Master Thesis

Fine-Grained Vision-Language Modeling for Multimodal Training Assistants in Augmented Reality

---

Author: Haochen Huang (2765409)

Daily Supervisor:	Jiahuan Pei (CWI)
Advisor & 1st Examiner:	Mohammad Alian Nejadi (UvA)
2nd Examiner:	Pablo Cesar (CWI)

A thesis submitted in fulfillment of the requirements for the joint UvA-VU Master of Science degree in Computer Science

July 15, 2024

"You can't connect the dots looking forward; you can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future. You have to trust in something - your gut, destiny, life, karma, whatever. This approach has never let me down, and it has made all the difference in my life."

— Steve Jobs

Abstract

Recent advancements in Augmented Reality (AR) and Vision-Language Modelss (VLMs) have significantly improved interactive user experiences, particularly in training and educational environments. However, when it comes to precise tasks such as manual assembly instruction, these technologies still face inherent limitations in understanding fine-grained details. Specifically, existing VLMs often struggle to accurately comprehend complex scenes and precisely position objects, limiting their full potential in AR-based training environments.

To address this challenge, we introduce a novel dataset specifically developed for AR training tasks, sourced from a diverse collection of multimodal data, including LEGO instruction manuals. This dataset serves as a foundation for a series of vision-language tasks, simulating real-life AR training scenarios such as scene understanding, object detection, and state detection. These tasks are designed to push the boundaries of current VLMs, offering a rigorous benchmark for evaluating their capabilities in handling fine-grained assembly instructions within AR environments.

Our findings demonstrate that even leading VLMs struggle with the challenges posed by our dataset. For instance, GPT-4V, a state-of-the-art commercial model, achieves an F1-score of only 40.54% on state detection tasks, underscoring the need for continued research and dataset development. These results reveal critical gaps in current models' ability to handle detailed vision-language tasks, suggesting the importance of creating more robust datasets and benchmarks to guide future advancements.

Ultimately, this work lays the foundation for future research in integrating VLMs into AR environments, highlighting areas where improvements are necessary and proposing new strategies for overcoming these limitations. By pushing the current boundaries of multimodal learning systems, this research opens the door to more effective and intelligent AR training assistants, driving progress in industrial assembly and beyond.

Acknowledgements

I am profoundly grateful for the opportunities and insights that have significantly shaped my research journey. I extend my deepest appreciation to Jiahuan for providing the initial research opportunity and the brilliant concept that led to a high-quality academic publication at CWI. This project has been instrumental in my academic development.

My sincere thanks also go to Pablo, whose mentorship rendered my tenure at CWI truly memorable. His support greatly enhanced my academic pursuits and personal growth during this enriching period.

I am indebted to Mohammad for his unwavering guidance and meticulous attention to detail, which were crucial in refining my work through multiple drafts. His expertise and patience were key in overcoming the challenges inherent in academic writing.

Reflecting on the past ten months, they stand out as an extraordinary chapter in my life, marked by a change to play in the Mango Cup, six bicycle tire blowouts, and two paper submissions. These experiences have highlighted a period of remarkable growth and camaraderie. Often, I am asked how different my year might have been without this dynamic team and these achievements. It's a question that, intriguingly, will remain unanswered.

Contents

List of Figures

List of Tables

List of Acronyms

AR Augmented Reality

VLM Vision-Language Models

VL Vision-Language

VQA Visual Question Answering

LoRA Low Rank Adaptation

LLM Large Language Models

1

Introduction

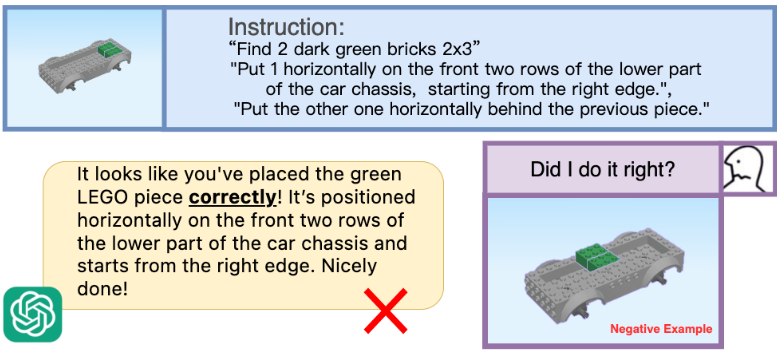


Figure 1.1: Demonstrating the limitations of GPT-4V(2024.2.10) in recognizing errors during LEGO assembly tasks. This figure illustrates an instance where the model incorrectly validates an incorrect step as correct despite having access to previous block selections and detailed step instructions. This example highlights the challenges of fine-grained detail recognition in AI-driven task execution.[2]

VLMs are crucial in integrating visual and textual data, significantly enhancing multimodal understanding. The development of encoder-decoder and transformer-based frameworks has greatly improved VLMs' abilities, particularly evident in their success in tasks like image captioning and visual question answering [51][33]. This advancement has contributed to the creation of robust multimodal representations, essential for the development of innovative assistive technologies[13][6]. In parallel, AR has emerged as a powerful tool for aiding individuals with visual impairments. Research indicates that AR can provide essential visual cues that substantially assist people with visual impairments in educational and navigational contexts[3].

Moreover, AR has proven invaluable in enhancing spatial awareness and improving educational outcomes for students with limited vision[21].In AR applications, VLMs are expected to comprehend and execute fine-grained tasks accurately. The inherent flexibility of VLMs allows for targeted fine-tuning with minimal adjustments, demonstrating their potential for broad applicability across diverse functions[40]. Similar works have also shown possibilities of applying large multi-media models into XR assisting framework.[31]

Effective incorporation of fine-grained visual nuances and robust VL correlations is essential, especially in complex tasks like image difference captioning, which suffer from sparse manual annotations [46]. Furthermore, the development of fine-grained visual-textual representations within VLMs aims to enhance the models' ability to discern intricate relationships between visual and

[2] https://openai.com/index/gpt-4v-system-card/

textual information, emphasizing the need for granularity in improving performance [19]. Additionally, employing deep learning for detailed image analysis addresses critical challenges in computer vision, proving vital for applications that demand precise classification and recognition of complex visual data [44][45].

In a specific scenario, we challenged the capabilities of the latest visual version of OpenAI's GPT-4 (GPT-4V) by presenting the correct answer for a step in a LEGO assembly manual (see Figure 1.1). Subsequently, we showed GPT-4V an image of our assembly attempt, which was intentionally incorrect. Despite the discrepancies between the provided correct answer and our erroneous assembly, GPT-4V validated the incorrect assembly as correct [15]. This incident not only underscores the existing limitations in VLMs' ability to interpret and verify visual tasks against given instructions accurately but also highlights the urgent need for advancements in understanding and validation mechanisms within such models, particularly in precision-critical and correctness-verification applications.

To this end, we have developed a specialized dataset from the Official LEGO dataset[3], specifically designed to enhance the capabilities of VLMs as AI assistants in AR systems. Additionally, we formulated VLM tasks that mirror the challenges found in assistant applications, thus fostering the integration of AR and VLM technologies. These tasks, illustrated in Figure 3.2, emphasize scene understanding, object detection, and state detection, which are crucial for equipping VLMs to manage real-world scenarios adeptly. Moreover, our architecture, shown in Figure 5.1, is crafted for compatibility and extensibility, facilitating the incorporation of advanced VLMs into AR systems.

In summary, our contributions are:

- We introduce fine-grained tasks for evaluating VLMs within AR training environments. These tasks have been applied to infer the performance of multiple state-of-the-art VLMs, promoting advanced instructional design in AR.
- We have developed an instructional multimodal dataset from complex LEGO manuals, establishing a new benchmark for multimodal data application in AR settings. This also includes a structured framework for both inference and fine-tuning of existing benchmark models on our dataset, enabling a thorough assessment of model capabilities and enhancements.
- Our tailored query architecture facilitates VLM application in AR training through comprehensive data preprocessing, feature adaptation, and model tuning. This query setting framework is also the result of an extensive query study, ensuring the settings are optimized based on empirical results and are highly effective in real-world applications.

[3] `https://legoaudioinstructions.com/instructions`

2

Background

2.1 Application-Specific VLMs

Recent advanced models that understand both vision and language have opened up many possibilities in different areas [12, 16, 48]. They can describe visual content in words and generate captions for images [20, 52]. This helps in various ways, like providing descriptions for visually impaired people or helping search engines index visual content. These models can also answer questions about images [4], making it easier for users to interact with visual content, especially for people with disabilities. They can understand phrases like "the red car" to identify objects in images, which is useful for tasks like image retrieval or content recommendation [37]. Additionally, these models enable conversational interactions about visual content [7], making it possible for virtual assistants to understand questions or requests related to images and videos. They can even translate between different types of information [18], like turning text descriptions into images or vice versa, which is helpful for language learning or creating multilingual content. Furthermore, they've led to innovative applications like personalized educational platforms [9] and assistive technologies [39] for visually impaired individuals.

While VLMs can be integrated into educational platforms and interactive learning environments to provide personalized feedback, generate educational content, and help learners understand visual concepts, including systems that describe visual scenes, identify objects, or provide navigation assistance based on visual input, but unfortunately the fine-grained VL modeling has not yet been captured by most applications.

2.2 Fine-Grained VL Modeling

The proliferation of VLMs underscores their significance in merging visual and textual information, presenting various possibilities across various applications. Yet, a noticeable gap emerges when modeling VLMs for fine-grained tasks. This gap delineates the difference between the competencies of current models and the anticipated capabilities necessary for comprehensive scene understanding, precise object recognition, or accurate identification of correctness in assembly tasks. Such a discrepancy underscores the challenges faced by existing models in meeting the intricate demands of tasks that require an advanced level of visual and linguistic synthesis, highlighting an essential area for further inquiry and enhancement within the VLM domain.

While modeling VLMs, the progression and diversification of models underscore their pivotal role in amalgamating visual and textual data. This landscape is significantly enriched by models such as CLIP, which heralds a novel paradigm with its broad-spectrum generalization capabilities across a diversity of tasks, demonstrating a formidable aptitude in zero-shot learning environments and multilingual adaptability.[32]While models like CLIP excel in broad generalization, they may lack the depth of specialization needed for tasks requiring intricate detail and nuanced understanding. EfficientVLM is a significant open-source model characterized by its size, speed, and accuracy optimization through innovative approaches like knowledge distillation and modal-adaptive pruning.[41]But it may not fully address specific complexities of fine-grained visual tasks that demand precise and nuanced interpretation. Adding to this diversity, MiniGPT-v2 is a versatile platform for multi-task learning, effectively bridging large language models with VL tasks. Still, it might not provide the depth of insight and specificity required for complex, fine-grained tasks.[8] Meanwhile, Qwen-VL excels in fine-grained visual comprehension, extending the applications of VLMs beyond traditional boundaries, but it may still struggle with the extreme specificity and contextual understanding required in certain specialized tasks.[2] OSCAR, with its object-semantics aligned pre-training, refines image-text representations, enhancing both understanding and generation tasks and reflecting the increasing sophistication of pre-training techniques. [25] VisionLLM explores the potential of large language models for decoding vision-centric tasks, promoting a versatile approach to task management through linguistic cues.[42] However, These models' adaptability to highly detailed and context-specific tasks remains challenging. The Otter model, while advancing the handling of complex sequential tasks, exhibits limitations in detailed analysis and description. It understands and guides various scenarios using the MIMIC-IT dataset but falls short in fine-grained modeling. This deficiency points to a broader need within the field for Vision-Language Models (VLMs) that can match human visual assistants' detailed comprehension and interaction capabilities in real-world settings [23].

The field's progression is notably driven by introducing VLMs capable of sophisticated task navigation and understanding, offering unprecedented adaptability. This advancement is particularly evidenced in the works of Zhong et al. [50] on procedure-aware video representation from instructional content and Ashutosh et al. [1] on generating task graphs for essential step recognition in instructional videos. These contributions herald a future where VLMs are seamlessly integrated into human-centric environments, demonstrating the capability of AI to undertake complex visual tasks with enhanced clarity and efficiency. However, despite the ability to locate and predict steps in procedural tasks, these models lack the capacity for error correction, underscoring an area ripe for further research and development.

In this work, we design text-image pairs that allow most models to receive data for the fine-grained tasks we create, which can be helpful for testing or training these benchmark models.

2.3 VL Datasets in AR

Many datasets tailored for guided tasks lack detailed task labeling, potentially limiting their effectiveness for training AR-guided assistants.

While rich in scene and state annotations, several datasets lack detailed object labeling, impacting their utility for specific augmented reality applications. The COIN dataset is a comprehensive resource for instructional video analysis, offering rich annotations that facilitate research in sequential task understanding and multimodal learning [38]. Similarly, the HoloAssist dataset by Wang et al. provides 166 hours of egocentric data from mixed-reality headsets, capturing detailed human-AI interaction dynamics for real-world tasks [43]. The HowTo100M dataset, comprising 136 million video clips with automatically transcribed narrations, excels in text-to-video retrieval and action localization but falls short in object-specific annotations [28]. Lastly, the TEACh dataset's 3,000+ interactive dialogues enhance understanding of dialogue dynamics and task execution within domestic settings but do not focus on fine-grained object detection [30].

Dataset	Scenes	Objects	States	VL pairs
HoloAssist[43]	✓	✗	✓	350
Assembly101[34]	✗	✓	✓	4,321
COIN[38]	✓	✗	✓	11,827
RareAct[29]	✗	✓	✓	7,607
HowTo100M[28]	✓	✗	✓	23,611
TEACh[30]	✓	✗	✓	3,215
Cross-task[54]	✗	✓	✓	4,713
EPIC-KITCHENS[11]	✗	✓	✓	89,977
MIMIC-IT[22]	✓	✗	✗	2.8M
Our dataset	✓	✓	✓	35,612

Table 2.1: Comparison between different instructional datasets. A large checkmark (✓) indicates that the dataset includes the task, while a cross (✗) indicates that the dataset does not include the task.

Still, several datasets notably enrich research in action recognition and learning efficiency but often lack detailed descriptions of procedural steps. The Assembly101 dataset, with over 4,000 videos of assembling and disassembling toys, offers multi-view recordings and detailed actions but does not focus on step-by-step procedural guidance [34]. Similarly, the CrossTask dataset employs narrations and step lists to facilitate weakly supervised learning, yet it lacks temporal annotations that detail the sequence of actions within tasks [54]. The EPIC-KITCHENS dataset provides an extensive collection of unscripted kitchen activities with rich annotations on actions and objects but falls short in narrating detailed procedural steps [11]. Lastly, the RareAct dataset captures unconventional interactions, challenging models to interpret complex actions without offering explicit step-by-step guidance [29].

The 'State-Aware Configuration Detection for Augmented Reality Step-by-Step Tutorials' introduces an innovative AR tutorial method that uses a 'consecutive state prior' for object detection to differentiate between similar assembly configurations. This method significantly enhances detection accuracy across various assembly tasks, such as furniture, Lego, and Origami, but it lacks textual descriptions of states and does not address object detection [36]. Meanwhile, the MIMIC-IT dataset provides a rich source of instruction-response pairs for training VLMs in interactive tasks, featuring 2.8 million multimodal pairs. However, while it excels in providing detailed conversational contexts and enhancing VLM functionalities for perception and planning, it falls short in step detection. It lacks scenario-specific descriptions that would provide more direct task guidance [22].

To address the gaps in current instructional datasets, we constructed a table to evaluate whether datasets effectively cover the three components of objects, states, and scenes. This analysis highlights each dataset's capacity to handle these elements, which is crucial for developing comprehensive AR-guided assistants. Additionally, our work introduces a simulated image dataset featuring nearly 400 objects. This dataset is designed for fine-grained tasks, systematically alternating between object detection and assembly functions at each step, optimizing the training and testing of advanced visual language models as detailed in Table 2.1.

3

Fine-grained Vision-Language Modeling

3.1 Query Architecture

To optimize the use of VLMs for intricate tasks, our research focuses on a specific configuration of VLMs that integrates both a visual encoder and a large language model. This setup is depicted in Figure 3.1. Such an architecture is pivotal as it facilitates the effective melding of visual and textual data, thus heightening the accuracy and contextual understanding of the model's responses. Our query design aims specifically to enhance the model's comprehension of our defined tasks. During the inference phase, the visual encoder interprets image data while the text encoder of the Large Language Models (LLM) processes structured queries, collectively producing contextually relevant textual outcomes. This dual-encoder approach ensures that our VLMs are not only efficient in data processing but also precise in executing task-specific directives.

The architecture is designed to deliver precise and context-aware responses, enhancing the model's understanding of complex tasks through several integrated components:

- **Task Tokens:** We introduce specialized tokens such as *[grounding]*, *[object]*, and *[state]* to specify tasks related to scene understanding, object positioning, and state detection. These tokens enhance the model's focus and accuracy.
- **Instructions:** Incorporating relevant manual instructions or task directives provides the model with a contextual framework, ensuring responses align with task requirements.
- **Form Directives:** These directives clarify the expected output format, ensuring outputs are precise and directly applicable.
- **Image Input:** Images, either from manuals or real-world scenes, provide essential visual context, crucial for tasks that involve interpreting current states or identifying objects.
- **Context Examples:** Providing typical Visual Question Answering (VQA) pairs as context examples help the model understand the task and the expected output format more effectively.

3.2 Definition of Fine-Grained Instruction Tasks

Fine-grained task definition in computer vision entails categorizing images into specific sub-categories distinguished by subtle visual differences within the same super-class [35]. This classification is

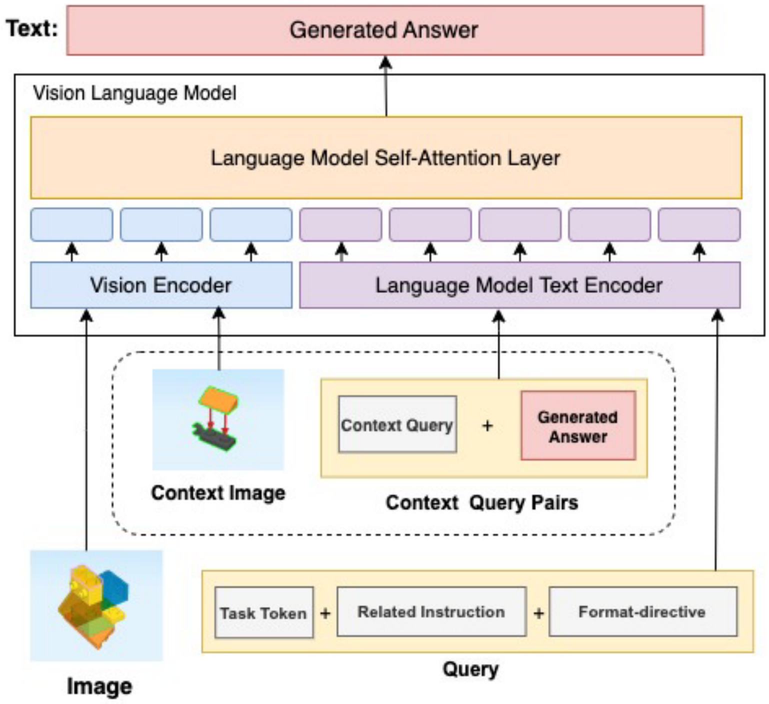


Figure 3.1: Query architecture for AR systems.

challenging due to minor visual variances between subcategories and significant intra-class variations [5]. Fine-grained image analysis, fundamental to computer vision and pattern recognition, supports a variety of real-world applications [44]. Fine-grained visual classification (FGVC) has emerged as a critical research area, focusing on distinguishing subtle differences between visually similar sub-categories [17].

In the context of AR and VLMs, we have defined tasks that address the detailed challenges of fine-grained visual and linguistic comprehension. These tasks aim to assess and enhance VLMs' abilities to interpret complex visual and linguistic cues within AR scenarios. We have introduced three core tasks: Scene Understanding, Object Detection, and State Detection, each targeting essential aspects of visual and linguistic analysis vital for AR applications. Detailed in Figure 3.2 and supported by symbols in Table 3.1, these tasks underpin our research framework and are designed to advance VLM capabilities in educational and training contexts.

To ensure our dataset reflects real-world complexities, we have closely aligned task progression with official manuals, typically alternating between object identification and assembly actions. We structured task pairs as $(t-1, t)$, where step $(t-1)$ involves identifying the required object, and step t involves assembling using this object. This methodology mirrors the procedural flow found in manuals and challenges VLMs to understand and execute a coherent sequence of actions, improving their utility in practical AR-based educational and training settings.

3.2.1 Scene Understanding (T1)

In the context of LEGO assembly, Scene Understanding evaluates the model's ability to interpret and describe a scenario as illustrated in a manual, particularly after obtaining the required items in step $T-1$. The task involves querying the model about step T, where it simulates observing the

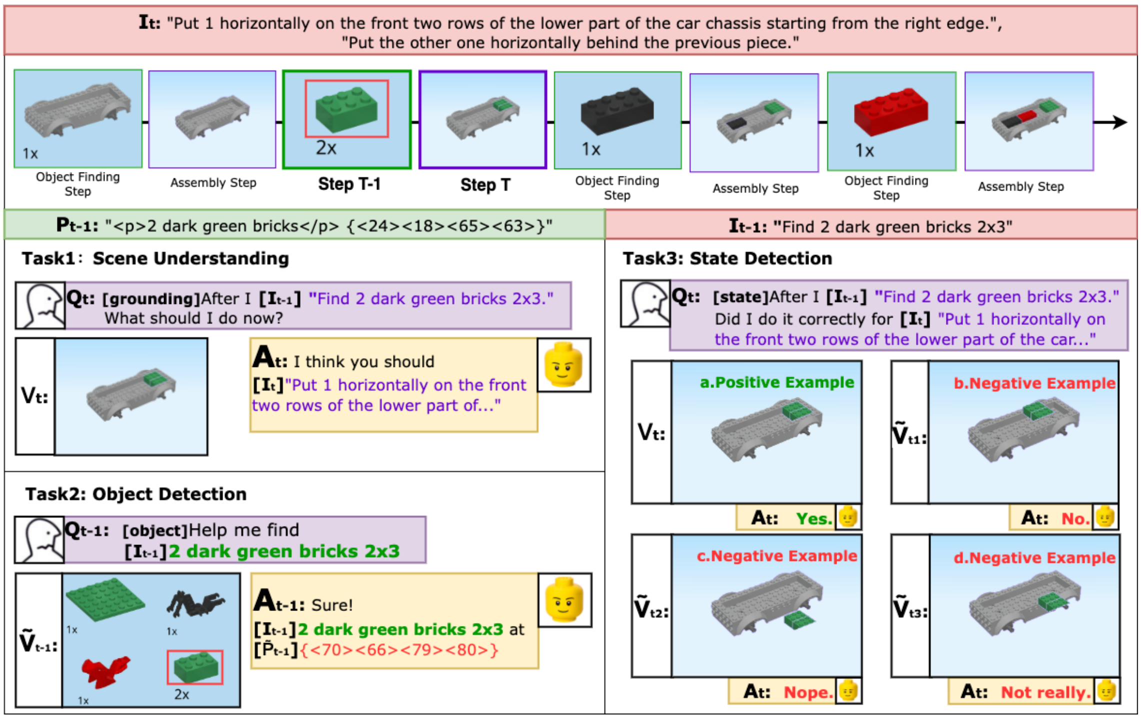


Figure 3.2: This figure illustrates the three core capabilities required by our model within a continuous LEGO assembly manual. The tasks are formulated as VL Pairs derived from specific sections of the manual, requiring the model to (1) understand the scene and identify relevant objects (Scene Understanding), (2) accurately determine the positioning of LEGO pieces based on the assembly step (Object Detection), and (3) detect the current state of assembly to provide context-specific guidance (State Detection). Each of these tasks challenges the model's ability to interpret detailed visual cues and linguistic instructions, demonstrating the need for fine-grained Vision-Language Modeling in creating effective multimodal training assistants for augmented reality environments.[2]

Symbol	Description
S_t	The current assembly step T.
S_{t-1}	The previous object finding step $t-1$.
I_t	Instruction from step T.
I_{t-1}	Instruction from step $t-1$.
P_{t-1}	Coordinates of identified object locations from step $t-1$.
P_{t-1}	Adjusted coordinates after collaging additional images.
Q_t	Query for step T.
Q_{t-1}	Query for step $t-1$.
V_{t-1}	Instruction image for step $t-1$.
V_t	Instruction image for step T.
V_{t-1}	Object image for step $t-1$ combined with three other objects.
V_t	Negative examples for step T
A_t	Proposed answer from the step$T$.
A_{t-1}	Proposed answer from the step$t-1$.

Table 3.1: Description of Symbols Used in the Dataset.

correct scene from the manual. The model must integrate visual cues with contextual knowledge to generate a description closely matching the intended assembly step, I_t. This task assesses the model's object recognition abilities and capacity to grasp the narrative or functional significance of the scene, which is crucial for delivering precise instructional guidance in AR applications.

Composition: Utilizes $I_{(t-1)}$, $Q_{(t)}$, and $V_{(t)}$ to evaluate the model's ability to integrate visual and textual information, identifying both the objects and their required assembly actions within the scene. The output texts are proposed to evaluate with $I_{(t)}$.

3.2.2 Object Detection (T2)

Object Detection in this framework specifically addresses step $T-1$ in the LEGO assembly process, where the model's task is to ascertain the location of specific objects required for the following assembly action. This involves identifying various object categories and precisely determining their positions within the visual input. The model's challenge is to output location information that closely matches the proposed positions $\tilde{P}_{(t-1)}$, essential for accurately guiding the assembly steps. This task is foundational for tasks requiring interaction with or manipulation of objects, impacting the model's efficacy in applications from AR-based educational tools to precise manufacturing automation.

Composition: Leverages $I_{(t-1)}$, $Q_{(t)}$, and $V_{(t-1)}$ to challenge the model's proficiency in pinpointing the precise location of specific, potentially unseen, objects based on prior instructions and visual cues. The output texts are proposed to evaluate with $P_{(T-1)}$.

3.2.3 State Detection (T3)

State Detection in our context is designed to evaluate the model's capability to determine the accuracy of assembly steps within the LEGO building process. Specifically, the user inputs I_t and I_{t-1}, and the model's task is to detect whether the assembly action at step T has been correctly executed based on the progression from I_{t-1} to I_t. This task extends beyond simple object recognition to assess the correctness of the assembly steps, crucial for applications where precise, real-time feedback on task execution is necessary. State detection thus plays a vital role in ensuring that interactive systems can accurately interpret and respond to complex assembly sequences, enhancing the model's utility in educational and detailed task-oriented AR environments.

Composition: Utilizes images from the previous step $I_{(t-1)}$, the query for the current step $Q_{(t)}$, and the visual outputs $V(t)$ or $V_{(t)}$ to assess the model's accuracy in recognizing the state of assembly steps. The evaluation focuses on validating assembly progression. Responses are compared against visual standards; a match with $V_{(t)}$ signifies a correct assembly, while alignment with $V_{(t)}$ suggests an error. This method ensures a meticulous verification of the model's ability to differentiate between correctly and incorrectly executed assembly steps, which is crucial for precision in AR-based instructional applications.

4

Dataset Creation

In this section, we will specifically describe how we create the dataset for all the fine-grained tasks that we defined.

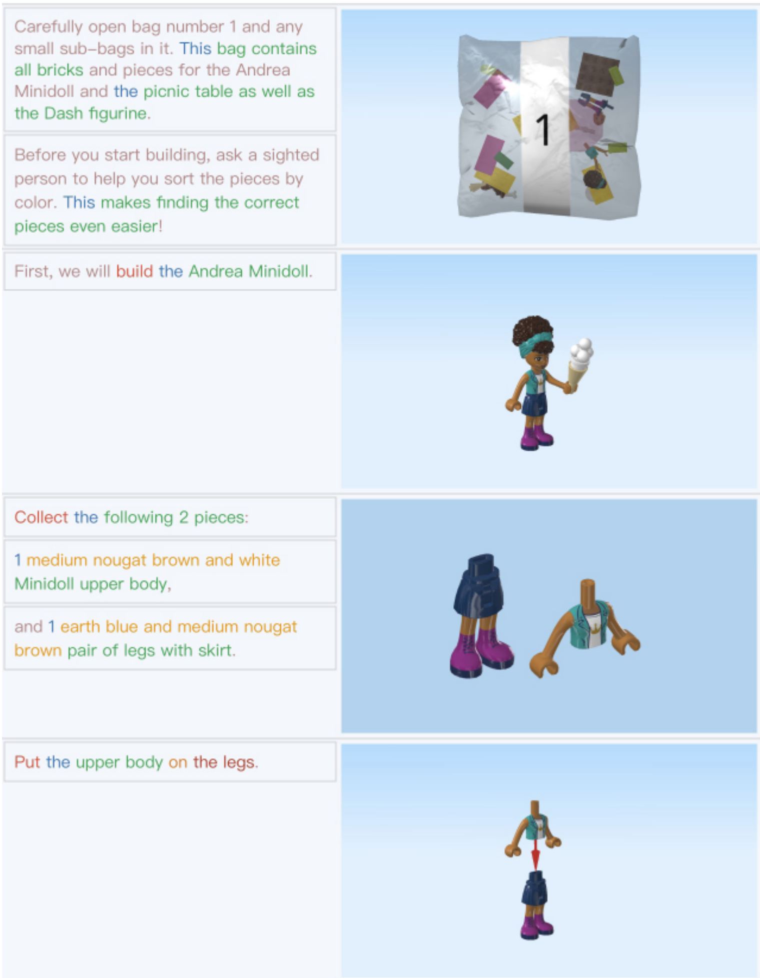


Figure 4.1: An example of LEGO instruction manual. It consists of a summary section at the beginning, followed by three sequential instruction steps. Each step includes textual instructions paired with corresponding images to guide the assembly process.

4.1 Instruction Manual Crawling

Our data collection initiative involved acquiring 65 instruction manuals from the official LEGO website[1], chosen specifically to aid blind and visually impaired users in assembling LEGO sets accurately. Figure4.1 shows an example of the manual. The dataset comprises several essential elements: (i) step-by-step instructions, (ii) corresponding images for each step, and (iii) tags officially provided by LEGO. Instruction tags, extracted from HTML files, categorize each word into classes such as "theme", "verb", "colorinfo", and "target", while image tags, derived from file names, include labels like "start", "eop" (end of part), and "step". The example for transferring the HTML files into JSON files can be seen in Section4.5

4.2 Scene Understanding Dataset Creation

The dataset construction process for scene understanding involves identifying assembly steps labeled as "step" in images, which correspond to a specific assembly step T. To create a contextual scene understanding dataset, we select pairs of images where step T is preceded by an "eop" step, designated as $T-1$. This pairing of I_t and I_{t-1} images forms the basis of our scene understanding data, facilitating the study of sequential task comprehension and execution. We iterated over all manuals to generate the requisite data for this task, ensuring comprehensive coverage and relevance. For this task, we only consider the objects found during the Object-finding step in the assembly process, determining how these objects should be assembled. The answer we seek under this task is the instruction for the assembly step.

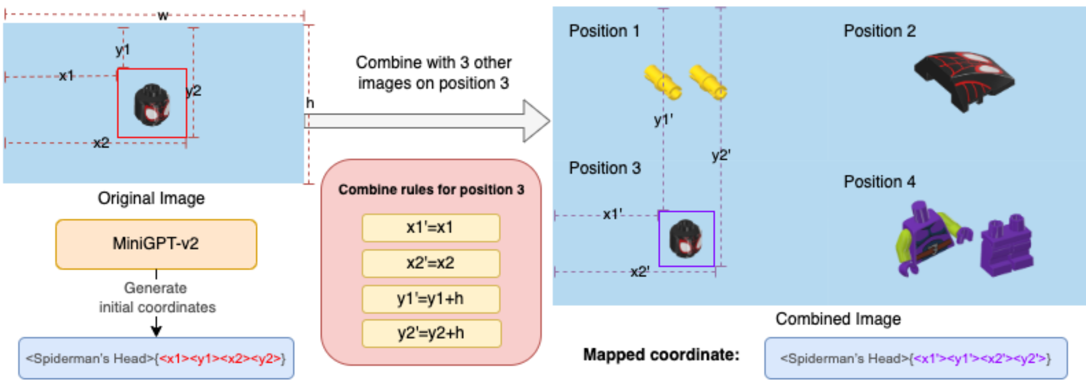


Figure 4.2: Flowchart of object detection dataset creation.

4.3 Object Detection Dataset Creation

The construction of this part of the dataset begins with the formulation of queries from step instructions. Each query is crafted to include a unique token, namely "[detection]", designated for the object detection task. Additionally, each query incorporates a single instruction derived from a step.

[1] https://legoaudioinstructions.com/instructions

In the subsequent stage, these queries and their corresponding images are fed into MiniGPT-v2[8]. This process utilizes MiniGPT-v2 to generate inference outputs, which serve as query responses. The format for these responses is structured as "<Object> <Xleft> <Ytop> <Xright> <Ybottom>".

The next step involves iterating through all instruction steps labeled as "eop" in the dataset, representing points where users are prompted to find specific objects. During each iteration, we employ the MINIGPTv2 model to perform inference on these identified steps, systematically generating results.

After obtaining the coordinate information for a single image, we then randomly select three different objects from the same manual, arranging them in a 2x2 grid in a random order. Using these four different positions, we can map the original coordinates of the single object to the coordinates in the four composite images, thus generating data for the object detection task.

4.4 State Detection Dataset Creation

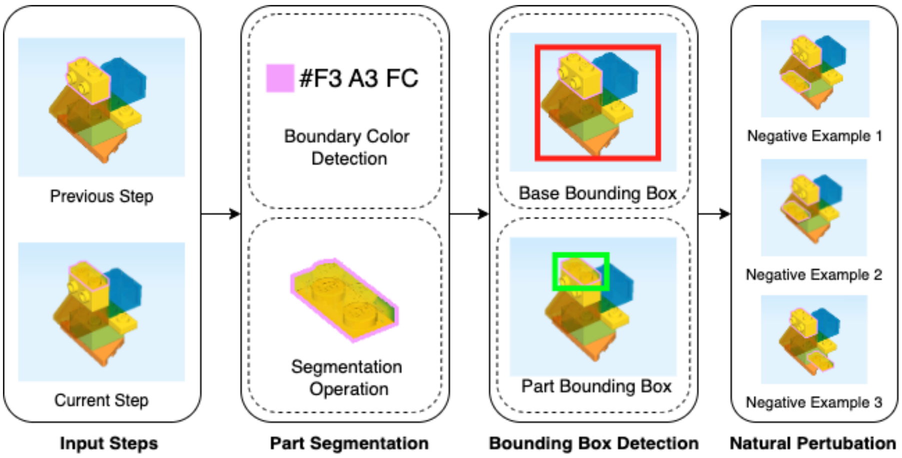


Figure 4.3: Negative sample synthesization for state detection task.

In constructing our dataset, we primarily utilized manuals highlighting the components to be assembled, encompassing 42 manuals. Each assembly session's image was tagged with the names of the items being assembled, revealing that the highlighted colors of the target items varied across different sessions.

Utilizing the K-means clustering algorithm, we initially identified the five predominant colors within each session, providing a broad overview of the color distribution. This step facilitated more focused analyses, where HUE filtering was applied across the session to isolate colors exhibiting higher saturation and frequent occurrence across steps. However, nearly 20 sessions required manual color extraction due to the target colors closely resembling the background hues. These colors were then designated as the boundary colors for target parts, essential for accurately delineating target boundaries.

Our methodology combined broad clustering with targeted HUE filtering to effectively pinpoint target boundary colors, crucial for the negative example generation process (refer to Figure 4.3). For each 'step' labeled image at step T, we took the image from two steps prior, also labeled as 'step' at $T-2$, and performed segmentation of the assembly item by color. We then automatically identified the bounding boxes for the background in steps T and $T-2$. The assembly item was randomly pasted within the background box, ensuring it moved by at least 5%. For each step, we generated three different negative examples. Subsequently, these generated examples underwent a manual screening process, removing any that were unsuitable as negative examples.

In contrast, positive examples were directly derived from the current step's original imagery, ensuring authenticity in representation.

4.5 JSON Schema for Dataset Construction

This section delineates the preliminary JSON data structure utilized in the initial stages of dataset preparation. The described JSON schema encapsulates the vital components necessary for streamlining the data preparation process before final dataset compilation. The structure is especially designed to support the integration of visual and textual data for enhanced model training and application.

```
{
"instruction_id": 95,
"text": [ "Find 2 transparent red flat tiles 1x2." ],
"VLM": {
"img_path": "...",
"parts_img_path": "None",
"id_sign": "Car",
"step_num": 28,
"step_class": "eop",
"other_sign": "AG",
"background_rect": "None",
"color_list": [],
"bound_color": [],
"task_label": "object",
"query": "Find 2 transparent red flat tiles 1x2.",
"MiniGPTv2_output": "<p>2 transparent red flat tiles</p> {<28><17><73><50>}"
},
"img": "...",
"entities": [
{ "this_line": "Find 2 transparent red flat tiles 1x2." },
{ "verb": "Find" },
{ "det": "2" },
{ "colorinfo": "transparent red" },
{ "theme": "flat tiles 1x2" },
{ "suffix": "." }
]
}
```

Table 4.1: Detailed JSON structure for VLM configuration

- **img_path**: Stores the local path of images that are downloaded through web scraping. This path points to the original image files saved after being fetched from the web.
- **id_sign**: Extracted from the image filename, this field denotes the name of the sub-assembly session, such as "CAR" in this example. It uniquely identifies each session within the dataset.

- **step_num**: Indicates the sequence number of the assembly step, which is step 28 in this example. This helps in tracking the progression of assembly steps in the session.
- **step_class**: A classification tag provided officially, such as "eop" in this example, which stands for "end of part" and signifies an object finding step within the assembly process.
- **MiniGPTv2_output**: This field holds the processed output from MiniGPTv2, specifically tailored for the step described. It contains annotations or modifications done post-initial processing, which are crucial for generating new coordinates in subsequent object dataset creation.
- **parts_img_path**, **bound_color**, **color_list**: These fields are reserved for steps classified under "step" in the assembly process. They are critical for the state detection dataset generation:
  - **color_list**: Accumulates all possible boundary colors used throughout the assembly session.
  - **bound_color**: Via a voting mechanism among all steps, this field consolidates the most probable boundary colors, ensuring consistency and accuracy across the dataset.

After establishing the initial JSON file, we selectively extract data corresponding to three distinct tasks and save them into separate JSON files for training and testing purposes. This allows us to tailor the data processing to the specific requirements of different machine learning models. To meet the fine-tuning needs of most models, our default JSON setting is based on the fine-tuning settings of the Llava model. This setup ensures that the JSON structure is optimized for general use cases and provides a good baseline for performance. Below is an example of the JSON structure used: Additionally, the entire dataset, consisting of JSON and PNG files, amounts to approximately 11.8 GB.


```
{
"id": "lego-60263-ocean-mini-submarine-readscr-12",
"image": "sm01-island-0_step_Step%20%23%23E.png",
"conversations": [
{
"from": "human",
"value": "[grounding] After I find 1 reddish brown chest bottom 2x4.
Tell me what to do for the current situation. "
},
{
"from": "gpt",
"value": "Put it horizontally on the table."
} ]
}
```


Table 4.2: Example JSON structure in dataset

For more specialized requirements, such as those needed by the Otter model, we convert all text and image data into Parquet files. This conversion facilitates large-scale distributed training and inference, making it possible to handle vast amounts of data efficiently.

Category	Item	Count
Overall Statistics	Manuals	65
	ObjectfindingStep	4,814
	AssemblyStep	5,614
	AssemblySessions	397
	VQA-Pairs	35612
Scene Understanding	Scene	5,612
	Avg Scenes/Session	14.1
	VQA-Pairs	5,612
Object Detection	Objects	4,784
	Avg Objects/Session	12.1
	VQA-Pairs	19136
State Detection	States	2,716
	Avg States/Session	12.2
	VQA-Pairs	10,864

Table 4.3: Statistics of instruction manuals in the LEGO-ARTA dataset.

4.6 Dataset Statistics

The entire dataset, which includes JSON and PNG files, amounts to approximately 11.8 GB. This size reflects the comprehensive nature of the dataset, encompassing a wide range of data types and formats to support diverse research and application needs.

We summarize the key statistics of our dataset in Table 4.3, which includes detailed metrics across various dimensions essential for evaluating Vision-Language Models in AR settings:

- **Scene Understanding:** Includes 5,614 steps for scene analysis, with an average of 14.1 scenes per sub-object.
- **Object Detection:** Comprises 4,814 steps dedicated to identifying objects, with an average of 12.1 objects per sub-object.
- **State Detection:** Encompasses 2,716 states with 222 distinct boundary colors, averaging 12.2 states per sub-object.

In contrast to other instructional datasets, our dataset uniquely integrates data required for Object Detection, State Detection, and Scene Understanding tasks within the same sub-object, enhancing its utility for comprehensive multimodal research and application in AR contexts.

5

Dataset Quality Assurance

5.1 Dataset Quality Assurance Design

In this project, rigorous quality assurance protocols were employed to ensure the high quality and utility of our dataset. The key steps taken in our quality assurance process are outlined below:

1. Comprehensive Documentation and Code Transparency: We provided complete documentation of the data processing methodologies and all associated code utilized during the production phase. This ensures traceability and transparency, allowing for a detailed understanding of the data's origin and handling. Such transparency is crucial for user comprehension of our documentation and scoring guidelines, highlighting the significance of this work.

2. User Scoring: Users were tasked with evaluating the dataset by scoring the generated coordinates, extracted entities, and the quality of negative examples in two main tasks: object detection and state detection. This manual scoring process ensures that each aspect of the dataset is meticulously assessed for quality and relevance.

3. Quality Assurance Metrics: The scores provided by users were used to calculate the Kappa score to measure inter-rater agreement. A Kappa score above 0.6 was ensured to confirm the reliability of the evaluations. The overall quality of the dataset was classified as "acceptable" or higher based on these evaluations.

These measures affirm our commitment to maintaining stringent data quality and underscore the importance of reliable user feedback. Through this comprehensive quality assurance strategy, we provide a precise and practical resource that significantly contributes to advancing research and development within the field.

5.1.1 Evaluating the Boundary Precision and Entity Disambiguity of the Detected Objects

In our study, we conducted a thorough quality assurance process for the user-generated dataset to evaluate the precision and accuracy of the detected objects. This evaluation focused on two main criteria: boundary precision and entity disambiguity.

Task Description: Given an image and the corresponding reference text, the task was to assess the detected objects highlighted by bounding boxes based on the following criteria:

Boundary Precision (C1): This criterion evaluates the extent to which a bounding box is well-aligned with the object in the image. The precision of the boundary is crucial for ensuring that the object is correctly and completely enclosed without unnecessary space or significant parts of the object being excluded. Scoring: Inaccurate: Assign 1 point if the bounding box does not contain the object it is supposed to enclose, or if the object is entirely missing from the box. Partially Correct: Award 2 points if the bounding box includes the object but the center of the

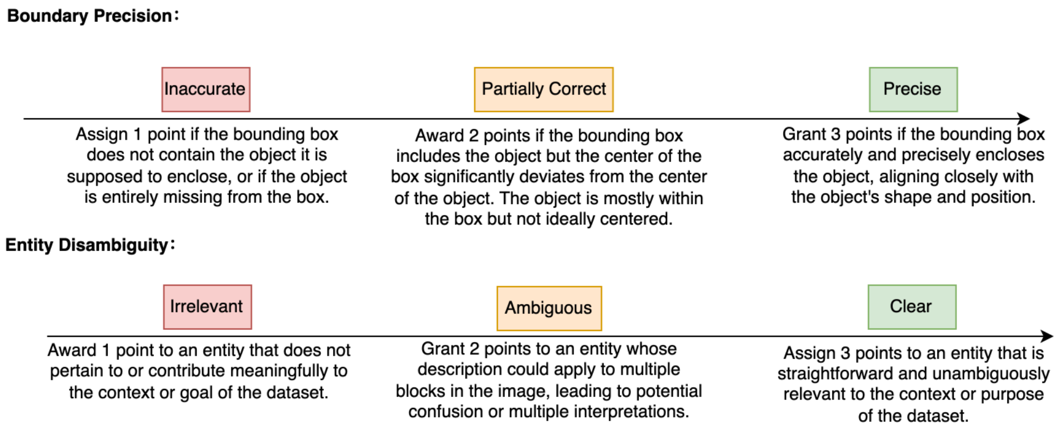


Figure 5.1: Detailed scoring rules for Task 2.

box significantly deviates from the center of the object. The object is mostly within the box but not ideally centered. Precise: Grant 3 points if the bounding box accurately and precisely encloses the object, aligning closely with the object's shape and position.

Entity Disambiguity (C2): This criterion assesses how well the text within the bounding box aligns with the reference text. It is essential that the detected text accurately represents the intended entity without ambiguity, ensuring clarity and correctness in the context of the reference text. Scoring: Irrelevant: Award 1 point to an entity that does not pertain to or contribute meaningfully to the context or goal of the dataset. Ambiguous: Grant 2 points to an entity whose description could apply to multiple blocks in the image, leading to potential confusion or multiple interpretations. Clear: Assign 3 points to an entity that is straightforward and unambiguously relevant to the context or purpose of the dataset. By adhering to these criteria, we ensured that the dataset maintained high standards of quality, crucial for the reliability and validity of our subsequent analyses and model training.

5.1.2 Evaluating the Relevance and Similarity Between the Evaluating Image and the Reference Image

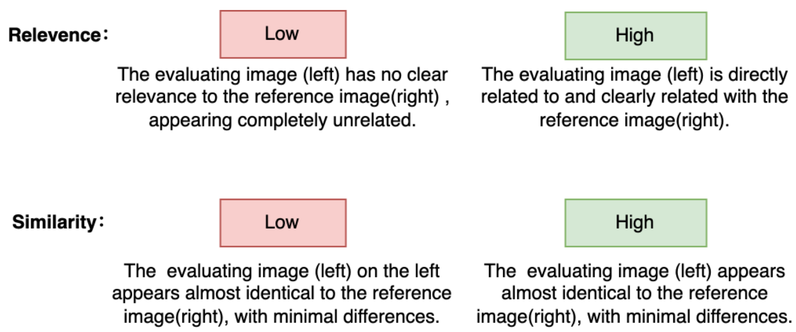


Figure 5.2: Detailed scoring rules for Task 3.

For state detection task, we assessed the dataset quality by evaluating the relevance and similarity between the evaluating image and the reference image. This process ensures that the selected images are both contextually appropriate and visually consistent.

Task Description: Given an evaluating image and a reference image, the task was to assess the relevance and similarity based on the following criteria:

Relevance (C1): This criterion involves evaluating how relevant the evaluating image is to the reference image. It requires an assessment of whether the content or context of the evaluating image directly pertains to the reference image, ensuring that the selected images are contextually appropriate and meaningful. Scoring: Low: The evaluating image (left) has no clear relevance to the reference image (right), appearing completely unrelated. High: The evaluating image (left) is directly related to and clearly related to the reference image (right). Similarity (C2): This criterion involves assessing how similar the evaluating image is to the reference image. The evaluation considers visual and thematic elements that align or correspond between the two images, ensuring consistency and coherence in the dataset. Scoring: Low: The evaluating image (left) appears almost identical to the reference image (right), with minimal differences. High: The evaluating image (left) appears almost identical to the reference image (right), with minimal differences. By following these guidelines, we ensured that the images in the dataset were not only contextually relevant but also visually similar, enhancing the quality and usability of the dataset for our research purposes.

5.2 Quality Assurance Examples

In this section, we will provide detailed examples of quality assurance scoring.

5.2.1 Boundary Precision Examples

As shown in Figure 5.3, the red box differs significantly from the target object, with almost no overlap; this is considered Inaccurate. The orange box partially overlaps with the target object, which is deemed Partially correct. The green box almost perfectly overlaps with the target object, making it Precise.

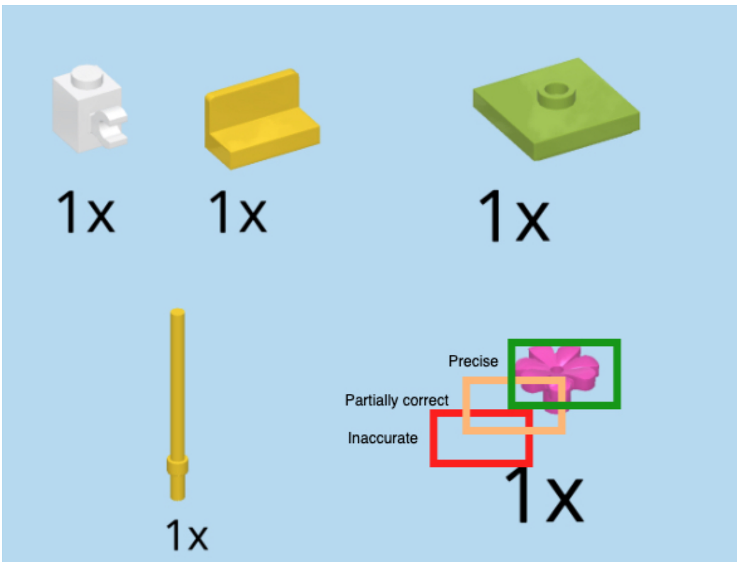


Figure 5.3: Examples for scoring boundaries in Task 2.

5.2.2 Entity Disambiguity Examples

In this section, we present examples of Entity Disambiguity.

The first type is irrelevant, as shown in Figure 5.4. In this example, the entity name "ImageContent" does not match the object within the box, indicating a significant disparity and irrelevance.

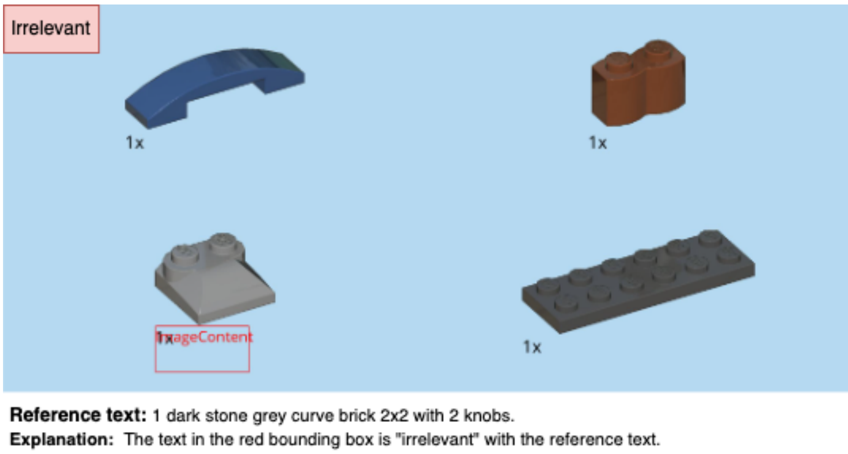


Figure 5.4: Object examples type 1: Irrelevant.

The second type is ambiguous, as shown in Figure 5.5. In this example, there are two entity names that are swapped, causing confusion. Additionally, the entity details are not specified clearly.

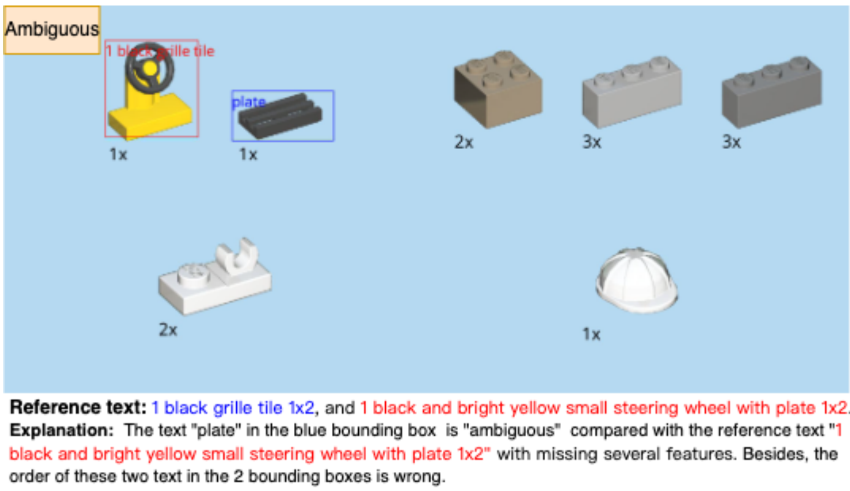


Figure 5.5: Object examples type 2: Ambiguous.

The third type is clear, as shown in Figure 5.6. In this example, the entity name perfectly matches the object, making it very clear.

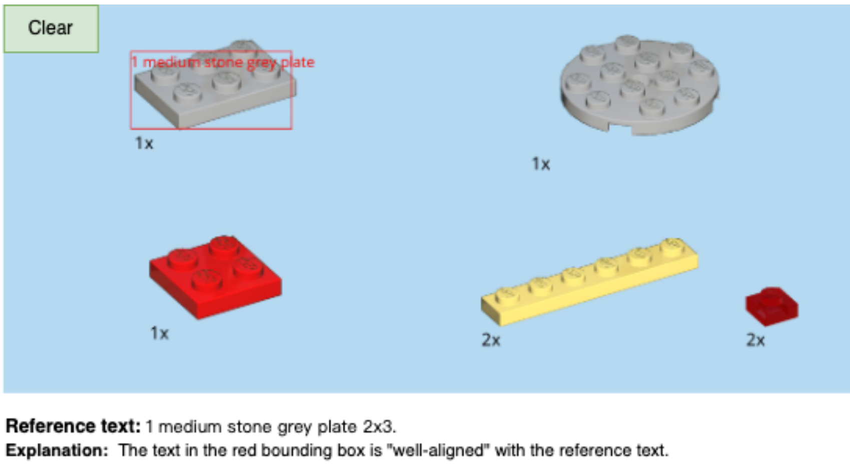


Figure 5.6: Object examples type 3: Clear.

5.2.3 Relevance and Similarity Examples

In this section, we present examples of Relevance and Similarity. On the right side is the correct (golden) example, and on the left side is the simulated example. Note that a category with high similarity and low relevance does not actually exist in this context, because in a manual, two images that look very similar under the same assembly part cannot be entirely unrelated in theme or image.

The first type is high relevance and high similarity, as shown in Figure 5.7. In this example, the evaluated image is both highly relevant and visually similar to the reference image. The differences are minimal, but details such as the addition or displacement of LEGO blocks can be observed.

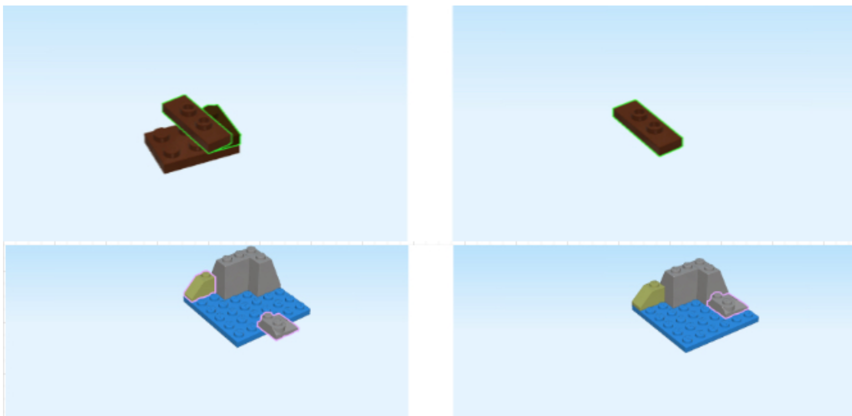


Figure 5.7: State examples type 1: High Relevance and High Similarity.

The second type is high relevance and low similarity, as shown in Figure 5.8. In this example, the evaluated image is relevant to the reference image in context but lacks visual similarity.

Figure 5.8: State examples type 2: High Relevance and Low Similarity.

The third type is low relevance and low similarity, as shown in Figure 5.9. In this example, the evaluated image is neither relevant nor visually similar to the reference image.

Figure 5.9: State examples type 3: Low Relevance and Low Similarity.

6

Implementation

6.1 Dataset Quality and Usage Assurance

This section discusses the quality control experiments for data generated by our automated data generation process and its validation for scientific usability.

6.1.1 User Study for Dataset Quality Assurance

We enlisted five individuals for this experiment to participate in our Dataset Quality Assurance experiment. Each participant assessed the quality of VLpairs generated for Task 2 (Object Detection) and Task 3 (State Detection). The quality assurance materials consisted of scripts randomly selecting 100 VL pairs for each task and automatically generating HTML files containing the data. Task 2 and Task 3 included three and five sample questions, respectively. The correctness of these sample questions ensured that participants fully reviewed our tutorial section. In addition to the automatically generated questions, the HTML files included a comprehensive scoring tutorial. The entire experiment lasted approximately 20-30 minutes. Upon completion of the data scoring, a JSON file with the results was automatically generated. The tutorial section can be found in the Appendix 11.1.

6.1.2 Copyright for LEGO data usage

The data used in our research consists of publicly available LEGO manuals. Our usage is strictly for research purposes and is non-commercial. We have adhered to LEGO's official copyright policies[1],[2].

Our research contributes to the broader scientific community by enhancing the understanding of scene recognition and sequential task execution. By leveraging LEGO manuals, we can create a comprehensive and relevant dataset that aids in developing advanced models for object detection and scene understanding.

Upon acceptance of our paper, we will publish our dataset on HuggingFace, similar to the LEGO Set dataset[3]. This will ensure that other researchers can access and utilize the dataset for further studies, promoting open science and collaboration.

The code used in this study will also be available on GitHub. Doing so aims to facilitate reproducibility and transparency in our research. Sharing our code will enable other researchers to replicate our findings, build upon our work, and contribute to the ongoing advancements in the field of computer vision and scene understanding.

GitHub link: `https://github.com/peterhuang-coding/VLM_ARTA`

[1] https://www.lego.com/en-us/legal/notices-and-policies/open-source/

[2] https://www.lego.com/en-nl/legal/notices-and-policies/fair-play/?locale=en-nl

[3] https://huggingface.co/datasets/merve/lego_sets_latest

6.2 Hardware and Computational Resources

In this study, we utilized the Snellius computational platform[1] to execute inference tasks on state-of-the-art models sourced from the Hugging Face repository[2]. Our dataset was randomized, with 25% reserved for testing to ensure a comprehensive evaluation of model performance. Training and inference phases were conducted using NVIDIA A100 SXM4 40GB GPUs, which significantly enhanced computational efficiency. This framework rigorously tested the model's ability to process complex multimodal data, laying the groundwork for future research in this field.

For model inference and fine-tuning on the Snellius server, we typically allocate GPUs from the "GPU" partition. Detailed computing resource allocation scripts are provided in the Appendix.

6.3 Inference Details

We applied a single GPU for the models' inference sessions in this experiment. Once the test set was fully processed, the results were analyzed separately and saved in JSON files. Detailed inference scripts can be seen in the Appendix11.3.

6.3.1 Query Setting

In this paper, we investigate three unique inference tasks and have devised specific query templates for each, as demonstrated in Table 6.1. We employ three distinct tokens to distinguish these tasks, ensuring that the selection rules and format directives align with the requirements of the respective inference tasks. These tokens enhance system identification and operation execution, thus improving precision and efficiency.

Each query template contains detailed selection rules and format directives that facilitate extracting and utilizing information from inputs for tasks such as scene understanding, object detection, and state detection. For instance, we precisely outline the requirement to locate and mark object positions with detailed coordinates and bounding box specifications in object detection. This specificity guarantees high-quality outputs, enabling accurate system responses to complex scenarios.

Moreover, these refined query templates aimed to enhance the VLM's adaptability to various scenarios, improving its ability to execute tasks accurately across different environments.

6.3.2 In-context Examples

To enhance the models' understanding of the desired outputs for our tasks, we provided two typical examples for each task in this experiment. For InstructBLIP and BLIP2, only text context was given as they do not support image context. For other models, all relevant images and information were provided. Detailed examples can be found in the Appendix11.2.

6.3.3 Query Study

We conducted a series of reduction tests to elucidate the practical effects of each component within our designed queries. These tests involved removing individual components from the queries and performing inference with the best-performing model under our initially designed settings for each task. The components examined in these discussions included the instructions and our in-context examples. This approach allowed us to assess whether the model's understanding of the tasks improved or deteriorated without certain elements of the queries.

[1] https://www.surf.nl/en/services/snellius-the-national-supercomputer

[2] https://huggingface.co/models

Task - Token - Selection Rule for Instruction - Format-Directive
Task: Scene Understanding **Token:** [grounding] **Selection Rule for Instruction:** <step_symbol> with <step>. Instruction from this step. **Format-Directive:** "Tell me what to do for the current situation."
Task: Object Detection **Token:** [object] **Selection Rule for Instruction:** <step_symbol> with <eop>, Word start with "Find" or "collect". Instruction from this step. **Format-Directive:** "Providing the positions in the format: object {<Xleft><Ytop><Xright><Ybottom>}, with X and Y coordinates normalized to [0,100]. Xleft and Ytop for the top-left corner. Xright and Ybottom for the bottom-right corner."
Task: State Detection **Token:** [state] **Selection Rule for Instruction:** <step_symbol> with <step>. Previous <step_symbol> is <eop>. Instruction from this step and previous step. **Format-Directive:** "Just tell me Yes or No."

Table 6.1: Query templates for fine-grained tasks.

6.3.4 GPT-4o Setting

To further validate the challenges posed by the state-of-the-art models, we also conducted inference using OpenAI's new model GPT-4o (released June 2, 2024). By applying the API keys, we ran the entire test set using the best settings (in-context learning) to address the research questions.

6.4 Fine-tune Details

For the fine-tuning process, we merged the data from the three tasks into a single JSON file and applied Low Rank Adaptation (LoRA) techniques for training. All models, except BLIP2 and InstructBLIP, were fine-tuned using DeepSpeed techniques. The average fine-tuning time was 2 hours on 2 GPUs from the Snellius.

It is important to note that MiniGPT-4 does not support VQA fine-tuning, and Otter does not support LoRA for the 7B model, so these two models were not fine-tuned. Additionally, GPT-4 is a closed-source model and cannot be fine-tuned.

Training records were logged in Weights and Biases (wandb). Detailed fine-tuning training settings on Snellius for all models can be found in Appendix11.3.

7

Evaluation

7.1 Benchmarks

The research community has established benchmarks such as MME[14], MMBench[27], and VLMBench[49] to systematically structure and assess complex multimodal tasks. For equitable comparisons, we selected models with comparable language model capacities, around 7B parameters, which have demonstrated strong performance across various competencies, and detailed the differences in their vision encoders.

- **LLaVA[26]**: Integrates a large language model with a CLIP vision encoder, excelling in multimodal tasks through superior integration of visual and textual data, pre-trained on diverse image-text pairs.
- **Otter[22]**: Specializes in few-shot learning using extensive web datasets containing mixed media, enhancing performance in complex visual-textual tasks.
- **Qwen-VL[2]**: Leads in vision-language modeling with exceptional capabilities in text and image comprehension, achieving standout zero-shot performance in key multimodal tasks.
- **MiniGPT-4[53]**: It integrates a frozen visual encoder and a large language model through a single projection layer, enabling it to generate complex multi-modal outputs
- **MiniGPT-v2[8]**: Features a versatile platform tailored for vision-language tasks, using task-specific identifiers to refine task differentiation and performance.
- **BLIP2[24]**: Employs a querying transformer and two-stage pre-training to effectively minimize the vision-language modality gap, optimizing task performance efficiently.
- **mPLUG-OWL2[47]**: Redefines performance benchmarks in multimodal AI with a streamlined vision backbone, demonstrating significant efficiency and perception accuracy.
- **BLIPinstruct[10]**: Innovates in vision-language AI with instruction tuning that transforms diverse datasets, enhancing zero-shot task performance across multiple domains.
- **GPT4o**[1]: A cutting-edge model in the vision-language domain, leveraging an enhanced transformer architecture to excel in generative tasks. GPT4o combines deep learning techniques with a vast multimodal dataset, providing robust performance across a range of challenging visual and textual understanding tasks.

[1] `https://platform.openai.com/docs/models/gpt-4o`

7.2 Evaluation Metric

We consider multiple metrics for comprehensively evaluating specific tasks.

7.2.1 Scene Understanding Metrics

- **Theme Accuracy (T_ACC):** Measures the precision in identifying specific entities ('theme', 'colorinfo') within reference instructions.
- **BLEU:** Assesses grammatical and semantic accuracy in language generation, crucial for the clarity and precision of short instructional texts.
- **ROUGE-1:** Evaluates the balance between recall and precision at the unigram level, essential for ensuring completeness and relevance in succinctly generated content.

7.2.2 Object Detection Metrics

- **Entity Detection Accuracy (E_ACC):** Measures the model's success in accurately identifying critical keywords such as 'theme' and 'colorinfo' from the instructions.
- **Intersection over Union (IoU):** The overlap between the model's predicted bounding box and the true bounding box. This metric directly measures how well the predicted bounding box matches the ground truth, considering both position and size.

7.2.3 State Detection Metrics

- **F1 Score (F1):** Measures the balance between precision and recall, providing a harmonic mean that reflects the accuracy and completeness of the model in identifying relevant instances accurately.
- **True Negative Rate (TNR):** Quantifies the model's ability to correctly identify negatives for a specific category, showcasing its precision in distinguishing non-relevant instances.
- **False Positive Rate (FPR):** Represents the proportion of false alarms, where the model incorrectly classifies negative instances as positive, indicating the misclassification rate among non-relevant instances.

8

Results

8.1 Benchmark Results

In this section, we discuss the performance of a series of open-source benchmark VLMs and a proprietary, closed-source model on tasks specifically designed for our dataset. These tasks have been crafted to rigorously test and demonstrate the capabilities of VLMs in an AR setting, focusing on their ability to understand and interpret complex multimodal inputs.

The results of these evaluations are presented in a structured format across three tables, each corresponding to different aspects of the VLM performance. Table 8.1 details the results for the Scene Understanding task, Table 8.2 for the Object Detection task, and Table 8.3 for the State Detection task. In these tables, a symbol ↑ indicates that higher values represent better performance, whereas a symbol ↓ indicates that lower values are preferable. Additionally, any values that are bolded within these tables indicate the best performance within that column, highlighting the most effective results across different metrics. The symbol '-' indicates that the model is unable to perform VQA LoRA fine-tuning, and a backslash '\' signifies that the value is a meaningless zero, indicating incorrect format output.

These benchmarks not only illustrate the effectiveness of each model in handling specific tasks but also provide insight into the practical applications of these technologies in enhancing AR experiences. The varied performance across different tasks highlights the strengths and limitations of each model, offering valuable insights into where further advancements are needed and how these models can be fine-tuned for optimal functionality in real-world scenarios.

8.1.1 Scene Understanding

In the Scene Understanding task, significant improvements were observed in the ROUGE and BLEU metrics after fine-tuning. For instance, the LLaVA model's ROUGE score increased from 13.63 to 42.97, and its BLEU score from 0.99 to 17.73. These enhancements suggest that the majority of models have become more aligned with the golden labels in their output, indicating better guidance in user language. However, there was no substantial improvement in Theme entity recognition, suggesting that key technical terms have not been learned effectively.

Despite these improvements in general linguistic alignment, Theme accuracy (T_ACC), which is crucial for the accurate recognition of specific LEGO block names and associated color information, did not show significant enhancement even after fine-tuning. This indicates that while the models are becoming better at handling general language tasks, they still struggle to accurately handle more specialized vocabulary associated with LEGO components. For example, after fine-tuning, both miniGPTv2 and mPLUG-OWL2 exhibited a decrease in T_ACC of nearly 2%. This decline further underscores the challenges faced when models are fine-tuned with a generalized approach

Model	T_ACC ↑		ROUGE ↑		BLEU ↑	
PEFT (LoRA)	w/o	w/	w/o	w/	w/o	w/
MiniGPTv2[8]	**12.94**	10.98	34.72	32.56	**9.81**	10.28
LLaVA[26]	12.81	12.68	13.63	**42.97**	0.99	**17.73**
MiniGPT-4[53]	12.68	-	15.05	-	1.88	-
GPT4o[1]	12.50	-	18.67	-	2.00	-
QWENVL[2]	11.90	**13.46**	**29.57**	39.08	5.19	13.13
InstrcutBLIP[10]	9.28	10.07	29.20	36.87	4.91	11.52
mPLUG-OWL2[47]	9.28	6.93	25.23	32.75	3.04	8.46
BLIP2[24]	8.50	9.17	29.55	40.35	6.40	12.50
Otter[22]	3.79	-	12.12	-	1.28	-

Table 8.1: Performance of VLMs on Task 1: Scene Understanding. Symbols: ↑ indicates higher is better, ↓ lower is better, bolded values signify top performance, '-' means no VQA LoRA fine-tuning.

that may not adequately address the specific needs of more specialized tasks like identifying precise LEGO parts. Such findings highlight the necessity for more targeted improvements in model training, particularly in developing strategies that enhance the model's capability to comprehend and utilize technical terms effectively. This gap illustrates the critical need for tailored training approaches that can bridge the current deficiencies, ensuring that advancements in general language processing translate more effectively to specialized domains.

8.1.2 Object Detection

Model	E_ACC ↑		IoU ↑	
PEFT (LoRA)	w/o	w/	w/o	w/
InstructBLIP[10]	**86.66**	69.71	/	47.20
BLIP2[24]	**72.26**	71.40	34.25	40.57
LLaVA[26]	14.03	**81.10**	/	**60.98**
mPLUG-OWL2[47]	55.98	71.70	14.23	34.88
MiniGPTv2[8]	57.02	54.68	**26.98**	25.94
MiniGPT-4[53]	45.34	-	30.20	-
Otter[22]	36.89	-	/	-
QWENVL[2]	32.41	57.91	25.60	30.08
GPT4o[2]	25.00	-	21.68	-

Table 8.2: Performance of VLMs on task 2: object detection. Symbols: ↑ indicates higher is better, ↓ lower is better, bolded values signify top performance, '-' means no VQA LoRA fine-tuning, and '\' indicates that the model is not applicable for this task.

In Object Detection, entity accuracy (E_ACC) was generally high across models. Notably, after fine-tuning, the QWENVL model showed a significant improvement, with E_ACC rising from 32.41% to 57.91%, and the LLaVA model from 14.03% to an impressive 81.1%. Similarly, the mPLUG-OWL2 model increased from 55.98% to 71.1%. These enhancements indicate an enhanced capability to recognize entities within queries. However, the BLIPins model experienced a slight decline, with a 20% decrease in performance, illustrating that such improvements are not uniformly experienced across all models.

However, the Intersection over Union (IoU) scores for the LLaVA, BLIPins, and Otter models were near zero before fine-tuning, with no correct output format produced. Despite changes in prompts, improvements in IoU remained minimal until after fine-tuning. For instance, the LLaVA model, which initially could not address the task, now manages to produce the correct output format and achieved an IoU of 60.98%. On the other hand, except for models like BLIPins that already could generate the correct format, there were no significant enhancements, indicating that while the dataset enables models to learn how to correctly output the necessary formats, the challenge of accurately identifying objects through textual information alone remains considerable.

8.1.3 State Detection

Model	F1 ↑		TNR ↑		FPR ↓	
PEFT (LoRA)	w/o	w/	w/o	w/	w/o	w/
GPT4o[1]	**40.54**	-	56.94	-	43.06	-
QWENVL[2]	39.77	**39.53**	2.01	3.18	97.99	96.82
MiniGPT-4[53]	37.19	-	32.63	-	67.37	-
MiniGPTv2[8]	36.76	38.64	39.58	19.28	60.42	80.72
mPLUG-OWL2[47]	36.82	15.00	36.59	25.00	63.41	35.00
Otter[22]	35.33	-	24.12	-	75.88	-
LLaVA[26]	25.04	30.00	0.00	20.00	99.41	50.00
BLIP2[24]	24.71	28.5	0.00	5.13	100.00	94.87
InstructBLIP[10]	0.00	0.02	**98.78**	**100.00**	**1.22**	**0.00**

Table 8.3: Performance of VLMs on task 3: state detection. Symbols: ↑ indicates higher is better, ↓ lower is better, bolded values signify top performance, '-' means no VQA LoRA fine-tuning.

The State Detection task showcased the strongest performance from the latest model, GPT4o, achieving an F1 score of 40.54. Despite this high F1 score, the actual accuracy rates hovered around 30%, suggesting that while GPT4o is good at balancing precision and recall, it may still be inaccurately classifying a significant portion of instances.

After fine-tuning, the LLaVA and mPLUG-OWL2 models reached an astonishing 100% accuracy, demonstrating their exceptional capability for understanding the scene and context. This remarkable achievement indicates that specific learning adaptations made during fine-tuning have led to highly accurate predictions. However, other models that did not incorporate default filename inputs into their training regimes did not show significant improvements, highlighting the importance of tailored data handling and model training techniques.

From the True Negative Rate (TNR) metrics, it is evident that before fine-tuning, the LLaVA and BLIP2 models were biased towards providing affirmative responses, with TNR close to 0 and False Positive Rate (FPR) near 100. This predisposition suggests a significant overfitting to positive predictions or an inability to correctly identify negative cases, which could severely limit their usability in practical applications.

Conversely, the InstructBLIP model, which had a TNR close to 100 and an FPR near 0, demonstrated a strong tendency to deny, effectively identifying what it should not respond to. However, its F1-score was almost zero, suggesting that while it is effective at rejecting incorrect inputs, it fails almost entirely at providing correct affirmative responses. This significant bias remained unimproved even after fine-tuning, indicating that while the model is cautious in making false alarms, it is overly conservative, missing opportunities to provide valuable information.

These findings underscore the need for a balanced approach in training models, ensuring they are neither too aggressive nor too conservative in their output. Enhancing model performance in state detection not only requires fine-tuning based on accuracy metrics but also careful consideration of how models handle the balance between sensitivity and specificity.

8.2 Query Setting Study Results

In order to identify the optimal query setting that yields the best results for our model, we conducted experiments on various query settings to determine which components most significantly influence the model's inference results. We selected the QWEN-VL model for this experiment due to its satisfactory performance under our base setting, which included both instructions and in-context examples.

As shown in Table 8.4, we systematically removed these components and performed inference experiments. The results indicate that in Task 1, the Theme accuracy is heavily influenced by the presence of prior instructions. Settings that included instructions consistently achieved accuracy rates above 10%, whereas those without instructions fell below this threshold, highlighting the importance of instructions in scene understanding for theme entity recognition. Furthermore, the metrics ROUGE and BLEU, which assess the sequence and format of the output, show that either metric alone can enhance the quality of the generated results. However, when both settings are absent, the performance on these metrics deteriorates significantly.

In Task 2, the Entity Accuracy (E_ACC) shows better results with the presence of instructions, likely because these instructions contain relevant information about the entities, leading to more accurate entity naming. When relying solely on in-context examples, the model tends to correctly guess only the names of simpler entities. This also explains why there is nearly a five-fold difference in this metric between settings with and without instructions. For the Intersection over Union (IoU) metric, results are better when examples are provided, as they help the model output the correct format more reliably. In settings where only instructions are provided without examples, the QWEN-VL model is prone to generating incorrect formats or defaulting to its built-in output patterns. Even with both settings enabling the correct output format, the actual accuracy achieved is only about 25%, which is still far below our requirements.

In Task 3, despite the presence of both settings, the F1 scores remain low, and there is no significant change in the True Negative Rate (TNR) and False Positive Rate (FPR). This suggests that even with extensive settings, the model tends to default to affirmative responses, as indicated by the TNR not exceeding 5% and an FPR over 95%.

Setting		Task 1			Task 2		Task 3		
Instr.	Ex.	T_ACC ↑	ROUGE ↑	BLEU ↑	E_ACC ↑	IoU ↑	F1 ↑	TNR ↑	FPR ↓
✓	✓	11.90	29.57	5.19	32.41	25.60	39.77	2.01	97.99
✗	✓	6.84	24.63	4.58	5.56	22.01	34.00	4.15	97.50
✓	✗	10.43	23.00	4.24	26.37	15.55	34.50	2.80	97.20
✗	✗	5.27	11.02	1.30	3.70	0.10	25.50	1.58	98.40

Table 8.4: Query setting study results for the model QWENVL across different tasks and metrics with Instruction directives only. Symbols: ↑ indicates higher is better, ↓ lower is better, bolded values signify top performance, '-' means no VQA LoRA fine-tuning.

8.3 Dataset Quality Assurance Results

After conducting dataset quality assurance with five participants, we obtained the results as shown in Figure 8.1. For Task 2 (Object Detection), the metrics for clarity and precision scored near 'Clear' (averaging 1.62 on a scale from 0 to 2) and 'Precise' (averaging 1.65 on the same scale),

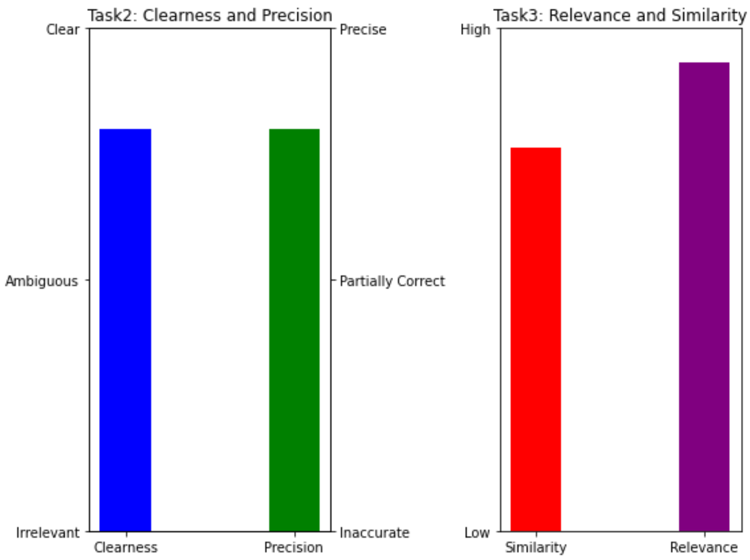


Figure 8.1: Statistic results for quality assurance.

respectively. These scores indicate that both the images generated and their corresponding annotations in this task are quite satisfactory. In Task 3, the metrics for similarity and relevance reached 0.73 and 0.92, respectively, which also suggests acceptable performance in these categories. Additionally, the calculated Kappa scores for all participants were around 0.62, which corresponds to a 'substantial' level of agreement according to the strength of agreement standards. This underlines the reliability of our dataset evaluation process.

9

Discussion

9.1 Task-Specific Examples and Model Performance

In this part, we will discuss the generated results for scene understanding and object detection tasks. In Table 9.1, it is noticeable that the results from the LoRA models in the latter rows predominantly begin with the word "Put", which aligns with the instructive sentences typically found in LEGO manuals. However, regarding content, only the response from MiniGPTv2 closely matches the assembly process. While LLaVA's reaction has been shortened from a long paragraph to a concise sentence, it lacks any meaningful content. Similarly, the models before fine-tuning were generally unable to provide correct instructional phrases.

However, in Table 9.2, it can be seen that models such as LLaVA, Otter, and even MiniGPT-4 struggled to produce correct output formats before fine-tuning. While mPLUG-owl2 provided the correct answer format, it did not fully understand its requirements; for example, it gave an X coordinate of 125, which is incorrect as it exceeds 100. After fine-tuning, most models could provide coordinates accurately, highlighting the challenge of producing acceptable coordinate formats. Models like GPT-4o often attempted to write code for filtering rather than directly understanding the scene. Even when instructed not to create code, it frequently resorted to using Python's PIL library for simple image processing but failed to provide precise coordinates. This demonstrates that using external tools by such models can often lead to improper answers.

9.2 User-friendliness of Object Detection Instructions

In the object detection task, the outputs consist of coordinates the application uses to display a box highlighting the detected objects. However, these coordinates are not intended to be directly shown to users. The initial design aimed to help the model understand the given object in as much detail as possible and provide exact location information. Nevertheless, in real-world interactions, the latency of such multimodal models might render these high-precision outputs less suitable.

Additionally, in many cases, the output from most models regarding positional information is not entirely satisfactory. Therefore, if such models are to be deployed directly for locating uncommon objects, evaluating using lower-precision positional information may be necessary.

9.3 Multi-round Conversation

In the scene understanding task, we have observed cases where some models exhibit high theme accuracy but low ROUGE and BLEU scores. For example, the LLaVA model demonstrates high

[0] `https://pypi.org/project/pillow/`

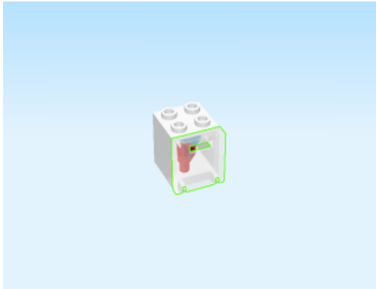
Aspect	Response	Image
Query	[grounding] After I find 1 transparent mailbox front 2x2. Tell me what to do for the current situation.	
LEGO Manual	Insert the two tabs at the bottom of the mailbox casing into the two small notches on the mailbox front.	
BLIP2	Put the mailbox front 2x2 horizontally on the table, clasp to the back. Put the mailbox back 2x2 vertically on top, slope to the left.	
InstructBlip	Put it horizontally on the table, clasp to the back.	
LLaVA	In the image, there is a small Lego-like object with a red liquid inside of it. The object is placed in a blue sky, and it appears to be a 3D model. To create a transparent mailbox, you need to find a transparent mailbox front that is 2x2 in size. Once you find it, you can use it as the base for your transparent mailbox. You can then place the small Lego-like object inside the mailbox to complete the scene.	
MiniGPT-4	Sure! Here's what you need to do with the pieces you have collected:1. Place the transparent mailbox front 2x2 on top of the plate 1x2, aligned with the short side of the plate.you have completed the current situation. Let me know if you need help with anything else!	
Otter	In view of the reasoning 'A mailbox is small and square with a see through window.', the optimal answer should be: build structure.	
mPLUG-OWL2	Place the mailbox front 2x2 next to the block on the table.	
QWEN-VL	Put the transparent mailbox front 2x2 on the right side of the white arch.	
LLaVA(LoRA)	Put it on the mailbox.	
mPLUG-OWL2(LoRA)	Put it on the rightmost edge of the high part of the structure.	
QWEN-VL(LoRA)	Put it on the table.	
MiniGPTv2(LoRA)	Put the mailbox front 2x2 horizontally in front of the previous piece.	

Table 9.1: Scene understanding inference results.

T_ACC (theme accuracy) but low ROUGE and BLEU scores. Conversely, InstructBLIP has a different performance pattern:

The core objective of this task is to assess the model's ability to understand the scene and provide accurate guidance to the user. The BLEU metric evaluates overall similarity to reference texts. InstructBLIP tends to produce concise sentences, aligning with the manual's instructive nature,

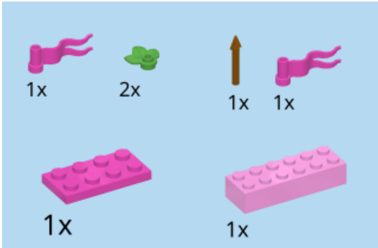
Aspect	Response	Image
Query	[detection]Please identify and highlight the positions of 1 light purple brick 2x6. Providing the positions in the format: object <Xleft><Ytop><Xright><Ybottom>, with X and Y coordinates normalized to [0,100]. Xleft and Ytop for the top-left corner.Xright and Ybottom for thr bottom-right corner.	
LEGO Manual	1 light stone grey plate <59><3><91><39>	
BLIP2	<p>1 light purple brick</p> <57><15><82><35><p>and<p>1 dark purple brick 2x6</p> <82><25><93><36>	
InstructiBLIP	<p>1 light purple brick 2x6</p> <57><15><82><35><p>and<p>1 light purple brick 2x6</p> <82><25><93><36>	
LLaVA	The light purple brick is located at the center of the image, surrounded by the yellow and black bricks.	
MiniGPT-4	Sure! Here are the positions of the light purple brick in the image: Xleft = 0, Ytop = 25, Xright = 30, Ybottom = 50.	
Otter	The 1 light purple brick 2x6 is on the left side of the image.	
mPLUG-OWL2	1 light purple brick <125><48><45><97></s>	
QWEN-VL	<20><61><30><78>	
LLaVA(LoRA)	1 light purple brick <62><15><87><35>	
mPLUG-OWL2(LoRA)	1 light purple brick <60><24><69><35></s>	
QWEN-VL(LoRA)	1 light purple brick <57><15><82><35>	
GPT-4o	“‘pythonPIL import Image Load the imageimage_path =...	

Table 9.2: Object detection inference results.

but often misses subtle details like shape, orientation, and color, leading to a lower T_ACC. Conversely, LLaVA captures these details accurately, resulting in a high T_ACC but generates longer, descriptive responses, resulting in a lower BLEU score. Thus, InstructBLIP excels in brevity but lacks detail, whereas LLaVA provides detailed descriptions at the cost of brevity and overall similarity.

The issue arises from the fact that we simulate multi-round conversations through prompts. Some models tend to provide an overall scene description instead of following our conversational guidance for the next step. This indicates that our assessment of a model's ability to guide the user can be improved by simulating continuous steps in our VQA task detection. By incorporating multi-round conversations as in-context examples, we can help models better understand the task.

9.4 Dataset Diversity

Our primary research motivation is to apply VLMs for practical scenarios of industrial product assembly in an AR environment. However, this framework can be extended to other types of tasks, such as vision-language tasks with and without instruction guides and images. Different tasks and settings can be easily defined by constructing applicable queries within the proposed framework.

Under our designed tasks, as long as there are continuous assembly steps and object-finding steps matched with corresponding VL pairs, dataset construction, and model capability evaluation can be performed using our approach. This highlights the versatility of our method. However, it is important to note that some datasets may not be as straightforward as the LEGO manual, which clearly highlights assembly points. For state detection tasks, the model may need to compare and analyze images before and after each step to determine changes.

By focusing on constructing diverse datasets with clear instructional guides and relevant VL pairs, we can better evaluate and enhance the model's ability to understand and assist in complex tasks.

9.5 Future Works

Upon concluding this series of experiments, significant improvements were observed across all three tasks when fine-tuning models with our bespoke dataset. This progress opens several potential avenues for future research, which we delineate below:

- **Generation of Graphical Manuals:** Our current study assumes a linear assembly process with input from manuals detailing only a finite number of steps. This assumption precludes scenarios such as step-skipping that are plausible during actual assembly tasks. Future work could explore methodologies to transition from linear to graphical manuals, potentially using algorithms that learn different assembly strategies or expand linear manuals into graphical formats. Such advancements would challenge the prevailing assumption of sequential task execution and bring vision assistance technologies closer to human-like interaction and understanding.

- **Dataset Quality Assurance Process Improvement:** Due to our HTML questionnaire being auto-generated by Python, the questions designed to check if users understand the tutorial were inadvertently placed at the beginning. Although this arrangement allows for the entire questionnaire to be completed within 25 minutes, a more effective approach would involve randomly interspersing these comprehension checks throughout the sections related to the two main tasks. This strategy would not only assess users' understanding more effectively but also help evaluate whether fatigue influences their performance.

- **Combined Tasks:** Our initial task design—encompassing scene comprehension, object detection, and state recognition—was informed by an extensive literature review. In practice, these capabilities might be employed concurrently. For instance, when a user incorrectly assembles a part, state detection could first assess the error, followed by object detection to identify the incorrect piece, and finally, scene understanding could guide the user on corrective measures. Further exploration into developing tasks that simulate a more comprehensive range of realistic instructional interactions may yield novel methods to generate data for authentic guidance-oriented question-and-answer scenarios.

10

Conclusion

10.1 Conclusion

This research highlights the intricate challenges involved in effectively integrating Vision-Language Models (VLMs) with AR technologies to enhance educational and interactive applications. By developing a bespoke dataset derived from LEGO instruction manuals and formulating precise multimedia tasks, we have advanced the field in mitigating prevalent limitations in scene comprehension, object detection, and state recognition—key areas where VLMs often underperform. The deployment of our novel query architecture, coupled with AR-specific tasks, represents significant progress in harnessing the full potential of VLMs for complex instructional tasks. Our comprehensive evaluation of contemporary VLMs demonstrates our approach's effectiveness and points to considerable opportunities for future advancements in model accuracy and functionality. These findings set a promising foundation for further research and development within the realm of augmented instructional technologies, potentially revolutionizing how educational content is delivered and interacted with in digital environments.

Bibliography

[1] Kumar Ashutosh, Santhosh Kumar Ramakrishnan, Triantafyllos Afouras, and Kristen Grauman. Video-mined task graphs for keystep recognition in instructional videos, 2023. 4

[2] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. 2023. 4, 25, 28, 29

[3] R. Baragash, H. Al-Samarraie, L. Moody, and F. Zaqout. Augmented reality and functional skills acquisition among individuals with special needs: a meta-analysis of group design studies. *Journal of Special Education Technology*, 37:74–81, 2020. doi: 10.1177/0162643420910413. 1

[4] Yakoub Bazi, Mohamad Mahmoud Al Rahhal, Laila Bashmal, and Mansour Zuair. Vision–language model for visual question answering in medical imagery. *Bioengineering*, 10(3):380, 2023. 3

[5] A. Bera, D. Bhattacharjee, and M. Nasipuri. Deep neural networks fused with textures for image classification. pages 103–111, 2023. doi: 10.1007/978-981-99-2680-0_10. 7

[6] E. Cetinić. Towards generating and evaluating iconographic image captions of artworks. *Journal of Imaging*, 7:123, 2021. doi: 10.3390/jimaging7080123. 1

[7] Feilong Chen, Duzhen Zhang, Xiuyi Chen, Jing Shi, Shuang Xu, and Bo Xu. Unsupervised and pseudo-supervised vision-language alignment in visual dialog. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 4142–4153, 2022. 3

[8] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*, 2023. 4, 12, 25, 28, 29

[9] Ta-Chung Chi, Minmin Shen, Mihail Eric, Seokhwan Kim, and Dilek Hakkani-Tur. Just ask: An interactive learning framework for vision and language navigation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2459–2466, 2020. 3

[10] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023. 25, 28, 29

[11] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018. 5

[12] Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. A survey of vision-language pre-trained models. *arXiv preprint arXiv:2202.10936*, 2022. 3

[13] B. Farah, S. Ayache, B. Favre, and E. Salin. Are vision-language transformers learning multimodal representations? a probing perspective. *Proceedings of the Aaai Conference on Artificial Intelligence*, 36:11248–11257, 2022. doi: 10.1609/aaai.v36i10.21375. 1

[14] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, and Rongrong Ji. Mme: A comprehensive evaluation benchmark for multimodal large language models, 2024. 25

[15] M. Fu, W. Fang, S. Gao, and J. Hong. Edge computing driven scene-aware intelligent augmented reality for manual assembly. 2021. doi: 10.21203/rs.3.rs-736006/v1. 2

[16] Zhe Gan, Linjie Li, Chunyuan Li, Lijuan Wang, Zicheng Liu, Jianfeng Gao, et al. Vision-language pre-training: Basics, recent advances, and future trends. *Foundations and Trends® in Computer Graphics and Vision*, 14(3–4):163–352, 2022. 3

[17] J. Guang and J. Liang. Cmsea: compound model scaling with efficient attention for fine-grained image classification. *Ieee Access*, 10:18222–18232, 2022. doi: 10.1109/access.2022.3150320. 7

[18] Jeremy Gwinnup and Kevin Duh. A survey of vision-language pre-training from the lens of multimodal machine translation. *arXiv preprint arXiv:2306.07198*, 2023. 3

[19] X. He. Fine-grained visual-textual representation learning. *Ieee Transactions on Circuits and Systems for Video Technology*, 30:520–531, 2020. doi: 10.1109/tcsvt.2019.2892802. 2

[20] Xiaowei Hu, Zhe Gan, Jianfeng Wang, Zhengyuan Yang, Zicheng Liu, Yumao Lu, and Lijuan Wang. Scaling up vision-language pre-training for image captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17980–17989, 2022. 3

[21] M. Jdaitawi and A. Kan'an. A decade of research on the effectiveness of augmented reality on students with special disability in higher education. *Contemporary Educational Technology*, 14:ep332, 2021. doi: 10.30935/cedtech/11369. 1

[22] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Mimic-it: Multi-modal in-context instruction tuning. *arXiv preprint arXiv:2306.05425*, 2023. 5, 25, 28, 29

[23] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. Otter: A multi-modal model with in-context instruction tuning, 2023. 4

[24] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023. 25, 28, 29

[25] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16*, pages 121–137. Springer, 2020. 4

[26] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 25, 28, 29

[27] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. Mmbench: Is your multi-modal model an all-around player?, 2023. 25

[28] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips, 2019. 4, 5

[29] Antoine Miech, Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Andrew Zisserman. Rareact: A video dataset of unusual interactions. *arxiv:2008.01018*, 2020. 5

[30] Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. Teach: Task-driven embodied agents that chat, 2021. 4, 5

[31] Jiahuan Pei, Irene Viola, Haochen Huang, Junxiao Wang, Moonisa Ahsan, Fanghua Ye, Jiang Yiming, Yao Sai, Di Wang, Zhumin Chen, Pengjie Ren, and Pablo Cesar. Autonomous workflow for multimodal fine-grained training assistants towards mixed reality, 2024. URL `https://arxiv.org/abs/2405.13034`. 1

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 4

[33] W. Robbins. Towards multimodal vision-language models generating non-generic text. *Proceedings of the Aaai Conference on Artificial Intelligence*, 36:13138–13139, 2022. doi: 10.1609/aaai.v36i11.21705. 1

[34] F. Sener, D. Chatterjee, D. Shelepov, K. He, D. Singhania, R. Wang, and A. Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. *CVPR 2022*, 2022. 5

[35] J. Sheng, H. Yao, X. Sun, and S. Zhou. Unsupervised semantic deep hashing. *Neurocomputing*, 351:19–25, 2019. doi: 10.1016/j.neucom.2019.01.020. 6

[36] Ana Stanescu, Peter Mohr, Mateusz Kozinski, Shohei Mori, Dieter Schmalstieg, and Denis Kalkofen. State-aware configuration detection for stap-by-step augmented reality tutorials. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2023. 5

[37] Sanjay Subramanian, William Merrill, Trevor Darrell, Matt Gardner, Sameer Singh, and Anna Rohrbach. Reclip: A strong zero-shot baseline for referring expression comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022. 3

[38] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1207–1216, 2019. 4, 5

[39] Bryan Wang, Yuliang Li, Zhaoyang Lv, Haijun Xia, Yan Xu, and Raj Sodhi. Lave: Llm-powered agent assistance and language augmentation for video editing. *arXiv preprint arXiv:2402.10294*, 2024. 3

[40] D. Wang, L. Feng, J. Ye, J. Zou, and Y. Zheng. Accelerating the integration of chatgpt and other large-scale ai models into biomedical research and healthcare. *Medcomm – Future Medicine*, 2, 2023. doi: 10.1002/mef2.43. 1

[41] TW Wang, T Wang, et al. Efficientvlm: Fast and accurate vision-language models via knowledge distillation and modal-adaptive pruning (2022). *arXiv preprint arXiv:2210.07795*, 2022. 4

[42] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *Advances in Neural Information Processing Systems*, 36, 2024. 4

[43] Xin Wang, Taein Kwon, Mahdi Rad, Bowen Pan, Ishani Chakraborty, Sean Andrist, Dan Bohus, Ashley Feniello, Bugra Tekin, Felipe Vieira Frujeri, Neel Joshi, and Marc Pollefeys. Holoassist: an egocentric human interaction dataset for interactive ai assistants in the real world, 2023. 4, 5

[44] X. Wei, Y. Song, O. Aodha, J. Wu, Y. Peng, J. Tang, J. Yang, and S. Belongie. Fine-grained image analysis with deep learning: a survey. *Ieee Transactions on Pattern Analysis and Machine Intelligence*, 44:8927–8948, 2022. doi: 10.1109/tpami.2021.3126648. 2, 7

[45] Y. Xie, Q. Gong, X. Luan, J. Yan, and J. Zhang. A survey of fine-grained visual categorization based on deep learning. *Journal of Systems Engineering and Electronics*, pages 1–20, 2023. doi: 10.23919/jsee.2022.000155. 2

[46] L. Yao, W. Wang, and Q. Jin. Image difference captioning with pre-training and contrastive learning. *Proceedings of the Aaai Conference on Artificial Intelligence*, 36:3108–3116, 2022. doi: 10.1609/aaai.v36i3.20218. 1

[47] Qinghao Ye, Haiyang Xu, Jiabo Ye, Ming Yan, Anwen Hu, Haowei Liu, Qi Qian, Ji Zhang, Fei Huang, and Jingren Zhou. mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration, 2023. 25, 28, 29

[48] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 3

[49] Kaizhi Zheng, Xiaotong Chen, Odest Chadwicke Jenkins, and Xin Eric Wang. Vlmbench: A compositional benchmark for vision-and-language manipulation, 2022. 25

[50] Yiwu Zhong, Licheng Yu, Yang Bai, Shangwen Li, Xueting Yan, and Yin Li. Learning procedure-aware video representation from instructional videos and their narrations, 2023. 4

[51] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. Corso, and J. Gao. Unified vision-language pre-training for image captioning and vqa. *Proceedings of the Aaai Conference on Artificial Intelligence*, 34:13041–13049, 2020. doi: 10.1609/aaai.v34i07.7005. 1

[52] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13041–13049, 2020. 3

[53] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models, 2023. 25, 28, 29

[54] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos, 2019. 5

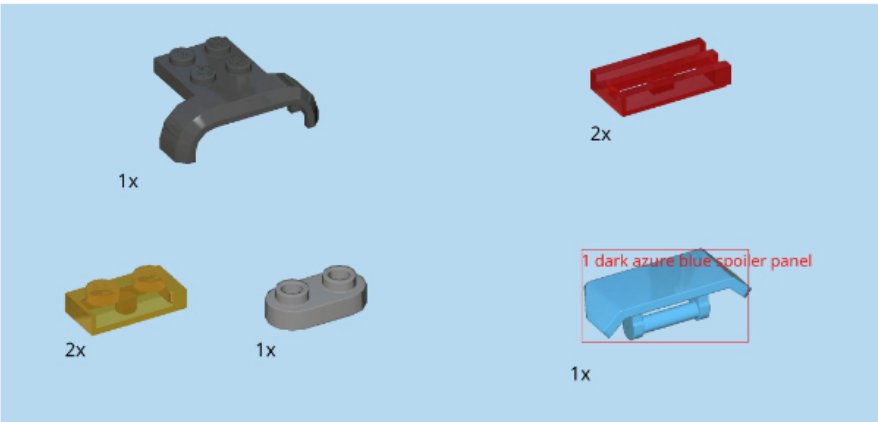
11

Appendix

11.1 Quality Assurance Settings

This section provides the quality assurance settings and example questions for two tasks.

Question 7



Reference text: 1 dark azure blue spoiler panel 2x4 with sloped ends and bottom shaft.

Bounding Boxes Precision:
- ○ Inaccurate
- ○ Partially precise
- ○ Precise

Entities Clearness:
- ○ Irrelevant
- ○ Ambiguous
- ○ Clear

Figure 11.1: Examples question for Task2 in quality assurance.

Criteria	Description
C1: Boundary precision	To what extent a bounding box is well-aligned with the object in an image.
C2: Entity disambiguity	To what extent the text in the bounding box is well-aligned with the reference text.
Total Questions	103
Time to Complete	20-40 minutes

Table 11.1: Evaluation criteria for Task 2.

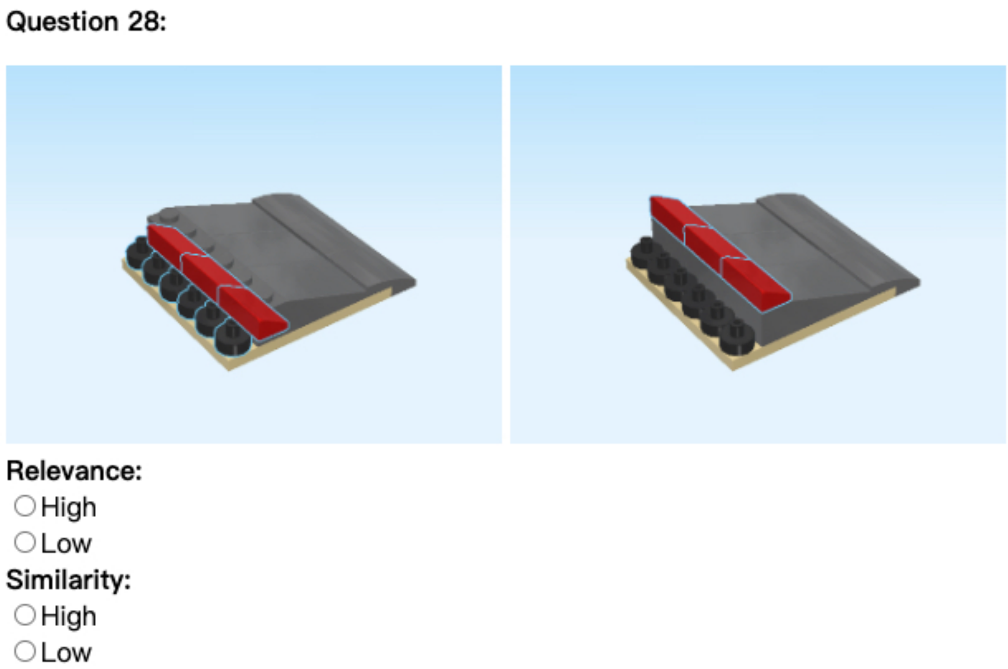


Figure 11.2: Examples question for Task3 in quality assurance.

Criteria	Description
C1: Relevance	Evaluate how relevant the evaluating image is to the reference image. Assess whether the content or context of the evaluating image pertains directly to that of the reference image.
C2: Similarity	Assess how similar the evaluating image is to the reference image. Consider the visual and thematic elements that align or correspond between the two images.
Total Questions	105
Time to Complete	15-35 minutes

Table 11.2: Evaluation criteria for Task 3.

11.2 In-context Examples

This section presents in-context examples from the dataset showcasing how the model handles various tasks such as scene understanding, object detection, and state detection. Each example consists of an image and a query-response pair, illustrating the model's application in practical scenarios.

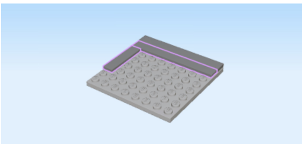
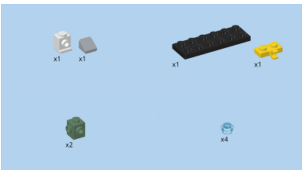
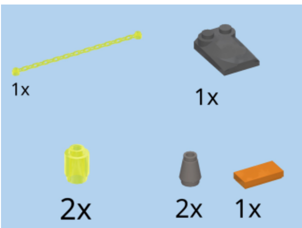
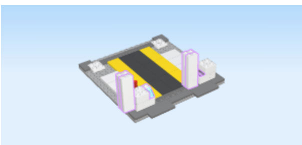
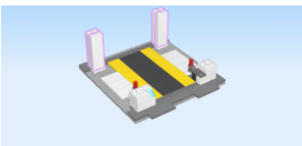
Image	Query and Generated Answer
	**Query:** [grounding] After I collect the following 2 pieces: 1 black plate 1x2 with the vertical clasp on the short side, and 1 bright orange smooth roof tile $1x2x\frac{2}{3}$. Tell me what to do. **Answer:** Put the plate 1x2 vertically on the table, clasp to the back. Put the roof tile 1x2x$\frac{2}{3}$ vertically on top, slope to the left.
	**Query:** [grounding] After I collect the following 2 pieces: 1 dark stone grey flat tile 1x8, and 1 dark stone grey flat tile 1x4. Tell me what to do for the current situation. **Answer:** Put the flat tile 1x8 horizontally in front of the previous piece. Put the flat tile 1x4 vertically in front of the previous piece on the leftmost column.
	**Query:** [object] Please identify and highlight the positions of the following 2 pieces: 1 black plate 2x6, and 1 bright yellow plate 1x2 with vertical clasp on long side. **Answer:** 1 black plate <57><15><82><35> and 1 bright yellow plate <82><25><93><36>
	**Query:** [object] Please identify and highlight the positions of 2 transparent fluorescent green round bricks 1x1. **Answer:** 2 transparent fluorescent green round bricks <20><61><30><78>
	**Query:** [state] After I find 4 white bricks 1x2x5. Did I do it correctly for they will form two pillars at the back. First, locate the two plates 2x2 in the left and right corners at the back. Put 2 horizontally on the plate 2x2 at the left back. Put the other 2 horizontally on the plate 2x2 at the right back. **Answer:** No.
	**Query:** [state] After I find 4 white bricks 1x2x5. Did I do it correctly for they will form two pillars at the back. First, locate the two plates 2x2 in the left and right corners at the back. Put 2 horizontally on the plate 2x2 at the left back. Put the other 2 horizontally on the plate 2x2 at the right back. **Answer:** Yes.

Table 11.3: In-context examples of model interactions with specific tasks within the dataset.

11.3 Snellius Scripts

In this section, we detail the configurations used for two critical experiments conducted on the Snellius computational platform using the Blip2 model. These experiments are differentiated by their specific goals: one for inference and the other for fine-tuning, each utilizing GPUs in configurations tailored to their computational needs.

11.3.1 Inference Configuration

The first script configures a single GPU job aimed at running inference tasks with the Blip2 model. This setup focuses on executing two separate inference runs with different parameters to test the model's performance under varied conditions.

```
#!/bin/bash
#SBATCH -job-name=blip2_inf_job
#SBATCH -nodes=1
#SBATCH -ntasks=1
#SBATCH -cpus-per-task=18
#SBATCH -gpus=1
#SBATCH -partition=gpu
#SBATCH -time=15:00:00
module load 2023
source .env/bin/activate
python inference_BLIP2.py 'Blip2' 'object' 'ablation_format'
```

Table 11.4: GPU job submission script for inference.

This script is an example of the Blip2 model's inference code, focusing on state and object detection tasks.

11.3.2 Fine-Tuning Configuration

The second script details the setup for a more resource-intensive job that utilizes two GPUs for fine-tuning the Blip2 model. This arrangement is intended to optimize the model's performance by adjusting weights based on extensive learning over new datasets.

```
#!/bin/bash
#SBATCH -job-name=blip_finetune
#SBATCH -nodes=1
#SBATCH -ntasks=2
#SBATCH -gpus=2
#SBATCH -cpus-per-task=18
#SBATCH -partition=gpu
#SBATCH -time=18:00:00
module load 2023
module load NCCL/2.18.3-GCCcore-12.3.0-CUDA-12.1.1
source .env/bin/activate
python blip2_ft_try.py
```

Table 11.5: Batch script for GPU job submission, formatted for clarity.

This script is an example of the Blip2 model's fine-tuning code, intended to improve the model's accuracy and efficiency in processing tasks.