

Visually Interpretable Subtask Reasoning for Visual Question Answering

Yu Cheng¹ Arushi Goel² Hakan Bilen¹
¹University of Edinburgh ²NVIDIA

s2521923@ed.ac.uk, arushig@nvidia.com, hbilen@ed.ac.uk

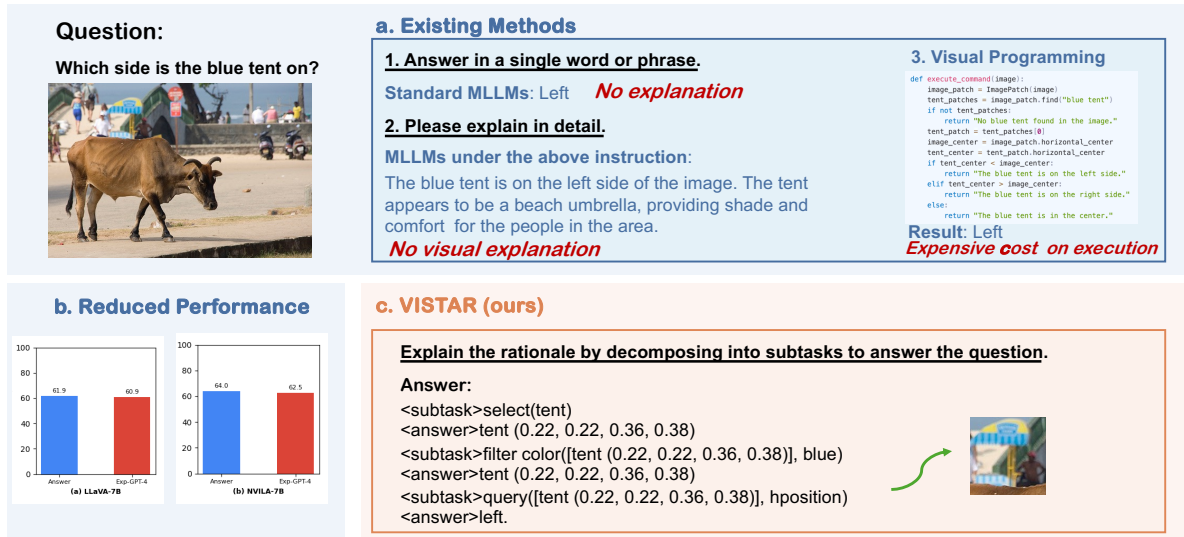


Figure 1. **Comparison of standard MLLMs, programmatic reasoning and VISTAR across the output.**(a) Existing methods either provide no explanations, lack visual grounding, or have high computational costs. (b) Performance degradation of standard MLLMs (LLaVA-1.5-7B [21] and NVILA-8B [23]) when forced to generate explanations alongside answers. ‘Exp-GPT-4’ evaluates semantic similarity using *GPT-4-turbo* [1]. (c) VISTAR effectively addresses these issues by decomposing the question into structured sub-tasks, providing both visual (bounding box, following the format (x_l, y_l, x_r, y_r) where x_l, y_l are the coordinates of the top-left corner and x_r, y_r are the coordinates of the bottom-right corner) and textual rationales without compromising accuracy.

Abstract

Answering complex visual questions like ‘Which red furniture can be used for sitting?’ requires multi-step reasoning, including object recognition, attribute filtering, and relational understanding. Recent work improves interpretability in multimodal large language models (MLLMs) by decomposing tasks into sub-task programs, but these methods are computationally expensive and less accurate due to poor adaptation to target data. To address this, we introduce VISTAR (Visually Interpretable Subtask-Aware Reasoning Model), a subtask-driven training framework that enhances both interpretability and reasoning by generating textual and visual explanations within MLLMs. Instead of relying on external models, VISTAR fine-tunes MLLMs to produce structured Subtask-of-Thought rationales (step-by-step reasoning sequences). Experiments on two benchmarks show that VISTAR consistently improves

reasoning accuracy while maintaining interpretability. Our code and dataset will be available at <https://github.com/ChengJade/VISTAR>.

1. Introduction

Recent advancements in multimodal large language models (MLLMs) have led to significant performance improvements on image-text paired tasks like image captioning [9, 38], visual question answering (VQA) [21, 22], and video understanding [13, 15]. Despite these advancements, a critical limitation of current MLLMs is their lack of interpretability. Notably, in terms of VQA task, recent MLLMs [21, 23, 35] are primarily trained under instructions to generate direct answers, optimizing for correctness rather than interpretability. When these models are prompted to generate explanations alongside their predic-

tions, their answer accuracy declines significantly. This phenomenon arises because current MLLMs are not explicitly trained to engage in stepwise reasoning, and introducing interpretability often interferes with their optimization for final answer prediction. Furthermore, most existing models primarily generate textual explanations, lacking visual justifications such as object-level grounding, which is essential for human-aligned reasoning in the VQA task.

To enhance interpretability in the VQA task, recent approaches such as VISPROG [11] and ViperGPT [32] have proposed decomposing complex queries into a sequence of sub-tasks in modular programming languages, where each step is executed by a pre-trained model (e.g., object detector, pose detector), thereby making the reasoning process more transparent. However these approaches have multiple shortcomings. First, executing multiple models sequentially incurs high computational costs. Second, they rely on pre-trained models that may have been trained on disparate data distributions, operating in a zero-shot setting without parameter adaptation, which can lead to performance drops. Third, these methods are inherently error-prone, as the generated programs may omit necessary steps, introduce spurious operations, or fail to recover from errors in intermediate outputs. To mitigate the last limitation, VPD [12] introduces an instruction-tuning framework that distills the reasoning capabilities of large language models (LLMs) into MLLMs, allowing them to sample multiple candidate programs and identify the correct one. Despite improving answer accuracy, VPD still relies on program execution thereby inheriting the first two limitations.

Building upon the limitations of the prior work, we introduce **Visually Interpretable Subtask-Aware Reasoning Model (VISTAR)**, a framework designed to enhance the interpretability and reasoning of MLLMs in VQA. To highlight the key differences between standard MLLMs and our approach, Fig. 1 presents two examples where both models correctly predict the answer. Our method goes beyond direct answer generation by providing structured reasoning steps, including a textual explanation that decomposes into sub-tasks and object-level visual grounding via bounding boxes. Specifically, VISTAR introduces LLM-prompted **Subtask-of-Thought (SoT)** rationales, which decompose visual queries into structured sub-tasks such as object selection, attribute filtering, and relationship verification while generating intermediate answers in the form of object names with bounding boxes, categorical attributes, or binary decisions. Next, VISTAR employs a multi-modal instruction tuning strategy to integrate these structured SoT rationales into the learning process, enabling MLLMs to generate both interpretable reasoning steps and accurate answers. Unlike existing interpretable methods that rely on computationally expensive program execution, VISTAR learns structured reasoning directly within an MLLM, elim-

inating the need for external program execution while preserving interpretability.

Hence, the main contributions of this work can be summarized as follows:

- We introduce SoT, an LLM-generated dataset that structures visual queries into step-by-step reasoning sub-tasks.
- We propose VISTAR, an instruction tuning pipeline that enables MLLMs to generate visual (object-level bounding boxes) and textual (sub-task reasoning process) explanations.
- Experimental results demonstrate that VISTAR achieves improved interpretability and reasoning accuracy, and generalizes well to datasets in the zero-shot setting.

2. Related work

Multimodal Large Language Models (MLLMs). With the advancement of LLMs in text understanding and reasoning—such as GPT [3, 27], Vicuna [5], LLaMA [33], and Qwen [2], there is growing demand in developing MLLMs that extend these capabilities to both text and images. Examples include BLIP2 [19], InstructBLIP [6], MiniGPT-4 [40], Qwen-VL [2, 34], LLaVA [21, 22], and NVILA [23].

MLLMs typically use a pre-trained visual encoder (e.g., CLIP [28] or SigLIP [36]) to extract visual features, which are then mapped into textual token embeddings via a connector before being processed by an LLM. Training involves two key stages: large-scale image-text pair alignment to synchronize visual and textual representations, followed by instruction tuning to enhance task adaptability. The visual encoder is usually frozen, while the connector and LLM weights are updated during instruction tuning. These models are commonly evaluated on various vision-language tasks [7, 13, 25, 39], including VQA [39], image captioning [38], visual grounding [25], and multimodal reasoning [7]. Among them, VQA [18] is a crucial benchmark for assessing MLLM’s ability to understand and reason over visual content by answering questions about images. However, most existing MLLMs generate only brief textual responses, lacking interpretability [21, 23].

Interpretability. Most existing MLLMs, such as LLaVA [21, 22], are primarily trained on VQA tasks with instructions like “answer the question in a single word or phrase”. While this ensures concise responses, it restricts the model’s ability to provide explanations. Although prompt tuning (e.g., “give a more detailed explanation”) can elicit longer explanations, the generated explanations often lack depth and explicit visual reasoning, limiting overall interpretability [24].

Recent research has focused on improving MLLM interpretability in VQA through modular programs that break

down visual queries into Python-like components [11, 16, 32]. These methods leverage the in-context learning ability of code-generation model [4] to dynamically assemble vision-language modules, enabling explicit reasoning and better generalization. By executing generated programs using predefined APIs, these frameworks eliminate the need for additional training. However, this approach relies on multiple pre-trained models—such as object detection [20], visual question answering [19], depth estimation [29] etc.—which significantly increases computational costs.

Beyond modular programs, another promising direction for enhancing the reasoning and interpretability of MLLMs in VQA is question decomposition [17, 26, 37]. This method breaks down complex queries into sub-questions, effectively introducing a new form of chain-of-thought (CoT) reasoning that allows models to reason step by step. Recent works [26, 37] have shown that question decomposition not only improves interpretability but also enhances accuracy by aligning with human reasoning strategies. Some approaches enable MLLMs to generate their own decompositions through in-context learning [17], while others propose fine-tuning models using LLM-generated question decomposition data [26, 37]. However, question decomposition is not always beneficial for interpretability, since it is also unable to provide visual explanations.

Most related to ours, Hu et al. [12] converted generated programs into CoT reasoning and distilled the resulting data into MLLMs. While this approach enhances reasoning capabilities, it remains reliant on program execution of pre-trained models, making it computationally intensive and failing to adapt them to the target data. In contrast, our model is directly trained for the target task while being computationally efficient and providing accurate answers along with textual and visual explanations.

3. Method

Our approach, VISTAR leverages LLM-generated SoT rationales, which decomposes the VQA task into structured intermediate reasoning steps, making the model’s decision-making process more transparent and interpretable (see Fig. 2). Here, we first present the pipeline for generating SoT rationales, detailing how these intermediate steps contribute to multimodal understanding. We then describe our training strategy, which incorporates these rationales into the learning process, enabling MLLMs to better utilize structured reasoning for improved interpretability and performance.

3.1. Problem Formulation

Given an image I and a question Q , our goal is to generate SoT rationales—a structured reasoning process that decomposes the question into a sequence of sub-tasks, each producing intermediate results that are required to arrive at the

correct answer along with the final answer. A SoT S (illustrated in Fig. 1.(c)) is defined as an ordered set of reasoning steps:

$$S = \{(\circ_{P_1}, r_1), (\circ_{P_2}, r_2), \dots, (\circ_{P_n}, r_n)\} \quad (1)$$

where \circ_{P_i} represents the i -th sub-task operation, such as selecting an object, filtering color, or querying an attribute, and r_i is its corresponding intermediate result and n is the total number of operations. The final result, r_n , is the predicted answer to the question. Each intermediate result r consists of $\{a, b\}$ where a is the intermediate textual answer and $b = (x_l, y_l, x_r, y_r)$ is the bounding box representation, a four-dimensional tuple containing top left and bottom right coordinates, normalized by dividing each coordinate by the larger dimension of the image (either width or height). Our goal is to learn a function $f_\theta : (I, Q) \rightarrow S$, an MLLM model parameterized by θ that maps a pair (I, Q) to S , the SoT sequence.

3.2. SoT generation

Here, we explain our pipeline for synthesizing SoT for the VQA task (see in the top part of Fig. 2) on the GQA dataset [14].

Given an image I with $(Q, SG, \{\circ_{P_i}\}_{i=1}^N, y)$ —where Q is a textual query, SG is the scene graph of the image encoding objects, attributes, and relationships between objects, \circ_P is the set of sub-task operations, and y is the ground truth answer for the Q query, we obtain the corresponding SoT by predicting the missing intermediate answers $\{r_i\}_{i=1}^{N-1}$ where $r_N = y$ through the following steps:

1. Prompt-based LLM generation: As manually labeling the intermediate answers is expensive, we automatically obtain the SoT. As the given SG includes all the required visual information, we use an LLM [33, 40] (not an MLLM) for this step by prompting it with a structured prompt consisting of (see Appendix §B for more details):

- Q : The original textual query from the dataset.
- **The ground truth answer y :** To guide the expected reasoning process.
- SG : A structured representation of objects, attributes, and relationships.
- $\{\circ_{P_i}\}_{i=1}^N$: The GQA dataset defines 145 sub-task operations in a specific format (Fig.8). To improve clarity and interpretability, we reformat these operations within our prompt, ensuring the LLM produces outputs in a structured SoT format (Fig.8). Each sub-task operation functions as a predefined function with a fixed set of arguments (see Appendix §C).
- **In-context Learning:** In the prompt, we illustrate the SoT generation through a few examples to get desirable outcomes.

Given this structured input, LLM can generate a SoT in the format of Eq. (1). In particular, the outputs of each

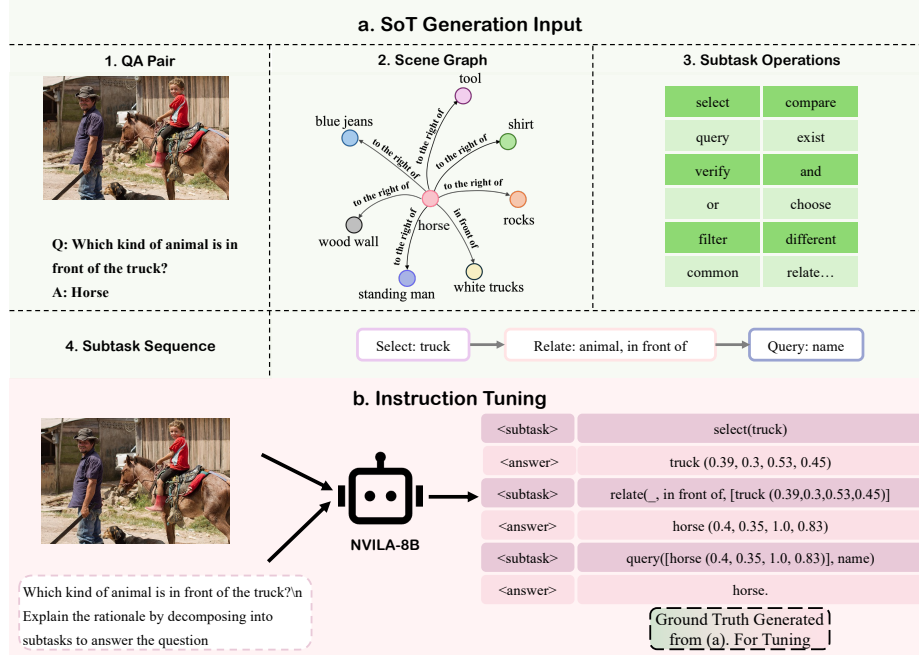


Figure 2. **Overview of VISATR.** VISTAR uses an LLM to generate faithful SoTs via in-context learning. Given the input including the query, ground truth, scene graph in an image and sub-task operation sequence inside the dataset (top), LLaMA-3.1-70B-Instruct would output a SoT to answer the query. The generated SoT is then used to fine-tune an MLLM, enabling it to produce both visual (object-level bounding boxes) and textual explanations during inference (bottom).

sub-task operation r_i can be objects with their bounding boxes, categorical attributes (e.g., ‘red’), or binary decisions (‘yes’/‘no’). For instance, as shown in Appendix §E Fig. 8, given a complex query, the model produces the SoT as:

- `<subtask>select(garland):` The model first identifies the target object (garland) and outputs its corresponding intermediate answer: `<answer>garland <bbox>(0.51, 0.0, 0.54, 0.09)`.
- `<subtask>relate(curtain, to the right of, [garland <bbox>(0.51, 0.0, 0.54, 0.09)]):` Next, the model determines which object satisfies the specified spatial relationship, identifying curtain as being “to the right of” the garland. The resulting answer is `<answer>curtain <bbox>(0.73, 0.0, 0.87, 0.58)`.
- `<subtask>relate(furniture, same color, [curtain <bbox>(0.73, 0.0, 0.87, 0.58)]):` The model then finds a piece of furniture that shares the same color as the curtain, producing the result `<answer>couch <bbox>(0.12, 0.48, 0.71, 0.97)`.
- `<subtask>query([couch <bbox>(0.12, 0.48, 0.71, 0.97)], name):` Finally, the model retrieves the name of the identified furniture, yielding the answer `<answer>couch`.

2. SoT filtration: Since LLM-generated data is prone to errors, we remove the noisy SoTs when their final answer does not align with the ground truth answers, they do not conform to the expected argument format, and their length exceeds a specified threshold to prevent excessively long reasoning.

3.3. Instruction tuning for interpretability

Here, we present the details for training using the data generated in §3.2, with the goal of providing both visual and textual explanations to support MLLM answer prediction. Building on the fine-tuning of MLLMs for instruction-based tasks, we employ a visual instruction tuning approach that aligns the model’s reasoning process with SoT rationales. Specifically, we prepend a task-specific instruction “Explain the rationale by decomposing into subtasks to answer the question”, to each query Q . Given a training sample (I, Q, S) , we fine-tune the MLLM f to minimize the loss associated with predicting the correct SoT, S , including the final answer. The resulting optimization is as follows:

$$\min_{\theta} \sum_{j=1}^N \ell(f_{\theta}(I_j, Q_j), S_j) \quad (2)$$

where I represents the input image, Q is the textual query appended with our instruction, and S denotes the struc-

tured SoT reasoning. The function $\ell(\cdot)$ represents the next-token prediction loss, and N is the total number of training samples. This objective encourages the model to generate well-structured reasoning sequences that align with human-interpretable step-by-step explanations. During training, we initialize from a pre-trained MLLM and fine-tune it using our generated SoT dataset. The vision encoder remains frozen, while we update the visual adaptor and the language model to effectively integrate both textual and visual explanations. By leveraging instruction tuning, we enhance the model’s interpretability and generalization, enabling it to provide more structured and explainable responses for the VQA task, even under the zero-shot setting.

4. GQA-SoT

We apply our SoT generation method on the GQA dataset [14] that originally consists of 113K images with 22M question-answer pairs along with sub-task operation sequences (see Table 1). For SoT generation, we retain all training images but selectively sample questions to ensure balanced coverage of all question types and allow our model to learn diverse reasoning patterns while avoiding redundant data generation and excessive computational costs associated with annotating the entire training set. As a result, we generate 249K SoT annotations on the GQA dataset [14] for fine-tuning, and 132K for validation ensuring quality evaluation of predicted SoT from the fine-tuned model (as shown in Table 1). Notably, as there is no ground truth of scene graph for the test set, we are unable to generate rationales for the test set. We call this SoT augmented dataset, GQA-SoT.

Table 1. Statistics of our LLM-generated SoT annotations used for training and evaluation on GQA.

Dataset	Split	Images	# QA Pairs	# SoTs
GQA [14]	Train	90.5K	14.3M	-
	Validation	8.1K	132K	-
	Test	3.0K	1.51M	-
	others	16.59K	4.95M	-
GQA-SoT(Ours)	Train-balanced	77.4K	994K	249K
	Validation	8.1K	132K	132K
	Test-dev-balanced	0.3K	12.5K	-

5. Experiments

Here, we evaluate the effectiveness of our approach in enhancing the interpretability of MLLMs for VQA. Specifically, we finetune *NVILA-8B* using our instruction tuning data, a state-of-the-art MLLM known for its strong performance in vision-language tasks. This section describes our experimental setup (§5.1), assesses compositional reasoning in VQA task rigorously (§5.2), interpretability of our

model (§5.3) and ablate various design decisions (§5.4).

5.1. Experimental Setup

Backbone model. We perform experiments with *NVILA-8B* [23] as our base model which is a strong baseline with competitive performance on both image and video benchmarks, and is equipped with strong multimodal reasoning capabilities.

Datasets. The generation of SoT relies on extracting sub-task operations from the GQA dataset [14] as discussed in §4, where other VQA datasets are unavailable for these sub-task operations. Hence, we fine-tune our model exclusively on the GQA-SoT and evaluate its performance on the same dataset. We also evaluate our model on the CRIC dataset [8], after sub-selecting questions that require complex reasoning steps from its evaluation set. Note that this evaluation is performed as zero-shot without training our model on CRIC to assess our model’s generalization ability beyond the GQA dataset while focusing on compositional reasoning in unseen scenarios.

Training setup. We fully fine-tune the visual adaptor and the LLM backbone of the *NVILA-8B* [23] initialized model using 8-A100 NVIDIA GPUs. Specifically, we use a learning rate of $1.5e-5$ with a warmup ratio of 0.3 and a cosine decay, a global batch size of 128, and a total of 1 training epochs. The training process takes approximately 4 hours.

Baselines. We compare our proposed method to baselines for the final answer accuracy on the test set of the GQA dataset [14]. Importantly, we compare to the original *NVILA-8B* model [23], other recent MLLMs including *MiniGPT-4* [40], *InstructBLIP* [6], *Qwen-VL-Chat* [2], *LLaVA-1.5* [21], and visual compositional reasoning methods such as *ViperGPT* [32] and *VPD* [12]. We also assess our model’s interpretability in terms of object localization and textual interpretability, and compare our model with *NVILA-8B* (backbone) [23].

Evaluation setup. Each SoT consists of the final prediction, sub-task operations and object-level bounding boxes. First we evaluate the answer accuracy in the VQA task by comparing the model’s final predictions to the ground-truth answers. Next, we assess the accuracy of the sub-task operations (e.g., select), focusing on the correctness of the predicted reasoning steps without considering their arguments (e.g., truck). This is crucial to determine whether the model follows a logically reasoning process when decomposing complex queries into sub-tasks. Since generating our SoT that contains sub-task operations and their arguments (e.g., select(truck)) is similar to open-ended text generation, we use *GPT-4-turbo* to better capture the similarity to the ground truth. In addition, we also include human evaluation to evaluate the consistency and quality of predicted SoT by our fine-tuning model.

Finally, to verify visual explanations based on the ob-

Table 2. **Comparison of accuracy (%) on the GQA [14] test set.** We report results for various state-of-the-art models.

	Accuracy (%) \uparrow
ViperGPT (zero-shot) [32]	48.1
MiniGPT-4 (Vicuna-13B) [40]	43.5
InstructBLIP (Vicuna-13B) [6]	49.5
Qwen-VL-Chat (9.7B) [2]	57.5
PaLI-3-VPD (5B) [12]	61.3
PaLI-3-VPD (55B) [14]	63.3
LLaVA-1.5 (Vicuna-7B) [21]	61.9
LLaVA-1.5 (Vicuna-13B) [21]	63.2
NVILA (8B) [23]	64.0
VISTAR (fine-tune on NVILA-8B)	65.1 \uparrow (+1.1)

ject bounding box, we use the validation set for evaluation (§4) as there is no bounding box provided in the test dataset of GQA [12]. For VQA-based object grounding evaluation, we adapt widely-used Intersection over Union (IoU) [30], precision, and recall to VQA-based object detection. Specifically, we calculate **IoU** by averaging the *IoU scores* of all correct predictions, considering only cases where the model’s answer is correct. We define *True Positives (TP)* as instances where the predicted answer is correct and the detected bounding box IoU exceeds a given threshold. *False Positives (FP)* include cases 1) where the answer is correct but IoU is below the threshold; 2) the answer is incorrect and IoU is below the threshold; 3) or the answer is incorrect but IoU is above the threshold. The total ground truth samples include all evaluated instances. Given this, **precision** is computed as the ratio of *TP* to the sum of *TP* and *FP* while **recall** is defined as the ratio of *TP* to the number of all the ground truth samples.

5.2. Compositional reasoning visual question answering

We assess our model’s ability finetuned on the GQA-SoT dataset to generate answers on benchmarks that require multi-step reasoning and compositional understanding. Table 2 shows the accuracy of baseline VQA models on the test set of the GQA dataset [14]. It is important to note that the baseline models, except for ViperGPT [32], generate the direct answer without providing any explanations. As shown in Table 2, our model achieves a new state-of-the-art performance on the GQA dataset [14] while providing step-by-step textual and visual reasoning.

Additionally, we conduct a zero-shot evaluation on the CRIC [8] dataset, a compositional reasoning and common sense dataset unseen during training, to assess the model’s ability to generate structured explanations without supervision. We sampled 10,000 questions from the validation set of CRIC [8] dataset. As CRIC has not been widely tested

on other baselines, there are no available results for direct comparison, so we compare directly to our baseline *NVILA-8B* [23]. In Table 3, we observe that our model outperforms the *NVILA-8B* [23] on CRIC [8], achieving an accuracy of 61.1% compared to 60.8%. The result suggest that our model not only improves performance on the seen GQA dataset but also generalizes to unseen datasets like CRIC [8].

Table 3. **Comparison of accuracy (%) on the randomly sampled CRIC dataset.** We report the result in a zero-shot setting.

	Accuracy (%) \uparrow
NVILA (8B zero-shot) [23]	60.8
VISTAR (8B zero-shot)	61.1 \uparrow (0.3)

5.3. Evaluation on interpretability

Here, we evaluate the quality of predicted SoT rationales across both visual and textual explanations. As mentioned in §4, the SoT for the test set is not available, we evaluate the performance of SoT prediction on a sampled subset of the validation set.

Quantitative results of visual explanations. First, we assess the accuracy of our object-level bounding box predictions. Since our backbone model, *NVILA-8B* [23], does not natively generate bounding boxes for the VQA task, we first obtain its predicted answer using the original VQA instruction, “Answer in a single word or phrase.”. We then append a follow-up instruction from its object grounding task, “Provide the bounding box coordinate of the region.”, to extract the corresponding bounding box (as shown in Appendix §D Fig. 7). For our proposed method *VISTAR*, we extract the operation `query([object name <bbox> (bounding box)], name)` at the final reasoning step from the SoT. It ensures that the predicted answer corresponds to an object, enabling a direct comparison of bounding boxes between *VISTAR* and *NVILA-8B* [23]. By aligning the final predicted object names with their respective bounding boxes, we effectively evaluate the object-level localization accuracy of both models. As shown in Fig. 3, *VISTAR* consistently outperforms *NVILA-8B* [23] in both answer correctness and bounding box accuracy. Specifically, *VISTAR* achieves higher accuracy in predicting correct answers while also improving object localization, as evidenced by increased IoU scores. When considering different IoU thresholds (0.5, 0.75, 0.95), *VISTAR* demonstrates superior precision and recall across all levels, indicating more accurate and consistent visual grounding of the predicted objects.

Quantitative results of textual explanations. We then evaluate the quality of textual rationales generated by our model compared to the backbone model *NVILA-8B* [23].

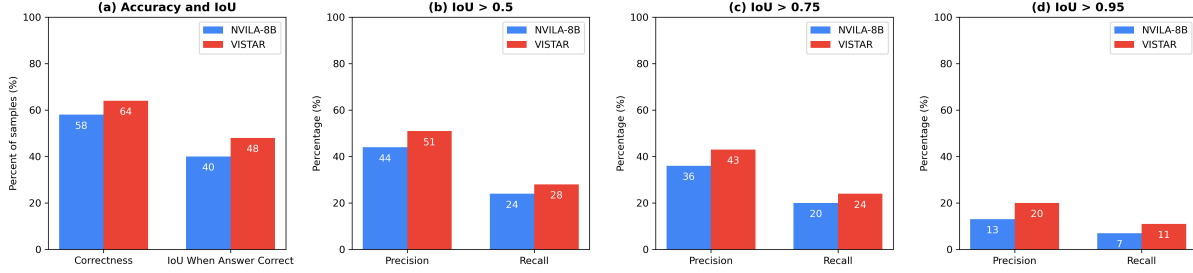


Figure 3. **Quantitative results of visual explanations.** (a) Comparison of answer correctness and object localization accuracy (measured by IoU when the answer is correct). (b-d) Precision & recall at different IoU thresholds (0.5, 0.75, 0.95). VISTAR consistently outperforms NVILA-8B in both accuracy and object-level visual grounding, demonstrating improved interpretability and localization quality.

For *NVILA-8B* [23], we prompt the model with the instruction “Please explain in detail.” during inference to obtain detailed explanations. Due to the open-ended nature of these explanations, we use *GPT-4-turbo* [1] to assess the semantic similarity between generated explanations and ground truth answer in terms of accuracy. Additionally, to ensure a fair comparison, we evaluate the semantic similarity, operation accuracy without arguments and logical consistency of our model’s predicted SoT with the ground truth annotations, also leveraging *GPT-4-turbo*. **Sub-task operation accuracy** (denoted as Op. Acc) specifically measures whether each reasoning step (operation prediction) aligns with expected sub-task operations without considering detailed arguments, and **logical accuracy** (denoted as Logical Acc) evaluates if the overall reasoning process logically supports the final answer, considering the operations and intermediate results. Results in Table 4 demonstrates that VISTAR outperforms on textual interpretability across all metrics compared to *NVILA-8B* [23], indicating its superior capability in maintaining logical coherence throughout the reasoning process. These results confirm the effectiveness of incorporating structured SoT rationales into MLLMs for improved interpretability and logical reasoning performance.

Table 4. **Quantitative evaluation of textual rationales.** We compare VISTAR with *NVILA-8B* [23] on answer accuracy (evaluated by *GPT-4-turbo* [1]), sub-task operation accuracy (logical correctness), and consistency of textual explanations on a sampled validation set.

	Ans. Acc	Op. Acc	Logical Acc
NVILA-8B [23]	61.2	-	88.9
VISTAR	64.8	98.5	90.3

Human evaluation on SoT. We conduct the human evaluation of predicted SoTs around 300 samples to present the **logical accuracy**. We assume that when the final predicted answer is correct, the correctness ratio of its sub-task operation sequences, including arguments and intermediate

a. Question: What is the gray device in this photo? ✓

Prediction:
 <subtask>select(device)
 <answer>keyboard <bbox>(0.13, 0.13, 0.75, 0.34),
 computer mouse <bbox>(0.4, 0.32, 0.7, 0.52),
 calculator <bbox>(0.32, 0.33, 1.0, 0.85)
 <subtask>filter color([keyboard <bbox>(0.13, 0.13, 0.75, 0.34),
 computer mouse <bbox>(0.4, 0.32, 0.7, 0.52),
 calculator <bbox>(0.32, 0.33, 1.0, 0.85)], gray)
 <answer>calculator <bbox>(0.32, 0.33, 1.0, 0.85)
 <subtask>query([calculator <bbox>(0.32, 0.33, 1.0, 0.85)], name)
 <answer>calculator.

b. Question: Which kind of furniture is to the left of the mattress? ✓

Prediction: **Semantically similar**
Ground Truth: Table
 <subtask>select(mattress)
 <answer>mattress <bbox>(0.35, 0.49, 0.95, 0.85)
 <subtask>relate(furniture, to the left of,
 [mattress <bbox>(0.35, 0.49, 0.95, 0.85)])
 <answer>nightstand <bbox>(0.18, 0.58, 0.41, 0.82)
 <subtask>query([nightstand <bbox>(0.18, 0.58, 0.41, 0.82)], name)
 <answer>nightstand.

c. Question: What kind of vehicle is red? ✗

Prediction: **Can not find all objects**
Two buses Here, only one output
 <subtask>select(vehicle)
 <answer>bus <bbox>(0.13, 0.01, 0.83, 0.97)
 <subtask>filter color([bus <bbox>(0.13, 0.01, 0.83, 0.97)], red)
 <answer>bus <bbox>(0.13, 0.01, 0.83, 0.97)
 <subtask>query([bus <bbox>(0.13, 0.01, 0.83, 0.97)], name)
 <answer>bus.

d. Question: What's the girl wearing? ✗

Prediction: **Predicate order wrong**
Should be (girl, wearing, ...)
 <subtask>select(girl)
 <answer>girl <bbox>(0.68, 0.46, 0.77, 0.71)
 <subtask>relate(, wearing, [girl <bbox>(0.68, 0.46, 0.77, 0.71)])
 <answer>boots <bbox>(0.7, 0.64, 0.75, 0.71)
 <subtask>query([boots <bbox>(0.7, 0.64, 0.75, 0.71)], name)
 <answer>boots.

e. Question: What is the man wearing? ✗

Prediction: **Wrong Answer**
Should be shorts
 <subtask>select(man)
 <answer>man <bbox>(0.44, 0.41, 0.52, 0.55)
 <subtask>relate([man <bbox>(0.44, 0.41, 0.52, 0.55)], wearing, _)
 <answer>wetsuit <bbox>(0.44, 0.44, 0.52, 0.55)
 <subtask>query([wetsuit <bbox>(0.44, 0.44, 0.52, 0.55)], name)
 <answer>wetsuit

Figure 4. **Examples of successful and failed cases on human evaluation.** We present some successful and failed cases for our SoT prediction based on VISTAR.

results, should also be high. Conversely, when the final answer is incorrect, either the sub-task operations or the in-

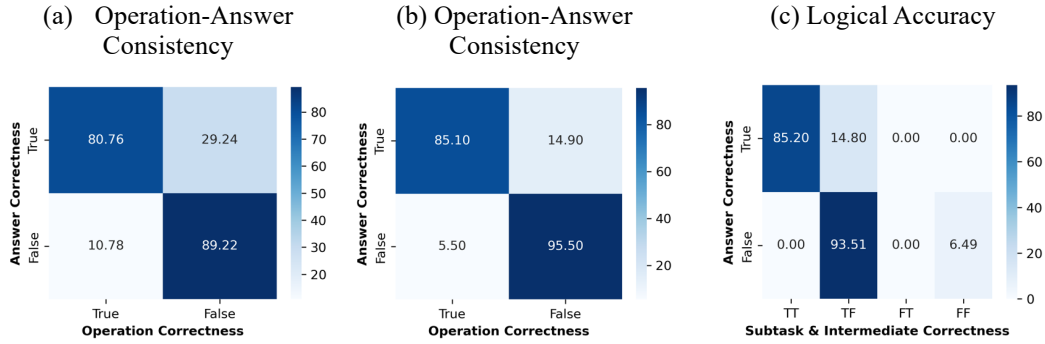


Figure 5. **Consistency evaluation between answers and reasoning steps.** (a) Ablation results without intermediate answer supervision, evaluated by *GPT-4-turbo* over operation and arguments with answers (b) Results from the full training procedure, evaluated by *GPT-4-turbo* over operation and arguments with answers (c) Human evaluation results, assessing the consistency of predicted sub-tasks with their arguments and intermediate results relative to the final answers. Notably, ‘T’ in the figure means true, ‘F’ means false and ‘TT’ indicates both the sub-task and intermediate answer are correct while ‘TF’, ‘FT’, and ‘FF’ represent cases where either the sub-task, intermediate answer, or both are incorrect.

intermediate answers should exhibit a high false ratio. As shown in Fig. 5(c), the correctness ratio of the sub-task operation sequences, including arguments and intermediate results achieves 85.2% when the prediction is correct, which reveals the high consistency of our prediction. For the situation where the answer is true and the operation is true but the intermediate answer is wrong, it is due to the following reasons: 1) Semantically similar, the model predicts an object that is closely related to the ground truth but not identical. For instance (Fig. 4 example (b)), it identifies a nightstand instead of a table. 2) fail to identify all relevant objects sometimes while it does not affect the final prediction, as shown in Fig. 4 example (c), the model correctly localizes one bus, but the scene contains two. 3) the wrong order of predicate, as shown in Fig. 4, example (d), the correct reasoning should follow the structure (girl, wearing, _), but the model sometimes generates (_, wearing, girl), leading to incorrect relational understanding. The majority of incorrect predictions stem from errors in object detection. As shown in Fig. 4 example (e), the model struggles to identify fine-grained objects like shorts.

5.4. Ablation studies

We conduct ablation studies by systematically removing different components in our SoT rationales to analyze their individual contributions to interpretability. Specifically, we evaluate the effects of (1) the inclusion of bounding boxes for visual explanations and (2) intermediate answers at each reasoning step. In terms of interpretability, we continue to perform our ablation studies on the previously mentioned sampled validation set. For the model trained without bounding boxes in SoT, we follow the instruction (see in Appendix §D Fig. 7) to generate the bounding boxes comparing these predictions using adapted IoU. Consequently,

removing the bounding boxes significantly reduces IoU scores from 48% to 44%. Next, we ablate the model trained without intermediate answers. For this setup, we focus on **operation-answer consistency** by examining whether operations with arguments correctly align with the final answers. Specifically, when the final answer is correct, we measure how frequently the corresponding operations with arguments are correct. Conversely, when the final answer is incorrect, we expect the operations with arguments to be predominantly incorrect. Fig. 5(a) shows the operation-answer consistency for the ablated model (trained without intermediate answers) and Fig. 5(b) shows this for our proposed model, VISTAR (trained with intermediate answers). As demonstrated in the figure, there is a decrease in consistency between the correctness of final answers and corresponding operations when intermediate answers are removed compared to our proposed method.

6. Conclusion

In this work, we introduce VISTAR, a framework designed to enhance the interpretability of MLLMs in VQA. By incorporating SoT rationales, VISTAR decomposes complex queries into structured reasoning steps, providing both object-level visual and textual explanations along with accurate answers. However, there are certain limitations. VISTAR’s reliance on datasets with explicit sub-task operations and scene graphs limits its applicability to unstructured datasets, which may be mitigated by generating pseudolabels for them by using predictions from a pre-trained model. Similarly, it cannot generate unseen sub-task operations that were not included in the training. For instance, as OCR operation was not included in the GQA [14] dataset, it fails to perform well in OCR-based tasks on TextVQA [31] and OCR-reliant VQAv2 [10] questions.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. [ArXiv](#), abs/2303.08774, 2023. 1, 7
- [2] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenhang Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, K. Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Yu Bowen, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xing Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. [ArXiv](#), abs/2309.16609, 2023. 2, 5, 6
- [3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. [NeurIPS](#), abs/2005.14165, 2020. 2
- [4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. [ArXiv](#), abs/2107.03374, 2021. 3
- [5] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023. 2
- [6] Wenliang Dai, Junnan Li, DONGXU LI, Anthony Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale N Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning. In [NeurIPS](#), pages 49250–49267. Curran Associates, Inc., 2023. 2, 5, 6
- [7] Yuhao Dong, Zuyan Liu, Hai-Long Sun, Jingkang Yang, Winston Hu, Yongming Rao, and Ziwei Liu. Insight-v: Exploring long-chain visual reasoning with multimodal large language models. [ArXiv](#), abs/2411.14432, 2024. 2
- [8] Difei Gao, Ruiping Wang, Shiguang Shan, and Xilin Chen. Cric: A vqa dataset for compositional reasoning on vision and commonsense. [IEEE TPAMI](#), 45(5):5561–5578, 2023. 5, 6
- [9] Elizaveta Goncharova, Anton Razzhigaev, Matvey Mikhailchuk, Maxim Kurkin, Irina Abdullaeva, Matvey Skripkin, Ivan Osledets, Denis Dimitrov, and Andrey Kuznetsov. Omnifusion technical report. [ArXiv](#), abs/2404.06212, 2024. 1
- [10] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Evaluating the role of image understanding in Visual Question Answering. In [CVPR](#), 2017. 8
- [11] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. [CVPR](#), pages 14953–14962, 2022. 2, 3
- [12] Yushi Hu, Otilia Stretcu, Chun-Ta Lu, Krishnamurthy Viswanathan, Kenji Hata, Enming Luo, Ranjay Krishna, and Ariel Fuxman. Visual program distillation: Distilling tools and programmatic reasoning into vision-language models. [CVPR](#), pages 9590–9601, 2023. 2, 3, 5, 6
- [13] Bin Huang, Xin Wang, Hong Chen, Zihan Song, and Wenwu Zhu. Vtimellm: Empower llm to grasp video moments. [CVPR](#), pages 14271–14280, 2023. 1, 2
- [14] Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. [CVPR](#), pages 6693–6702, 2019. 3, 5, 6, 8, 11
- [15] Abisek Rajakumar Kalarani, Pushpak Bhattacharyya, and Sumit Shekhar. Seeing the unseen: Visual metaphor captioning for videos. [ArXiv](#), abs/2406.04886, 2024. 1
- [16] Fucui Ke, Zhixi Cai, Simindokht Jahangard, Weiqing Wang, Pari Delir Haghighi, and Hamid Rezatofighi. Hydra: A hyper agent for dynamic compositional visual reasoning. In [ECCV](#), pages 132–149. Springer, 2024. 3
- [17] Zaid Khan, Vijay Kumar BG, Samuel Schultzer, Manmohan Chandraker, and Yun Fu. Exploring question decomposition for zero-shot vqa. [NeurIPS](#), 36:56615–56627, 2023. 3
- [18] Jian Li and Weiheng Lu. A survey on benchmarks of multimodal large language models. [ArXiv](#), abs/2408.08632, 2024. 2
- [19] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In [ICML](#), 2023. 2, 3
- [20] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. Grounded language-image pre-training. [CVPR](#), pages 10955–10965, 2021. 3
- [21] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. [CVPR](#), pages 26286–26296, 2023. 1, 2, 5, 6
- [22] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. [NeurIPS](#), 36:34892–34916, 2023. 1, 2
- [23] Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, Yuming Lou, Shang Yang, Haocheng Xi, Shiyi Cao, Yuxian Gu, Dacheng Li, Xiuyu Li, Yunhao Fang, Yukang Chen, Cheng-Yu Hsieh, De-An Huang, An-Chieh Cheng, Vishwesh Nath, Jinyi Hu, Sifei Liu, Ranjay Krishna, Daguang Xu, Xiaolong Wang, Pavlo Molchanov, Jan Kautz, Hongxu Yin, Song Han, and Yao Lu. Nvila: Efficient frontier visual language models. [ArXiv](#), abs/2412.04468, 2024. 1, 2, 5, 6, 7, 11
- [24] Hantao Lou, Changye Li, Jiaming Ji, and Yaodong Yang. Sae-v: Interpreting multimodal models for enhanced alignment. [ArXiv](#), abs/2502.17514, 2025. 2

- [25] Chuofan Ma, Yi Jiang, Jiannan Wu, Zehuan Yuan, and Xiaojuan Qi. Groma: Localized visual tokenization for grounding multimodal large language models. In *ECCV*, pages 417–435. Springer, 2024. 2
- [26] Feipeng Ma, Yizhou Zhou, Yueyi Zhang, Siying Wu, Zheyu Zhang, Zilong He, Fengyun Rao, and Xiaoyan Sun. Task navigator: Decomposing complex tasks for multimodal large language models. *CVPRW*, pages 2248–2257, 2024. 3
- [27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 2
- [28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2
- [29] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE TPAMI*, 44:1623–1637, 2019. 3
- [30] Seyed Hamid Reza Tofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. *CVPR*, pages 658–666, 2019. 6
- [31] Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *CVPR*, pages 8317–8326, 2019. 8
- [32] D’idac Sur’is, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. *ICCV*, pages 11854–11864, 2023. 2, 3, 5, 6
- [33] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023. 2, 3
- [34] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Ke-Yang Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *ArXiv*, abs/2409.12191, 2024. 2
- [35] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiyang Yu, Yingli Zhao, Yulong Ao, Xuebin Min, Tao Li, Boya Wu, Bo Zhao, Bowen Zhang, Lian zi Wang, Guang Liu, Zheqi He, Xi Yang, Jingjing Liu, Yonghua Lin, Tiejun Huang, and Zhongyuan Wang. Emu3: Next-token prediction is all you need. *ArXiv*, abs/2409.18869, 2024. 1
- [36] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *ICCV*, pages 11941–11952, 2023. 2
- [37] Haowei Zhang, Jianzhe Liu, Zhen Han, Shuo Chen, Bailan He, Volker Tresp, Zhiqiang Xu, and Jindong Gu. Visual question decomposition on multimodal large language models. *ArXiv*, abs/2409.19339, 2024. 3
- [38] Haotian Zhang, Haoxuan You, Philipp Dufter, Bowen Zhang, Chen Chen, Hong-You Chen, Tsu-Jui Fu, William Yang Wang, Shih-Fu Chang, Zhe Gan, and Yinfei Yang. Ferret-v2: An improved baseline for referring and grounding with large language models. *ArXiv*, abs/2404.07973, 2024. 1, 2
- [39] Jiarui Zhang, Mahyar Khayatkhoei, Prateek Chhikara, and Filip Ilievski. Mllms know where to look: Training-free perception of small visual details with multimodal llms. *ArXiv*, abs/2502.17422, 2025. 2
- [40] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *ArXiv*, abs/2304.10592, 2023. 2, 3, 5, 6

A. Examples Outputs of Subtask-of-Thought

We present some examples of LLM-generated SoT here, which decomposes complex tasks into sub-task and provides visual explanations, namely object-level bounding boxes, shown in Fig. 6.

	<p>Question: Are there coffee tables to the right of the gray rug? SoT: <subtask>select(rug) <answer>rug (0.12, 0.84, 0.65, 1.0) <subtask>filter color([rug (0.12, 0.84, 0.65, 1.0)], gray) <answer>rug (0.12, 0.84, 0.65, 1.0) <subtask>relate(coffee table, to the right of, [rug (0.12, 0.84, 0.65, 1.0)]) <answer>[None] <subtask>exist([None]) <answer>no.</p>
	<p>Question: Is this fruit both purple and large? SoT: <subtask>select(fruit) <answer>fruit (0.22, 0.68, 0.59, 0.98) <subtask>verify size([fruit (0.22, 0.68, 0.59, 0.98)], large) <answer>[Yes] <subtask>verify color([fruit (0.22, 0.68, 0.59, 0.98)], purple) <answer>[No] <subtask>and(Yes, No) <answer>no.</p>
	<p>Question: The couple underneath the umbrella is using what? SoT: <subtask>select(umbrella) <answer>umbrella (0.09, 0.13, 0.68, 0.45) <subtask>relate(couple, underneath, [umbrella (0.09, 0.13, 0.68, 0.45)]) <answer>couple (0.01, 0.33, 0.59, 0.87) <subtask>relate([couple (0.01, 0.33, 0.59, 0.87)], using, _) <answer>fishing pole (0.57, 0.48, 0.83, 0.58) <subtask>query([fishing pole (0.57, 0.48, 0.83, 0.58)], name) <answer>fishing pole.</p>
	<p>Question: Who is on the animal that is in the water? SoT: <subtask>select(water) <answer>water (0.0, 0.24, 1.0, 0.83) <subtask>relate(animal, in, [water (0.0, 0.24, 1.0, 0.83)]) <answer>elephant (0.25, 0.45, 0.75, 0.72) <subtask>relate(person, on, [elephant (0.25, 0.45, 0.75, 0.72)]) <answer>man (0.44, 0.26, 0.59, 0.49) <subtask>query([man (0.44, 0.26, 0.59, 0.49)], name) <answer>man.</p>

Figure 6. Examples of LLM-generated Subtask-of-Thought

B. Prompt For Subtask-of-Thought Generation

We take each query, related scene graph, sub-task operation sequences and the following prompt as input for *LLaMA-3.1-70B-Instruct* [33] to generate SoT.

These are all the operations possibly presented in the programs:

```
1. select:
{
  Input: ([Object]),
  Output: [Object 1 (location), Object 2 (location), ...]
}
2. exist:
{
  Input: ([Object_List]),
  Output: [Boolean/None]
}
3. relate:
{
  Input: (Subject, Relation, Object)/(Subject, Relation)
        /(Relation, Object),
  Output: [Object 1 (location), Object 2 (location), ...]
}
4. filter:
{
  Input: ([Object_List], Attribute),
  Output: [Object 1 (location), Object 2 (location), ...]
}
5. verify:
{
  Input: ([Object_List], Attribute),
  Output: [Boolean]
},
6. verify rel:
{
  Input: (Subject, Relation, Object),
  Output: [Boolean]
},
7. choose name, hposition (means horizontal position),
   vposition (means vertical position),
   material, location, color, place and other attribute:
{
  Input: (Object, Choice1|Choice2),
  Output: [Right Choice]
}
8. choose rel:
{
  Input: (Subject, Choice1|Choice2)/(Choice1|Choice2, Object),
  Output: [Right Choice]
}
9. and:
{
  Input: (Attribute, Attribute),
  Output: [Boolean]
}
```

```

10. or:
{
  Input: (Attribute, Attribute),
  Output: [Boolean]
}
11. common:
{
  Input: (Object, Object),
  Output: [Attribute]
}
12. query:
{
  Input: ([Object_List], Attribute),
  Output: [Attribute]
}
13. compare:
{
  Input: (Object, Object, Attribute),
  Output: [Attribute]
}
14. same color, shape, material, attr:
{
  Input: (Object, Object),
  Output: [Boolean]
}
15. same:
{
  Input: ([Object_List], Attribute),
  Output: [Boolean]
}
16. different color, shape, material, attr:
{
  Input: (Object, Object),
  Output: [Boolean]
}
17. different:
{
  Input: ([Object_List], Attribute),
  Output: [Boolean]
}

```

Examples:

Question: Do the bananas to the left of the plantains look large and yellow?

Operation:

```

[
  {"operation": "select", "dependencies": [], "argument": "plantains (681259)"},
  {"operation": "relate", "dependencies": [0],
   "argument": "bananas, to the left of s (681264)"},
  {"operation": "verify size", "dependencies": [1], "argument": "large"},
  {"operation": "verify color", "dependencies": [1], "argument": "yellow "},
  {"operation": "and", "dependencies": [2, 3], "argument": ""}
]

```

Object description:

```
{
```

```

"#1":
{
  "id":"681253","name":"banana",
  "attributes":["small","yellow"],"location":[237,87,310,117],
  "relations":["to the left of #10"]
},
"#2":
{
  "id":"681254","name":"meal",
  "attributes":[],"location":[58,121,188,232],
  "relations":[]
},
"#3":
{
  "id":"681255","name":"plate",
  "attributes":["white","full"],"location":[30,111,206,249],
  "relations":["to the left of #5","of #2","with #2","near #6","to the left
↔ of #6"]
},
"#4":
{
  "id":"681256","name":"spoon",
  "attributes":["large","metal","silver"],"location":[0,196,140,261],
  "relations":["on #3","to the left of #5","in #3","to the left of #6"]
},
"#5":
{
  "id":"681257","name":"dish",
  "attributes":["cream colored"],"location":[187,199,295,280],
  "relations":["inside #6","to the right of #4","in #6","to the right of
↔ #3"]},
"#6":
{
  "id":"681258","name":"bowl",
  "attributes":["full"],"location":[178,184,293,283],
  "relations":["next to #3","of #5","near #3",
"to the right of #4","to the right of #8","to the right of #3"]},
"#7":
{
  "id":"681259","name":"plantains",
  "attributes":["red"],"location":[346,0,391,70],
  "relations":["to the right of #12"]
},
"#8":
{
  "id":"681260","name":"rice",
  "attributes":["piled","white"],"location":[57,162,150,219],
  "relations":["on #3","to the left of #6"]},
"#9":
{
  "id":"681261","name":"meat",
  "attributes":["small","brown","delicious"],"location":[68,123,92,150],
  "relations":["on #3","inside #3"]},

```

```

"#10":
{
  "id":"681262","name":"straw",
  "attributes":["white","plastic"],"location":[402,55,417,150],
  "relations":["to the right of #15",
  "to the right of #14","to the right of #1"]},
"#11":
{
  "id":"681263","name":"picnic",
  "attributes":["delicious"],"location":[0,0,499,374],
  "relations":[]},
"#12":
{
  "id":"681264","name":"bananas",
  "attributes":["small","yellow"],"location":[268,32,317,82],
  "relations":["to the left of #7"]},
"#13":
{
  "id":"681265","name":"spots",
  "attributes":[],"location":[245,92,271,108],
  "relations":[]},
"#14":
{
  "id":"681267","name":"banana",
  "attributes":["small","yellow"],"location":[248,55,312,89],
  "relations":["to the left of #10"]},
"#15":
{
  "id":"681268","name":"tablecloth",
  "attributes":["white"],"location":[0,0,396,374],
  "relations":["to the left of #10"]},
"#16":
{
  "id":"681269","name":"onions",
  "attributes":["green"],"location":[90,147,114,163],
  "relations":[]}
}

```

Final Answer: No

Result:

```

[
  {
    "Operation": "select (plantains)",
    "Answer": "[#7 (346, 0, 391, 70)]"
  },
  {
    "Operation": "relate (bananas, to the left of, [#7])",
    "Answer": "[#12 (268, 32, 317, 82)]"
  },
  {
    "Operation": "verify size ([#12], large)",
    "Answer": "[No]"
  },
  {

```

```
    "Operation": "verify color([#12], yellow)",
    "Answer": "[Yes]"
  },
  {
    "Operation": "and(No, Yes)",
    "Answer": "[No]"
  }
]
Question: Are there both bowls and bananas in this image?
Operation:
```

```
[
  {"operation": "select", "dependencies": [], "argument": "banana (681253)"},
  {"operation": "exist", "dependencies": [0], "argument": "?"},
  {"operation": "select", "dependencies": [],
  "argument": "bowl (681258) "},
  {"operation": "exist", "dependencies": [2], "argument": "?"},
  {"operation": "and", "dependencies": [1, 3], "argument": ""}
]
Final Answer: Yes
```

```
Result:
[
  {
    "Operation": "select(banana)",
    "Answer": "[#1 (237, 87, 310, 117),
    #12 (268, 32, 317, 82), #14 (248, 55, 312, 89)]"
  },
  {
    "Operation": "exist([#1, #12, #14])",
    "Answer": "[Yes]"
  },
  {
    "Operation": "select(bowl)",
    "Answer": "[#6 (178,184,293,283)]"
  },
  {
    "Operation": "exist([#6])",
    "Answer": "[Yes]"
  },
  {
    "Operation": "and(Yes, Yes)",
    "Answer": "[Yes]"
  }
]
```

Note:
Operation in Response:

```
1. Important:
{
  "operation": "choose rel", "dependencies": [0],
  "argument": "rice, to the left of|to the right of, s (681260)"
}
```

represents 'rice' is the Subject (681260) of Relation 'to the left of|to the right
↪ of',
should be

```
{
  "choose rel(rice, to the left of|to the right of, ['The Object List of
  ↪ Relation'])".
}
{
  "operation":"verify rel","dependencies":[0],
  "argument":"bed,to the left of,o (2287058)"
}
```

represents 'bed' is the Object (2287058) of Relation 'to the left of', should be

```
{
  "verify rel(['The Subject List of Relation', to the left of, bed])".
}
{
  "operation": "relate", "dependencies": [0],
  "argument": "_ ,in,s (681257)"
}
```

represents '_' is the Subject (681257) of Relation 'in',
should be

```
{
  "relate(_ , in, ['The Object List of Relation'])"
}
{
  "operation": "relate", "dependencies": [0], "argument": "_ ,wearing,o
  ↪ (2029572)"
}
```

represents '_' is the Object (2029572) of Relation 'wearing',
should be

```
{
  "relate(['The Subject List of Relation'], wearing, _)".
}
{
  "operation":"verify rel","dependencies":[1],
  "argument":"skateboard,riding,o (-)"
}
```

represents 'skateboard' is the Object of Relation 'riding',
should be

```
{
  "verify rel('The Subject List of Relation', riding, skateboard)".
}
```

Namely,

"s" in the argument represents current object in the argument is Subject of
↪ current Relation.

"o" in the argument represents current object in the argument is Object of current
↪ Relation.

'_' in the argument represents the placeholder.

"o" and "s" should not appear in the Operation. Ensure the argument follows the
↪ subject-relation-object order based on context.

Intermediate Answer in Response:

1. Important:

For all operations involving selection, like 'select(person)', Answer should include all related objects for the target, including singular and plural in naming or attributes, When the question or operation

- ↪ specifies a relationship or context, such as Operation 'relate' and 'filter',
- ↪ filter the selection to include only those objects that match the specified
- ↪ context or relation.

In cases where a verification is needed, such as checking attributes ('verify size', 'verify color'), ensure that the object being verified is relevant

- ↪ to the relationship or position specified in the previous steps.

Avoid including Unrelated Objects that do not fit the exact positional or

- ↪ contextual requirement of the query.

Use the attributes, locations, and relations provided in the object description to

- ↪ make accurate selections.

2. Important: If the Object does not exist, the answer should be [None].

3. Important: The final answer should always match the logical conclusion drawn

- ↪ from the operations and their reasoning steps. Noticeably, For Operation
- ↪ 'choose rel' in the last step, keeping identical with Final Answer. For
- ↪ example, choose rel(Subject, to the left of|to the right of, Object) should
- ↪ simply Return 'left' or 'right' based on reasoning steps.

4. Important: Answer should align with Given Relations and Locations in Object

- ↪ Description.

5. Important: For Operation 'select(scene)' should return 'there are' with key

- ↪ object IDs without bounding boxes, which are necessary to answer the question
- ↪ like 'there are [#2, #3, #5]' where #2, #3, #5 are key objects ID to answer
- ↪ the question and should be the argument for the next operation.

Based on the above example, Just Output your response in JSON format without Any

- ↪ Extra Explanation, keeping the same format with the given example.

C. Sub-task Operation Definition

In Table 5 and Table 6, we show all the sub-task operations used for the generation of SoT, including the name of each operation, the corresponding arguments, the function of each operation and its output.

Table 5. List of sub-task operations and their descriptions including arguments, meaning and output (part one)

Operation	Arguments	Meaning	Output
select	(object)	find the object	object name with bounding box
relate	(subject, relation, object)	find the subject or object that has the relation with object or subject	object name with its bounding box
filter realism filter brightness filter texture filter depth filter weight filter orientation filter event filter liquid filter company filter race filter hardness filter room filter pattern filter length filter material filter hposition filter position filter size filter pose filter activity filter shape filter height filter age filter sportActivity filter face expression filter length filter cleanliness filter sport filter weather filter state filter thickness filter opaqness filter flavor filter fatness filter width filter tone filter gender filter	objects, attribute	filer the objects that have the specified "attribute value"	object name with bounding box
choose weather choose hposition choose vposition choose color choose name choose material choose location choose size choose place choose younger choose older choose length choose pose choose activity choose height choose less healthy choose sportActivity choose shape choose healthier choose cleanliness choose state choose thickness choose pattern choose fatness choose shorter choose higher choose company choose taller	(subject, choice 1 choice 2) (choice 1 choice 2, object)	choose which attribute value does the subject/object have	choice 1/choice 2

Table 6. List of sub-task operations and their descriptions including arguments, meaning and output (part two)

Operation	Arguments	Meaning	Ouput
choose realism choose larger choose hardness choose smaller choose brightness choose lower choose rel choose age choose weight choose depth choose flavor choose race choose opaqness choose gender choose face expression choose tone choose width choose	(subject, choice 1 choice 2) (choice 1 choice 2, object)	choose which attribute value does the subject/object have	choice 1/choice 2
verify rel	(subject, relation, object)	verify the relation between subject and object	yes/no
verify state verify pose verify height verify location verify position verify size verify material verify length verify weather verify shape verify place verify pattern verify cleanliness verify thickness verify activity verify tone verify hardness verify face expression verify age verify sportActivity verify width verify fatness verify opaqness verify weight verify depth verify gender verify company verify realism verify type verify flavor verify brightness verify texture verify color verify race verify room verify	(object, attribute)	verify is the object has the attribute	yes/no
same	(objects, attribute)	if objects are of the same attributes	yes/no
different	(objects, attribute)	if objects are of different attributes	yes/no
same material	(object 1, object 2)	if objects 1 and object 2 are of the same material	yes/no
same color	(object 1, object 2)	if objects 1 and object 2 are of the same color	yes/no
different color	(object 1, object 2)	if objects 1 and object 2 are of different color	yes/no
common	(object 1, object 2)	what do object 1 and object 2 have in common	attribute
and	(attribute 1, attribute 2)	if both attribute 1 and attribute 2 are true	yes/no
or	(attribute 1, attribute 2)	if either attribute 1 and attribute 2 are true	yes/no

D. Prompt for the output of bounding boxes

For the backbone model *NVILA-8B* [23] and the ablation model that removes bounding box, we need to compare with our visual explanation. Hence, we prompt these models during training using the following prompt, as shown in Fig. 7

```
### For the prediction of the final answer
Question: [Q]
Answer the question using a single word or phrase.
Response: [X]

### For the prediction of bounding box
Question: [Q]
Answer: [X]
Provide the bounding box coordinate of the region [X].
```

Figure 7. Instruction for bounding box prediction during inference

E. Example of converting sub-task operations into subtask-of-thought

In the GQA [14] dataset, they provide sub-task operation sequences, consisting of the sub-task operation and the corresponding arguments. We reformat the sub-task operation sequences by restructuring the format of each operation and its arguments and providing intermediate results after each operation.

```
### Question
What is the name of the item of furniture that has the same color as the curtain that is to the right of the garland?

### Sub-task Operation Sequence
Select: garland → Relate: curtain, to the right, garland → Relate: furniture, same color, curtain → Query: name

### Subtask-of-Thought
<subtask>select(garland)<answer>garland <bbox>(0.51, 0.0, 0.54, 0.09)
<subtask>relate(curtain, to the right of, [garland <bbox>(0.51, 0.0, 0.54, 0.09)])<answer>curtain
<bbox>(0.73, 0.0, 0.87, 0.58)
<subtask>relate(furniture, same color, [curtain <bbox>(0.73, 0.0, 0.87, 0.58)])<answer>couch <bbox>(0.12,
0.48, 0.71, 0.97)
<subtask >query([couch <bbox>(0.12, 0.48, 0.71, 0.97)], name)<answer>couch
```

Figure 8. Example of sub-task steps in GQA [14] dataset and our subtask-of-thought formate. For the operations of the given query, it means first we need to identify the garland in the image, and then find the curtain positioned to the right of the garland. Next, we can locate furniture that has the same color as the curtain. Finally, retrieving the name of the identified furniture is the end. Converting from sub-task operation sequences to our SoT requires the question, scene graph, and ground truth answer, prompt(see in Appendix § B.