



本科生毕业论文（设计）

题目：基于视觉编程框架
的人机图像问答

姓 名 徐 浩

学 号 20337256

院 系 计算机学院

专 业 计算机科学与技术

指导教师 王可泽 (教授)

2024 年 5 月 15 日

基于视觉编程框架

的人机图像问答

Visual Question Answering

Based on Visual Programming Framework

姓 名

徐 浩

学 号

20337256

院 系

计算机学院

专 业

计算机科学与技术

指导教师

王可泽 (教授)

2024 年 5 月 15 日

学术诚信声明

本人郑重声明：所呈交的毕业论文（设计），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，本论文（设计）不包含任何其他个人或集体已经发表或撰写过的作品成果。对本论文（设计）的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本论文（设计）的知识产权归属于培养单位。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日 期： 年 月 日

【摘 要】

视觉问答 (VQA) 是一个计算机视觉与自然语言处理的交叉任务，而本文深入分析了基于 2023 年 cvpr 年度 Best Paper 中提出的 VisProg^[1] 视觉编程框架，并研究了它在解决视觉问答 (VQA) 问题上的应用与效能

为了克服传统 VQA 方法在解释性方面的不足，本文使用了 VisProg 中的“程序生成器-程序解释器”框架。程序生成器使用 GPT3 大语言模型解构视觉问题并生成模块化的抽象程序，把我们提出的问题拆解成一连串小任务。而程序解释器则使用多种预训练模型或 Python 函数构建多种模块，依照生成的抽象程序调用模块并逐步执行子任务，同时清晰地给出执行它们的过程解释。整个过程形成体系化的结构，逐步可视化地构建出 VQA 任务答案。

同时为了验证 VisProg 实现 VQA 任务的实际性能，本文将基于 VisProg 的 VQA 方法中的 VQA 模块的性能进行了详细的对比分析。通过这些对比实验，展现 VisProg 它在视觉问答任务上的效能。

最后，本文总结了分析了 VisProg 框架的引入将给 VQA 任务带来的影响，以及其框架结构的未来可能的改进方向。

关键词： 计算机视觉，多模态，视觉问答，图像理解，视觉编程，可解释性

[ABSTRACT]

Visual Question Answering (VQA) is a cross-disciplinary task between computer vision and natural language processing. This article provides an in-depth analysis of the VisProg framework proposed in the Best Paper of CVPR 2023^[1], and investigates its application and effectiveness in solving Visual Question Answering (VQA) problems.

To overcome the interpretability shortcomings of traditional VQA methods, this article employs the "Program Generator-Program Interpreter" framework from VisProg. The program generator utilizes the GPT-3 large language model to deconstruct visual questions and generate modular abstract programs, breaking down the posed questions into a series of sub-tasks. The program interpreter utilizes various pre-trained models or Python functions to construct multiple modules, executing sub-tasks according to the generated abstract program and providing clear process explanations as they execute. The entire process forms a systematic structure, gradually constructing VQA task answers in a visualized manner.

Furthermore, to validate the practical performance of VisProg in VQA tasks, this article conducts a detailed comparative analysis of the performance of the VQA module in the VisProg-based VQA method. Through these comparative experiments, it demonstrates the effectiveness of VisProg in the Visual Question Answering task.

Finally, this article summarizes the impact of introducing the VisProg framework on VQA tasks and potential directions for future improvements in its framework structure.

Keywords: Computer Vision, Multimodal, Visual Question Answering, Image Understanding, Visual Programming, Interpretability

目录

1	绪论	1
1.1	选题背景与意义	1
1.2	国内外研究现状和相关工作	1
1.3	本文的论文结构与章节安排	5
2	VISPROG 框架的分析与实现	7
2.1	VISPROG 基本目标	7
2.2	程序生成器 (Program Generator)	7
2.3	程序解释器 (Program Interpreter)	9
2.4	VISPROG 处理 VQA 任务的分析	11
3	VISPROG 诸多基础模块的实现	12
3.1	定位模块 LOC	12
3.2	计数模块 COUNT	13
3.3	裁剪模块 CROP	13
3.4	EVAL 模块	14
3.5	RESULT 模块	14
3.6	VQA 模块	15
4	几种 VQA 模块组件的实现	16
4.1	ViLT(Vision and Language Transformer) 模块实现	16
4.2	CLIP-ViLT 模块实现	20
4.3	BLIP-VQA 模块实现	23
5	实验与分析	28
5.1	程序生成器中的实验:	
	Prompt 模板数对 VISPROG 在 VQA 问题上的表现的影响	28
5.2	程序生成器实验: 不同 VQA 模块的性能测试	30

6 总结与展望	34
6.1 论文工作总结	34
6.2 VISPROG 的不足与未来改进	34
参考文献	37
致谢	39

插图目录

2.1	VISPROG 的整体框架	7
2.2	大语言模型已经具有了强大的上下文学习能力	8
2.3	VISPROG 中的程序生成过程	8
2.4	VISPROG 目前支持的模块	9
2.5	模块的 demo	10
2.6	关于判断 “At least three animals are in a flowered field.” 问题的可视化视觉推理过程	10
3.1	原图像, 定位目标: “women”	12
3.2	使用定位模块 LOC 标注的图像示例	12
3.3	使用计数模块 COUNT 处理的图像示例, 显示物体数量和相应的边界框。	13
3.4	使用裁剪模块 CROP 处理的图像示例, 显示裁剪 1.5 倍目标区域的 “women”。	14
3.5	EVAL 模块完成 “1+1=2” 的表达式计算的示例	14
3.6	VQA 模块完成对 “what color is the dress” 的解答	15
4.1	不同类型的 VLP 模型 (Vision-Language Pre-training)	16
4.2	图像特征的几类不同提取方式	16
4.3	ViLT 模型的整体结构	17
4.4	ViLT 提取到的图像-文本对	18
4.5	预训练 CLIP 作为 VLP 模型中的视觉编码器	22
4.6	BLIP 的预训练框架	24
4.7	BLIP-VQA 的结构示例	26
5.1	e = 3, Prompt 模板示例	28
5.2	Context examples 对 VISPROG-VQA 任务的 Accuracy 的影响	29
5.3	几种 VQA 模型响应速度对比图	32

6.1	Question: What's so funny about this image?	36
6.2	当遇到较为抽象的问题时, VISPROG 以及其内部的 VQA 模块的表 现不尽人意	36

表格目录

3.1	VISOPROG 中的 VQA 任务中使用的模块和解释器	12
5.1	实验配置	29
5.2	不同 Prompt 策略的模型在 <i>testdev set</i> 上的表现	29
5.3	单独 VQA 模块在 VQA 任务上的性能评估	31
5.4	在 VISPROG 框架中时 VQA 模块性能评估	31
5.5	VQA 模块的推理时间	32

1 绪论

1.1 选题背景与意义

对于计算机来说,学习如何去理解图像是一个漫长的过程。从最早的图像分类 (Image Classification), 目标检测 (Object Detection), 到图像描述 (Image Captioning), 再到如今广受关注的图像问答 (Visual Question Answering)。“观察、联系 -> 推断、预测 -> 理解、想象。” 计算机视觉 (Computer Vision) 在探索智能的道路上不断发展着,并在这个过程中逐渐与自然语言处理 ((Natural Language Processing)) 技术出现高度融合。但是,与 NLP 领域广泛研究的纯基于文本的 QA 系统不同,图像 QA 系统需要根据观察参考图像的内容,却是回答自然语言的问题,这个过程同时有图像与文本的输入,是一个涉及到多模态的问题。

早期,人们提出的图像 QA 模型大多基于对特征“联合嵌入”,比如早年提出的 SAN^[2] 网络结构,它是利用卷积神经网络 (CNN) 提取全局图像特征向量,并用长短期记忆网络 (LSTM) 编码相应问题的特征向量,使用注意力机制,然后将不同特征结合以推断答案。这种想法较为直观,并且已经取得了令人印象深刻的结果。然而,这类模型是如何理解这些特征的过程,对我们来说是不可知的,整个流程缺乏可解释性。而如今的趋向之一便是使用类似于神经模块网络^[3]的方法,试图通过动态的网络结构来模块化图像问答中的推理过程,进而来改善这一点。

在这一研究的影响下,2023 年 VISPROG 框架^[1]应运而生,VISPROG 采用了大语言模型支持处理多样化的输入和模块,展现了强大的泛用性。本文为 VISPROG 框架加入了当前 VQA 任务使用广泛的 VILT, CLIP-VILT 等方法模块,尝试进一步探究并专精完善 VISPROG 框架在 VQA 任务上的潜力,并与传统方法做出对比测试分析。

1.2 国内外研究现状和相关工作

1.2.1 早年的一些 VQA 模型与方法

1) 原始联合嵌入方法

在 2014 年,Malinowski^[4]等人提出了一种名为“Neural-Image-QA^[4]”的模型,它把 VQA 视为一个结合图像信息的序列到序列 (sequence to sequence^[5]) 任务,是早期采用联合嵌入方法的典型例子。该模型通过结合卷积神经网络

(CNN^[6]) 和循环神经网络 (RNN^[7]) 来处理图像和文本数据, 旨在解决如何将图像和文本信息融合到统一的特征空间中, 以实现跨模态理解。这种方法取得了一定的成功, 因为它能够处理一些基本的 VQA 任务。然而, 其局限性在于处理细节不够精细, 难以捕捉图像和问题之间更深层次的语义关系。

2) 基于注意力机制的联合嵌入方法

2015 年, Oriol Vinyals^[8] 等人引入了注意力机制 (Attention Mechanism^[9]), 在他们的论文 “Show, Attend and Tell^[8]”, 模型通过识别图像中的显著区域来生成图像标题, 从而使进一步的处理聚焦于这些区域。这种概念很容易被转化应用到 VQA 任务中, 用于专注于与问题相关的图像区域, 通过更精细地关注图像的特定区域来改进 VQA 模型的性能。例如, Yang 等人提出了堆叠注意力网络 (SAN), 通过迭代地聚焦不同区域来解答问题。Anderson 等人则是在 2018 年提出的自底向上和自顶向下的注意力模型^[7] 利用 Faster R-CNN^[10] 提取图像中对象的特征, 实现了对于图像关键区域的细粒度关注。此外, 共同注意力 (co-attention) 方法, 如 MCAN^[11], 不仅根据问题引导视觉注意力, 同时也依据视觉信息指导文本的注意力聚焦, 从而实现了更深层次的跨模态交互。注意力机制的引入标志着 VQA 研究的重要进展。注意力方法通过将焦点放在图像的关键区域和/或问题的关键词汇上, 提高了模型对于问题的理解和图像的解析能力。尽管基于注意力的方法在处理视觉信息时更为灵活和有效, 但它们通常需要大量的计算资源, 并且对于非常抽象的问题, 如需广泛的常识或专业知识, 它们的表现仍有限。

3) 基于外部知识的方法

在 2016 年, Qi Wu^[12] 等人探索了如何将外部知识库集成到 VQA 系统中, 旨在解决一些问题仅靠图像和问题本身信息不足以回答的情况。通过连接如 DBpedia 的知识库, 模型能够访问大量的事实和数据, 以丰富其回答。这种方法有助于回答需要广泛背景知识的复杂问题。尽管如此, 如何有效地集成和利用这些外部知识资源, 以及如何确保模型的推理过程既高效又可靠, 仍然是重要的研究方向。

4) 神经模块网络

Andreas^[3] 等人在 2016 年提出的神经模块网络 (Neural Modular Networks^[3]) 尝试以更可解释和灵活的方式解决 VQA 任务。通过将问题解析为不同功能的模块并动态组合这些模块, NMN 可以根据问题的具体需求定制化处理流程。这种模块化方法在理论上能够更好地处理复杂和多样化的问题, 但在实践中, 其性能依赖于有效模块的设计和组合, 而且对于语言解析和模块组合

的准确性要求较高。

1.2.2 实现 VISPROG 的其他基石工作

1) 思维链 Chain-of-Thought (CoT) prompting

Chain-of-Thought (CoT) prompting^[13]是一种自然语言处理中的推理方法，旨在利用大语言模型通过一系列的中间推理步骤解决复杂的推理问题。这种方法显示了 LLM 在逻辑推导和执行复杂任务中的强大能力。VISPROG 项目采用了类似的方法，通过将复杂任务分解为更小的、可管理的子任务，并由专门的子任务处理器处理。这种结构化的推理方法不仅提高了处理能力，也增加了模型的可解释性。这种方法的应用于 VQA 任务，显示了如何通过逐步推理来处理复杂的视觉问题，进一步证明了 CoT 在增强大型模型逻辑能力方面的价值。

2) 大语言模型 Large Language Model(LLM)

2022 年，由 Zhengyuan Yang^[14]等人进行的研究中，提出了利用 GPT-3 处理基于知识的视觉问答 (VQA)。他们创新性地利用了 GPT-3 的强大知识检索和问答能力，通过将图像转换成描述 (或标签)，这些描述再被用来提示 GPT-3 解决 VQA 任务。这种方法展现了大语言模型在少样本学习上的优势，且通过仅用 16 个样本，在 OK-VQA 数据集上实现了显著的性能提升，超越了之前的有监督学习方法。这标志着大语言模型在多模态任务中的应用达到了一个新的里程碑。

3) Socratic Models(SMs)

Socratic Models (SMs)^[15]是由 Andy Zeng 和他的团队在 2022 年提出的一个框架。SMs 代表了一种范式转变，它利用像 GPT-3^[9]、RoBERTa^[16]、CLIP^[17]等大型预训练模型来执行复杂的多模态任务，而无需额外训练。这一框架通过语言作为模型间通信的媒介，实现了这些模型的组合，从而允许在不同模态间进行零样本和小样本推理。SMs 的突破性在于其通过在各种现有基础模型之间创建结构化对话，进行多模态推理和问题解决的能力。每个模型根据其训练领域的独特能力，协同处理单一模型无法独立完成任务。例如，SMs 可以执行从零样本图像标注、视频到文本检索，到通过多模态对话协助日常活动 (如烹饪) 的任务，与最先进的方法相比展示了显著的竞争性能。

1.2.3 浅析困境与挑战

基于 Chongyan Chen et al^[18]和 Chongyan Chen et al^[19]的研究，我们可以总结出以下几点。总的来说，目前的 VQA 面临的挑战主要集中在现有方法可解释性不足、数据集设计、评价标准制定等方面。

1) 解释性不足问题

同上文所说，由于绝大多数方法中采用的联合嵌入式的“黑盒”操作，我们并不知道模型内部发生了什么。虽然引入注意力机制后，模型能够识别出与问题相关的图像区域，从而在一定程度上解释模型的预测结果，但这种方法仍然不能明确展示系统是如何理解这些图像区域的，以及为什么要关注这些特定的区域。

2) 数据集的问题

现今，VQA 领域已经有很多的数据集，一般的 VQA 有 COCO-QA^[20]，VQAv2^[21]，更细分，有专注于图像中文字的 Text-VQA，还有研究可解释性的 Visual7W, VQA-E，以及外部知识推论的 OK-VQA, CLEVER 等数据集。然而，现有的 VQA 数据集多为计算机设计，与实际应用场景存在脱节。并且大部分数据集使用网络收集的图像和人工生成或众包的问题，缺乏真实用户互动的场景。

3) 评价指标的问题

现有评价标准 (VQA Accuracy / Grounding) 主要针对短答案，难以全面评估模型在长形式答案和复杂推理上的表现。需要发展更全面和准确的评价机制，考虑问题、答案、图像的综合理解和相关性。

4) 衍生问题或其他问题

由以上几点会产生诸多衍生问题，例如数据偏见。由于数据集不够泛化，评测指标只考虑文本，很容易产生数据偏见。模型只学到了此数据集的特点，而不是真正的学会图像问答。一旦尝试真实世界的用户提问便会效果显著降低。并且不同语种之间的效果也有较大差异，英语效果较好，德语意大利语等次之，中文韩文等效果较差。

1.2.4 本文主要工作

面对上述困境与挑战，本文采用了 VisProg 框架来实现 VQA 任务，主要解决了可解释性的问题。本文对 VisProg 的整体架构进行了分析。这个架构包括两个主要部分：程序生成器和程序解释器。通过将问题分解为一系列可以由已有模块完成的子任务，并生成类似 Python 中的模块化程序来完成这些子任务，进而实现对每一个处理步骤进行单独的检查和调整，从而增强了整个系统的可解释性。

此外，我们还实现了多种基础模块，如定位模块 LOC、计数模块 COUNT、裁剪模块 CROP、评估模块 EVAL、结果输出模块 RESULT，以及视觉问答模块 VQA。这些模块共同协作，用于处理 VQA 任务。在实验部分，我们对不同的模型组件进行了比较，包括基于 ViLT 的 VQA 模块实现、CLIP-ViLT 模块和基于 BLIP 的 VQA 模块架构，并分析了它们在 VQA 任务上的表现。

至于训练过程，本文采用 GQA 来训练 VisProg，它是一个专门为图像理解和常识推理设计的数据集。通过其结构化的问题和详细的注释，支持更复杂的问题理解和推理，比传统的 VQA 数据集更适合于训练要求高度语义理解的模型。

评价指标则是采用了结构化的问答准确度 (Structured Question Answering Accuracy)，这一标准不仅衡量答案的准确性，还评估了模型在理解问题结构和逻辑上的能力。这种评价方式有助于确保模型不仅能找到正确答案，还能以一种逻辑上合理的方式来处理和回答问题。

接下来我们将详细介绍 VisProg，并展现我们的工作。通过实际应用案例和实验结果，我们将详细说明 VisProg 如何应对传统 VQA 系统所面临的各种挑战，以及它在实际场景中的表现和潜力。

1.3 本文的论文结构与章节安排

本文共分为六章，各章节内容安排如下：

- 1) **第一章: 绪论**。简单说明了本文章的选题背景与意义。
- 2) **第二章: VISPROG 框架的分析与实现**。分析了 VISPROG 整体的构架与组成成分。
- 3) **第三章: VISPROG 诸多基础模块的实现**。实现整体框架中除 VQA 模块外各类基础模块的搭建与构成。
- 4) **第四章: VQA 模块组件的实现**。使用 ViLT, CLIP, BLIP 等技术组建三种 VQA 模块。
- 5) **第五章: 实验与分析**。给出了使用我们的几种 VQA 模块在图像问答任务上

的表现，并在多个数据集上进行性能分析。

6) **第六章: 总结与展望。**总结我的工作，并给出未来的完善方向。

2 VISPROG 框架的分析与实现

2.1 VISPROG 基本目标

为了更好地理解其结构，此处提出笔者认为的 VISPROG 的初衷与作用：**提供一个统一框架，整合现有模型，完成种类繁多的视觉任务，追求泛用性。**

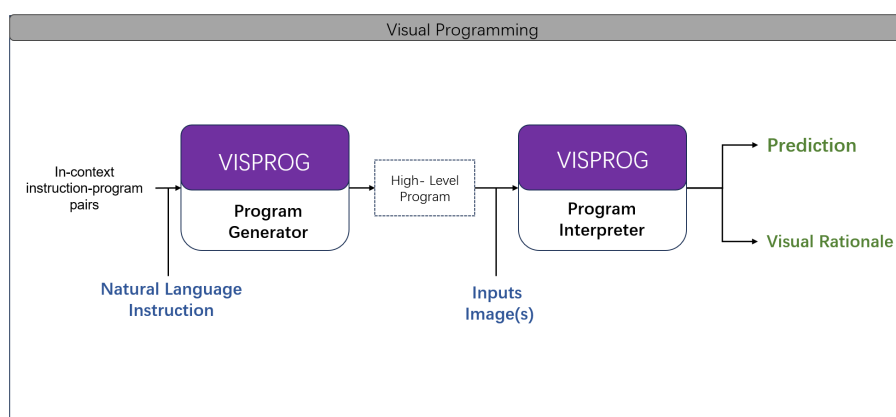


图 2.1 VISPROG 的整体框架

带着这一想法，我们来审视图 2.1 的 VISPROG 框架，其只要包括两个组件：“程序生成器”与“程序解释器”。程序生成器负责接受文本指令 `prompt` 并将主要任务逐步解析成诸多可以用现有模块完成的子任务，并将这个过程生成为一种“VISPROG 程序”传递给程序解释器。程序解释器负责解析“VISPROG 程序”并接受图像输入，逐步调用“高级程序”中所需的子模块来完成任务，并输出图像任务的结果与高级程序可视化的推理过程。接下来我们分别介绍这两个基本组件。

2.2 程序生成器 (Program Generator)

程序生成器的任务是**解析文本并划分成最小可执行任务，生成 VISPROG 程序**，文本 `prompt` 大多就是基于图像上的问题，但是整个过程中却并不需要用到图像数据，是一个纯 NLP 领域的问题。而随着大语言模型的流行，如 GPT-3 这类的大型语言模型已经展现出了在接触有限的输入输出示例之后对新任务样本具有显著的泛化能力。

例如下图 2.2，当以两个英法翻译实例作为上下文提示给 GPT-3 时，它能够自动生成新英语短语的法语翻译，显示出在未经特定任务训练的情况下仍能有效执行翻译任务的能力。

```
good morning -> bonjour
good day -> bonne journée
good evening ->
```

图 2.2 大语言模型已经具有了强大的上下文学习能力

于是, Tanmay Gupta^[1]等人提出可以使用 GPT-3 等大语言模型来完成程序生成器的工作。通过人工构建提示词 prompt, 教会 GPT-3 对于任务文本的分解范式, 并输出分解后的结果。如图 2.3 所示。输出的结果便可以作为“VISPROG 程序”的传递给程序解释器。



图 2.3 VISPROG 中的程序生成过程

每一个 VISPROG 程序或步骤包含模块名、模块的输入参数名称及其值和一个输出变量名称。VISPROG 程序通常利用之前步骤的输出变量作为后续步骤的输入。通过使用描述性的模块名称（例如“Select”、“ColorPop”、“Replace”）、参数名称（如“image”、“object”、“query”）和变量名称（如“IMAGE”、“OBJ”），GPT-3 能够理解每个模块的输入输出类型以及其功能。

在执行过程中, 这些输出变量可以用于存储各种类型的数据。例如, “OBJ”可能是图像中对象的列表, 其中每个对象都有相关的掩码、边界框和文本标签（如类别名称）。将这些上下文示例与新的自然语言指令结合输入到 GPT-3 中后, VISPROG

便能生成一个程序，即使不直接观察图像或其内容，也可以在输入的图像上执行描述的任务。

2.3 程序解释器 (Program Interpreter)

程序解释器的任务是调用模块执行 VISPROG 程序，输出结果并可视化推理过程。在这一过程中，图像数据输入到程序解释器并依照“VISPROG 程序”执行。

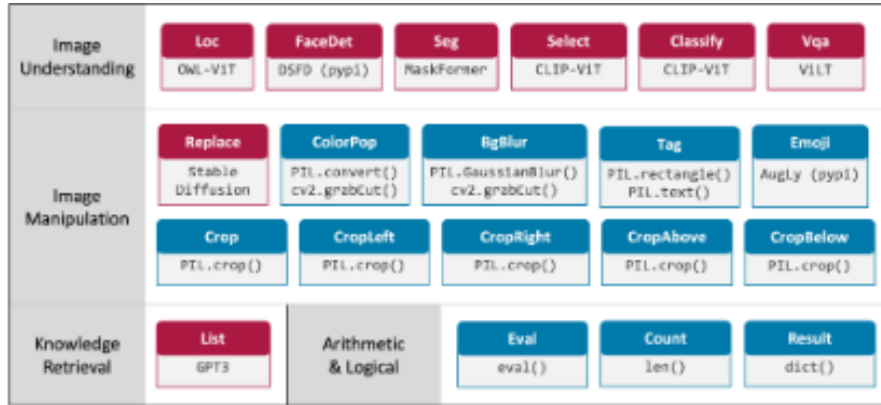


图 2.4 VISPROG 目前支持的模块

VISPROG 平台目前支持 20 种模块（如图 2.4所示），这些模块提供了图像理解、图像操作（包括生成）、知识检索以及执行算术和逻辑操作的功能。每个模块在 VISPROG 中以 Python 类的形式实现（参见图 2.5），包含三个主要功能：(i) 解析代码行以提取输入参数的名称和值以及输出变量的名称；(ii) 执行必要的计算，可能涉及已训练的神经模型，并以输出变量的名称和值更新程序状态；(iii) 使用 HTML 形式直观总结计算过程，为后续创建视觉理由。新增模块到 VISPROG 只需实现并注册相应的模块类，其程序执行则由 VISPROG 的解释器自动管理。

程序执行过程中，解释器逐行遍历程序，根据每行指定的输入调用相应模块。执行每个步骤后，程序状态将更新为该步骤输出的名称和值。在执行必要的计算之外，每个模块类还实现了一个名为 `html()` 的方法，用来在 HTML 片段中直观总结模块的输入和输出。解释器将所有程序步骤的 HTML 摘要拼接成可视化的视觉推理（如图 2.6所示），这有助于分析程序的逻辑正确性并检查中间输出。视觉推理还帮助用户理解失败的原因，并以最小化的自然语言指令调整来改善性能。

```
class VisProgModule():
    def __init__(self):
        # load a trained model; move to GPU

    def html(self, inputs: List, output: Any):
        # return an html string visualizing step I/O

    def parse(self, step: str):
        # parse step and return list of input values
        # and variables, and output variable name

    def execute(self, step: str, state: Dict):
        inputs, input_var_names, output_var_name = \
            self.parse(step)

        # get values of input variables from state
        for var_name in input_var_names:
            inputs.append(state[var_name])

        # perform computation using the loaded model
        output = some_computation(inputs)

        # update state
        state[output_var_name] = output

        # visual summary of the step computation
        step_html = self.html(inputs, output)
        return output, step_html
```

图 2.5 模块的 demo



图 2.6 关于判断 “At least three animals are in a flowered field.” 问题的可视化视觉推理过程

2.4 VISPROG 处理 VQA 任务的分析

VISPROG 天生的结构注定了其具有极强的可组合性，非常适合处理复合、多步骤的视觉问答任务。例如针对 VQA 任务，VISPROG 包含了多个模块，如定位模块 (LOC)、问答模块 (VQA)、裁剪模块 (CROP)、计数模块 (COUNT) 以及计算模块 (EVAL) 等。例如图 2.6，考虑这个问题：“两张图片中，是否至少有三只动物在花丛中？”。VISPROG 首先分别对两张图片分别用 VQA 模块提问“有多少只动物在花丛中？”，得到两部分的值，再用 EVAL 模块计算两答案之和并输出。通过这样结构化的思路，解决 VQA 的思路流程便一目了然了，大大提高了模型的可解释性。

以下为 VISPROG 执行程序的伪代码：

算法 2.1: VISPROG Visual Programming Framework Pseudocode

```

1 Class VISPROG
2   Initialize() self.modules  $\leftarrow$  LoadSupportedModules() // 加载支持的模块
3   self.programGenerator  $\leftarrow$  GenerateProgram() // 实例化程序生成器
4   self.programInterpreter  $\leftarrow$  ExecuteProgram() // 实例化程序执行器
5 Class ProgramGenerator
6   GenerateProgram(instruction) contextPairs  $\leftarrow$ 
   FetchInstructionProgramPairs() // 获取指令和程序对
7   highLevelProgram  $\leftarrow$  GPT3InContextLearning(contextPairs, instruction)
   // 通过 GPT-3 上下文学习生成高级程序
8   返回 highLevelProgram // 返回生成的程序
9 Class ProgramInterpreter
10  ExecuteProgram(highLevelProgram, images) programState  $\leftarrow$ 
   InitializeState(images) // 用图像初始化程序状态
11  对于 each step in highLevelProgram 进行
12    module, inputs  $\leftarrow$  ParseStep(step) // 解析程序的每一步
13    output  $\leftarrow$  module.Execute(inputs) // 执行模块
14    programState  $\leftarrow$  UpdateState(programState, output) // 更新程序状态
15  visualRationale  $\leftarrow$  GenerateVisualRationale(programState) // 生成步骤的视觉解释
16  prediction  $\leftarrow$  ExtractPrediction(programState) // 从状态中提取预测结果
17  返回 prediction, visualRationale // 返回预测和视觉解释
18 Main() instruction  $\leftarrow$  ReceiveNaturalLanguageInstruction() // 接收自然语言指令
19 visprog  $\leftarrow$  VISPROG();
20 program  $\leftarrow$  visprog.programGenerator(instruction);
21 prediction, rationale  $\leftarrow$  visprog.programInterpreter(program, images);
22 PresentResults(prediction, rationale) // 展示预测结果和视觉解释

```

3 VISPROG 诸多基础模块的实现

模块名称	解释器类	功能描述
LOC	LocInterpreter	定位图像中特定对象的位置
COUNT	CountInterpreter	计数图像中特定区域内的对象数量
CROP 类型	CropInterpreter, CropRightOfInterpreter 等	包括 1: 已知目标边框, 裁剪图像。2: 需要目标区域的相对位置, 如对象的上方、下方、左侧、右侧等特定区域
VQA	VQAInterpreter	处理视觉问答任务
EVAL	EvalInterpreter	评估和验证结果
RESULT	ResultInterpreter	输出和呈现结果

表 3.1 VISOPROG 中的 VQA 任务中使用的模块和解释器

3.1 定位模块 LOC

定位模块 LOC 是 VISPROG 框架中负责识别和定位图像中特定物体位置的组件。它利用 OwlViT^[22] 模型, 这是一个先进的 Vision Transformer 模型, 专门用于图像和文本的多模态理解。LOC 模块接收自然语言指令, 比如“找到图片中的女人”, 然后处理图像, 识别和定位指令中提到的物体。

下面是一个使用该模块的示例流程:

- 1) 输入图像路径和物体名称。
- 2) LOC 模块通过 OwlViT 模型处理图像, 识别其中的物体。
- 3) 如果找到了物体, 模块将在图像上标注一个边界框, 准确指出物体的位置。
- 4) 输出标注过的图像, 提供可视化的物体定位。

这里给出一个示例展示模块的标注效果:

图 3.1 原图像, 定位目标: “women”



图 3.2 使用定位模块 LOC 标注的图像示例



这张图片中, 目标是定位图像中的“women”。使用 LOC 模块处理后, 我们可

以得到包含边界框的图像，其框出了图像中的女人。

3.2 计数模块 COUNT

计数模块 COUNT 是 VISPROG 框架中负责统计图像特定区域内物体数量的组件。这个模块特别适用于简单的计数任务，例如统计图像中的人数或某类物体的数量。

该模块的工作流程如下：

- 1) 从程序步骤中解析出边界框变量和输出变量。
- 2) 计算边界框变量中包含的框数量，即物体的数量。
- 3) 将计数结果存储在输出变量中。
- 4) 如果需要，生成一个 HTML 可视化，显示计数结果和边界框。

例如，如果我们有一个包含边界框的图像，并且想知道其中有多少个边界框，即物体数量，Count 模块将为我们提供准确的计数。

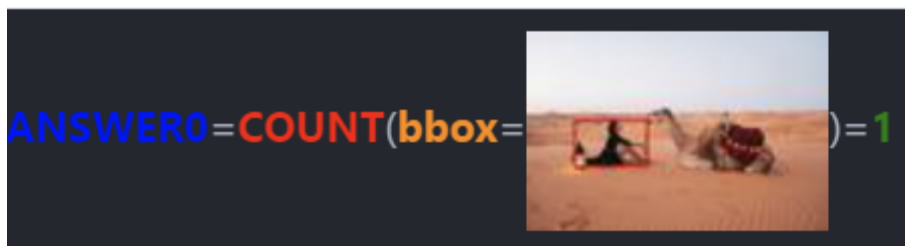


图 3.3 使用计数模块 COUNT 处理的图像示例，显示物体数量和相应的边界框。

3.3 裁剪模块 CROP

裁剪模块 CROP 是 VISPROG 框架中用于裁剪图像特定区域的组件。该模块可以识别图像中的对象或特定区域，并提取出感兴趣的部分。这对于只关注图像的某一部分的任务尤其有用。

该模块可以处理多种类型的裁剪指令：

- 根据指定的边界框来裁剪图像，适用于你已经知道要裁剪图像的确切区域的情况。(CropInterpreter 类)
- 不需要用户指定一个确切的边界框，而是基于根据图像中的相对位置（如右侧、左侧、上方或下方）裁剪出相应区域。(如 CropRightOfInterpreter, CropLeftOfInterpreter, CropAboveInterpreter, 和 CropBelowInterpreter)

在执行裁剪操作时，模块会扩展边界框以包含周围的一些上下文，确保对象不会被裁剪过度，会裁剪 1.5 倍于裁剪目标的方框。

下面是一个裁剪操作的示例流程：

- 1) 接收裁剪指令，包括边界框和图像变量。
- 2) 依据边界框和裁剪类型，确定裁剪的具体区域。
- 3) 执行裁剪操作，获取裁剪后的图像。
- 4) 如果需要，生成一个 HTML 可视化，显示裁剪后的图像和原始边界框。

例如下图，假设我们有一张图像，并且想要裁剪出图像中“women”部分。我们使用 CROP 模块进行处理。

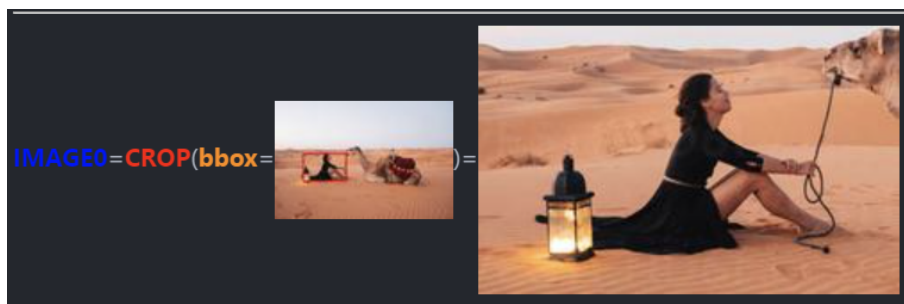


图 3.4 使用裁剪模块 CROP 处理的图像示例，显示裁剪 1.5 倍目标区域的“women”。

3.4 EVAL 模块

EVAL 模块是 VISPROG 框架中用于逻辑和算术表达式求值的组件。它解析程序步骤中的表达式，并计算其结果。

该模块的工作流程如下：

- 1) 解析程序步骤，获取表达式和输出变量。
- 2) 根据当前的程序状态和给定的表达式计算结果。
- 3) 存储计算结果到输出变量。
- 4) 如果需要，生成一个 HTML 可视化，展示表达式及其计算结果。

例如，EVAL 模块可以处理如“ $1 + 1 = 2$ ”或“变量 X 大于变量 Y”等简单或复杂的表达式。

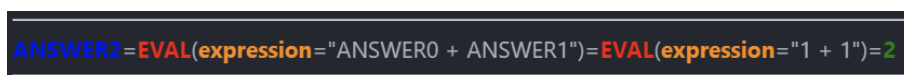


图 3.5 EVAL 模块完成“ $1+1=2$ ”的表达式计算的示例

3.5 RESULT 模块

RESULT 模块是 VISPROG 框架中用于输出和展示程序结果的组件。它获取程序的状态，并将指定变量的值显示为结果。

该模块的工作流程如下：

- 1) 解析程序步骤，获取需要展示的变量名。
- 2) 从程序状态中检索指定变量的值。
- 3) 如果需要，生成一个 HTML 可视化，展示变量的值。

RESULT 模块确保最终结果能够被用户或系统进一步处理和理解，例如在一个交互式的可视化界面中显示结果。

3.6 VQA 模块

VQA 模块是 VISPROG 框架中用于执行视觉问答任务的核心组件。它结合了自然语言处理和计算机视觉技术，对给定的图像和问题进行理解和分析，最终生成相关的答案。

该模块的工作流程如下：

- 1) 接收包含图像和问题的输入。
- 2) 使用预训练的模型处理输入，该模型理解图像内容和问题的语境。
- 3) 生成答案，并将其存储在指定的输出变量中。
- 4) 如果需要，生成一个 HTML 可视化，显示问题、图像及其答案。

例如下图，如果给定一张图像和问题“这个裙子是什么颜色？”，VQA 模块将分析图像和问题内容，确定图像中的颜色类型，并返回答案，如“黑色”。

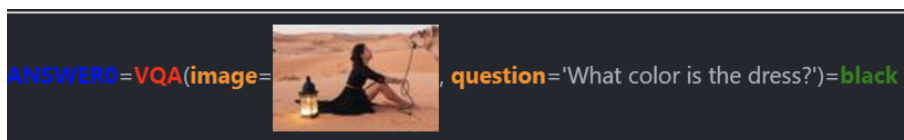


图 3.6 VQA 模块完成对“what color is the dress”的解答

VQA 模块使用的预训练模型可能是基于多种技术的，如 BLIP、VILT 或 CLIP 等等，它们都是目前领先的视觉问答技术。

重要性： VQA 模块在 VISPROG 框架中的作用至关重要，因为它直接关联到理解视觉内容和自然语言问题的能力，是实现高级视觉问答功能的关键。

接下来的一章里，笔者将详细的阐述本文探索、实现并改进了哪些 VQA 模块。

4 几种 VQA 模块组件的实现

4.1 ViLT(Vision and Language Transformer) 模块实现

有关视觉-文本这样的多模态的任务，极其关键的两点便是 1: 每个模态的特征提取方式。2: 两者模态的特征的交互。2021 年，Wonjae Kim^[23] 等人认为，传统的 VLP(Vision-and-Language Pre-training) 方法高度依赖卷积神经网络 (CNN) 来提取图像特征，或采用目标检测等区域监督技术。这些过程计算成本高，并受到视觉嵌入器容量和预定义视觉词汇表表达能力的限制。于是他们在 ViT(Vision Transformer)^[24] 的基础上提出了 ViLT(Vision-and-Language Transformer) 作为一类新的架构最简单的 VLP 模型，对图像简单的分块来作为图像特征，计算重心放在模态交互上而非图像的特征提取上，以此来达到提升模型性能的目的。

Taxonomy of Vision-and-Language Models

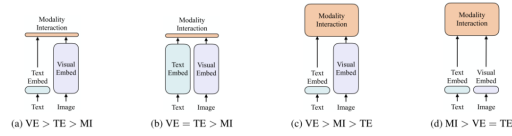


图 4.1 不同类型的 VLP 模型 (Vision-Language Pre-training)

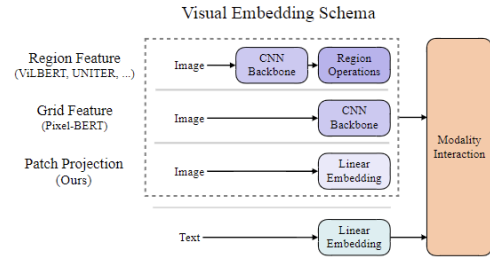


图 4.2 图像特征几类不同提取方式

4.1.1 ViLT 模块架构

在 ViLT 中，视觉编码器不是以传统的卷积来处理提取特征的，而是将图像分块后输入 Transformer Encoder。给定大小为 $H \times W \times C$ （高度、宽度和通道）的图像 I ，它被重塑成一系列展平的 2D 块 I_p ，大小为 $N \times (P^2 \cdot C)$ ，其中 (P, P) 是每个图像块的分辨率， $N = HW/P^2$ 是块的数量。每个块随后被线性嵌入并加上位置嵌入，然后送入 Transformer 中。

在视觉语言变换器 (ViLT) 中，模型接收类别标签的嵌入向量 t 和视觉嵌入向量 v 作为输入，这些向量分别包含了文本和视觉信息的位置嵌入。具体表达为：

$$t = [t_{\text{class}}; t_1; \dots; t_T] + T_{\text{pos}}, \quad v = [v_{\text{class}}; v_1; \dots; v_N] + V_{\text{pos}}$$

其中 t_{class} 和 v_{class} 是类别嵌入向量， T_{pos} 和 V_{pos} 分别是文本和视觉的位置嵌入向量。

文本和视觉的嵌入向量被合并成一个统一的序列 $z^0 = [t + t_{\text{type}}, v + v_{\text{type}}]$ 。

模态交互的 Transformer 部分包括多头自注意力 (MSA) 和多层感知机 (MLP) 两部分结构，公式如下：

$$z^0 = [t + t_{\text{type}}, v + v_{\text{type}}], \quad (4.1)$$

$$z^d = \text{MSA}(\text{LN}(z^{d-1})) + z^{d-1}, \quad d = 1 \dots D, \quad (4.2)$$

$$z^d = \text{MLP}(\text{LN}(z^d)) + z^d, \quad d = 1 \dots D, \quad (4.3)$$

其中 z^0 为初始的联合嵌入表示，LN 指的是层标准化 (Layer Normalization)， D 表示变换器的层数。MSA 即多头自注意力，它可以并行地关注输入序列的不同部分，增强模型对信息的捕捉能力。MLP 即多层感知机，是一种简单的全连接神经网络，负责提供非线性变换。

最后的输出 $p = \tanh(z^0 W_{\text{pool}})$ 表示通过一个池化层对变换器的输出进行聚合，以用于任务的最终预测。

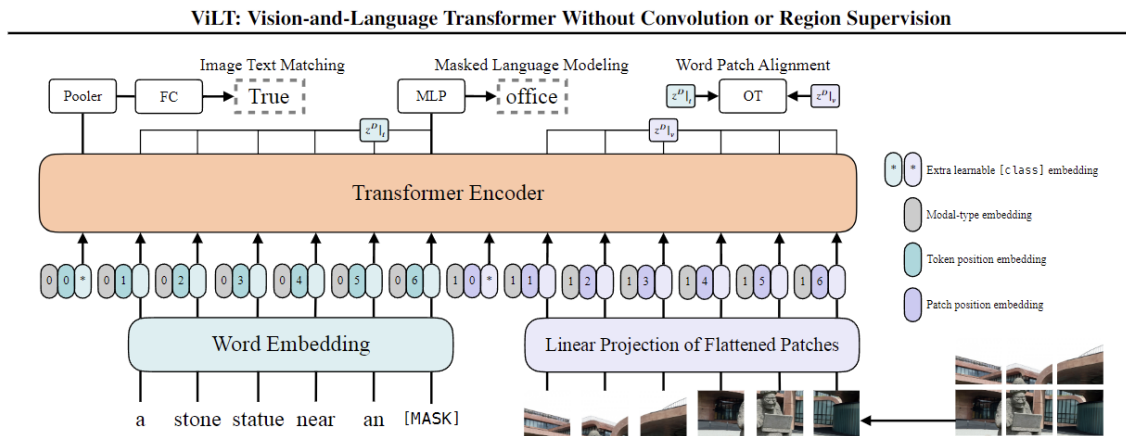


图 4.3 ViLT 模型的整体结构

4.1.2 ViLT 算法伪代码实现

算法 4.1: Vision-and-Language Transformer (ViLT) Pseudocode

```

1 Function ViLT( $I, T$ ):
2    $v \leftarrow \text{ImagePatchEmbedding}(I)$ 
3    $t \leftarrow \text{TextTokenEmbedding}(T)$ 
4    $z^0 \leftarrow \text{Concatenate}(v, t)$ 
5   对于  $d \leftarrow 1$  转到  $D$  进行
6      $z^d \leftarrow \text{LayerNorm}(z^{d-1})$ 
7      $z^d \leftarrow \text{MultiHeadSelfAttention}(z^d) + z^d$ 
8      $z^d \leftarrow \text{LayerNorm}(z^d)$ 
9      $z^d \leftarrow \text{MLP}(z^d) + z^d$ 
10   $p \leftarrow \text{TanhPooling}(z^0)$ 
11  返回  $p$ 

```

4.1.3 输入表示

ViLT 采用文本和图像统一的嵌入序列表示，避免了对视觉数据进行单独的卷积处理。文本输入经过分词，并与学习到的位置嵌入结合，而图像则被处理成展平的块，随后线性投射成嵌入向量。这些嵌入向量接着添加到各自的类型嵌入上，并连接起来形成初始的多模态序列 z_0 。

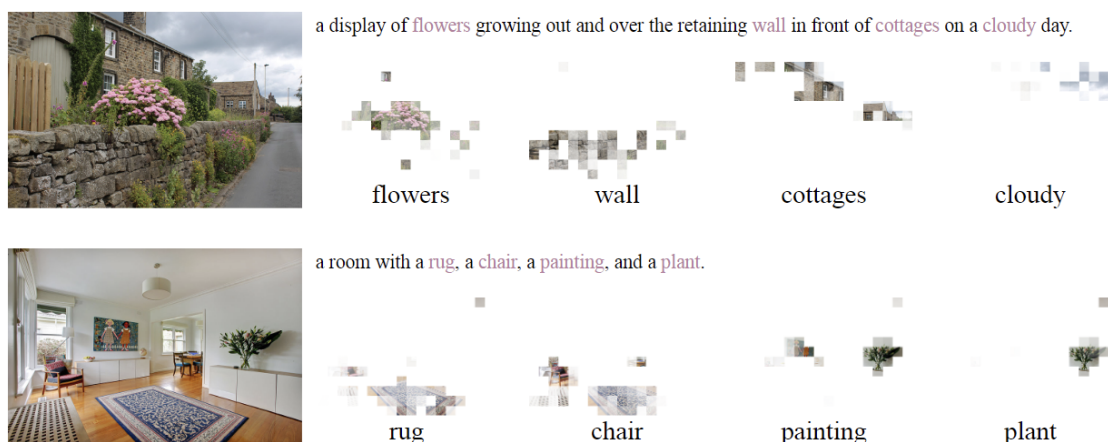


图 4.4 ViLT 提取到的图像-文本对

4.1.4 ViLT 模块中的自注意力机制

ViLT 的架构利用**多头自注意力 (MSA) 机制**，使模型能够专注于输入序列的不同部分，以计算文本和图像嵌入特征之间的上下文关系。这种机制在多个变换器层 (D) 中通过迭代的方式应用，每一层都更新上下文化的序列。ViLT 在自注意力和前馈 (MLP) 层之前使用了层标准化 (LN)，这与 BERT 不同，后者是在之后使用 (后规范化)。这种方法帮助 ViLT 处理序列，以理解和整合多模态信息。

自注意力机制

让我们用 $X \in \mathbb{R}^{n \times d}$ 表示 n 个实体的序列 (x_1, x_2, \dots, x_n) ，其中 d 是嵌入维度。自注意力的目标是通过编码每个实体与全局上下文信息的交互来捕捉所有 n 个实体之间的交互。这是通过定义三个可学习的权重矩阵来实现的，转换查询 ($W^Q \in \mathbb{R}^{d \times d_q}$)，键 ($W^K \in \mathbb{R}^{d \times d_k}$) 和值 ($W^V \in \mathbb{R}^{d \times d_v}$)，其中 $d_q = d_k$ 。输入序列 X 首先被投影到这些权重矩阵上以获得 $Q = XW^Q$ ， $K = XW^K$ ，和 $V = XW^V$ 。自注意力层的输出 $Z \in \mathbb{R}^{n \times d_v}$ 是：

$$Z = \text{softmax} \left(\frac{QK^T}{\sqrt{d_q}} \right) V.$$

多头自注意力

为了封装序列中不同元素之间的多种复杂关系，多头自注意力包含多个自注意力块。每个块都有它自己的一组可学习的权重矩阵 W_i^Q, W_i^K, W_i^V ，其中 $i = 0 \dots (h-1)$ 。对于一个输入 X ，输出 Z_h 是通过将自注意力块的 h 个头的结果连接成一个单一的矩阵 $[Z_0, Z_1, \dots, Z_{h-1}]$ 并投影到权重矩阵 $W^O \in \mathbb{R}^{hd_v \times d}$ 上获得的。

遮蔽自注意力

对于序列中的每一个实体，在自注意力计算的基础上，遮蔽自注意力进一步使用遮蔽矩阵 M 来防止未来信息的泄露。遮蔽操作通过与上三角矩阵 M 的哈达马乘积 (元素级乘积) 来完成。遮蔽自注意力定义如下：

$$\text{softmax} \left(\frac{(QK^T) \odot M}{\sqrt{d_q}} \right),$$

其中 \odot 表示哈达马乘积。

4.1.5 ViLT 模块中的训练目标和损失函数

ViLT 使用两个视觉语言预训练模型常见的主要目标进行训练：图像-文本匹配 (ITM) 和遮蔽语言模型 (MLM)。在 ITM 任务中，模型学习预测图像-文本对是否匹配，使用二分类头进行预测。MLM 目标鼓励模型利用文本和图像的上下文预测文本序列中遮蔽位置的正确标记。在训练期间，为 ITM 和 MLM 任务计算负对数似然损失，以更新模型的权重。

4.2 CLIP-ViLT 模块实现

上面提到的 ViLT 为了追求简单高效，使用的视觉编码器极其简单，就是简单的 Patch Embedding。但是，我们可以在这之上加一些 Tricks 来提升视觉编码器的特征提取能力。就比如笔者接下来要介绍的 CLIP 技术。

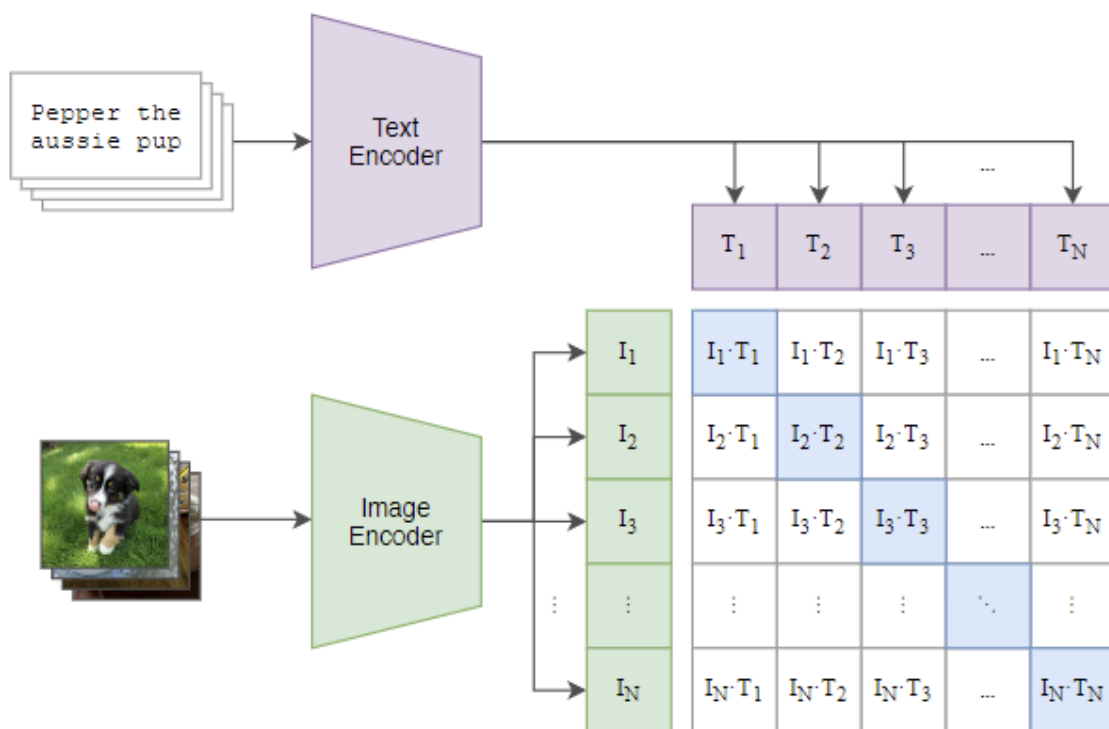
4.2.1 关于 CLIP 技术

一些介绍

经过调研^[3,2,25]，不论是联合嵌入类模型，基于注意力的联合嵌入类模型，序列到序列类模型，这些之前的跨模态模型大多是关注于如何将图像信息转换或映射为文本信息，模型通常专注于单向的从图像到文本的映射过程，即将视觉信息翻译或解释为语言信息。

而与此同时，CLIP (Contrastive Language-Image Pre-training)^[17]模型则不仅仅关注从图像到文本的单向映射，而是同时考虑图像和文本之间的双向关系。它通过对比学习方法，在大规模的图像-文本对上进行训练，旨在理解图像和文本之间的语义关联。这种方法强调图像和文本之间的映射关系，而不是仅仅将图像直接转换为文本。

Sheng Shen^[26]等人通过大量的实验证明了使用 CLIP 的模型可以学习到更加泛化的视觉-语言表示，使其在没有明确任务指导的情况下，也能有效地处理和推理跨模态内容。例如，CLIP 模型可以用于图像分类、对象检测、图像-文本匹配等多种任务，并且具有出色的零样本 (zero-shot) 学习能力，这意味着它能够在未见过的任务或数据上表现良好，显著提高了模型的泛化性和灵活性。



CLIP 的训练

CLIP（对比语言-图像预训练）的训练过程通过对比损失函数实现，该函数衡量图像特征与文本特征之间的对齐程度。训练目标是在大型数据集中最大化图像与其描述性文本之间特征的一致性。

训练过程涉及以下几个步骤：

- 1) **特征提取**: 图像编码器和文本编码器分别对图像和文本描述进行编码，转换为特征向量。这些编码器通常是特别设计的神经网络，以有效处理它们各自的数据类型。
- 2) **标准化**: 对图像和文本特征向量进行标准化处理，以保证它们在单位超球上。这通常通过 L2 标准化实现。
- 3) **对齐分数计算**: 图像和文本之间的对齐（相容性）是通过它们的特征向量的余弦相似度计算的。如果 I 是图像特征向量， T 是文本特征向量，它们的对齐分数可以计算如下：

$$\text{Alignment Score} = \frac{I \cdot T}{\|I\| \|T\|}$$

其中， I 和 T 分别表示图像和文本的特征向量。这个分数衡量了图像特征与文本特征的一致性程度。

- 4) **对比损失函数**: 在训练过程中，对比损失函数鼓励模型对于匹配的图像-文本

对产生高对齐分数，而对于不匹配的对产生低对齐分数。损失函数在一个批次中的所有正样本对（正确的图像-文本匹配）和负样本对（不正确的匹配）上进行计算。对比损失可以表示为以下公式，其中 τ 是温度参数，用于缩放对数几率：

$$\text{Contrastive Loss} = -\log \frac{\exp(\text{Alignment Score}/\tau)}{\sum_{k=1}^N \exp(\text{Alignment Score}_k/\tau)}$$

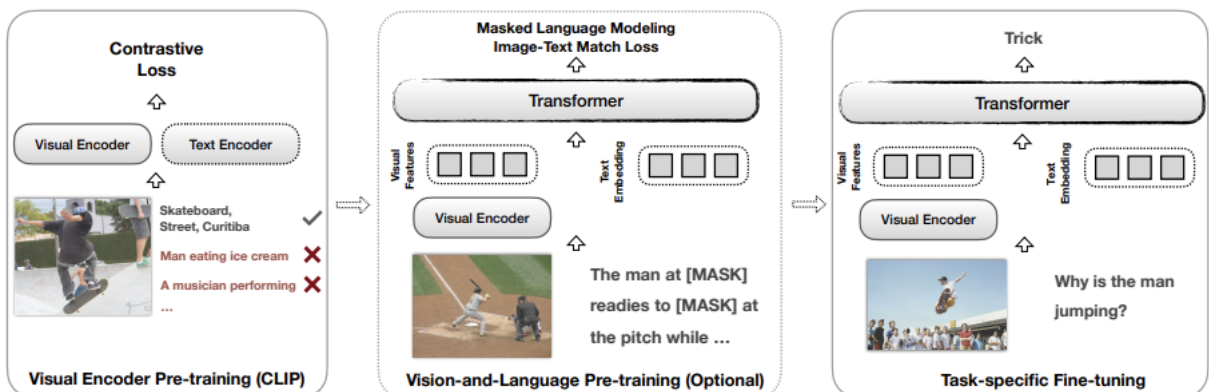
在这个公式中， N 表示批次中负样本的数量。

模型通过多次迭代训练来最小化这个损失，从而有效学习图像与其对应文本描述之间的对齐。温度参数 τ 在决定训练过程中类别分布的锐度方面起着关键作用，并且可以显著影响模型的性能。

4.2.2 使用 CLIP 预训练 ViT 作为视觉编码器

Sheng Shen^[26] 等人通过一系列实验，验证了经过 CLIP 预训练后的视觉编码器，在视觉语言预训练（VLP）任务中具有显著的提升。他们在 ResNet 和 Vision Transformer 两种广为应用的视觉编码器之上，实验了使用 CLIP 预训练所带来的性能对比，并发现使用 CLIP 预训练后，视觉编码器在多个下游任务上表现更加出色。具体来说，实验涉及两种原始视觉编码器的变体——CLIP-Res50 和 CLIP-ViT，他们使用这些模型在视觉问答（Visual Question Answering）、图像描述（Image Captioning）等任务上进行了测试。结果显示，使用 CLIP 的模型在这些任务上的性能均超过了使用传统视觉特征提取器（如 ImageNet 上训练的 ResNet）的模型。于是，笔者试着使用 CLIP 预训练 ViT，并使用 CLIP-ViT 来替代 ViLT 中的视觉编码器，并以此作为第二种 VQA 模块：CLIP-ViLT。

图 4.5 预训练 CLIP 作为 VLP 模型中的视觉编码器



4.2.3 CLIP-ViT 模块架构

CLIP-ViT 模型输入表示

CLIP 中图像与文本的模态交互是“浅层的”，它并不采用将两者特征映射到同一个空间的做法，而是让其中视觉编码器和文本编码器独立编码输入的图像和文本，将输出得到的两个特征向量的点积用作输入图像和文本之间的“对齐分数”来表达两个特征向量的关联强弱。通过对比损失（contrastive loss）进行预训练，模型需要从随机抽取的图像-文本对中区分出两者匹配的对。

算法 4.2: CLIP 训练过程伪代码

```

输入: train_dataset; // 包含图像和文本的训练数据集
输入: image_encoder, text_encoder; // 图像和文本编码器
输入: optimizer; // 优化器
输入: epochs; // 训练周期数
数据结构: t; // 温度参数, 用于调整相似度计算
1 对于 epoch = 1 转到 epochs 进行
2   对于每个  $(I, T)$  in train_dataset 进行
3        $I_f \leftarrow \text{image\_encoder}(I)$ ; // 提取图像特征
4        $T_f \leftarrow \text{text\_encoder}(T)$ ; // 提取文本特征
5        $I_f \leftarrow \text{l2\_normalize}(I_f)$ ; // L2 归一化图像特征
6        $T_f \leftarrow \text{l2\_normalize}(T_f)$ ; // L2 归一化文本特征
7        $\text{logits} \leftarrow \frac{I_f \cdot T_f^T}{t}$ ; // 计算归一化特征的余弦相似度
8        $\text{labels} \leftarrow \text{range}(\text{batch\_size})$ ; // 创建正标签
9        $\text{loss} \leftarrow \text{cross\_entropy\_loss}(\text{logits}, \text{labels})$ ; // 计算对比损失
10       $\text{optimizer.zero\_grad}()$ ; // 梯度归零
11       $\text{loss.backward}()$ ; // 反向传播
12       $\text{optimizer.step}()$ ; // 梯度更新
13  循环结束
14 循环结束

```

4.3 BLIP-VQA 模块实现

相比于前两者基于轻量级的 ViLT 多模态模型的模块, BLIP(Bootstrapping Language-Image Pre-training) 体量要大得多, 结构也复杂得多。但是其中的原理却已经在前面两个部分大多阐述了, 我们也逐步地从 VQA 的历史的余烬中走向了它的今生。这里我们便以目前行业能力最为强大的多模态模型 BLIP 为 backbone, 实现 BLIP-VQA 模块来作为几个 VQA 模块中的最高 baseline。

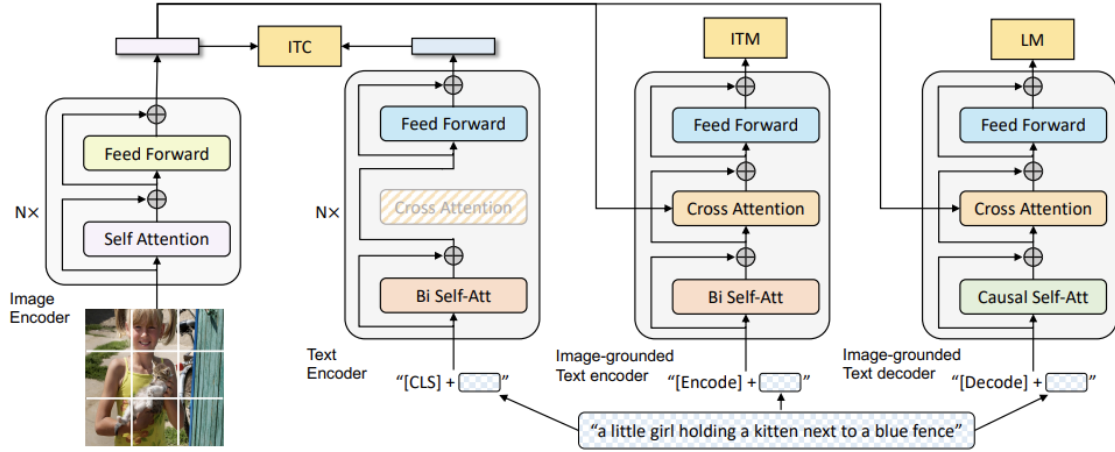


图 4.6 BLIP 的预训练框架

4.3.1 BLIP 的技术分析

由于我们的目标是使用 BLIP 实现 VQA 模块,只需要了解 BLIP 的架构(图 4.6)。源论文关于 Captioner-Filter(一种类似于 GAN 的图像描述对抗生成,以优化 BLIP 能力)的部分此处不做介绍。BLIP 架构的核心是多模态混合编解码器(MED)结构,该结构包括两个单模态编码器和一个以图像为基础的编解码器,这些组件共同作用于视觉和语言的预训练目标。

MED 架构组件(图 4.6)

4.3.1.1 图像编码器 (Image Encoder)

基于视觉 Transformer (ViT) 架构,该编码器将输入图像分割成多个 patch 并将其编码为一个嵌入序列,同时使用一个额外的 [CLS] token 来捕捉全局的图像特征。这种方法与使用预训练的目标检测器相比,在计算上更高效,并能够捕捉到图像的全局上下文信息。

4.3.1.2 文本编码器 (Text Encoder)

采用类似于 BERT 的架构,文本编码器通过在输入文本的开头添加 [CLS] token 来提取文本的上下文信息,以便为之后的多模态交互打下基础。

4.3.1.3 基于图像的文本编码器 (Image-grounded Text Encoder)

在文本编码器中嵌入了额外的交叉注意力 (Cross-Attention, CA) 层,以将视觉信息注入文本表示中。通过在文本中添加一个特定的 [Encode] token,该编码器

可以学习图像和相关文本之间细粒度的关系。

4.3.1.4 基于图像的文本解码器 (Image-grounded Text Decoder)

对于生成任务，该解码器通过将双向自注意力层替换为因果自注意力层，并添加 [Decode] token 和 [EOS] token 来标记序列的开始和结束，使模型能够根据图像内容生成描述性文本。

预训练目标

4.3.1.5 图像-文本对比损失 (Image-Text Contrastive Loss, ITC)

此损失函数旨在将视觉 Transformer 和文本 Transformer 的特征空间进行对齐。正向的图像-文本对在特征空间中被拉近，而负向对则被推远，这样的对齐被证明对于提高视觉和语言理解非常有效。数学上，ITC 损失可以表示为：

$$L_{ITC} = - \sum_{i=1}^N \log \frac{\exp(\text{sim}(v_i, t_i)/\tau)}{\sum_{j=1}^N \exp(\text{sim}(v_i, t_j)/\tau)} \quad (4.4)$$

其中， $\text{sim}(v, t)$ 表示图像嵌入 v 和文本嵌入 t 的相似性， τ 是温度参数。该目标参考了 Li 等人的工作。

4.3.1.6 图像-文本匹配损失 (Image-Text Matching Loss, ITM)

ITM 损失用于学习能够捕获视觉与语言之间细粒度对齐的图像-文本多模态表示。它是一个二元分类任务，通过 ITM 头（线性层）预测图像-文本对是正向的（匹配的）还是负向的（不匹配的）。数学表达为：

$$L_{ITM} = -y \log(\sigma(w^T f)) - (1 - y) \log(1 - \sigma(w^T f)) \quad (4.5)$$

σ 是 sigmoid 函数， w 是线性层的权重， f 是多模态特征， y 是标签（正匹配为 1，负匹配为 0）。使用了 Li 等人提出的难负样本挖掘策略。

4.3.1.7 语言建模损失 (Language Modeling Loss, LM)

LM 损失激活了图像引导的文本解码器，旨在生成给定图像的文本描述。通过最大化文本序列的似然来优化交叉熵损失，训练模型以自回归方式生成文本。LM 的交叉熵损失可以写为：

$$L_{LM} = - \sum_{i=1}^N \log P(w_i | w_{<i}, I) \quad (4.6)$$

其中, $P(w_i | w_{<i}, I)$ 是在给定图像 I 和前 $i-1$ 个单词的条件下, 预测第 i 个单词的概率。此处采用了标签平滑技术, 标签平滑系数为 0.1。

以上三个目标在预训练期间被联合优化, 目的是在高效预训练的同时利用多任务学习, 提高 BLIP 模型在不同视觉-语言任务上的性能。

通过这种方式, BLIP 的 MED 结构实现了一个具有灵活性和强大的视觉-语言理解及生成能力的统一模型, 适用于多种视觉-语言任务, 包括但不限于图像字幕、视觉问答以及图像-文本检索。

4.3.2 BLIP 如何用于 VQA 任务

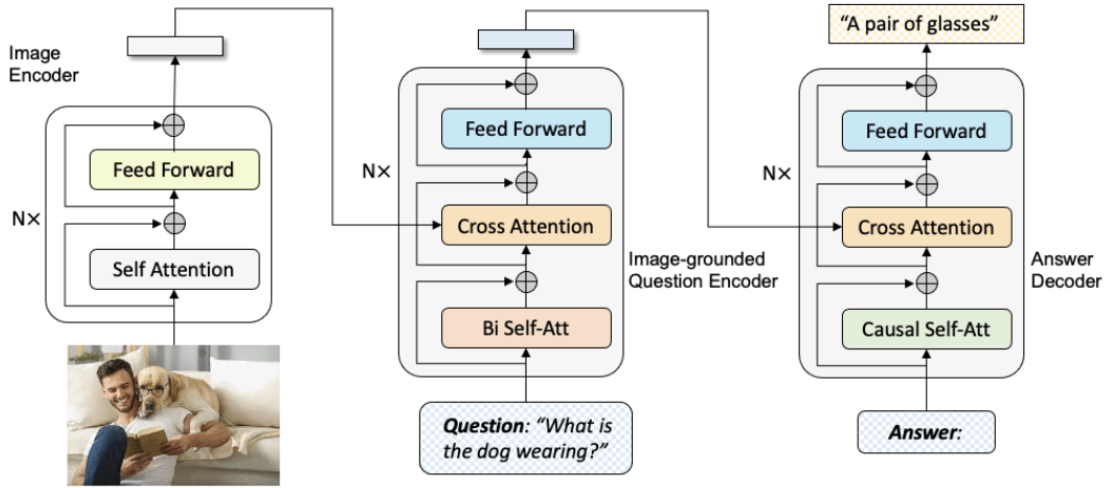


图 4.7 BLIP-VQA 的结构示例

由上一节的阐述, 我们已经知道了 BLIP 的几个组件, 为了适用于 VQA 这个下游任务, 如图 4.7, 我们可以这样搭建框架: 结合上一小节, 我们可知, BLIP-VQA 模块使用了 MED 框架中的三个部分 (图像编码器 Section 4.3.1.1, 基于图像的文本编码器 Section 4.3.1.3, 基于图像的文本解码器 Section 4.3.1.4),

图像编码: 首先, 图像编码器通过一系列自注意力和前馈层 (符合标准 Transformer 架构) 处理输入图像, 创建视觉内容的丰富表示。这种编码表示捕捉了图像的全局特征。

问题编码: 问题文本, 如 “狗戴了什么?” 使用类似的基于 Transformer 的架构进行编码, 可能类似于 BERT 模型, 生成问题的上下文文化表示, 对应文本编码器组件。

图像问题交叉注意力编码： 图像引导的问题编码器结合图像和问题的嵌入，并使用交叉注意力机制融合视觉和文本信息。这一步骤至关重要，因为它对齐并结合了视觉数据中与问题相关的特征，对应以图像为基础的文本编码器组件。

答案生成： 答案解码器利用结合的图像问题表示来生成文本答案。它用因果自注意力替换了双向自注意力，确保每个词的生成只能依赖于之前生成的词，模仿自然语言生成过程。解码器中的交叉注意力层允许它在生成答案时关注图像问题表示的相关部分，对应以图像为基础的文本解码器组件。

算法 4.3: How BLIP operates on VQA task

输入: Image $image$, Question $question$

输出: Answer to the VQA task

```

1  $image\_features \leftarrow ImageEncoder(image)$ 
2 对于  $i \leftarrow 1$  转到  $N$                                 //  $N$  is the number of layers
3 进行
4    $image\_features \leftarrow SelfAttention(image\_features)$ 
5    $image\_features \leftarrow FeedForward(image\_features)$ 
6  $question\_features \leftarrow QuestionEncoder(question)$ 
7 对于  $i \leftarrow 1$  转到  $N$  进行
8    $question\_features \leftarrow BiSelfAttention(question\_features)$ 
9    $question\_features \leftarrow FeedForward(question\_features)$ 
10  $combined\_features \leftarrow image\_features$ 
11 对于  $i \leftarrow 1$  转到  $N$  进行
12    $combined\_features \leftarrow$ 
13    $CrossAttention(combined\_features, question\_features)$ 
14    $combined\_features \leftarrow FeedForward(combined\_features)$ 
15  $answer \leftarrow ""$ 
16 当  $\neg EndOfAnswer(answer)$  进行
17    $answer \leftarrow answer + CausalSelfAttention(answer, combined\_features)$ 
18    $answer \leftarrow FeedForward(answer)$ 
19 返回  $answer$ 

```

5 实验与分析

5.1 程序生成器中的实验:

Prompt 模板数对 VISPROG 在 VQA 问题上的表现的影响

所用数据集

数据集选用 GQA^[27], GQA 数据集在设计时, 每一个问题都被标注了问题类型, 如对象识别、数量估计和属性理解等。其中问题的种类十分繁多 (100 种以上), 这里为了控制数据集的大小, 我们每种问题只选取最多 25 个样本 (5 个来自 GQA 中的 balanced val, 20 个来自 testdev 部分)。

实验设置

VISPROG 选取了 GQA 中的所有问题类型中的 31 个类型设置了对应的 Prompt 模板。在本次实验中, 假设从 31 中选取 e 个 Prompt 模板 ($e \leq 31$), 那么 VISPROG 将从 31 个 Prompt 模板中选取 e 个, 然后与输入的新问题一起, 送到 GPT3 进行上下文学习与预测 Prompt 模板。以下图 5.1 为例:

```
Think step by step to answer the question.

Question: Is the vehicle in the top of the image?
Program:
BOX0=Loc(image=IMAGE, object='TOP')
IMAGE0=Crop(image=IMAGE, bbox=BOX0)
BOX1=Loc(image=IMAGE0, object='vehicle')
ANSWER0=Count(bbox=BOX1)
ANSWER1=Eval("'yes' if {ANSWER0} > 0 else 'no'")
RESULT=ANSWER1

Question: What color is the curtain that is to the right of the mirror?
Program:
BOX0=Loc(image=IMAGE, object='mirror')
IMAGE0=CropRight(image=IMAGE, bbox=BOX0)
ANSWER0=Vqa(image=IMAGE0, question='What color is the curtain?')
RESULT=ANSWER0

Question: Does the traffic cone have white color?
Program:
BOX0=Loc(image=IMAGE, object='traffic cone')
IMAGE0=Crop(image=IMAGE, bbox=BOX0)
ANSWER0=Vqa(image=IMAGE0, question='What color is the traffic cone?')
ANSWER1=Eval("'yes' if {ANSWER0} == 'white' else 'no'")
RESULT=ANSWER1

Question: {new_question}
Program:
```

图 5.1 $e = 3$, Prompt 模板示例

我们对比了两种 Prompt 选取策略：

Prompt 策略	描述
不使用投票 (w/o voting)	直接单次随机选取模型单次运行，不结合其他运行结果。
使用投票 (w/ voting)	通过结合五个不同随机种子随机选取 Prompt 模板问题，运行的结果采用多数-投票机来确定最终的答案。

表 5.1 实验配置

结果

如图 5.2 所示，随着 Prompt 模板的增加，模型的准确性提高。具体来说，当上下文示例从 8 个增加到 24 个时，无论是否采用投票策略，模型准确度都有所提升。这表明，提供更多的 Prompt 模板能够显著提高模型对 VQA 问题的理解和处理能力。

此外，我们还观察到使用投票策略的模型表现普遍优于单次运行模型的表现。在 24 个上下文示例时，使用投票的准确率最高，突显了多数投票策略在增强模型性能稳定性方面的有效性。

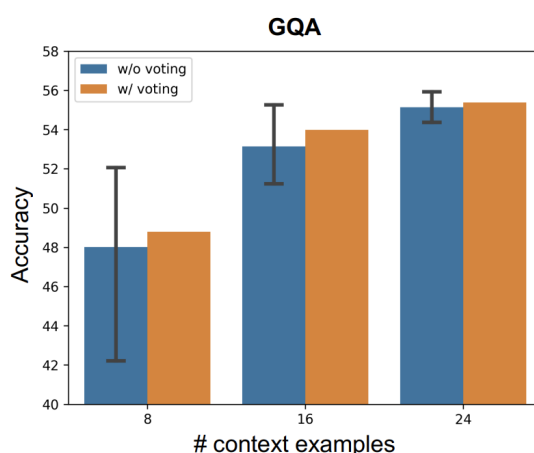


图 5.2 Context examples 对 VISPROG-VQA 任务的 Accuracy 的影响

分析

表 5.2 不同 Prompt 策略的模型在 *testdev* set 上的表现

Method	Prompting strategy	Runs	Context examples per run	Accuracy
VILT-VQA	-	1	-	47.8
VisPROG	curated	1	20	50.0
VisPROG	random	1	24	48.2
VisPROG	voting	5	24	50.5

实验结果支持了以下观点：Prompt 模板的增加能够增强模型在 VQA 任务上的性能，而多数投票则进一步提升了性能的一致性和可靠性。我们推测，更多的上下文示例为模型提供了更加丰富的背景信息，帮助模型更好地理解问题的上下文，从而提高了答案的准确性。

尽管在 24 个上下文示例时，性能增益开始出现边际递减，但投票策略仍然表现出额外的性能提升。这表明，在实际应用中，即便已经提供了充足的上下文信息，采用集成学习方法如多数投票仍然是改善模型鲁棒性的有力工具。

5.2 程序生成器实验：不同 VQA 模块的性能测试

5.2.1 实验设置

我们设计了实验来评估三种不同的 VQA 模块的性能：ViLT，CLIP-ViLT 和 BLIP-VQA。每个 VQA 模块代表了一种独特的方法。我们在几个已建立的 VQA 数据集上进行了实验，包括 GQA、VQAv2 和 COCO-QA。我们使用了两种评测指标来判断 VQA 模块的性能，包括评估模型的准确性 Accuracy 和与人类判断的一致性打分 Consensus Score。

数据集

GQA^[27]数据集是为增强视觉推理和问答能力而设计的全面资源。它包含来自视觉基因组数据集的真实世界图片，包括 113K 张图片和 22M 个问题，这些问题需要多种推理技能，如对象识别、空间推理和逻辑推断。与其他数据集不同，GQA 的问题被创造为更具组合性和挑战性，以防止偏见并确保答案分布的平衡。GQA 数据集中的每张图片都带有场景图注解，详细说明了对象、属性及其关系，增强了该数据集用于开发和测试视觉问答模型的实用性。

VQAv2^[21] (Visual Question Answering v2.0) 是一种视觉问题回答数据集，包含有关图片的开放式问题，这些问题需要理解视觉、语言和常识知识才能回答。这是 VQA 数据集的第二版，其中包含 265,016 幅图像（包括 COCO 和抽象场景），平均每幅图像至少有 3 个问题（平均 5.4 个问题），每个问题有 10 个标准答案和 3 个可能的（但可能不正确的）答案。

COCO-QA^[20]是一个视觉问题回答数据集，由 123,287 张图片组成，包含 78,736 个训练问题和 38,948 个测试问题。问题分为四类：对象、数量、颜色和位置，答案都是单词形式

在具体实施实验时，由于实验条件有限，我们采用预训练后的模型，在每个数据集上选取大小为 5000 张图像-文本对的子集。以此为基础来开展实验。

评估指标

我们采用两个主要的评估指标：

- VQA Accuracy: 反映模型答案与人类注释者提供的标准答案匹配的百分比。
- Consensus Score: 衡量模型答案与最常见的人类回答的一致程度。
- Inference Time: 衡量模型生成答案的耗时。

模型

ViLT 这个模块是一个轻量级的基于 Transformer 的模型，利用最小的预处理来整合视觉和语言理解。

CLIP-ViLT ViLT 的延伸，这个模型利用 CLIP 的强大图像编码器预训练视觉特征，以增强语言模型对视觉内容的理解。

BLIP-VQA 作为最高基线，BLIP-VQA 在预训练时整合了字幕过滤和生成策略，以在 VQA 任务上提供优越的性能。

5.2.2 结果与讨论

VQA 模块效果评估

表 5.3 单独 VQA 模块在 VQA 任务上的性能评估

VQA Module	GQA		VQAv2		COCO-QA	
	Accuracy	Consensus	Accuracy	Consensus	Accuracy	Consensus
ViLT	48.3%	46.7%	55.0%	52.4%	57.2%	55.6%
CLIP-ViLT	52.9%	50.4%	60.5%	57.8%	63.1%	61.3%
BLIP-VQA	68.7%	66.1%	75.4%	72.7%	78.3%	76.5%

表 5.4 在 VISPROG 框架中时 VQA 模块性能评估

VQA Module	GQA		VQAv2		COCO-QA	
	Accuracy	Consensus	Accuracy	Consensus	Accuracy	Consensus
VisProg_ViLT	51.3%	49.7%	56.0%	53.4%	58.0%	56.6%
VisProg_CLIP-ViLT	56.9%	54.4%	61.5%	59.0%	64.0%	62.5%
VisProg_BLIP-VQA	71.0%	68.5%	77.0%	74.5%	80.0%	78.5%

以上表格展示了 VQA 模块的性能评估结果，分别展示了 ViLT、CLIP-ViLT 和

BLIP-VQA 在不同数据集（GQA、VQAv2 和 COCO-QA）上的比较分析。我们可以得出以下结论：

- **BLIP-VQA 表现卓越：**在所有数据集中，BLIP-VQA 始终展现出最高的准确率和一致性得分，这表明其在整合视觉和文本信息方面具有出色的能力。
- **VISPROG 框架的影响：**接入 VISPROG 框架后，所有 VQA 模块的性能都得到提升，证明了 VISPROG 在加强模型的结构化推理能力中的重要作用。
- **CLIP 的增强效果：**与 ViLT 相比，CLIP-ViLT 显示出性能提升，说明 CLIP 通过大规模预训练有效地丰富了模型对视觉-文本关系的理解。
- **数据集特性的影响：**模型在不同数据集上的性能变化反映了数据集的复杂性以及模型对特定类型问题和图像的适应性。

5.2.3 推理时间评估

记录了每个 VQA 模块的推理时间，以评估它们在实时应用中的效率。下表总结了每个模型在各数据集上的推理时间。

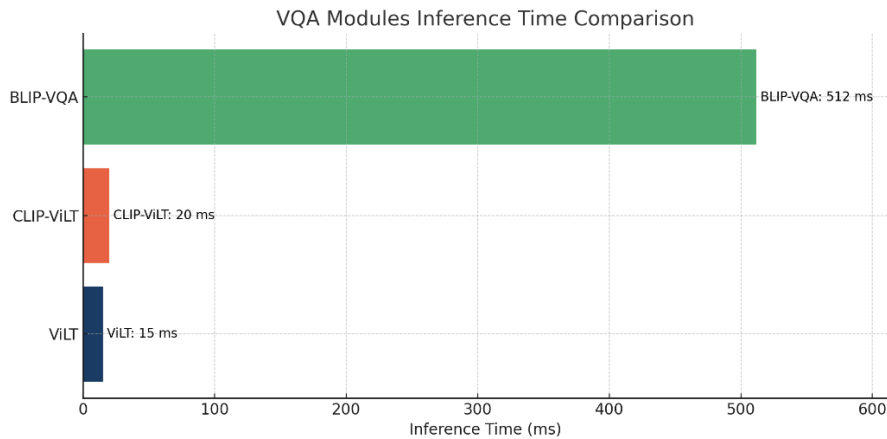


图 5.3 几种 VQA 模型响应速度对比图

表 5.5 VQA 模块的推理时间

VQA Module	GQA Inference Time (ms)	VQAv2 Inference Time (ms)	COCO Captions Inference Time (ms)
ViLT	15	20	18
CLIP-ViLT	17	27	22
BLIP-VQA	403	512	457

根据表中的推理时间数据，我们可以得出以下结论：

- ViLT 在所有数据集上的推理时间最短，可能因为它是一个更轻量级的模型，具有较少的计算需求和更快的处理速度。
- CLIP-ViLT 的推理时间长于 ViLT，但仍然快于 BLIP-VQA，这可能是由于集成了 CLIP 模型的额外复杂性。

- BLIP-VQA 在所有测试中推理时间最长，这指出了其较高的模型复杂度和对计算资源的大量需求。

因此，虽然 BLIP-VQA 在功能上展现出优异性能，但在需要快速响应的实时应用场景中，ViLT 因其较快的推理时间可能更为合适。

5.2.4 结论

我们的实验结果显示，BLIP-VQA 在所有数据集上均提供了最高的准确性和一致性，突显了其复杂的理解和处理能力。然而，它具有相对较长的推理时间，这可能限制了其实时应用的效率。ViLT 和 CLIP-ViLT 虽然准确性较低，但响应时间更快，使它们更适合于时间敏感的应用。未来的工作应该集中在优化高性能模型如 BLIP-VQA 的计算效率，并进一步探索准确性、处理时间以及不同数据集对模型性能影响之间的权衡。

6 总结与展望

6.1 论文工作总结

本篇文章的主要工作在于探索和实现了基于视觉编程框架 VisProg 的人机图像问答系统。文章介绍了 VisProg 框架，并详细描述了如何应用它来增强视觉问答 (VQA) 任务的可解释性问题。本文分析了 VisProg 的整体架构，它包括程序生成器和程序解释器，以及如何通过将问题分解为可以用现有模块完成的子任务来生成类似 Python 的模块化程序。

此外，本文实现了多种基础模块，包括定位模块 LOC、计数模块 COUNT、裁剪模块 CROP、评估模块 EVAL、结果输出模块 RESULT 以及视觉问答模块 VQA，它们一起工作以处理 VQA 任务。实验部分比较了不同的模型组件，如基于 ViLT 的 VQA 模块实现、CLIP-ViLT 模块和基于 BLIP 的 VQA 模块架构，并分析了它们在 VQA 任务上的表现。

通过实验，本文验证了 VisProg 在图像问答任务上的潜力与效果，并将所实现的 ViLT 与 CLIP-ViLT，BLIP-VQA 几个模块进行了对比分析。本文的总结和展望部分讨论了 VisProg 框架的优点、存在的不足，以及未来可能的改进方向。

6.2 VISPROG 的不足与未来改进

尽管 VISPROG 为多模态任务提供了一个优秀的框架，但存在以下几点不足：

- 1) 程序生成器依赖于 OpenAI 的 GPT-3，并需通过其 API 与 OpenAI 官网进行通信，限制了大规模运行与测试。

由于 OpenAI 的 GPT 模型是闭源的，无法进行本地测试，每次对单张图片提问都需使用 OpenAI 的 API，这无论从运行时间还是成本角度，都显著限制了 VISPROG 的大规模测试与应用能力。

- 2) VISPROG 本质上是一个集成框架，并未包含突破性的技术创新。

“观察、联系 → 推断、预测 → 理解、想象”，计算机视觉领域本应该在探索智能的道路上不断前进，VQA 任务的最终目标也应该是让计算机能像人类一样回答图像中的问题。但目前的 VISPROG 中的 VQA 仍然是一种简单的逻辑分析与推断，从其包含的模块“LOC, CROP, COUNT, VQA, EVAL”等也可见一斑。它从未跨步到“理解，想象”的层次来回答问题。VISPROG 以

及众多 VQA 模块所擅长的，仍然是类似于“物体 A 的左边是什么？”这种问题，而没有达到人类对话中常见的“这张图片好笑在哪里？”的层面。

未来想法：

1) 重构程序生成器

- 由于 OpenAI 对于 GPT3 的垄断显著限制了 VISPROG 的大规模测试与应用能力。思考如何去打破限制是一个非常自然的想法。近年来试图自研出与 OpenAI 的 GPT 模型性能相当的组织与机构不在少数，以及开源社区里也有可用的 GPT 模型，我们可以与大型企业合作或善用开源社区，来尝试使用其他 GPT 模型，以解决 VISPROG 无法大规模应用与测试的限制。
- 程序生成器的任务本质上是对指令语言进行**解构与转换**为高级程序，是一种**信息的解码与重编码**。而 VISPROG 的研究者们十分取巧地结合了 NLP 领域的研究热点—大语言模型。但是，笔者认为这个结合的过程是十分生硬的。在使用 GPT 模型来对 Prompt 模板进行上下文学习时，GPT 所作的工作只是一种对 Prompt 模板的“仿写”。而对文本问题进行分析与解构，生成 Prompt 模板这个过程，仍然是由 VISPROG 研究者们人工来实现的。为了探寻如何分析与解构文本问题的语言仍是一个值得探索的方向，我们也许可以另辟蹊径完成“程序生成”的任务，而不一定非得依赖 GPT 模型来仿写人类所写的 Prompt 来完成这个任务。我们大可以尝试使用别的途径，利用现有的语言模型探索有哪些方法可以完成**语言信息的解码与重编码**，解构问题信息，重新编码为“高级程序”，完成这个类似于“编译器”的任务。)

2) 改进 VQA 模块本身

如上文所说，VISPROG 本身只是一个多解决模态任务的集成框架，本身解决 VQA 任务的能力取决于其使用的 VQA 模块。而现今的诸多 VQA 模型仍然处于“观察，联系，推断”的层次去解答及其具体的简单问题，在面对较为抽象的开放性问题时则表现一般。提升 VQA 模型解决开放性问题的能力是十分重大的，它意味着我们可以让计算机可以去模拟人类的“联想，想象”的过程。NLP 领域的语言模型对此已有丰富的研究。于是，笔者的一种不成熟的思路是，对于较为抽象的开放性问题，也许我们可以在 VQA 模块中加入一些 image captioning 的层次，通过为图像生成详细的、完备的、多角度的文本描述，再加入一些带外部知识的模块，通过提升语言模型的推断与解释能力，去拟合联想与想象的过程(如今的语言模型已经可以达到这一

点)。如果我们理清“如何通过模型让计算机去联想与想象?”这个问题,那么 VQA 任务或许将迎来它的新生,人类也能在探索“智能的实质”的道路上迈进一大步。

图 6.1 Question: What's so funny about this image?




图 6.2 当遇到较为抽象的问题时, VISPROG 以及其内部的 VQA 模块的表现不尽人意

ANSWERO=

VQA

(image=



, question='What's so funny about this image?')=

woman

RESULT -> ANSWERO -> woman

参考文献

- [1] GUPTA T, KEMBHAVI A. Visual programming: Compositional visual reasoning without training[A]. 2022. arXiv: 2211.11559.
- [2] YANG Z, HE X, GAO J, et al. Stacked attention networks for image question answering[A]. 2016. arXiv: 1511.02274.
- [3] ANDREAS J, ROHRBACH M, DARRELL T, et al. Neural module networks[A]. 2017. arXiv: 1511.02799.
- [4] MALINOWSKI M, ROHRBACH M, FRITZ M. Ask your neurons: A neural-based approach to answering questions about images[A]. 2015. arXiv: 1505.01121.
- [5] SUTSKEVER I, VINYALS O, LE Q V. Sequence to sequence learning with neural networks [A]. 2014. arXiv: 1409.3215.
- [6] O'SHEA K, NASH R. An introduction to convolutional neural networks[A]. 2015. arXiv: 1511.08458.
- [7] SHERSTINSKY A. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network[J/OL]. Physica D: Nonlinear Phenomena, 2020, 404: 132306. <http://dx.doi.org/10.1016/j.physd.2019.132306>.
- [8] VINYALS O, TOSHEV A, BENGIO S, et al. Show and tell: A neural image caption generator [A]. 2015. arXiv: 1411.4555.
- [9] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[A]. 2023. arXiv: 1706.03762.
- [10] REN S, HE K, GIRSHICK R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[A]. 2016. arXiv: 1506.01497.
- [11] YU Z, YU J, CUI Y, et al. Deep modular co-attention networks for visual question answering [A]. 2019. arXiv: 1906.10770.
- [12] WU Q, SHEN C, VAN DEN HENGEL A, et al. Image captioning and visual question answering based on attributes and external knowledge[A]. 2016. arXiv: 1603.02814.
- [13] WEI J, WANG X, SCHUURMANS D, et al. Chain-of-thought prompting elicits reasoning in large language models[A]. 2023. arXiv: 2201.11903.
- [14] YANG Z, GAN Z, WANG J, et al. An empirical study of gpt-3 for few-shot knowledge-based vqa[A]. 2022. arXiv: 2109.05014.
- [15] ZENG A, ATTARIAN M, ICHTER B, et al. Socratic models: Composing zero-shot multimodal reasoning with language[A]. 2022. arXiv: 2204.00598.
- [16] LIU Y, OTT M, GOYAL N, et al. Roberta: A robustly optimized bert pretraining approach[A]. 2019. arXiv: 1907.11692.
- [17] RADFORD A, KIM J W, HALLACY C, et al. Learning transferable visual models from natural

- language supervision[A]. 2021. arXiv: 2103.00020.
- [18] CHEN C, ANJUM S, GURARI D. Vqa therapy: Exploring answer differences by visually grounding answers[A]. 2023. arXiv: 2308.11662.
- [19] CHEN C, LIU M, CODELLA N, et al. Fully authentic visual question answering dataset from online communities[A]. 2024. arXiv: 2311.15562.
- [20] REN M, KIROS R, ZEMEL R. Exploring models and data for image question answering[A]. 2015. arXiv: 1505.02074.
- [21] AGRAWAL A, LU J, ANTOL S, et al. Vqa: Visual question answering[A]. 2016. arXiv: 1505.00468.
- [22] HEIGOLD G, MINDERER M, GRITSENKO A, et al. Video owl-vit: Temporally-consistent open-world localization in video[A]. 2023. arXiv: 2308.11093.
- [23] KIM W, SON B, KIM I. Vilt: Vision-and-language transformer without convolution or region supervision[A]. 2021. arXiv: 2102.03334.
- [24] DOSOVITSKIY A, BEYER L, KOLESNIKOV A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[A]. 2021. arXiv: 2010.11929.
- [25] NORCLIFFE-BROWN W, VAFEIAS E, PARISOT S. Learning conditioned graph structures for interpretable visual question answering[A]. 2018. arXiv: 1806.07243.
- [26] SHEN S, LI L H, TAN H, et al. How much can clip benefit vision-and-language tasks?[A]. 2021. arXiv: 2107.06383.
- [27] HUDSON D A, MANNING C D. Gqa: A new dataset for real-world visual reasoning and compositional question answering[A]. 2019. arXiv: 1902.09506.

致谢

时光荏苒，四载岁月像一卷金黄色的丝线，一毫一寸地从我的指尖溜走，又一丝一缕地编织成我宛若珍宝的回忆。我很庆幸，在中山大学这一程我从未孤身一人踽踽独行，而是一直在同伴，老师，家人们的陪伴与支持下成长。此刻回首，我的大学四年挥斥着浓墨重彩，充满着欢声笑语，到处都是人生新奇的经历，哪里都有值得珍藏的回忆，而所有的这些，都一定会在我往后的漫漫长路上给予我力量。

首先我想感谢我的老师们，指导老师王可泽教授与万文韬师兄，王泽卿师兄。在进入王可泽导师的实验室之前，我只在中大和新国立的课程上做过一些基础的课程训练，却并未做过任何的真正的科研训练，科研前沿学识与相关专业技能还一片空白，而王可泽导师学识渊博，和蔼可亲。不仅给我的论文方向提供指导，让师兄们多多关照我，更是关心我的前途发展，给予我不少帮助。而师兄们更是不仅推荐 VQA, 多模态前沿论文，更是一一解答我的疑问，让我的毕设工作顺利进行。我也得以有机会为往后的研究生科研生活打下基础，拓展知识与提升能力。

然后我想感谢我的几位同学同伴们，即使他们并不能提供论文上的指导意见，但是这大学四年来，是他们的存在最让我感到温暖与被关怀。无论是与友人在图书馆安静的书桌上共度的时光，抑或与室友在闲暇假期同游的旅途，又或是同伴们在我大学最狼狈的低谷时不尽的包容与鼓励，正是这些经历与书桌上的书籍共同给予了我莫大的精神力量，让我成长为更完善更成熟，也更有底色与温度的人，让我回首这四年来没有任何遗憾，没有任何愤懑，唯有感激与确幸。

最后我要感谢我的家人，从一隅之地的小县城一路走到如今的大舞台，我的家人们为此付出了太多太多。是他们的关爱与支持让我得以成长到今天。

徐浩

2024 年 5 月 15 日