arXiv:2603.15997v1 [cs.MM] 16 Mar 2026

# 视觉集合程序合成器

Zehua Cheng[1,2], Wei Dai[2], Wenhu Zhang[3],Thomas Lukasiewicz[4,1],和 Jiahao Sun[21]牛津大学计算机科学系[2]FLock.io[3]香港科技大学计算机科学与工程系[4]维也纳工业大学逻辑与计算研究所 zehua.cheng@cs.ox.ac.uk

**摘要—用户将手机对准超市货架并询问“哪种汽水含糖量最低？”，这对当前的视觉人工智能助手构成了严峻挑战。这类查询不仅需要目标识别，还需要明确的基于集合的推理，例如筛选、比较和聚合。标准的端到端多模态大语言模型往往无法胜任此类任务，因为它们缺乏明确的组合逻辑机制。我们提出将视觉推理视为视觉程序合成：模型首先生成一个符号程序，再由一个基于视觉场景的独立执行引擎执行。我们还推出了 Set-VQA，这是一项专为评估基于集合的视觉推理而设计的新基准测试。实验表明，我们的方法在复杂推理任务上的表现显著优于最先进的基线，行为更加系统、透明，同时大幅提升了答案准确率。这些结果表明，程序驱动的推理为黑箱视觉语言推理提供了一种有原则的替代方案。**

索引词—**多模态推理、视觉程序合成、基于集合的视觉问答基准测试。**

## 一、引言

现代多模态大语言模型（MLLMs）在视觉语言理解方面取得了显著进展，使系统能够回答基于图像的问题 [1], [2]。然而，对于需要对视觉对象集合进行显式逻辑推理，而非简单识别的查询，它们仍然难以应对。例如，回答“货架上哪种汽水含糖量最低？”需要识别所有汽水，获取每件商品的含糖量，并选出最小值。这种基于集合的视觉推理需要针对动态构成的对象集合进行结构化、逐步的操作，超出了传统视觉问答的范畴。

现有多模态大语言模型通常通过单一的端到端推理过程来解决这类问题。推理过程隐含在模型内部，因此难以进行错误诊断，且常导致模型表现脆弱，例如筛选或比较错误 [3]。即使是工具增强系统，也仍以自然语言形式表达推理，因此难以验证中间步骤在逻辑上是否正确。因此，当前方法既缺乏透明度，也无法对推理过程进行可靠控制。

在这项工作中，我们将基于集合的视觉推理表述为一个视觉程序合成问题。模型并非直接预测答案，而是由多模态大语言模型生成一个可执行的符号程序，用以表示推理过程。如图1所示，另一个执行引擎



**Query:** Of the noodles, which one has the most calories?
**LLAVA-1.6-34B (End-to-End):** "The noodles with the most calories is the one on the far right, the SUPER NOOLES" (Incorrect and Hallucinated)
**Our Model (CASTER):**
**Generated Program:** SELECT(MAX(CALORIES), FILTER(objects, class='noodles'))
**Execution - Step 1 (FILTER):** The engine detects five objects classified as **'noodles'** and retrieves their calorie information. (This retrieval is handled by our execution engine, which first attempts OCR on the packaging. If OCR fails or the attribute is not present, it uses the recognized object name—e.g., 'Soba'—to query a simulated external knowledge source, which is a key-value store mapping item names to nutritional info.)
**Execution - Step 2 (MAX & SELECT):** The engine compares the calorie counts (210, 220, 190, 240, 200) and identifies 240 as the maximum.
**Final Answer:** "The 'Soba' has the most calories with 240." (Correct and Verifiable)

图1. VisualSet 程序员使用 CASTER 的示例，与端到端多模态大语言模型进行比较。

随后将程序落实到视觉场景中，并检索所需的属性，例如 SELECT(MIN(sugar), FILTER(objects, class='soda'))。这种分离使模型能够通过筛选、排序、计数和比较等显式集合操作进行推理。与纯粹的端到端建模相比，这种设计使推理流程可验证，因为每个中间操作都会产生具体输出。它还支持更可靠的错误归因，使我们能够区分感知失败与逻辑错误。此外，它也具有可解释性，因为可以逐步检查和评估每个步骤。

除了提高任务准确率外，这一表述还为多模态推理提供了统一视角，将视觉感知与符号计算联系起来。在实践中，它使模型能够处理日益复杂的查询，而无需依赖脆弱且隐式的神经启发式方法。我们认为，这是构建可信且可审计的视觉人工智能助手的一个进步。

为了进一步提高程序生成质量，我们提出**组合式行动者集合论奖励（CASTER）**，这是一种稠密奖励函数，用于评估中间程序组件的正确性，而非仅仅依赖最终答案准确率。这种设计提供了更丰富的学习信号，并鼓励模型构建逻辑连贯的程序，从而产生更加稳定且可解释的推理行为。

总之，我们的贡献如下：

- 我们将**基于集合的视觉推理**形式化为多模态大语言模型面临的一项关键但尚未得到充分关注的挑战，它要求进行显式的

视觉对象集合上的组合逻辑推理。

- 我们提出了**视觉集合程序合成器（VSPS）**，这是一种新颖的神经符号框架，其中多模态大语言模型充当视觉程序合成器，通过合成并执行形式化程序，可靠地回答视觉查询。
- 我们推出了**Set-VQA** 基准测试，这是一个专为训练和评估这项能力而设计的新数据集，其中包含第一人称视角图像，以及与之配对的基于集合的查询及其对应的真实标注程序，涵盖广泛的逻辑组合。
- 大量实验表明，我们的程序化方法显著优于最先进的端到端多模态大语言模型基线，大幅减少逻辑错误并提高最终答案的准确率。

## II. 相关工作

**基于隐式结构的端到端视觉推理**。在基于大语言模型的模型兴起之前，组合式视觉推理通常通过整体式深度网络来实现，这些网络以隐式方式建模关系。这类模型包括用于构建视觉实体与场景文本之间图关系的图卷积网络（GCN），[3], [4],以及其他组合网络；后者将称为“神经模块”的独立深度网络结合起来，以回答有关图像的查询 [5], [6]。尽管这些模型能够有效表示实体间的相关性，但其组合过程不透明，并通过“整体式”模型完成。对于计数或依据条件进行筛选等符号运算而言，这一组合过程很难做到足够精确，因此容易产生不合逻辑的错误。我们提出的视觉集合程序模型则确保组合过程能够通过其生成的形式化程序变得透明且可验证。

**通过程序合成进行视觉推理**我们的工作与一个新兴研究方向相关，即神经符号视觉推理，该方向将视觉推理视为代码合成问题。在这种方法中，预训练大语言模型充当程序员，编写可执行代码来调用特定的视觉模型以回答问题。近期方法如 VisProg [5]和 ViperGPT [7] 已展示了大语言模型如何利用预训练大语言模型，在零样本或少样本设置下为复杂且组合式的视觉问题编写代码。近期研究着重通过稳健的层次化代码合成模型来增强代码生成过程 [8]。尽管我们同样基于这一洞见，使用可执行程序分解推理过程，但本研究在概念上有三项创新。

## III. 方法

### A. 问题形式化

核心挑战在于训练一个多模态大语言模型，将多模态源——视觉线索 $I$ 和自然语言问题 $Q$——转换为形式化的可执行代码 $P$。这段可执行代码需要表达用于查询的组合逻辑。目标是开发一种策略 $\pi_\theta$，使其能够为给定的 ($I$， $Q$) 生成代码，从而在执行后得到正确答案$A_{gt}$。这绝非儿戏，而是需要精通程序语言语法，并能从逻辑运算的角度理解自然语言查询。为具体说明，我们在附录 A 中定义了程序语言和执行引擎。

### B. 组合式行动者集合论奖励（*CASTER*）

图 2 展示了所提出的 CASTER 的整体结构。针对具有可验证结果的程序（例如机器学习程序），一个近期且直接相关的基线方法是可验证奖励强化学习（RLVR）[9], [10]。该方法通过优化多模态大语言模型，使其成功完成任务，而不是仅依赖监督微调（SFT）来模仿语法正确的程序。RLVR 将多模态大语言模型建模为策略网络，通过最大化基于正确答案评估的奖励来生成程序。

RLVR 的核心是一种二元、可验证的奖励函数$\mathcal{R}$，用于比较生成的答案 $A_{gen}$ 与真实答案 $A_{gt}$:

$$\mathcal{R}(A_{\text{gen}}, A_{\text{gt}}) = \begin{cases} 1 & \text{if } A_{\text{gen}} = A_{\text{gt}} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

训练目标是最大化训练数据分布上的期望奖励$J(\theta)$:

$$J(\theta) = \mathbb{E}_{(I,Q,A_{\text{gt}})\sim\mathcal{D}} \left[ \mathbb{E}_{P\sim\pi_\theta(\cdot|I,Q)} [R(\text{Exec}(P, I), A_{\text{gt}})] \right] \tag{2}$$

可以使用 REINFORCE 等策略梯度方法来优化这一目标。目标函数的梯度为:

$$\nabla_\theta J(\theta) = \mathbb{E}_{P\sim\pi_\theta} [R(\text{Exec}(P, I), A_{\text{gt}}) \nabla_\theta \log \pi_\theta(P|I, Q)] \tag{3}$$

为防止训练过程中出现此类失衡，并缓解遗忘先前所学内容的问题（灾难性遗忘），人们探索了近端策略优化（PPO） [11], 、组相对策略优化（GRPO） [12],等信赖域方法。这些方法会加入一个涉及 KL散度 的惩罚项，使新策略 $\pi_\theta$ 与预先计算的基线策略$\pi_{ref}$保持接近，例如通过监督微调优化得到的策略。

RLVR 基线存在两个主要缺陷，导致其难以高效学习复杂的组合式程序：（1）**稀疏奖励**：二元奖励机制无法提供任何关于部分正确程序的信息。逻辑上不正确的程序，以及逻辑上正确但存在细微缺陷的程序，获得的奖励都同为零。（2）**信用分配问题**：全局奖励函数无法指出复杂计算中的哪一部分出错。算法无法区分错误究竟出在筛选步骤还是最小值步骤。

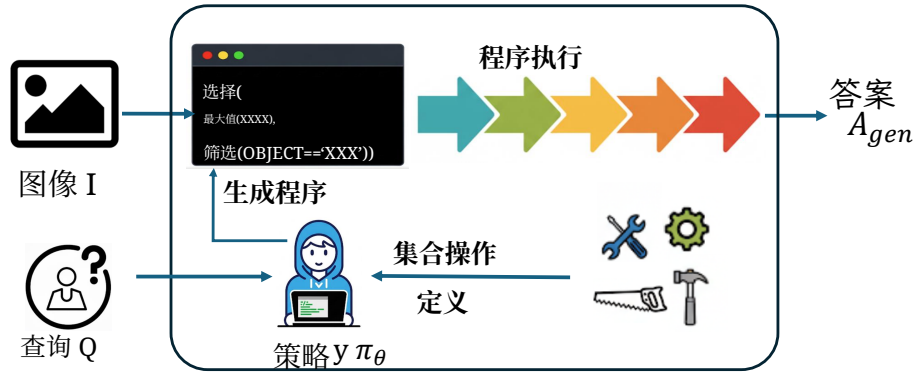



图 2. 视觉集合程序合成器是一种通过显式引入结构化、机器可读的程序语言来解决复杂视觉查询的框架。多模态大语言模型生成程序，随后由程序执行引擎运行。该引擎依托感知栈（例如目标检测、光学字符识别）和知识库，将程序中的集合运算（例如筛选、选择）落地到视觉场景中，并检索所需属性。

为应对这些挑战，我们提出了**组合式行动者集合论奖励（CASTER）**，这是一种新的神经符号学习技术，可在训练过程中提供稠密且语义丰富的奖励信号。关键的是，CASTER 并非从头开始学习；它会对经监督微调初始化的策略进行显式优化，为程序生成提供必要的语法先验。CASTER 的核心创新在于，将奖励构建为抽象语法树（AST）上的最大权二分匹配问题，并通过杰卡德指数提供稠密的语义反馈。这直接解决了上述稀疏奖励和信用分配问题。CASTER 不会只关注是否正确回答问题，而是会根据模型推导所生成程序逻辑结构时的组合正确性来给予奖励。具体而言，它会执行所有子程序，并在这些子程序的每个执行步骤中，将其生成的对象集合与真实程序本应生成的集合进行比较。如此一来，它便能将学习反馈从稀疏奖励转变为稠密奖励。

CASTER 框架将程序解析为抽象语法树（AST），并通过比较相应子程序的语义输出，计算稠密奖励。该算法首先由行动者 $\pi_\theta$ 生成候选程序 $P$，并将 $P$ 和真实程序 $P_{gt}$ 分别解析为对应的抽象语法树 $T$ 和 $T_{gt}$。抽象语法树中的每个节点 $n$ 都表示一个子程序 $P_n$。

接下来，我们对**子程序进行语义落地**。我们使用子程序执行器 Exec_sub($P_n$, $I$) 来评估节点 $n \in N(T)$ 和 $n_{gt} \in N(T_{gt})$，其中 $N(\cdot)$ 是抽象语法树中的节点集合。执行后会得到经视觉落地的对象标识符中间集合 $O_n$ 和 $O_{n_{gt}}$，将符号化程序组件映射到具体的视觉证据。

通过这些集合，我们可以借助**最优对齐计算结构奖励**。生成的子程序节点 $n_i \in N(T)$ 与真实值节点 $n_j^{gt} \in N(T_{gt})$ 之间的语义相似度，定义为它们所生成对象集合的杰卡德指数 $J$：

$$S_{ij} = J(O_{n_i}, O_{n_j^{gt}}) = \frac{|O_{n_i} \cap O_{n_j^{gt}}|}{|O_{n_i} \cup O_{n_j^{gt}}|} \tag{4}$$

此计算会生成一个大小为$|N(T)| \times |N(T_{gt})|$的相似度矩阵$\mathbf{S}$。程序 $P$的总奖励可以简洁地表述为一个**最大权二分匹配**问题，该问题基于由两个节点集合及相似度矩阵$\mathbf{S}$定义的图。设 $M$ 为两个抽象语法树中节点之间的一种匹配。最优匹配 $M^*$ 是使总相似度最大化的匹配：

$$M^* = \underset{M}{\operatorname{argmax}} \sum_{(n_i, n_j^{gt}) \in M} \mathbf{S}_{ij} \tag{5}$$

这个问题可以使用匈牙利算法等方法高效求解。程序 $P$ 的稠密奖励是该最优匹配的总权重，我们将其定义为势函数 $\Phi(P, P_{gt})$，用于衡量它与 $P_{gt}$之间最大程度的结构和语义对齐：

$$R_{\text{CASTER}}(P) = \Phi(P, P_{gt}) = \sum_{(n_i, n_j^{gt}) \in M^*} S_{ij} \tag{6}$$

这种匹配型奖励的目标是通过在程序语法匹配、且与其子步骤语义相同的情况下给予奖励，来鼓励语义鲁棒性。尽管如此，我们发现这种奖励形式存在一些问题。它未能强制匹配节点之间进行类型匹配（例如，如果 SORT 节点的输出相同，就应该允许匹配），也未能惩罚树 $T$中存在的冗余节点。

这种奖励不再是稀疏的；对于程序中每个表述正确且有依据的子部分，它都会给予部分得分，从而直接解决信用分配问题。这种表述受到基于势函数的奖励塑形 [13], [14], 的启发，能够有效引导智能体穿越复杂的状态空间。

最后，我们将这种稠密奖励应用于**策略优化**。CASTER 算法使用 GRPO 进行训练；在这种情况下，GRPO 效果非常好，因为它无需学习单独的批判神经网络。对于从 $\pi_\theta$ 采样得到的一批 K 个程序 $P_k$ ，其 CASTER奖励 $R_{\text{CASTER}}(P_k)$ 计算如下。这些奖励可以进行标准化以得到优势估计 $A_k$，例如通过 Z 分数标准化。GRPO 的目标是优化批次中相对奖励更高的程序的概率：

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}\left[\sum_{k=1}^{K} A_k \cdot \log \pi_\theta(P_k|I,Q) + \beta \cdot D_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right] \tag{7}$$

请注意， $\beta$ 表示正则化系数，$\pi_{ref}$ 表示经监督微调初始化的策略，有助于稳定训练。在[15] 中开发的神经符号反馈循环，使多模态大语言模型能够学习逻辑分解和视觉定位规则，其样本效率和鲁棒性显著优于稀疏奖励 RLVR 基线。

## IV. 实验

### *A.* 实验设置

为了评估我们的框架在集合推理任务中的表现，以及其对不同场景的泛化能力，我们使用新开发的模型作为基准，并采用另外两组由视觉问答数据集改编而来的基准。在附录B中，我们介绍了超参数值以及有关基线方法的更多信息。

1) **Set-VQA（我们的方法）**：用于训练的主要基准与评估。它包含 11 万条数据（6 万条合成数据，5 万条真实世界）图像（例如商店货架、桌面、街道场景），以及与之配对的复杂自然语言查询，标准可执行程序和最终答案。
这些查询旨在均衡覆盖各类集合运算，包括筛选、计数、排序（含最小值/最大值）、存在及其组合。
数据集划分为 8 万个训练样本、1 万个验证样本和 2 万个测试样本。Set-VQA 的详细信息见附录 C。

2) **GQA-Set**：为了评估模型对新视觉领域和问题分布的泛化能力，我们基于以组合式问题和场景图而闻名的 GQA 数据集 [16],创建了一个具有挑战性的测试划分。我们挑选了五千个与对象集合推理相关的问题（例如，“有多少红色物体位于小球的左边？”），并对其真实答案程序进行了人工标注。

3) **PTR-Count**：为了构建评估测试集，我们从基于部件的视觉推理（PTR） [17]中收集了 3,000 个样本。问题侧重于根据对象的属性进行计数（例如：“有多少把椅子有四条腿？”）。该测试将评估在三维空间中推理对象部件级聚合的能力。相关程序已同时完成标注和验证。

为了专门评估组合泛化（表 III），我们确保 8 万个训练集样本明确排除了以下五种复杂的嵌套程序结构，这些结构仅出现在测试集中：

- 任务 1：COUNT(SORT(...))（例如，“按价格排序后，数一数这些物品。”）。
- 任务 2：排序(筛选(...))（例如：“将红色物品排序”）
- 任务 3：计数(筛选(对象, 属性 != 值))(例如：“有多少件物品不是红色的？”)
- 任务 4：选择(最大值(attr1), 排序(attr2, ...))（例如：“在最便宜的物品中，哪一种的卡路里最多？”）
- 任务 5：计数(筛选(对象, 关系='左侧', ...))（例如：“数一数瓶子左侧的罐子”）

### *B.* 评估

我们使用评估指标来衡量中间程序和最终答案，从而进行全面评估。

1) **程序准确率（PA）**：生成程序与真实程序字符串完全匹配的比例。该指标用于评估合成器生成逻辑正确计划的能力。对于不生成程序的模型，其得分记为 N/A。

2)**最终答案准确率（AA）**：执行后生成答案与真实答案完全匹配的比例。这是衡量任务整体成功与否的主要指标。

### *C.* 主要结果

视觉集合程序合成器（VSPS）框架显著优于所有基线方法，包括端到端模型和少样本程序合成器。最佳配置（CASTER-Qwen）在 Set-VQA 上达到 93.8% 的平均准确率（AA），比最强的单体基线（Qwen2.5-VL-72B，38.6% AA）绝对提升了 55 个百分点以上。它还超过了竞争力最强的工具增强基线（51.2% AA），这表明，仅仅让模型能够使用外部感知或知识源是不够的。真正重要的是能够以可执行程序的形式明确表达推理，而不是依赖隐式神经推理。

在两种骨干网络架构上，CASTER学习算法始终展现出更优的性能。采用通义千问骨干网络时，与RLVR相比，其程序准确率（PA）提高了超过15个百分点（95.1%对79.5%），最终答案准确率（FAA）提高了超过7个百分点（93.8%对86.1%）。尽管RLVR取得了相对较高的最终答案准确率（AA）（86.1%），但这是在其程序准确率较低（79.5

该框架展现出很强的通用性，使用通义千问和 LLaVA 作为骨干模型时，最终答案准确率分别达到 93.8% 和 92.1%，

表 I 在 Set-VQA、GQA-SET 和 PTR-COUNT 基准测试上的主要性能比较。PA 和 AA 分别表示程序准确率和最终答案准确率。最佳结果以**粗体**显示，次佳结果以下划线显示

| 模型 | Type | Set-VQA（PA/AA） | GQA-SET（PA/AA） | PTR-COUNT（PA/AA） |
|---|---|---|---|---|
| 端到端基线 | | | | |
| Qwen2.5-VL-72B | 端到端 | 不适用 / 38.6% | 不适用 / 35.1% | 不适用 / 25.4% |
| CogVLM-17B | 端到端 | 不适用 / 37.5% | 不适用 / 34.2% | 不适用 / 24.0% |
| InternVL-2.5-26B | 端到端 | 不适用 / 36.4% | 不适用 / 33.7% | 不适用 / 23.1% |
| Qwen2.5-VL-32B | 端到端 | 不适用 / 36.1% | 不适用 / 32.8% | 不适用 / 23.9% |
| LLaVA-1.6-72B | 端到端 | 不适用 / 34.8% | 不适用 / 31.5% | 不适用 / 22.4% |
| LLaVA-1.6-34B | 端到端 | 不适用 / 31.2% | 不适用 / 28.5% | 不适用 / 19.8% |
| Qwen2.5-VL-7B | 端到端 | 不适用 / 28.9% | 不适用 / 26.5% | 不适用 / 18.1% |
| BLIP-2 | 端到端 | 不适用 / 26.5% | 不适用 / 24.1% | 不适用 / 16.2% |
| LLaVA-1.6-7B | 端到端 | 不适用 / 24.5% | 不适用 / 22.1% | 不适用 / 15.3% |
| 少样本合成器基线 | | | | |
| ViperGPT | 少样本合成 | 22.1% / 34.5% | 18.9% / 29.8% | 14.3% / 22.0% |
| VisProg | 少样本合成 | 15.3% / 25.8% | 11.2% / 19.4% | 8.5% / 14.2% |
| 视觉集合程序合成器（Qwen2.5-VL-7B 基础版） | | | | |
| 监督微调（我们的方法） | Ours | 77.2% / 73.5% | 70.1% / 67.2% | 65.4% / 61.3% |
| RLVR（我们的方法） | Ours | 79.5% / 86.1% | 73.1% / 80.2% | 67.6% / 74.5% |
| CASTER（我们的方法） | Ours | **95.1% / 93.8%** | **91.2% / 89.9%** | **88.3% / 87.1%** |
| 视觉集合程序合成器（LLaVA-1.6-7B Base） | | | | |
| 监督微调（我们的方法） | Ours | 75.8% / 71.3% | 68.2% / 64.9% | 63.5% / 59.1% |
| RLVR（我们的方法） | Ours | 78.2% / 84.6% | 71.5% / 78.3% | 65.9% / 72.4% |
| CASTER（本文方法） | Ours | 94.3% / 92.1% | 89.6% / 87.5% | 86.2% / 84.9% |

这表明，性能提升的关键在于范式本身，而非底层语言模型的选择。

**基线方法的公平性：知识增强评估**。一个自然的疑问是，CASTER 与端到端基线之间的性能差距，是否仅仅是因为我们的执行引擎能够访问外部知识（例如知识库中的营养信息），而标准模型无法获取这些信息。为确保比较绝对公平，我们进行了知识增强评估，将完全相同的键值形式外部知识直接注入所有端到端基线和少样本基线方法的文本提示中。表 II 报告了在 Set-VQA 测试子集上，采用标准版（St.）和增强版（Aug.）推理设置时的结果。

即使能够获取相同的外部知识，表现最强的端到端模型（Qwen2.5-VL-72B）的总体 AA 也仅从 38.6% 小幅提升至 42.1%。值得注意的是，它在极值（31.2%）和排除（28.6%）等复杂推理类别上的表现，仍远低于 CASTER 的 91.2% 和 90.1%。这证实，端到端模型表现不佳并非因为缺乏信息，而是因为缺少显式的组合逻辑机制。少样本合成器（VisProg、ViperGPT）在增强后也有所提升，但仍远不及 CASTER，这进一步表明，可靠的基于集合的推理需要结构化程序合成，而不是不受约束的自然语言推理。

*D.* 组合泛化

为拓展组合推理的边界，我们评估了“组合划分”设置（定义见第 IV-A 节）。该设置不仅考验模型的基础嵌套能力，还涉及集合排除逻辑、多层级比较和空间推理。结果见表 III，评估指标为程序准确率（PA）和答案准确率（AA）。端到端模型不会生成程序，因此这些方法的 PA 标记为 N/A。

CASTER 模型即使面对最困难的组合任务，也能保持较高的准确率。当面对要求逻辑排除（FILTER_NOT）或多步比较推理（MAX(ATTR, SORT)）的查询时，CASTER 的答案准确率仍然高达 88.1% 和 86.5%，表现十分出色。这表明，它对程序语义的理解足够深入，能够处理全新的逻辑结构。此外，其在空间查询上的表现（83.4%）进一步说明，它能够成功整合不同的推理模态。

端到端模型在执行传统组合式问题方面展现出的能力十分有限，对于回答此类复杂查询则完全无能为力。例如，通义千问 Qwen2.5-VL-72B 在回答“未见比较”问题时的准确率仅为 9.8%，比 CASTER 的表现低十倍。这凸显了在需要明确进行多步逻辑推理时，端到端架构所面临的困难。

基线模型 Qwen-7B-Tool 的表现远优于端到端模型，但仍在某种程度上不尽如人意

表 II 标准（St.）和知识增强版（Aug.）推理设置下，端到端模型与 CASTER 在 Set-VQA 测试子集上的比较。增强基线获得与 CASTER 执行引擎相同的外部知识，并将其直接注入文本提示。VP = VisProg，VG = ViperGPT，CR = CASTER。

| 模型 | 参数量 | 推理设置 | 单步 AA | 多跳 AA | 极值 AA | 排除 AA | 总体 AA |
|---|---|---|---|---|---|---|---|
| Qwen2.5 | 72B | St. | 46.2 | 31.5 | 29.8 | 25.1 | 38.6 |
| | | Aug. | 51.4 | 36.1 | 31.2 | 28.6 | 42.1 |
| LLaVA1.6 | 72B | St. | 42.1 | 28.6 | 24.5 | 21.0 | 34.8 |
| | | Aug. | 46.8 | 32.5 | 26.8 | 23.4 | 37.9 |
| Qwen2.5 | 32B | St. | 43.8 | 29.4 | 27.1 | 22.8 | 36.1 |
| | | Aug. | 47.5 | 33.8 | 28.5 | 24.9 | 39.4 |
| LLaVA1.6 | 34B | St. | 38.5 | 24.1 | 20.4 | 18.5 | 31.2 |
| | | Aug. | 43.2 | 28.6 | 22.1 | 20.2 | 33.7 |
| Qwen2.5 | 7B | St. | 36.4 | 21.8 | 18.2 | 16.1 | 28.9 |
| | | Aug. | 41.5 | 25.3 | 20.0 | 18.3 | 31.5 |
| LLaVA1.6 | 7B | St. | 31.2 | 17.9 | 15.1 | 13.8 | 24.5 |
| | | Aug. | 36.8 | 21.4 | 16.5 | 15.0 | 26.8 |
| VP（GPT-4o） | N/A | St. | 38.2 | 22.5 | 19.5 | 18.2 | 25.8 |
| | | Aug. | 44.5 | 26.8 | 22.1 | 20.5 | 31.4 |
| VG（GPT-4o） | N/A | St. | 46.8 | 29.1 | 24.8 | 22.6 | 34.5 |
| | | Aug. | 53.4 | 34.2 | 28.5 | 26.1 | 40.8 |
| **CASTER（Qwen2.5）** | 7B | 执行引擎 | **97.8** | **94.2** | **91.2** | **90.1** | **93.8** |

表 III 详细的组合泛化性能。我们在日益复杂的组合上进行评估，包括逻辑排除（FILTER_NOT）、多层级比较（MAX(ATTR, SORT)）（任务#4）以及组合空间推理（COUNT(FILTER_SPATIAL)）（任务#5）。任务#1 指未见过的 COUNT(SORT)；任务#2 指未见过的 SORT(FILTER)；任务#3 指未见过的排除 FILTER NOT。_

| 模型 | 总体（最终答案准确率） | 任务#1 (PA/AA) | 任务#2 (PA/AA) | 任务#3 (PA/AA) | 任务#4 (PA/AA) | 任务#5 (PA/AA) |
|---|---|---|---|---|---|---|
| CASTER（通义千问） | 88.90% | 90.5% / 89.5% | 91.2% / 90.1% | 89.5% / 88.1% | 87.2% / 86.5% | 85.1% / 83.4% |
| CASTER（LLaVA） | 87.50% | 89.1% / 88.3% | 90.3% / 89.2% | 88.0% / 86.8% | 85.9% / 84.9% | 83.5% / 81.9% |
| RLVR（通义千问） | 74.50% | 78.2% / 75.1% | 79.9% / 76.3% | 71.3% / 68.2% | 65.0% / 61.7% | 63.8% / 59.5% |
| 监督微调（通义千问） | 58.20% | 61.3% / 55.4% | 63.8% / 60.1% | 55.1% / 51.5% | 49.8% / 44.3% | 47.2% / 41.1% |
| Qwen2.5-VL-72B | 21.30% | 不适用 / 19.8% | 不适用 / 22.5% | 不适用 / 15.1% | 不适用 / 9.8% | 不适用 / 11.2% |
| CogVLM-17B | 20.10% | 不适用 / 18.5% | 不适用 / 21.1% | 不适用 / 13.5% | 不适用 / 8.5% | 不适用 / 10.5% |
| LLAVA-1.6-34B | 15.80% | 不适用 / 14.2% | 不适用 / 16.3% | 不适用 / 10.2% | 不适用 / 6.1% | 不适用 / 7.3% |
| ViperGPT | 24.10% | 25.1% / 22.7% | 28.0% / 25.8% | 20.1% / 17.6% | 15.7% / 12.0% | 18.9% / 15.2% |
| VisProg | 16.50% | 17.3% / 15.1% | 19.2% / 17.2% | 13.9% / 11.4% | 10.1% / 7.8% | 12.5% / 9.9% |

（总体 AA 为 28.7%）这一结果表明，尽管通过工具获取信息非常有帮助，但对于涉及多个步骤、需要以自然语言语法进行逻辑正确推理的复杂问题，模型的回答仍不够稳健。

尽管性能优于端到端模型，但在这种情况下，监督微调和 RLVR 模型似乎相当脆弱。由于从未见过如此复杂的程序句法形式，监督微调在比较任务上的性能降至 44.3%。尽管表现相对较好，RLVR 的性能也大幅降至 61.7%，因为它无法从稀疏奖励函数中学习如何做出如此复杂的决策，导致生成的程序在这种复杂情形下出错，成为被篡改的程序。

表 IV Set-VQA 测试集上的 CASTER 奖励组件消融实验。

| 模型 | 程序准确率（均值±标准差） | 最终答案准确率（均值±标准差） |
|---|---|---|
| CASTER（完整） | **95.1(±0.4)%** | **93.8(±0.3)%** |
| 采用仅类型匹配 | 91.2(±0.7)% | 89.5(±0.6)% |
| 使用二元奖励 | 85.4(±1.0)% | 84.0(±1.1)% |
| 使用归一化奖励 | 94.8(±0.5)% | 93.5(±0.4)% |
| RLVR | 79.5(±1.2)% | 86.1(±0.9)% |

### *E.* 消融研究

我们在 Set-VQA（表 IV）上开展消融实验，以分离 CASTER 奖励中各个组件的贡献。我们研究了四个因素。（1）奖励密度：我们测试

基于公式 4 的部分得分是否比二元反馈更有效。在 CASTER（二元奖励）变体中，杰卡德得分 $S_{ij}$ 被替换为一个二元值：只有当中间集合 $O_n$ 和 $O_{n_{gt}}$ 完全相同时，该值才为 1。（2）匹配策略：我们通过与更严格的变体 CASTER（仅类型匹配）进行比较，考察公式 5 中的语义二分图匹配是否必要。在该变体中，节点只能与算子类型相同的节点匹配（例如 FILTER 与 FILTER）。（3）奖励归一化：为检验较长的程序是否会使奖励虚高，CASTER（归一化奖励）将最终的 $R_{CASTER}$ 除以真实标注 AST 节点的数量 $|N(T_{gt})|$。（4）奖励稀疏性：我们将 RLVR 纳入基线，其使用纯粹稀疏的 0/1 答案奖励。

我们观察到，CASTER（二元奖励）使程序准确率降低了约 10%，这证实，与仅依据严格正确性相比，部分语义相似性能够提供更多信息量的学习信号。因此，杰卡德指数在奖励部分正确的子程序方面发挥着重要作用，使模型能够逐步完善其推理结构，而不是遭遇灾难性失败。

CASTER（仅类型匹配）变体的性能也明显下降。这表明，灵活的语义对齐是有益的，因为即使程序在语法形式上不同，它也会对逻辑等价的程序给予认可，而不是惩罚无害的结构差异。这说明，组合推理受益于语义监督，而非纯粹的结构监督。

最后，奖励归一化的影响微乎其微。这表明，程序长度导致的奖励膨胀在实践中并非主要因素，CASTER 的整体性能提升也并不取决于程序复杂度。

### *F.* 错误分析

为了解 CASTER 尚存的失败模式，我们在完整的 Set-VQA 测试集（20,000 个样本）上进行了细粒度分析，对比标准流水线执行与预言机执行器的结果。后者使用真实标注目标检测结果，从而将感知错误与逻辑错误区分开来。表 V 报告了 19 种逻辑操作以及五个查询复杂度等级（深度 1 至深度 5+）下的结果。

总体而言，在标准执行下，CASTER 的程序准确率（PA）达到 95.1%，最终答案准确率（AA）达到 93.8%。与预言机执行器（其最终答案准确率达到 95.1%）相比，我们观察到**逻辑差距仅为 4.9%**，而**感知差距仅为 1.3%**。这表明，CASTER 剩余错误中的绝大多数源于程序生成，而非感知失误。从操作层面看，基本操作（按颜色/形状筛选、计数、存在）准确率近乎完美（≥96.5% AA）；而涉及依赖 OCR 的属性的复杂操作（筛选品牌/光学字符识别：92.4% AA）或深度嵌套（深度 5+：85.1% AA）则更具挑战性。值得注意的是，即使在深度 5+ 下，感知差距仍然很小（1.5%），这证实执行引擎的检测模块并非瓶颈；真正的难点在于合成器难以生成深度嵌套的程序。

表 V：CASTER（Qwen2.5-VL-7B）在 Set-VQA 上的细粒度错误分析。标准执行器与预言机执行器的结果揭示了逻辑操作和查询深度上的感知差距。

| 逻辑操作 | 深度 | 标准程序准确率 | 标准最终答案准确率 | 逻辑 Gap | 百分比。Gap |
|---|---|---|---|---|---|
| 筛选（颜色） | 1 | 98.1 | 96.8 | 1.9 | 1.3 |
| 筛选（形状） | | 97.9 | 96.5 | 2.1 | 1.4 |
| 筛选（纹理） | | 97.0 | 95.2 | 3.0 | 1.8 |
| 筛选（品牌/光学字符识别） | | 95.5 | 92.4 | 4.5 | 3.1 |
| 计数（基本） | | 98.5 | 97.1 | 1.5 | 1.4 |
| 存在（基本） | | 99.1 | 98.2 | 0.9 | 0.9 |
| 选择（基本） | | 97.5 | 96.0 | 2.5 | 1.5 |
| 排序（空间） | 2 | 95.8 | 94.5 | 4.2 | 1.3 |
| 排序（数值） | | 95.1 | 93.9 | 4.9 | 1.2 |
| 最小值（属性） | | 95.3 | 94.1 | 4.7 | 1.2 |
| 最大值（属性） | | 95.2 | 93.8 | 4.8 | 1.4 |
| 筛选+计数 | 3 | 96.2 | 95.0 | 3.8 | 1.2 |
| 筛选+选择 | | 95.6 | 94.2 | 4.4 | 1.4 |
| 排序+最小值/最大值 | | 93.1 | 91.8 | 6.9 | 1.3 |
| 筛选+排序 | 4 | 92.8 | 91.5 | 7.2 | 1.3 |
| 排除（非） | | 91.5 | 90.1 | 8.5 | 1.4 |
| FLT+排序+最小值 | 5+ | 89.9 | 88.5 | 10.1 | 1.4 |
| 多重筛选+计数 | | 90.5 | 89.2 | 9.5 | 1.3 |
| 密集嵌套 | | 86.6 | 85.1 | 13.4 | 1.5 |
| **总体** | **All** | **95.1** | **93.8** | **4.9** | **1.3** |

这项分析表明，瓶颈已从逻辑转向感知：CASTER 已有效解决了组合推理这一挑战。

## 五、结论

我们提出了视觉集合程序合成器（VSPS）框架，以解决当前多模态大语言模型在组合式基于集合的视觉推理方面的局限。VSPS 的核心是组合式行动者集合论奖励（CASTER），这是一种密集且考虑语义的奖励函数，旨在鼓励模型生成逻辑正确且可解释的程序。实验表明，CASTER 不仅优于所有基线方法，还能缓解稀疏奖励强化学习中常见的“奖励劫持”行为。此外，即使为端到端模型提供相同的外部知识，VSPS 的表现仍比公平的工具增强基线高出 42 个百分点，这凸显了显式程序合成相较于不受约束地使用工具的优势。消融研究证实了所提出奖励设计中各个组成部分的必要性，而细粒度错误分析则表明，当前的主要失败模式源于感知而非逻辑：感知差距仅为 1.3%，而逻辑差距为 4.9%。这为未来研究指明了明确方向：改进底层视觉检测器和光学字符识别模块。我们相信，我们的框架为构建可信、透明的多模态推理系统奠定了有原则的基础。

## 致谢

本研究部分由奥地利科学基金（FWF）10.55776/COE12和AXA Research Fund资助。

# 参考文献

[1] Haotian Liu、Chunyuan Li、Qingyang Wu和Yong Jae Lee，《视觉指令调优》，《神经信息处理系统进展》，第36卷，第34892–34916页，2023年。[2] Josh Achiam、Steven Adler、Sandhini Agarwal、Lama Ahmad、Ilge Akkaya、Florencia Leoni Aleman、Diogo Almeida、Janko Altenschmidt、Sam Altman、Shyamal Anadkat等，《GPT-4技术报告》，arXiv预印本 arXiv:2303.08774，2023年。[3] Ali Furkan Biten等，《场景文本视觉问答》，载于CVPR，2019年，第4291–4301页。[4] Zehua Cheng、Di Yuan和Thomas Lukasiewicz，《基于亲和图的引导式收缩学习：以最少标注实现无前置任务医学图像分割》，载于2024年IEEE生物信息学与生物医学国际会议（BIBM）。IEEE，2024年，第3127–3133页。[5] Tanmay Gupta和Aniruddha Kembhavi，《视觉编程：无需训练的组合式视觉推理》，载于CVPR，2023年，第14953–14962页。[6] Kexin Yi等，《神经符号视觉问答：将推理与视觉和语言理解解耦》，NeurIPS，第31卷，2018年。[7] Dídac Surís、Sachit Menon和Carl Vondrick，《ViperGPT：通过Python执行进行视觉推断与推理》，载于CVPR，2023年，第11888–11898页。[8] Victor-Alexandru Pˇadurean、Georgios Tzannetos和Adish Singla，《用于视觉编程的神经任务合成》，arXiv预印本 arXiv:2305.18342，2023年。[9] Youssef Mroueh，《可验证奖励强化学习：GRPO的有效损失、动态与成功放大》，arXiv预印本 arXiv:2503.06639，2025年。[10] Xumeng Wen等，《可验证奖励强化学习隐式激励基础大语言模型进行正确推理》，arXiv预印本 arXiv:2506.14245，2025年。[11] John Schulman、Filip Wolski、Prafulla Dhariwal、Alec Radford和Oleg Klimov，《近端策略优化算法》，arXiv预印本 arXiv:1707.06347，2017年。[12] Zhihong Shao等，《DeepSeekMath：推动开放语言模型数学推理能力的极限》，arXiv预印本 arXiv:2402.03300，2024年。[13] Babak Badnava、Mona Esmaeili、Nasser Mozayani和Payman Zarkesh-Ha，《一种用于强化学习智能体的新型基于势函数的奖励塑形方法》，arXiv预印本 arXiv:1902.06239，2019年。[14] Grant C. Forbes、Nitish Gupta、Leonardo Villalobos-Arias、Colin M. Potts、Arnav Jhala和David L. Roberts，《基于势函数的奖励塑形与内在动机》，arXiv预印本 arXiv:2402 07411，2024年。[15] Alessandra Mileo，《迈向以人为中心的可解释性神经符号循环》，《神经符号人工智能》，第1卷，第NAI–240740页，2025年。[16] Drew A Hudson和Christopher D Manning，《GQA：用于真实世界视觉推理和组合式问答的新数据集》，载于CVPR，2019年，第6700–6709页。[17] Yining Hong等，《PTR：面向基于部件的概念、关系和物理推理的基准测试》，arXiv预印本 arXiv:2112 05136，2021年。[18] Tianheng Cheng、Lin Song、Yixiao Ge、Wenyu Liu、Xinggang Wang和Ying Shan，《YOLO-World：实时开放词汇目标检测》，载于CVPR，2024年。[19] Xinyi Hou、Yanjie Zhao、Shenao Wang和Haoyu Wang，《模型上下文协议（MCP）：现状、安全威胁与未来研究方向》，arXiv预印本 arXiv:2503.23278，2025年。[20] Shuai Bai等，《Qwen2.5-VL技术报告》，arXiv预印本 arXiv:2502.13923，2025年。[21] Weihan Wang等，《CogVLM：面向预训练语言模型的视觉专家》，2023年。[22] Haotian Liu等，《LlavaNext：改进推理、光学字符识别和世界知识》，2024年。[23] Junnan Li、Dongxu Li、Silvio Savarese和Steven Hoi，《BLIP-2：通过冻结图像编码器和大语言模型引导语言-图像预训练》，载于ICML。PMLR，2023年，第19730–19742页。

## 附录 A 程序语言和执行引擎定义

程序语言：多模态大语言模型会生成由嵌套集合运算组成的领域特定语言（DSL）程序。其核心运算符包括：

- 筛选(objects, <条件>)：根据属性返回对象的一个子集（例如，类别='汽水'，卡路里 > 100）。
- 选择(attribute, objects)：从一组对象中提取特定属性。
- 计数(objects)：返回集合的基数。
- 最小值(attribute, objects) / 最大值(attribute, objects)：查找具有最小/最大属性值的对象。
- SORT(attribute, objects)：对集合进行排序。
- EXISTS(objects)：如果集合非空，则返回 true。

执行引擎($Exec$)：该模块负责针对图像 $I$ 执行给定的程序 $P$ 。它是一个确定性的不可学习组件，用于将符号程序连接到视觉世界。

- **感知栈**：我们使用 YOLO-World [18], 这一预训练目标检测器来识别场景中的所有潜在对象，并使用带有 Model Context Protocol [19] 的 PaddleOCR 来提取任何可见文本。
- **属性定位**：当程序需要某个属性（例如卡路里）时，执行引擎首先尝试通过光学字符识别来查找该属性。
- **知识检索**：如果某个属性在视觉上不可见（例如"面条"包装上的"卡路里"），执行引擎会使用对象的类别名称，在模拟的外部知识库（一个简单的键值字典）中进行查询。
- **子程序执行器**($Exec_sub$)：此函数对于我们的奖励至关重要，它执行一个子程序（抽象语法树中的一个节点），并返回其生成的具体对象标识符集合。

需要指出的是，使用此执行引擎，尤其是模拟知识源，为我们的模型提供了标准端到端多模态大语言模型无法获得的信息。为确保比较公平，我们还将在第4节评估一个经过工具增强的基线多模态大语言模型，它可以使用相同的感知和检索功能，但必须学习通过自然语言提示调用这些功能，而不是使用结构化程序。

## 附录B

### 实验设置与超参数

为评估我们框架的鲁棒性和灵活性，我们采用两种不同的开源基础多模态大语言模型来实现视觉集合程序生成器中的合成器：Qwen2.5-VL- 7B模型和LLaVA-1.6-7B模型。这使我们能够评估所提方法在不同底层模型架构上的性能。训练时，我们所有的框架变体首先通过监督微调初始化3个轮次。RLVR和CASTER模型随后使用GRPO额外训练5个轮次，KL散度系数 $\beta$ 为0.05，并采用AdamW优化器和学习率 $1e^{-6}$ 。所有实验均在8块NVIDIA H100 GPU上进行。

我们将视觉集合程序生成器与三类最先进模型进行比较。

**端到端多模态大语言模型**：这些模型通过提示被要求直接一步作答。我们评估了若干领先模型：

- 通义千问Qwen2.-VL [20](7B、32B、72B)：这是一个先进的多模态大语言模型系列，在各类视觉语言基准测试中表现出色。我们纳入了多个规模，以检验单纯增加参数量能否弥补其在基于集合的推理方面的架构缺陷。
- CogVLM-17B [21]：这是一款功能强大的多模态大语言模型，其注意力机制和前馈网络层中均设有视觉专家模块，因此能够在模型的多个层级深度融合视觉和文本信息，而非仅在输入层进行融合。
- LLaVA-1.6（7B、34B、72B） [22]：这是一个知名且广泛使用的多模态大语言模型系列，率先将GPT生成的数据用于视觉指令调优。其架构通过一个简单的投影层，将视觉编码器连接到大语言模型，因此成为评估指令遵循能力的重要基线。我们测试了多个规模，包括假设的72B变体，以评估其在我们任务上的规模扩展特性。
- BLIP-2 [23]：一种基础型多模态大语言模型，引入了Q-Former，作为冻结图像编码器与冻结大语言模型之间的轻量级桥梁。这种参数高效的方法使其成为当今许多多模态大语言模型在架构上的重要前身。

**少样本程序合成器**：这些模型以少样本、基于提示的方式生成程序，无需针对特定任务进行训练。

- VisProg [5]：一种基础的神经符号框架，通过基于上下文示例组合视觉模块来生成 Python代码。
- ViperGPT [7]：一种先进的框架，同样能够生成 Python 程序来回答视觉查询；其显著特点是能够生成自己的上下文示例，从而减少手动提示工程。

## 附录 C

## SET-VQA

Set-VQA 基准旨在全面测试基于集合的视觉推理能力。为此，该基准采用双管齐下的方法构建了 110,000 个示例：

由 60,000 个示例组成的合成部分采用程序优先方法生成。具体而言，系统根据形式文法自动生成复杂的嵌套程序。随后，类似于 CLEVR 的渲染引擎创建了与每个程序的

逻辑精确匹配的三维场景和对象。然后，这些形式化程序被转换为模板化问题，最后由人工标注员改写，以确保自然语言表达的多样性。

第二部分由 50,000 个真实世界示例组成。该流程始于收集第一人称视角图像，例如商店货架或桌面的照片，这些图像中包含丰富的对象。随后，人工标注员针对这些图像编写复杂的集合类问题。之后，接受过系统专用程序语言培训的专家标注员编写相应的标准可执行程序，并核验最终答案。这种结合方式确保该基准测试既具备逻辑复杂性（来自合成数据），又具有视觉真实性（来自真实世界数据）。

Set-VQA 中的每个数据点都包含四项关键信息。如示例数据点所示，其中包括图像_id，例如“real_shelf_0451.jpg”；自然语言查询，例如“顶层货架上的饮料中，哪一种最便宜？”；对应的真实程序，即“SELECT(MIN(PRICE), FILTER(FILTER(objects, class='drink'), relation='on_top_shelf'))”；以及最终的正确_答案，本例中为“泉水”。这种结构使系统能够针对逻辑程序推理进行训练和评估。