指导教师签名栏	侯凤贞
	签名即确认纸质版与电子版终稿一致。

中国药科大学

本　科　毕　业　论　文

论文题目　小学数学题自动解答系统设计与开发

英文题目　Design and Development of Primary School Mathematics Question Automatic Answer System

专　　业　信息管理与信息系统专业

院　　部　理学院

学　　号　2020170837

姓　　名　卢艳峰

指导教师　侯凤贞

课题完成场所　中国药科大学

论文工作时间：　2021　年　2　月　至　2021　年　5　月

承诺书

本人所呈交的纸质版毕业论文（设计）是个人在导师的指导下进行研究工作所取得的原创性研究成果，且与上传至学校毕业设计（论文）智能管理系统中的电子版终稿完全一致。

除参考文献中所列出的内容外，本论文（设计）不包含其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，也已在文中明确说明并表示谢意。

在此，本人郑重承诺：如经核查，发现本人存在剽窃他人研究成果等学术不端行为，本人愿承担责任，并接受学校或相关部门关于学历、学位等方面的处理，且不受时间的限制。

作者签名：

日期：2021 年 6 月 7 日

小学数学题自动解答系统设计与开发

目录

小学数学题自动解答系统设计与开发

2020170837　卢艳峰

摘要：

本文利用深度学习技术对分数应用题解答算式做出预测并完成了 Android 系统 App1.0 版本的开发。本研究使用卷积循环神经网络（Convolutional Recurrent Neural Network, CRNN）识别图片中的文字，在 Caffe-ocr 中文合成数据集上精度为 96.7%，通过比较 CRNN 存在的问题，提出了一种能够一次识别多行文字的神经网络模型。然后通过对比传统的循环神经网络（Recurrent Neural Network, RNN）和 Transformer 神经网络，本文最终采用 Transformer 神经网络模型预测分数应用题的解题算式，在自制的分数应用题的数据集上达到 96.8%的准确率。最后，使用 Kotlin 语言在 Android Studio 上开发 App，使用 Chaquopy 在移动端集成了 Transformer 神经网络模型。研究成果表明，深度学习拥有自动解答小学数学应用题，尤其是分数应用题的能力；对于分数应用题的解题算式预测，宜采用具有较强时序关系的编码器和较弱时序关系的解码器的深度学习模型，不宜通过条件概率的方法生成目标字符序列。本文的研究为小学数学题自动解答系统的设计与开发提供了新的思路。

关键词：自我注意力机制；文字识别；安卓开发；

Design and Development of Primary School Mathematics Question Automatic Answer System

Abstract:

In this paper, deep learning technology is used to predict the solution formula of fraction application problems, and the development of Android App 1.0 is completed. In this study, Convolutional Recurrent Neural Network (CRNN) is used to recognize the text in the picture, and the accuracy is 96.7% on the Caffe OCR Chinese composite data set. By comparing the problems of CRNN, a neural network model which can recognize multiple lines of text at one time is proposed. Then, by comparing the traditional Recurrent Neural Network (RNN) and Transformer Neural Network, this paper finally uses the Transformer Neural Network to predict the solution formula of the fraction application problems, and achieves 96.8% accuracy on the self-made data set of the fraction application problems. Finally, Kotlin language is used to develop app on Android Studio, and Chaquopy is used to integrate Transformer Neural Network on mobile terminal. The research results show that deep learning has an ability to automatically solve primary school mathematics application problems, especially the fraction application problems; The deep learning model of encoder with strong temporal relationship and decoder with weak temporal relationship should be adopted for the prediction of solving formula of fraction application problems, and the target character sequence should not be generated by conditional probability method. The research of this paper provides a new idea for the design and development of the primary school mathematics automatic answer system.

Keywords: Transformer; Optical Character Recognition; Android

第一章 绪论

1.1 研究背景

近几年，机器学习的研究者利用深度学习实现了聊天机器人的搭建[1]，图像的自动描述[2]，机器的算法学习[3]，这些成果已经证明了神经网络有能力模拟人类的思考方式，如人类语言的语法、语义关系甚至是编程语言中蕴含的复杂的算法结构[4]。

目前，市面上出现了很多用于帮助学生解题的 App，如“作业帮”、“小猿搜题”、“学小易”等。以上的习题软件都是通过 Client/Server（客户机/服务器）模式进行搭建运行。首先，用户在客户机端的 App 上输入需要解答的题目，然后 App 将用户输入的题目通过互联网传输到后端的服务器，服务器在题库查找用户输入的题目并将搜索到结果回传到客户机，最终用户在客户机端的 App 上看到服务器返回的结果。

目前的帮助学生自主学习的 App 有以下问题：

(1) 在服务器端，公司需要部署数据库用于存放海量题目，这一措施引起的成本包括题目的获取成本、商业级数据库管理系统的购买成本和维护成本等。

(2) 在服务器端，公司需要开发类似搜索引擎的搜题系统用于从数据库中获取与用户输入的相同或类似的习题参考答案，增加了 App 的开发难度和人力成本。

(3) 客户机和服务器之间数据交换需要通过互联网进行，用户体验易受到网络延迟的影响。

(4) 公司需要定期的扩充题库，并且需要保证习题的一致性即同一习题不能出现几种明显相左的答案，但可以有多种解题方法。

(5) 由于不同 App 服务器端的数据库范围不同且没有对接连通，有时用户需要使用多个 App 才能搜索到习题参考答案。

(6) 随着人类社会的发展，习题的情景会发生改变，虽然习题考察的知识即相应的解题模式并没有改变，但题库需要进行更新，增加了公司运维成本。

(7) 在刚进入市场阶段，App 会因为题库数据过少会出现用户搜索不到习题的现象，降低用户的评价，影响 App 长远的发展。

1.2 设计思路

考虑现有习题 App 的优势和不足，本文决定设计一款基于深度学习模型解题的 App。这款 App 的解题流程为：首先，用户在 App 上拍照输入题目或者直接输入题目；然后，深度学习模型利用用户输入的题目描述进行解题；最终，用户在 App 上看到深度学习模型的返回的参考答案。

本文设计的解题 App 的优势如下：

(1) 公司不需要数据库去保存大量习题，因此没有相应的商业级数据库管理系统的购买开销和相应的维护成本。

(2) 不需要内置搜索引擎去搜索数据库中的题目，降低了程序开发成本。

(3) App 自动调用内嵌于客户机端的深度学习模型来实现解题，不需要互联网的支持，能够适应更广阔的应用场景。

(4) 本文设计的深度学习模型属于自然语言处理（Natural Language Processing, NLP）模型，能够随着习题的变化不断进行学习优化，并具有一定的泛化能力。因此公司不需要频繁地更新题库和降低了相应地维护成本。

(5) 本文设计的深度学习模型可以在用户许可的前提下自动的学习用户输入的习题，进而达到“永久成长”的效果，学习到广泛的解题模式，变成真正的具有“智慧”的习题助手。

(6) 深度学习模型能够独立地解决用户提交的习题，因此在 App 刚进入市场阶段，用户不会遇到获取不到参考答案的问题。

目前的 NLP 模型已经能够学习到稳定的人类语言中语法和语义特性，甚至能够达到语用标准，因此本研究希望深度学习模型能够学习到习题中的相对稳定的解题模式并给出习题参考答案。

由于习题科目众多，并且各个学科不同学习阶段都有大量习题，通过仔细地比较各种类型和内容的习题，发现小学三四年级的分数应用题满足本研究的要求，具有以下特点：固定的解题模式、简单的数学计算和稳定的提问方式。下面是一个分数应用题的实例：

问题：一根电线长 49 米，先用去 3/10，后又用去 10 米，这跟电线还剩多少米？

答案：49*(1-3/10)-10=24.30

综上，本研究需要完成以下任务：

(1) 文字识别。App 能够识别图片中的文字，用于用户拍照输入题目。

(2) 深度学习模型的搭建。利用该深度学习模型完成习题的解答。

(3) App 的开发。将文字识别和深度学习模型部署到 Android 系统上，完成解题 App 1.0 版本的开发。

第二章 文字识别

2.1 模型

在本研究中，文字识别的目的是通过某种方法获得用户拍的习题图片中的文字，并且图片中的题目描述文字一般是连续、不定长的，因此选用卷积循环神经网络（Convolutional Recurrent Neural Network, CRNN）[5]去完成文字识别工作。

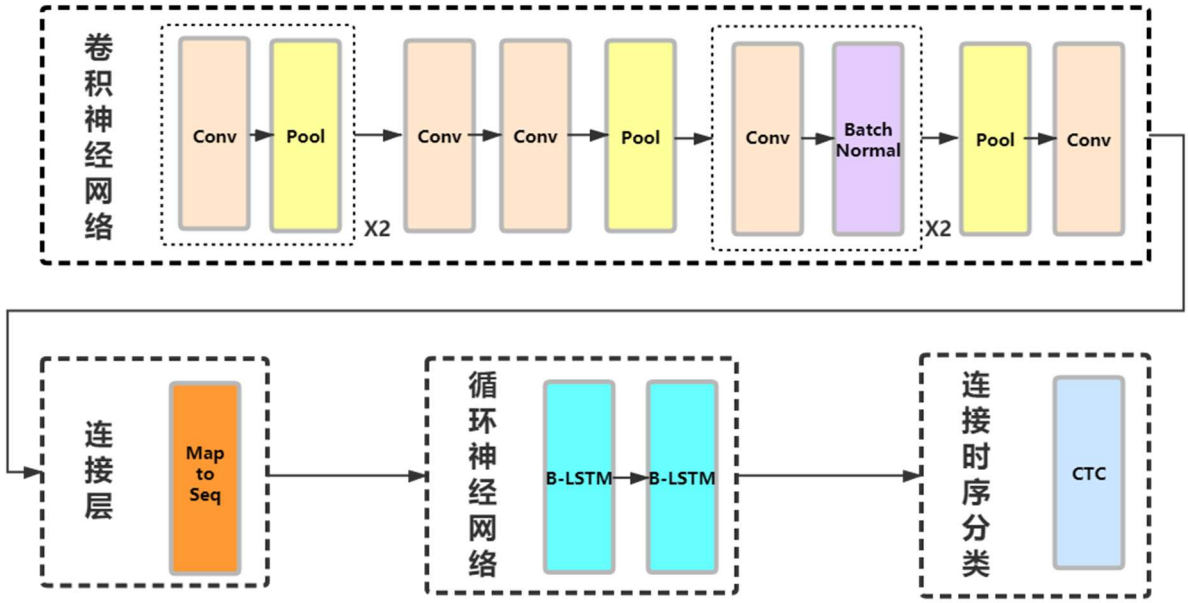

图 2-1 CRNN 结构图

如图 2-1 所示，CRNN 的网络结构分为三部分，第一部分是卷积神经网络（Convolutional Neural Networks, CNN）[6]，第二部分是循环神经网络（Recurrent Neural Network, RNN）[7]，第三部分是连接时序分类（Connectionist Temporal Classification, CTC）[8]。用 Map-to-Sequence 层连接 CNN 的输出与 RNN 的输入，因为 CNN 输出维度与 RNN 输入维度不同，需要进行调整 CNN 输出数据维度的个数和顺序。CNN 经过七次卷积计算、四次 Max 池化运算、两次正则化计算将图像高度维度由 32 降低到 1，通道维度由 1 升到 512，实现了图像在不同维度的线性空间内信息转化和特征提取。单向的长短期记忆神经网络（Long Short Term Memory network, LSTM）[9]只能定向的表达时序关系，因此选

择双向 LSTM 去弥补单向 LSTM 的不足。RNN 能够把 CNN 输出的文字图像特征当作带有时序的信息进而预测出文字序列。CNN 输出的图像的宽度维度会被 RNN 的时间序列维度接收，但是一般最终的文字标签会小于这个时间序列的维度，因此 RNN 的输出结果会存在冗余。最终，CTC 引入 blank 字符计算整个文本标签存在形式的各种可能性的条件概率最大值实现去冗余操作。

2.2 训练

文字识别的数据集为 Caffe-ocr 中文合成数据集，该数据集由 360 万张内容为中文的图片组成，训练集与预测集的比例为 10 比 1，图片分辨率为 280*32。

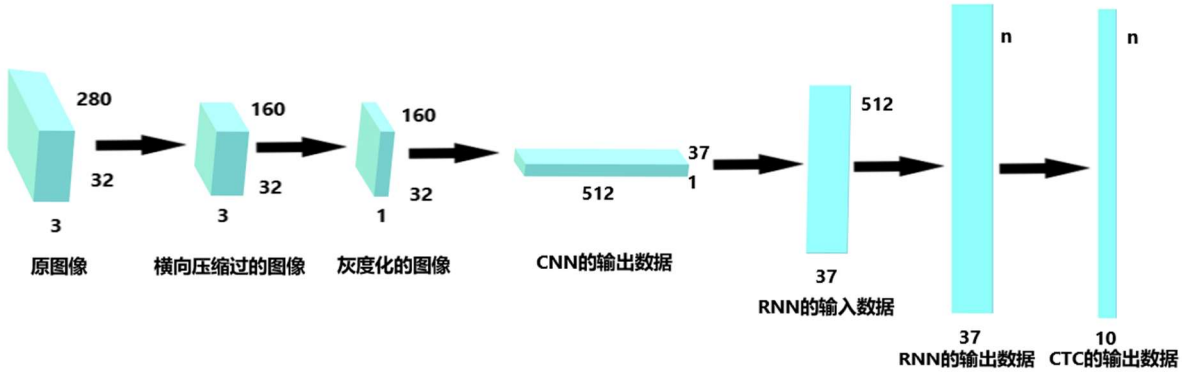

图 2-2 CRNN 中单个图片的数据维度变化图

如图 2-2，首先将原始图片的宽度从 280 像素压缩到 160 像素，然后将图像从彩色图像转变为灰度图像，此时是 CRNN 模型输入的数据维度，为（C*H*W=1*32*160），C 为图片的通道维度，H 为图片的高度维度，W 为图片的宽度维度。CNN 输出结果的数据维度为（C*H*W=512*1*37），这时需要用 Map-to-Sequence 层去除掉数据的高度维度 H 并将数据的维度顺序改为（W*C=37*512），转换成 LSTM 的输入变量的表示形式为（T*Class_num=37*512），T 代表时序维度，Class_num 代表预测字符种类数维度。将该数据传递给 RNN，最终 RNN 的输出数据维度为（T*Class_num=37*n），其中 n 代表字符的种类，本数据集为 6736。可以发现本模型 RNN 预测本数据集的最大字符数量是 37，Caffe-ocr 中文合成数据集数据标签的字符长度一般是 10，即需要通过 CTC 去掉三倍的冗

余度。

2.3 结果分析和讨论

CRNN 模型训练六轮，在预测集上达到 96.7%的预测精度。

下面是关于这个模型的分析与讨论：

(1) 通过分析模型，第一步并不需要横向压缩图片。如图 2-2，CRNN 中的 RNN 并不改变时序维度 T，因此 RNN 输出的数据的时序维度 T 与原始图像的宽成正比。且本模型是通过 CTC 算法去冗余，前两个神经网络并不参与去冗余工作。因此如果第一步不压缩图像的宽度，并不影响前两步神经网络的计算，只是会导致 RNN 的输出数据的时序维度 T 变大，影响 CTC 算法去冗余的倍数。在神经网络学习过程中，学习数据集的处理操作保持统一，因此不会使本模型预测结果下降。

(2) 设想新的文字识别模型。注：本次假设多行文本的行间距为 0。如图 2-2，CRNN 中的 CNN 输出的数据高度维度 H 为 1，然后将该输出数据的高度维度 H 去掉并改变维度顺序然后传入到 RNN。通过上面的描述，想到假如 CNN 输出数据的高度维度 H 不是 1 是 10，将该数据在高度维度 H 方向上进行拆分，拆分成 10 个高度维度 H 为 1 的数据，分别将这个 10 个数据输入到 10 个 RNN 中，进而得到 10 个预测文本输出结果。回到本次文字识别实验，CTC 算法在时间序列维度 T 上将 37 去冗余到 10，证明 CTC 算法具有对齐功能。进一步，假如刚才的猜想的情景是一张具有 10 行像本实验数据集图片中的那样文字的图片，然后将这张图片送到 CRNN 第一部分的 CNN 中，输出数据的高度维度 H 为 37，然后高度维度 H 通过 CTC 算法降低维度到 10，接着在高度维度 H 方向上将数据为拆分成 10 个高度维度 H 为 1 的数据，并将这 10 个数据送入到 10 个 RNN 中，得到 10 个预测文本，然后按照顺序整合，就能获得包含 10 行文本的图片识别结果。该设想最难实现的地方是如何利用 CTC 算法实现动态的改变 RNN 的数量，以上就是本部分的新的文字识别模型的设想。

(3) 不同的图片因为像素、高宽的压缩比、字符间距甚至是字体的不同，导致 CRNN 模型中的 RNN 输出数据的冗余度不同。通过同一数据集学习到的模型会出现 CTC 算法的去冗余程度基本相同的问题，如本实验目前训练好的的模型的去冗余程度大约为 3 倍（去冗余程度为 37 到 10），因此当利用该模型预测与本次实验不同高宽压缩比图像中的

文字时会出现无法识别图片文字的现象。给出的解决方案是通过改变数据集中图片的高宽压缩比扩增数据，然后用扩增的数据训练模型，这样就能得到能够适应不同高宽压缩比的 CRNN 模型。本实验还发现使用训练集的十分之一就已经能达到 95%的精度，在扩大数据集方法方面，因此可以先将总数据集随机分成等数量的 10 份，其中 9 份改变高宽压缩比，然后将已经改变高宽压缩比的 9 份数据集和那一份未作改变的数据集合并成 360 万张图片，分出十一分之一作为预测集，剩余部分作为训练集。

第三章 深度学习模型搭建与结果分析

3.1 循环神经网络模型的搭建与结果分析

3.1.1 模型

本部分运用增加网络深度[10]、权重共享[11]、双向 LSTM，泛化的注意力机制[12]等方法设计出图 3-1 的从时序到时序神经网络（Sequence-to-Sequence, Seq2Seq）[13]。

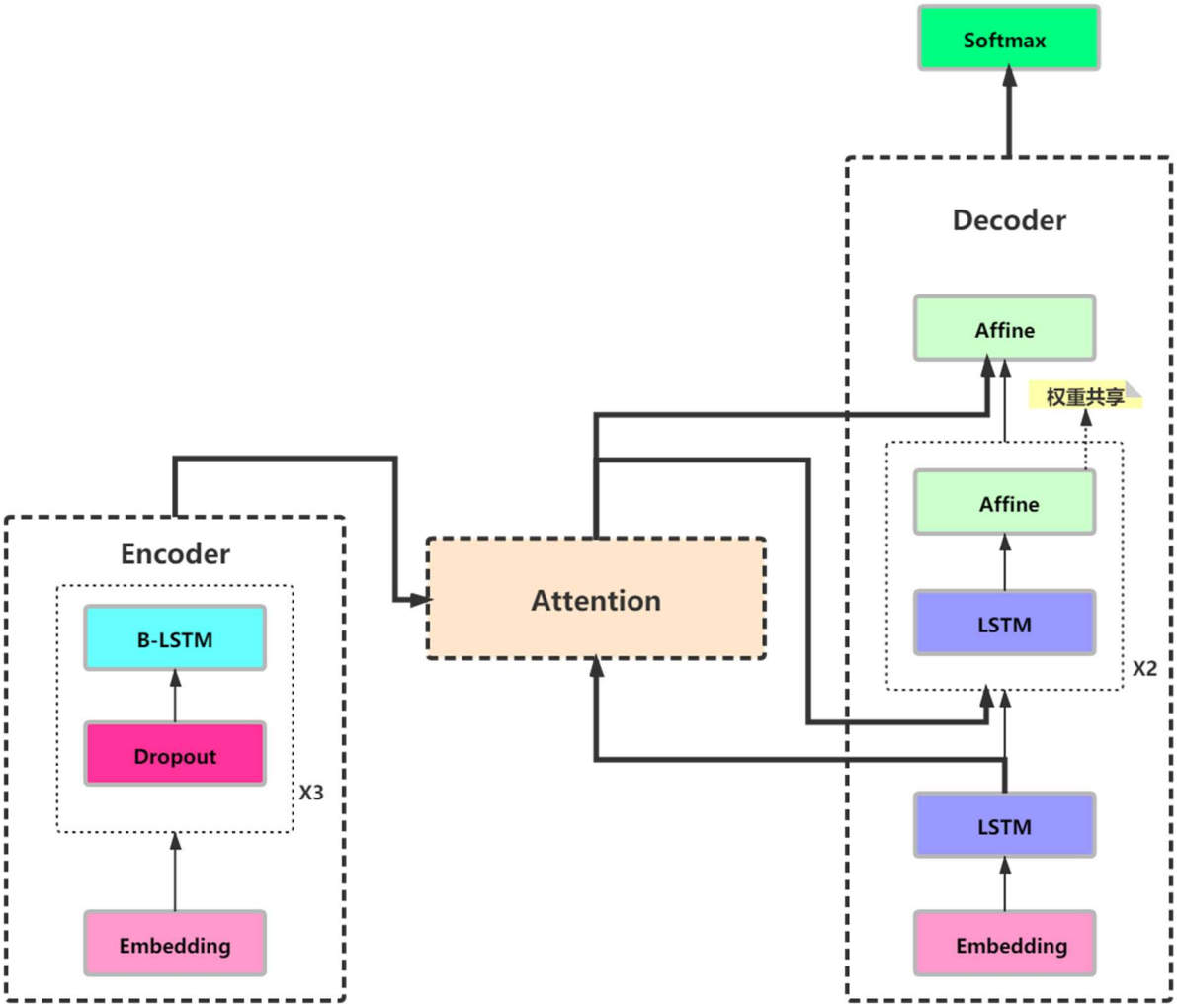

图 3-1 Seq2Seq 模型

如图 3-1 所示，Seq2Seq 包括三部分，第一部分是编码器（Encoder），第二部分是解码器（Decoder），第三部分是注意力机制（Attention）。

如图 3-1 所示，Encoder 首先是 Embedding 层[14]，维度是（V, D=32），其中 V 代表字符种类的维度，D 的代表单词分布式向量的维度，然后是三个双向 LSTM 的网络结构（B-LSTM），并在其中穿插 0.5 抛弃率的 Dropout 层去降低过拟合，Encoder 最终输出的矩阵维度为（T, 8*H=256），T 代表时序维度，H 代表 B-LSTM 处理单个字符的神经元个数。Encoder 中的 B-LSTM 会使每次的输出数据维度 H 扩大一倍。

如图 3-1 所示，Encoder 输出维度为（T,8*H=256）的数据，与 Decoder 的第一层单向的 LSTM 输出数据进行了 Attention 的计算[12]，将 Attention 输出结果和相应层（一般为全连接层（Affine））的输出结果合并输出到下一层（一般为单向的 LSTM），我们使用了 Affine 层降低数据的维度，并将权重共享机制应用到了 Decoder 中重复部分的 Affine 层。最后通过 Softmax 函数计算每种字符的生成概率。

3.1.2 训练

本部分实验从互联网上选择了 500 道分数应用题作为原始数据。数据预处理工作如下：

(1) 将除法用“\”字符表示，分数用“/”字符表示。

(2) 去除原始数据中的相向问题和一些复杂的列式问题。

(3) 通过将问题情景中的主人公、物体名称改变，汉字数字和阿拉伯数字的相互转换进行扩增，最终数据量为 20000 个。

(4) 将习题描述字符串的字符个数规定在 100 以内，去除掉参考答案字符串中的空格。

将预处理完成的数据应用到上面复杂的 Seq2Seq 模型，模型的预测结果为大量的“/”、“\”符号，没有数字被识别出来。

3.1.3 结果分析与讨论

下面是本部分结果分析与讨论：

(1) 较少数量的神经元不能够充分地表达出原始信息。Encoder 中开始阶段的 B-LSTM 的 H 维度等于 Embedding 层的 D 维度为 32，本实验数据集中的字符种类数达到 300 个以上。模型需要用 32 个维度信息去表示 300 个种类，难度极大。当模型应用到英语等高词汇量的环境时，这个问题将会更加突出。

(2) Encoder 中的 B-LSTM 处理单个字符的神经元个数变化不合理。B-LSTM 中的 H

维度从 32 变成了 256。第二章节的 CRNN 中 CNN 部分表明神经网络的学习是数据挖掘和数据转化的过程。第二章节的数据集中的图片输入到 CRNN[5]中时宽高通道维度为（160*32*1），保留了原始图像的信息，CRNN 中的 CNN 部分输出宽高通道维度为（37*1*512），表明卷积是一个把图片中信息从宽高维度转化到通道维度的过程。本节的 Encoder 计算的数据维度为(T*H)，在神经网络学习预测过程中，H 维度蕴含着单个字符的全部信息，并且 H 维度在开始阶段需要包含全部原始信息，而且根据上述卷积过程的启示，需要保证 Encoder 的输入和输出的数据包含的信息量大致稳定，因此本文的 Encoder 的 H 维度不应该改变。

(3) 反向输入问题描述字符串可以提高模型效果[13]。由于填充字符的存在，反向输入原数据能够降低原数据与目标数据的平均距离，进而达到提高模型精度的效果。

(4) 各个层神经元数量太少。在 Seq2Seq 模型中，研究者要求每层的隐藏神经元为 1000 个[13]。

(5) RNN 稳定的时序关系长度较短。分数应用题问题的时序长度在 70 个左右。有研究表明常规 RNN 组成的 Seq2Seq 模型一般只能处理 40 个字符以内问题[13]。

(6) 降低维度增加了需要学习的参数个数。Decoder 中通过使用 Affine 层降低数据维度，增加了需要学习的参数个数。

(7) Attention 机制不合理。本模型是根据 Decoder 的第一层 LSTM 和 Encoder 的输出计算 Attention 的，并且将这个 Attention 传送给了后面的 LSTM 和最后一个 Affine 层，这样做会使得得出的注意力只是根据第一层 LSTM 解码的结果，不够深入。所以可以将 Attention 的 Decoder 部分的输入深度加深，得到更深度的信息，进而得到更好的结果。

(8) 常规 RNN 组成的 Seq2Seq 模型不能解决分数应用题。本部分实验希望深度学习模型能够将分数应用题的描述性文字作为模型的输入，输出分数应用题的参考答案。因为 Seq2Seq 模型是基于翻译问题设计的，即 Decoder 预测出的目标序列相邻词汇间存在语法、情景或者时态的关系即目标序列中各个字符间存在时序关系。但本部分实验的问题中目标序列是四则运算的数学计算式，相邻字符间并不存在时序关系，如“4”字符既可以和加号减号相邻，又可以和乘号除号相邻，甚至它的周围还有小括号；在翻译问题中，“drive”和“car”同时出现可能性就很大，“drive”和“cat”的同时出现的频率就很低。因此该

问题的 Decoder 不需要太强的时序关系，而是需要对问题的深入思考，常规的 RNN 模型搭建的 Seq2Seq 并不能解决本问题。

3.2 自我注意力模型搭建与结果分析

3.2.1 模型

基于 3.1.3 节的讨论，本实验重新选择自我注意力模型（Transformer）作为习题解答的深度学习模型。

Transformer 模型用自我注意力机制完全取代了 Seq2Seq 模型中的 LSTM 网络[15]。注意力机制如图 3-2 左半部分，Transformer 利用向量点积运算实现注意力机制。Q、K 和 V 分别代表需要查询的内容、全部内容的键和全部内容的值。矩阵 Q 与矩阵 K 相乘并且规范化后求概率就获得到了需要查询内容的权重，然后通过这个权重与全部内容的值的乘积就获得到了查询的内容。由于自我注意力机制自身不带有时序关系，Transformer 会出现 Decoder 生成的目标序列前半部分能够学习到后半部分的问题，因此 Decoder 中的自我注意力机制需要 Mask 可选结构解决上面问题。图 3-2 右半部分表示将注意力机制并行化实现为 Multi-Head-Attention 层的过程。图 3-2 并没有显示出自我注意力机制，Q、K 和 V 相同时表示自我注意力机制，如图 3-3。

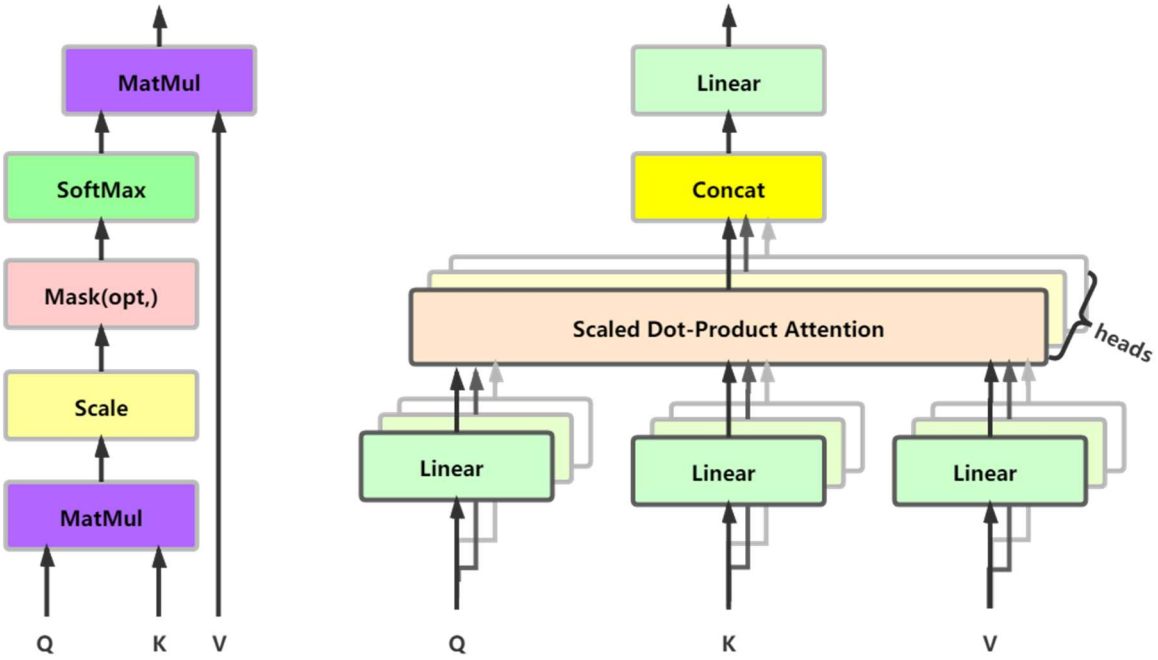

图 3-2 自我注意机制[15]

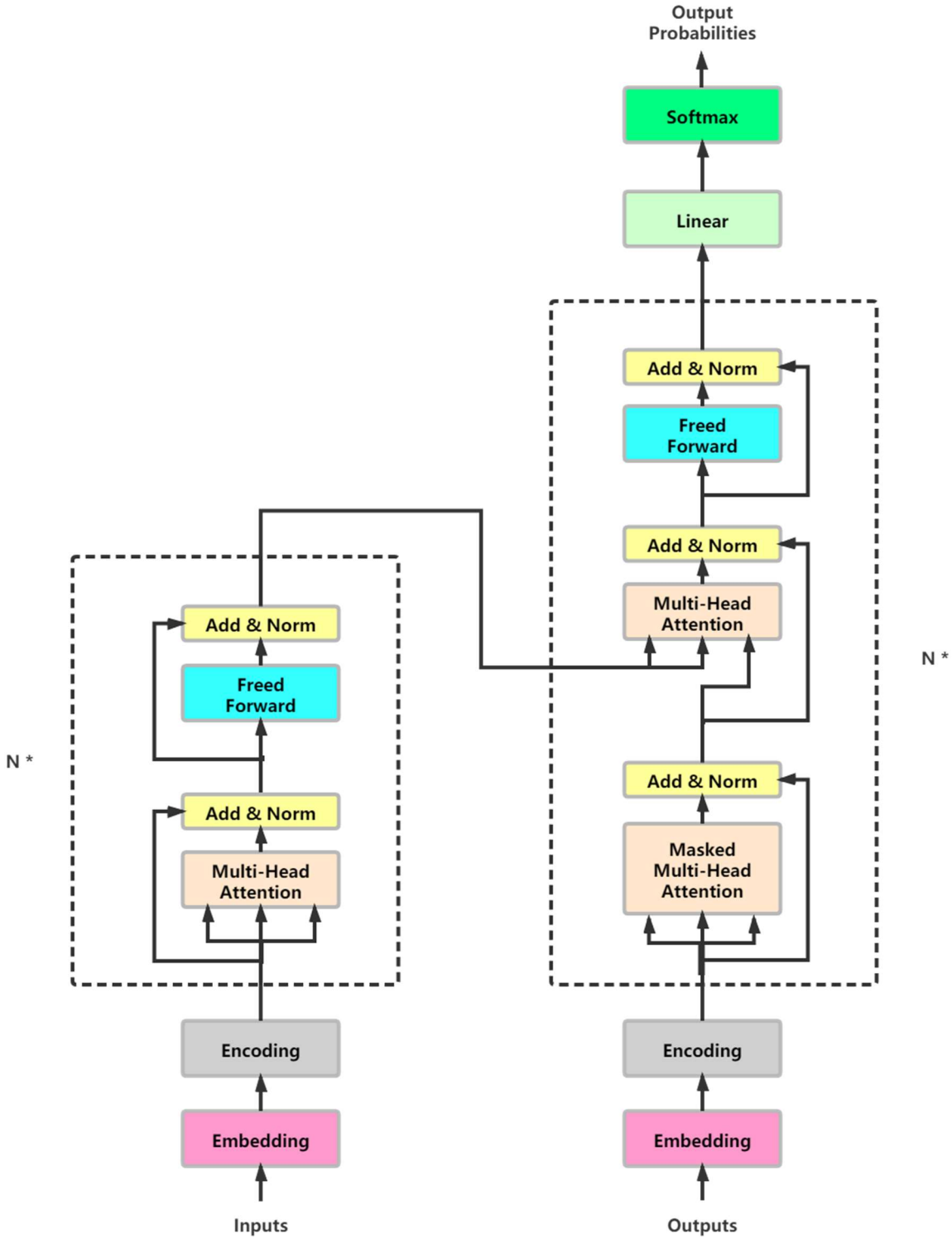

图 3-3 Transformer 模型[15]

Transformer 模型的结构如图 3-3，该模型也是 Encoder-Decoder 模型，Encoder 和

Decoder 的开始部分都是由 Embedding 层构成的[14]。由于 Transformer 没有时序关系，所以为了表示输入字符和输出字符的顺序，引入了正余弦函数的编码层即 Encoding 层。Decoder 最后部分通过 Softmax 函数计算每种字符的生成概率。Encoder 和 Decoder 都具有可以重复的结构，里面包含图 3-2 右半部分的 Multi-Head-Attention 层、全连接的前馈网络和正则化层。Encoder 的 Multi-Head-Attention 层使用三个相同的数据分别作为 Q、K 和 V 的输入，Decoder 的 Masked Multi-Head-Attention 层也使用三个相同的数据分别作为 Q、K 和 V 的输入。Encoder 中的输出作为 Decoder 中间部分的 Multi-Head-Attention 层的 K 和 V，Decoder 中 Masked Multi-Head-Attention 层的输出作为 Decoder 中间部分的 Multi-Head-Attention 层的 Q，通过这个三个变量 K、V、Q 实现了 Encoder 和 Decoder 间的数据交换。

3.2.2 训练

数据集处理过程如下：

(1) 精选 50 道简单分数应用题（习题描述字符串只含有标准分数、不包含百分数、汉字比例等其他描述方式，且参考答案的计算式最多只能有一个小括号），并将习题解题思路固定。

(2) 将所有的“\”字符换成“@”字符代表除法防止程序语言为“\”字符加上转义符号。

(3) 将分数题的描述字符串的字符个数降低到 70 以下。

(4) 将每个分数习题的描述的分数的分母从 1-10，整数从 0-50 进行遍历扩增习题个数，最终分数习题数量从 50 变为 245000。

(5) 编码格式：分数应用题的描述字符串和参考答案字符串的填充字符为 0 号编码的空格字符，填充后的字符串个数都为 70。

(6) 用 8 个 head 的 Encoder 和 Decoder 各三层的 Transformer 去训练，学习 30epoch，模型得到 10%的准确率。

3.2.3 结果分析与讨论

以下是模型改进和结果分析：

(1) 简单的层数叠加并不是好的策略。发现 Transformer 能够有 10%正确率预测出参

考答案算式等号后面小数点后两位的结果，认为加深 Decoder 的层数进而增加目标序列自身的注意力程度可以得到更好的预测结果。实验结果表明将 Decoder 的层数加深到 8 层，会出现模型预测不到数字的情况。有研究表明[6]，卷积神经网络深层化效果变好是由于参数个数的减少和 ReLU 等非线性函数的穿插导致的，本次实验的简单的叠加 Decoder 的深度其实是增大了参数的数量，降低了模型的学习效果。

(2) Decoder 可以单独使用。实验结果表明，目标答案字符串有效字符个数一般是 20，因此模型大部分的损失值都用于处理后面 50 个无效填充字符，导致学习效果不佳。在考虑到最长答案序列后，选择了将有效的答案序列乘以 3 且剩余部分用空格填充的办法去提高有效字符的占比。在本次实验的学习阶段，发现生成参考答案的第二个重复部分变为准确，但是任何一个生成的参考答案第一个重复部分的都是错乱的。然而在本次实验的预测阶段发现三个重复片段并没有预测出标准答案。其他编码形式也没有取得效果，如将空格的填充位置改到目标序列的前面、改到中间或者将目标有效字符串直接乘以 10 截取前 70 位。通过分析网络结构发现是因为 Decoder 的自我注意机制能够通过自身的输入的目标字符串去学习，换句话说 Transformer 的 Decoder 可以单独使用，因此目标序列后半部分的内容可以通过目标序列前半部分获得，除了第一个以外，后面的两个部分的预测是可以将第一部分的结果作为参考的序列。预测阶段与学习阶段效果不同的主要原因是预测阶段与学习阶段目标字符串的生成方式不同，在学习阶段，因为能够获得完整的目标序列降低了损失值，但是这部分损失值在预测的阶段不会降低，进而导致重复片段的第一部分没有办法继续学习。

(3) 编码 0 不应该作为填充字符的编码。通过以上的实验，本模型并不能预测出填充字符，进而导致模型中的大部分损失值无法降低。发现可能是因为填充字符为 0 号编码的空格字符。因为 Transformer 由 Softmax 层和交叉熵误差配合计算损失值，所以神经网络误差反向传播的起点是 y-t，当标签是 0 的时候有可能会出现无法学习的问题，因此将填充字符换成了 2 号编码的“$”符号。实验结果表明上述编码能够预测出填充符号，而且填充符号大多数情况和正确标签的位置一样，因此相当于减少了 50 个字符的损失值，大大的提升了神经网络学习效果。

(4) SGD 优化器能够达到最高的预测精度。进一步实验，结果表明模型的预测精度停

留在 85%左右。查阅了一些文献后[16]，发现有可能是 Adam 优化器的问题[17]，最后经过实验选择了每过 15epoch 改变一下学习率的 SGD 优化器（模型的学习率梯度 0.01、0.005、0.0025）。

(5) 增大损失值提高模型训练效率。分析得出目标序列大部分是由阿拉伯数字组成，所以可以将阿拉伯数字均匀分到全部的字符编码序列中，进而提高损失值大小方便梯度下降。结果表明在一定程度上确实增大了损失值，进而减弱了最终阶段学习率过小的问题。

(6) 分离模型任务。因为在得到分数习题解答算式后可以通过计算机直接得出精确答案，因此把模型的算式预测精度作为深度学习模型的精度。

综上，通过上面的改进，我们的式子的识别率最终达到了 96.8%。

第四章 Android 系统实现

4.1 Android 基础知识

Actvity 在 App 中的用途与网页在网站中用途类似，就是用来提供给用户一个 UI 界面和相关控制功能。一个 Activity 一般由一个或者多个布局文件和一个后台同名的控制类组成。设计人员将在布局文件中用 xml 代码设计 UI 界面和编写一些静态的功能，在同名控制类中用 Java 或者 Kotlin 进行编写逻辑控制代码。

为了防止因手机屏幕翻转导致临时数据丢失影响用户体验，可以为相应的 Activity 加上用于处理数据的具有与 Activity 不同生命周期的中间层 ViewModel。ViewModel 也减清 Activity 后台控制类的复杂度，使 Activity 专注于 UI 控制工作。

神经网络模型的预测结果需要较长但不准确的时间，因为神经网络返回预测结果与当时的设备繁忙程度、输入数据复杂度等多种因素有关。因此神经网络不能放入到主线程内，且当神经网络返回结果时，调用这个神经网络的 Activity 需要收到通知更新页面数据。这个机制本文是通过使用 Jetpack 中 LiveData 响应式编程组件完成的。

因为深度学习模型是利用 Python 语言编写的，所以本文使用第三方代码库 Chaquopy 实现在 App 端嵌入 Python 神经网络模型的操作。

图像文字识别如 2.3 节讨论的那样，目前本模型无法去除图像其他高宽压缩比导致的冗余，因此没有将其部署到 Android 平台上。

本文采用 Kotlin 语言在 Android Studio 平台上开发 App。

4.2 Android 系统实现

按照图 4-1 编写前端代码，其中输入框、结果显示的文本框、悬浮按钮分别命名为：nlp1EditText、nlp1Infer、fab。下面是后端控制代码实现步骤：

(1) 将 Chaquopy 的 Python 接口代码写入到继承 Application 的自定义类 MyMathApplication 中，该类用于 App 启动时的资源初始化，并且属性具有全局性，能被任何该工程的类调用。代码如下：

```
class MyMathApplication: Application() {
    companion object{
        @SuppressLint("StaticFieldLeak")
        lateinit var python: Python
    }

    override fun onCreate() {
        super.onCreate()
        initPython()
        python = Python.getInstance()
    }
    fun initPython(){
        if (! Python.isStarted()) {

Python.start(AndroidPlatform(MyMathApplication.context));
        }
    }
}
```

(2) 新建 Repository 单例类，利用 MyMathApplication 初始化的 Python 接口调用 transformer 文件中的 inference 函数，并传入 strq 问题字符串参数。因为需要 LiveData 支持线程上下文，所以我们调用 suspendCoroutine 开启协程，并将协程放在低并发高密集计算的默认线程 Dispatchers.Default 下。代码如下：

```
object Repository {

    fun transformerResult(strq: String) =
liveData(Dispatchers.Default){
        val result = try{
            Result.success(transformerRequest(strq))
        }catch (e:Exception){
            Result.failure<String>(e)
        }
        emit(result)
    }
    private suspend fun transformerRequest(strq:
String):String{
        return suspendCoroutine {
            try {

it.resume(MyMathApplication.python.getModule("transformer").
callAttr("inference",strq).toString())
            }catch (e:Exception){
                it.resumeWithException(e)
            }
        }
    }
}
```

(3) 在 ViewModel 中在封装一下 Repository 代码供 Activity 调用，并用 LiveData 的实时监测功能获取返回结果。代码如下：

```
class NLPViewModel : ViewModel() {
    private val transformerLiveData =
MutableLiveData<String>()
    val answer =
Transformations.switchMap(transformerLiveData){
            strq -> Repository.transformerResult(strq)
    }
    fun transformerResult(strq:String){
        transformerLiveData.value = strq
    }
}
```

(4) 在 Activity 中实例化 ViewModel 为 viewModel。代码如下：

```
val viewModel by lazy
{ ViewModelProviders.of(this).get(NLPViewModel::class.java)
}
```

(5) 在 Activity 中自定义收起输入键盘的函数。代码如下：

```
private fun hideSoftInput(fab: FloatingActionButton){
    val manager =
getSystemService(Context.INPUT_METHOD_SERVICE)
          as InputMethodManager
    manager.hideSoftInputFromWindow(
        fab.windowToken,
        InputMethodManager.HIDE_NOT_ALWAYS
    )
}
```

(6) 在 Activity 中的 onCreate 函数中写下如下代码，该代码是悬浮按钮 fab 点击事件响应函数，第一部分用于将输入框 nlp1EditText 的文字传输给 LiveData，第二部分代码用于将 LiveData 的结果通知给文本框 nlp1Infer 并让其显示返回结果，效果如图 4-1。

```
fab.setOnClickListener {

viewModel.transformerResult(nlp1EditText.text.toString())
    nlp1Infer.text = ""
    hideSoftInput(fab)
}
viewModel.answer.observe(this, androidx.lifecycle.Observer
{ answer ->
    nlp1Infer.text = answer.getOrNull()
})
```


20:07
MyMath
问题:
水果市场运来香蕉，苹果，梨三种水果，其中香蕉，苹果共1吨，梨占水果总重的1/3。水果市场一共运来多少吨水果?
解答:
1/(1-1/3)=1.50

图 4-1 App 效果图

第五章 研究总结

研究通过尝试多个 NLP 模型去解决小学分数类应用题，发现这一问题不同于普通的翻译问题，它需要较强的时序依赖性的 Encoder，但不希望具有较强时序依赖性的 Decoder。同时，虽然本文使用 Transformer 取得了很好的效果，但是通过 3.1.3 节的分析，后续可以将 Encoder 用 LSTM 等 RNN 进行替代，以取得更好的效果。因为程序中 Decoder 用条件概率去生成具有时序关系目标序列，影响了模型的效果，后续将设计出能够同时输出多个字符的网络结构，以使模型获得更优的结果。

关于图像文字识别部分，本次课题使用了 CRNN 去预测分辨率为 280*32 的图片，取得了很好的效果。但是用户的拍照获取的图像和 CRNN 需要的图像分辨率不同，还没找到合适的方法将其部署在 Android 系统上。后续将通过改变原数据集的高宽压缩比进而训练出具有较强适应性的 CRNN 模型；并且在 2.3 节讨论中也设想出了一种能够直接进行多行文本识别的网络模型。关于文字识别，将在以上两个方面继续开展工作。

参考文献

[1] Vinyals O, Le Q. A Neural Conversational Model [J]. arXiv preprint, 2015, arXiv: 1506.05869.

[2] Zaremba W, Sutskever I. Learning to Execute [J]. arXiv preprint, 2014, arXiv: 1410.4615.

[3] Vinyals O, Toshev A, Bengio S, et al. Show and Tell: A Neural Image Caption Generator [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2015: 3156-3164.

[4] Karpathy A, Fei-Fei L. Deep Visual-Semantic Alignments for Generating Image Descriptions [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2015: 3128-3137.

[5] Shi B, Xiang B, Cong Y. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(11): 2298-2304.

[6] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition [J]. arXiv preprint, 2014, arXiv: 1409.1556.

[7] Talathi SS, Vartak A. Improving performance of recurrent neural network with relu nonlinearity [J]. arXiv preprint, 2015, arXiv: 1511.03771.

[8] Graves A. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks [C]// Proceedings of the 23rd International Conference on Machine Learning. New York: Association for Computing Machinery, 2006: 369-376.

[9] Chung J, Gulcehre C, Cho KH, et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling [J]. arXiv preprint, 2014, arXiv: 1412.3555.

[10] Wu Y, Schuster M, Chen Z, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation [J]. arXiv preprint, 2016, arXiv: 1609.08144.

[11] Y Gal, Ghahramani Z. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks [J]. arXiv preprint, 2016, arXiv: 1512.05287.

[12] Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate [J]. arXiv preprint, 2014, arXiv: 1409.0473.

[13] Sutskever I, Vinyals O, Le QV. Sequence to Sequence Learning with Neural Networks [J]. arXiv preprint, 2014, arXiv: 1409.3215.

[14] Press O, Wolf L. Using the Output Embedding to Improve Language Models [J]. arXiv preprint, 2016, arXiv: 1608.05859.

[15] Vaswani A, Bengio S, Brevdo E, et al. Tensor2Tensor for Neural Machine Translation [J]. arXiv preprint, 2018, arXiv: 1803.07416.

[16] Duchi J, Hazan E, Singer Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization [M]. JMLR.org, 2011: 2121-2159.

[17] Kingma D, Ba J. Adam: A Method for Stochastic Optimization [J]. arXiv preprint, 2014, arXiv: 1412.6980.

致谢

回顾自己的求学生涯，我主要感谢三位改变我人生轨迹的恩师。

首先是我的初中的数学老师，任培祥老师用他的朴实的话语和不辞的辛劳地解答我的问题，将我从初中的迷茫中拉了回来，可以说是我的启蒙老师，为我的大学生涯埋下了伏笔。

然后感谢我的高中的物理老师，李双进老师非常喜欢我，我也不知道为啥，也许是我的学习天赋，也许是我的勤奋，但我更觉得是我的单纯，回顾我的前 20 年，我发现是我的单纯使我变得跟其他人不一样。

最后是我的未来的研究生导师侯凤贞老师，我和她认识的挺早的，源于她教我们的 C++，其实她给我最大的感觉就是心中有梦想，这也是我选择跟从她继续深造的原因，我觉得心中有一个小梦想才能支持一个人继续前进。感谢她不辞辛劳地指导我的论文写作。

另外，感谢一下邵瑜、李元晶两位同学对我的初稿提出了很多宝贵意见。

我感谢一下斋藤康毅和郭霖。他们两人的著作给了我很大的启发。

最后，我感谢一下我的父母，我大部分的价值观和思维方式都是源于他们的教导、信任和鼓励。