Python在机器学习中的应用

学 生： 周轶轩

学 号： 14122621

指导老师： 陈杰

专 业： 电子信息工程

完成年月： 2017年6月

# 摘要

在当今大数据时代，机器学习是计算机科学领域中前沿的研究热点，已在语音识别、自然语言处理、网络搜索、智能机器人、推荐系统以及计算机视觉等复杂系统中崭露头角。在光学字符识别（OCR）领域，机器学习技术由于不需要任何字符结构方面的知识，并以准确率上的显著优势，已经被开发人员用来取代传统技术。

本课题主要基于Google机器学习框架TensorFlow和IM2LATEX-100K数据集，结合卷积神经网络CNN和循环神经网络RNN的方法，使用Python开发一个图像翻译系统，主要功能是识别印刷形式的数学表达式图像，并将其翻译转换成可编辑的LaTeX 语言格式的数学表达式，以便使用者进行编辑和修改。该系统不仅使用到了机器学习技术，也涉及到图像处理方面的相关知识。

**关键词：**人工智能，机器学习，卷积神经网络，循环神经网络，图像处理

# Abstract

In today's large data era, machine learning is a research focus in the forefront of computer science, and has emerged in complex systems such as speech recognition, natural language processing, web search, intelligent robots, recommendation systems and computer vision. In the field of optical character recognition (OCR), machine learning technology has been used by developers to replace traditional technology because it does not require any knowledge of character structure and has a significant advantage in accuracy.

Based on the Google machine learning framework TensorFlow and IM2LATEX-100K dataset, this topic combined convolution neural network CNN and recurrent neural network RNN with the use of Python to develop an image translation system whose the main function is to identify the mathematical expression images and convert it into editable LaTeX format which is convenient for users to edit and modify. The system not only uses the machine learning technology, but also applied image processing related knowledge.

**Key words:** Artificial Intelligence, Machine Learning, Convolutional Neural Network, Recurrent Neural Network, Image Processing.

目录

[摘要 1](#_Toc485031193)

[Abstract 2](#_Toc485031194)

[第一章 绪论 5](#_Toc485031195)

[第二章 技术背景与需求分析 7](#_Toc485031196)

[2.1 机器学习 7](#_Toc485031197)

[2.1.1 人工智能与机器学习 7](#_Toc485031198)

[2.1.2 机器学习的分类 8](#_Toc485031199)

[2.1.3 机器学习的算法 10](#_Toc485031200)

[2.1.4 机器学习的评估 12](#_Toc485031201)

[2.2卷积神经网络CNN 12](#_Toc485031202)

[2.2.1 卷积（Convolution） 13](#_Toc485031203)

[2.2.2 池化（Pooling） 14](#_Toc485031204)

[2.2.3 ReLU 14](#_Toc485031205)

[2.2.4 卷积神经网络的结构 15](#_Toc485031206)

[2.2.5 Softmax回归 16](#_Toc485031207)

[2.3 循环神经网络RNN 17](#_Toc485031208)

[2.3.1 LSTM 18](#_Toc485031209)

[2.3.2 编码器与解码器 19](#_Toc485031210)

[2.4 项目概述 20](#_Toc485031211)

[2.4.1 Python与机器学习 20](#_Toc485031212)

[2.5项目可行性分析 21](#_Toc485031213)

[2.5.1 需求可行性 21](#_Toc485031214)

[2.5.2 技术可行性 22](#_Toc485031215)

[2.6 本章小结 22](#_Toc485031216)

[第三章 数学表达式翻译系统实施方案 23](#_Toc485031217)

[3.1 开发环境概述 23](#_Toc485031218)

[3.1.1 Anaconda科学计算环境概述 23](#_Toc485031219)

[3.1.2 CUDA概述 23](#_Toc485031220)

[3.1.3 PyQt概述 24](#_Toc485031221)

[3.2 TensorFlow概述 24](#_Toc485031222)

[3.3 LaTeX概述 25](#_Toc485031223)

[3.4 IM2LATEX-100K数据集概述 26](#_Toc485031224)

[3.4.1 数据集预处理 27](#_Toc485031225)

[3.5 模型概述 28](#_Toc485031226)

[3.5.1 卷积神经网络 29](#_Toc485031227)

[3.5.2 行编码器 30](#_Toc485031228)

[3.5.3 解码器 30](#_Toc485031229)

[3.6 用户界面设计 31](#_Toc485031230)

[3.7 本章小结 33](#_Toc485031231)

[第四章 方案实现 34](#_Toc485031232)

[4.1 模型的TensorFlow实现 34](#_Toc485031233)

[4.1.1 卷积神经网络 34](#_Toc485031234)

[4.1.2 循环神经网络 36](#_Toc485031235)

[4.2 用户界面实现 39](#_Toc485031236)

[4.2.1 界面框架实现 40](#_Toc485031237)

[4.2.1 界面功能实现 42](#_Toc485031238)

[第五章 模型准确率比较与软件测试 43](#_Toc485031239)

[第六章 总结与展望 45](#_Toc485031240)

[致谢 46](#_Toc485031241)

[参考文献 47](#_Toc485031242)

[附录一 49](#_Toc485031243)

[英译中原文 49](#_Toc485031244)

[英译中译文 63](#_Toc485031245)

# 第一章 绪论

机器学习（Machine Learning），是实现人工智能的一种途径。机器学习的核心思想是使用算法来分析数据，从中学习，然后对现实世界中的事件做出预测和决定。不同于解决具体任务和硬编码的传统软件程序，机器学习是“训练”大量数据，并学习如何通过各种算法从数据中完成任务。

近几年来，随着大数据技术的发展，机器学习技术正逐渐成为IT领域最火热的研究领域之一，北美IT巨头FLAAG（Facebook、LinkedIn、Amazon、Apple以及Google）均投入大量资源开发研究机器学习应用[1]。

Python，是一种面向对象的解释性计算机编程语言。它包含一套完整的功能标准库，可以轻松应对大部分常见任务。它的语法简洁明了，并独创性地使用缩进来定义语句块。目前，绝大多数机器学习框架都支持并倾向于Python，诸如著名的顶级机器学习框架Theano、TensorFlow和Scikit-learn，还有科学计算相关的库NumPy、SciPy、NLTK等等。因为Python强大的第三方支持特性，它被用来链接其他编程语言使用的各种模块。

在光学字符识别（Optical Character Recognition，OCR）领域，机器学习技术正越来越多的替代传统OCR来将纸质、手写文档转换为电子文档。与此同时，在学术界愈发流行的电子形式文档也给文档的纸质-电子格式转换带来了更高标准的要求。相比较于用自然语言书写的传统文本，学术论文往往包含许多当前OCR技术难以识别的数学公式、符号等。此外，在电子文档中输入公式也是一件令人困扰的工作：尽管当下有诸如LaTeX这样的排版系统帮助使用者能够流畅的创建数学公式以及生成文档，创建纷繁复杂的数学公式仍然不方便并且极易犯错。因此，学术工作者期望找到一种解决方案，以帮助他们能够既享受着电子文档的优点，又可以利用手写文档的便利。

光学字符识别（OCR）常用于从图像中识别自然语言，然而，早在20世纪60年代，Anderson就有研究将图像转换成能够定义文本本身及其显示的标记语义的结构化语言[2]。这项工作的主要重点是针对数学表达式的OCR技术，以及如何处理表示方面如sub和superscript符号，特殊符号和嵌套分数[3][4]。1998年，Miller和Viola提出了一个系统，该系统能够有效结合字符分割和底层数学布局语言[5]。这个系统最杰出的实例就是由Suzuki等人于2003发布的INFTY系统[6]，用于转换印刷体数学表达式到LaTeX和其他标记格式。

近来，随着深度学习模型在图像处理和文本数据两个领域中的发展，OCR也再次成为研究热点。在一些领域诸如手写识别（Handwriting Recognition）[7]、自然场景识别（Natural Scenes Recognition）[8][9]和图像翻译（Image Caption Generation）[10][11]里，结合深度学习的OCR技术已经取得了显著成果。从理论上的来说，这些系统都学习一个输入图像抽象的编码表示，然后解码得到文本输出。最重要的是，这些系统完全是基于数据驱动的，这也令它们能够广泛适应于其他数据集。

基于数据驱动的神经网络模型为图像识别任务提供了更多的解决方案，在当今信息时代，以电子形式存储的学术论文中拥有海量的数学表达式图像信息，然而当前的OCR系统还无法有效地识别数学表达式图像。为此，本文提出了一个基于机器学习技术，能够将印刷体数学表达式图像翻译成可编辑的LaTeX源码格式的系统。

本课题主要基于Google机器学习框架TensorFlow和IM2LATEX-100K数据集，结合卷积神经网络CNN和循环神经网络RNN的方法，使用Python编程语言编写开发一个图像翻译系统，主要功能是识别印刷形式的数学表达式图像，并将其翻译为LaTeX源代码。该系统不仅使用到了机器学习技术，也涉及到图像处理方面的相关知识。将数字图像格式的数学表达式转换成可编辑的LaTeX 语言格式的数学表达式，可以方便使用者进行数学表达式的编辑和修改。

本文的主要工作和内容组织结构主要包括：

第二章：介绍了机器学习与人工智能的历史、研究背景和意义、详细阐述了卷积神经网络CNN、循环神经网络RNN的基本原理和模型结构，以及对本项目数学表达式翻译系统的概述和可行性分析。

第三章：对项目的开发环境，使用的机器学习框架TensorFlow和用户界面设计PyQt进行了概述。详细介绍了实现数学表达式翻译系统的项目方案和神经网络的模型结构。

第四章：详细阐述了数学表达式翻译系统的实现，从使用TensorFlow构建模型到用户界面的实现，给出了具体的代码实现过程。

第五章：对数学表达式翻译系统进行运行测试，与市面上流行的传统OCR软件进行准确率上的比较，给出结果。

第六章：总结毕业设计的收获和不足，表达对未来的展望。

# 第二章 技术背景与需求分析

## 2.1 机器学习

### 2.1.1 人工智能与机器学习

人工智能（Artificial Intelligence，AI）是计算机科学（Computer Science）的一个分支，它通过模仿人类智能的方式去解决一些计算机和工程领域上的问题，该领域的研究包括智能机器人、语音识别、图像翻译、自然语言处理和专家系统等[12]。

人工智能自1956年于美国达特茅斯大学（Dartmouth University）诞生以来，在这短短60年内快速发展，得到无数不同专业背景的学者以及企业家和各国政府的空前重视，是一个涵盖众多学科（如计算机科学、电子信息学、信息论、生物行为学、心理学、语言学、哲学等[13]）且学科交叉性很强的一类技术领域。当今，该技术已普遍使用于语音识别（Speech Recognition）、自然语言处理（Natural Language Processing）、网络搜索、智能机器人、推荐系统以及计算机视觉等领域。

而机器学习（Machine Learning，ML），是人工智能的一个分支。人工智能的研究脉络是从“推理”出发，以“知识”为方向，最后以“学习”为重点。而机器学习则是人工智能的一个实现方法、一个子类，也就是通过将机器学习作为去解决人工智能问题的方法。机器学习在这三十多年的发展中已成为一门多领域交叉学科，涉及统计学、概率论、凸分析、逼近论、计算复杂性理论等多门学科。通俗的来说，机器学习理论是设计和分析一些能够使计算机（软件）自动“学习”和“领悟”的算法。确切的来讲，机器学习算法是基于大数据进行分析提取建立规则，并利用所建立的规则对新数据进行预测的算法。实际上，机器学习算法中含有大量的统计学知识和理论，因此机器学习与统计学之间的联系十分紧密，也有一些人称机器学习为统计学习理论。

机器学习的历史发展大致分为三个阶段。第一阶段为上世纪50年代至60年代的婴儿期，这一段时期受益于计算机科学技术的发展，使得于上世纪40年代提出的神经网络模型具备了实现的可能，也因此衍生出进化学习和判别函数法等机器学习算法，当时比较有代表性的是被称作“机器学习之父”阿瑟·塞缪尔(Arthur Lee Samuel)的下棋程序。但这段时期的算法的学习成果很有限，还远远不能达到实际应用的需求。第二阶段为上世纪70年代，在经历了令人兴奋的萌芽期之后，相关研究学者意识到许多机器学习领域的障碍以及与实际应用之间的差距，首当其冲的是神经网络学习被证明存在理论上的缺陷以后，机器学习方面相关研究进入一个低潮期，政府和企业在这方面的投资大大减少。第三阶段为上世纪80年代到现如今的复兴时期，得益于大数据技术的发展，这一阶段中机器学习的概念开始扩展基于数据领域驱动，除神经网络之外，学界也提出了多种学习模型和学习算法，这些模型与算法在一些应用中获得显著成果，也因此重燃了学者的研究热情。机器学习领域的研究再一次地兴起并进入到一个新的发展阶段。

当下，机器学习技术已普遍应用于数据挖掘（Data Mining）、计算机视觉（Computer Vision）、自然语言处理、生物特征识别（BIOMETRICS）、搜索引擎（Search Engine）、医学诊断、识别信用卡欺诈、金融市场分析、DNA测序（DNA Sequencing）、语音和模式识别、战略游戏和智能机器人等领域。2016年3月，由谷歌Deep Mind公司开发的智能围棋机器人阿尔法狗（AlphaGO）以4:1的总比分战胜围棋世界冠军李世石的赛果引起了社会上对人工智能与机器学习的广泛关注。而最近在中国浙江乌镇举行的围棋峰会上，AlphaGO以3：0的大比分战胜现围棋世界第一选手柯洁更是让人工智能与机器学习话题成为了大众茶余饭后的热点。

训练数据

机器学习算法

模型

新数据

预测

图 1. 典型机器学习流程

### 2.1.2 机器学习的分类

机器学习技术主要分为监督学习（Supervised learning）、无监督学习（Unsupervised learning）以及强化学习（Reinforcement learning）三个类别：

1. 监督学习（Supervised learning）

监督学习是从已给定标签的训练数据中找到规则的一类机器学习任务。训练数据里包含一系列训练样本。在监督学习中，每个样本都是一对输入对象和期望输出。监督学习算法通过分析训练数据找到能够用来预测新数据的规则。训练集中的期望输出是由预先由人为标注的。典型的监督学习算法有分类（classification）和回归（regression）。

1. 无监督学习（Unsupervised learning）

无监督学习是从未给定标签的数据中找到规则能够描述数据的一类机器学习任务。由于样本是没有标签的，因此无监督学习算法无法进行准确率的评估。典型的无监督学习算法有聚类（Clustering）、FP树以及深度学习（Deep Learning）。

1. 强化学习（Reinforcement learning）

强化学习受生物行为学的启发，通过模仿生物学习的过程，不断地试错，从中学习最后找到规律的一类机器学习任务。

实际应用中，机器学习任务主要用来解决以下问题：

分类（Classification）：基于对已有训练数据的分类状态，分类能够预测新数据属于哪个类别。分类还分为二分类（two-class classification）和多分类（multi-class classification）。比如将含有“猫”或“狗”的图片识别出来，分类为“猫”或“狗”，这就是二分类问题。当存在更多类别时（例如预测下一届世界杯冠军是谁），这就是所谓的多分类问题。

回归（Regression）：从一个连续数据集中预测一个值称为回归。例如，基于诸如地理位置、大小和购买时间这类因素去预测房屋价格。

聚类（Clustering）：将一组数据分成不同的聚类，使得在同一聚类下的数据相较于其他聚类下的数据具备更好的同类性。例如，网易云音乐拥有不同聚类下众多的用户，每个聚类下的用户拥有相似的音乐品好。

推荐系统（Recommendation Systems）：基于用户的产品或者服务数据，使用机器学习算法来帮助用户找到适合该用户的新产品或新服务。例如，亚马逊能够基于用户的商品浏览、购买记录，为用户推送相似的流行产品。

异常检测（Anomaly Detection）：识别数据集中不符合已知规则或和其他数据完全不同的异常数据。举例来说，一次异常信用卡消费会在信用卡欺诈检测中被检测为信用卡欺诈；信用卡欺诈拥有大量不同的形式，而可供训练的样本比例非常少。异常检测所使用的策略就是通过学习正常情况下的表现行为以检测出显著不同的异常表现行为。

降维（Dimensionality Reduction）：在某些限定条件下，为了得到一组“不相关”主变量而降低随机变量个数的过程称为降维。

### 2.1.3 机器学习的算法

根据机器学习算法的功能和形式上的相似性，其算法可以分为基于树的算法、基于类的算法和基于神经网络的算法等等。有些算法可以适用于不同的问题情况下，分属于不同的类别，因此这里介绍一些常用的算法分类。

1. 回归算法

回归算法是尝试使用误差的度量来探究一类算法的变量之间的关系。回归算法是统计机器学习的工具。在机器学习领域，常见的回归算法包括：最小二乘法（Ordinary Least Square），逻辑回归（Logistic Regression），逐步式回归（Stepwise Regression），多元自适应回归样条（Multivariate Adaptive Regression Splines）以及本地散点平滑估计（Locally Estimated Scatterplot Smoothing）。

1. 基于案例的算法

基于案例的算法一般用来建模决策模型，该模型通常首先选择一批样本数据，然后根据某些近似值将新数据与样本数据进行比较。通过这种方式来寻找最佳的匹配。因此，基于案例的算法通常被称为“获胜者”或“基于记忆的学习”。常见的算法包括 k-Nearest Neighbor (KNN), 学习矢量量化（Learning Vector Quantization, LVQ），以及自组织映射算法（Self-Organizing Map, SOM）。

1. 决策树学习

决策树算法使用树结构根据数据的属性建立决策模型。决策树模型经常用于解决分类和回归问题。常见的算法包括：分类及回归树（Classification And Regression Tree, CART）， ID3 (Iterative Dichotomiser 3)，C4.5，Chi-squared Automatic Interaction Detection(CHAID), Decision Stump, 随机森林（Random Forest），多元自适应回归样条（MARS）以及梯度推进机（Gradient Boosting Machine, GBM）。

1. 贝叶斯方法

贝叶斯方法是基于贝叶斯定理的一类算法，主要用于解决分类和回归问题。常见算法包括：朴素贝叶斯算法，平均单依赖估计（Averaged One-Dependence Estimators, AODE），以及Bayesian Belief Network（BBN）。

1. 基于核的算法

最有名的基于内核的算法是支持向量机（SVM）。基于内核的算法将输入数据映射到较高阶向量空间。在这些高阶向量空间中，可以更容易地解决一些分类或回归问题。常见的基于核的算法包括：支持向量机（Support Vector Machine, SVM）， 径向基函数（Radial Basis Function, RBF)， 以及线性判别分析（Linear Discriminate Analysis, LDA)等。

（6） 聚类算法

聚类，如回归，有时人们会描述一类问题，有时被描述为一类算法。聚类算法通常以中心或分层方式合并输入数据。因此，聚类算法试图找到数据的基础结构，以便根据最大公分母对数据进行分类。常见的聚类算法包括 k-Means算法以及期望最大化算法（Expectation Maximization, EM）。

1. 关联规则学习

关联规则学习通过查找最能解释数据变量之间关系的规则，可以为大量多变量数据集找到有用的关联规则。常见算法包括 Apriori算法和Eclat算法等。

1. 遗传算法（genetic algorithm）

遗传算法模拟了突变，交换和达尔文自然选择（每个生态环境中的生存）的生物繁殖。它将问题解码为一个称为个体的向量，向量的每个元素称为基因，并用目标函数（对应于自然选择标准）评估组（个体集）中的每个个体。根据评估价值（适应度）选择个体，交换，突变等基因操作，产生新群体。遗传算法在非常复杂和困难的环境中使用，例如具有大量噪声和不相关的数据，事情不断更新，问题目标不能被清楚和精确地定义，并且通过很长的实现过程来确定目前的行为。像神经网络一样，遗传算法研究已经发展成为人工智能的独立分支，其代表是霍勒德（J.H.Holland）。

1. 人工神经网络

生物神经网络的人工神经网络模拟是一种模式匹配算法。通常用于解决分类和回归问题。人工神经网络是机器学习的巨大分支，具有数百种不同的算法。重要的人工神经网络算法包括：感知器神经网络（Perceptron Neural Network）, 反向传递（Back Propagation）， Hopfield网络，自组织映射（Self-Organizing Map, SOM）。

1. 深度学习（Deep Learning）

深度学习算法是人工神经网络的发展，是近几年来十分火热的一个研究方向，特别是国内百度（Baidu）也开始强化深度学习方面的投入，在国内引起了不少关注。随着计算能力今天越来越便宜，深度学习尝试构建更大更复杂的神经网络。许多深度学习算法是处理具有少量未识别数据的大数据集的半监督学习算法。常见的深度学习算法包括：受限波尔兹曼机（Restricted Boltzmann Machine， RBN）， Deep Belief Networks（DBN），卷积神经网络（Convolutional Neural Network）, 堆栈式自动编码器（Stacked Auto-encoders）。

### 2.1.4 机器学习的评估

对于机器学习模型的评估最重要的标准是准确率。机器学习的主要目的就是正确的判断、识别新的数据，学习模型的成功与否可以直接通过测试数据样本分类的正确百分比来加以度量。

除了准确率之外，还有一些指标也可以用来衡量一个机器学习模型。其中一个是效率，学习是要讲究效率的，机器学习下的效率就是能够在最小数据量的情况下获得一定的准确率。此外计算能力，即计算所花费的空间复杂度和时间复杂度，也是一个值得需要考虑的指标。

另外一个标准就是鲁棒性。有时样本或标签中存在噪声，例如用户提供的数据集不准确，或者模型的参数过于主观。这些情况下都会造成机器学习结果的偏差，因此需要模型具备鲁棒性。

## 2.2卷积神经网络CNN

卷积神经网络（Convolutional Neural Network，CNN）于1998年由纽约大学教授Yann LeCun提出，在图像识别领域得到广泛应用[14]。卷积神经网络实质上是一个多层次感知器，其关键机制是使用本地链路，共享权值以减轻权重数量的权重，以优化网络，另外也减少了过度拟合的可能性。

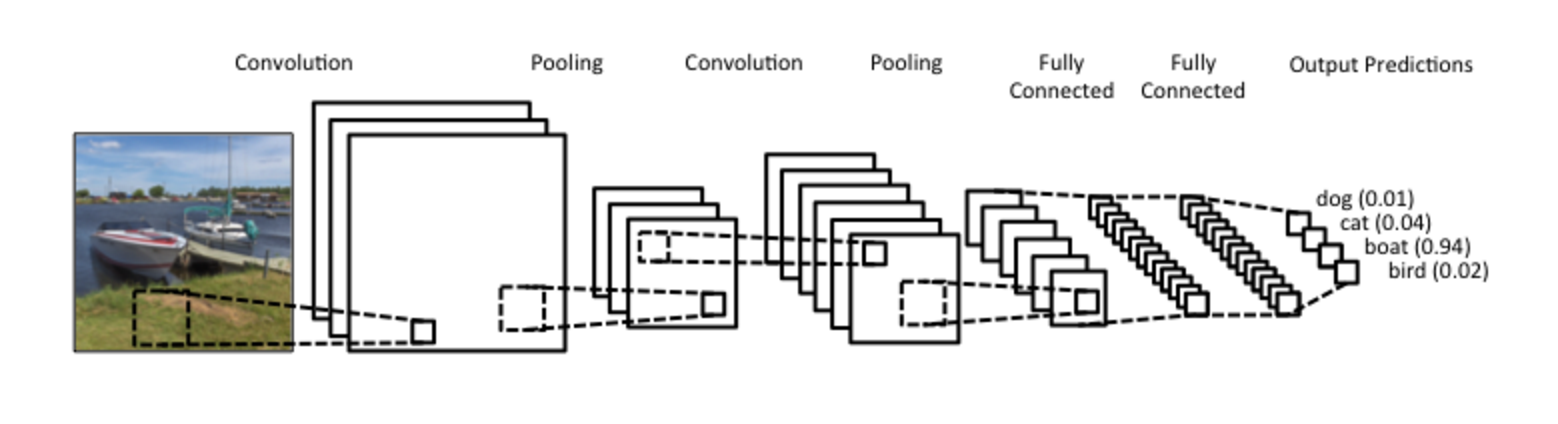


图 2. 一个用于图像分类的典型卷积神经网络示意图

卷积神经网络CNN的基本结构分为两个神经元层，一是卷积层（Convolutional layer），每个神经元的输入和上一层的局部相连接，并提取局部的特征；二是池化层（Pooling layer），作用于计算层的本地敏感度和二次特征提取。该二次特征提取结构降低了特征分辨率并减少了需要优化的参数数量。

卷积神经网络可用于识别，缩放和满足二维或三维图像的扭曲不变性的其他形式。通过学习训练数据获得卷积神经网络的特征提取层参数，从而有效避免了人工特征提取，但从训练数据中学习。另一方面，相同特征图的神经元共享权值减少了网络参数，这比其他神经网络更接近真正的生物神经网络。因此，卷积神经网络在图像处理领域具有独特的优势。权值共享降低了网络的复杂度，多维输入信号可以直接输入到网络中，避免了特征提取和分类中特征重排的过程。

### 2.2.1 卷积（Convolution）

卷积是高等数学中的一种运算，在信号处理中有着广泛的应用。设是上的两个可积分函数，则两者的卷积如下：

该式定义了函数的卷积。

自然界中的物理信号具有其本质特征，换句话说，信号的一部分的统计特性在更大程度上与其他部分相同。例如，在大型分辨率图像中随机选择3×3小片作为样本，并从样本中学习一些特征，那么我们可以从这个3×3样本中学习检测器应用于其他任何区域的特性。特别地，我们可以使用从3×3样本学习的特征检测器对原始大分辨率图像进行卷积运算，从而为大分辨率图像的任何区域获得不同的特征值。如图3是图像的一种卷积形式。

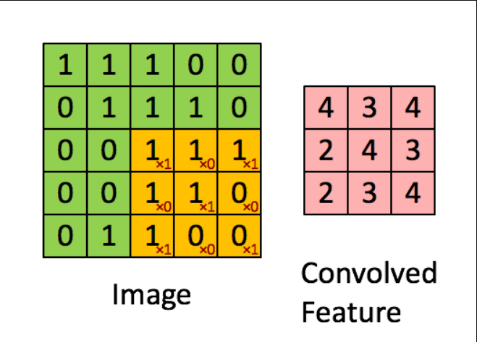


图 3. 图像的卷积操作，左图为原始图像，右图是与3x3核卷积后的特征图

### 2.2.2 池化（Pooling）

卷积运算之所以尤其适用于图像处理领域，是由于图像数据具有“静态性”：图像区域中的有用特征可能同样适用于另一区域。因此，为了表达大分辨率图像，很直观的方法就是统计不同区域的特征，例如计算卷积之后的区域的特征的平均值（或最大值）。这部分统计特征的计算具有非常低的维度，相当于做了一次下采样（Sub-sampling），而且还改进了模型的泛化，不容易过度拟合。这类聚合的操作称为池化（Pooling）。池化分为两种，一种是最大池化（max-pooling），在选定区域中选取最大值作为抽样后的值，另一种是平均值池化（average-pooling），选取选定区域中的平均值作为抽样后的值，这样就可以减少之后全连接时的连接数。

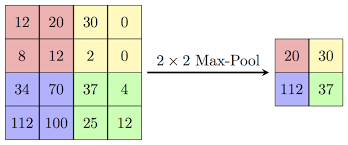


图 4. 2×2最大池化示例

在目标和语音识别等工作中，具有平移不变性毫无疑问是极有益的，因为平移不变性使得发生了位移的语音或图像信号，其数据标记不会发生改变。比如输入信号是图像时，假设图像中的连续区块被选择为池化区域，并且仅通过池化重复相同的卷积核产生的特征，那么这些池化单元就具有局部的平移不变性。这也表示说如果图像发生位移操作，仍旧具有相同的池化特征。

### 2.2.3 ReLU

ReLU（Rectified Linear Unit），线性整流，是一种激励函数，其原理就是“截尾巴”，如图5所示，将负值化零，正值不变。ReLU将所有负值设零之后，留下的正值正是“感兴趣区域”，该区域表示了某个特征。ReLU的作用在于增加卷积神经网络CNN的非线性特性。非线性指的是神经网络的输入和输出之间的对应关系是非线性的。

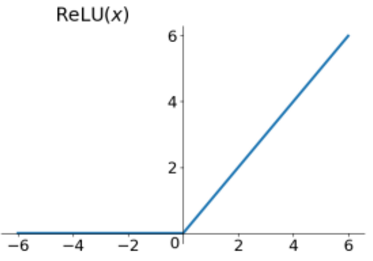


图 5. ReLU函数

另一方面，ReLU减少了参数优化时特征值的复杂性，更便于卷积神经网络CNN进行参数优化。从而，ReLU的出现大大提升了卷积神经网络模型的准确性。

### 2.2.4 卷积神经网络的结构

https://adeshpande3.github.io/assets/Table.png

图 6. 卷积神经网络结构

图6是一个常见的卷积神经网络结构，可以看到，输入之后的第一层是卷积层（Conv），接下来是一个激励函数ReLU，在此基础上继续堆积卷积、池化（Pool），最后是全连接层（Fully Connected layer）。全连接层指的是计算输出需要所有的输入的层。它的作用在于在多次卷积和池化后，输出包含对于输入图像的分布式表示特征，此时就需要利用这些特征建立起更强的特征。当前用于模式识别的卷积神经网络，惯用的做法是在全连接层使用Softmax回归。

当最开始的输入经过所有的层之后，还会有一个反向传播机制，该机制是用来调节卷积神经网络CNN 的参数以达到最优化效果。假设现在是一个图像分类数据，那么输出就是一个所有类别维度的向量，比如有3个类别，那么输出就是3 维的，每个维度上的数据对应了该输入属于此类别的后验概率。

首个取得巨大成功的卷积神经网络结构是LeNet-5[14]，由Yann LeCun提出，其结构如图7所示，LeNet-5的出现使得模式识别技术风靡于金融领域，美国的许多银行使用LeNet-5来识别支票上的手写数字。

LeNet-5深层卷积网络：

1. 输入图像大小为32\*32，卷积核5\*5，卷积后的图像维度，也即是C1层的大小，为28\*28。假设使用6个卷积核，那么就会有6个不相同的C1层，而每个C1层的卷积核是相同的。
2. S2层是池化层。由4个点池化为1个点，池化后的大小为14\*14。
3. C3层的大小为10\*10，C3包含16个5\*5\*J的卷积核：S2是多层的，按照一定顺序组合来进行卷积操作。LeNet-5中给出下面表格，如图7所示：

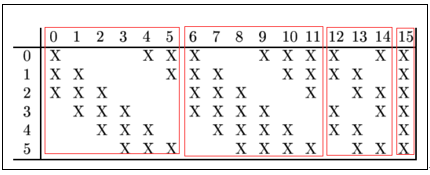


图 7. LeNet-5连接关系表

例如对于C3层第0张特征图，其每一个神经元与S2层的第0张特征图，第l张特征图，第2张特征图，总共与3\*5\*5个神经元相连接。后面依次类推，C3层每一张特征映射图的不同神经元对应的卷积核是相同的，如第0张特征图的卷积核为5\*5\*3，第15张特征图的卷积核为5\*5\*6。

1. S4层是池化层，C5层是卷积层，F6是全连接层，输出层是10维的Softmax。

### 2.2.5 Softmax回归

Softmax回归模型是逻辑回归模型在多分类问题上的自然推广[15]。前文2.1.2节中提到过，在多分类问题里，类别的标签可以取多个值。

在Softmax回归中，相对于逻辑回归所解决的二分类问题，解决的是多分类问题。类别标签*y*可以取*k*个不同的值。也因此，对于训练集，有。

对于既有的输入*x*，假设函数是*x*属于类别*j*的概率值，换句话说，*p*是*x*的每种分类结果可能的概率。因此假设函数的值是一个*k*维向量，假设输出函数的形式如下所示：

其中为模型的参数。是对概率分布进行归一化处理，使得所有概率之和为1。

## 2.3 循环神经网络RNN

循环神经网络（Recurrent Neural Network, RNN）是人工神经网络的一种，它的特点在于神经网络中带有自循环系统，能够保持信息的再利用。

和卷积神经网络类似，循环神经网络也启发于人类行为。试想，人类在思考某样事物时，往往不是从零想起，而是有一个基于上下文的情境系统。比如，当一个人在读一片文章时，是根据上下文去理解文中的每一个词，不会有人不顾上下文去理解一个词。从这个角度上来说，人类的理解是有持续性的。

循环神经网络RNN，顾名思义，就是带有循环（loops）的神经网络。传统的神经网络的工作流程是这样的：通过监督式学习将输入转化为标签输出，这个标签就是类别。但是该网络对每个输入是独立的，也就是说，前一个输入对后一个输入没有影响。但是循环神经网络RNN的输出会回到下一个网络的输入一起进行运算，形成一个循环网络。这样做仿佛神经网络有了记忆，能够使用之前的运算结果对当前的输入造成影响。传统神经网络的一个缺陷在于不能模仿人类理解的持续性，因此，循环神经网络RNN就诞生出来解决这个问题。循环神经网络的意义在于能够准确地对序列输入进行分类。

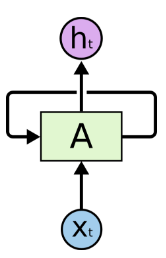


图 8. 单一循环神经网络RNN结构

图8是一个简单的循环神经网络，A，有输入和输出。图中可以看到A循环有一个循环指向自身，该循环使得信息能够当作输入，再次传入该神经网络A中。

循环神经网络RNN的本质和一个普通神经网络无异，它可以当作是一连串的相同的神经网络，每个传递信息给下一个，如图9所示。

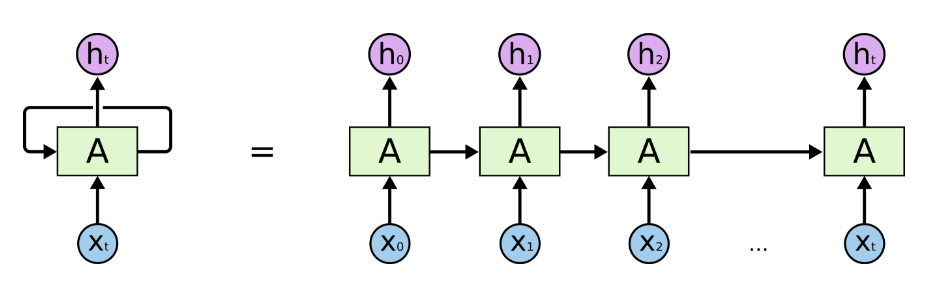


图 9. 循环神经网络RNN原理图

循环神经网络RNN在语音识别，自然语言建模、翻译和图像标注等问题中表现优异。[16][17]

### 2.3.1 LSTM

LSTM（Long Short Term Memory），中文名长短期记忆，是一种特殊形式的循环神经网络，于1997年由Hochreiter和Schmidhuber提出[18]。LSTM的出现解决了循环神经网络RNN一个棘手的问题，那就是长期可靠性问题。长期可靠性问题指的是在循环次数较多的情况下，普通的循环神经网络RNN无法利用到较旧的信息。

具体的来说，普通的循环神经网络采用的是定时反向传播算法（Backpropagation Through Time），其原理就是令循环神经网络RNN平展为前馈神经网络，之后使用反向传播算法（Backpropagation）：

上式表示循环神经网络RNN的目标函数为序列各个时期的损失函数的累加。

式中可以看到，由于循环神经网络存在前后网络链接，因此循环神经网络在循环次数较多的情况时，如果权值接近于0，梯度会衰减直至消失（等于0）；而如果权值大于1，梯度则会递增直至无穷大。这也就是所谓的梯度消失（Vanishing Gradient）和梯度爆炸（Exploding Gradient）问题。

而长短期记忆LSTM则有效避免了这个问题。LSTM的不同之处在于每个神经元（neuron）都有一个权值为1的自连接(self-connection)和三个开关(Gating Unit)，分别是“读”、“写”和“保持”，模型示意图如图10所示。有了开关，LSTM可以通过开关来控制神经元的值。举例来说，当神经元的“保持”打开时，该神经元就不会更新从前一层传递过来的梯度值，而是将自身的梯度值传递到下一层。这样不仅可以避免梯度消失的问题，也就从而解决了序列中的长期可靠性问题。

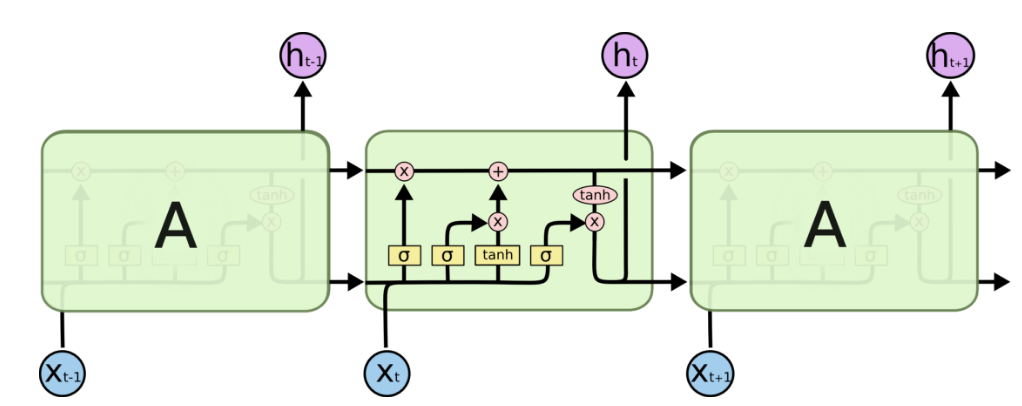


图 10. LSTM模型示意图

### 2.3.2 编码器与解码器

在卷积神经网络中，编码器（Encoder）的作用就是将输入序列编码成一个定长的隐藏状态（Hidden Status），其中，是输入序列，是跟随时间更新的隐藏状态。由得知，当出现新的输入*x*时，就更新为和新输入*x*相关的，而距离当前时间越早输入的序列，在更新后的状态中的权值就越小，因此表现出时间相关性。

在已知隐藏状态序列的情况下，就能够基于进行预测下一个输入:

其中是时间*t*时刻的输出，是基于时刻*t*之前所有的输出得到。的计算依赖于一个非线性方程：

其中，是所有隐藏状态的编码。

同样的，卷积神经网络还可以应用解码器（Decoder），解码器能够将之前通过编码器编码的信息解码方便使用者识别的信息，即

## 2.4 项目概述

“数学表达式翻译系统”是一款基于Google机器学习框架TensorFlow，使用Python语言开发的图像翻译应用程序，它能够识别数学表达式图像文件（png格式），并将其转换为LaTeX源代码。为了完成该系统的实施，需要具备Python编程语言知识，机器学习算法、图像处理、深度学习等方面的知识，才能实现对数据集的训练和验证，以及使用训练好的模型对新数据进行预测等一系列的工作。

### 2.4.1 Python与机器学习

Python, 是一种面向对象、解释型的计算机程序设计语言，是纯粹的自由软件，语法简洁明了，适合于完成各种高层任务。它既可以用来快速开发程序脚本，也可以用来开发大规模的软件。目前有大量的机器学习的框架都偏向于Python应用，诸如著名的顶级机器学习框架Theano、TensorFlow和Scikit-learn。因为目前Python拥有海量的功能强大的包，它也常被程序员昵称为“胶水语言”，被用来将其他编程语言编写的模块（尤其是C/C++）结合使用。

随着NumPy, SciPy, Matplotlib, Pandas等众多科学计算程序库的开发，Python逐渐成为科学计算、绘制高质量的2D以及3D图像的首选语言。相比于科学计算领域最流行的商业软件MATLAB，Python是一门通用的程序设计语言，比MATLAB所采用的脚本语言的应用范围更广泛，具备更多程序库的支持。尽管MATLAB中的许多高级功能和toolbox目前还尚无法被取代的，不过在日常的科研开发之中Python已经可以胜任很大一部分工作。

本项目开发过程中大量使用到了Numpy程序包，NumPy是Python语言的一个扩充程序库。其支持大量高级的维度数组与矩阵运算，除此之外也针对数组运算提供大量的数学函数库支持。

Python语言内置有列表(list)模块用来存储数值，列表可以用来当作数组使用，然而因为列表的子元素可以是任何对象，因此列表中所存储的实际上是对象的指针。因而，如果要存储数组[1,2,3]，就需要有三个整数对象和三个指向它们的指针。对于数值运算来说这种结构显然比较浪费内存和CPU计算时间。

另外，Python还内置一个array模块，array对象和列表list不同，它在无指针的情况下直接存储数值，这方面和C++的一维数组相似。然而因为它无法扩展到多维，同时也缺少运算函数的支持，也不适用于数值运算。

NumPy的出现弥补了Python的缺陷，NumPy提供了两种基本的对象：ndarray（N-dimensional array object）和 ufunc（universal function object）。ndarray是存储单一数据类型的多维数组，而ufunc则是能够对数组进行处理的函数。这样编译器可以直接有效率地操作多维数组的函数与运算符，因此在NumPy上只要能被表示为针对数组或矩阵运算的算法，其运行效率几乎都可以与编译过的等效C语言代码一样快。

## 2.5项目可行性分析

可行性研究与分析对整个项目的开发与实施阶段具有重要实际意义，一般来说从需求可行性和技术可行性两方面对项目的内容展开研究和分析，从而提供对整个项目的开展的指导性方向。

### 2.5.1 需求可行性

二十一世纪以来，随着通信技术的进步以及互联网产业的飞速发展，利用互联网传播和交换信息日益增多。同时，大量的学术论文、文献资料等以前主要以纸质媒体存在的信息媒介目前也已广泛数字化。其中，数学表达式通常是印刷体，以图像形式存在。众所周知，图像格式文件的数据容量比较庞大，编辑、修改不如文本格式方便。对图像形式的数学表达式无法编辑、修改，导致许多学术工作者感到不便，不能充分利用现有资源。而OCR（Optical Character Recognition)技术尽管在识别文字等内容上表现出色，但出于数学表达式复杂的结构以及多样的数学符号，导致其在识别数学表达式上举步维艰。

本课题提出的数学表达式翻译系统是一种全新的图像处理系统，充分利用了计算机技术、图像处理技术以及模式识别和机器学习技术，能够在脱机状态下准确的识别图像中的数学表达式，很好地解决了现阶段OCR系统识别准确率不高的问题。

### 2.5.2 技术可行性

该系统在Linux(Ubuntu)系统中进行开发和调试，使用Anaconda2 64-bit (Python 2.7) & CUDA 8.0 & Cudnn5.1 & TensorFlow0.12框架进行模型的开发和验证、使用PyQt5进行用户界面设计，采用Python语言进行编写设计和开发，按此设计的开发平台能够完全满足需求。

数学表达式翻译系统以深度学习技术为基础，并结合计算机技术、图像处理等相关技术，早在1967年，Anderson便提出将图像转换成结构化语言或定义文本本身及其显示的标记语义学。这项工作的主要重点是OCR用于数学表达式，以及如何处理表示方面如sub和上标符号，特殊符号和嵌套分数。最有效的系统是结合用语法进行专门的字符分割底层的数学布局语言。这个方法的一个最好的例子是INFTY用于转换印刷数学表达式到LaTeX和其他标记格式的系统。

目前市面上已经出现了一些数学公式翻译软件，比如上文提到的由日本开发者Suzuki开发的InftyReader以及美国开发者开发的Mathpix均已取得一定的成功，为本系统的研究和开发提供了有利的技术保证和支持。

## 2.6 本章小结

本章首先介绍了人工智能与机器学习方面的相关历史和背景知识，机器学习在学术界和工业界的应用，概述了机器学习的一些常见算法和类别。之后介绍了卷积神经网络CNN和循环神经网络RNN的相关理论。在卷积神经网络方面，介绍了经典的LeNet-5卷积神经网络结构，阐述了其中的细节。在循环神经网络方面，阐述了其与普通神经网络的区别，并介绍了改进版的循环神经网络LSTM模型。最后对本课题的可行性展开分析和研究。

# 第三章 数学表达式翻译系统实施方案

## 3.1 开发环境概述

本项目在Linux操作系统Ubuntu 16.04 LTS下，使用Anaconda2 64-bit (Python 2.7) & CUDA 8.0 & Cudnn5.1 & TensorFlow0.12 & PyQt5的框架进行开发与调试工作。

### 3.1.1 Anaconda科学计算环境概述

Anaconda由CONTINUUM ANALYTICS公司开发和维护，是一款用于科学计算的Python发行版，同时支持 Linux, Mac, Windows系统。

Anaconda的优越性在于解决了使用Python进行科学计算时最令开发者头疼的包管理和Python版本的兼容问题。因为其提供了包管理与环境管理的功能，可以很方便地解决多版本Python并存、切换以及各种第三方包安装问题。

### 3.1.2 CUDA概述

CUDA（Compute Unified Device Architecture），中文名统一计算架构，是由英伟达（NVIDIA）公司所推出的一款并行计算架构。通过这个架构，开发者能够使用类C语言编写的程序实现在GPU上的运行。该架构的优越性在于能通过利用GPU资源极大地提升程序的计算和运行效率。

支持CUDA的GPU结构如图11所示，在工业领域和学术领域，数据科学家们早已开创性地利用 GPU资源进行机器学习应用上的开发，这些应用包括图像分类、视频分析、语音识别以及自然语言处理等等。 尤其在深度学习（Deep Learning）领域，近几年科学家们在正大力投资发开和研究这方面的应用。

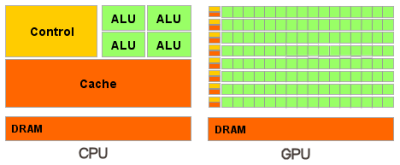


图 11. CPU和GPU的架构比较

目前已经有许多一线的互联网公司将GPU应用于机器学习应用中，与单单使用 CPU 相比，GPU 具有数百上千倍的计算核，与CPU相比，GPU在峰值浮点计算的吞吐量是CPU的十至百倍。因此，GPU计算所提供的强大而高效的并行计算使得人们训练和测试机器学习模型的效率也更高，在一定程度上来说，CPU计算和机器学习是相辅相成的。

### 3.1.3 PyQt概述

PyQt是跨平台GUI工具Qt的Python语言合集，它是一款免费的软件，由英国Riverbank Computing公司开发。与它的父集Qt相同，PyQt具有跨平台特性，可以运行于Microsoft Windows、Mac OS X、Linux以及Unix等多数操作系统上。

PyQt包含了大约440个类型、超过6000个的函数和方法。它的核心思想与Qt相同，采用了信号（signal）和槽（slot）的概念来处理GUI应用中的用户自定义事件。任何Python类型都可以定义信号和槽，并与GUI控件的信号和槽相连接。

PyQt的一大优势在于，其API与Qt极其相似，开发者可以参考Qt的帮助文档进行PyQt的开发。因此，如果开发人员拥有Qt方面的使用经验，他就可以很快地过渡到PyQt平台上。类似的，使用PyQt的程序员，如果同时熟悉C++的话，也能够很快地过渡到Qt平台上。

## 3.2 TensorFlow概述

TensorFlow是一款由Google发布的开源的机器学习框架，允许用户快速、有效地实施各种基于神经网络的算法。TensorFlow 最初由Google DeepMind的研究员和工程师们开发，主要用于机器学习和深度学习上的研究，但这个框架的通用性使其也可广泛使用于其他计算领域。

数据流图用“结点”（nodes）和“线”(edges)的有向图来描述数学计算。“节点” 一般用来表示施加的数学操作，但也可以表示数据输入（feed in）的起点/输出（push out）的终点，或者是读取/写入持久变量（persistent variable）的终点。“线”表示“节点”之间的输入/输出关系。这些数据“线”可以输运“size可动态调整”的多维数据数组，即“张量”（tensor）。TensorFlow的命名来源于本身的运行原理。Tensor（张量）代表N维数组，Flow（流）代表基于data flow graphs（数据流图）的计算，TensorFlow，顾名思义，意味着张量从流图的一端流动到另一端计算过程。

TensorFlow是将复杂的数据结构传送至人工智能神经网中进行分析和处理过程的系统。相比于其他深度学习框架，TensorFlow的不同之处在于一个操作定义为一个表示在流图结构中特定操作的节点，并且不是按序执行。因此，TensorFlow工作流主要分为两个阶段：构图和执行。

1. 构图：这里声明了表示公式和特定功能函数的符号运算，包含卷积、损失（loss）、交叉熵计算（cross entropy calculation）、丢失概率（dropout probabilities）、池化（pooling）和它们构成的符合运算。
2. 执行：数据被喂进流图（graph）中，之前被定义好的模型在可执行环境中运算，称为对话（sessions）。

TensorFlow支持在多设备上高效的并行式计算，这项特性使得该框架具有很强的普适性，也就是说，不同的工程项目只需一些细微的改动，就能够在不同的设备上进行操作。在多核GPU运算里，TensorFlow的并行式计算特性允许不同的设备为超参数优化方案存储变量。

## 3.3 LaTeX概述

LaTeX是一种基于TEX的排版系统，由美国计算机科学家莱斯利·兰伯特（Leslie Lamport）于1985年初期开发，是当今互联网上最著名和使用最广泛的TEX包。通过使用LaTeX语言，即便用户无任何关于排版和程序设计相关的知识，也能够充分发挥TEX系统独有的强大特性，能在短时间内生成高质量的文档。相较于微软发布MS Office中的Word软件，LaTeX的优势在于能够简洁有效地生成复杂表格和数学公式，因而它在生成高印刷质量的科技和数学类文档中表现显著。同样的，LaTeX也适用于生成从简历到完本书籍之类的其他种类的文档。LaTeX有良好的社区氛围，在互联网上有海量的LaTeX模板供使用者参考。

LaTeX源文件是一般的文本文件格式，其中不但包括所要排版的文本，还包括 LaTeX所能识别的排版命令。因此，使用者可以使用任何文本编辑器（最简单的如记事本和写字板）创建 LaTeX源文件。值得一提的是，LaTeX支持DVI和PDF格式的输出，在配合其他软件的情况下可以很容易地生成PNG、JPG等图像格式的输出。

在该项目的实施中，需要使用到LaTeX生成的数学表达式功能，下面简单给出两个示例，方便理解。首先是给出LaTeX代码：\cos (2\theta) = \cos^2 \theta - \sin^2 \theta，该示例是一个三角函数：



图 12. LaTeX生成三角函数表达式图像示例

然后是LaTeX的一串多项式代码：k\_{n+1} = n^2 + k\_n^2 - k\_{n-1}，得到：



图 13. LaTeX生成多项式表达式图像示例

## 3.4 IM2LATEX-100K数据集概述

OpenAI公司发布了一系列的开源合作项目，旨在为机器学习研究者提供合适的课题来进行时间与操作。Im2Latex是其中的一个项目，目的是应用近几年比较火热的注意力模型去完成将PDF格式数学表达式转化为能够生成该公式的LaTeX源代码。于是乎，OpenAI公司为参与者提供了一个名为IM2LATEX-100K的数据集。

顾名思义，IM2LATEX-100K数据集包含了100K张，具体为103,556张不同的数学表达式图像，这些表达式都是从arXiv里的论文集提取而来。其中，每张图像都是固定尺寸，A4大小、分辨率为1654\*2339的PNG图像格式文件，数学表达式的颜色是黑色，其余的背景皆为透明。

该数据集被分为训练、验证与测试三类数据集供机器学习使用，其中训练集包含83,883个公式、验证集9,319个公式和测试集10,354个公式。在一般的机器学习任务中，训练集被用来训练模型，测试集用来测试模型，而验证集被用来确保该模型的性能是能力范围内最好的。这部分工作是通过在训练过程中周期性的使用验证集测试，以调整模型的参数。比方说，观察训练集和验证集准确率的趋势能告诉使用者模型是否过拟合或者欠拟合，于是可以对照信息修改相应的参数以确保模型的性能。

### 3.4.1 数据集预处理

数据集预处理工作主要包括对LaTeX表达式源码的每个字符间插入空格方便使用Python进行索引，以生成LaTeX表达式源码中的相应的语法关键词，以及裁剪一些尺寸过大的图片以满足主机训练的要求。

模型训练有两种选择：一是基于LaTeX表达式的单个字符的模型，还有就是基于LaTeX表达式的单个词语的模型。显然后者的训练效率更高，因此在模型训练之前就要生成LaTeX语法关键词。大致过程就是将源码分割为一个个字符串，然后对每个字符串使用脚本进行关键词匹配，匹配成功的关键词使用Python字典进行索引。

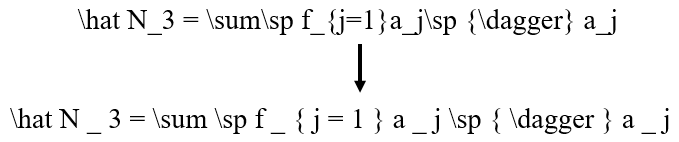


图 14. 插入空格后的LaTeX表达式

原始数据集IM2LATEX-100K中的图片是尺寸为A4大小，除了数学表达式部分皆为透明的图像文件，这样的图像文件容量太大，无效信息过多，为了加快训练模型的效率，必须要对该数据集进行图像预处理，去除背景部分，提取出只含有数学表达式部分的PNG图像，然后将尺寸大小相近的图像进行分批训练，这样可以大幅提高训练模型的效率。

这部分实现主要使用了Python的图像处理PIL模块和科学计算numpy模块，大致流程就是使用pil.open函数载入图像文件，使用np.asarray和np.where提取非空白区域的数值，如果这部分数值为空，意味着这是空白页面或图像过小，舍弃该文件。对于非空白区域，提取它的边界值，然后用pil.crop抓取这个边界的图像输出、保存即可。

1. old\_im = Image.open(img).convert('L') # 以灰度模式载入图像'
2. img\_data = np.asarray(old\_im, dtype=np.uint8) # 读取图像的尺寸数据
3. nnz\_inds = np.where(img\_data!=255)
4. y\_min, y\_max = np.min(nnz\_inds[0]), np.max(nnz\_inds[0]) # 图像的上下边界
5. x\_min, x\_max = np.min(nnz\_inds[1]), np.max(nnz\_inds[1]) # 图像的左右边界
6. old\_im = old\_im.crop((x\_min, y\_min, x\_max+1, y\_max+1)) # 抓取图像

提取完成之后，需要做的就是把相似大小的图像文件打包处理，方便之后按批次训练。根据提取出来的图像文件尺寸大小，我主要分了以下大小的批次供使用（前为长、后为宽）：[240,100],[320,80],[400,80],[400,100],[480,80],[480,100],[560,80],[560,100],[640,80],[640,100],[720,80],[720,100],[720,120],[720,200],[800,100],[800,320],[1000,200]。这部分实现就是循环比较图像文件与批次的大小，由于批次是从小到大顺序排列，因此循环中判断如果图像尺寸小于批次，就将该图像投入该批次中。如果没有合适的批次，就保持原图大小不变。

1. old\_im = Image.open(img)
2. j = -1
3. **for** i **in** range(len(buckets)): # 进入循环
4. **if** old\_size[0]<=buckets[i][0] **and** old\_size[1]<=buckets[i][1]: # 图像小于该批次大小
5. j = i
6. **break** # 退出循环
7. new\_size = buckets[j]
8. new\_im = Image.new("RGB", new\_size, (255,255,255))

这样就完成了图像文件预处理，如图15所示。

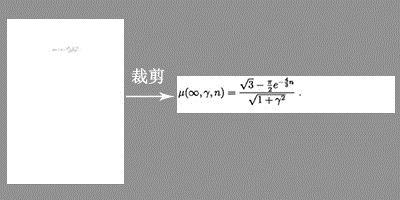


图 15. 完成预处理后的数学表达式图像

## 3.5 模型概述

所使用的模型结合了视觉和自然语言处理方面的标准神经单元。模型首先使用卷积神经网络CNN提取出图像的特征，然后将特征放置于网格中。网格的每一行都使用循环神经网络RNN进行编码。编码得到的特征通过RNN解码器应用视觉注意力模型机制，确切的说，RNN解码器应用了一个基于语义的条件语言模型，该模型被训练成尽可能地实现观察到的标记[19]。图16展示了模型的整体架构。从图中可以看到，给定一个输入图像，首先应用卷积神经网络CNN提取视觉特征，然后对特征网格的每一行进行循环神经网络RNN编码，最后对编码特征使用结合了视觉注意力机制的循环神经网络RNN进行解码得到最后的输出。

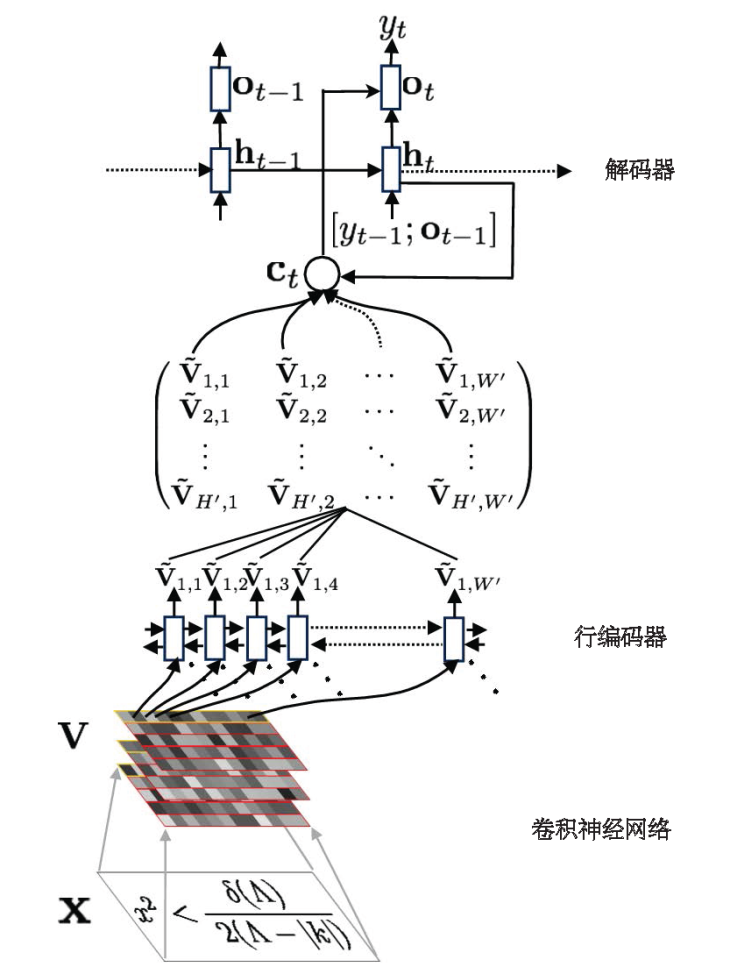


图 16. 模型整体结构

### 3.5.1 卷积神经网络

图像的视觉特征通过交叉结构的多层卷积神经网络CNN和最大池化层得到。该网络结构目前有一个通用标准模型，该项目所采用的是基于Shi等人提出的图像识别模型。第一层输出512个特征，是因为之前预处理的LaTeX关键词中包含了499（第一层输出必须大于这个数字，不然不能覆盖所有元素）个LaTeX代码元素。具体参数如下：

表1 卷积神经网络CNN详细参数

|  |  |
| --- | --- |
| 卷积层 | 最大池化层 |
| c:512, k:(3,3), s:(1,1), p:(0,0), bn | - |
| c:512, k:(3,3), s:(1,1), p:(1,1), bn | Po:(1,2), s:(1,2), p:(0,0) |
| c:256, k:(3,3), s:(1,1), p:(1,1) | Po:(2,1), s:(2,1), p:(0,0) |
| c:256, k:(3,3), s:(1,1), p:(1,1), bn |  |
| c:128, k:(3,3), s:(1,1), p:(1,1) | Po:(2,2), s:(2,2), p:(0,0) |
| c:64, k:(3,3), s:(1,1), p:(1,1) | Po:(2,2), s:(2,2), p:(2,2) |

其中，左边卷积层中，“c”代表滤波器数目，“k”代表核大小（kernel），“s”代表步长大小（stride），“p”代表窗口大小（padding），“bn”代表带有批标准化（batch normalization）；右边最大池化层中，“po”表示池化核大小。

### 3.5.2 行编码器

在Xu等人提出的基于注意力的图像标注模型中，解码器能够直接对图像特征网格进行操作[20]。对于OCR，解码器解码视觉特征需要额外的但是极其重要的序列顺序信息。因此，本项目模型使用额外的循环神经网络RNN对图像特征网格的每一行进行再编码。这样做的基础在于：1、LaTeX语言是基于从左到右的阅读顺序，容易为编码器所学习；2、循环神经网络RNN能够利用水平方向、周围的上下文进行语义强化，确保准确性。

确切的说，一个循环神经网络RNN是一个参数化的RNN函数，他能够循环的将输入向量和隐藏状态投影到一个新的隐藏状态中。在t时刻，隐藏状态通过以下的方式被输入向量更新：

其中是初始状态。在实际使用过程中，循环神经网络有许多不同的结构，然而，长短期记忆网络（Long Short-term Memory Networks，LSTMs）已经被证明能够胜任NLP任务，因此该项目将应用LSTM网络进行实验与开发。

在LSTM模型中，新的特征网格通过对的每一行运行循环神经网络RNN得到。从循环角度看，对于在特征网格中的所有行和列，新特征网格的定义为：

### 3.5.3 解码器

对于想要得到的LaTeX符号，可以通过解码一系列的基于注释的特征网格得到。此时所使用的解码器被训练为一个条件语言模型，以支持在给予历史信息和注释的情况下，能够判断下一个符号的概率。

这个使用到的语言模型在循环神经网络RNN解码器的顶层定义：

公式中，为一个已学习的线性变换，等于。

向量用来概括解码历史：

上下文向量用来在注释特征网格中获取上下文信息，在任意时刻*t*，上下文向量与注释网格有关。然而绝大部分的注释单元是不相关的，因此模型需要明白哪些单元是有用的，因此，需要使用一个注意力模型,定义为：

公式中是基于计算所得的权重值，权重向量和所有注释向量结合生成一个上下文向量。向量和相连接，以预测符号的可能性。图17给出了一个在模型的每一个步骤中的注意力分布的示例。示例中，模型基于输入图像在不同时刻生成一个相应的LaTeX符号。

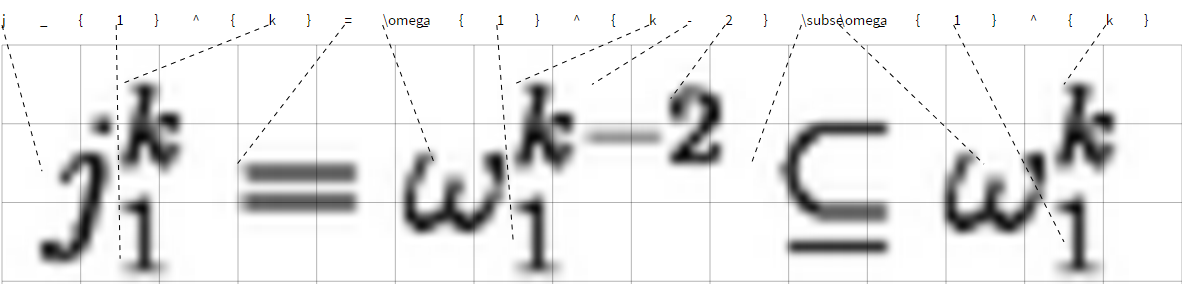


图 17. 模型生成LaTeX示例

## 3.6 用户界面设计

本项目有3个界面：欢迎界面、功能界面，以及关于界面。三个界面可以通过主界面顶部功能栏进行切换，也可以通过各个界面提供的接口按钮进行切换。主要设计如下：

欢迎使用

开始使用

致谢声明

介绍

数学表达式翻译系统

开始使用

图 18. 欢迎界面设计

界面设计的较为简洁，顶部是设计的功能栏。主界面中有程序LOGO和简短的介绍，点击“开始使用”按钮，程序会切换到功能界面窗口。

欢迎使用

开始使用

致谢声明

原始图像显示区域

数学表达式翻译系统

生成LaTeX源码区域

预测图像显示区域

载入图像

启动预测

显示结果

重置

图 19. 功能界面设计

功能界面是程序的主要界面，从图19可以看到，有“载入图像”、“启动预测”和“显示结果”三个按钮，点击“载入图像”，用户可以载入想要识别的数学表达式图像，该图像会显示与“原始图像显示区域”。点击“启动预测”，程序会后台调取模型的预测功能对载入图像进行处理，预测的LaTeX会显示在“生成LaTeX源码区域”。最后点击“显示结果”，程序会将之前预测的LaTeX代码转换成对应的表达式图像，显示在“预测图像显示区域”。点击“重置”按钮，程序会清楚重置之前的所有操作。

欢迎使用

开始使用

致谢声明

说明

数学表达式翻译系统

返回

图 20. 关于界面设计

关于界面主要内容是一些总结和感谢性话语，点击“返回”程序会回到欢迎界面。

## 3.7 本章小结

本章首先介绍了本项目的开发环境，科学计算环境Anaconda，图形计算单元CUDA，以及使用的机器学习框架TensorFlow。然后介绍了本项目使用的数据集IM2LATEX-100K，以及对该数据集进行的预处理操作。最后阐述了本项目使用到的卷积神经网络+循环神经网络模型的整体结构和原理，卷积神经网络的具体参数，循环神经网络编码器和解码器的实现、以及用户界面的功能和整体设计。

# 第四章 方案实现

## 4.1 模型的TensorFlow实现

### 4.1.1 卷积神经网络

根据3.5节对模型的叙述，首先是在TensorFlow中定义卷积神经网络CNN，正如前文所提到的，卷积神经网络的要点就是局部连接（Local Connection）、权值共享（Weight sharing）和池化层（Pooling）中的降采样（Down-Sampling）。

首先定义权重和偏差的初始化构造函数，以便之后在卷积神经网络的定义中直接调用：

1. **def** weight\_variable(name, shape):  # 权重函数
2. initial = tf.truncated\_normal(shape, stddev = 0.1)
3. **return** tf.get\_variable(name + "\_weights", initializer= initial)
5. **def** bias\_variable(name, shape):  # 偏差函数
6. initial = tf.constant(0.1, shape = shape)
7. **return** tf.get\_variable(name + "\_bias", initializer= initial)

tf.truncated\_normal函数的功能是从一个正态分布片段中输出随机数值。其中，生成的值会遵循一个指定了平均值和标准差的正态分布，只保留两个标准差以内的值，超出的值会被弃掉重新生成。而tf.constant函数的功能是创建一个常量张量。

下一步是定义节点准备接收数据，这里使用了占位符tf.placeholder，X是特征，mask是输入图像，seqs是输入序列。

1. X = tf.placeholder(shape=(None,None,None,None),dtype=tf.float32)
2. mask = tf.placeholder(shape=(None,None),dtype=tf.int32)
3. seqs = tf.placeholder(shape=(None,None),dtype=tf.int32)
4. learn\_rate = tf.placeholder(tf.float32)
5. input\_seqs = seqs[:,:-1]
6. target\_seqs = seqs[:,1:]
7. emb\_seqs = tflib.ops.Embedding('Embedding',V,EMB\_DIM,input\_seqs)

其中，tf.placeholder函数的功能是插入一个待初始化的张量占位符。由于输入图像格式是一个四维的张量，所以要求类型是float32。

emb\_seqs表示一个嵌入集，也就是将词语密集地嵌入到一个集合中，其作用是在RNN网络中允许模型高效地表示词语。

接下来是定义卷积层和池化层，具体表现为一个滑动步长为1的二维卷积核,和一个大小为2x2，滑动步长为2x2的池化层：

1. **def** conv2d(x, W):  # 卷积层
2. **return** tf.nn.conv2d(x, W, strides = [1,1,1,1], padding = 'SAME')
4. **def** max\_pool\_2\_2(x, ksize = [1,2,2,1], strides = [1,2,2,1]):  # 池化层
5. **return** tf.nn.max\_pool(x, ksize, strides, padding = 'SAME')

其中tf.nn.conv2d函数能够通过输入规定的四维输入和滤波器计算二维卷积。给定一个输入的张量（Tensor）x，x也就是需要做卷积的输入图像，具有[batch, in\_height, in\_width, in\_channels]这样的形式，和一个卷积核张量W：[weight\_height, weight\_width, in\_channels, out\_channels]后，执行以下操作：

（1）将卷积核转换为一个形状为[weight\_height \* weight\_width \* in\_channels, output\_channels]的二维矩阵。

（2）从输入x中提取图像矩阵形成一个大小为[batch, out\_height, out\_width, weight\_height \* weight\_width \* in\_channels]的虚拟张量。

（3）对每个图像矩阵循环进行weight矩阵右乘操作。

而tf.nn.max\_pool函数的功能是对输入进行池化，其中“ksize”表示池化窗口的大小，和卷积相同，“strides”也表示滑动步长大小，“padding='SAME'”表示依靠边界补零，使得输出和输入的大小保持一致，。

然后是实现卷积神经网络CNN的整体结构，其中的层数和参数设置与表1中所给出的参数设置相同，假设输入表达式图片的维数为20\*200\*500\*1的张量，那么cnn函数输出的是20\*28\*65\*64维数的张量：

1. **def** cnn(inp):
2. # 第一层，特征图为512个
3. W\_conv1 = weight\_variable("conv1", [3,3,1,512])
4. b\_conv1 = bias\_variable("conv1", [512])
5. h\_conv1 = tf.nn.relu(conv2d(inp,W\_conv1) + b\_conv1)
6. h\_bn1   = tf.contrib.layers.batch\_norm(h\_conv1)
7. # 第二层，特征图512个
8. W\_conv2 = weight\_variable("conv2", [3,3,512,512])
9. b\_conv2 = bias\_variable("conv2", [512])
10. h\_pad2  = tf.pad(h\_bn1, [[0,0],[1,1],[1,1],[0,0]], "CONSTANT")
11. h\_conv2 = tf.nn.relu(conv2d(h\_pad2, W\_conv2) + b\_conv2)
12. h\_bn2   = tf.contrib.layers.batch\_norm(h\_conv2)
13. h\_pool2 = max\_pool\_2\_2(h\_bn2, [1,1,2,1], [1,1,2,1])
14. # 第三层，特征图256个
15. W\_conv3 = weight\_variable("conv3", [3,3,512,256])
16. b\_conv3 = bias\_variable("conv3", [256])
17. h\_pad3  = tf.pad(h\_pool2, [[0,0],[1,1],[1,1],[0,0]], "CONSTANT")
18. h\_conv3 = tf.nn.relu(conv2d(h\_pad3, W\_conv3) + b\_conv3)
19. h\_pool3 = max\_pool\_2\_2(h\_conv3, [1,2,1,1], [1,2,1,1])
20. # 第四层，特征图256个
21. W\_conv4 = weight\_variable("conv4", [3,3,256,256])
22. b\_conv4 = bias\_variable("conv4", [256])
23. h\_pad4  = tf.pad(h\_pool3, [[0,0],[1,1],[1,1],[0,0]], "CONSTANT")
24. h\_conv4 = tf.nn.relu(conv2d(h\_pad4, W\_conv4) + b\_conv4)
25. h\_bn4   = tf.contrib.layers.batch\_norm(h\_conv4)
26. # 第五层，特征图128个
27. W\_conv5 = weight\_variable("conv5", [3,3,256,128])
28. b\_conv5 = bias\_variable("conv5", [128])
29. h\_pad5  = tf.pad(h\_bn4, [[0,0],[1,1],[1,1],[0,0]], "CONSTANT")
30. h\_conv5 = tf.nn.relu(conv2d(h\_pad5, W\_conv5) + b\_conv5)
31. h\_pool5 = max\_pool\_2\_2(h\_conv5, [1,2,2,1], [1,2,2,1])
32. # 第六层，特征图64个
33. W\_conv6 = weight\_variable("conv6", [3,3,128,64])
34. b\_conv6 = bias\_variable("conv6", [64])
35. h\_pad6  = tf.pad(h\_pool5, [[0,0],[1,1],[1,1],[0,0]], "CONSTANT")
36. h\_conv6 = tf.nn.relu(conv2d(h\_pad6, W\_conv6) + b\_conv6)
37. h\_pad6  = tf.pad(h\_conv6, [[0,0],[2,2],[2,2],[0,0]], "CONSTANT")
38. h\_pool6 = max\_pool(h\_pad6, [1,2,2,1], [1,2,2,1])
39. **return** h\_pool6

这里使用了tf.nn.relu作为激励函数，激励函数的作用是增加神经网络模型的非线性。如果没有激励函数，神经网络的每层都相当于矩阵相乘，因此无论神经网络有多少层，输出都是输入的线性组合，与没有隐藏层效果相当。

除了普通的CNN卷积层和池化层外还有tf.contrib.layers.batch\_norm，使深层神经网络训练更加稳定，加快了收敛的速度，同时起到了正则化的作用。

### 4.1.2 循环神经网络

3.5节中提到过，在完成卷积神经网络CNN提取图像特征完毕之后，需要使用一个基于注意力模型的循环神经网络RNN对特征网格的每一行进行编码。如上述所述CNN输出20\*28\*65\*64，由于使用双向RNN，LSTM的隐藏层特征数为enc\_lstm\_dim = 256，且经维数输入和输出变化后，编码器输出为20\*28\*65\*512。以下为双向RNN编码器代码：

1. **def** fn(inp):
2. # 对特征网格的每一行进行RNN处理
3. batch\_size, enc\_lstm\_dim = 20, 256
4. enc\_init\_shape = [batch\_size, enc\_lstm\_dim]
5. with tf.variable\_scope('encoder\_rnn'):
6. with tf.variable\_scope('forward'):
7. lstm\_cell\_fw = tf.nn.rnn\_cell.LSTMCell(enc\_lstm\_dim)
8. init\_fw = tf.nn.rnn\_cell.LSTMStateTuple(\
9. tf.get\_variable("enc\_fw\_c", enc\_init\_shape),\
10. tf.get\_variable("enc\_fw\_h", enc\_init\_shape)
11. )
12. with tf.variable\_scope('backward'):
13. lstm\_cell\_bw = tf.nn.rnn\_cell.LSTMCell(enc\_lstm\_dim)
14. init\_bw = tf.nn.rnn\_cell.LSTMStateTuple(\
15. tf.get\_variable("enc\_bw\_c", enc\_init\_shape),\
16. tf.get\_variable("enc\_bw\_h", enc\_init\_shape)
17. )
18. output, \_ = tf.nn.bidirectional\_dynamic\_rnn(lstm\_cell\_fw, \
19. lstm\_cell\_bw, \
20. inp, \
21. sequence\_length = \
22. tf.fill([batch\_size],\
23. tf.shape(inp)[1]), \
24. initial\_state\_fw = init\_fw,\
25. initial\_state\_bw = init\_bw \
26. )
27. **return** tf.concat(2,output)

tf.nn.rnn\_cell.LSTMCell函数功能是实现的是上文3.5.2节所提出的LSTM模型，这里分为前向（forward）和后向（backward）双向的LSTM网络，这样做的目的在于双向的LSTM可以同时使用过去和未来两个时间方向上的信息，从而大大提高了最终结果的准确性。而tf.nn.bidirectional\_dynamic\_rnn 的功能是创建双向循环神经网络RNN的动态版本，也就是将之前提到的前向和后向两个LSTMCell以及输入X进行整合。

编码器输出到带注意力机制的LSTM解码器，LSTM的隐藏层特征为512，最后得到图片对应的公式LaTeX代码的概率分布：

1. fun = tf.make\_template('fun', fn)
2. rows\_first = tf.transpose(cnn,[1,0,2,3])
3. res = tf.map\_fn(fun, rows\_first, dtype=tf.float32)
4. encoder\_output = tf.transpose(res,[1,0,2,3])
6. dec\_lstm\_cell = tf.nn.rnn\_cell.LSTMCell(dec\_lstm\_dim)
7. dec\_init\_shape = [batch\_size, dec\_lstm\_dim]
8. dec\_init\_state = tf.nn.rnn\_cell.LSTMStateTuple( tf.truncated\_normal(dec\_init\_shape),\
9. tf.truncated\_normal(dec\_init\_shape) )
11. init\_words = np.zeros([batch\_size,1,vocab\_size])
13. decoder\_output = decoder.attention\_decoder(dec\_init\_state,\
14. tf.reshape(encoder\_output,\
15. [batch\_size, -1,\
16. 2\*enc\_lstm\_dim]),\
17. dec\_lstm\_cell,\
18. vocab\_size,\
19. dec\_seq\_len,\
20. batch\_size,\
21. embedding\_size,\
22. feed\_previous=True)
23. **return** (encoder\_output, decoder\_output)

再一步就是整个TensorFlow运行环境的搭建，在这里定义损失函数loss和训练步骤（train\_step），并使用GradientDescentOptimizer梯度下降优化器以最小化损失函数。

1. loss = tf.reshape(tf.nn.sparse\_softmax\_cross\_entropy\_with\_logits(
2. tf.reshape(logits,[-1,V]),
3. tf.reshape(seqs[:,1:],[-1])
4. ), [tf.shape(X)[0], -1])
6. mask\_mult = tf.to\_float(mask[:,1:])
7. loss = tf.reduce\_sum(loss\*mask\_mult)/tf.reduce\_sum(mask\_mult)
8. optimizer = tf.train.GradientDescentOptimizer(learn\_rate)
9. gvs = optimizer.compute\_gradients(loss)
10. capped\_gvs = [(tf.clip\_by\_norm(grad, 5.), var) **for** grad, var **in** gvs]
11. train\_step = optimizer.apply\_gradients(capped\_gvs)

最后就是创建一个TensorFlow会话（使用了8个线程），对全局的变量进行初始化，然后迭代进行50次训练。最后就是在训练时计算预测的准确率。

1. sess = tf.Session(config=tf.ConfigProto(intra\_op\_parallelism\_threads=8))
2. init = tf.global\_variables\_initializer()
3. **for** i **in** range(0,50):
4. iter=0
5. costs=[]
6. itr = data\_loaders.data\_iterator('train', BATCH\_SIZE)
7. **for** train\_img,train\_seq,train\_mask **in** itr:
8. iter += 1
9. \_ , \_loss = sess.run([train\_step,loss],feed\_dict={X:train\_img,seqs:train\_seq,mask:train\_mask,learn\_rate:lr})
10. costs.append(\_loss)

然后就是在验证集valid set中测试模型的准确率：

1. score\_itr = data\_loaders.data\_iterator(set=’valid’,batch\_size)
2. losses = []
3. **for** score\_imgs,score\_seqs,score\_mask **in** score\_itr:
4. \_loss = sess.run(loss,feed\_dict={X:score\_imgs,seqs:score\_seqs,mask:score\_mask})
5. losses.append(\_loss)
6. set\_loss = np.mean(losses)
7. perp = np.mean(map(lambda x: np.power(np.e,x),losses))

## 4.2 用户界面实现

用户界面的实现如图21、22和23所示，其中，用户界面的风格和样式是使用QSS语言进行编写设计的。



图 21. 开始界面



图 22. 功能界面

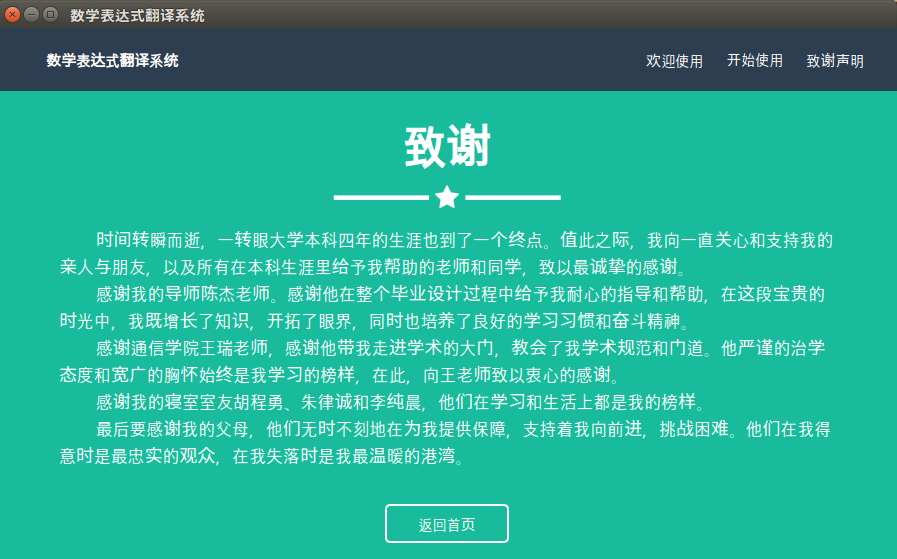


图23. 致谢界面

### 4.2.1 界面框架实现

数学表达式翻译系统的GUI设计是基于PyQt完成的，PyQt的编程风格和Qt类似，都是面向对象的编程。如图24所示，该系统的三个界面共享一个顶部功能栏（Header），点击功能栏的相应按钮（Button）就可以切换到对应的界面。要实现这样的功能，定义了如下的Qt类：

QTabWidget Header

QFrame StartWindow

QFrame MainWindow

QFrame AboutWindow

QWidget Window

图 24. 界面窗口结构

具体实现的方法就是，先创建一个主框架类QWidget Window，用来承载和排列所有的功能部件。顶部功能栏Header是一个Qt控件QTabWidget，其作用就是提供Tab按钮，允许用户点击相应的Tab按钮来切换不同的窗口。因此，如图可见，这里的Header有三个Tab：QFrame StartWindow（开始界面）、QFrame MainWindow（功能界面）和QFrame AboutWindow（致谢界面）。

为了提高用户产品体验，我为窗口之间的切换增加了滑动切换的动画效果。具体实现中，从左到右切换窗口（开始界面到功能界面）需要逝去窗口向左划出，类似的，从右到左切换窗口需要逝去窗口向右划出。要实现这个效果，就需要将三个Tab分别编号0、1和2，并设定一个CurrentIndex变量使得Header记住当前窗口，并通过Tab的触发信号判断下一窗口是当前窗口的左边还是右边。

动画效果的实现使用了Qt的QPropertyAnimation函数。其基本思想就是，给定一个物件初始位置和最终位置，设置一个动画效果持续时间，就可以完成在该持续时间内完成物件的从初始位置到最终位置的平移。

下面代码是一个从窗口右到左滑动的函数：

1. self.HideAnimation = QtCore.QPropertyAnimation(self, "geometry")
2. self.HideAnimation.setDuration(300) # 动画持续时间300ms
3. self.startGeometry = QtCore.QRect(self.geometry())   # 窗口初始位置
4. self.endGeometry = QtCore.QRect(0,  # 窗口最终位置
5. self.geometry().y(),
6. 0,  self.height())
7. self.HideAnimation.setStartValue(self.startGeometry)
8. self.HideAnimation.setEndValue(self.endGeometry)
9. self.HideAnimation.start()
10. self.setWindowOpacity(0.3)  # 窗口淡出效果

### 4.2.1 界面功能实现

数学表达式翻译系统的主要功能有三：1、打开载入需要识别的数学表达式图片（png格式）；2、调用模型的预测功能识别图片转换的LaTeX源码；3、通过预测的LaTeX源码再生成对应的数学表达式图像，方便用户比较预测正确与否。

三个功能都在功能界面（MainWindow）中实现，首先是定义打开图片这个函数：

1. **global** imgName
2. imgName,imgType= QFileDialog.getOpenFileName(self, "Load image file", "", " \*.png;;All Files (\*)")
3. png = QPixmap(imgName).scaled(self.Input\_img.width(), self.Input\_img.height(), Qt.KeepAspectRatio)
4. self.Input\_img.setPixmap(png)

其中，imgName设定的是全局变量，方便之后将载入图像的数据供模型预测时的调用。打开图片使用的控件是Qt内置函数QFileDialog文件对话框，能够方便用户选择图像文件。显示图像使用的是Qt内置函数QPixmap，其功能就是在控件区域内显示图像文件。显示LaTeX用的是专用来显示文本的QTextBrowser。

最后一个关键部分就是如何将预测出的LaTeX语言再转为图像格式，以供核查比较预测的正确与否，这一部分我使用了matplotlib库中的LaTeX数学表达式转换功能：

1. fig = plt.figure(figsize=(0.01, 0.01))
2. fig.text(0, 0, u'${}$'.format(formula), fontsize=fontsize)
3. buffer\_ = StringIO()
4. fig.savefig(buffer\_, dpi=dpi, transparent=False, format=format\_, bbox\_inches='tight', pad\_inches=0.0)
5. plt.close(fig)
6. buffer\_.getvalue()

# 第五章 模型准确率比较与软件测试

模型的训练总共经过7个epoch，耗费84小时。

评估本项目成果的最关键一点就是模型的准确率，我将本项目模型与一些当前比较流行的能够用于图像-LaTeX转换的OCR系统进行比较。

当前最流行的基于OCR的数学表达式翻译系统是前文介绍过的InftyReader系统，该系统结合了符号识别和结构分析的特性。另外就是开源OCR系统AbiWord，该系统也包含了一个Tex生成模式。

实验环境：操作系统使用的是Linux系统Ubuntu 16.04LTS，CPU型号Intel Core i5 7200U，主频2.5GHz，4GB内存，GPU型号NVIDIA GeForce 940MX，显存为4GB。

数据集：IM2LATEX-100K数据集的测试集，包含10,354张数学表达式图像。

评价指标：该项目准确率的核心评估方法是检查的由预测LaTeX生成的数学表达式图像与原数学表达式图像之间的差异。这两者之间的差异在于原图与预测图像之间每一列的距离的差异。评价过程明确地离散化每一生成的列，并比较序列的编辑距离。

表2 基于IM2LATEX-100K数据集的准确率实验结果

|  |  |  |
| --- | --- | --- |
| **应用名称** | **正确数（个）** | **准确率（%）** |
| AbiWord | 5,227 | 50.48 |
| InftyReader | 6,743 | 65.12 |
| 本项目 | 7,248 | 70.09 |

数学表达式识别准确率的实验比较了几个流行的图像转换LaTeX的OCR系统。AbiWord的表现最差，其结果符合评价指标的样本数为5,227，准确率只是刚过半；经典的INFTY系统在准确性方面比较出色，符合条件样本数为6,743，准确率为65.12%。而本项目使用的模型的准确率最佳，高达70.09%，符合条件样本数为7,248。

以下是在软件中的运行实例：

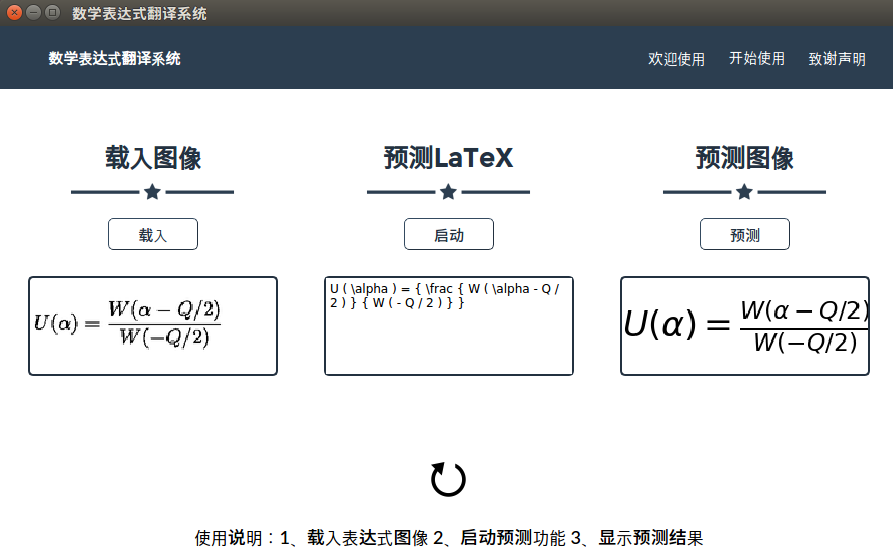


图 25. 软件运行实例1

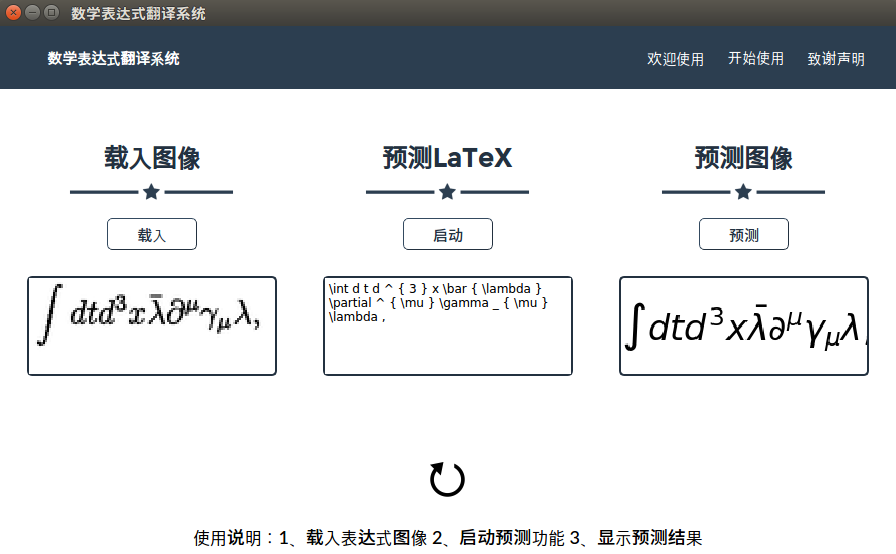


图 26. 软件运行实例2

# 第六章 总结与展望

当前，机器学习是一个十分火热的研究领域，而Python语言是进行机器学习实践最好的语言。在完成本课题的过程中，结合对Python的使用，我学会了对不同的机器学习任务选用合适的机器学习算法，如何对数据进行预处理以使得能够更好地训练和测试模型，还学会了编写用户界面方便展示和供他人的使用。

最重要的是，通过完成整个课题，我学会了一套处理和解决问题的方法，掌握了如何去学习新事物、将想法用Python代码去实现。完成毕业论文的经历也锻炼了自己的资料收集以及文字梳理和表达能力。所有的这些磨练都将使我受益一生。

本项目“数学表达式翻译系统” 基于Google最新机器学习框架TensorFlow，应用卷积神经网络CNN和循环神经网络RNN的方法，使用Python编程语言编写开发一个图像翻译系统。相比于传统OCR的基于文本结构分析的特点，使用机器学习的好处在于即便没有任何关于LaTeX方面的知识，基于大数据驱动的模型表现更加出色。相较于市面上的传统OCR系统，该系统有效地提升了在翻译数学表达式图像上的准确性。并且使用PyQt进行用户界面的设计，能给予用户良好的使用体验。但是由于整个开发和测试时间比较有限，仍有一些不足和需要改进的地方。

在模型的训练方面，由于数据集比较庞大，受限于计算机硬件配置的不足，尽管使用了GPU计算，但还是只对数据集训练了7个Epoch，导致准确率还未到最佳表现。未来希望能够通过分布式计算或者云服务弥补这块缺陷。

另外，本项目适用于翻译印刷体数学表达式，未来希望在条件允许的情况下，开发能够翻译手写数学表达式的系统。

# 致谢

时间转瞬而逝，一转眼大学本科四年的生涯也到了一个终点。值此论文完成之际，我向一直关心和支持我的亲人与朋友，以及所有在本科生涯里给予我帮助的老师和同学，致以最诚挚的感谢。

首先，我要感谢我的导师陈杰老师。感谢他在整个毕业设计过程中给予我耐心的指导和帮助，在这段宝贵的时光中，我既增长了知识，开拓了眼界，同时也培养了良好的学习习惯和奋斗精神。

感谢通信学院王瑞老师，感谢他带我走进学术的大门，教会了我学术规范和门道。王老师是我学习的榜样，在此，向王老师致以衷心的感谢。

感谢我的父母，他们无时不刻地在为我提供保障，支持着我向前进，挑战困难。他们在我得意时是最忠实的观众，在我失落时是我最温暖的港湾。

最后要感谢上海大学、上海大学通信与信息工程学院的悉心栽培，十年树木，百年树人。身为一名上大人，会时刻牢记钱老的尊尊教诲。

# 参考文献

[1] 周志华著.机器学习.北京： 清华大学出版社; 2016.1.

[2] Anderson, R. H. 1967. Syntax-directed recognition of handprinted two-dimensional mathematics. In Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium, 436–459. ACM.

[3] Belaid, A., and Haton, J.-P. 1984. A syntactic approach for handwritten mathematical formula recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (1):105–111.

[4] Chan, K., and Yeung, D. 2000. Mathematical expression recognition: a survey. IJDAR 3(1):3–15.

[5] Miller, E. G., and Viola, P. A. 1998. Ambiguity and constraint in mathematical expression recognition. In AAAI/IAAI, 784–791.

[6] Suzuki M, Tamari F, Fukuda R, et al. INFTY: an integrated OCR system for mathematical documents[C]// ACM Symposium on Document Engineering. ACM, 2003:95-104.

[7] Ciresan, D. C.; Meier, U.; Gambardella, L. M.; and Schmidhuber, J. 2010. Deep, big, simple neural nets for handwritten digit recognition. Neural computation 22(12):3207–3220.

[8] Jaderberg, M.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2015. Deep structured output learning for unconstrained text recognition. ICLR.

[9] Jaderberg, M.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2016. Reading text in the wild with convolutional neural networks. International Journal of Computer Vision 116(1):1–20.

[10] Karpathy, A., and Fei-Fei, L. 2015. Deep visual-semantic alignments for generating image descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3128–3137.

[11] Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015b. Show and tell: A neural image caption generator. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3156–3164.

[12] [美] 凯文•凯利著，张行舟,陈新武,王钦 等译.失控.北京：电子工业出版社,2016.3.

[13] 倪桂博 印刷文字识别的研究 [D].河北:华北电力大学, 2008 :1.

[14] LeCun, Y., Bottou, L., and Haffner, P．Gradient-based learning applied to document recognition[J]．Proc of the IEEE，1 998，86(1 1)：2278-2324.

[15] Greene, Will Jam H., Econometric Analysis, fifth edition, Preutice Hall, 1993: 720—723．

[16] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, J. Schmidhuber. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, 2009

[17] H. Sak and A. W. Senior and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Proc. Interspeech, pp338-342, Singapore, Sept. 201

[18] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[19] Deng Y, Kanervisto A, Rush A M. What You Get Is What You See: A Visual Markup Decompiler[J]. 2016.

[20] Xu K, Ba J, Kiros R, et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention[J]. Computer Science, 2015:2048-2057.

# 附录一

## 英译中原文

**Show, Attend and Tell: Neural Image Caption**

**Generation with Visual Attention**

Kelvin Xu KELVIN.XU@UMONTREAL.CA

Jimmy Lei Ba JIMMY@PSI.UTORONTO.CA

Ryan Kiros RKIROS@CS.TORONTO.EDU

Kyunghyun Cho KYUNGHYUN.CHO@UMONTREAL.CA

Aaron Courville AARON.COURVILLE@UMONTREAL.CA

Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU

Richard S. Zemel ZEMEL@CS.TORONTO.EDU

Yoshua Bengio FIND-ME@THE.WEB

**Abstract**

Inspired by recent work in machine translation and object detection, we introduce an attention based model that automatically learns to describe the content of images. We describe how we can train this model in a deterministic manner using standard backpropagation techniques and stochastically by maximizing a variational lower bound. We also show through visualization how the model is able to automatically learn to fix its gaze on salient objects while generating the corresponding words in the output sequence. We validate the use of attention with state-of-theart performance on three benchmark datasets: Flickr8k, Flickr30k and MS COCO.

1. **Introduction**

Automatically generating captions of an image is a task very close to the heart of scene understanding—one of the primary goals of computer vision. Not only must caption generation models be powerful enough to solve the computer vision challenges of determining which objects are in an image, but they must also be capable of capturing and expressing their relationships in a natural language. For this reason, caption generation has long been viewed as a difficult problem. It is a very important challenge for machine learning algorithms, as it amounts to mimicking the remarkable human ability to compress huge amounts of salient visual infomation into descriptive language.

Despite the challenging nature of this task, there has been a recent surge of research interest in attacking the image caption generation problem. Aided by advances in training neural networks (Krizhevsky et al., 2012) and large classification datasets (Russakovsky et al., 2014), recent work has significantly improved the quality of caption generation using a combination of convolutional neural networks (convnets) to obtain vectorial representation of images and recurrent neural networks to decode those representations into natural language sentences (see Sec. 2).

One of the most curious facets of the human visual system is the presence of attention (Rensink, 2000; Corbetta & Shulman, 2002). Rather than compress an entire image into a static representation, attention allows for salient features to dynamically come to the forefront as needed. This is especially important when there is a lot of clutter in an image. Using representations (such as those from the top layer of a convnet) that distill information in image down to the most salient objects is one effective solution that has been widely adopted in previous work. Unfortunately, this has one potential drawback of losing information which could be useful for richer, more descriptive captions. Using more low-level representation can help preserve this information. However working with these features necessitates a powerful mechanism to steer the model to information important to the task at hand.

In this paper, we describe approaches to caption generation that attempt to incorporate a form of attention with two variants: a “hard” attention mechanism and a “soft” attention mechanism. We also show how one advantage of including attention is the ability to visualize what the model “sees”. Encouraged by recent advances in caption generation and inspired by recent success in employing attention in machine translation (Bahdanau et al., 2014) and object recognition (Ba et al., 2014; Mnih et al., 2014), we investigate models that can attend to salient part of an image while generating its caption.

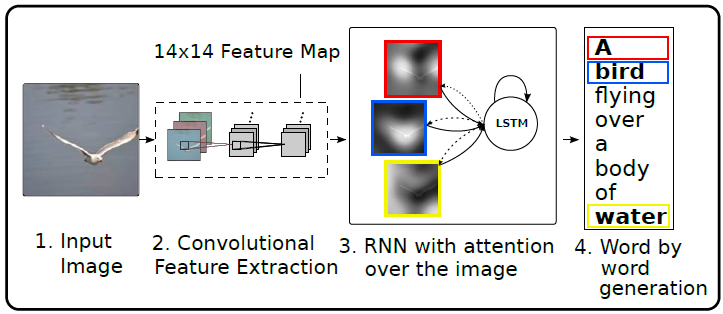


Figure 1. Our model learns a words/image alignment. The visualized

attentional maps (3) are explained in section 3.1 & 5.4

The contributions of this paper are the following:

⚫ We introduce two attention-based image caption generators under a common framework (Sec. 3.1): 1) a “soft” deterministic attention mechanism trainable by standard back-propagation methods and 2) a “hard” stochastic attention mechanism trainable by maximizing an approximate variational lower bound or equivalently by REINFORCE (Williams, 1992).

⚫ We show how we can gain insight and interpret the results of this framework by visualizing “where” and “what” the attention focused on. (see Sec. 5.4)

⚫ Finally, we quantitatively validate the usefulness of attention in caption generation with state of the art performance (Sec. 5.3) on three benchmark datasets: Flickr8k (Hodosh et al., 2013), Flickr30k (Young et al., 2014) and the MS COCO dataset (Lin et al., 2014).

1. **Related Work**

In this section we provide relevant background on previous work on image caption generation and attention. Recently, several methods have been proposed for generating image descriptions. Many of these methods are based on recurrent neural networks and inspired by the successful use of sequence to sequence training with neural networks for machine translation (Cho et al., 2014; Bahdanau et al., 2014; Sutskever et al., 2014). One major reason image caption generation is well suited to the encoder-decoder framework (Cho et al., 2014) of machine translation is because it is analogous to “translating” an image to a sentence.

The first approach to use neural networks for caption generation was Kiros et al. (2014a), who proposed a multimodal log-bilinear model that was biased by features from the image. This work was later followed by Kiros et al. (2014b) whose method was designed to explicitly allow a natural way of doing both ranking and generation. Mao et al. (2014) took a similar approach to generation but replaced a feed-forward neural language model with a recurrent one. Both Vinyals et al. (2014) and Donahue et al. (2014) use LSTM RNNs for their models. Unlike Kiros et al. (2014a) and Mao et al. (2014) whose models see the image at each time step of the output word sequence, Vinyals et al. (2014) only show the image to the RNN at the beginning. Along with images, Donahue et al. (2014) also apply LSTMs to videos, allowing their model to generate video descriptions.

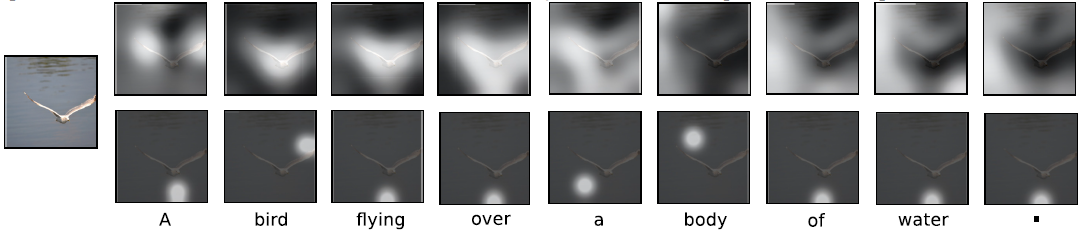


Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. “soft” (top row) vs “hard” (bottom row) attention. (Note that both models generated the same captions in this example.)

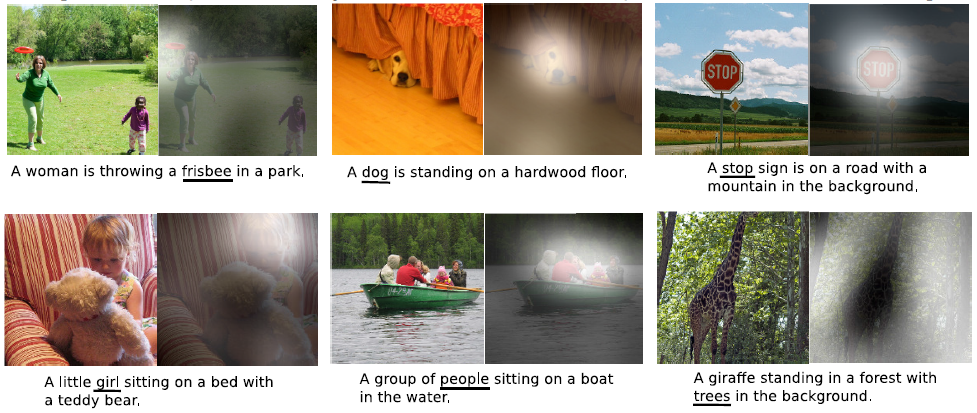


Figure 3. Examples of attending to the correct object (white indicates the attended regions, underlines indicated the corresponding word)

All of these works represent images as a single feature vector from the top layer of a pre-trained convolutional network. Karpathy & Li (2014) instead proposed to learn a joint embedding space for ranking and generation whose model learns to score sentence and image similarity as a function of R-CNN object detections with outputs of a bidirectional RNN. Fang et al. (2014) proposed a three-step pipeline for generation by incorporating object detections. Their model first learn detectors for several visual concepts based on a multi-instance learning framework. A language model trained on captions was then applied to the detector outputs, followed by rescoring from a joint image-text embedding space. Unlike these models, our proposed attention framework does not explicitly use object detectors but instead learns latent alignments from scratch. This allows our model to go beyond “objectness” and learn to attend to abstract concepts.

Prior to the use of neural networks for generating captions, two main approaches were dominant. The first involved generating caption templates which were filled in based on the results of object detections and attribute discovery (Kulkarni et al. (2013), Li et al. (2011), Yang et al. (2011), Mitchell et al. (2012), Elliott & Keller (2013)). The second approach was based on first retrieving similar captioned images from a large database then modifying these retrieved captions to fit the query (Kuznetsova et al., 2012; 2014). These approaches typically involved an intermediate “generalization” step to remove the specifics of a caption that are only relevant to the retrieved image, such as the name of a city. Both of these approaches have since fallen out of favour to the now dominant neural network methods.

There has been a long line of previous work incorporating attention into neural networks for vision related tasks. Some that share the same spirit as our work include Larochelle & Hinton (2010); Denil et al. (2012); Tang et al. (2014). In particular however, our work directly extends the work of Bahdanau et al. (2014); Mnih et al. (2014); Ba et al. (2014).

1. **Image Caption Generation with Attention Mechanism**
   1. **Model Details**

In this section, we describe the two variants of our attention-based model by first describing their common framework. The main difference is the definition of the ‑ function which we describe in detail in Section 4. We denote vectors with bolded font and matrices with capital letters. In our description below, we suppress bias terms for readability.

3.1.1. ENCODER: CONVOLUTIONAL FEATURES

Our model takes a single raw image and generates a caption y encoded as a sequence of 1-of-K encoded words.



where K is the size of the vocabulary and C is the length of the caption.

We use a convolutional neural network in order to extract a set of feature vectors which we refer to as annotation vectors. The extractor produces L vectors, each of which is a D-dimensional representation corresponding to a part of the image.



In order to obtain a correspondence between the feature vectors and portions of the 2-D image, we extract features from a lower convolutional layer unlike previous work which instead used a fully connected layer. This allows the decoder to selectively focus on certain parts of an image by selecting a subset of all the feature vectors.

3.1.2. DECODER: LONG SHORT-TERM MEMORY NETWORK

We use a long short-term memory (LSTM) network (Hochreiter & Schmidhuber, 1997) that produces a caption by generating one word at every time step conditioned on a context vector, the previous hidden state and the previously generated words. Our implementation of LSTM closely follows the one used in Zaremba et al. (2014) (see Fig. 4). Using to denote a simple affine transformation with parameters that are learned,

  (1)

 (2)

 (3)

Here, , , , are the input, forget, memory, output and hidden state of the LSTM, respectively. The vector is the context vector, capturing the visual information associated with a particular input location, as explained below. is an embedding matrix. Let m and n denote the embedding and LSTM dimensionality respectively and and be the logistic sigmoid activation and element-wise multiplication respectively.

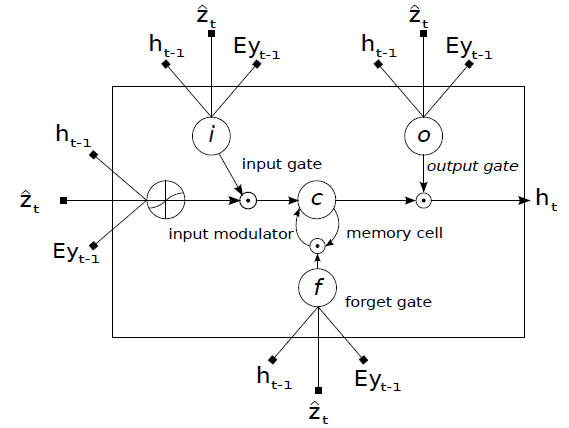


Figure 4. A LSTM cell, lines with bolded squares imply projections with a learnt weight vector. Each cell learns how to weigh its input components (input gate), while learning how to modulate that contribution to the memory (input modulator). It also learns weights which erase the memory cell (forget gate), and weights which control how this memory should be emitted (output gate).

In simple terms, the context vector (equations (1)–(3)) is a dynamic representation of the relevant part of the image input at time t. We define a mechanism that computes from the annotation vectors corresponding to the features extracted at different image locations. For each location *i*, the mechanism generates a positive weight which can be interpreted either as the probability that location *i* is the right place to focus for producing the next word (the “hard” but stochastic attention mechanism), or as the relative importance to give to location *i* in blending the ’s together. The weight of each annotation vector is computed by an attention model for which we use a multilayer perceptron conditioned on the previous hidden state . The soft version of this attention mechanism was introduced by Bahdanau et al. (2014). For emphasis, we note that the hidden state varies as the output RNN advances in its output sequence: “where” the network looks next depends on the sequence of words that has already been generated.

 (4)

 (5)

Once the weights (which sum to one) are computed, the context vector is computed by

 (6)

where is a function that returns a single vector given the set of annotation vectors and their corresponding weights. The details of function are discussed in Sec. 4.

The initial memory state and hidden state of the LSTM are predicted by an average of the annotation vectors fed through two separate MLPs (init,c and init,h):





In this work, we use a deep output layer (Pascanu et al., 2014) to compute the output word probability given the LSTM state, the context vector and the previous word:

 (7)

where ,, and E are learned parameters initialized randomly.

1. **Learning Stochastic “Hard” vs Deterministic “Soft” Attention**

In this section we discuss two alternative mechanisms for the attention model : stochastic attention and deterministic attention.

**4.1 Stochastic “Hard” Attention**

We represent the location variable as where the model decides to focus attention when generating the word. is an indicator one-hot variable which is set to 1 if the *i*-th location (out of *L*) is the one used to extract visual features. By treating the attention locations as intermediate latent variables, we can assign a multinoulli distribution parametrized by , and view as a random variable:

 (8)

 (9)

We define a new objective function *L*s that is a variational lower bound on the marginal log-likelihood log *p*(y | a) of observing the sequence of words y given image features a. The learning algorithm for the parameters *W* of the models can be derived by directly optimizing *Ls*:





 (10)

 (11)

Equation 11 suggests a Monte Carlo based sampling approximation of the gradient with respect to the model parameters. This can be done by sampling the location from a multinouilli distribution defined by Equation 8.



 (12)

A moving average baseline is used to reduce the variance in the Monte Carlo estimator of the gradient, following Weaver & Tao (2001). Similar, but more complicated variance reduction techniques have previously been used by Mnih et al. (2014) and Ba et al. (2014). Upon seeing the mini-batch, the moving average baseline is estimated as an accumulated sum of the previous log likelihoods with exponential decay:



To further reduce the estimator variance, an entropy term on the multinouilli distribution *H*[*s*] is added. Also, with probability 0.5 for a given image, we set the sampled attention location to its expected value . Both techniques improve the robustness of the stochastic attention learning algorithm. The final learning rule for the model is then the following:



where, and are two hyper-parameters set by crossvalidation. As pointed out and used in Ba et al. (2014) and Mnih et al. (2014), this is formulation is equivalent to the REINFORCE learning rule (Williams, 1992), where the reward for the attention choosing a sequence of actions is a real value proportional to the log likelihood of the target sentence under the sampled attention trajectory.

In making a hard choice at every point, from Equation 6 is a function that returns a sampled at every point in time based upon a multinouilli distribution parameterized by .

4.2 Deterministic “Soft” Attention

Learning stochastic attention requires sampling the attention location each time, instead we can take the expectation of the context vector directly,

 (13)

and formulate a deterministic attention model by computing a soft attention weighted annotation vector as introduced by Bahdanau et al. (2014). This corresponds to feeding in a soft weighted context into the system. The whole model is smooth and differentiable under the deterministic attention, so learning end-to-end is trivial by using standard backpropagation.

Learning the deterministic attention can also be understood as approximately optimizing the marginal likelihood in Equation 10 under the attention location random variable from Sec. 4.1. The hidden activation of LSTM is a linear projection of the stochastic context vector followed by non-linearity. To the first order Taylor approximation, the expected value is equal to computing using a single forward prop with the expected context vector . Considering Eq. 7, let , denotes computed by setting the random variable value to . We define the normalized weighted geometric mean for the softmax word prediction:



The equation above shows the normalized weighted geometric mean of the caption prediction can be approximated well by using the expected context vector, where. It shows that the NWGM of a softmax unit is obtained by applying softmax to the expectations of the underlying linear projections. Also, from the results in (Baldi & Sadowski, 2014), under softmax activation. That means the expectation of the outputs over all possible attention locations induced by random variable is computed by simple feedforward propagation with expected context vector . In other words, the deterministic attention model is an approximation to the marginal likelihood over the attention locations.

4.2.1. DOUBLY STOCHASTIC ATTENTION

By construction, as they are the output of a softmax. In training the deterministic version of our model we introduce a form of doubly stochastic regularization, where we also encourage . This can be interpreted as encouraging the model to pay equal attention to every part of the image over the course of generation. In our experiments, we observed that this penalty was important quantitatively to improving overall BLEU score and that qualitatively this leads to more rich and descriptive captions. In addition, the soft attention model predicts a gating scalar from previous hidden state at each time step *t*, such that, , where . We notice our attention weights put more emphasis on the objects in the images by including the scalar .

Concretely, the model is trained end-to-end by minimizing the following penalized negative log-likelihood:

 (14)

**4.3. Training Procedure**

Both variants of our attention model were trained with stochastic gradient descent using adaptive learning rate algorithms. For the Flickr8k dataset, we found that RMSProp (Tieleman & Hinton, 2012) worked best, while for Flickr30k/MS COCO dataset we used the recently proposed Adam algorithm (Kingma & Ba, 2014).

To create the annotations used by our decoder, we used the Oxford VGGnet (Simonyan & Zisserman, 2014) pretrained on ImageNet without finetuning. In principle however, any encoding function could be used. In addition, with enough data, we could also train the encoder from scratch (or fine-tune) with the rest of the model. In our experiments we use the 1414512 feature map of the fourth convolutional layer before max pooling. This means our decoder operates on the flattened 196512 (i.e *LD*) encoding.

As our implementation requires time proportional to the length of the longest sentence per update, we found training on a random group of captions to be computationally wasteful. To mitigate this problem, in preprocessing we build a dictionary mapping the length of a sentence to the corresponding subset of captions. Then, during training we randomly sample a length and retrieve a mini-batch of size 64 of that length. We found that this greatly improved convergence speed with no noticeable diminishment in performance. On our largest dataset (MS COCO), our soft attention model took less than 3 days to train on an NVIDIA Titan Black GPU.

In addition to dropout (Srivastava et al., 2014), the only other regularization strategy we used was early stopping on BLEU score. We observed a breakdown in correlation between the validation set log-likelihood and BLEU in the later stages of training during our experiments. Since BLEU is the most commonly reported metric, we used BLEU on our validation set for model selection.

In our experiments with soft attention, we also used Whetlab1 (Snoek et al., 2012; 2014) in our Flickr8k experiments. Some of the intuitions we gained from hyperparameter regions it explored were especially important in our Flickr30k and COCO experiments.

We make our code for these models based in Theano (Bergstra et al., 2010) publicly available upon publication to encourage future research in this area.

1. **Experiments**

We describe our experimental methodology and quantitative results which validate the effectiveness of our model for caption generation.

**5.1. Data**

We report results on the popular Flickr8k and Flickr30k dataset which has 8,000 and 30,000 images respectively as well as the more challenging Microsoft COCO dataset which has 82,783 images. The Flickr8k/Flickr30k dataset both come with 5 reference sentences per image, but for the MS COCO dataset, some of the images have references in excess of 5 which for consistency across our datasets we discard. We applied only basic tokenization to MS COCO so that it is consistent with the tokenization present in Flickr8k and Flickr30k. For all our experiments, we used a fixed vocabulary size of 10,000.

Results for our attention-based architecture are reported in Table 4.2.1. We report results with the frequently used BLEU metric2 which is the standard in the caption generation literature. We report BLEU from 1 to 4 with-out a brevity penalty. There has been, however, criticism of BLEU, so in addition we report another common metric METEOR (Denkowski & Lavie, 2014), and compare whenever possible.

**5.2. Evaluation Procedures**

A few challenges exist for comparison, which we explain here. The first is a difference in choice of convolutional feature extractor. For identical decoder architectures, using more recent architectures such as GoogLeNet or Oxford VGG Szegedy et al. (2014), Simonyan & Zisserman (2014) can give a boost in performance over using the AlexNet (Krizhevsky et al., 2012). In our evaluation, we compare directly only with results which use the comparable GoogLeNet/Oxford VGG features, but for METEOR comparison we note some results that use AlexNet.

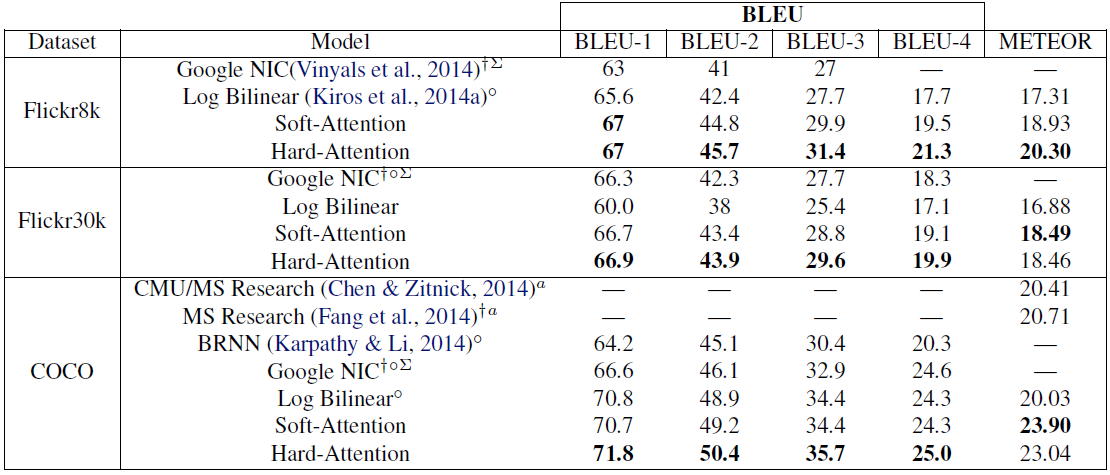
The second challenge is a single model versus ensemble comparison. While other methods have reported performance boosts by using ensembling, in our results we report a single model performance.

Finally, there is challenge due to differences between dataset splits. In our reported results, we use the predefined splits of Flickr8k. However, one challenge for the Flickr30k and COCO datasets is the lack of standardized splits. As a result, we report with the publicly available splits3 used in previous work (Karpathy & Li, 2014). In our experience, differences in splits do not make a substantial difference in overall performance, but we note the differences where they exist.

**5.3. Quantitative Analysis**

In Table 4.2.1, we provide a summary of the experiment validating the quantitative effectiveness of attention. We obtain state of the art performance on the Flickr8k, Flickr30k and MS COCO. In addition, we note that in our experiments we are able to significantly improve the state of the art performance METEOR on MS COCO that we speculate is connected to some of the regularization techniques we used 4.2.1 and our lower level representation. Finally, we also note that we are able to obtain this performance using a single model without an ensemble.

Table 1. BLEU-1,2,3,4/METEOR metrics compared to other methods, y indicates a different split, (—) indicates an unknown metric, indicates the authors kindly provided missing metrics by personal communication, indicates an ensemble, a indicates using AlexNet



**5.4. Qualitative Analysis: Learning to attend**

By visualizing the attention component learned by the model, we are able to add an extra layer of interpretability to the output of the model (see Fig. 1). Other systems that have done this rely on object detection systems to produce candidate alignment targets (Karpathy & Li, 2014). Our approach is much more flexible, since the model can attend to “non object” salient regions.

The 19-layer OxfordNet uses stacks of 3x3 filters meaning the only time the feature maps decrease in size are due to the max pooling layers. The input image is resized so that the shortest side is 256 dimensional with preserved aspect ratio. The input to the convolutional network is the center cropped 224x224 image. Consequently, with 4 max pooling layers we get an output dimension of the top convolutional layer of 14x14. Thus in order to visualize the attention weights for the soft model, we simply upsample the weights by a factor of 24 = 16 and apply a Gaussian filter. We note that the receptive fields of each of the 14x14 units are highly overlapping.

As we can see in Figure 2 and 3, the model learns alignments that correspond very strongly with human intuition. Especially in the examples of mistakes, we see that it is possible to exploit such visualizations to get an intuition as to why those mistakes were made. We provide a more extensive list of visualizations in Appendix A for the reader.

1. **Conclusion**

We propose an attention based approach that gives state of the art performance on three benchmark datasets using the BLEU and METEOR metric. We also show how the learned attention can be exploited to give more interpretability into the models generation process, and demonstrate that the learned alignments correspond very well to human intuition. We hope that the results of this paper will encourage future work in using visual attention. We also expect that the modularity of the encoder-decoder approach combined with attention to have useful applications in other domains.

**Acknowledgments**

The authors would like to thank the developers of Theano (Bergstra et al., 2010; Bastien et al., 2012). We acknowledge the support of the following organizations for research funding and computing support: the Nuance Foundation, NSERC, Samsung, Calcul Qu´ebec, Compute Canada, the Canada Research Chairs and CIFAR. The authors would also like to thank Nitish Srivastava for assistance with his ConvNet package as well as preparing the Oxford convolutional network and Relu Patrascu for helping with numerous infrastructure related problems.

**References**

Ba, Jimmy Lei, Mnih, Volodymyr, and Kavukcuoglu, Koray. Multiple object recognition with visual attention. arXiv:1412.7755, December 2014.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473, September 2014.

Baldi, Pierre and Sadowski, Peter. The dropout learning algorithm. Artificial intelligence, 210:78–122, 2014.

Bastien, Frederic, Lamblin, Pascal, Pascanu, Razvan, Bergstra, James, Goodfellow, Ian, Bergeron, Arnaud, Bouchard, Nicolas, Warde-Farley, David, and Bengio, Yoshua. Theano: new features and speed improvements. Submited to the Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

Bergstra, James, Breuleux, Olivier, Bastien, Fr´ed´eric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for Scientific Computing Conference (SciPy), 2010.

Chen, Xinlei and Zitnick, C Lawrence. Learning a recurrent visual representation for image caption generation. arXiv preprint arXiv:1411.5654, 2014.

Cho, Kyunghyun, van Merrienboer, Bart, Gulcehre, Caglar, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In EMNLP, October 2014.

Corbetta, Maurizio and Shulman, Gordon L. Control of goaldirected and stimulus-driven attention in the brain. Nature reviews neuroscience, 3(3):201–215, 2002.

Denil, Misha, Bazzani, Loris, Larochelle, Hugo, and de Freitas, Nando. Learning where to attend with deep architectures for image tracking. Neural Computation, 2012.

Denkowski, Michael and Lavie, Alon. Meteor universal: Language specific translation evaluation for any target language. In Proceedings of the EACL 2014 Workshop on Statistical Machine Translation, 2014.

Donahue, Jeff, Hendrikcs, Lisa Anne, Guadarrama, Segio, Rohrbach, Marcus, Venugopalan, Subhashini, Saenko, Kate, and Darrell, Trevor. Long-term recurrent convolutional networks for visual recognition and description. arXiv:1411.4389v2, November 2014.

Elliott, Desmond and Keller, Frank. Image description using visual dependency representations. In EMNLP, 2013.

Fang, Hao, Gupta, Saurabh, Iandola, Forrest, Srivastava, Rupesh, Deng, Li, Doll´ar, Piotr, Gao, Jianfeng, He, Xiaodong, Mitchell, Margaret, Platt, John, et al. From captions to visual concepts and back. arXiv:1411.4952, November 2014.

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

Hodosh, Micah, Young, Peter, and Hockenmaier, Julia. Framing image description as a ranking task: Data, models and evaluation metrics. Journal of Artificial Intelligence Research, pp. 853–899, 2013.

Karpathy, Andrej and Li, Fei-Fei. Deep visual-semantic alignments for generating image descriptions. arXiv:1412.2306, December 2014.

Kingma, Diederik P. and Ba, Jimmy. Adam: A Method for Stochastic Optimization. arXiv:1412.6980, December 2014.

Kiros, Ryan, Salahutdinov, Ruslan, and Zemel, Richard. Multimodal neural language models. In International Conference on Machine Learning, pp. 595–603, 2014a.

Kiros, Ryan, Salakhutdinov, Ruslan, and Zemel, Richard. Unifying visual-semantic embeddings with multimodal neural language models. arXiv:1411.2539, November 2014b.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey. ImageNet classification with deep convolutional neural networks. In NIPS. 2012.

Kulkarni, Girish, Premraj, Visruth, Ordonez, Vicente, Dhar, Sagnik, Li, Siming, Choi, Yejin, Berg, Alexander C, and Berg, Tamara L. Babytalk: Understanding and generating simple image descriptions. PAMI, IEEE Transactions on, 35(12):2891– 2903, 2013.

Kuznetsova, Polina, Ordonez, Vicente, Berg, Alexander C, Berg, Tamara L, and Choi, Yejin. Collective generation of natural image descriptions. In Association for Computational Linguistics.ACL, 2012.

Kuznetsova, Polina, Ordonez, Vicente, Berg, Tamara L, and Choi, Yejin. Treetalk: Composition and compression of trees for image descriptions. TACL, 2(10):351–362, 2014.

Larochelle, Hugo and Hinton, Geoffrey E. Learning to combine foveal glimpses with a third-order boltzmann machine. In NIPS, pp. 1243–1251, 2010.

Li, Siming, Kulkarni, Girish, Berg, Tamara L, Berg, Alexander C, and Choi, Yejin. Composing simple image descriptions using web-scale n-grams. In Computational Natural Language Learning. ACL, 2011.

Lin, Tsung-Yi, Maire, Michael, Belongie, Serge, Hays, James, Perona, Pietro, Ramanan, Deva, Doll´ar, Piotr, and Zitnick, C Lawrence. Microsoft coco: Common objects in context. In ECCV, pp. 740–755. 2014.

Mao, Junhua, Xu, Wei, Yang, Yi, Wang, Jiang, and Yuille, Alan. Deep captioning with multimodal recurrent neural networks (m-rnn). arXiv:1412.6632, December 2014.

Mitchell, Margaret, Han, Xufeng, Dodge, Jesse, Mensch, Alyssa, Goyal, Amit, Berg, Alex, Yamaguchi, Kota, Berg, Tamara, Stratos, Karl, and Daum´e III, Hal. Midge: Generating image descriptions from computer vision detections. In European Chapter of the Association for Computational Linguistics, pp. 747–756. ACL, 2012.

Mnih, Volodymyr, Hees, Nicolas, Graves, Alex, and Kavukcuoglu, Koray. Recurrent models of visual attention. In NIPS, 2014.

Pascanu, Razvan, Gulcehre, Caglar, Cho, Kyunghyun, and Bengio, Yoshua. How to construct deep recurrent neural networks. In ICLR, 2014.

Rensink, Ronald A. The dynamic representation of scenes. Visual cognition, 7(1-3):17–42, 2000.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge, 2014.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical bayesian optimization of machine learning algorithms. In NIPS, pp. 2951–2959, 2012.

Snoek, Jasper, Swersky, Kevin, Zemel, Richard S, and Adams, Ryan P. Input warping for bayesian optimization of nonstationary functions. arXiv preprint arXiv:1402.0929, 2014.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. JMLR, 15, 2014.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc VV. Sequence to sequence learning with neural networks. In NIPS, pp. 3104– 3112, 2014.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. arXiv preprint arXiv:1409.4842, 2014.

Tang, Yichuan, Srivastava, Nitish, and Salakhutdinov, Ruslan R. Learning generative models with visual attention. In NIPS, pp. 1808–1816, 2014.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5 - rmsprop. Technical report, 2012.

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. arXiv:1411.4555, November 2014.

Weaver, Lex and Tao, Nigel. The optimal reward baseline for gradient-based reinforcement learning. In Proc. UAI’2001, pp. 538–545, 2001.

Williams, Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4):229–256, 1992.

Yang, Yezhou, Teo, Ching Lik, Daum´e III, Hal, and Aloimonos, Yiannis. Corpus-guided sentence generation of natural images. In EMNLP, pp. 444–454. ACL, 2011.

Young, Peter, Lai, Alice, Hodosh, Micah, and Hockenmaier, Julia. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. TACL, 2:67–78, 2014.

Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. arXiv preprint arXiv:1409.2329, September 2014.

## 英译中译文

摘要

受到机器翻译和物体检测方面近期工作的启发，我们引入了一种自动学习描述图像内容的注意力模型。我们描述了如何使用标准反向传播技术以确定性的方式训练该模型，并通过最大化变分下限来随机地训练该模型。我们还通过可视化显示了模型如何能够自动学习将注视力固定在突出物体上，同时在输出序列中生成相应的单词。我们通过三个基准数据集Flickr8k，Flickr30k和MS COCO来验证使用最新的性能。

**一 引言**

自动生成图像的标题是一个非常接近场景理解的核心的任务，这是计算机视觉的主要目标之一。字幕生成模型不仅能够强大到足以解决计算机视觉中确定哪些对象在图像中的挑战，而且还必须能够以自然语言捕获和表达自己的关系。因此，长期以来，字幕产生被认为是一个难题。对于机器学习算法来说，这是一个非常重要的挑战，因为它相当于模仿了将大量显着视觉信息压缩成描述语言的显着人类能力。

尽管这项任务具有挑战性，但近来对攻击图像字幕生成问题的兴趣日益激增。在训练神经网络（Krizhevsky等人，2012）和大型分类数据集（Russakovsky等人，2014）的进展的帮助下，近来的工作使用卷积神经网络（convnets）的组合来显着提高字幕生成的质量，从而获得图像的矢量表示和循环神经网络将这些表示解码成自然语言句子（见第2节）。

人类视觉系统最好奇的一个方面是注意力的存在（Rensink，2000; Corbetta＆Shulman，2002）。不要将整个图像压缩成静态表示，而是注意允许突出特征根据需要动态地达到最前沿。当图像中有很多混乱时，这尤其重要。使用将图像中的信息提取到最突出的对象上的表示（例如来自于convnet的顶层的表示）是在以前的工作中被广泛采用的一种有效的解决方案。不幸的是，这有一个丢失信息的潜在缺点，这对于更丰富，更具描述性的字幕可能是有用的。使用更多的低级代表可以帮助保留这些信息。然而，使用这些功能需要一个强大的机制来引导模型到对手头任务重要的信息。

在本文中，我们描述了字幕生成方法，试图将注意形式与两种变体相结合：“硬”注意机制和“软”注意机制。我们还展示了包括关注在内的一个优点是能够可视化模型“看到”的能力。受到最近在字幕生成中的进步的鼓舞，受到机器翻译中最近成功使用（Bahdanau等人，2014）和对象识别（Ba等人，2014; Mnih等，2014）的启发，我们调查了可以在产生图片的同时参与图像的显着部分。

本文的贡献如下：

⚫我们在一个共同的框架下引入了两个基于注意力的图像字幕发生器（第3.1.1节）通过标准反向传播方法可训练的“软”确定性注意机制，以及2）通过最大限度地实现可训练的“硬”随机注意机制近似变分下限或等价于REINFORCE（Williams，1992）。

⚫我们展示如何通过可视化“在哪里”和“什么”的注意力来获得洞察力并解释这个框架的结果。（见5.4节）

⚫最后，我们在三个基准数据集上对Flipr8k（Hodosh等，2013），Flickr30k（Young 等人，2014）和MS COCO数据集（Lin等人，2014）。

**二 相关工作**

在本节中，我们提供了关于图像字幕生成和关注的以前工作的相关背景。近来，已经提出了几种用于生成图像描述的方法。许多这些方法都是基于复发性神经网络的，灵感来自成功使用序列来训练神经网络进行机器翻译（Cho等人，2014; Bahdanau等人，2014; Sutskever等，2014）。图像字幕生成非常适合机器翻译的编码器 - 解码器框架（Cho等人，2014）的一个主要原因是因为它类似于将图像“翻译”为句子。

使用神经网络进行字幕生成的第一种方法是Kiros等人（2014a），他提出了一种多边形对数双线性模型，该模型被图像的特征所偏好。此后，Kiros等人进行了这项工作。 （2014b），其方法旨在明确地允许自然地进行排名和生成。毛等人（2014年）采用了类似的方法来代替一种反复的前馈神经语言模型。 Vinyals等人（2014）和Donahue等人（2014）使用LSTM RNN作为模型。不像Kiros等（2014a）和Mao等（2014），其模型在输出字序列的每个时间步长中看到图像，Vinyals等人（2014）仅在首先将图像显示给RNN。随着图像，Donahue等人（2014）也将LSTM应用于视频，让他们的模型生成视频描述。

所有这些作品将图像表示为来自预处理卷积网络顶层的单个特征向量。 Karpathy＆Li（2014）提出学习一个联合嵌入空间，用于排名和生成，其模型学习将句子和图像相似度作为具有双向RNN输出的R-CNN对象检测的函数。方等（2014）提出了一个三步骤的生成流程，包括对象检测。他们的模型首先基于多实例学习框架来学习几种视觉概念的检测器。然后将对字幕训练的语言模型应用于检测器输出，然后从联合图像文本嵌入空间中重新获取。与这些模型不同，我们提出的注意框架没有明确地使用对象检测器，而是从头开始学习潜在的对齐。这允许我们的模型超越“客观性”，并学习参与抽象概念。

在使用神经网络产生字幕之前，主要有两种方法。第一个涉及基于对象检测和属性发现的结果（Kulkarni等人，2013；Li等，2011；Yang等，2011；Mitchell等人，2012；Elliott＆Keller，2013）。第二种方法是基于首先从大型数据库检索相似的标题图像，然后修改这些检索的字幕以适合查询（Kuznetsova等，2012; 2014）。这些方法通常涉及中间“泛化”步骤，以移除仅与检索到的图像相关的标题的细节，例如城市的名称。这两种方法都不利于现在主导的神经网络方法。

以前的工作已经有很长一段时间，将注意力集中到神经网络中，用于视觉相关任务。一些与我们工作同样精神的人包括Larochelle＆Hinton（2010）; Denil等（2012）；唐等（2014）。特别是，我们的工作直接扩展了Bahdanau等人的工作。（2014）; Mnih等人（2014）以及 Ba等人（2014）。

**三 基于注意力机制的图像标题生成**

**3.1模型细节**

在本节中，我们首先描述其共同框架，来描述我们注意力模型的两个变体。主要区别在于我们在第4节中详细描述的函数的定义。我们用粗体字母和大写字母表示向量。在下面的描述中，我们抑制偏差条件的可读性。

**3.1.1 编码器：卷积特征**

我们的模型采用单个原始图像，并生成编码为1-K编码字的序列的标题y。



其中K是词汇的大小，C是标题的长度。

我们使用卷积神经网络来提取一组我们称为注释向量的特征向量。提取器产生L个向量，每个向量是对应于图像的一部分的D维表示。



为了获得特征向量和2-D图像的部分之间的对应关系，我们从以前的工作中提取来自较低卷积层的特征，而不是使用完全连接的层。这允许解码器通过选择所有特征向量的子集来选择性地聚焦在图像的某些部分上。

* + 1. **解码器：长短期记忆**

我们使用长短期记忆（LSTM）网络（Hochreiter＆Schmidhuber，1997），它通过在条件是上下文矢量，先前的隐藏状态和先前产生的单词的每个时间步长上产生一个单词来产生一个字幕。我们的LSTM的实施与Zaremba等人（2014）的使用密切相关。（见图4）。使用表示具有学习参数的简单仿射变换，

这里，, , , 分别是LSTM的输入、遗忘、内存、输出和隐藏状态。向量是上下文矢量，捕获与特定输入位置相关联的视觉信息，如下所述。是嵌入矩阵。令m和n分别表示嵌入和LSTM维数，和分别为逻辑S形激活和元素乘法。

简单来说，上下文矢量（等式（1）-（3））是在时间t的图像输入的相关部分的动态表示。我们定义一个从注释向量计算的机制对应于在不同图像位置处提取的特征。对于每个位置i，该机制产生一个正的权重，其可以被解释为位置i是用于产生下一个单词的正确的地点的概率（“硬”但随机的注意机制），或者作为相对重要性给我的位置我混合在一起的。每个注释向量的权重通过注意模型来计算，对于该注解模型，我们使用以先前隐藏状态为条件的多层感知器。2014年，Bahdanau等人介绍了这种注意机制的软版本。为了强调，我们注意到，隐藏状态随着输出RNN在其输出序列中前进而变化：“其中”网络看起来取决于已经生成的单词的顺序。