



# 本科毕业设计（论文）

学院(部)	计算机科学与技术学院		
题 目			
年 级		专 业	
班 级		学 号	
姓 名			
指导老师		职 称	
论文提交日期	2023 年 5 月		

## 苏州大学 本科毕业设计（论文）独创性声明

本人郑重声明：所提交的本科毕业设计（论文）是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本设计（论文）不含其他个人或集体已经发表或撰写过的研究成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

作者签名：\_\_\_\_\_ 日 期：\_\_\_\_\_

## 苏州大学 本科毕业设计（论文）使用授权声明

本人完全了解苏州大学关于收集、保存和使用本科毕业设计（论文）的规定，即：本科毕业设计（论文）的著作权以及文中研究成果的知识产权归属苏州大学。苏州大学有权向国家有关部门或第三方机构送交毕业设计（论文）的复印件和电子文档，允许毕业设计（论文）被查阅和借阅，可以采用影印、缩印或其他复制手段保存和汇编毕业设计（论文），可以将毕业设计（论文）的全部或部分内容编入有关数据库进行检索。

涉密设计（论文）☐

本设计（论文）属 \_\_\_\_\_ 在 \_\_\_\_\_ 年 \_\_\_\_\_ 月解密后适用本规定。

非涉密设计（论文）☒

论文作者签名： \_\_\_\_\_ 日 期： \_\_\_\_\_

导 师 签 名： \_\_\_\_\_ 日 期： \_\_\_\_\_



# XXXXXXXXXXXXX 的研究和实现

## 摘 要

视觉对话是将自然语言处理和计算机视觉相结合的跨学科研究的重要问题之一。视觉对话是根据图像内容展开的一系列对话，语义意图和对话主题随着对话的进行不断调整。视觉对话任务要求机器理解视觉和文本两种异构输入模态的信息，是一项颇具挑战性的深度学习任务，同时也具有良好的发展前景。

本文为了实现视觉对话模型架构，通过研究基于注意力机制的视觉对话模型，复现并改进了基于注意力的双视图网络模型，在相对短期的训练安排上取得了较好的结果。本文的具体工作如下。

第一，整理了从视觉对话任务首次提出到现在所诞生的性能较好的模型，总结了它们的算法类别并分析了它们的特点。鉴于基于注意力的模型在视觉和文本交互方面的优势，最终选定基于注意力的双视图网络模型作为基模型来研究。

第二，研究并分析了基于注意力的双视图网络模型的优缺点，主要分析了该模型所添加的三个模块在理解图像和文本语义方面的优势，同时分析了该模型在表达序列顺序和损失函数等方面的局限性。在理解基模型原理的基础上，复现并部署在了云端 GPU 平台上。

第三，改进了基模型所用的优化器和部分激活函数。此外，根据所分析的基模型的局限性，阅读了大量文献寻找改进方法，最终选定了绝对位置编码和大间隔交叉熵损失函数，在深入理解原理及其公式推导的基础上，参考了它们在其他任务上的用例，最终完成了代码的编写和调试。

第四，添加了模型对其他数据集的读取方法，通过实验测试了基模型及改进后模型在两种数据集上的结果，并对结果进行分析，总结其优势和不足。

**关键词** 视觉对话；深度学习；注意力机制；语义交互；损失函数

# Research and Implementation of XXXXXXXXXX

## Abstract

Visual dialogue is one of the important issues in interdisciplinary research that combines natural language processing and computer vision. A visual dialogue is a series of dialogues based on the content of the images, and the semantic intent and the topic of the dialogue are continuously adjusted as the dialogue progresses. The visual dialogue task requires machines to understand the information of two heterogeneous input modalities, visual and text. It is a challenging deep learning task, and it also has good development prospects.

In order to realize the visual dialogue model architecture, this paper reproduces and improves the attention-based dual-view network model by studying the visual dialogue model based on the attention mechanism, and achieves good results in a relatively short-term training arrangement. The specific work of this paper is as follows.

First, we sort out the models with better performance from the visual dialogue task that was first proposed to the present, summarize their algorithm categories and analyze their characteristics. In view of the advantages of attention-based models in visual and textual interaction, the attention-based dual-view network model is finally selected as the base model for research.

Second, research and analyze the advantages and disadvantages of the attention-based dual-view network model, mainly analyze the advantages of the three modules added to the model in understanding the semantics of images and texts, and analyze the model's expression sequence order and text. Limitations of loss functions, etc. On the basis of understanding the principle of the base model, it is reproduced and deployed on the cloud GPU platform.

Third, the optimizer and partial activation function used by the base model are

improved. In addition, according to the limitations of the analyzed base model, I read a lot of literature to find improvement methods, and finally selected the absolute position coding and large interval cross entropy loss function. Use cases on other tasks, and finally complete the writing and debugging of the code.

Fourth, the method of reading other data sets by the model is added, and the results of the base model and the improved model on the two data sets are tested through experiments, and the results are analyzed to summarize their advantages and disadvantages.

**Keywords** visual dialog; deep learning; attention mechanism; semantic interaction; loss function

## 目 录

前言 .....	1
第 1 章 绪论 .....	1
1.1 研究背景和意义 .....	1
1.2 研究现状 .....	2
1.3 本文主要工作及创新 .....	3
1.4 本文的组织结构 .....	4
第 2 章 基础知识概述 .....	5
2.1 视觉对话基础概念 .....	5
2.1.1 视觉对话定义 .....	5
2.1.2 数据集 .....	5
2.1.3 评价指标 .....	6
2.2 注意力机制简介 .....	6
2.3 视觉对话模型概述 .....	7
2.3.1 基准模型介绍 .....	7
2.3.2 基于预训练的模型 .....	9
2.3.3 基于视觉共指消解的模型 .....	9
2.3.4 基于图结构的方法 .....	9
2.3.5 基于注意力机制的模型 .....	10
2.3.6 视觉对话模型总结 .....	11
2.4 本章小结 .....	12
第 3 章 基于注意力机制的双视图网络模型及其改进 .....	13
3.1 基模型研究与改进目的 .....	13
3.2 双视图网络基模型 .....	13
3.2.1 基础介绍 .....	13
3.2.2 长短期记忆网络简介 .....	14
3.2.3 模型整体框架 .....	14



3.2.4 模块分析 .....	15
3.3 模型改进 .....	17
3.3.1 位置编码的添加 .....	17
3.3.2 损失函数及其改进 .....	18
3.3.3 优化器和其他细节修改 .....	21
3.4 本章小结 .....	21
第 4 章 模型实现及结果分析 .....	23
4.1 模型实现过程 .....	23
4.1.1 实验环境 .....	23
4.1.2 实验流程 .....	23
4.2 数据集扩展 .....	25
4.3 模型改进实现 .....	28
4.3.1 优化器和激活函数的消融研究 .....	28
4.3.2 位置编码改进 .....	29
4.3.3 损失函数改进 .....	30
4.3.4 实验结果分析与可视化 .....	32
4.4 本章小结 .....	34
第 5 章 总结 .....	36
5.1 本文总结 .....	36
5.2 工作展望 .....	36
参考文献 .....	38
致谢 .....	41



## 前言

随着机器学习和深度学习的发展，计算机感知和认知能力的不断提高，同时处理视觉和语言两种甚至更多模态的深度学习任务得到了广泛关注，如视觉对话、图像描述生成和视觉问答（Visual Question Answering, VQA）等。

视觉对话任务在 2017 年由 Das 等人<sup>[1]</sup>首次提出，不同于只需要回答一次图像问题的视觉问答任务，视觉对话要求计算机回答一系列关于先前对话和给定图像的问题，因此视觉对话也可以看作多轮的视觉问答。在视觉对话任务中，计算机除了依据图像理解问题之外，还需结合历史对话信息进行语义推理，因为随着对话的进行，与问题主题相关的视觉内容也会发生变化，从而学习更全面和语义丰富的图像表示。

从视觉对话任务首次提出到目前为止，已经出现了许多视觉对话模型算法，基础的方法主要是简单地对特征联合嵌入，这种模型的缺点是没有很好的理解问题的语义，确定对话的主题。而更广泛的研究是基于注意力的方法，这种方法普遍利用注意力机制对视觉和文本语义进行一个细粒度的交互，对于语义和主题的理解相对具有优势。

本文在分析了现有的各种主要模型的特点基础上，以基于注意力的双视图网络作为视觉对话框架，深入理解模型的工作原理，对其进行复现，并根据在不同数据集上的实验测试效果，尝试在优化器和激活函数、词向量编码的优化、损失函数等几个方面进行了改进，并在扩展数据集上进行了模型改进前后的测试评估，最终完成了视觉对话模型架构。具体完成的工作有：

（1）分析了主流的各种视觉对话模型，对每一类挑选代表性的模型对比总结它们的特点，并选定了本文研究的基模型。

（2）分析了基于注意力的双视图网络模型的原理，框架模块及优缺点，基于该模型提供的模板代码对其进行复现并部署在云端 GPU 平台。

（3）根据基模型的局限性对其进行改进，包括对词向量进行位置编码，改进损失函数以及对优化器和部分激活函数进行修改。

（4）添加了模型对 VisDial v0.9 数据集<sup>[1]</sup>的读取方法，对基模型和改进后的模型在 VisDial v1.0 数据集<sup>[1]</sup>和 VisDial v0.9 数据集上进行测试评估，并为模型添加了可视

化的训练过程和评估结果。

## 第 1 章 绪论

本章首先陈述了视觉对话模型的研究背景和应用意义，然后简单介绍了注意力机制的基本原理和其优势，并简述了本文完成的主要工作和贡献。最后，本章还给出了本文的整体组织结构。

### 1.1 研究背景和意义

随着计算机视觉和人工智能研究的不断发展，该领域取得了前所未有的进步。从普通的人工智能任务，如目标识别<sup>[2]</sup>，图像分类<sup>[3]</sup>，语义分割等，到更为复杂的 AI 任务，例如围棋学习<sup>[4]</sup>，回答阅读理解问题，回答图像或视频的问题等。人工智能系统逐渐开始学习通过自然语言来和人类对视觉内容进行对话的能力，尽管我们距离实现一个能真正“看”和“说”的机器的目标还很遥远。近年来提出的视觉对话任务不仅要求计算机理解图像，还要结合对图像的理解进行对话。它需要回答有关图像的一系列问题，对图像、问题和对话历史进行推理。它的任务描述如下：给定图像，对话历史由若干问答对组成，机器需要回答一个新的问题。例如，在图 1.1 中，对话历史为（Q1：“房间里有几只猫？”，A1：“2 只”，Q2：“右边的猫是白色的吗”，A2：“是”），问题是（Q3：“它睡着了吗？”），机器需要结合历史来确定“它”指代的对象，并结合图像给出答案（A3：“否”）。可以发现，视觉对话任务是对视觉问答的推广，这种带有历史信息的问答更接近于现实世界的对话。



图 1.1 视觉对话样例图像

视觉对话的研究具有重要意义，在不同的领域中有广泛的应用前景。例如，可以帮助视觉障碍用户理解他们周围的环境或是社交媒体的内容。可以帮助游戏分析师根据大量的赛事数据做出合理的决策。可以应用于机器人的搜索或救援等任务，救援人员处于“情境盲区”，可以通过自然语言来和机器人展开对话，机器人根据收到的自然语言信息，在救援场所结合视觉信息执行命令并加以反馈，再根据之后的命令开展行动。通过这种方式开展救援，救援人员既不用亲自去涉险，又可了解现场具体情形。

由于视觉对话任务的复杂性以及数据集的特殊性，基于以往先进的视觉问答算法的简单改编模型并没有取得很好的结果，反而简单地对图片，问题，历史进行融合编码嵌入，再判别式解码后取得了更好的结果。但是，视觉对话不仅要求处理异构模态地问题，还要隐式地理解视觉和上下文文本间的语义关系，仅仅是简单的进行特征融合，而不考虑异构输入间的语义交互理解，这种模型的表现也不能完全令人满意。鉴于文本上下文和视觉之间潜在的依赖性，诞生了许多表现优秀的视觉对话模型，包括基于注意力机制的模型，基于图结构的模型和基于预训练的模型等。其中，在注意力模型下，视觉和文本可以进行细粒度的交互，这类模型达到的效果总体不错。

本文主要研究的基于注意力机制的双视图模型则是在学习以往模型的基础上，通过设置两个互补的模块来加强对语义的理解，尽管训练所需要耗费的时间比较久，但它达到的效果整体上令人满意。

## 1.2 研究现状

为了解决视觉对话任务，出现了许多从不同角度去提高模型性能的方法。目前，大部分的模型都把视觉对话看成带有历史的视觉问答，针对视觉对话的两大难点，异构输入的处理和视觉共指消解，主流的方法分别通过提取单模态特征后再进行交互关联和从对话历史中明确指代词的具体对象来确定语义的方法处理这两大问题。

但是，简单的把视觉对话看出带有历史的 VQA 有时候不能取得很好的结果，因为机器在进行关于图像的对话时，随着对话的进行，图像和历史问答之间的交互作用也会逐渐变化，也就是语义主题会随着对话的进行发生改变。由于文本和视觉上下文

之间具有潜在的依赖性，结合图像和问题对上下文进行推理就至关重要。因此，如何有效的理解语义和主题也成为了成为该任务的一大难点。

针对这一问题，目前也已出现了不少的处理对策。比如切断历史对答案的直接影响，通过构建一个复杂的因果图来模拟随着对话的进行，主题语义和回答偏好的变化。本文研究的基于注意力的双视图网络模型则是通过构建两个模块来从对话历史中捕获语义主题信息。

### 1.3 本文主要工作及创新

本文以视觉对话任务为问题导向，主要以基于注意力机制的双视图模型为研究对象，分析了目前流行的几种视觉对话模型，对比检测了它们的优缺点。在深入分析了基于注意力机制的双视图网络模型的基础上，仔细研究了它所提供的模板代码，在 VisDial v1.0 数据集上复现了其实验结果，并将其应用到旧数据集 VisDial v0.9 版本上测试其效果。在模型方面，给词向量添加了位置编码来记录单词在句子中的位置信息，使其能更好地表示句子的顺序信息。对解码器所用的损失函数做了改进，该损失函数使得学习起来更加困难缓慢，但在训练的中前期评估时取得了更好的结果。此外，本文优化了其源码，对其优化器和激活函数做了改进，并通过消融研究来测试其作用。最后，为该模型添加了可视化的训练过程和评估结果。本文的主要工作及贡献如下：

（1）分析了主流的各种视觉对话模型，对每一类挑选代表性的模型对比总结它们的特点，并选定了本文研究的基模型。

（2）分析了基于注意力的双视图网络模型的原理，框架模块及优缺点，基于该模型提供的模板代码对其进行复现并部署在云端 GPU 平台。

（3）根据基模型的局限性对其进行改进，包括对词向量进行位置编码，改进损失函数以及对优化器和部分激活函数进行修改。

（4）添加了模型对 VisDial v0.9 数据集的读取方法，对基模型和改进后的模型在 VisDial v1.0 数据集和 VisDial v0.9 数据集上进行测试评估，并为模型添加了可视化的训练过程和评估结果。

## 1.4 本文的组织结构

本文共分为五章，各章内容如下：

第一章绪论介绍了本文研究对象的背景和意义，简单介绍了视觉对话的研究现状和难点，然后介绍了本文完成的主要工作和贡献，最后介绍了本文的组织结构。

第二章为相关知识概述，首先阐述了视觉对话的基础概念，包括定义，数据集和评估指标等，然后简单介绍了注意力机制，最后总结了主流模型分类并分析它们的特点。

第三章首先介绍了基于注意力机制的双视图网络模型的研究价值和基本框架，随后深入分析了该模型所包含的模块的原理和实现，最后提出了对它的改进和实现。

第四章描述了基于注意力机制的双视图网络模型在云端 GPU 上对不同数据集的测试结果，并描述了改进部分的具体代码实现，实验测试了其训练后的效果以及优缺点，并对部分结果进行可视化展示。

第五章对全文做了一个总结，提出了一些关于该课题的未来工作，以及对视觉对话任务的展望。



## 第 2 章 基础知识概述

本章首先介绍了视觉对话的基础概念，然后简单介绍了注意力机制的原理，最后分析了主流的模型分类并总结其特点。

### 2.1 视觉对话基础概念

#### 2.1.1 视觉对话定义

该任务的输入集由图像  $I$ 、当前问题  $Q_t$ ，对话历史集  $H_t = \{C, (Q_1, A_1^{gt}), \dots, (Q_{t-1}, A_{t-1}^{gt})\}$ ，其中包含了图像标题  $C$  和  $t-1$  个连续的问答对，以及一组候选答案的集合  $A_t = \{A_t^1, A_t^2, \dots, A_t^{100}\}$ 。要求模型通过判别或生成正确的答案来回答当前问题。

#### 2.1.2 数据集

视觉对话的常用数据集为 VisDial v1.0。它基于 MSCOCO 数据集的标题和图像进行收集，其中，图片对应的对话由两人通过提问和回答的方式进行收集，对于数据集中的每张图片而言，提问者只能看到标题和对话历史，而回答者可以看到标题、历史和图像。每张图片的对话由 10 轮问答组成。任何一个当前问题的答案都不包括在对话历史中。VisDial v1.0 分为训练集、验证集和测试集 3 个子集。其中，训练集包括 123287 个对话，验证集包括 2064 个对话，测试集包含 8000 个对话。

此外，VisDial 的旧版本 v0.9 也常用于视觉对话，由于它包含的对话和图片较少，数据集的质量也不如 v1.0 版本，很多模型已经弃用了它，但是它还是可以用来评估模型的质量，并且它数据集较小，模型训练起来更加方便。它共包含 1.4M 的问答对。

### 2.1.3 评价指标

视觉对话的评估方法借鉴了检索的评估方法。每个问题的候选答案有 100 个，模型需要返回这 100 个候选答案的排序。模型的评估指标有两类，对于候选答案中只有一个正确答案的答案注释，评估指标包括标准答案在前  $k$  个答案中响应的比例 (Recall@K)，标准答案的平均排序 (Mean)（越低越好），平均倒数排序 (MRR)。其中 MRR 是对所有正确答案在模型结果中的排序去倒数后在求取平均值，结果越高越好，该指标侧重于人类的真实答案，缺点是会忽略很多其他可能正确的答案。第二类评估指标基于密集的答案注释，候选答案中有数个正确答案，正确程度通过值域为 (0, 1) 的相关性比率来表示，评估指标为归一化折现累计增益 (NDCG)，由于该指标依赖第三人标记所有的正确答案，所以对于不确定性的问题，该指标效果更佳。

## 2.2 注意力机制简介

在序列转导模型上，编码过程会生成一个中间表示来存储原始序列的语义信息，由于这个中间向量的长度是确定的，当模型的表达能力很强时，模型存储的信息量也非常巨大，定长的中间向量没法存储全部的语义信息，导致信息过载的问题。而且，现实中自然语言句子中的不同语素的重要性不是相同的，但是这种编码方法却对序列中不同部分保持相同的关注程度，导致模型的理解能力受到很大限制。因此就需要引入注意力机制，更加关注原始输入信息中更为重要的语义信息，降低对不重要信息的注意力。对于不相关的信息直接舍弃。通过把注意力集中在重要的信息上，提高模型的精准度。

注意力机制的计算方法很多，这里介绍一种常用的方法。解码层的初始状态  $x'_i$  是编码层的最后一个状态，与普通的 seq2seq 模型不同，这里需要保留编码层的所有隐藏状态  $h_i$ ，然后计算  $x'_i$  与编码层每一个隐藏状态的相关性：

$$\alpha_{ii} = \text{align}(x'_i, h_i) \quad (2.1)$$

其中函数 *align* 用来计算解码层  $x'_i$  和编码层第  $i$  个神经元之间的相关性。具体来说，可以通过使用两个参数矩阵对它们做线性变换，得到两个向量，再计算两个向量的内积  $e_{ii}$ ，对  $x'_i$  与编码层所有的神经元做上述步骤，就得到了一个由  $e_{ii}$  组成的向量  $e_i$ ，用 *softmax* 函数对其进行归一化处理，最后对编码层的每一个隐藏状态使用  $e_i$  来做一个加权线性组合，即得到了所需的上下文状态。

本文的研究重点基于注意力的双视图网络模型是一种加入注意力机制的 seq2seq 模型，这种方法的新颖之处在于，它着重于确定问题的语义意图，采用两个模块来捕获对话历史中与主题相关的语义信息。这两个模块分别在历史中寻找与问题相关的信息和对问题和历史进行主题的聚合，这种方法的提出对视觉对话任务中语义主题的理解具有重要的意义。

## 2.3 视觉对话模型概述

时至今日，视觉对话诞生了许多效果良好的模型，主流模型包括基于预训练的模型，基于视觉指代消解的模型，基于图结构的模型和基于注意力机制的模型等。本节首先介绍视觉对话的基准模型，然后分类介绍主流模型的特点并挑选代表模型分析，最后对各类模型的特点作了一个总结

### 2.3.1 基准模型介绍

视觉对话的基准模型由 Das 等人<sup>[1]</sup>提出，它是一个编码器-解码器的模型，该模型共提出了 3 种基础的可选的编码器，包括后期融合(Late Fusion, LF)，层次递归编码(Hierarchical Recurrent Encoder, HRE)和记忆网络(Memory Network, MN)。其中，后期融合将图像、历史和问题嵌入到向量空间中，再把所提取的这些特征拼接起来作为联合特征。层次递归编码则是先将图像和问题通过长短期记忆网络(Long Short-Term Memory, LSTM)联合嵌入，再嵌入之前每一轮的历史，最后，由一个添加注意力机

制 RNN 循环块选择与当前问题相关的历史。如图 2.1 所示。

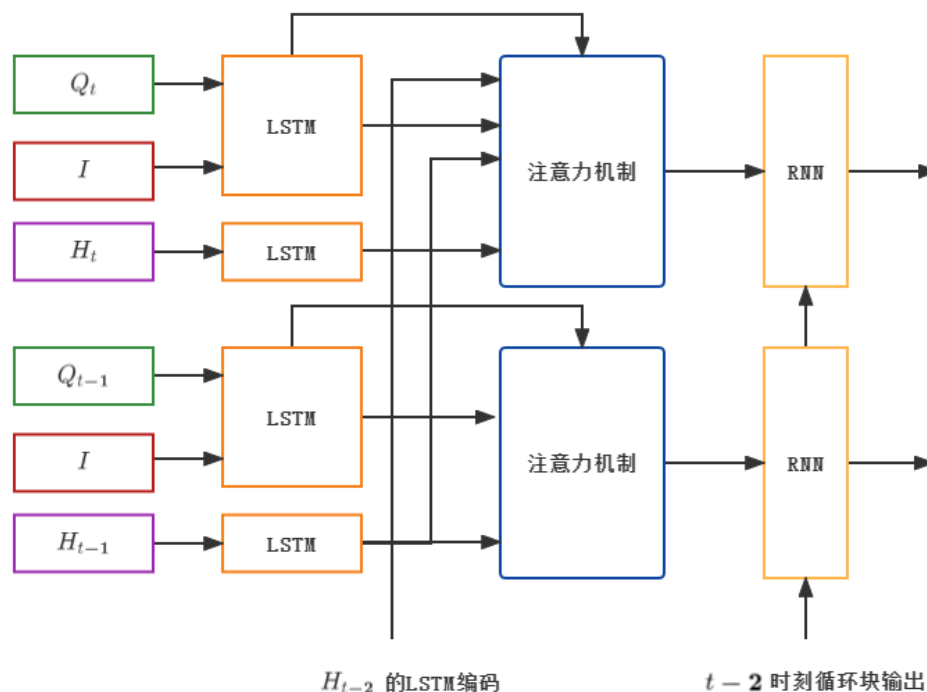


图 2.1 HRE 模型

记忆网络为对话历史设置一个记忆缓存,用 LSTM 分别对问题和历史进行编码,再计算问题向量和每个历史向量的内积,喂入 *softmax* 来获得注意力概率计算得到最终的对话历史向量,再将它经过一个全连接层并添加到问题向量中,得到问答对编码,储存在记忆缓存中,遇到新的问题时,通过查找记忆缓存来获取历史信息。

该模型提出了两种解码器,生成式模型(Generative LSTM)和判别式模型(Discriminative softmax)<sup>[1]</sup>。其中判别式解码器使用 LSTM 来对候选答案进行编码,并与编码层的输出作点积,输入 *softmax* 来计算选项的后验概率,通过最大化正确选项的对数似然率来训练。评估时,判别式模型输出候选答案上的概率分布,选择概率大的作为输出答案。生成式模型的答案并不来源于候选答案的集合,而是由模型直接生成自然语言句子。

### 2.3.2 基于预训练的模型

由于各种视觉语言任务不断地应用预训练的方法，并且取得了较好的效果，视觉对话模型也开始采用预训练的方法，主要包括 VD-BERT<sup>[19]</sup>和 VisDial-BERT<sup>[20]</sup>等。它们采用 transformer 模型作为基本架构。其中 VisDial-BERT 在应用到视觉对话之前，先在 VQA 和 Conceptual 数据集上进行预训练，再在 VisDial 上进行微调，对于带有密集注释的 VisDial v1.0 数据集，在 NDCG 指标上评价时取得了很好的结果。

### 2.3.3 基于视觉共指消解的模型

由于在自然语言中，人们不可避免的采用指代词来指称同一实体，导致语句指代不明，语义表达并不明确，当机器试图理解句子语义时，容易产生障碍。因此，将代表同一对象的代词划分到同一个集合，也即指代消解，就至关重要。在视觉对话任务中，回答者需要消解指代词，并将其与图像中的目标相联系，从而准确的回答问题。

主流的视觉指代消解的模型包括神经模块网络模型(CorefNMN)<sup>[5]</sup>、自适应的视觉记忆网络(AVMN)<sup>[6]</sup>、解决视觉指代的注意力记忆模型(Attention Memory, AMEM)<sup>[7]</sup>和循环对偶注意力网络(Recurrent Dual Attention Network, ReDAN)<sup>[8]</sup>等。其中，CorefNMN 通过引入涉及和排除两个模块，将指代词与图像中的具体对象关联起来，实现了更为精细的单词级别的视觉共指消解。AVMN 直接将视觉信息存储于外部记忆库，整合文本和视觉定位过程，进而有效缓解了在这两个过程中所产生的误差。AMEM 提出了一种新颖的注意力机制，配备了一个注意力记忆存储器存储视觉注意力，模型通过检索以往的注意力来计算当前问题最相关的目标，该模型的缺点是只能实现句子级别的视觉指代消解。

### 2.3.4 基于图结构的方法

随着深度学习对非欧氏数据的逐渐重视，图结构和图神经网络也得到了快速的发展。由于传统的注意力机制难以捕捉两个异构模态间对于单个对象的语义关系，而图网络在处理结构化数据方面表现出了较强的优势，使用图网络来提取多模态特征的方法越来越多的被采纳。基于图结构的模型包括图神经网络(Graph Neural Network, GNN)<sup>[9]</sup>、因子图注意力模型(Factor Graph Attention, FGA)<sup>[10]</sup>、自适应的对偶视觉对话模型(DualVD)<sup>[11]</sup>、因果图结构模型<sup>[12]</sup>和知识桥图网络<sup>[13]</sup>等。其中，GNN 将多轮历史对话采用图结构来表示，通过图结构来计算当前问答对的表示，图结构的节点状态更新则通过消息传递机制来进行。FGA 将图结构和注意力机制相结合，使用基于图的公式来表示注意力框架，通过因子对节点的两两交互进行建模，实现了 MRR 高达 1.1% 的改进。DualVD 借鉴了认知学中的双向编码理论，认为人类的认知包含对视觉所见和相关文本的交互，提出通过语义和视觉两个模块分别描述抽象的语义信息和视觉对象关系，提取物体间的关系特征。KBGN 以图结构方法为基础，也采用模块化的流程设计，对视觉和文本间潜在的语义交互联系进行细粒度地建模。因果图结构则是认为视觉对话并非只是带有历史的 VQA，它从因果推理的角度添加新的参数（提问者真实偏好，答案标注人根据对话历史，确定答案偏好）来创建一个描述现实世界对话场景的因果图，这种方法切断了对话历史和正确答案间的直接因果联系。

### 2.3.5 基于注意力机制的模型

在视觉对话任务中，计算机需要确定问题的语义意图，同时需要在异构模态输入中对齐与问题相关的文本和视觉内容。这种视觉和语言的细粒度交互可以通过注意力机制来实现。

代表性的基于注意力机制的模型包括递归视觉注意力模型(Recursive Visual Attention, RvA)<sup>[14]</sup>、记忆网络、基于注意力的双视图网络模型(Multi-View Attention Network)、基于历史的图像注意力编码模型(History-Conditioned Image Attentive Encoder, HCIAE)<sup>[15]</sup>和联合注意力模型(Co-Attention, CoAtt)<sup>[16]</sup>等。其中，MN 和 HCIAE 提取图像或问题的特征，结合对话历史进行注意力加权计算。RvA 递归地查看与主题

相关的对话历史并细化视觉注意力，直到对视觉基础感到“自信”，或已回溯到对话历史的开头(如图 2.2 所示)。基于注意力的双视图网络模型利用两个互补模块(主题聚合和上下文匹配)来捕获与问题相关的对话历史中和历史中的主题信息，并通过模态对齐来构建多模态表示。CoAtt 将强化学习和生成对抗网络相结合，联合训练两个子模块：一个基于图像内容和历史对话生成句子地序列生成模型，和一个利用记忆模块区分人类对话和机器对话地鉴别器。

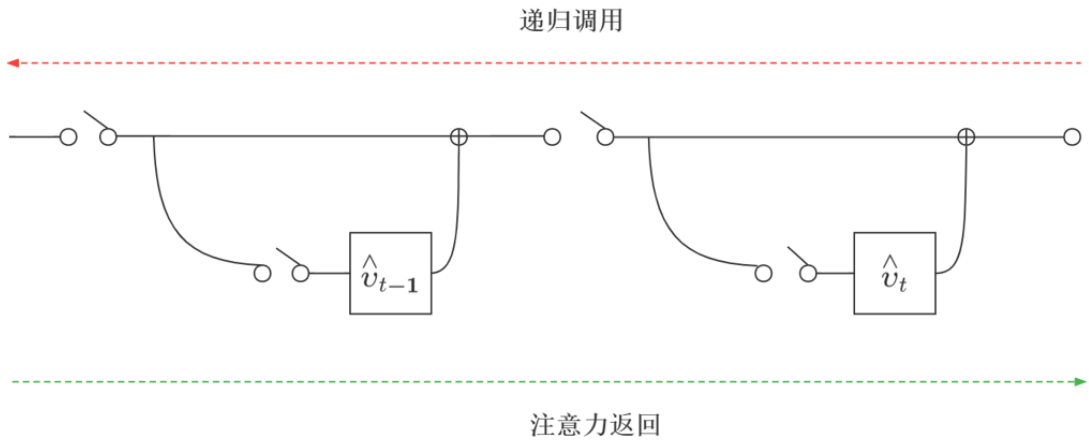


图 2.2 RvA 递归注意力模型

### 2.3.6 视觉对话模型总结

通过对上述主流模型类别的介绍和典型代表的分析，可以初步归纳出各类模型的优缺点。

基准模型采取了简单的特征融合方法，这类方法仅将多源特征联合嵌入，优点是简单便于实现，缺点是没有考虑图像、历史和问题间的语义交互。

基于预训练的模型对模型的分支模块在大规模数据集上进行预训练，再迁移到视觉对话任务上取得较好的性能，缺点是如果独立地对图像和文本进行预训练，图像和历史的语义就不能充分交互。

基于视觉共指消解的方法很好的解决了视觉共指问题，明确指代词的具体对象，

缺点是由于对话中的对象是有重要性区分的, 过分关注代词指代对象可能会忽略对话的重点交流对象。

基于图结构的方法使用图结构来处理异构模态的输入, 可以很好地捕捉图像和对话间潜在的语义依赖, 避免对话历史过分影响答案。缺点是如果结点间交互信息过多可能淡化结点本身的特征。

基于注意力机制的模型对历史和图像进行注意力分数加权计算, 优点是可以很好地进行语义交互, 缺点是对话历史直接参与编码层的输出计算, 可能会扩大历史对答案的影响。

## 2.4 本章小结

本章介绍了视觉对话的相关概念, 包括任务定义, 常用数据集和评估指标等概念, 然后介绍了注意力机制的原理, 最后总结了主流的视觉对话模型, 挑选了部分代表性模型分析它们的特点。



## 第3章 基于注意力机制的双视图网络模型及其改进

本章首先介绍视觉对话任务最重要的需解决的问题，阐述研究基于注意力机制的双视图网络模型的原因与改进目的，然后重点介绍基于注意力机制的双视图网络模型的框架模块，最后提出了对该基模型的改进方法。

### 3.1 基模型研究与改进目的

语义交互理解是视觉对话的一个重要研究方向，对语义的理解程度会直接影响模型的性能，因此，如何在多模态输入上学习其语义交互关系，并根据对话的进行动态地捕捉这种语义关系的变化，成为视觉对话任务亟需解决的问题。本章研究的基于注意力机制的模型通过构建三个模块，学习语义的交互关系，理解对话主题并执行异构模态输入的对齐，是解决语义交互理解问题的一个很好的模型范例。

尽管基于注意力的双视图网络模型已经充分进行了视觉，问题和对话历史间的语义交互理解，该模型还是存在着很多改进空间。本文希望在该模型源代码所用的优化器和激活函数作改进的基础上，寻找在序列编码时加强序列顺序信息表示的方法，另外希望采用交叉熵损失函数的变种，来加强类间的可分性。

### 3.2 双视图网络基模型

#### 3.2.1 基础介绍

基于注意力的双视图网络模型是由 Sungjin Park<sup>[17]</sup>等人在 2020 年提出的模型。该模型侧重于确定问题的语义意图，并对齐与问题相关的文本和视觉内容。模型采用 VisDial v1.0 作为训练和评估的数据集，对于图像编码，它采用预训练的 Faster-RCNN 图像特征<sup>[18]</sup>。对两种不同的文本输入：对话历史和当前问题，它首先使用 Glove 预训练的词向量<sup>[19]</sup>进行嵌入，再把词嵌入输入到双向长短期记忆网络（BiLSTM）中进行

编码,隐藏层大小为 512。对于候选答案,由于它们的序列长度较短,故使用单向 LSTM 来表示。该模型共进行 8 轮迭代, `batch_size` 设置为 32。

### 3.2.2 长短期记忆网络简介

长短期记忆网络,即 LSTM,是一种特殊的 RNN,它增加了一个单元状态  $c$  来保存长期状态,它的结构包括三个门,输入门决定  $t$  时刻网络的输入  $x_t$  哪些将被更新到  $c_t$ ,遗忘门决定  $t-1$  时刻的单元状态  $c_{t-1}$  保留多少到  $c_t$ ,输出门决定单元状态  $c_t$  的哪些部分输出到  $t$  时刻的输出值  $h_t$ ,相比于短期输入非常敏感的 RNN, LSTM 改善了长期依赖的问题,且由于它的导数不是乘积,解决了梯度消失的问题,缺点是每个单元里面都有四个全连接层,计算量很大,训练耗时长。

### 3.2.3 模型整体框架

基于注意力的双视图网络模型采用经典的 Encoder-Decoder 框架。不同的是,为了更好的捕获问题的语义,明确主题信息,它的编码层由三个模块组成。首先,上下文匹配模块连接问题和对话历史的 BiLSTM 的最后隐藏状态表示,通过注意力机制和门控机制有效地获取与问题相关的对话历史信息。其次,主题聚合模块关注历史中与主题相关的单词,从对话历史和问题中捕获主题引导的线索。同样,它也使用注意力机制和门控函数来传播与语义意图相关的文本信息。最后,模态对齐模块包含两个对齐过程,首先根据主题线索学习异构输入之间的对齐,然后将它们与上下文信息对齐,从而达到视觉和文本间语义对齐的效果。

该模型的解码层与主流视觉对话模型的解码层类似,它的判别式解码层首先用 LSTM 的最后隐藏状态来表示候选答案,再计算候选答案与编码层输出的点积,喂入 *softmax* 归一化处理获得概率,由于该任务是个多分类问题,所以使用交叉熵损失函

数来训练，以求达到较好的效果。

该模型在 VisDial v1.0 数据集上训练 5 轮后达到的指标如表 3.1 所示。

表 3.1 原模型在 VisDial 数据集训练 5 轮后的评估结果

指标	结果
r@1	0.5049903392791748
r@5	0.8076550364494324
r@10	0.8981589078903198
mean	4.165552139282227
mrr	0.6407473683357239
ndcg	0.5726126432418823

### 3.2.4 模块分析

这里分析基于注意力的双视图网络模型的模块具体组成及其作用，总体结构如图 3.1 所示。

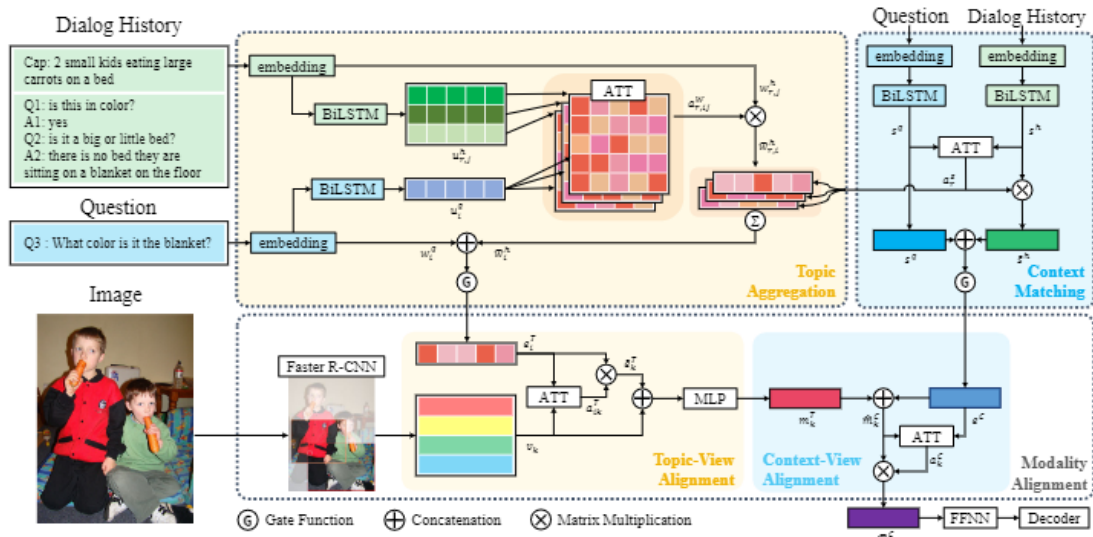


图 3.1 基于注意力的双视图网络模型结构

上下文匹配模块：该模型的语义历史表示是通过连接问题和对话历史的 BiLSTM

的最后隐藏状态来构建的。使用 BiLSTM 的好处是除了上文的信息，还可以携带下文的语义信息。然后，应用注意力机制来关注与问题相关的对话历史。将问题表示和对话历史表示分别进行非线性变换，再逐元素相乘后通过一个全连接层，归一化处理后得到了历史相对于问题的注意力，对历史表示和注意力加权求和后得到历史语义表示。

此外，该模块加入了一个门控机制来过滤掉与问题无关的历史，门控机制的公式如下：

$$gate = \sigma(Wx + b) \quad (3.1)$$

$$e = gate \circ x \quad (3.2)$$

在本模块中，先将问题表示和历史的语义表示拼接起来，经过一个以 *Sigmoid* 为激活函数的全连接层后，与自身逐元素相乘得到了最终的语义历史表示，这种方法的好处有选择地结合历史表示中与问题相关的历史。

主题聚合模块：问题的主题通常以单个单词或短语来表达，因此主题聚合模块，通过利用初始的词嵌入向量来表示主题含义。同样，它也应用了注意力机制，将单词级别的问题和历史向量非线性变换后作点积，再使用 *softmax* 函数进行归一化，最后对历史表示加权求和得到了问题引导的历史特征。将该特征与上下文匹配模块中求得的注意力分数加权求和，得到了对话历史的主题表示。

与之前一样，先将问题向量和对话历史的主题表示拼接，再应用一个门控函数来过滤掉无关信息，该模块优点是可以选择性地关注对话历史中与问题主题相关的单词。

模态对齐模块：得到上下文匹配模块的输出语义历史表示和主题聚合模块的输出主题历史表示后，模态对齐模块将它们和预训练的图像特征对齐。首先进行的是主题和图像之间的异构模态对齐。通过点积注意力来得到与图像相关的主题表示，也即对图像特征和主题表示进行非线性变换后再作点积，然后进行 *softmax* 归一化处理，最后将主题表示与注意力分数加权求和。再通过多层感知机来融合图像特征和图像主题表示，这样就完成了图像和主题对齐，得到了一个融合特征，即主题视图的对齐表示。

随后进行上下文语义历史的对齐步骤，将主题视图的对齐表示与语义历史表示先进行非线性变换，再逐元素相乘，归一化处理后得到注意力分数，再与主题视图的对

齐表示加权求和，得到视图的对齐表示，这种多重对齐过程使模型能够以互补的观点理解语义意图，并有效的对齐异构模态。最后视图的对齐表示输入到用 *Relu* 作为激活函数的感知机中，得到最终视图对齐表示，也即编码层的输出，它被喂入解码层中。

### 3.3 模型改进

#### 3.3.1 位置编码的添加

基于注意力机制的双视图网络模型能取得优秀的结果主要依赖于编码层三个模块对语义主题的充分理解，以及 LSTM 在长序列训练上的优秀表现，但是，LSTM 能记忆的信息时间是有限的，大约在 100 秒左右，由于视觉对话数据集极为庞大，所以问题和历史的顺序信息依然可能被遗忘。针对这种情况，我仔细阅读了谷歌提出的 Transformer 模型<sup>[20]</sup>，由于 Transformer 模型不包含递归和卷积，它的注意力机制没有包含位置信息，也即词在句子中的不同位置对该模型来说是没有区别的。为了利用序列的顺序，该模型在编码器和解码器的输入编码中添加“位置编码(Positional Encoding, PE)”。位置编码和输入嵌入维度相同，因此可以相加。尽管该模型采用位置编码的目的是弥补没有递归或卷积的缺陷，但我希望将位置编码应用在 LSTM 上也能起到锦上添花的效果。

位置编码有多种选择，比如可学习的、固定的<sup>[21]</sup>和相对的等。其中可学习的位置编码对于短句训练数据不太友好，而相对位置表达并非原论文提出的方法，而是之后的改进方法，我对其并不是很了解，所以这里采用绝对位置编码来测试效果，如果能对某些指标有一定优化，那么采用更先进的位置编码理应能达到更好的结果。

绝对位置编码也即正弦位置编码，本质上是一个二维矩阵。通过使用不同频率的正弦和余弦函数产生后，再和对应位置的词向量相加即可。其计算公式如下：

$$PE(pos, 2i) = \sin(pos / 10000^{2i/d_{model}}) \quad (3.3)$$

$$PE(pos, 2i+1) = \cos(pos / 10000^{2i/d_{model}}) \quad (3.4)$$

其中  $pos$  是词在句中的位置， $i$  是词向量所在的维度。基于该公式我编写了位置

编码的相关类，其核心代码如下：

```
class PositionalEncoding(nn.Module):
    def __init__(self, d_model, max_len=2000, dropout=0.1):
        super(PositionalEncoding, self).__init__()
        self.dropout = nn.Dropout(p=dropout)
        pe = torch.zeros(max_len, d_model)
        position = torch.arange(0, max_len, dtype=torch.float).unsqueeze(1)
        exp_term = torch.exp(torch.arange(0, d_model, 2).float() * -math.log(10000.0) / d_model)
        pe[:, 0::2] = torch.sin(position * exp_term)
        pe[:, 1::2] = torch.cos(position * exp_term)
        pe = pe.unsqueeze(0).transpose(0, 1)
        self.register_buffer('pe', pe)

    def forward(self, x):
        x += self.pe[:x.size(0), :]
        return self.dropout(x)
```

它继承自 `nn.Module` 类，并重写了构造函数和前向传播方法。在词向量用 LSTM 前，调用它来添加位置编码信息。

其中 `exp_term` 的求解通过数学变换获得，具体公式如下：

$$1/10000^{2i/d_{model}} = e^{\log(10000^{-2i/d_{model}})} \quad (3.5)$$

此外，通过 `register_buffer()` 方法，将 `tensor` 注册成 `buffer`，便于模型保存和加载。

添加位置编码后，将模型在 VisDial v1.0 上测试评估，在训练的前几轮的评估结果，最后几轮的评估结果反而不好，但在训练到中间轮次时，在召回率，标准答案的平均排序和平均倒数排名上取得了略微更好的结果，但是在更加适用于多个答案候选标注的 NDCG 指标上，结果略微降低。具体实验结果将在下一章给出。

### 3.3.2 损失函数及其改进

基于注意力的双视图网络模型使用的损失函数是多分类问题常用的 *soft max loss*，也即先进行一个 *softmax* 处理得到概率：

$$p_i = e^{z_i} / (\sum_k e^{z_k}) \quad (3.6)$$

其中， $z_i$  为 **encoder** 层的输出和候选答案的 LSTM 编码之间的点积结果，在此基础上采用交叉熵损失函数：

$$L_D = -\sum_{i=1}^{100} y_i \log(p_i) \quad (3.7)$$

其中， $y_i$  为候选答案的独热编码。

将上述(3.6)和(3.7)式合并可得：

$$L_D = -\log(e^{z_i} / \sum_k e^{z_k}) \quad (3.8)$$

其中  $z_k$  表示类别得分向量的第  $k$  个元素，它事实上是一个内积结果。

*softmax loss* 的优点是类间的距离优化的比较好，但是类内的距离相对还是比较松散。为了解决这种情况，2016 年 Liu W 等人提出了 Large-Margin Softmax Loss<sup>[22]</sup>，简称 *L-softmax loss*。它在传统的 *softmax loss* 公式中添加了超参数  $m$  来表示分类间隔，用来增加学习的难度，使模型在训练过程中学习更具区分性的特征，从而使类内更加紧凑。由于学习难度变大，该方法一定程度可以避免过拟合。下面介绍 *L-softmax loss* 的具体公式推导。

首先依据 *softmax loss* 的公式也就是（3.8）式，上面已经提过， $z_k$  事实上是编码层输出和候选答案编码的内积，因此可以这样变换：

$$z_k = \|W_k\| \|x_i\| \cos(\theta_k) \quad (3.9)$$

其中， $W_k$  为候选答案编码， $x_i$  为训练得到的输出。因此（3.8）式可以变为下式

$$L_i = -\log\left(\frac{e^{\|W_{y_i}\| \|x_i\| \cos(\theta_{y_i})}}{\sum_k e^{\|W_k\| \|x_i\| \cos(\theta_k)}}\right) \quad (3.10)$$

对于分类问题的两个类别 1 和 2，如果样本  $x$  属于类别 1，则模型希望：

$$\|W_1\| \|x\| \cos(\theta_1) > \|W_2\| \|x\| \cos(\theta_2) \quad (3.11)$$

为使分类更加严格， $L-\text{softmax loss}$  引入了超参数  $m$  作为分类间隔。也就是要求：

$$\|W_1\| \|x\| \cos(m\theta_1) > \|W_2\| \|x\| \cos(\theta_2) \quad (3.12)$$

其中， $0 \leq \theta_1 \leq \frac{\pi}{m}$ ， $m$  是一个大于 1 的正整数。 $m=1$  时，就是特例  $\text{softmax loss}$ 。

由于余弦函数在  $(0, \pi)$  上是单调递减的，易证：

$$\|W_1\| \|x\| \cos(\theta_1) \geq \|W_1\| \|x\| \cos(m\theta_1) > \|W_2\| \|x\| \cos(\theta_2) \quad (3.13)$$

从而(3.12)式带来的分类效果更加显著。 $L-\text{softmax loss}$  定义为：

$$L_i = -\log\left(\frac{e^{\|W_{y_i}\| \|x_i\| \psi(\theta_{y_i})}}{e^{\|W_{y_i}\| \|x_i\| \psi(\theta_{y_i})} + \sum_{k \neq y_i} e^{\|W_k\| \|x_i\| \cos(\theta_k)}}\right) \quad (3.14)$$

其中  $\psi(\theta)$  定义如下：

$$\psi(\theta) = \begin{cases} \cos(m\theta), 0 \leq \theta \leq \frac{\pi}{m} \\ P(\theta), \frac{\pi}{m} < \theta \leq \pi \end{cases} \quad (3.15)$$

其中  $P(\theta)$  是单调递减函数，确保正确性。

为了简化定义，目前一般方法是构建一个特殊的  $\psi(\theta)$ ：

$$\psi(\theta) = (-1)^t \cos(m\theta) - 2t, \theta \in \left[\frac{t\pi}{m}, \frac{(t+1)\pi}{m}\right] \quad (3.16)$$

其中  $t \in [0, m-1]$ 。由于  $\cos(\theta_k)$  可以方便的替换为  $\frac{W_k x_i}{\|W_k\| \|x_i\|}$ ，且根据棣莫弗定理：

$$(\cos(x) + i \sin(x))^m = \cos(mx) + i \sin(mx) \quad (3.17)$$

与其二项式展开联立，得

$$\cos(mx) + i \sin(mx) = \sum_{n=0}^m \frac{m!}{n!(m-n)!} i^n \sin^n(x) \cos^{m-n}(x) \quad (3.18)$$

由于  $n$  为奇数时为纯虚数，故只需考虑偶数情况，则  $\cos(m\theta)$  可以推导为：



$$\cos(m\theta) = C_m^0 \cos^m(\theta) - C_m^2 \cos^{m-2}(\theta) \sin^2(\theta) + \dots (-1)^n C_m^{2n} \cos^{m-2n}(\theta) \sin^{2n}(\theta) + \dots \quad (3.19)$$

本文选择取  $m=4$ ，由于添加了类间距离  $m$  后，学习变得更加困难，应用该损失函数的情况下，训练变得更加缓慢，但是当训练到第 2, 3 轮次时部分指标有了改善，证明了该损失函数的可行性。但是由于该损失函数的应用样例较少，本文只能根据理想情况来编写其代码，所以在训练足够多的轮次后，评估结果几乎没有区别。如果能参考到更多的应用方法细节，应该能更有效的实现原作者提出的效果。

### 3.3.3 优化器和其他细节修改

除对模型添加位置编码和改进损失函数外，本文还改进了学习率的优化器，并对问题表示和历史表示在计算注意力分数时所通过的全连接网络的激活函数做了修改。

基于注意力的双视图网络模型中所用的学习率优化器 **RMSProp**。它对梯度计算量微分平方加权平均数，有利于消除摆动幅度大的方向，用来修正摆动幅度。考虑到 **Adam** 优化器结合了 **Momentum** 算法和 **RMSProp** 算法的特点，这里我对该模型改用 **Adam** 优化器来调整，既能适应视觉对话稀疏梯度的特点，又能缓解梯度摆动的问题。

在计算问题表示和历史表示的注意力分数时，原模型采用的激活函数是 *Sigmoid* 函数，考虑到网络使用 *Sigmoid* 存在梯度消失且训练过慢的特点，同时为了避免神经元死亡，我改用了 *LeakyRelu* 或 *ELU* 函数，由于网络过于复杂，使用 *ELU* 速度很慢，所以我最终决定选用 *LeakyRelu* 作为此处的激活函数。

具体实验细节将在下一章阐述。

## 3.4 本章小结

本章首先介绍对话任务最重要的需解决的问题，阐述研究基于注意力机制的双视图网络模型的原因与改进目的。然后重点分析本文研究的基于注意力机制的双视图网

络模型的框架，又深入剖析了该模型各个模块的原理，添加了位置编码来加强该模型对序列顺序的表达，并对损失函数和学习率等做了一个改进，下一章将阐述具体的实验细节。

## 第 4 章 模型实现及结果分析

本章在深入分析了基于注意力的双视图网络模型的原理后,对该方法进行了复现,并扩展了其数据集,在 VisDial v0.9 上也做了训练和评估,然后将上节提到的优化器和激活函数的改进写入其中,并以消融实验验证了它们的效果,之后测试了位置编码和损失函数的实际效果,最后进行了部分结果的可视化展示。

### 4.1 模型实现过程

#### 4.1.1 实验环境

由于该模型网络结构复杂,所采用的 VisDial v1.0 数据集规模庞大,大小约 100G,所以起初在自己的 1050ti 上训练极为缓慢,训练了 38.5 小时后才训练了 6 个轮次,还发生显存溢出的错误,所以之后我在 AutoDL 的云端 GPU 上运行,用了两块 2080tiGPU,完成一次训练大概 8 小时,尽管还是缓慢,但勉强可以接受。

实验环境如下: GPU 为 RTX2080ti (11GB) \*2, CPU 为 8 核 Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz,实际 Conda 环境下 CUDA 版本为 10.1,Python 版本 3.7。

#### 4.1.2 实验流程

实验流程如图 4.1 所示。首先在本地运行环境下按模板代码的要求将代码部署完成,下载该模型所用的数据集,修改要求的参数后运行,由于训练过于缓慢,改用云端 GPU 来训练,通过 AutoDL 提供的云盘压缩上传代码和数据集,缺点是不太方便,如果需要同步训练两份修改后的代码,需要将庞大的数据集再传到另一个实例上。在云平台上配置完基本环境后开始进行实验。

其次在 VisDial v1.0 数据集上复现了模型,评估结果基本与原文相符。

然后添加了对 VisDial v0.9 的数据读取等方法,扩展了它对 VisDial v0.9 数据集

的支持，在两种数据集上对基模型进行训练评估。具体实验细节将在 4.2 节。

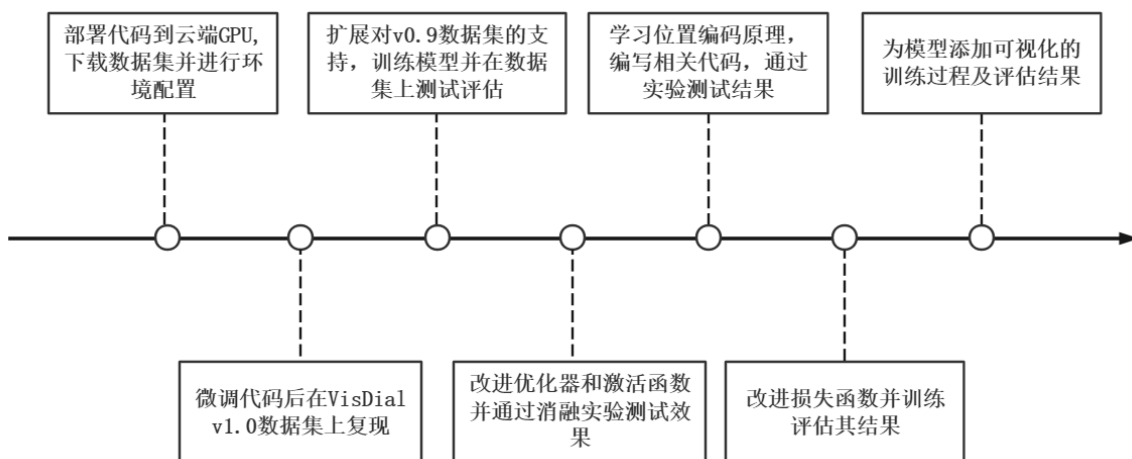


图 4.1 实验流程

之后改进了优化器，计算问题和历史表示的注意力分数时采用的激活函数，并通过消融实验验证它们的作用。

随后为模型添加了绝对位置编码，其核心类代码在 3.3.1 节已经给出，这里不再赘述，实验测试其效果，并分析其不足之处与原因。

然后对损失函数进行改进，基于 Github 作者 amirhfarzaneh 对  $L-\text{softmax loss}$  在 CNN 上的应用示例<sup>[23]</sup>，我编写了本模型使用该损失函数的相关类。核心代码包括：

(1) 二项式系数，三角函数指数的生成：

```

self.C_m_2n = torch.Tensor(binom(margin, range(0, margin + 1, 2))).to(device)
self.cos_powers = torch.Tensor(range(self.margin, -1, -2)).to(device)
self.sin2_powers = torch.Tensor(range(len(self.cos_powers))).to(device)
self.signs = torch.ones(margin // 2 + 1).to(device)
self.signs[1::2] = -1
  
```

(2) 计算  $\cos(\theta)$  的方法：

```

def calculate_cos_m_theta(self, cos_theta):
    sin2_theta = 1 - cos_theta**2
    cos_terms = cos_theta ** self.cos_powers
    sin2_terms = (sin2_theta ** self.sin2_powers)
    cos_m_theta = (self.signs * self.C_m_2n *
                   cos_terms * sin2_terms)
  
```

然后对改进后的模型在两种数据集上进行测试评估，并分析其优势与不足。  
最后，采用 tensorboard 为模型添加了可视化的训练过程与评估结果。

## 4.2 数据集扩展

视觉对话模型所常用的数据集已经介绍过，以 VisDial v1.0 数据集为例，它由图片和包含图片 id、对话历史和当前问题的 json 文件组成。如图 4.2 和图 4.3 所示。

使用原模型在 VisDial v1.0 数据集上的训练共耗时约 8 小时，它在最后两个轮次上的训练结果基本达到了原文所述水平，具体的达到的指标如表 4.1 所示，已经处于视觉对话任务的先进水平。

此外，它在第一轮次上的训练也已经达到了相对于以往模型来说不错的效果，如表 4.2 所示。

图 4.4 展现了该模型在 VisDial v1.0 训练集上的完整训练过程。

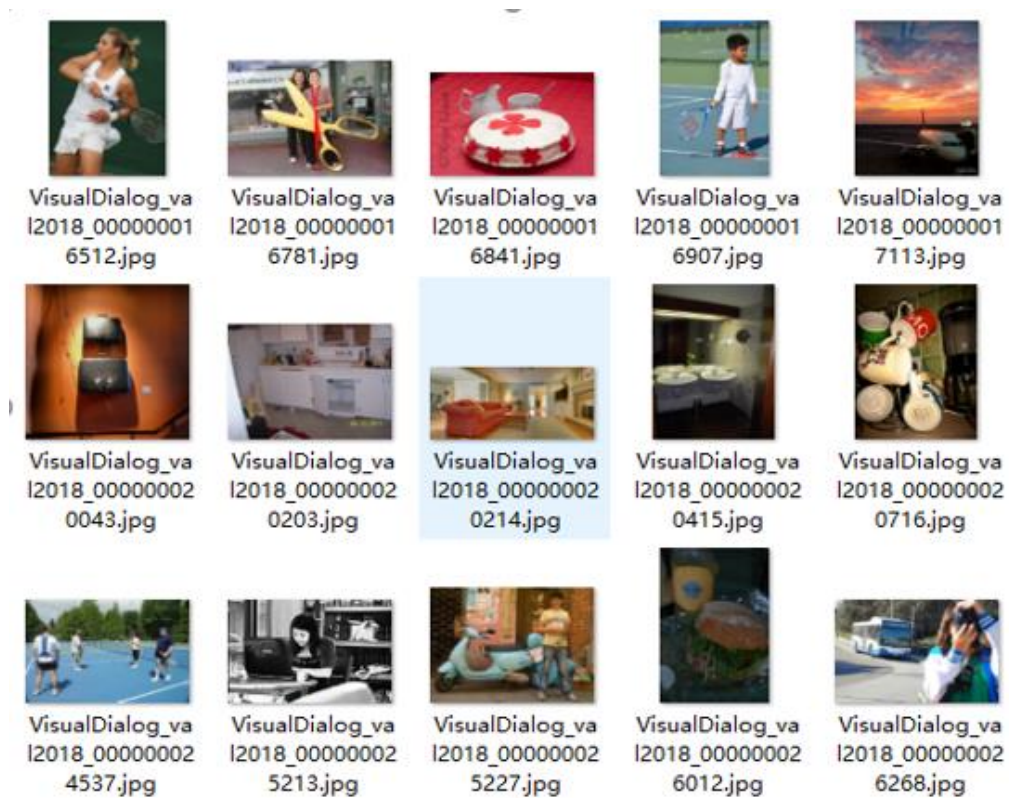


图 4.2 数据集图像样例

```
{
  'data': {
    'questions': [
      'does it have a doorknob',
      'do you see a fence around the bear',
      ...
    ],
    'answers': [
      'no, there is just green field in foreground',
      'countryside house',
      ...
    ],
    'dialogs': [
      {
        'image_id': <image id>,
        'caption': <image caption>,
        'dialog': [
          {
            'question': <index of question in `data.questions` list>,
            'answer': <index of answer in `data.answers` list>,
            'answer_options': <100 candidate answer indices from `data.answers`>,
            'gt_index': <index of `answer` in `answer_options`>
          },
          ... (10 rounds of dialog)
        ]
      },
      ...
    ]
  },
  'split': <VisDial split>,
  'version': '1.0'
}
```

图 4.3 数据集 json 文件样例

表 4.1 原模型最终评估指标

指标	结果
r@1	0.5179747939109802
r@5	0.8198158740997314
r@10	0.9070736169815063
mean	3.9489340782165527
mrr	0.6523899435997009
ndcg	0.603452742099762

表 4.2 原模型一轮训练后评估结果

指标	结果
r@1	0.4140504002571106
r@5	0.7136143445968628
r@10	0.8198158740997314
mean	6.359447479248047
mrr	0.5536373853683472
ndcg	0.47224414348602295

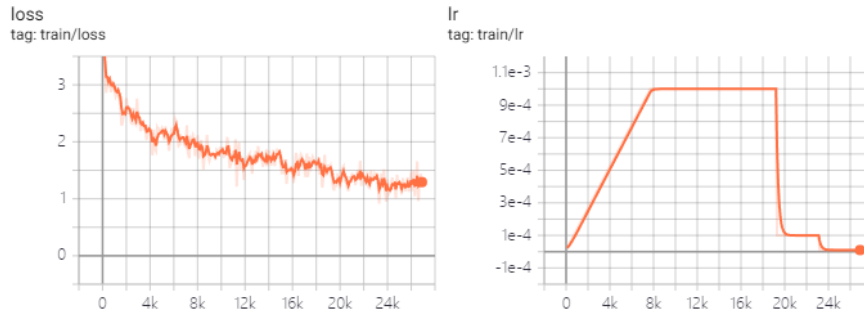


图 4.4 原模型完整训练过程可视化

在 VisDial v1.0 数据集上完成其结果的复现后，我对该模型扩展了对 VisDial v0.9 数据集的支持，主要是数据读取方法的添加：

```
if self.hparams.dataset_version == '0.9':
    with open('data/v0.9/visdial_0.9_%s.json', "r") as file:
        visdial_olddata = json.load(file)
        self.text_feature_id_list = [image_dialog["image_id"] for image_dialog in
visdial_olddata["data"]["dialogs"]]
        with open('data/v0.9/visdial_0.9_train.json', "r") as file_train:
            visdial_traindata = json.load(file_train)
            self.train_text_feature_id_set = set(
[image_dialog["image_id"] for image_dialog in visdial_traindata["data"]["dialogs"]])
            with open('data/v0.9/visdial_0.9_val.json', "r") as file_val:
                visdial_valdata = json.load(file_val)
                self.val_text_feature_id_set = set(
[image_dialog["image_id"] for image_dialog in visdial_valdata["data"]["dialogs"]])
```

在 VisDial v0.9 数据集上训练一轮后，评估结果如表 4.3 所示。

表 4.3 原模型在 VisDial v0.9 上经过一轮训练后结果

指标	结果
r@1	0.3907461166381836
r@5	0.6865794658660889
r@10	0.7937015295028687
mean	7.115697860717773
mrr	0.531388521194458
ndcg	0.4673159122467041

## 4.3 模型改进实现

### 4.3.1 优化器和激活函数的消融研究

基于之前所述的采用 Adam 优化器和 *LeakyRelu* 激活函数的改进，我通过消融实验对比了训练结果，同样是经过一轮的训练，采用 *LeakyRelu* 激活函数后评估指标对比如表 4.4 所示。实验结果表明，mean 指标的提升结果比较明显。

表 4.4 采用 *LeakyRelu* 激活函数后的评估结果

指标	原模型	LeakyRelu
r@1	0.4140504002571106	0.4195736348628998
r@5	0.7136143445968628	0.7191860675811768
r@10	0.8198158740997314	0.8205910921096802
mean	6.359447479248047	6.273159027099609
mrr	0.5536373853683472	0.558556318283081
ndcg	0.47224414348602295	0.47497835755348206

仅采用 Adam 优化器的评估结果与原模型对比如表 4.5 所示。



表 4.5 采用 Adam 优化器后的评估结果

指标	原模型	Adam
r@1	0.4140504002571106	0.41884690523147583
r@5	0.7136143445968628	0.7195736169815063
r@10	0.8198158740997314	0.8253875970840454
mean	6.359447479248047	6.149224758148193
mrr	0.5536373853683472	0.5591735243797302
ndcg	0.47224414348602295	0.4886162281036377

综合采用 Adam 优化器和 *LeakyRelu* 激活函数后，再次在 VisDial v1.0 数据集上进行训练，最后的评估结果对比如表 4.6 所示。

表 4.6 采用 Adam 优化器和 *LeakyRelu* 激活函数后的实验结果

指标	原模型	Adam+ LeakyRelu
r@1	0.4140504002571106	0.41986432671546936
r@5	0.7136143445968628	0.7232073545455933
r@10	0.8198158740997314	0.8280038833618164
mean	6.359447479248047	6.08357572555542
mrr	0.5536373853683472	0.5604297518730164
ndcg	0.47224414348602295	0.4932383894920349

实验结果表明，用 Adam 优化器和 *LeakyRelu* 激活函数后，模型效果的提升比较明显，尤其是针对 ndcg 指标，经过一轮的训练取得了不错的结果。

### 4.3.2 位置编码改进

如上文所述为词向量添加了位置编码，并进行训练，评估结果在第一轮上并没有发生改进，随着训练的进行，在第 3 轮上的部分指标明显有进步。但是由于本文只是

粗略地对词向量在 LSTM 编码前添加位置编码,没有考虑整个模型其他的细节优化,也没有在模型改变后进行参数地调优,所以训练后期的评估指标不如未采用位置编码的模型结果。尽管如此,位置编码能在某一轮训练的部分指标上取得不错的效果,一定程度上可以体现其优点,经过 3 轮训练后的测试结果如表 4.7 所示。

表 4.7 添加位置编码后的模型结果

指标	原模型	采用位置编码的模型
r@1	0.49278101325035095	0.49777132272720337
r@5	0.7951065897941589	0.7968991994857788
r@10	0.8889535069465637	0.8891472816467285
mean	4.37727689743042	4.351308345794678
mrr	0.6288343071937561	0.6329203844070435
ndcg	0.5490666031837463	0.5372791886329651

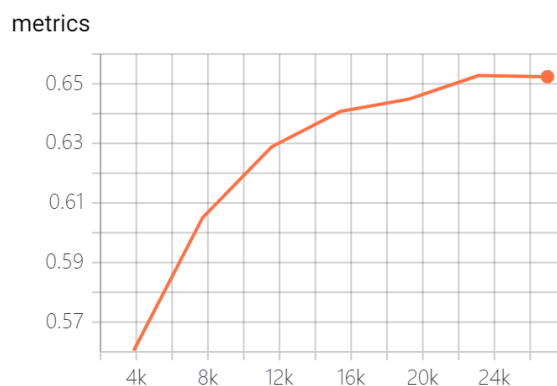
实验结果表明,第三轮训练后,添加位置编码后的模型在大部分指标上的评估结果取得了并不是很明显的优势,但是在 ndcg 指标的评估结果比较差,说明未经优化的位置编码对于有多个正确答案标注的候选答案数据没有一个很好的分类效果。

### 4.3.3 损失函数改进

在 4.3.1 节改进的基础上,采用  $L-\text{softmax loss}$  并进行实验,由于学习的难度变大,且没有对参数做一个调优,仅仅套用了原模型的各种网络参数,训练速率明显降低。在训练的开始,结果反而不如之前的模型,但随着训练的进行,逐渐取得了更好的评估结果,尽管在多次实验测试后发现并不稳定,且即使是有改进的情况下相对目前的最先进方法也有较大差距。同样由于缺乏对细节的调优知识,完成整个训练流程后,模型评估效果相对之前并没有明显的提升。如果采用更灵活地训练方式,也即训练更少轮次就进行评估测试,可以起到不错的效果。模型结果如表 4.8 和图 4.5 所示。

表 4.8 采用  $L-\text{softmax loss}$  训练两轮后的结果

指标	原模型	改进损失函数的模型
r@1	0.4661821722984314	0.47335270047187805
r@5	0.7719476819038391	0.7727712988853455
r@10	0.8699612617492676	0.8713662624359131
mean	4.8938469886779785	4.8322672843933105
mrr	0.6050736904144287	0.60967618227005
ndcg	0.5239806175231934	0.5381038188934326

图 4.5 采用  $L-\text{softmax loss}$  后模型整个训练过程 mrr 指标的变化

采用  $L-\text{softmax loss}$  后在 VisDial v0.9 数据集上的最终结果如表 4.9 所示。模型在训练 2-4 轮时，loss 值的下降速度略优于基模型，随后逐渐趋于平缓。同样由于缺乏  $L-\text{softmax loss}$  在基模型上的应用知识，尤其是分类间隔的设置依据、如何根据  $L-\text{softmax loss}$  的特性制定新的学习策略以及其他参数的调优，模型的最终评估结果相对基模型来说没有明显改进。

同时，由于 VisDial v0.9 数据集在质量和数量上都不如 VisDial v1.0 数据集，其最终结果相对于在 VisDial v1.0 数据集上的训练结果还有一定差距，仅达到了在 VisDial v1.0 数据集上训练 4 轮后的水准。

表 4.9 采用  $L-\text{softmax loss}$  后模型在 VisDial v0.9 数据集上的最终结果

指标	结果
r@1	0.5028100609779358
r@5	0.804215133190155
r@10	0.8944767713546753
mean	4.273691654205322
mrr	0.6373743414878845
ndcg	0.600285530090332

#### 4.3.4 实验结果分析与可视化

经改进后的基于注意力的双视图网络模型在需要灵活地规划训练任务时,可以获得更加满意的结果,如果不添加位置编码的话,即使采用原先的训练计划,它的模型效果也几乎不会下降。在对原模型添加 `tensorboard` 可视化训练功能后,图 4.6 展示了改进后的模型在 VisDial v1.0 数据集上完整的训练指标变化过程。

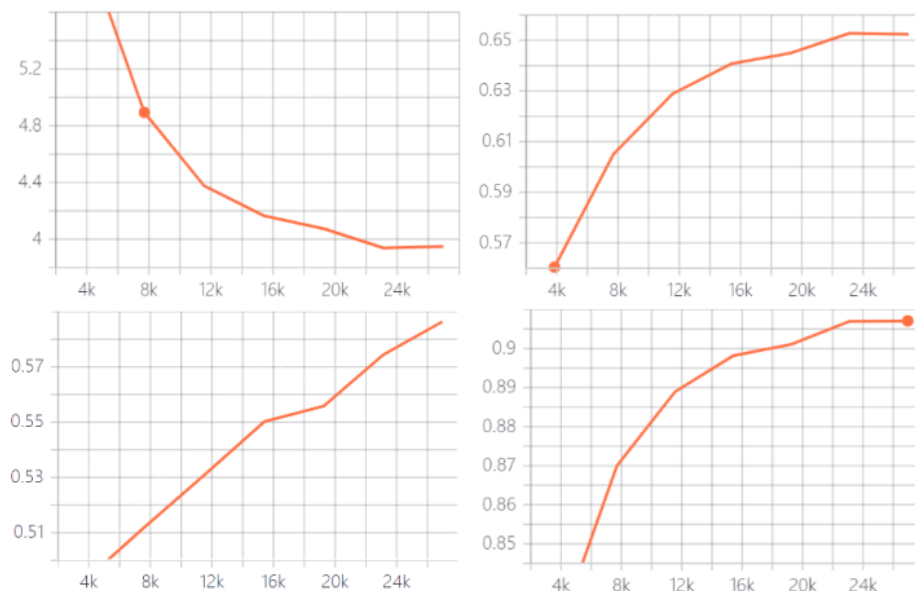


图 4.6 改进后的模型训练过程指标变化, (左上): mean 指标, (右上): mrr 指标, (左下): ndcg 指标, (右下): r@10 指标。



图片 id:239030

对话历史

问题 1: what color is the tablet  
回答 1: black and silver  
问题 2: is it on a desk  
回答 2: no

当前问题: is it on a table  
正确答案: yes

基模型答案: yes  
现模型答案: yes

对话轮次: 3

(a):基模型 (✓) 现模型 (✓)



图片 id:99643

对话历史


问题 1: is this in color  
回答 1: white and blue  
问题 2: how old does the girl look  
回答 2: around 3  
问题 3: is this inside  
回答 3: yes, i believe so  
问题 4: are there other people around  
回答 4: no

当前问题: is the room well lit  
正确答案: yes

基模型答案: yes  
现模型答案: it doesn't seem to have any visible dirt

对话轮次: 5

(b):基模型 (✓) 现模型 (✗)



图片 id: 36690

对话历史

问题 1: what color is the jet  
回答 1: white and blue  
问题 2: is it a commercial plane  
回答 2: yes  
问题 3: can the sky be seen  
回答 3: yes  
问题 4: are there any clouds  
回答 4: no  
问题 5: do you see any other planes in the picture  
回答 5: no  
问题 6: can you see anything else other than the plane  
回答 6: no

当前问题: is the plane crashing  
正确答案: no

基模型答案: not that i can see  
现模型答案: no

对话轮次: 7

(c):基模型 (✗) 现模型 (✓)

图 4.7 模型对话结果对比

图 4.7 和图 4.8 展示了几组基模型和改进后模型的对话结果。图 4.7 中问题的答

案为二元性答案，其中(a)结果基模型和现模型都正确，(b)结果问题为模糊性问题，基模型的结果更好，(c)结果的问题比较精确，现模型回答正确。图 4.8 中问题的答案为长序列，(a)结果问题为模糊性问题，基模型和现模型都错误，但现模型更接近答案，(b)结果基模型错误而现模型正确。



图 4.8 长序列答案下模型结果对比

#### 4.4 本章小结

本文在深入分析了基于注意力的双视图网络模型后，完成了对该模型的改进，本章通过实验测试了它的可行性，效果和缺陷。实验表明，优化器和激活函数的改进取得了不错的效果，而模型输入编码和损失函数的改进存在局限性，体现在训练速率降低或在训练多轮次后改进效果变差或者没有明显改进上，但是在进行短期且灵活的训练和评估时效果可以接受。此外，位置编码和  $L-\text{softmax loss}$  损失函数本身可能还存

在更好更适用的变种，同时模型修改后缺乏对细节调优的知识和经验，未来应该在继续学习更先进的位置编码和损失函数的基础上，学习模型参数调优的相关知识经验，进一步改进模型的效果，并在更多数据集上进行测试。

## 第 5 章 总结与展望

### 5.1 本文总结

图像和语义理解逐渐成为自然语言和计算机视觉具有挑战性的跨领域研究任务。视觉对话作为视觉问答的姊妹任务，已经成为了当前深度学习的重要研究对象。机器对于图像语义的理解，对于推动能以自然语言交流的机器的发展具有重要意义。

本文首先介绍了视觉对话任务并分析其难点，然后介绍了主流的视觉对话模型类别并挑选代表性模型分析其特点。然后，本文重点分析了基于注意力的双视图网络模型的框架和优点，深入剖析了其各个模块的构建细节和作用，之后完成了对它的测试，优化和模型改进，取得较为满意的效果，最终完成了基于注意力的视觉对话模型架构。

本文的主要工作及贡献如下：

（1）分析了主流的各种视觉对话模型，对每一类挑选代表性的模型对比总结它们的特点，并选定了本文研究的基模型。

（2）分析了基于注意力的双视图网络模型的原理，框架模块及优缺点，基于该模型提供的模板代码对其进行复现并部署在云端 GPU 平台。

（3）根据基模型的局限性对其进行改进，包括对词向量进行位置编码，改进损失函数以及对优化器和部分激活函数进行修改。

（4）添加了模型对 VisDial v0.9 数据集的读取方法，对基模型和改进后的模型在 VisDial v1.0 数据集和 VisDial v0.9 数据集上进行测试评估，并为模型添加了可视化的训练过程和评估结果。

### 5.2 工作展望

在进行了大量的实验后，发现位置编码的添加所取得的评估结果不能令人满意，尤其在进行了大量轮次的训练后评估指标不如原模型，因此位置编码的应用细节仍然需



要大量工作来测试和优化。

与此相同，损失函数的改进在进行充分训练后优势也不明显，而这与其理应达到的效果是不符的，后续应当仔细阅读  $L-\text{softmax loss}$  原论文及源代码样例，并寻找更多的应用示例来学习改进。

此外，本文研究的视觉对话模型主要基于判别式模型，易于评测，但是实际对话中，给定范围内的答案不一定满足用户的需求，所以可以进一步学习能生成多样化答案的生成式模型。

## 参考文献

- [1] Das Abhishek, Kottur Satwik, Gupta Khushi, Singh Avi, Yadav Deshraj, Lee Stefan, Moura Jose, Parikh Devi, Batra Dhruv. Visual Dialog[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019, 41(5): 1242-1256.
- [2] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector[C]// Proceedings of the European Conference on Computer Vision (ECCV). Springer, Cham, 2016: 21-37.
- [3] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition [C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016: 770-778.
- [4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search[J]. Nature, 2016, 529(7587): 484-489.
- [5] Kottur S, Moura J M F, Parikh D, Batra D, Marcus R. Visual Coreference Resolution in Visual Dialog using Neural Module Networks[C]// Proceedings of the European Conference on Computer Vision (ECCV). 2018: 153-169.
- [6] 赵磊, 高联丽, 宋井宽. 面向视觉对话的自适应视觉记忆网络[J]. 电子科技大学学报, 2021, 50(05): 749-753.
- [7] Seo P H, Lehrmann A, Han B, et al. Visual Reference Resolution using Attention Memory for Visual Dialog[C]// Advances in Neural Information Processing Systems. 2017: 3719-3729.
- [8] Zhe Gan, Yu Cheng, Ahmed El Kholy, Linjie Li, Jingjing Liu, Jianfeng Gao. Multi-step Reasoning via Recurrent Dual Attention for Visual Dialog[J/OL]. CoRR, 2019, abs/1902.00579.
- [9] Zheng Z, Wang W, QI S, et al. Reasoning Visual Dialogs with Structural and Partial

- Observations[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 6669-6678.
- [10] Schwartz I, Yu S, Hazan T, et al. Factor Graph Attention[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 6463-6474.
- [11] Jiang X, Yu J, Qin Z., et al. DualVD: an Adaptive Dual Encoding Model for Deep Visual Understanding in Visual Dialogue[C]// Proceedings of the AAAI Conference on Artificial Intelligence,. 2020, 34(7): 11125-11132.
- [12] Qi J., Niu Y., Huang J., Zhang H. Two Causal Principles for Improving Visual Dialog[C]// Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2020: 10860-10869.
- [13] Jiang, X., Du, S., Qin, Z., Sun, Y., & Yu, J. KBGN: Knowledge-Bridge Graph Network for Adaptive Vision-Text Reasoning in Visual Dialogue[C]// Proceedings of the 28th ACM International Conference on Multimedia. 2020:1265-1273.
- [14] Yulei Niu, Hanwang Zhang, Manli Zhang, Jianhong Zhang, Zhiwu Lu, and Ji-Rong Wen. Recursive Visual Attention in Visual Dialog[C]// IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, USA, 2019: 6672-6681.
- [15] Lu J, Kannan A, Yang J, et al. Best of Both Worlds: Transferring Knowledge from Discriminative Learning to a Generative Visual Dialog Model[C]// Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017: 313-323.
- [16] Wu Q, Wang P, Shen C, et al. Are You Talking to Me? Reasoned Visual Dialog Generation through Adversarial Learning[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6106-6115.
- [17] Park Sungjin, Whang Taesun, Yoon Yeochan, Lim Heuseok. Multi-View Attention Network for Visual Dialog[J]. Applied Sciences, 2021, 11(7): 3009.
- [18] Ren S., He K., Girshick R., and Sun J. Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks[J]. IEEE Transaction Pattern Analysis

- Machine Intelligence. 2017, 39(6): 1137-1149.
- [19] Pennington J., Socher R., and Manning C. D. Glove: Global Vectors for Word Representation[C]// Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014: 1532-1543.
- [20] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., and Polosukhin I. Attention is all you need[C]// Advances in Neural Information Processing Systems, 2017: 5998-6008.
- [21] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional Sequence to Sequence Learning[C]// Proceedings of the 34th International Conference on Machine Learning (ICML). 2017, 70: 1243-1252.
- [22] Weiyang Liu, Yandong Wen, Zhiding Yu, Meng Yang. Large-Margin Softmax Loss for Convolutional Neural Networks[C]// Proceedings of The 33rd International Conference on Machine Learning (ICML). 2016: 507-516.
- [23] Yi Wei, Haibo Pu, Yu Zhu, XiaoFan Li. Integrate Receptive Field Block into Large-margin SoftMax Loss for Face Recognition[C]// Proceedings of 2019 3rd International Conference on Machine Vision and Information Technology (CMVIT 2019). 2019: 55-62.

## 致谢



## 苏州大学本科生毕业设计（论文）任务书

学院（部）：计算机科学与技术学院

设计（论文）选题：XXXXXXXXXX 研究和实现					
指导教师姓名		职 称		类 别	毕业论文
学 生 姓 名		学 号		设计（论文）类型	应用型
年 级		专 业	计算机科学与技术	是否隶属科研项目	否
<p>1、设计（论文）的主要任务及目标</p> <p>目标：本课题在分析现有 visual dialog 模型的基础上，分析各种基于注意力模型的利弊，并针对缺点进行改进，以完成对 visual dialog 任务的优化。</p> <p>主要任务：</p> <ul style="list-style-type: none"> <li>（1） 收集并分析现有模型的框架的利弊。</li> <li>（2） 复现主流模型框架。</li> <li>（3） 对模型进行设计方案并优化。</li> <li>（4） 完成代码的编写与方案特点分析以及与主流框架的对比。</li> </ul>					
<p>2、设计（论文）的主要内容</p> <ul style="list-style-type: none"> <li>（1） 阅读大量文献资料，分析对比现有的 visual dialog 模型的优缺点。</li> <li>（2） 实现主流 visual dialog 模型。</li> <li>（3） 分析模型特点并做出优化。</li> <li>（4） 对实验结果进行分析，并对整体模型进行评估。</li> </ul>					
<p>3、设计（论文）的基本要求</p> <ul style="list-style-type: none"> <li>（1） 能够对现存的 visual dialog 框架进行总结归纳，对模型特点进行分析。</li> <li>（2） 将主流的模型进行实现跑通，并做出改进优化，最后对模型进行评估对比与分析。</li> <li>（3） 完成 1000 字左右的文献综述、不少于 3000 个单词的外文翻译。</li> <li>（4） 提交毕业论文和源代码。</li> </ul>					

## 4、主要参考文献

- [1] Das Abhishek, Kottur Satwik, Gupta Khushi, Singh Avi, Yadav Deshraj, Lee Stefan, Moura Jose, Parikh Devi, Batra Dhruv. Visual Dialog[J]. IEEE transactions on pattern analysis and machine intelligence, 2018, 41(5): 1242-1256.
- [2] Park Sungjin, Whang Taesun, Yoon Yeochan, Lim Heuiseok. Multi-View Attention Network for Visual Dialog[J]. Applied Sciences, 2021, 11(7): 3009.
- [3] Shubham Agarwal, Trung Bui, Joon-Young Lee, Ioannis Konstas, and Verena Rieser. 2020. History for Visual Dialog: Do we really need it? [C]. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8182–8197, Online. Association for Computational Linguistics.
- [4] Jiang, X., Du, S., Qin, Z., Sun, Y., & Yu, J. KBGN: Knowledge-Bridge Graph Network for Adaptive Vision-Text Reasoning in Visual Dialogue. Proceedings of the 28th ACM International Conference on Multimedia. 2020.
- [5] Yulei Niu, Hanwang Zhang, Manli Zhang, Jianhong Zhang, Zhiwu Lu, and Ji-Rong Wen. Recursive Visual Attention in Visual Dialog, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Lo-ng Beach, USA, 2019.
- [6] Zhou Yu, Jun Yu 0002, Yuhao Cui, Dacheng Tao, Qi Tian 0001. Deep Modular Co-Attention Networks for Visual Question Answering[C]. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 6274-6283.
- [7] Zhe Gan, Yu Cheng, Ahmed El Kholy, Linjie Li, Jingjing Liu, Jianfeng Gao. Multi-step Reasoning via Recurrent Dual Attention for Visual Dialog[J/OL]. CoRR, 2019, abs/1902.00579.
- [8] Qi J., Niu Y., Huang J., Zhang H.. Two causal principles for improving visual dialog[C]. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2020.
- [9] 赵磊, 高联丽, 宋井宽. 面向视觉对话的自适应视觉记忆网络[J]. 电子科技大学学报, 2021, 50(05): 749-753.
- [10] Badri N., Patro, Anupriy, Vinay P., Namboodiri. Probabilistic framework for solving visual dialog[J]. Pattern Recognition, Volume 110, 2021, 107586.

## 5、进度安排

	设计（论文）各阶段任务	起 止 日 期
1	资料收集，明确任务，完成任务书	2022-02-13 至 2022-02-22
2	完成文献阅读，文献综述和外文翻译	2022-02-23 至 2022-03-06
3	分析模型特点，总结主流框架	2022-03-07 至 2022-03-25
4	改进优化主流模型	2022-03-26 至 2022-04-23
5	评估对比分析，撰写毕业论文	2022-04-24 至 2022-05-10
6	修改论文，准备答辩	2022-05-11 至 2022-05-19



---

# Attention Is All You Need

---

**Ashish Vaswani\***  
Google Brain  
avaswani@google.com

**Noam Shazeer\***  
Google Brain  
noam@google.com

**Niki Parmar\***  
Google Research  
nikip@google.com

**Jakob Uszkoreit\***  
Google Research  
usz@google.com

**Llion Jones\***  
Google Research  
llion@google.com

**Aidan N. Gomez\*†**  
University of Toronto  
aidan@cs.toronto.edu

**Lukasz Kaiser\***  
Google Brain  
lukaszkaizer@google.com

**Illia Polosukhin\*‡**  
illia.polosukhin@gmail.com

## Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English- to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

## 1 Introduction

Recurrent neural networks, long short-term memory<sup>[13]</sup> and gated recurrent<sup>[7]</sup> neural networks in particular, have been firmly established as state-of-the-art approaches in sequence modeling and transduction problems such as language modeling and machine translation<sup>[33, 2, 5]</sup>. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures<sup>[36, 23, 15]</sup>.

Recurrent models typically factor computation along the symbol positions of the input and output

sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states  $h_t$ , as a function of the previous hidden state  $h_{t-1}$  and the input for position  $t$ . This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks<sup>[20]</sup> and conditional computation<sup>[31]</sup>, while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences<sup>[2, 18]</sup>. In all but a few cases<sup>[26]</sup>, however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

## 2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU<sup>[22]</sup>, ByteNet<sup>[17]</sup> and ConvS2S<sup>[9]</sup>, all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions<sup>[12]</sup>. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations<sup>[4, 26, 27, 21]</sup>.

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [16, 17] and [9].

## 3 Model Architecture

Most competitive neural sequence transduction models have an encoder-decoder structure<sup>[5, 2, 33]</sup>. Here, the encoder maps an input sequence of symbol representations  $(x_1, \dots, x_n)$  to a

sequence of continuous representations  $z = (z_1, \dots, z_n)$ . Given  $z$ , the decoder then generates an output sequence  $(y_1, \dots, y_m)$  of symbols one element at a time. At each step the model is auto-regressive<sup>[10]</sup>, consuming the previously generated symbols as additional input when generating the next.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

### 3.1 Encoder and Decoder Stacks

**Encoder:** The encoder is composed of a stack of  $N = 6$  identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection<sup>[11]</sup> around each of the two sub-layers, followed by layer normalization<sup>[1]</sup>. That is, the output of each sub-layer is  $\text{LayerNorm}(x + \text{Sublayer}(x))$ , where  $\text{Sublayer}(x)$  is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension  $d_{\text{model}} = 512$ .

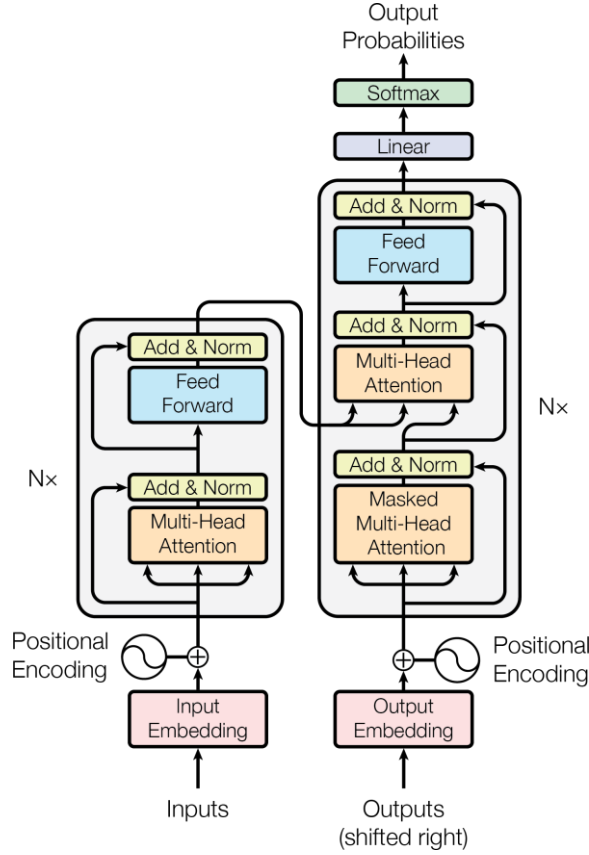


Figure 1: The Transformer - model architecture.

**Decoder:** The decoder is also composed of a stack of  $N = 6$  identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position  $i$  can depend only on the known outputs at positions less than  $i$ .

### 3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

#### 3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension  $d_k$ , and values of dimension  $d_v$ . We compute the dot products of the query with all keys, divide each by  $\sqrt{d_k}$ , and apply a softmax function to obtain the weights on the values.

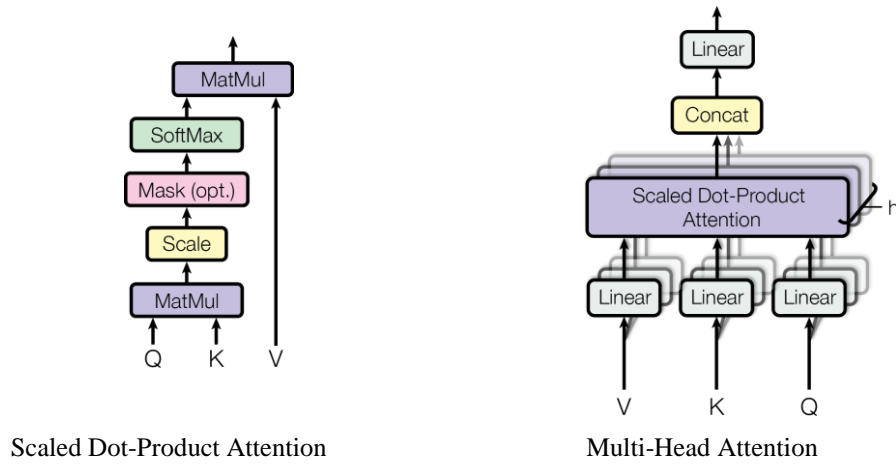


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix  $Q$ . The keys and values are also packed together into matrices  $K$  and  $V$ . We compute

the matrix of outputs as:

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

The two most commonly used attention functions are additive attention <sup>[2]</sup>, and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of  $\frac{1}{\sqrt{d_k}}$ . Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of  $d_k$  the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of  $d_k$  <sup>[3]</sup>. We suspect that for large values of  $d_k$ , the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients. To counteract this effect, we scale the dot products by  $\frac{1}{\sqrt{d_k}}$ .

### 3.2.2 Multi-Head Attention

Instead of performing a single attention function with  $d_{model}$ -dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values  $h$  times with different, learned linear projections to  $d_k$ ,  $d_k$  and  $d_v$  dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding  $d_v$ -dimensional output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$MultiHead(Q, K, V) = \text{Concat}(head_1, \dots, head_h)W^O$$

$$\text{where } head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices  $W_i^Q \in R^{d_{model} \times d_k}$ ,  $W_i^K \in R^{d_{model} \times d_k}$ ,  $W_i^V \in R^{d_{model} \times d_v}$  and  $W^O \in R^{hd_v \times d_{model}}$ .

In this work we employ  $h = 8$  parallel attention layers, or heads. For each of these we use  $d_k = d_v = d_{model}/h = 64$ . Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

### 3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and

the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [36, 2, 9].

- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to  $-\infty$ ) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

### 3.3 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is  $d_{model} = 512$ , and the inner-layer has dimensionality  $d_{ff} = 2048$ .

In the appendix, we describe how the position-wise feed-forward network can also be seen as a form of attention.

### 3.4 Embeddings and Softmax

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension  $d_{model}$ . We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [29]. In the embedding layers, we multiply those weights by  $\sqrt{d_{model}}$ .

### 3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the

bottoms of the encoder and decoder stacks. The positional encodings have the same dimension  $d_{model}$  as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types.  $n$  is the sequence length,  $d$  is the representation dimension,  $k$  is the kernel size of convolutions and  $r$  the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention(restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

In this work, we use sine and cosine functions of different frequencies:

$$PE(pos, 2i) = \sin(pos / 10000^{2i} / d_{model})$$

$$PE(pos, 2i+1) = \cos(pos / 10000^{2i} / d_{model})$$

where  $pos$  is the position and  $i$  is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from  $2\pi$  to  $10000 \cdot 2\pi$ . We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset  $k$ ,  $PE_{pos+k}$  can be represented as a linear function of  $PE_{pos}$ .

We also experimented with using learned positional embeddings [9] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

## 4 Why Self-Attention

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations  $(x_1, \dots, x_n)$  to another sequence of equal length  $(z_1, \dots, z_n)$ , with  $x_i, z_i \in \mathbb{R}^d$ , such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.

One is the total computational complexity per layer. Another is the amount of computation that can

be parallelized, as measured by the minimum number of sequential operations required.

The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies<sup>[12]</sup>. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types.

As noted in Table 1, a self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires  $O(n)$  sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence length  $n$  is smaller than the representation dimensionality  $d$ , which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece<sup>[36]</sup> and byte-pair<sup>[30]</sup> representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size  $r$  in the input sequence centered around the respective output position. This would increase the maximum path length to  $O(n/r)$ . We plan to investigate this approach further in future work.

A single convolutional layer with kernel width  $k < n$  does not connect all pairs of input and output positions. Doing so requires a stack of  $O(n/k)$  convolutional layers in the case of contiguous kernels, or  $O(\log_k(n))$  in the case of dilated convolutions<sup>[17]</sup>, increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of  $k$ . Separable convolutions<sup>[6]</sup>, however, decrease the complexity considerably, to  $O(k \cdot n \cdot d + n \cdot d^2)$ . Even with  $k = n$ , however, the complexity of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer, the approach we take in our model.

As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

## 5 Training

This section describes the training regime for our models.

### 5.1 Training Data and Batching

We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding<sup>[3]</sup>, which has a shared source-target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece



vocabulary<sup>[36]</sup>. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

## 5.2 Hardware and Schedule

We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 100,000 steps or 12 hours. For our big models, (described on the bottom line of table 3), step time was 1.0 seconds. The big models were trained for 300,000 steps (3.5 days).

## 5.3 Optimizer

We used the Adam optimizer<sup>[19]</sup> with  $\beta_1 = 0.9$ ,  $\beta_2 = 0.98$  and  $\epsilon = 10^{-9}$ . We varied the learning rate over the course of training, according to the formula:

$$lrate = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_step^{-1.5}) \quad (3)$$

This corresponds to increasing the learning rate linearly for the first *warmup\_steps* training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used *warmup\_steps* = 4000.

## 5.4 Regularization

We employ three types of regularization during training:

**Residual Dropout** We apply dropout<sup>[32]</sup> to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of  $P_{drop} = 0.1$ .

**Attention Dropout** Query to key attentions are structurally similar to hidden-to-hidden weights in a feed-forward network, albeit across positions. The softmax activations yielding attention weights can then be seen as the analogue of hidden layer activations. A natural possibility is to extend dropout [32] to attention. We implement attention dropout by dropping out attention weights as,

$$Attention(Q, K, V) = dropout(\text{softmax}(\frac{QK^t}{\sqrt{d}}))V$$

In addition to residual dropout, we found attention dropout to be beneficial for our parsing experiments.

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [17]	23.75			
Deep-Att + PosUnk [37]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [36]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [31]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [37]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [36]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	<b><math>3.3 \cdot 10^{18}</math></b>	
Transformer (big)	<b>28.4</b>	<b>41.0</b>	$2.3 \cdot 10^{19}$	

**Label Smoothing** During training, we employed label smoothing of value  $\epsilon_{ls} = 0.1$  [34]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

## 6 Results

### 6.1 Machine Translation

On the WMT 2014 English-to-German translation task, Our big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this model is listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the previous best models.

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of 41.17, outperforming all of the previously published single models, at less than 1/4 the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French did not share source and target embeddings,  $d_{ff} = 8192$  and used dropout rate  $p_{drop} = 0.1$ , instead of 0.3.

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of 4 and length penalty  $\alpha = 0.6$  [36]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible [36].

Table 2 summarizes our results and compares our translation quality and training costs to other model architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained

single-precision floating-point capacity of each GPU.

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	$N$	$d_{\text{model}}$	$d_{\text{ff}}$	$h$	$d_k$	$d_v$	$P_{\text{drop}}$	$\epsilon_{ls}$	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$	
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65	
(A)					1	512	512				5.29	24.9	
					4	128	128				5.00	25.5	
					16	32	32				4.91	25.8	
					32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58	
					32					5.01	25.4	60	
(C)	2									6.11	23.7	36	
	4									5.19	25.3	50	
	8									4.88	25.5	80	
	256				32	32				5.75	24.5	28	
	1024				128	128				4.66	26.0	168	
			1024								5.12	25.4	53
			4096								4.75	26.2	90
(D)							0.0			5.77	24.6		
							0.2			4.95	25.5		
								0.0		4.67	25.3		
								0.2		5.47	25.7		
(E)	positional embedding instead of sinusoids									4.92	25.7		
big	6	1024	4096	16				0.3	300K	<b>4.33</b>	<b>26.4</b>	213	

## 6.2 Model Variations

To evaluate the importance of different components of the Transformer, we varied our base model in different ways, measuring the change in performance on English-to-German translation on the development set, newstest2013. We used beam search as described in the previous section, but no checkpoint averaging. We present these results in Table 3.

In Table 3 rows (A), we vary the number of attention heads and the attention key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads.

In Table 3 rows (B), we observe that reducing the attention key size  $d_k$  hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings<sup>[9]</sup>, and observe nearly identical results to the base model.

## 6.3 English Constituency Parsing

To evaluate if the Transformer can generalize to other tasks we performed experiments on English constituency parsing. This task presents specific challenges: the output is subject to strong structural constraints and is significantly longer than the input. Furthermore, RNN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes<sup>[35]</sup>.

We trained a 4-layer transformer with  $d_{model} = 1024$  on the Wall Street Journal (WSJ) portion of the Penn Treebank<sup>[24]</sup>, about 40K training sentences. We also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences<sup>[35]</sup>. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting.

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser et al. (2014) [35]	WSJ only, discriminative	88.3
Petrov et al. (2006) [28]	WSJ only, discriminative	90.4
Zhu et al. (2013) [38]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [38]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [25]	semi-supervised	92.1
Vinyals & Kaiser et al. (2014) [35]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Dyer et al. (2016) [8]	generative	93.3

We performed only a small number of experiments to select the dropout, both attention and residual (section 5.4), learning rates and beam size on the Section 22 development set, all other parameters remained unchanged from the English-to-German base translation model. During inference, we increased the maximum output length to input length + 300. We used a beam size of 21 and  $\alpha = 0.3$  for both WSJ only and the semi-supervised setting.

Our results in Table 4 show that despite the lack of task-specific tuning our model performs surprisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar<sup>[8]</sup>.

In contrast to RNN sequence-to-sequence models<sup>[35]</sup>, the Transformer outperforms the Berkeley-Parser<sup>[28]</sup> even when training only on the WSJ training set of 40K sentences.

## 7 Conclusion

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles. We also provide an indication of the broader applicability of our models through experiments on English constituency parsing.

We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours.

The code we used to train and evaluate our models is available at <https://github.com/tensorflow/tensor2tensor>.

## References

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.
- [3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017.
- [4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.
- [5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.
- [6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.
- [7] Junyoung Chung, Çağlar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.
- [8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. of NAACL*, 2016.
- [9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.
- [10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*,

pages 770–778, 2016.

- [12]Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.
- [13]Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [14]Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841. ACL, August 2009.
- [15]Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.
- [16]Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In *International Conference on Learning Representations (ICLR)*, 2016.
- [17]Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Ko- ray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099v2*, 2017.
- [18]Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017.
- [19]Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
- [20]Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. *arXiv preprint arXiv:1703.10722*, 2017.
- [21]Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.
- [22]Samy Bengio Łukasz Kaiser. Can active memory replace attention? In *Advances in Neural Information Processing Systems, (NIPS)*, 2016.
- [23]Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.
- [24]Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.
- [25]David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159. ACL, June 2006.
- [26]Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In *Empirical Methods in Natural Language Processing*, 2016.
- [27]Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

- 
- [28]Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 433–440. ACL, July 2006.
  - [29]Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.
  - [30]Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.
  - [31]Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.
  - [32]Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
  - [33]Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.
  - [34]Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.
  - [35]Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2015.
  - [36]Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.
  - [37]Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016.
  - [38]Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 434–443. ACL, August 2013.





# Attention Is All You Need

## 摘要

目前主流的序列转导模型都基于包括编码器和解码器的复杂的循环或卷积神经网络，性能最好的模型还通过注意力机制来连接编码器和解码器。我们提出了一种新的简单网络 **Transformer**，它仅仅基于注意力机制，完全摒弃了递归和卷积。在两个机器翻译任务中，实验表明，**Transformer** 模型在质量上更优越，同时更具并行化，且所需要的训练时间显著减少。我们的模型在 WMT 2014 英语转德语翻译任务上达到了 28.4 BLEU，比现有的最好结果（包括集成）提高了 2 BLEU。WMT 2014 英语转法语翻译任务中，我们的模型在 8 个 GPU 上训练 3.5 天后，建立了一个新的单一模型，取得了 41.0 的 BLEU 得分，这只是文献中最佳模型的训练开销的一小部分。我们通过成功地将 **Transformer** 应用于具有大量或有限训练数据的英语依存句法分析任务，证明了 **Transformer** 可以很好地推广到其他任务。

## 1 简介

循环神经网络，尤其是长短期记忆<sup>[13]</sup>和门控循环<sup>[7]</sup>神经网络，在语言建模和机器翻译<sup>[33,25]</sup>等序列建模和转导问题中已被牢固确立为最先进的方法。此后，很多工作不断地扩大了循环语言模型和编码器-解码器架构的界限<sup>[36,23,15]</sup>。

循环模型通常沿输入和输出序列的位置特征来进行计算。为了将位置与计算的步骤对齐，它们生成一系列隐藏状态  $h_t$ ，作为先前隐藏状态  $h_{t-1}$  和位置  $t$  的输入函数。这种固有的顺序性使得训练样本间无法并行，但是并行化在序列长度较长时至关重要，因为内存约束限制了样本的批处理。最近的工作研究通过分解技巧<sup>[20]</sup>和条件计算<sup>[31]</sup>显著提高了计算效率，同时条件计算也提高了模型性能。然而，影响计算效率的根本因素即顺序性仍然存在。

注意力机制已成为很多任务中优秀的序列建模和转导模型中不可分割的一部分，允许对依赖项进行建模而无需考虑它们在输入或输出序列中的距离<sup>[2, 18]</sup>。除了少数情况<sup>[26]</sup>外，注意力机制与循环网络广泛地结合使用。

在这项工作中，我们提出了 Transformer，这是一种避免重复，反之完全依赖注意力机制来捕获输入和输出之间的全局依赖关系的模型架构。在八个 P100 GPU 上经过短短 12 小时的训练后，Transformer 可以实现更大程度的并行化，并且可以在翻译质量上达到一个新的水平。

## 2 背景

减少序列处理计算量的目标也是扩展神经 GPU<sup>[22]</sup>、ByteNet<sup>[17]</sup>和 ConvS2S<sup>[9]</sup>的基础，所有这些都使用卷积神经网络作为基本构建块，并行计算所有输入和输出位置的隐藏表示。在这些模型中，关联来自两个任意输入或输出位置的信号所需的操作数量随着位置之间的距离而增长，例如 ConvS2S 呈线性增长，而 ByteNet 则呈对数增长。这使得学习远距离位置之间的依赖关系变得更加困难<sup>[12]</sup>。在 Transformer 中，这被减少到一个常数级的操作次数，尽管由于计算平均注意力加权位置值而降低了有效分辨率，我们使用多头注意力来抵消这种影响，如第 3.2 节所述。

自注意力，有时称为内注意力，是一种将单个序列的不同位置关联起来以计算序列表示的注意力机制。自注意力已成功用于各种任务，包括阅读理解、文本摘要、文本蕴涵和学习任务无关的句子表示<sup>[4, 26, 27, 21]</sup>。

然而，据我们所知，Transformer 是第一个完全依赖自注意力机制来计算其输入和输出表示而不使用循环或卷积的转换模型。在接下来的部分中，我们将介绍 Transformer，说明自注意力并讨论其相对于[16、17]和[9]等模型的优势。

## 3 模型架构

大多数有竞争力的神经序列转导模型具有编码器-解码器结构<sup>[5, 2, 33]</sup>。这里，编码

器将符号表示的输入序列  $(x_1, \dots, x_n)$  映射到一个连续表示的序列  $z = (z_1, \dots, z_n)$ 。编码得到  $z$  之后，解码器一次生成一个元素的符号输出序列  $(y_1, \dots, y_m)$ 。在每个步骤中，模型都是自回归的<sup>[10]</sup>，在生成下一个符号时将先前生成的符号用作附加输入。

Transformer 遵循这种整体架构，对编码器和解码器使用堆叠的自注意力和逐点全连接层，分别如图 1 的左半部分和右半部分所示。

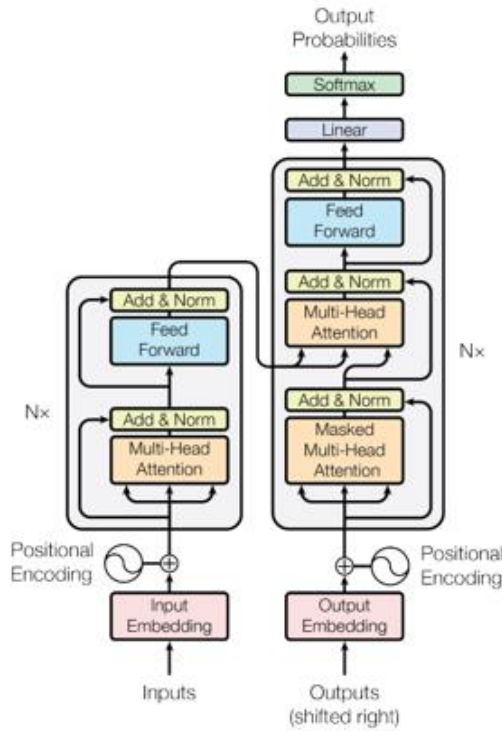


图 1 Transformer 模型架构

### 3.1 编码器和解码器堆栈

**编码器：**编码器由  $N = 6$  个相同层的堆栈组成。每层有两个子层。第一个是多头自注意力机制，第二个是简单的位置全连接前馈网络。我们在两个子层的两端都使用残差连接<sup>[11]</sup>，然后进行层归一化<sup>[1]</sup>。也就是说，每个子层的输出为  $LayerNorm(x + Sublayer(x))$ ，其中  $Sublayer(x)$  是子层自己实现的函数。为了方便这些残差连接，模型中的所有子层以及嵌入层都会产生维度  $d_{model} = 512$  的输出。

**解码器：**解码器也由  $N=6$  个相同的层组成。除了每个编码器层中的两个子层之外，解码器还插入了第三个子层，该子层对编码器堆栈的输出进行多头注意力处理。与编码器类似，我们在每个子层两端使用残差连接，然后进行层归一化。我们还修改了解码器堆栈中的自注意力子层，以防止一个位置影响它的后继位置。这种掩蔽考虑到了输出的嵌入要偏移一个位置的事实，确保对位置  $i$  的预测只依赖于位置小于  $i$  的已知输出。

## 3.2 注意力

注意力函数可以描述为将一个查询和一组键值对映射到一个输出，其中查询、键、值和输出都是向量。输出计算为值的加权和，其中分配给每个值的权重由查询与相应键的兼容性函数计算。

### 3.2.1 缩放点积注意力

我们将我们的特指的注意力称为“缩放点积注意力”（图 2）。输入由维度为  $d_k$  的查询和键以及维度为  $d_v$  的值组成。我们计算出所有键和查询的点积，将每个结果除以  $\sqrt{d_k}$ ，并应用一个 SoftMax 函数来获得值的权重。

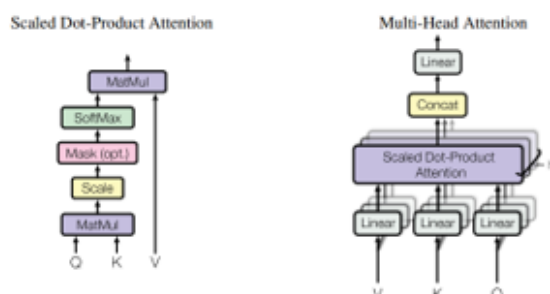


图 2 (左) 缩放点积注意力 (右) 由多个并行运行的注意力层组成的多头注意力

在实际计算过程中，我们用注意力函数计算一组查询的同时把查询打包到一个矩阵  $Q$  中。键和值也打包到矩阵  $K$  和  $V$  中。我们计算出的输出矩阵如下：

$$Attention(Q, K, V) = \text{soft max}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

通常两个最常用的注意力函数是加注意力<sup>[2]</sup>和点积（乘）注意力。这里除了  $\frac{1}{\sqrt{d_k}}$  的比例因子，点积注意力和我们的算法完全是相同的。加注意力使用具有单个隐藏层的前馈网络计算兼容性函数。虽然两者在理论上的复杂度相似，但点积注意力在实际计算中更快且更节省空间，因为它可以使用高度优化的矩阵乘法代码来实现。

尽管对于较小的  $d_k$  值来说，这两种机制的性能相差不大，但加注意力还是优于点积注意力的，因为它无需对较大的  $d_k$  值进行缩放<sup>[3]</sup>。我们猜测，对于较大的  $d_k$  值，点积的结果增幅会很大，从而使得 SoftMax 函数落入具有极小梯度的区域。为了抵消这种影响，我们将点积缩放  $\frac{1}{\sqrt{d_k}}$ 。

### 3.2.2 多头注意力

与使用  $d_{model}$  维键、值和查询执行单个注意力函数不同，我们发现将查询、键和值经过  $h$  次不同的线性映射之后，能分别训练得到  $d_k$ 、 $d_k$  和  $d_v$  维度的线性投影。然后，在每个查询、键和值的投影版本上，我们并行执行注意力函数，得到  $d_v$  维的输出值。这些值被连接起来并再次投影，输出最终值，如图 2 所示。

多头注意力允许模型共同关注来自不同位置的不同子空间的信息。使用单个注意力头的话，平均化会解决这种问题。

$$MultiHead(Q, K, V) = \text{Concat}(head_1, \dots, head_h)W^O \quad (2)$$

$$\text{其中 } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (3)$$

其中投影是参数矩阵  $W_i^Q \in R^{d_{model} \times d_k}$ ,  $W_i^K \in R^{d_{model} \times d_k}$ ,  $W_i^V \in R^{d_{model} \times d_v}$  和  $W^O \in R^{hd_v \times d_{model}}$ 。

在这项工作中，我们使用  $h=8$  个并行注意力层或头。对于其中的每一个注意力层，我们所用的参数是  $d_k = d_v = d_{model} / h = 64$ 。由于每个头的维度减少，总计算成本和具有全维度的单头注意力差不多。

### 3.2.3 注意力在我们模型上的应用

Transformer 以三种不同的方式使用多头注意力：

- 在“编码器-解码器注意力”层中，查询来自前一个解码器层，而记忆键和值来自编码器的输出。这允许解码器中的每个位置参与输入序列中的所有位置。这模仿了序列到序列模型中典型的编码器-解码器注意力机制，例如[36, 2, 9]。

- 编码器包含自注意力层。在自注意力层中，所有的键、值和查询都来自同一个地方，在这个案例中，来自编码器中前一层的输出。编码器中的每个位置都可以关注编码器上一层中的所有位置。

- 类似地，解码器中的自注意力层允许解码器中的每个位置关注解码器中当前位置之前包括自己的所有位置。我们需要防止解码器中的信息向左流动，以保持自回归特性。我们在缩放的点积注意力中实现了这一点，就是屏蔽掉（设置为  $-\infty$ ）SoftMax 输入中与非法连接相对应的所有值。参见图 2。

## 3.3 逐点前馈神经网络

除了注意力子层之外，我们的编码器和解码器中的每一层都包含一个完全连接的前馈网络，它分别相等地应用于每个位置。它包括两个线性变换，采用 ReLU 激活函数。

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (4)$$

虽然线性变换在不同位置上是相同的，但它们在层与层之间使用不同的参数。另

一种描述方式是可以看成卷积核大小 1 的两个卷积。输入输出的维度为  $d_{model} = 512$ ，内层的维数为  $d_{ff} = 2048$ 。

在附录中，我们介绍了位置前馈网络如何也可以被视为一种注意力形式。

### 3.4 嵌入和 Softmax

与其他序列转导模型类似，我们使用训练形成的嵌入将输入和输出标记表示为维度为  $d_{model}$  的向量。我们还使用常用的训练形成的线性变换和 softmax 函数将解码器输出转换为预测的下一个标记的概率。在我们的模型中，我们在两个嵌入层和 pre-softmax 线性变换之间共享相同的权重矩阵，类似于[29]。在嵌入层中，我们将这些权重乘以  $\sqrt{d_{model}}$ 。

### 3.5 位置编码

虽然我们的模型不包含递归和卷积，为了让模型利用序列的顺序，我们必须注入一些关于序列中记号间的相对或绝对位置的信息。为此，我们在编码器和解码器堆栈底部的输入嵌入中添加“位置编码”。位置编码与嵌入具有相同的维度  $d_{model}$ ，因此可以相加。位置编码有很多选择，可学习的和固定的<sup>[9]</sup>。

在这项工作中，我们使用不同频率的正弦和余弦函数：

$$PE(pos, 2i) = \sin(pos / 10000^{2i} / d_{model}) \quad (5)$$

$$PE(pos, 2i+1) = \cos(pos / 10000^{2i} / d_{model}) \quad (6)$$

其中  $pos$  是位置， $i$  是维度。也就是说，位置编码的每个维度都符合正弦曲线。波长形成从  $2\pi$  到  $10000 \cdot 2\pi$  的几何级数。我们选择这个函数是因为我们假设它可以让模型更容易学习到位置关系，对于任何固定的偏移量  $k$ ， $PE_{pos+k}$  都可以表示为  $PE_{pos}$  的

线性函数。

我们还尝试使用学习的位置嵌入<sup>[9]</sup>，发现这两种方式产生了几乎相同的结果。我们选择了正弦版本，因为它可以让模型推出比训练期间遇到的序列长度更长的序列。

## 4 为什么使用自注意力

在本节中，我们将自注意力层的各个方面与循环层和卷积层进行比较，它们通常用于将一个可变长度的符号表示序列  $(x_1, \dots, x_n)$  映射到另一个等长序列  $(z_1, \dots, z_n)$ ，其中  $x_i, z_i \in \mathbb{R}^d$ ，例如典型序列转导编码器或解码器中的隐藏层。为了激发我们对自注意力的使用，我们考虑了三个必要条件。

一是每层的总计算复杂度。另一个是可并行化的计算量，以所需的最小顺序操作数来衡量。

第三是网络中远程依赖关系之间的路径长度。学习长程依赖关系是许多序列转导任务中的难点。影响学习这种依赖关系能力的一个关键因素是前向和后向信号必须在网络中遍历的路径长度。输入和输出序列中任意位置组合之间的路径越短，就越容易学习到远程依赖关系<sup>[12]</sup>。因此，我们还比较了由不同类型的层组成的网络中任意两个输入和输出位置之间的最大路径长度。

如表 1 所示，自注意力层连接所有位置只需常数级的顺序执行操作，而循环层需要  $O(n)$  级的顺序操作。在计算复杂度方面，当序列长度  $n$  小于表示维数  $d$  时，自注意力层比循环层更快，通常机器翻译中使用句子表示的先进模型时都是如此，例如 word-piece<sup>[36]</sup>和 byte-pair<sup>[30]</sup>的表示方法。为了提高涉及很长序列的任务的计算性能，自注意力可限制为仅考虑输入序列中以各自的输出位置为中心，大小为  $r$  的邻域。这会将最大路径长度增加到  $O(n/r)$ 。我们计划在未来的工作中进一步研究这种方法。

卷积核宽度  $k < n$  的单个卷积层不会连接所有输入和输出位置对。我们需要堆叠  $O(n/k)$  的连续核卷积层，或者需要  $O(\log_k(n))$  的空洞卷积<sup>[17]</sup>，从而增加网络中任意两个位置之间最长路径的长度。由于  $k$  卷积层通常比循环层开销更大。然而，可分离卷



积<sup>[6]</sup>将复杂度大大降低到  $O(k \cdot n \cdot d + n \cdot d^2)$ 。即使  $k = n$ ，可分离卷积的复杂度也等于自注意力层和逐点前馈层的组合，这是我们在模型中采用的方法。

表 1 不同层类型的最大路径长度、每层复杂度和最小顺序操作数。 $n$  是序列长度， $d$  是表示维度， $k$  是卷积的内核大小， $k$  是受限自注意中的邻域大小。

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

作为附带的好处，自注意力可以产生更多可解释的模型。我们检查了模型中的注意力分布，并在附录中展示和讨论了样本。不仅个别注意力头能清楚的学习执行不同的任务，许多注意力头似乎表现出与句子的句法和语义结构相关的行为。

## 5 训练

本节介绍我们模型的训练机制。

### 5.1 训练数据和批处理

我们在 WMT 2014 标准的英语转德语数据集上进行了训练，该数据集由大约 450 万个句子对组成。句子使用 byte-pair<sup>[3]</sup>进行编码，该编码具有大约 37000 个标记的共享源目标词汇。对于英语转法语，我们使用了更大的 WMT 2014 英语转法语数据集，该数据集由 3600 万个句子组成，并将标记拆分为 32000 个词条词汇<sup>[36]</sup>。句子对按近似的句子长度分批在一起。每个训练批次包含一组句子对，其中包含大约 25000 个源标记和 25000 个目标标记。

表 2：在 2014 年英语到德语和英语到法语的 newstest 测试中，Transformer 的 BLEU 分数比之前最先进的模型更好，而训练成本只是其中的一小部分

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [17]	23.75			
Deep-Att + PosUnk [37]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [36]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [31]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [37]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [36]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	<b><math>3.3 \cdot 10^{18}</math></b>	
Transformer (big)	<b>28.4</b>	<b>41.0</b>	$2.3 \cdot 10^{19}$	

## 5.2 硬件和时间表

我们在一台带有 8 个 NVIDIA P100 GPU 的机器上训练我们的模型。对于我们整篇论文中描述使用超参数的基础模型，每个训练步骤大约需要 0.4 秒。我们对基础模型进行了总共 100,000 步 12 小时的训练。对于我们的大型模型，（在表 3 的最后一行进行了介绍），每步花费 1 秒。大型模型训练了 300,000 步（3.5 天）。

表 3：Transformer 架构的变体。未列出的值与基本模型的值相同。所有指标都在英语到德语的翻译开发集，newstest2013。根据我们的字节对编码，列出的复杂度是每个单词的，不应与每个单词的复杂度进行比较。

	$N$	$d_{\text{model}}$	$d_{\text{ff}}$	$h$	$d_k$	$d_v$	$P_{\text{drop}}$	$\epsilon_{\text{ls}}$	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)				16						5.16	25.1	58
				32						5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
							0.0			5.77	24.6	
(D)							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)		positional embedding instead of sinusoids								4.92	25.7	
big	6	1024	4096	16			0.3		300K	<b>4.33</b>	<b>26.4</b>	213

### 5.3 优化器

我们使用 Adam 优化器<sup>[19]</sup>，其中  $\beta_1 = 0.9$ ， $\beta_2 = 0.98$  和  $\epsilon = 10^{-9}$ 。我们在训练过程中改变学习率，符合下面的公式：

$$lrate = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_step^{-1.5}) \quad (7)$$

这相当于在第一个  $warmup\_steps$  训练步骤中线性增加学习率，然后根据步数的平方根的倒数按比例减少学习率。我们使用的参数是  $warmup\_steps = 4000$ 。

### 5.4 正则化

我们在训练过程中采用了三种正则化：

**Residual Dropout** 我们将 dropout<sup>[32]</sup>应用于每个子层的输出，然后将其添加到子层输入并进行归一化。此外，我们对编码器和解码器堆栈中嵌入和位置编码的总和进行了 dropout。在基本模型中，我们使用  $P_{drop} = 0.1$  的比率。

**Attention Dropout** 对关键注意力的查询在结构上类似于前馈网络中的隐藏到隐藏权重，尽管是跨位置的。然后将产生注意力权重的 SoftMax 激活函数视为隐藏层激活函数的类似物。一种自然的可能想法是将 dropout<sup>[32]</sup>扩展到注意力机制。我们通过丢弃注意力权重来实现注意力 dropout，

$$Attention(Q, K, V) = dropout(\text{softmax}(\frac{QK^T}{\sqrt{d}}))V \quad (8)$$

除了残差 dropout，我们发现注意力 dropout 对我们的解析实验是有益的。

**Label Smoothing** 在训练过程中，我们采用了值为  $\epsilon_{ls} = 0.1$ <sup>[34]</sup>的标签平滑。难以理解的是，模型会变得更加不确定，但却提高了准确性和 BLEU 分数。

## 6 结果

## 6.1 机器翻译

在 WMT 2014 英德翻译任务中，我们的大 transformer 模型（表 2 中的 Transformer (big)）比之前报道的最佳模型（包括集成）高出 2.0 BLEU 以上，创立了一个新的最好的 28.4 的 BLEU 分数。该模型的配置列于表 3 的最后一行。在 8 个 P100 GPU 上训练耗时 3.5 天。甚至我们的基础模型也超过了所有先前发布的模型和集成，其训练开销仅为之前任何最佳模型的一小部分。

在 WMT 2014 英法翻译任务中，我们的大模型的 BLEU 得分为 41.17，优于之前发布的所有单一模型，训练开销不到之前最先进模型的 1/4。为英语转法语训练的 Transformer（大）模型不共享源嵌入和目标嵌入， $d_{ff} = 8192$ ，并且使用 dropout 率  $P_{drop} = 0.1$ ，而不是 0.3。

对于基本模型，我们使用的单个模型来自最后 5 个检查点的平均值，这些检查点以 10 分钟的间隔写入。对于大型模型，我们平均了最后 20 个检查点。我们使用束搜索，束大小为 4，长度惩罚  $\alpha = 0.6$  [36]。这些超参数是在对开发集进行实验后选择的。我们将推断期间的最大输出长度设置为输入长度 +50，在可能的情况下提前终止 [36]。

表 2 总结了我们的结果，并将我们模型的翻译质量和训练开销与文献中的其他模型架构进行了比较。我们通过将训练时间、使用的 GPU 数量和每个 GPU 的持续单精度浮点容量的估计值相乘来估计用于训练模型的浮点运算的数量。

## 6.2 模型变化

为了评估 Transformer 不同组件的重要性，我们以不同的方式改变我们的基础模型，测量开发集 newstest 2013 上英德翻译性能的变化。我们使用了上一节中描述的束搜索，但没有平均检查点。我们在表 3 中展示了这些结果。

在表 3 行(A)中, 我们改变了注意力头的数量以及注意力键和值的维度, 保持计算量不变, 如第 3.2.2 节所述。虽然单头注意力比最佳设置差 0.9 BLEU, 但质量也会随注意力头过多而下降。

在表 3 行(B)中, 我们观察到减小注意力键大小  $d_k$  会损害模型质量。这表明确定兼容性并不容易, 且比点积更复杂的兼容性函数可能是有益的。我们在行(C)和(D)中进一步观察到, 正如预期的那样, 更大的模型更好, 并且 dropout 对于避免过度拟合很有帮助。在第(E)行中, 我们用学习到的位置嵌入<sup>[9]</sup>替换我们的正弦位置编码, 并观察到与基本模型几乎相同的结果。

### 6.3 英语句法成分解析

为了评估 Transformer 是否可以泛化到其他任务, 我们对英语句法成分分析进行了实验。这项任务提出了具体的挑战: 输出有严格的结构约束, 并且明显长于输入。此外, RNN 序列到序列模型无法在小数据体系中获得最好的结果<sup>[35]</sup>。

我们在 Penn Treebank<sup>[24]</sup>的华尔街日报(WSJ)部分训练了一个  $d_{model}=1024$  的 4 层转换器, 大约 4 万条训练句子。我们还在半监督环境中对其进行了训练, 使用有大约 1700 万个句子<sup>[35]</sup>的高置信度 BerkeleyParser 语料库。我们将 1.6 万条标记的词汇表用于 WSJ 唯一设置, 将一个 3.2 万条标记的词汇表用于半监督设置。

我们只进行了少量实验来选择 dropout 率、注意力和残差 (第 5.4 节)、学习率和集束大小, 所有其他参数与英德基础翻译模型保持一致。在推断过程中, 我们将最大输出长度增加到输入长度 +300。我们对 WSJ 和半监督设置都使用了 21 的集束大小,  $\alpha=0.3$ 。

我们在表 4 中的结果表明, 尽管缺乏针对特定任务的调整, 但我们的模型表现依然非常好, 比除递归神经网络语法<sup>[8]</sup>之外的所有先前报告过的模型取得了更好的结果。

与 RNN 序列到序列模型<sup>[35]</sup>相比, 即使仅在 WSJ 40K 句子的训练集上进行训练, Transformer 也优于 BerkeleyParser<sup>[28]</sup>。

表 4: Transformer 很好地推广到英语选区解析（结果在《华尔街日报》第 23 节）

Parser	Training	WSJ 23 F1
Vinyals & Kaiser et al. (2014) [35]	WSJ only, discriminative	88.3
Petrov et al. (2006) [28]	WSJ only, discriminative	90.4
Zhu et al. (2013) [38]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [38]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [25]	semi-supervised	92.1
Vinyals & Kaiser et al. (2014) [35]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Dyer et al. (2016) [8]	generative	93.3

## 7 结论

在这项工作中，我们提出了 Transformer，这是第一个完全基于注意力的序列转导模型，用多头自注意力取代了编码器-解码器架构中最常用的循环层。

对于翻译任务，Transformer 的训练速度明显快于基于循环或卷积层的架构。在 WMT 2014 英语转德语和 WMT 2014 英语转法语的翻译任务上，我们都取得了最好的成绩。在前一项任务中，我们最好的模型甚至优于所有先前报道过的集成模型。我们还通过英语句法成分分析实验提供了我们模型更广泛适用性的示例。

我们对基于注意力的模型的未来感到兴奋，并计划将它们应用到其他任务中。我们计划将 Transformer 扩展到除文本以外的涉及输入和输出模式的问题，并研究局部的受限注意力机制，以有效处理图像、音频和视频等大型输入和输出。减少生成的顺序性是我们的另一个研究目标。

我们用于训练和评估模型的代码可在 <https://github.com/tensorflow/tensor2tensor> 获得。

## 参考文献

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly

- learning to align and translate. CoRR, abs/1409.0473, 2014.
- [3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. CoRR, abs/1703.03906, 2017.
  - [4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733, 2016.
  - [5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. CoRR, abs/1406.1078, 2014.
  - [6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357, 2016.
  - [7] Junyoung Chung, Çağlar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555, 2014.
  - [8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In Proc. of NAACL, 2016.
  - [9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. arXiv preprint arXiv:1705.03122v2, 2017.
  - [10] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.
  - [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.
  - [12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.
  - [13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.
  - [14] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 832–841. ACL, August 2009.

- [15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410, 2016.
- [16] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In International Conference on Learning Representations (ICLR), 2016.
- [17] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. arXiv preprint arXiv:1610.10099v2, 2017.
- [18] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In International Conference on Learning Representations, 2017.
- [19] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015.
- [20] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. arXiv preprint arXiv:1703.10722, 2017.
- [21] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130, 2017.
- [22] Samy Bengio Łukasz Kaiser. Can active memory replace attention? In Advances in Neural Information Processing Systems, (NIPS), 2016.
- [23] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attentionbased neural machine translation. arXiv preprint arXiv:1508.04025, 2015.
- [24] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. Computational linguistics, 19(2):313–330, 1993.
- [25] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In Proceedings of the Human Language Technology Conference of the NAACL, Main Conference, pages 152–159. ACL, June 2006.
- [26] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In Empirical Methods in Natural Language Processing, 2016.
- [27] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304, 2017.
- [28] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and



- interpretable tree annotation. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 433–440. ACL, July 2006.
- [29] Ofir Press and Lior Wolf. Using the output embedding to improve language models. arXiv preprint arXiv:1608.05859, 2016.
- [30] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909, 2015.
- [31] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538, 2017.
- [32] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [33] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.
- [34] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.
- [35] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2015.
- [36] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016.
- [37] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016.
- [38] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 434–443. ACL, August 2013.



# 视觉对话综述

**摘要** 视觉对话是深度学习的一项重要研究任务，是将自然语言处理和计算机视觉相结合的跨学科研究的重要问题之一。视觉对话是针对图像的持续会话，要求目标和语义关联随着对话动态调整。视觉对话要求机器理解视觉和文本两种模态的信息，具有相当的挑战性和实用的前景。本文综述近年来视觉对话的研究发展进程，总结了目前主流的模型和算法，最后提出了视觉对话的发展趋势和方向。

**关键词** 视觉对话；深度学习；视觉注意力；图结构

## 1 引言

随着自然语言处理和计算机视觉研究的发展，计算机感知和认知能力的不断提高，同时处理视觉和语言两种甚至更多模态的人工智能任务得到了广泛关注，如视觉对话（Visual Dialogue）、图像描述生成（Image Captioning）和视觉问答（VQA）等。不同于视觉问答任务，视觉对话要求计算机回答被一系列关于先前对话和给定图像的问题，可以看作多轮的视觉问答，计算机除了理解图像和问题之外，还需结合历史对话信息进行推理，因为随着对话的进行，与问题主题相关的视觉内容也会发生变化，相对于其他视觉语言任务更具挑战性。视觉对话在不同的领域中有广泛的应用前景。例如，可以帮助视觉障碍用户理解他们周围的环境或是社交媒体的内容。可以帮助游戏分析师根据大量的赛事数据做出合理的决策。可以应用于机器人的搜索或救援等任务，救援人员处于“情境盲区”，可以通过语言来和机器人进行交互，机器人根据自然语言指令，在现场执行命令并反馈，在根据后续语言命令进行操作，在这个过程中，救援人员既能保障自身安全，又可了解场景情况。

本文总结了近年来视觉对话的研究进展，分析了现有的模型和方法，讨论了视觉对话的研究难点和未来的发展方向。

## 2 概述

### 2.1 两种模型

视觉对话有两种模型,生成式模型 (Generative LSTM) 和判别式模型 (Discriminative softmax)<sup>[1]</sup>。判别式模型对候选答案集合进行相关性排序,每个候选答案有一个相关性分数,选择最大值的候选作为目标答案。判别式模型输出候选答案上的概率分布,选择概率大的作为输出答案。该模型采用 encoder-decoder 的基本模型,大致可以分为 3 类:

- i. 后期融合 (Late Fusion, LF)<sup>[1]</sup>分别提取图像、问题和历史的特征,再进行特征拼接作为多模态特征。
- ii. 层次递归编码 (Hierarchical Recurrent Encoder, HRE)<sup>[1]</sup>并行提取图像、问题和历史的特征,将 3 个特征通过层次 LSTM 进行融合,作为多模态特征。
- iii. 记忆网络 (Memory Network, MN)<sup>[1]</sup>为对话历史设置一个记忆缓存,保存对话历史中的一个问答对的编码结果,作为一条事实储存在记忆缓存中,新问题通过检索记忆缓存获取信息,作为解码器的输入。

生成式模型的答案并不来源于候选答案的集合,而是由模型生成自然语言句。

### 2.2 数据集

视觉对话任务的常用数据集为 VisDial<sup>[1]</sup>。对于数据集中的每张图片而言,提问者只能看到标题和对话历史,而回答者可以看到标题、历史和图像。每张图片的对话由 10 轮问答组成。VisDial 分为 v0.9 和 v1.0 两个版本。v0.9 分为训练集和验证集两个子集。其中训练集包含 8.3 万个对话,验证集包含 4 万个对话。v1.0f 则分为训练集、验证集和测试集 3 个子集。其中,训练集由 v0.9 的全部数据组成,验证集和测试集基于 Flickr 的图像生成。验证集包含 2000 个对话,测试集包含 8000 个对话。

除了 VisDial 之外, GuessWhat<sup>[2]</sup>数据集也常用于视觉对话任务。提问者和回答者均可以看到图像,回答者被随机告知图像中的一个目标对象,提问者提出一系列以“是”

或“否”为答案的问题，从而猜出该目标对象。

### 3 研究方法分类

#### 3.1 基于注意力机制的模型

在视觉对话任务中，计算机需要确定问题的语义意图，同时需要在异构模态输入中对齐与问题相关的文本和视觉内容。这种视觉和语言的细粒度交互可以通过注意力机制来实现。基于注意力机制的模型包括基于注意力的层次递归编码模型（Hierarchical Recurrent Encoder with Attention, HERA）<sup>[1]</sup>、记忆网络，多视图注意力网络（Multi-View Attention Network, MVAN）<sup>[3]</sup>，基于历史的图像注意力编码模型（History-Conditioned Image Attentive Encoder, HCIAE）<sup>[4]</sup>和联合注意力模型（Co-Attention, CoAtt）<sup>[5]</sup>。具体而言，HERA, MN 和 HCIAE 提取图像或问题的特征，结合对话历史进行注意力加权计算。CoAtt 则是序列化不同来源的输入。而 MVAN 利用两个互补模块（主题聚合和上下文匹配）捕获对话历史中的相关信息，并通过模态对齐来构建多模态表示。

#### 3.2 基于图的模型

随着深度学习对非欧氏数据的逐渐重视，图结构和图神经网络也得到了快速的发展。由于传统的注意力机制难以捕捉两个模态间单个实体的语义关系，而图神经网络在处理结构化数据方面展现出较强的优势，使用图神经网络来提取多模态特征的方法也越来越多的被采纳。基于图结构的模型包括图神经网络（Graph Neural Network, GNN）<sup>[6]</sup>、因子图注意力模型（Factor Graph Attention, FGA）<sup>[7]</sup>、对偶视觉对话模型（DualVD）<sup>[8]</sup>、因果图结构模型<sup>[9]</sup>和知识桥图网络<sup>[10]</sup>等。其中，GNN 将多轮对话表示为图结构，通过图结构计算当前问答的特征表示，节点则通过消息传递机制来进行状态更新。FGA 对图结构的实体两两建立关系。DualVD 提取物体间的关系特征。KBGN 采用模块化的流程设计，对视觉-文本间潜在的语义关联进行细粒度地建模。因果图结构则是认为视觉对话并非只是带有历史的 VQA，从因果推断的角度添加因子（用户偏好）来构建真实世界的因果图，切断了对话历史和答案间的直接因果效应。

### 3.3 基于视觉指代消解的模型

主流的视觉指代消解的模型有神经模块网络（CorefNMN）<sup>[11]</sup>、自适应视觉记忆网络（AVMN）<sup>[12]</sup>、递归视觉注意力模型（Recursive Visual Attention, RvA）<sup>[13]</sup>、注意力记忆模型（Attention Memory, AMEM）<sup>[14]</sup>和循环对偶注意力网络（Recurrent Dual Attention Network, ReDAN）<sup>[15]</sup>等。具体而言，CoreNMN 结合符号化计算和神经网络，将指代词与目标视觉对象相关联，实现单词级的视觉指代消解。AVMN 直接将视觉信息存储于外部记忆库，整合文本和视觉定位过程，进而有效缓解了在这两个过程中所产生的误差。RvA 递归地查看与主题相关的对话历史并细化视觉注意力。直到对视觉基础感到“自信”，或已回溯到对话历史的开头。AMEM 对视觉注意力进行记忆存储，并根据当前问题对其加权，在句子层面进行视觉指代消解。ReDAN 使用多步推理来优化视觉注意力与多模态特征。

## 4 研究发展方向

目前，视觉对话通常采用判别式模型，易于评测，但是实际对话中，给定范围内的答案不一定满足用户的需求，而生成式模型能生成多样化的答案，对生成式模型的研究仍具有较大的空间。此外，视觉对话已不再被看作为仅带有历史的 VQA，通过推理机制可以更好的提高视觉对话的准确性，这也将成为未来的一个研究方向。

### 参考文献

- [1] Das Abhishek, Kottur Satwik, Gupta Khushi, Singh Avi, Yadav Deshraj, Lee Stefan, Moura Jose, Parikh Devi, Batra Dhruv. Visual Dialog[J]. IEEE transactions on pattern analysis and machine intelligence, 2018, 41(5): 1242-1256.
- [2] DE VRIES H, STRUB F, CHANDAR S, et al. Guesswhat?! visual object discovery through multi-modal dialogue[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 5503-5512.
- [3] Park Sungjin, Whang Taesun, Yoon Yechan, Lim Heuseok. Multi-View Attention Network for Visual Dialog[J].

- Applied Sciences, 2021, 11(7).
- [4] LU J, KANNAN A, YANG J, et al. Best of both worlds: transferring knowledge from discriminative learning to a generative visual dialog model[C]// Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017: 313-323.
- [5] WU Q, WANG P, SHEN C, et al. Are you talking to me? reasoned visual dialog generation through adversarial learning[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018:6106-6115.
- [6] ZHENG Z, WANG W, QI S, et al. Reasoning visual dialogs with structural and partial observations[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019:6669-6678.
- [7] SCHWARTZ I, YU S, HAZAN T, et al. Factor graph attention[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019:6463-6474.
- [8] JIANG X, YU J, QIN Z, et al. DualVD: An Adaptive Dual Encoding Model for Deep Visual Understanding in Visual Dialogue[C]// AAAI. 2020, 1(3):5.
- [9] Qi J., Niu Y., Huang J., Zhang H.. Two causal principles for improving visual dialog[C]. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2020.
- [10] Jiang, X., Du, S., Qin, Z., Sun, Y., & Yu, J. (2020). KBGN: Knowledge-Bridge Graph Network for Adaptive Vision-Text Reasoning in Visual Dialogue. Proceedings of the 28th ACM International Conference on Multimedia.
- [11] KOTTUR S, MOURA J M F, PARIKH D, et al. Visual coreference resolution in visual dialog using neural module networks[C]// Proceedings of the European Conference on Computer Vision(ECCV). 2018:153-169.
- [12] 赵磊, 高联丽, 宋井宽. 面向视觉对话的自适应视觉记忆网络[J]. 电子科技大学学报, 2021, 50(05):749-753.
- [13] Yulei Niu, Hanwang Zhang, Manli Zhang, Jianhong Zhang, Zhiwu Lu\*, and Ji-Rong Wen, Recursive Visual Attention in Visual Dialog, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, USA, 2019.
- [14] SEO P H, LEHRMANN A, HAN B, et al. Visual reference resolution using attention memory for visual dialog[C]// Advances in Neural Information Processing Systems. 2017:3719-3729.
- [15] Zhe Gan, Yu Cheng, Ahmed El Kholy, Linjie Li, Jingjing Liu, Jianfeng Gao. Multi-step Reasoning via Recurrent Dual Attention for Visual Dialog.[J/OL]. CoRR, 2019, abs/1902.00579.





## 苏州大学本科生毕业设计（论文）中期进展情况检查表

学院（部）：计算机科学与技术学院

学生姓名		年级		专业	计算机科学与技术	填表日期	
设计（论文）选题	XXXXXXXXXX 研究和实现						
已完成的任务	完成了任务书，外文翻译和文献综述，基本了解了主流模型框架的特点，复现了部分主流模型。						
	是否符合任务书要求进度	是					
尚须完成的任务	对主流 visual dialog 模型的优化改进，评估改进后的模型						
	能否按期完成任务	是					
存在的问题和解决	存在的问题	部分模型缺少特征集等必要数据，无法复现。模型数据集过于庞大，外网下载缓慢。自己电脑性能差，跑代码时间太长，云服务器按月计费太贵，只能按量计费，但每次开始实例都要配置一次很麻烦。					
	拟采取的办法	在主流的各种大类的模型框架中挑选代表性的模型进行复现。寻找好用点的云平台。					
指导教师意见	进展正常。  签名：						
中期检查专家组意见	同意  组长签名：						
学院（部）意见	同意  教学院长签名：						

检查日期： 年 月 日



## 苏州大学本科生毕业设计（论文）答辩记录表

学院（部）：计算机科学与技术学院

学生姓名		年 级		专 业	
设计（论文）题目	XXXXXXXXXXXXXXXXXXXX 研究和实现				
答辩时间			答辩地点		
答辩小组成员：					
<p>答辩中提出的主要问题及学生回答问题的简要情况：</p> <p>Q1:损失函数的改进是否存在问题，出在理论上还是实践上？  A1:改进损失函数后学习更慢，且只有中前期有优势，问题在于实践上，一是对原模型的复现与原论文结果有一定差异，主要尤其原模型缺少部分参数，例如 <code>random_seed</code> 的值；二是模型更改后没有对学习率的更新策略做一个调整；三是针对改进后的损失函数难以收敛的情况，一般添加一个 <code>lambda</code> 参数来平滑，后续需要通过这种方法来改进。</p> <p>Q2:学习率是怎么调整的，为什么这样调整，训练后期是学习率根据训练情况下降是怎么调整的？  A2:学习率先增大再减小，这样调整可以减小振幅。学习率的调整方式沿用原模型的方法，主要通过函数更新乘数，具体细节没有仔细研究，需后续学习。</p> <p>Q3:在位置编码添加上，提到 LSTM 记忆时间只有 100s，是否是指训练时间，如果是，那么不同设备是否结果是否相差很大？  A3:是指训练时间，不同设备结果确实有差异（但本模型训练一批不用 100s）</p> <p>Q4:数据集提问者看不到图像，那么怎么提问题呢？  A4:从图像是什么样的，是什么颜色的，图像上有什么来开始提问，然后再对图像具体对象进行提问。</p> <p>答辩小组组长签名：</p> <p>答辩小组成员签名：</p> <p>记录人签名：</p> <p style="text-align: right;">年 月 日</p>					

## 苏州大学本科生毕业设计（论文）成绩评定表



## 苏州大学本科生毕业设计（论文）成绩评定表

学院（部）：计算机科学与技术学院

设计（论文）题目：XXXXXXXXXXXXXXXXXXXX 研究和实现			
姓 名		学 号	
年 级		专 业	
指导教师评语	该同学在毕设过程中，学习态度端正，能够积极主动的与老师进行交流沟通。在较为全面的文献阅读基础上，根据毕设任务要求，设计论文的实验方案。此外，更为重要的是，学生在较为充分的实验基础上，分析总结基模型存在的问题，并提出了可行的改进策略。最后，在扩展的数据集上，验证了改进策略的可行性，同时采取对比分析，验证了提出方法的优越性。体现出学生具有较好的基础理论知识和专业知识，并能综合运用所学知识进行较深入的研究性学习和实践。论文结构层次清晰，文字描述简洁通顺，实验设计合理，实验结果可靠。是一篇优秀的本科毕业设计论文。		
	评 价 内 容		得 分
	设计（论文）方案设计、文献检索、阅读及综述能力、进度等情况评价分（计 25 分）		
	毕业设计（论文）质量和工作量评价分（计 50 分）		
	科学素养、学习态度、纪律表现等情况评价分（计 25 分）		
	成绩（满分 100 分）：      签名：      年      月      日		

苏州大学本科毕业设计（成绩评定表）

评阅教师评语	<p>论文通过研究基于注意力机制的视觉对话模型，复现并改进了基于注意力的双视图网络模型，实现了视觉对话模型架构,并在相对短期的训练安排上取得了较好的结果。</p> <p>论文选题符合专业培养目标，能够达到综合训练目标，题目有一定难度，工作量较为饱满。写作过程中能综合运用所学的理论知识，较为全面分析当前的问题。具有一定的综合运用知识的能力。文章，内容较为完整，层次结构安排较为科学，主要观点突出，逻辑关系清楚，格式基本正确，较好的完成了毕业设计的相关工作。</p>	
	评 价 内 容	得 分
	毕业设计（论文）文字书写评价分（计 20 分）	
	毕业设计（论文）质量评价分（计 40 分）	
	工作量情况评价分（计 20 分）	
	毕业设计（论文）创新及分析问题、解决问题能力评价分（计 20 分）	
成绩（满分 100 分）：      签名：                  年      月      日		
答辩小组评语	<p>XXXX 同学能运用所学理论和专业知识，简明扼要、有重点突出地阐述论文的主要内容，论点基本正确，思路较清晰，语言表达较准确，表现出对所从事的研究内容掌握得较好。回答问题有理论根据，思路较清晰，论点较正确，概念基本清楚，对主要问题回答较正确，按照培养目标，围绕本学科和专业选择有实用价值的、具有所学课程知识、能力训练的题目，较好地完成了任务书所规定的各项要求；论文书写合规范化要求</p>	
	评 价 内 容	得 分
	毕业设计（论文）介绍表达情况评价分（计 20 分）	
	回答问题表现评价分（计 40 分）	
	毕业设计（论文）水平和工作量评价分（计 40 分）	
	成绩（总分 100 分）：      组长签名：                  年      月      日	
按权重折算总成绩		
审定成绩：    答辩委员会主任签名：		
日    期：		

## 文本复制检测报告单

No: <input type="text"/>	
检测文献:	的研究和实现
作者:	
检测范围:	中国学术期刊网络出版总库 中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库 中国重要会议论文全文数据库 中国重要报纸全文数据库 中国专利全文数据库 图书资源 优先出版文献库 大学生论文联合比对库 互联网资源(包含贴吧等论坛资源) 英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等) 港澳台学术文献库 互联网文档资源 CNKI大成编客-原创作品库
时间范围:	1900-01-01至2022-05-18
检测时间:	2022-05-18 15:16:32

总文字复制比: 4.4%			
去除引用: 4.3%	去除本人: 4.4%	重合字数: 1152	文献总字数: 26070
总段落数: [ 6 ]	疑似段落数: [ 5 ]	疑似段落最大重合字数: [ 540 ]	
前部重合字数: [ 361 ]	后部重合字数: [ 791 ]	疑似段落最小重合字数: [ 30 ]	
6%	中英文摘要等(总4461字)		
7.9%	第1章绪论(总2643字)		
10.8%	第2章基础知识概述(总5008字)		
1.8%	第3章基于注意力机制的双视图网络模型及其改进(总5939字)		
0%	第4章模型实现及结果分析(总6846字)		
2.6%	第5章总结(总1173字)		