch_normalization_49 (BatchNo (None, None, None, 1 576         conv2d_49[0][0]
__________________________________________________________________________________________________
batch_normalization_50 (BatchNo (None, None, None, 1 576         conv2d_50[0][0]
__________________________________________________________________________________________________
activation_41 (Activation)      (None, None, None, 1 0           batch_normalization_41[0][0]
__________________________________________________________________________________________________
activation_44 (Activation)      (None, None, None, 1 0           batch_normalization_44[0][0]
__________________________________________________________________________________________________
activation_49 (Activation)      (None, None, None, 1 0           batch_normalization_49[0][0]
__________________________________________________________________________________________________
activation_50 (Activation)      (None, None, None, 1 0           batch_normalization_50[0][0]
__________________________________________________________________________________________________
mixed5 (Concatenate)            (None, None, None, 7 0           activation_41[0][0]
                                                                 activation_44[0][0]
                                                                 activation_49[0][0]
                                                                 activation_50[0][0]
__________________________________________________________________________________________________
conv2d_55 (Conv2D)              (None, None, None, 1 122880      mixed5[0][0]
__________________________________________________________________________________________________
batch_normalization_55 (BatchNo (None, None, None, 1 480         conv2d_55[0][0]
__________________________________________________________________________________________________
activation_55 (Activation)      (None, None, None, 1 0           batch_normalization_55[0][0]
__________________________________________________________________________________________________
conv2d_56 (Conv2D)              (None, None, None, 1 179200      activation_55[0][0]
__________________________________________________________________________________________________
batch_normalization_56 (BatchNo (None, None, None, 1 480         conv2d_56[0][0]
__________________________________________________________________________________________________
activation_56 (Activation)      (None, None, None, 1 0           batch_normalization_56[0][0]
__________________________________________________________________________________________________
conv2d_52 (Conv2D)              (None, None, None, 1 122880      mixed5[0][0]
__________________________________________________________________________________________________
conv2d_57 (Conv2D)              (None, None, None, 1 179200      activation_56[0][0]
__________________________________________________________________________________________________
batch_normalization_52 (BatchNo (None, None, None, 1 480         conv2d_52[0][0]
__________________________________________________________________________________________________
batch_normalization_57 (BatchNo (None, None, None, 1 480         conv2d_57[0][0]
__________________________________________________________________________________________________
activation_52 (Activation)      (None, None, None, 1 0           batch_normalization_52[0][0]
__________________________________________________________________________________________________
activation_57 (Activation)      (None, None, None, 1 0           batch_normalization_57[0][0]
```

```
__________________________________________________________________________________________________
conv2d_53 (Conv2D)              (None, None, None, 1 179200      activation_52[0][0]
__________________________________________________________________________________________________
conv2d_58 (Conv2D)              (None, None, None, 1 179200      activation_57[0][0]
__________________________________________________________________________________________________
batch_normalization_53 (BatchNo (None, None, None, 1 480         conv2d_53[0][0]
__________________________________________________________________________________________________
batch_normalization_58 (BatchNo (None, None, None, 1 480         conv2d_58[0][0]
__________________________________________________________________________________________________
activation_53 (Activation)      (None, None, None, 1 0           batch_normalization_53[0][0]
__________________________________________________________________________________________________
activation_58 (Activation)      (None, None, None, 1 0           batch_normalization_58[0][0]
__________________________________________________________________________________________________
average_pooling2d_6 (AveragePoo (None, None, None, 7 0           mixed5[0][0]
__________________________________________________________________________________________________
conv2d_51 (Conv2D)              (None, None, None, 1 147456      mixed5[0][0]
__________________________________________________________________________________________________
conv2d_54 (Conv2D)              (None, None, None, 1 215040      activation_53[0][0]
__________________________________________________________________________________________________
conv2d_59 (Conv2D)              (None, None, None, 1 215040      activation_58[0][0]
__________________________________________________________________________________________________
conv2d_60 (Conv2D)              (None, None, None, 1 147456      average_pooling2d_6[0][0]
__________________________________________________________________________________________________
batch_normalization_51 (BatchNo (None, None, None, 1 576         conv2d_51[0][0]
__________________________________________________________________________________________________
batch_normalization_54 (BatchNo (None, None, None, 1 576         conv2d_54[0][0]
__________________________________________________________________________________________________
batch_normalization_59 (BatchNo (None, None, None, 1 576         conv2d_59[0][0]
__________________________________________________________________________________________________
batch_normalization_60 (BatchNo (None, None, None, 1 576         conv2d_60[0][0]
__________________________________________________________________________________________________
activation_51 (Activation)      (None, None, None, 1 0           batch_normalization_51[0][0]
__________________________________________________________________________________________________
activation_54 (Activation)      (None, None, None, 1 0           batch_normalization_54[0][0]
__________________________________________________________________________________________________
activation_59 (Activation)      (None, None, None, 1 0           batch_normalization_59[0][0]
__________________________________________________________________________________________________
activation_60 (Activation)      (None, None, None, 1 0           batch_normalization_60[0][0]
__________________________________________________________________________________________________
mixed6 (Concatenate)            (None, None, None, 7 0           activation_51[0][0]
activation_54[0][0]
activation_59[0][0]
```

```
activation_60[0][0]
__________________________________________________________________________________________________
conv2d_65 (Conv2D)              (None, None, None, 1 147456      mixed6[0][0]
__________________________________________________________________________________________________
batch_normalization_65 (BatchNo (None, None, None, 1 576         conv2d_65[0][0]
__________________________________________________________________________________________________
activation_65 (Activation)      (None, None, None, 1 0           batch_normalization_65[0][0]
__________________________________________________________________________________________________
conv2d_66 (Conv2D)              (None, None, None, 1 258048      activation_65[0][0]
__________________________________________________________________________________________________
batch_normalization_66 (BatchNo (None, None, None, 1 576         conv2d_66[0][0]
__________________________________________________________________________________________________
activation_66 (Activation)      (None, None, None, 1 0           batch_normalization_66[0][0]
__________________________________________________________________________________________________
conv2d_62 (Conv2D)              (None, None, None, 1 147456      mixed6[0][0]
__________________________________________________________________________________________________
conv2d_67 (Conv2D)              (None, None, None, 1 258048      activation_66[0][0]
__________________________________________________________________________________________________
batch_normalization_62 (BatchNo (None, None, None, 1 576         conv2d_62[0][0]
__________________________________________________________________________________________________
batch_normalization_67 (BatchNo (None, None, None, 1 576         conv2d_67[0][0]
__________________________________________________________________________________________________
activation_62 (Activation)      (None, None, None, 1 0           batch_normalization_62[0][0]
__________________________________________________________________________________________________
activation_67 (Activation)      (None, None, None, 1 0           batch_normalization_67[0][0]
__________________________________________________________________________________________________
conv2d_63 (Conv2D)              (None, None, None, 1 258048      activation_62[0][0]
__________________________________________________________________________________________________
conv2d_68 (Conv2D)              (None, None, None, 1 258048      activation_67[0][0]
__________________________________________________________________________________________________
batch_normalization_63 (BatchNo (None, None, None, 1 576         conv2d_63[0][0]
__________________________________________________________________________________________________
batch_normalization_68 (BatchNo (None, None, None, 1 576         conv2d_68[0][0]
__________________________________________________________________________________________________
activation_63 (Activation)      (None, None, None, 1 0           batch_normalization_63[0][0]
__________________________________________________________________________________________________
activation_68 (Activation)      (None, None, None, 1 0           batch_normalization_68[0][0]
__________________________________________________________________________________________________
average_pooling2d_7 (AveragePoo (None, None, None, 7 0           mixed6[0][0]
__________________________________________________________________________________________________
conv2d_61 (Conv2D)              (None, None, None, 1 147456      mixed6[0][0]
__________________________________________________________________________________________________
```

```
conv2d_64 (Conv2D)              (None, None, None, 1 258048      activation_63[0][0]
__________________________________________________________________________________________________
conv2d_69 (Conv2D)              (None, None, None, 1 258048      activation_68[0][0]
__________________________________________________________________________________________________
conv2d_70 (Conv2D)              (None, None, None, 1 147456      average_pooling2d_7[0][0]
__________________________________________________________________________________________________
batch_normalization_61 (BatchNo (None, None, None, 1 576         conv2d_61[0][0]
__________________________________________________________________________________________________
batch_normalization_64 (BatchNo (None, None, None, 1 576         conv2d_64[0][0]
__________________________________________________________________________________________________
batch_normalization_69 (BatchNo (None, None, None, 1 576         conv2d_69[0][0]
__________________________________________________________________________________________________
batch_normalization_70 (BatchNo (None, None, None, 1 576         conv2d_70[0][0]
__________________________________________________________________________________________________
activation_61 (Activation)      (None, None, None, 1 0           batch_normalization_61[0][0]
__________________________________________________________________________________________________
activation_64 (Activation)      (None, None, None, 1 0           batch_normalization_64[0][0]
__________________________________________________________________________________________________
activation_69 (Activation)      (None, None, None, 1 0           batch_normalization_69[0][0]
__________________________________________________________________________________________________
activation_70 (Activation)      (None, None, None, 1 0           batch_normalization_70[0][0]
__________________________________________________________________________________________________
mixed7 (Concatenate)            (None, None, None, 7 0           activation_61[0][0]
                                                                 activation_64[0][0]
                                                                 activation_69[0][0]
                                                                 activation_70[0][0]
__________________________________________________________________________________________________
conv2d_73 (Conv2D)              (None, None, None, 1 147456      mixed7[0][0]
__________________________________________________________________________________________________
batch_normalization_73 (BatchNo (None, None, None, 1 576         conv2d_73[0][0]
__________________________________________________________________________________________________
activation_73 (Activation)      (None, None, None, 1 0           batch_normalization_73[0][0]
__________________________________________________________________________________________________
conv2d_74 (Conv2D)              (None, None, None, 1 258048      activation_73[0][0]
__________________________________________________________________________________________________
batch_normalization_74 (BatchNo (None, None, None, 1 576         conv2d_74[0][0]
__________________________________________________________________________________________________
activation_74 (Activation)      (None, None, None, 1 0           batch_normalization_74[0][0]
__________________________________________________________________________________________________
conv2d_71 (Conv2D)              (None, None, None, 1 147456      mixed7[0][0]
__________________________________________________________________________________________________
conv2d_75 (Conv2D)              (None, None, None, 1 258048      activation_74[0][0]
```

```
batch_normalization_71 (BatchNo (None, None, None, 1 576         conv2d_71[0][0]
batch_normalization_75 (BatchNo (None, None, None, 1 576         conv2d_75[0][0]
activation_71 (Activation)      (None, None, None, 1 0           batch_normalization_71[0][0]
activation_75 (Activation)      (None, None, None, 1 0           batch_normalization_75[0][0]
conv2d_72 (Conv2D)              (None, None, None, 3 552960      activation_71[0][0]
conv2d_76 (Conv2D)              (None, None, None, 1 331776      activation_75[0][0]
batch_normalization_72 (BatchNo (None, None, None, 3 960         conv2d_72[0][0]
batch_normalization_76 (BatchNo (None, None, None, 1 576         conv2d_76[0][0]
activation_72 (Activation)      (None, None, None, 3 0           batch_normalization_72[0][0]
activation_76 (Activation)      (None, None, None, 1 0           batch_normalization_76[0][0]
max_pooling2d_4 (MaxPooling2D)  (None, None, None, 7 0           mixed7[0][0]
mixed8 (Concatenate)            (None, None, None, 1 0           activation_72[0][0]
activation_76[0][0]
max_pooling2d_4[0][0]
conv2d_81 (Conv2D)              (None, None, None, 4 573440      mixed8[0][0]
batch_normalization_81 (BatchNo (None, None, None, 4 1344        conv2d_81[0][0]
activation_81 (Activation)      (None, None, None, 4 0           batch_normalization_81[0][0]
conv2d_78 (Conv2D)              (None, None, None, 3 491520      mixed8[0][0]
conv2d_82 (Conv2D)              (None, None, None, 3 1548288     activation_81[0][0]
batch_normalization_78 (BatchNo (None, None, None, 3 1152        conv2d_78[0][0]
batch_normalization_82 (BatchNo (None, None, None, 3 1152        conv2d_82[0][0]
activation_78 (Activation)      (None, None, None, 3 0           batch_normalization_78[0][0]
```

```
activation_82 (Activation)      (None, None, None, 3 0           batch_normalization_82[0][0]
conv2d_79 (Conv2D)              (None, None, None, 3 442368      activation_78[0][0]
conv2d_80 (Conv2D)              (None, None, None, 3 442368      activation_78[0][0]
conv2d_83 (Conv2D)              (None, None, None, 3 442368      activation_82[0][0]
conv2d_84 (Conv2D)              (None, None, None, 3 442368      activation_82[0][0]
average_pooling2d_8 (AveragePoo (None, None, None, 1 0           mixed8[0][0]
conv2d_77 (Conv2D)              (None, None, None, 3 409600      mixed8[0][0]
batch_normalization_79 (BatchNo (None, None, None, 3 1152        conv2d_79[0][0]
batch_normalization_80 (BatchNo (None, None, None, 3 1152        conv2d_80[0][0]
batch_normalization_83 (BatchNo (None, None, None, 3 1152        conv2d_83[0][0]
batch_normalization_84 (BatchNo (None, None, None, 3 1152        conv2d_84[0][0]
conv2d_85 (Conv2D)              (None, None, None, 1 245760      average_pooling2d_8[0][0]
batch_normalization_77 (BatchNo (None, None, None, 3 960         conv2d_77[0][0]
activation_79 (Activation)      (None, None, None, 3 0           batch_normalization_79[0][0]
activation_80 (Activation)      (None, None, None, 3 0           batch_normalization_80[0][0]
activation_83 (Activation)      (None, None, None, 3 0           batch_normalization_83[0][0]
activation_84 (Activation)      (None, None, None, 3 0           batch_normalization_84[0][0]
batch_normalization_85 (BatchNo (None, None, None, 1 576         conv2d_85[0][0]
activation_77 (Activation)      (None, None, None, 3 0           batch_normalization_77[0][0]
mixed9_0 (Concatenate)          (None, None, None, 7 0           activation_79[0][0]
activation_80[0][0]
```

```
concatenate_1 (Concatenate)     (None, None, None, 7 0           activation_83[0][0]
activation_84[0][0]
__________________________________________________________________________________________________
activation_85 (Activation)      (None, None, None, 1 0           batch_normalization_85[0][0]
__________________________________________________________________________________________________
mixed9 (Concatenate)            (None, None, None, 2 0           activation_77[0][0]
mixed9_0[0][0]
concatenate_1[0][0]
activation_85[0][0]
__________________________________________________________________________________________________
conv2d_90 (Conv2D)              (None, None, None, 4 917504      mixed9[0][0]
__________________________________________________________________________________________________
batch_normalization_90 (BatchNo (None, None, None, 4 1344        conv2d_90[0][0]
__________________________________________________________________________________________________
activation_90 (Activation)      (None, None, None, 4 0           batch_normalization_90[0][0]
__________________________________________________________________________________________________
conv2d_87 (Conv2D)              (None, None, None, 3 786432      mixed9[0][0]
__________________________________________________________________________________________________
conv2d_91 (Conv2D)              (None, None, None, 3 1548288     activation_90[0][0]
__________________________________________________________________________________________________
batch_normalization_87 (BatchNo (None, None, None, 3 1152        conv2d_87[0][0]
__________________________________________________________________________________________________
batch_normalization_91 (BatchNo (None, None, None, 3 1152        conv2d_91[0][0]
__________________________________________________________________________________________________
activation_87 (Activation)      (None, None, None, 3 0           batch_normalization_87[0][0]
__________________________________________________________________________________________________
activation_91 (Activation)      (None, None, None, 3 0           batch_normalization_91[0][0]
__________________________________________________________________________________________________
conv2d_88 (Conv2D)              (None, None, None, 3 442368      activation_87[0][0]
__________________________________________________________________________________________________
conv2d_89 (Conv2D)              (None, None, None, 3 442368      activation_87[0][0]
__________________________________________________________________________________________________
conv2d_92 (Conv2D)              (None, None, None, 3 442368      activation_91[0][0]
__________________________________________________________________________________________________
conv2d_93 (Conv2D)              (None, None, None, 3 442368      activation_91[0][0]
__________________________________________________________________________________________________
average_pooling2d_9 (AveragePoo (None, None, None, 2 0           mixed9[0][0]
__________________________________________________________________________________________________
conv2d_86 (Conv2D)              (None, None, None, 3 655360      mixed9[0][0]
__________________________________________________________________________________________________
batch_normalization_88 (BatchNo (None, None, None, 3 1152        conv2d_88[0][0]
__________________________________________________________________________________________________
```

```
batch_normalization_89 (BatchNo (None, None, None, 3 1152        conv2d_89[0][0]
__________________________________________________________________________________________________
batch_normalization_92 (BatchNo (None, None, None, 3 1152        conv2d_92[0][0]
__________________________________________________________________________________________________
batch_normalization_93 (BatchNo (None, None, None, 3 1152        conv2d_93[0][0]
__________________________________________________________________________________________________
conv2d_94 (Conv2D)              (None, None, None, 1 393216      average_pooling2d_9[0][0]
__________________________________________________________________________________________________
batch_normalization_86 (BatchNo (None, None, None, 3 960         conv2d_86[0][0]
__________________________________________________________________________________________________
activation_88 (Activation)      (None, None, None, 3 0           batch_normalization_88[0][0]
__________________________________________________________________________________________________
activation_89 (Activation)      (None, None, None, 3 0           batch_normalization_89[0][0]
__________________________________________________________________________________________________
activation_92 (Activation)      (None, None, None, 3 0           batch_normalization_92[0][0]
__________________________________________________________________________________________________
activation_93 (Activation)      (None, None, None, 3 0           batch_normalization_93[0][0]
__________________________________________________________________________________________________
batch_normalization_94 (BatchNo (None, None, None, 1 576         conv2d_94[0][0]
__________________________________________________________________________________________________
activation_86 (Activation)      (None, None, None, 3 0           batch_normalization_86[0][0]
__________________________________________________________________________________________________
mixed9_1 (Concatenate)          (None, None, None, 7 0           activation_88[0][0]
activation_89[0][0]
__________________________________________________________________________________________________
concatenate_2 (Concatenate)     (None, None, None, 7 0           activation_92[0][0]
activation_93[0][0]
__________________________________________________________________________________________________
activation_94 (Activation)      (None, None, None, 1 0           batch_normalization_94[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)           (None, None, None, 2 0           activation_86[0][0]
mixed9_1[0][0]
concatenate_2[0][0]
activation_94[0][0]
__________________________________________________________________________________________________
avg_pool (GlobalAveragePooling2 (None, 2048)         0           mixed10[0][0]
==================================================================================================
Total params: 21,802,784
Trainable params: 21,768,352
Non-trainable params: 34,432
__________________________________________________________________________________________________
```

B.2 ResNet50

附表 3 描述模型中的 ResNet50 图像模型参数详情

Extra Table 3 Parameters of the ResNet50 image model

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 224, 224, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_1[0][0]
conv1 (Conv2D)	(None, 112, 112, 64)	9472	conv1_pad[0][0]
bn_conv1 (BatchNormalization)	(None, 112, 112, 64)	256	conv1[0][0]
activation_1 (Activation)	(None, 112, 112, 64)	0	bn_conv1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 55, 55, 64)	0	activation_1[0][0]
res2a_branch2a (Conv2D)	(None, 55, 55, 64)	4160	max_pooling2d_1[0][0]
bn2a_branch2a (BatchNormalizati	(None, 55, 55, 64)	256	res2a_branch2a[0][0]
activation_2 (Activation)	(None, 55, 55, 64)	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, 55, 55, 64)	36928	activation_2[0][0]
bn2a_branch2b (BatchNormalizati	(None, 55, 55, 64)	256	res2a_branch2b[0][0]
activation_3 (Activation)	(None, 55, 55, 64)	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, 55, 55, 256)	16640	activation_3[0][0]
res2a_branch1 (Conv2D)	(None, 55, 55, 256)	16640	max_pooling2d_1[0][0]
bn2a_branch2c (BatchNormalizati	(None, 55, 55, 256)	1024	res2a_branch2c[0][0]
bn2a_branch1 (BatchNormalizatio	(None, 55, 55, 256)	1024	res2a_branch1[0][0]
add_1 (Add)	(None, 55, 55, 256)	0	bn2a_branch2c[0][0] bn2a_branch1[0][0]

activation_4 (Activation)	(None, 55, 55, 256)	0	add_1[0][0]
res2b_branch2a (Conv2D)	(None, 55, 55, 64)	16448	activation_4[0][0]
bn2b_branch2a (BatchNormalizati	(None, 55, 55, 64)	256	res2b_branch2a[0][0]
activation_5 (Activation)	(None, 55, 55, 64)	0	bn2b_branch2a[0][0]
res2b_branch2b (Conv2D)	(None, 55, 55, 64)	36928	activation_5[0][0]
bn2b_branch2b (BatchNormalizati	(None, 55, 55, 64)	256	res2b_branch2b[0][0]
activation_6 (Activation)	(None, 55, 55, 64)	0	bn2b_branch2b[0][0]
res2b_branch2c (Conv2D)	(None, 55, 55, 256)	16640	activation_6[0][0]
bn2b_branch2c (BatchNormalizati	(None, 55, 55, 256)	1024	res2b_branch2c[0][0]
add_2 (Add)	(None, 55, 55, 256)	0	bn2b_branch2c[0][0] activation_4[0][0]
activation_7 (Activation)	(None, 55, 55, 256)	0	add_2[0][0]
res2c_branch2a (Conv2D)	(None, 55, 55, 64)	16448	activation_7[0][0]
bn2c_branch2a (BatchNormalizati	(None, 55, 55, 64)	256	res2c_branch2a[0][0]
activation_8 (Activation)	(None, 55, 55, 64)	0	bn2c_branch2a[0][0]
res2c_branch2b (Conv2D)	(None, 55, 55, 64)	36928	activation_8[0][0]
bn2c_branch2b (BatchNormalizati	(None, 55, 55, 64)	256	res2c_branch2b[0][0]
activation_9 (Activation)	(None, 55, 55, 64)	0	bn2c_branch2b[0][0]
res2c_branch2c (Conv2D)	(None, 55, 55, 256)	16640	activation_9[0][0]
bn2c_branch2c (BatchNormalizati	(None, 55, 55, 256)	1024	res2c_branch2c[0][0]
add_3 (Add)	(None, 55, 55, 256)	0	bn2c_branch2c[0][0] activation_7[0][0]

activation_10 (Activation)	(None, 55, 55, 256)	0	add_3[0][0]
res3a_branch2a (Conv2D)	(None, 28, 28, 128)	32896	activation_10[0][0]
bn3a_branch2a (BatchNormalizati	(None, 28, 28, 128)	512	res3a_branch2a[0][0]
activation_11 (Activation)	(None, 28, 28, 128)	0	bn3a_branch2a[0][0]
res3a_branch2b (Conv2D)	(None, 28, 28, 128)	147584	activation_11[0][0]
bn3a_branch2b (BatchNormalizati	(None, 28, 28, 128)	512	res3a_branch2b[0][0]
activation_12 (Activation)	(None, 28, 28, 128)	0	bn3a_branch2b[0][0]
res3a_branch2c (Conv2D)	(None, 28, 28, 512)	66048	activation_12[0][0]
res3a_branch1 (Conv2D)	(None, 28, 28, 512)	131584	activation_10[0][0]
bn3a_branch2c (BatchNormalizati	(None, 28, 28, 512)	2048	res3a_branch2c[0][0]
bn3a_branch1 (BatchNormalizatio	(None, 28, 28, 512)	2048	res3a_branch1[0][0]
add_4 (Add)	(None, 28, 28, 512)	0	bn3a_branch2c[0][0] bn3a_branch1[0][0]
activation_13 (Activation)	(None, 28, 28, 512)	0	add_4[0][0]
res3b_branch2a (Conv2D)	(None, 28, 28, 128)	65664	activation_13[0][0]
bn3b_branch2a (BatchNormalizati	(None, 28, 28, 128)	512	res3b_branch2a[0][0]
activation_14 (Activation)	(None, 28, 28, 128)	0	bn3b_branch2a[0][0]
res3b_branch2b (Conv2D)	(None, 28, 28, 128)	147584	activation_14[0][0]
bn3b_branch2b (BatchNormalizati	(None, 28, 28, 128)	512	res3b_branch2b[0][0]
activation_15 (Activation)	(None, 28, 28, 128)	0	bn3b_branch2b[0][0]
res3b_branch2c (Conv2D)	(None, 28, 28, 512)	66048	activation_15[0][0]

```
bn3b_branch2c (BatchNormalizati (None, 28, 28, 512)  2048        res3b_branch2c[0][0]
__________________________________________________________________________________________________
add_5 (Add)                     (None, 28, 28, 512)  0           bn3b_branch2c[0][0]
                                                                 activation_13[0][0]
__________________________________________________________________________________________________
activation_16 (Activation)      (None, 28, 28, 512)  0           add_5[0][0]
__________________________________________________________________________________________________
res3c_branch2a (Conv2D)         (None, 28, 28, 128)  65664       activation_16[0][0]
__________________________________________________________________________________________________
bn3c_branch2a (BatchNormalizati (None, 28, 28, 128)  512         res3c_branch2a[0][0]
__________________________________________________________________________________________________
activation_17 (Activation)      (None, 28, 28, 128)  0           bn3c_branch2a[0][0]
__________________________________________________________________________________________________
res3c_branch2b (Conv2D)         (None, 28, 28, 128)  147584      activation_17[0][0]
__________________________________________________________________________________________________
bn3c_branch2b (BatchNormalizati (None, 28, 28, 128)  512         res3c_branch2b[0][0]
__________________________________________________________________________________________________
activation_18 (Activation)      (None, 28, 28, 128)  0           bn3c_branch2b[0][0]
__________________________________________________________________________________________________
res3c_branch2c (Conv2D)         (None, 28, 28, 512)  66048       activation_18[0][0]
__________________________________________________________________________________________________
bn3c_branch2c (BatchNormalizati (None, 28, 28, 512)  2048        res3c_branch2c[0][0]
__________________________________________________________________________________________________
add_6 (Add)                     (None, 28, 28, 512)  0           bn3c_branch2c[0][0]
                                                                 activation_16[0][0]
__________________________________________________________________________________________________
activation_19 (Activation)      (None, 28, 28, 512)  0           add_6[0][0]
__________________________________________________________________________________________________
res3d_branch2a (Conv2D)         (None, 28, 28, 128)  65664       activation_19[0][0]
__________________________________________________________________________________________________
bn3d_branch2a (BatchNormalizati (None, 28, 28, 128)  512         res3d_branch2a[0][0]
__________________________________________________________________________________________________
activation_20 (Activation)      (None, 28, 28, 128)  0           bn3d_branch2a[0][0]
__________________________________________________________________________________________________
res3d_branch2b (Conv2D)         (None, 28, 28, 128)  147584      activation_20[0][0]
__________________________________________________________________________________________________
bn3d_branch2b (BatchNormalizati (None, 28, 28, 128)  512         res3d_branch2b[0][0]
__________________________________________________________________________________________________
activation_21 (Activation)      (None, 28, 28, 128)  0           bn3d_branch2b[0][0]
__________________________________________________________________________________________________
res3d_branch2c (Conv2D)         (None, 28, 28, 512)  66048       activation_21[0][0]
__________________________________________________________________________________________________
```

```
bn3d_branch2c (BatchNormalizati (None, 28, 28, 512)  2048        res3d_branch2c[0][0]
__________________________________________________________________________________________________
add_7 (Add)                     (None, 28, 28, 512)  0           bn3d_branch2c[0][0]
                                                                 activation_19[0][0]
__________________________________________________________________________________________________
activation_22 (Activation)      (None, 28, 28, 512)  0           add_7[0][0]
__________________________________________________________________________________________________
res4a_branch2a (Conv2D)         (None, 14, 14, 256)  131328      activation_22[0][0]
__________________________________________________________________________________________________
bn4a_branch2a (BatchNormalizati (None, 14, 14, 256)  1024        res4a_branch2a[0][0]
__________________________________________________________________________________________________
activation_23 (Activation)      (None, 14, 14, 256)  0           bn4a_branch2a[0][0]
__________________________________________________________________________________________________
res4a_branch2b (Conv2D)         (None, 14, 14, 256)  590080      activation_23[0][0]
__________________________________________________________________________________________________
bn4a_branch2b (BatchNormalizati (None, 14, 14, 256)  1024        res4a_branch2b[0][0]
__________________________________________________________________________________________________
activation_24 (Activation)      (None, 14, 14, 256)  0           bn4a_branch2b[0][0]
__________________________________________________________________________________________________
res4a_branch2c (Conv2D)         (None, 14, 14, 1024) 263168      activation_24[0][0]
__________________________________________________________________________________________________
res4a_branch1 (Conv2D)          (None, 14, 14, 1024) 525312      activation_22[0][0]
__________________________________________________________________________________________________
bn4a_branch2c (BatchNormalizati (None, 14, 14, 1024) 4096        res4a_branch2c[0][0]
__________________________________________________________________________________________________
bn4a_branch1 (BatchNormalizatio (None, 14, 14, 1024) 4096        res4a_branch1[0][0]
__________________________________________________________________________________________________
add_8 (Add)                     (None, 14, 14, 1024) 0           bn4a_branch2c[0][0]
                                                                 bn4a_branch1[0][0]
__________________________________________________________________________________________________
activation_25 (Activation)      (None, 14, 14, 1024) 0           add_8[0][0]
__________________________________________________________________________________________________
res4b_branch2a (Conv2D)         (None, 14, 14, 256)  262400      activation_25[0][0]
__________________________________________________________________________________________________
bn4b_branch2a (BatchNormalizati (None, 14, 14, 256)  1024        res4b_branch2a[0][0]
__________________________________________________________________________________________________
activation_26 (Activation)      (None, 14, 14, 256)  0           bn4b_branch2a[0][0]
__________________________________________________________________________________________________
res4b_branch2b (Conv2D)         (None, 14, 14, 256)  590080      activation_26[0][0]
__________________________________________________________________________________________________
bn4b_branch2b (BatchNormalizati (None, 14, 14, 256)  1024        res4b_branch2b[0][0]
__________________________________________________________________________________________________
```

activation_27 (Activation)	(None, 14, 14, 256)	0	bn4b_branch2b[0][0]
res4b_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_27[0][0]
bn4b_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4b_branch2c[0][0]
add_9 (Add)	(None, 14, 14, 1024)	0	bn4b_branch2c[0][0] activation_25[0][0]
activation_28 (Activation)	(None, 14, 14, 1024)	0	add_9[0][0]
res4c_branch2a (Conv2D)	(None, 14, 14, 256)	262400	activation_28[0][0]
bn4c_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024	res4c_branch2a[0][0]
activation_29 (Activation)	(None, 14, 14, 256)	0	bn4c_branch2a[0][0]
res4c_branch2b (Conv2D)	(None, 14, 14, 256)	590080	activation_29[0][0]
bn4c_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024	res4c_branch2b[0][0]
activation_30 (Activation)	(None, 14, 14, 256)	0	bn4c_branch2b[0][0]
res4c_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_30[0][0]
bn4c_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4c_branch2c[0][0]
add_10 (Add)	(None, 14, 14, 1024)	0	bn4c_branch2c[0][0] activation_28[0][0]
activation_31 (Activation)	(None, 14, 14, 1024)	0	add_10[0][0]
res4d_branch2a (Conv2D)	(None, 14, 14, 256)	262400	activation_31[0][0]
bn4d_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024	res4d_branch2a[0][0]
activation_32 (Activation)	(None, 14, 14, 256)	0	bn4d_branch2a[0][0]
res4d_branch2b (Conv2D)	(None, 14, 14, 256)	590080	activation_32[0][0]
bn4d_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024	res4d_branch2b[0][0]

activation_33 (Activation)	(None, 14, 14, 256)	0	bn4d_branch2b[0][0]
res4d_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_33[0][0]
bn4d_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4d_branch2c[0][0]
add_11 (Add)	(None, 14, 14, 1024)	0	bn4d_branch2c[0][0] activation_31[0][0]
activation_34 (Activation)	(None, 14, 14, 1024)	0	add_11[0][0]
res4e_branch2a (Conv2D)	(None, 14, 14, 256)	262400	activation_34[0][0]
bn4e_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024	res4e_branch2a[0][0]
activation_35 (Activation)	(None, 14, 14, 256)	0	bn4e_branch2a[0][0]
res4e_branch2b (Conv2D)	(None, 14, 14, 256)	590080	activation_35[0][0]
bn4e_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024	res4e_branch2b[0][0]
activation_36 (Activation)	(None, 14, 14, 256)	0	bn4e_branch2b[0][0]
res4e_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_36[0][0]
bn4e_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4e_branch2c[0][0]
add_12 (Add)	(None, 14, 14, 1024)	0	bn4e_branch2c[0][0] activation_34[0][0]
activation_37 (Activation)	(None, 14, 14, 1024)	0	add_12[0][0]
res4f_branch2a (Conv2D)	(None, 14, 14, 256)	262400	activation_37[0][0]
bn4f_branch2a (BatchNormalizati	(None, 14, 14, 256)	1024	res4f_branch2a[0][0]
activation_38 (Activation)	(None, 14, 14, 256)	0	bn4f_branch2a[0][0]
res4f_branch2b (Conv2D)	(None, 14, 14, 256)	590080	activation_38[0][0]
bn4f_branch2b (BatchNormalizati	(None, 14, 14, 256)	1024	res4f_branch2b[0][0]

activation_39 (Activation)	(None, 14, 14, 256)	0	bn4f_branch2b[0][0]
res4f_branch2c (Conv2D)	(None, 14, 14, 1024)	263168	activation_39[0][0]
bn4f_branch2c (BatchNormalizati	(None, 14, 14, 1024)	4096	res4f_branch2c[0][0]
add_13 (Add)	(None, 14, 14, 1024)	0	bn4f_branch2c[0][0] activation_37[0][0]
activation_40 (Activation)	(None, 14, 14, 1024)	0	add_13[0][0]
res5a_branch2a (Conv2D)	(None, 7, 7, 512)	524800	activation_40[0][0]
bn5a_branch2a (BatchNormalizati	(None, 7, 7, 512)	2048	res5a_branch2a[0][0]
activation_41 (Activation)	(None, 7, 7, 512)	0	bn5a_branch2a[0][0]
res5a_branch2b (Conv2D)	(None, 7, 7, 512)	2359808	activation_41[0][0]
bn5a_branch2b (BatchNormalizati	(None, 7, 7, 512)	2048	res5a_branch2b[0][0]
activation_42 (Activation)	(None, 7, 7, 512)	0	bn5a_branch2b[0][0]
res5a_branch2c (Conv2D)	(None, 7, 7, 2048)	1050624	activation_42[0][0]
res5a_branch1 (Conv2D)	(None, 7, 7, 2048)	2099200	activation_40[0][0]
bn5a_branch2c (BatchNormalizati	(None, 7, 7, 2048)	8192	res5a_branch2c[0][0]
bn5a_branch1 (BatchNormalizatio	(None, 7, 7, 2048)	8192	res5a_branch1[0][0]
add_14 (Add)	(None, 7, 7, 2048)	0	bn5a_branch2c[0][0] bn5a_branch1[0][0]
activation_43 (Activation)	(None, 7, 7, 2048)	0	add_14[0][0]
res5b_branch2a (Conv2D)	(None, 7, 7, 512)	1049088	activation_43[0][0]
bn5b_branch2a (BatchNormalizati	(None, 7, 7, 512)	2048	res5b_branch2a[0][0]
activation_44 (Activation)	(None, 7, 7, 512)	0	bn5b_branch2a[0][0]

```
res5b_branch2b (Conv2D)          (None, 7, 7, 512)    2359808     activation_44[0][0]
__________________________________________________________________________________________________
bn5b_branch2b (BatchNormalizati (None, 7, 7, 512)    2048        res5b_branch2b[0][0]
__________________________________________________________________________________________________
activation_45 (Activation)       (None, 7, 7, 512)    0           bn5b_branch2b[0][0]
__________________________________________________________________________________________________
res5b_branch2c (Conv2D)          (None, 7, 7, 2048)   1050624     activation_45[0][0]
__________________________________________________________________________________________________
bn5b_branch2c (BatchNormalizati (None, 7, 7, 2048)   8192        res5b_branch2c[0][0]
__________________________________________________________________________________________________
add_15 (Add)                     (None, 7, 7, 2048)   0           bn5b_branch2c[0][0]
                                                                  activation_43[0][0]
__________________________________________________________________________________________________
activation_46 (Activation)       (None, 7, 7, 2048)   0           add_15[0][0]
__________________________________________________________________________________________________
res5c_branch2a (Conv2D)          (None, 7, 7, 512)    1049088     activation_46[0][0]
__________________________________________________________________________________________________
bn5c_branch2a (BatchNormalizati (None, 7, 7, 512)    2048        res5c_branch2a[0][0]
__________________________________________________________________________________________________
activation_47 (Activation)       (None, 7, 7, 512)    0           bn5c_branch2a[0][0]
__________________________________________________________________________________________________
res5c_branch2b (Conv2D)          (None, 7, 7, 512)    2359808     activation_47[0][0]
__________________________________________________________________________________________________
bn5c_branch2b (BatchNormalizati (None, 7, 7, 512)    2048        res5c_branch2b[0][0]
__________________________________________________________________________________________________
activation_48 (Activation)       (None, 7, 7, 512)    0           bn5c_branch2b[0][0]
__________________________________________________________________________________________________
res5c_branch2c (Conv2D)          (None, 7, 7, 2048)   1050624     activation_48[0][0]
__________________________________________________________________________________________________
bn5c_branch2c (BatchNormalizati (None, 7, 7, 2048)   8192        res5c_branch2c[0][0]
__________________________________________________________________________________________________
add_16 (Add)                     (None, 7, 7, 2048)   0           bn5c_branch2c[0][0]
                                                                  activation_46[0][0]
__________________________________________________________________________________________________
activation_49 (Activation)       (None, 7, 7, 2048)   0           add_16[0][0]
__________________________________________________________________________________________________
avg_pool (AveragePooling2D)      (None, 1, 1, 2048)   0           activation_49[0][0]
__________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 2048)         0           avg_pool[0][0]
==================================================================================================
Total params: 23,587,712
Trainable params: 23,534,592
```

```
Non-trainable params: 53,120
_________________________________________________________________
```

C 部分核心代码和解释

C.1 语言模型构建

```
from PIL import Image
from keras.layers import Input
from keras.models import Model
from keras.layers import RepeatVector, Embedding, LSTM, TimeDistributed, Dense,
Activation, Concatenate
from keras.utils import plot_model

# 这些参数来自于对于数据的预处理
INPUT_DIM = 2048  # 预训练后得到的图像输入。根据采用的预训练模型来确定和变换，记为 D
# VGG - 16 是 4096, Inception V3 是 2048
VOCAB_SIZE = 7709  # 7709（不区分大小写） 8258， 记为 V
WORDVEC_DIM = 512  # 用来表示特定一个单词的向量表示（可以是 one - hot 标识），其他表示
# 也可以视作是 one - hot 向量和某个参数矩阵相乘的结果。记为 W.有的地方表示成
EMBEDDING_DIM
EMBEDDING_DIM = WORDVEC_DIM
HIDDEN_DIM = 512   # 隐藏状态的维度，记为 H
MAX_LEN = 40       # 时间序列的长度（描述的最大长度），记为 T

BATCH_SIZE = 128    # 采样 N 个构成一个 batch，记为 N
NUM_EPOCHES = 50   # 对全部训练数据跑 num_epoches 遍

def create_model(input_dim=INPUT_DIM,
                 max_len=MAX_LEN,
                 vocab_size=VOCAB_SIZE,
                 embedding_dim=EMBEDDING_DIM,
                 hidden_dim=HIDDEN_DIM,
                 is_compiled=False,
                 show_detail=False,

                 ):

    # 一种写法
    # image_input = Input(shape = (input_dim, ))
    # image_x1 = Dense(embedding_dim, activation = 'relu')(image_input)
    # image_x1 = RepeatVector(max_cap_len)(image_x1)
```

```
    # image_model = Model(inputs = [image_input], outputs = image_x1)
    # print("Image Model created!")
    #
    # caption_input = Input(shape = (max_cap_len, ), name = 'input')
    # caption_x2 = Embedding(vocab_size, hidden_dim, input_length =
max_cap_len)(caption_input)
    # caption_x2 = LSTM(hidden_dim, return_sequences = True)(caption_x2)
    # caption_x2 = TimeDistributed(Dense(embedding_dim))(caption_x2)
    # caption_model = Model(inputs = caption_input, outputs = caption_x2)
    # print("Caption Model created!")
    #
    # final_input = Concatenate()([image_model.output, caption_model.output])
    # final_x = LSTM(1000, return_sequences = False)(final_input)
    # final_x = Dense(vocab_size)(final_x)
    # final_x = Activation('softmax')(final_x)
    # final_model = Model(inputs = [image_input, caption_input], outputs = final_x)
    # print("Final_model created!")

    # 第二种写法
    image_input = Input(shape=(input_dim,), name='image_input')
    image_x1 = Dense(embedding_dim, activation='relu', name='dense_1')(image_input)
    image_x1 = RepeatVector(max_len)(image_x1)
    image_model = Model(inputs=[image_input], outputs=image_x1)
    print("Image Model created!")

    caption_input = Input(shape=(max_len,), name='caption_input')
    caption_x2 = Embedding(vocab_size, embedding_dim, input_length=max_len,
name='embedding')(caption_input)
    caption_x2 = LSTM(hidden_dim, return_sequences=True, name='lstm_1')(caption_x2)
    caption_x2 = TimeDistributed(Dense(embedding_dim, name='dense_2'),
name='time_distributed')(caption_x2)
    caption_model = Model(inputs=caption_input, outputs=caption_x2)
    print("Caption Model created!")

    final_input = Concatenate(name='merge')([image_model.output, caption_model.output])
    final_x = LSTM(hidden_dim, return_sequences=False, name='lstm_2')(final_input)
    final_x = Dense(vocab_size, name='dense_3')(final_x)
    final_x = Activation('softmax', name='softmax')(final_x)
    final_model = Model(inputs=[image_input, caption_input], outputs=final_x)
    print("Final_model created!")

    plot_model(final_model, 'named_final_model.png', show_layer_names=True,
```

```
show_shapes=True)
    Image.open('named_final_model.png')

    if show_detail:
        final_model.summary()
    if is_compiled == False:
        return final_model
    else:
        final_model.compile(loss='categorical_crossentropy',
                            optimizer='rmsprop',  # 随机梯度下降
                            metrics=['accuracy'])
        return final_model

tell_model = create_model()
```

C.2 模型训练

```
import pandas as pd
# Image.open('image/caption_model.png')

def data_generator(batch_size, all_features, word_to_idx):
    partial_caps = []  # 作为输入的描述词汇向量矩阵
    next_words = []  # 每个时间步上输出的单词向量 矩阵
    images = []  # 图像特征向量 矩阵

    print('Generating data.....')
    df = pd.read_csv('cache/flickr8k_train_dataset.txt', delimiter='\t') # 这里目前是从文件中读入，之后改成传入参数
    num_samples = df.shape[0]
    imgs = []  # 训练集数据 id 组成的字符串列表（6000*5）
    caps = []  # 训练集数据描述组成的字符串列表(6000*5)
    iter = df.iterrows()

    for i in range(num_samples):
        x = iter.__next__()
        imgs.append(x[1][0])
        caps.append(x[1][1])

    count = 0
    while True:
        for j, cap in enumerate(caps):
```

```
            current_image = all_features[imgs[j]]  # 其实上当前图片的特征向量
            current_cap_words = cap.split()
            current_cap_len = len(current_cap_words)
            for i in range(current_cap_len - 1):  # 输入是除去<end>标记之外
                count += 1

                partial = [word_to_idx[word] for word in current_cap_words[:i + 1]]  # 其实就是将描述映射为索引值，上面已经实现
                partial_caps.append(partial)

                # 创建一个 One-hoe 矩阵
                one_hot = np.zeros(VOCAB_SIZE)
                next_cap_word = current_cap_words[i + 1]
                one_hot[word_to_idx[next_cap_word]] = 1  # 下一个
                next_words.append(one_hot)

                images.append(current_image)

                if count >= batch_size:
                    next_words = np.asarray(next_words)
                    images = np.asarray(images)
                    partial_caps = sequence.pad_sequences(partial_caps, maxlen=MAX_LEN,
                                                          padding='post')  # keras 只能处理定长序列,不够的填充

                    yield [[images, partial_caps], next_words]  # 返回生成器
                    # 释放空间
                    partial_caps = []
                    next_words = []
                    images = []
                    count = 0


def train(batch_size, total_samples, epochs, caption_model):
    file_name = 'cache/resnet50_weights_improvement_{epoch:02d}.hdf5'  # 保存模型的权重
    checkpoint = ModelCheckpoint(file_name, monitor='loss', verbose=1, save_best_only=True, mode='min')  # 检查点
    tfboard_callback = TensorBoard(log_dir="G:/model_logs",  # 一定要对,这里用 /
                                   # histogram_freq=1,
                                   write_graph=True,
                                   write_images=True)
    callbacks_list = [checkpoint, tfboard_callback]
```

```
    print('Ready.....')
    print('Training...')
    % time
    caption_model.fit_generator(generator=data_generator(batch_size=BATCH_SIZE,
all_features=all_resnet50_features_new,

word_to_idx=train_word_to_idx),
                                steps_per_epoch=total_samples / BATCH_SIZE,
epochs=NUM_EPOCHES, verbose=2,
                                callbacks=callbacks_list)
    try:

caption_model.save('result/models/resnet50_cap_bs{0}_ep{1}.h5'.format(BATCH_SIZE,
NUM_EPOCHES), overwrite=True)

caption_model.save_weights('result/models/resnet50_cap_weight_bs{0}_ep{1}.h5'.format(BA
TCH_SIZE, NUM_EPOCHES),
                                    overwrite=True)
        print("Training complete!\n")
    except Exception as e:
        print("Error in saving model.")
        print(e)

train(batch_size=BATCH_SIZE,
      total_samples=total_samples,
      epochs=NUM_EPOCHES,
      caption_model=caption_model
     )
```

C.3 描述生成

```
# 基于贪心的描述生成
def gen_caption(image_url, image_filename, model=caption_model):
    start_word = ['<start>']
    while True:
        partical_caps = [train_word_to_idx[w] for w in start_word]
        partical_caps = sequence.pad_sequences([partical_caps], maxlen=MAX_LEN,
padding='post')
        test_feature = all_features[ image_filename ]
        pred = model.predict( [ np.array([test_feature]), np.array(partical_caps)])  #
```

```
广播机制
        word_pred = train_idx_to_word[np.argmax(pred[0])]
        start_word.append(word_pred)

        if word_pred == '<end>' or len(start_word) >MAX_LEN:
            break
    gen_image_caption = ' '.join(start_word[1:-1])
    # print(gen_image_caption)
    return gen_image_caption

# 基于 beamsearch 的生成
def beam_search_predictions(image_url, image_filename, beam_index=3,
final_model=caption_model):
    start = [word_to_idx["<start>"]]

    start_word = [[start, 0.0]]

    while len(start_word[0][0]) < max_len:
        temp = []
        for s in start_word:
            par_caps = sequence.pad_sequences([s[0]], maxlen=max_len, padding='post')
            e = all_features[image_filename]
            preds = final_model.predict([np.array([e]), np.array(par_caps)])

            word_preds = np.argsort(preds[0])[-beam_index:]

            for w in word_preds:
                next_cap, prob = s[0][:], s[1]
                next_cap.append(w)
                prob += preds[0][w]
                temp.append([next_cap, prob])

        start_word = temp
        start_word = sorted(start_word, reverse=False, key=lambda l: l[1])
        start_word = start_word[-beam_index:]

    start_word = start_word[-1][0]
    intermediate_caption = [idx_to_word[i] for i in start_word]

    final_caption = []

    for i in intermediate_caption:
```

```
        if i != '<end>':
            final_caption.append(i)
        else:
            break

    final_caption = ' '.join(final_caption[1:])
    return final_caption
```

C.4 模型测试

```
def test_model_on_image(test_image_url, test_image_filename, caption_model):
    title = ''
    with open('result/{0}.txt'.format(test_image_filename), 'a') as f:
        tmp_max_search = gen_caption(test_image_url, test_image_filename,
caption_model)
        beam_width = [3, 7, 9, 20]
        for beam_index in beam_width:
            tmp_beam_search = beam_search_predictions(test_image_url,
test_image_filename, beam_index = beam_index, final_model=caption_model)
            f.write('Beam Search, k={0}:'.format(beam_index)+tmp_beam_search)
            f.write('\n')
        f.write('Normal Max search:'+tmp_max_search)
        f.write('\n')
        f.write('\n')
        title+= 'Greedy: {0} \n'.format(tmp_max_search)
        print ('Normal Max search:',tmp_max_search)
        for beam_index in beam_width:
            tmp_beam_search =
beam_search_predictions(test_image_url,test_image_filename, beam_index = beam_index,
final_model=caption_model)
            # print('Beam Search, k={0}:'.format(beam_index), tmp_beam_search)
            title+='Beam k={0}: '.format(beam_index)+tmp_beam_search+'\n'
        print()
    return title

def show_and_tell(images_urls,images_filenames, a, b, n, save_to):
    for i in range(n):
        r = random.randint(a, b)
        tmp_test_image_url = images_urls[r]
        tmp_test_image_filename = images_filenames[r]
        cap_title = test_model_on_image(tmp_test_image_url, tmp_test_image_filename,
```

```
caption_model)
        # tmp_x = preprocess(tmp_test_image_url)
        # print(type(tmp_x), tmp_x.shape)
        plt.xticks([]) # 关闭刻度
        plt.yticks([])
        tmp_x = mpimg.imread(tmp_test_image_url)
        plt.imshow(tmp_x)
        plt.xlabel(cap_title, fontsize=14)
        plt.savefig('{0}/{1}'.format(save_to, tmp_test_image_filename))
        plt.show()
```

D 图索引

D 表索引