

# OWLViz: An Open-World Benchmark for Visual Question Answering

Anonymous ACL submission

## Abstract

We present a challenging benchmark for the Open World VISual (OWLViz) question answering benchmark. OWLViz presents short queries that require integrating multiple capabilities, including common-sense knowledge, visual understanding, web exploration, and specialized tool usage. While humans achieve 69.2% accuracy on these intuitive tasks, even state-of-the-art VLMs struggle, with the best model, Gemini, achieving only 26.6% accuracy. Current tool-calling agents and GUI agents, which rely on limited vision and vision-language models as tools, perform even worse. This performance gap reveals significant limitations in multimodal systems’ ability to select appropriate tools and execute complex reasoning sequences, establishing new directions for advancing practical AI research.

## 1 Introduction

Large Vision Language Models (VLMs) have recently demonstrated impressive visual understanding and reasoning capabilities across numerous tasks (Liu et al., 2024b). Equipped with these advanced capabilities, VLMs are rapidly surpassing existing AI benchmarks (Chen et al., 2024). As the AI community pursues increasingly challenging problems, benchmark tasks are shifting from conventional fundamentals toward more human-centric challenges (Hu et al., 2024b; Roger et al., 2023; Brohan et al., 2023). These human-centric tasks typically demand broader world knowledge, situation awareness, and more complex reasoning.

While there are many benchmarks for VQA that have been created, existing efforts face several notable problems. Many early benchmarks only focus on visual understanding tasks such as entity detection and entity attribution (Ren et al., 2015; Malinowski and Fritz, 2014; Goyal et al., 2017). As a result, these benchmarks can only be used to evaluate the VLM’s grounding capability, while many

questions require additional capabilities such as reasoning. Recent benchmarks have shifted the focus towards complex questions where compositional reasoning and spatial reasoning are evaluated (Hudson and Manning, 2019; Gao et al., 2023; Bitton et al., 2023). Even though tool calling capability was investigated in a close-environment (Liu et al., 2023b; Li et al., 2023; Patil et al., 2024), these datasets only addressed a limited set of tools. As a result, these benchmarks fail to fully capture the complexity of real human queries that often require interpreting complex visual contexts, combining heterogeneous sources of knowledge, and reasoning over long, multi-step chains of actions (Mialon et al., 2023). As such, current evaluations do not sufficiently evaluate a model’s ability to act as an agent that can meaningfully invoke and coordinate tools to solve practical, open-ended questions grounded in the visual world.

To address this gap, we introduce OWLViz, a novel benchmark dataset specifically designed to evaluate vision-language models’ ability to utilize tools in complex, multimodal reasoning tasks. OWLViz necessitates three distinct skill sets that challenge both VLMs and agentic systems. **First**, the dataset incorporates visually degraded inputs featuring low brightness, poor contrast, or blur, scenarios where visual enhancement tools may be required to improve image quality for accurate processing. **Second**, tasks demand sophisticated reasoning capabilities to solve complex problems involving counting, projection, and measurement operations. **Third**, certain challenges require models to explore the internet and retrieve external data to answer questions based on minimal visual cues. Figure 1 illustrates representative examples of these challenging scenarios.

OWLViz introduces a benchmark for Agentic AI Assistant featuring 248 carefully annotated questions and answers where big proprietary VLM models failed to answer. OWLViz dataset is easy to



(a) **Question:** “How many people are visible on the left side of the white line that cuts across the photo? Provide a numeric answer.” **Answer:** 2  
**Skills:** using external API, human recognition. **Difficulty level:** 1



(b) **Question:** “How many umbrellas have 3 or more colors? Provide a numeric answer.” **Answer:** 2.  
**Skills:** object recognition, attribute identification, counting, object detection. **Difficulty level:** 2



(c) **Question:** “This is in Fairfax, Virginia. What is the name of the road shown in the photo?” **Answer:** Shadowridge Dr; Shadowridge drive; Shadowridge.  
**Skills:** OCR, knowledge search, knowledge retrieval, GUI, comparison, spatial relationships. **Difficulty level:** 3

Figure 1: **Examples of the three core challenges in our OWLViz dataset.** (a) Challenging visual conditions requiring image enhancement or specialized recognition tools to count people on a white line in a low-contrast night scene. (b) Complex reasoning tasks demanding object detection, attribute identification, and precise counting of multi-colored umbrellas in a dynamic street scene. (c) Knowledge-intensive queries requiring internet exploration and external data retrieval to identify specific locations based on minimal visual cues.

understand, challenging for both human and AI and featuring extensive tool-use skills. Yet, evaluation on this dataset is simple and can be done automatically.

Despite their success in many visual grounding tasks, even the best-performing VLMs demonstrated surprisingly poor performance on our benchmark. Most models achieved less than 20% accuracy in exact-match evaluation and below 30% in LLM-match evaluation, while college students easily completed these tasks with 69.2% accuracy in under one minute. OWLViz presents novel challenges and opportunities for advancing research in open-world visual understanding. We keep our dataset private to minimize data contamination. Additional information on how to access the data is available upon request.

## 2 Related work

**VQA Dataset.** Early visual question answering datasets focus on evaluating visual abilities such as visual entity and entity attributions (Ren et al., 2015; Malinowski and Fritz, 2014). Later works shift the focus towards object relation understanding (Antol et al., 2015), compositional reasoning on spatial relations (Johnson et al., 2017; Hudson and Manning, 2019). Recent studies expand the scope of the question into common-sense knowledge (Marino et al., 2019). However, these datasets are quite limited in coverage of real user queries.

**VLMs Dataset.** State-of-the-art VLMs have rapidly advanced in vision understanding (Hu et al., 2024a; Zhang et al., 2024; Zhou et al., 2024; Le et al., 2024), task comprehension (Jing et al., 2024), reasoning (Yue et al., 2024; Dang et al., 2024), and tool-use capabilities (Liu et al., 2025; Nguyen et al., 2024), enabling AI systems to tackle general human tasks (Kelly et al., 2024). These advances have led VLMs to surpass previously challenging AI benchmarks (Mialon et al., 2023; Marino et al., 2019; Hendrycks et al., 2020), prompting a shift in benchmarking toward real-world applications (Jimenez et al., 2024; Glazer et al., 2024) that incorporate human-centric considerations like ethical alignment and societal impact (Hu et al., 2024b).

**Agentic Dataset.** Many agentic datasets for tool calling are dedicated to some selected environments with limited tool space (Liu et al., 2023b; Li et al., 2023; Patil et al., 2024). However, these datasets risk the ability to evaluate general-purpose agents. Recent works have shifted towards open-domain, unlimited tools (Mialon et al., 2023). While GAIA is close to our work, its questions provide detailed plans to achieve the answer, hence only testing how well an agentic system is integrated with available tools rather than exploration and tool retrieval (Nguyen et al., 2024).

## 3 Dataset Creation

OWLViz is designed to evaluate models’ capabilities in comprehending image content and lever-

aging external tools to answer image-related questions. The dataset comprises 248 human-designed and annotated questions, each associated with an image and an unambiguous ground truth answer, allowing exact-match evaluation.

This section presents our methodology for constructing a dataset that more effectively evaluates Visual Question Answering (VQA) systems’ performance on open-world visual queries. The dataset development process encompasses five principal phases: (1) systematic image acquisition, (2) open-world question design, (3) establishment of reasoning paths to ensure reliable answer derivation, (4) comprehensive data annotation, and (5) implementation of standardized answer formatting protocols.

### 3.1 Image Acquisition

Images were collected from a range of publicly accessible sources. For questions requiring only visual interpretation, we intentionally selected dense and detailed images to ensure sufficient visual complexity. For questions designed to prompt external searches, images were chosen to provide minimal but sufficient cues such as brand names, logos, or partial text while avoiding overexposure of information that could reduce the challenge of the task.

The five most frequent image sources are *redfin.com*, *istockphoto.com*, *zillowstatic.com*, *rdcpix.com*, *shutterstock.com*, and *pexels.com*. Together with 17 screenshots, these sources account for approximately half of the dataset. The 17 author-generated images were created in cases where direct image URLs were unavailable, such as screenshots taken from PDF reports, social media posts, or frames from YouTube videos.

### 3.2 Questions Design

The dataset shares several design principles with GAIA, including (1) targeting questions that are conceptually simple yet practically useful and challenging for contemporary AI systems, (2) ensuring interpretability, (3) maintaining robustness against memorization, and (4) facilitating ease of evaluation. However, OWLViz distinguishes itself in several key aspects: (5) exclusive focus on images and (6) concise and practical.

First, every question in OWLViz is directly linked to a photograph. Questions can be answered using (i) image content alone, (ii) a combination of image content and metadata embedded within the image, or (iii) an integration of image content, metadata, and external knowledge inferred from

visual cues. Unlike other datasets where questions can be excessively lengthy and unlikely to reflect the way people naturally inquire about information, our dataset prioritizes practicality.

Second, OWLViz is designed to capture questions that closely mirror the types of inquiries people naturally make when interpreting images in everyday contexts. These include tasks such as counting objects, identifying locations or addresses, checking property costs, extracting key details, or interpreting relationships between elements in a scene. This emphasis on practical, context-driven questions ensures the dataset’s relevance to real-world applications and everyday problem-solving scenarios.

We develop an initial set of skills that was incorporated into the annotation web interface; the skill list includes three main categories: visual skills, reasoning skills, and tool skills (See Table 1). During the annotation, we allow annotators to add new skills if needed, which results in some less frequent skills as shown in Figure 4.

### 3.3 Data Annotation

All questions in OWLViz were designed and annotated by the authors. Each question was carefully crafted and reviewed to ensure clarity, relevance, and alignment with a corresponding image. An exact-match answer was provided for each question, establishing a clear ground truth.

To ensure the quality, solvability, and objectivity of the dataset, the annotation process was divided into three distinct phases. In the first phase, one author was responsible for collecting the images and constructing the initial set of questions. In the second phase, five other authors independently reviewed the questions and provided feedback to refine and clarify them where necessary. In the third phase, to promote annotation consistency and minimize bias, an internal tool was developed to randomly assign each question to at least two reviewers. Reviewers answered the questions without any additional context beyond the image and the question text.

Any question that could not be reliably answered by the reviewers without additional input from the original author was removed from the dataset. This three-phase process ensured that the final dataset includes only questions that are independently answerable and clearly grounded in the accompanying visual content.



	Skill	Description
Visual	human recognition	Identify humans and their locations in the image
	object recognition	Identify non-human objects and their location in the image
	identify 2 endpoints	Identify two specific endpoints of an object in an image
	object detection	Identifies what objects are present and where they are located by using a bounding box
	object segmentation	Provides pixel-level understanding of what and where all objects are
	spatial relationships	Understands 3D spatial relations between objects in the image.
	attribute identification	Specifies attributes of objects (e.g., color, shape, size)
Reasoning	object measuring	Measures object dimensions or distances in the image
	arithmetic calculation	Comprises operations such as Addition, Subtraction, Multiplication and Division
	counting	Handles counting queries such as 'how many...'
	comparison	Handles comparison queries
	logical operations	Handles logical operations such as AND (intersection) and OR (union) on sets
Tools	QR code scanning	Scans and decodes QR codes from the image
	OCR	Optical Character Recognition — extracts text and their locations in the image
	GUI	Handles graphical user interface-based tasks
	knowledge search	Searches for external knowledge beyond the image itself
	using external API	Uses an external API such as image enhancement and reading metadata.
	knowledge retrieval	Retrieve the necessary information in the metadata of images

Table 1: Taxonomy of skills required for our dataset, categorized into visual, reasoning, and tool-based capabilities. Each skill is accompanied by a functional description indicating the specific capability it represents in the context of visual question answering tasks.

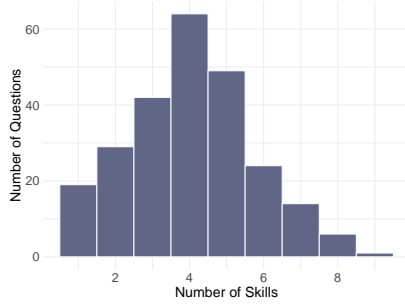


Figure 2: Number of unique skills used per question.

### 3.4 Answer Format Standardization

We explicitly specify the format of the expected answer for each question. This includes defining whether the answer should be a *yes/no response*, a *multiple-choice selection*, or a *short exact-match answer*. For some questions, multiple answers are allowed. Acceptable answers are separated by semicolons. The exact output format instruction is provided in Appendix C.1. By standardizing the answer format, we facilitate straightforward evaluation, enabling seamless testing of both exact-match answers and multiple-choice responses. This design ensures that the dataset is both practical and robust, catering to diverse testing and benchmarking needs while maintaining ease of implementation.

It is important to note that, in order to conform to an exact-match evaluation format, questions were

transformed into constrained response types such as multiple choice, yes/no, single numerical values, or short text answers limited to a few words. While this approach facilitates consistent evaluation and benchmarking across models, it may also increase the likelihood of correct responses, as it narrows the range of possible outputs.

Preliminary experiments using free-form, open-ended answers reveal significantly higher failure rates, suggesting that exact-match formats may overestimate model performance by simplifying the response space. This is a trade-off between evaluation consistency and the complexity of real-world language understanding.

### 3.5 Difficulty Level

Following (Mialon et al., 2023), we categorize questions into three levels of increasing difficulty based on the size of unique skills needed to answer the question. Figure 2 shows the a number of unique skills used per question.

We broadly define difficulty levels as follows:

- Level 1: Typically involves no more than 2 unique skills and at most 1 external tool.
- Level 2: Involves a greater number of skills - generally between 3 and 5 skills, typically includes the combination of two tools.

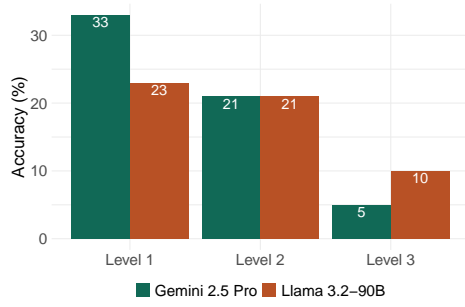


Figure 3: **Model performance degradation across difficulty levels.** Results show both Gemini 2.5 Pro and Llama 3.2-70B exhibit declining accuracy as task difficulty increases, illustrating current limitations in complex visual reasoning.

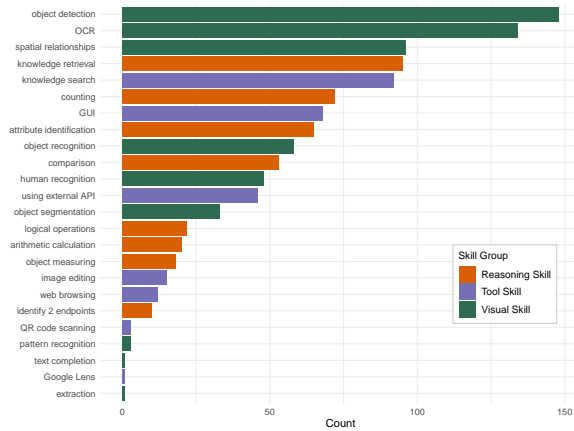


Figure 4: Distribution of skills annotated by human. Noted that annotators are allowed to add additional skills or tools that they used (e.g. Google Lens)

- Level 3: Designed for an ideal general-purpose assistant, these questions may require arbitrarily long sequences of actions, unrestricted use of tools, and general access to the whole Internet.

Figure 3 shows the performance of Gemini Pro and Llama 3.2 90B grouped by difficulty level. Overall, these VLMs’ performance deteriorates, highlighting the persistent gap between current AI capabilities and human-like reasoning.

We also displays the distribution of skills needed that were generated by annotators in Figure 4 . Among visual skills, object detection, OCR, and spatial reasoning are most common. In reasoning skills, counting and knowledge retrieval dominate. For tool-use skills, knowledge search and GUI interaction are most frequent.

## 4 Experiments

In this session, we assess the capabilities of three powerful methodological approaches on OWLViz: Vanilla VLMs, Tool-Calling Agents and GUI Agents. This comparative analysis allows us to identify and examine the key challenges presented by our dataset.

### 4.1 Evaluation Metrics

We employ two metrics to evaluate model performance: *Exact Match (EM)* and *LLM-based Match (LM)* (Zheng et al., 2023). The EM metric requires the model’s output to be identical to the ground truth answer, allowing for minor variations in capitalization and whitespace. In practice, some models still fail to follow output format instructions, leading to false negative errors. To allow for more flexible evaluation, we use LM, where GPT-4o acts as a judge to determine semantic equivalence between the predicted and ground truth answers (See Appendix C.3).

### 4.2 Vanilla VLMs

**Models.** We evaluate a wide set of open-source VLM models: DeepSeek-VL (Wu et al., 2024), Qwen-VL (Yang et al., 2024), InternVL (Zhang and Zhou, 2022), LLaVa (Liu et al., 2023a), and Molmo (Deitke et al., 2024) as well as proprietary models, i.e., Anthropic’s Claude, OpenAI’s GPT4V/o, and Google’s Gemini.

**Results.** Table 2 presents our evaluation of VLMs across three categories: small open-source, large open-source, and proprietary models. The results demonstrate that OWLViz poses significant challenges even for state-of-the-art VLMs. The best-performing models, Gemini-2.5-pro-preview and Gemini-2.0-flash, achieve only 21.51% EM accuracy, indicating substantial room for improvement on this benchmark. This relatively low performance across all models suggests that our dataset effectively probes the limitations of current VLMs in handling complex visual reasoning tasks when not equipped with tools.

Within the open-source category, larger models generally perform better than their smaller counterparts, but the improvement is modest. For instance, Llama-3.2-11B-Vision-Instruct tops the small model leaderboard at 25.10% LM, its larger variant Llama-3.2-90B-Vision-Instruct yields a lower LM score (24.70%). This suggests that simply scaling up model size may not be sufficient

	Model	EM	LM
	<b>Human</b>	<b>69.21</b>	
Small Open Source	DeepSeek-VL2-small (2.8B)	11.16	12.75
	DeepSeek-VL2 (4.5B active)	11.16	14.34
	Qwen2-VL-7B-Instruct	12.75	17.93
	Qwen2.5-VL-7B-Instruct	13.94	19.52
	InternVL3-8B	14.34	<u>21.12</u>
	LLaVa-v1.6-mistral-7B	14.74	15.54
	Llama-3.2-11B-Vision-Instruct	14.74	<b>25.10</b>
	InternVL2.5-8B	14.74	18.73
	LLaVa-v1.5-13B	16.33	16.33
	Molmo-7B-D-0924	<u>17.13</u>	20.32
	LLaVa-v1.5-7B	<b>18.33</b>	19.92
Large Open Source	Qwen2.5-VL-32B-Instruct	2.79	<u>25.90</u>
	InternVL2.5-38B	13.94	19.52
	InternVL3-78B	15.54	20.72
	Molmo-72B-0924	15.94	22.71
	InternVL2.5-78B	15.94	21.91
	InternVL3-38B	16.73	23.11
	Qwen2-VL-72B-Instruct	19.92	25.90
	Qwen2.5-VL-72B-Instruct	<u>20.32</u>	<b>26.29</b>
	Llama-3.2-90B-Vision-Instruct	<b>20.72</b>	24.70
Proprietary	Claude-3-5-sonnet-20241022	11.55	19.92
	GPT-4V	14.34	20.00
	Gemini-2.5-Flash	15.54	<u>25.50</u>
	GPT-4o (2024-11-20)	16.33	19.52
	Gemini-1.5-Pro	<u>19.52</u>	21.91
	Gemini-2.0-Flash	<b>21.51</b>	24.30
	Gemini-2.5-Pro	<b>21.51</b>	<b>27.09</b>

Table 2: Model performance breakdown into 3 groups ordered by EM performance. The best and second-best of each group are **bolded** and underlined, respectively.

to address the challenging nature of our benchmark. Among proprietary models, while Gemini-2.0-Flash sets the current state-of-the-art, its performance (21.51% EM, 27.09% LM) still falls significantly short of human-level understanding, highlighting the substantial gap between current AI capabilities and human visual reasoning abilities. Notably, the consistent gap between EM and LM scores across all models indicates that even when models grasp the correct concept, they often struggle to express it in the exact required format.

### 4.3 Tool-Calling Agents

**Models.** Tool-Calling Agents extend traditional VLMs by integrating external tools and action capabilities to solve complex visual reasoning tasks. Our evaluation examines six representative systems: LLaVa-Plus (Liu et al., 2025), which enhances LLaVA (Liu et al., 2023a) with pre-trained

Model	MLLM	EM	LM
LLaVa-Plus	gpt-4o-2024-11-20	0.00	2.50
ViperGPT	gpt-4o-2024-11-20	7.56	12.35
GPT4Tools	vicuna-7b-v1.5	11.15	14.34
HYDRA	gpt-4o-2024-11-20	10.75	12.35
HF Agent	gpt-4o-2024-11-20	<b>18.32</b>	<u>24.08</u>
DynaSaur	gpt-4o-2024-11-20	<u>16.23</u>	<b>26.67</b>

Table 3: Performance of Agentic models with tool-uses. The best and second-best of each group are **bolded** and underlined, respectively.

vision tools for improved reasoning; ViperGPT (Surís et al., 2023), which uses GPT-4o to generate Python code orchestrating multiple vision models; GPT4Tools (Yang et al., 2023), built on Vicuna with instruction tuning for visual tool control; HYDRA (Jalaian et al., 2025), employing deep reinforcement learning to fine-tune LLMs for dynamic visual reasoning; HF Agent includes predefined tools for visual tasks and web browsing; and DynaSaur (Nguyen et al., 2024), supporting real-time generation and composition of actions with capabilities for continual learning through action storage and reuse.

**Results.** Table 3 presents the performance comparison of various large language models (LLM) agents equipped with tool-use capability. Among the models, HF Agent achieves the highest EM score (18.32%), indicating superior accuracy in exact task execution, followed closely by DynaSaur (16.23%). DynaSaur leads in LM score (26.67%), suggesting strong overall language understanding and generation capabilities. In contrast, LLaVa-Plus performs poorly across both metrics, particularly with an EM score of 0.00, highlighting limitations in task precision.

**Tool Exploration Incentives.** Our analysis reveals that the original tool-calling agent baselines demonstrate insufficient motivation to utilize external models or tools for visual question-answering tasks, instead defaulting to generating code comments for reasoning before directly producing answers. To mitigate this limitation, we implemented an explicit instruction requiring agents to employ at least one external tool for answer verification prior to submission. We subsequently conducted a comprehensive analysis of all libraries and machine learning models generated by DynaSaur across 100 sample instances. Figure 5 illustrates the compara-

Model	EM	LM	Click	Hover	Scroll
UI-TARS	0.00	12.31	0.91	0.51	0.68
ShowUI	0.00	12.80	0.97	0.19	0.10

Table 4: The performance in EM and LM of GUI Agents on OWLViz with the average number of mouse actions.

tive frequency of tool utilization between models with and without these incentives. This strategic modification yielded a 2% improvement in Exact Match (EM) score. For detailed results, readers may refer to Table 5 in the appendix.

In the no-incentive scenario (Figure 5), tool usage is more concentrated and limited in diversity, with image (118), tags (38), and pytesseract (34) being the most frequently used tools. Most other tools appear only a handful of times, reflecting a poor coverage of the tools being used. In contrast, models with incentives show both a significant increase in tool usage and a broader variety of tools. pytesseract leads with 262 mentions, followed by image (218), and YOLO (120). Additionally, a wide array of tools such as cv2, torch, transforms, and various deep learning components (e.g., fasterrcnn\_resnet50\_fpn, cnn) appear in the incentivized setting. This indicates that the presence of incentives not only increases effort but also encourages a broader adoption of more sophisticated, task-specific tools.

#### 4.4 GUI Agents

**Models.** Our evaluation includes two GUI interaction baselines combining GPT-4o (2024-11-20) with specialized action determination components. UI-TARS (Qin et al., 2025) pairs GPT-4o’s reasoning capabilities with UI-TARS-2B-SFT for action determination in graphical interface tasks, enabling structured human-computer interactions. Similarly, ShowUI (Lin et al., 2024) employs the same GPT-4o model for reasoning but integrates ShowUI-2B for GUI action determination, facilitating efficient multimodal instruction-following in computer interfaces.

**Results.** Table 4 presents the performance of two GUI agents—UI-TARS and ShowUI—evaluated in task accuracy and the average number of mouse actions (Click, Hover, Scroll). Both agents fail to achieve any correct task as indicated by their EM scores of 0.00, highlighting a complete lack of task-following capabilities in output format. While LM scores are modest, with ShowUI slightly better

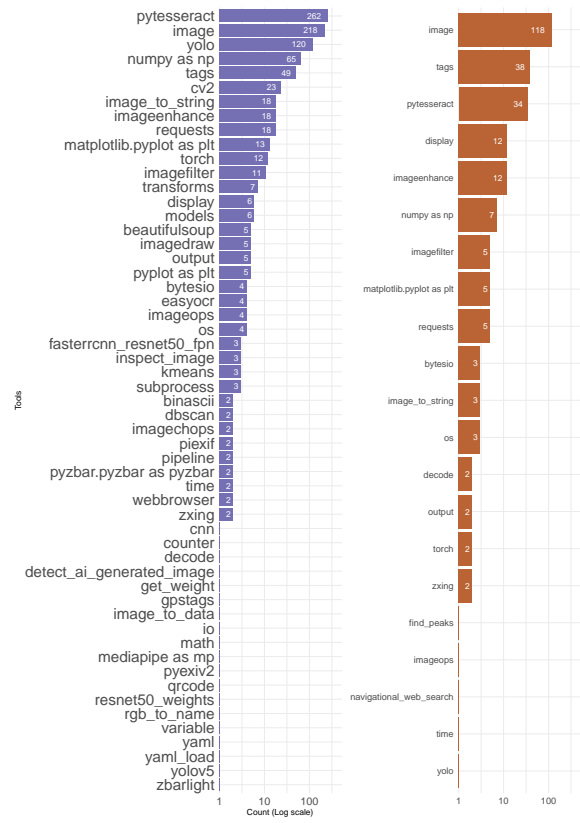


Figure 5: Comparison of libraries and ML models used by DynaSaur with (left) and without incentive (right).

than UI-TARS (12.80% vs. 12.31%).

Notably, the total number of actions performed by these agents is extremely low, reflecting minimal engagement with the interface. UI-TARS averaged fewer than 1 click (0.91), hover (0.51), and scroll (0.68) actions per task, while ShowUI executed slightly more clicks (0.97) but significantly fewer hovers (0.19) and scrolls (0.10). These numbers indicate that these GUI agents barely interact with the user interface. This somehow explains the low performance of the GUI agents, even when the model is allowed to freely explore.

#### 4.5 Comparison between orchestrations

In our evaluation of model performance across vision-language models (VLMs), agent-based systems, and GUI agents, we observe distinct capability profiles across these paradigms. Vision-language models, particularly proprietary systems such as Gemini-2.5-Pro, demonstrate the strongest overall performance, achieving the highest exact match (EM) and LLM match (LM) scores. Agent-based systems that incorporate tool use, such as HF Agent and DynaSaur powered by GPT-4o, show a marked improvement in EM over their base models





Figure 6: **Qualitative results comparing different model capabilities on OWLViz.** Results demonstrate varying capabilities across model types: Gemini (vanilla VLM) fails to identify the target, DynaSaur (tool-calling agent) produces an incorrect answer despite external search capabilities, and ShowUI (GUI agent) provides no answer.

(e.g., HF Agent achieves 18.32 EM vs. 16.33 for standalone GPT-4o), suggesting that tool augmentation enables more effective task execution and grounded reasoning. However, GUI agents such as UI-TARS and ShowUI exhibit significant limitations, with EM scores remaining at 0.00 and LM scores peaking at only 12.80%. Their low interaction metrics across click, hover, and scroll actions further highlight their current inability to perform even basic interface-driven tasks reliably.

## 4.6 Qualitative Example

We present a qualitative example that demonstrates the varying capabilities of different model types (see Figure 6). **Gemini**, representing vanilla VLMs, attempts to locate the "For Sale" lot but fails to identify the street name and the requested business, resulting in a conclusion "Not identifiable". **DynaSaur**, the tool-calling agent, leverages OCR and Search to identify the correct lot address ("821-825 E 13th Ave, Eugene, OR") and employs external search tools to find businesses in the vicinity. Through progressive refinement of search queries and cross-referencing, DynaSaur identifies "Starbucks Coffee" as the shop across from the lot. However, this is an incorrect answer, highlighting challenges in accurate spatial reasoning and

external knowledge integration (e.g., maps with street view). **ShowUI**, representing GUI agents, demonstrates the limitations of restricted action capabilities. With only two available actions, the model attempts to scroll but ultimately fails to produce an answer. This illustrates how constraints on interaction modalities can significantly impact performance on complex visual reasoning tasks. Please refer to Section D in the supplementary materials for detailed output.

## 5 Conclusion

We introduced OWLViz, a challenging benchmark for evaluating AI models’ visual understanding, reasoning, and tool use. By integrating real-world tasks requiring image comprehension, metadata extraction, web exploration, and external tool use, OWLViz highlights the limitations of current VLMs, tool-use agents, and GUI agents, which fall short of human performance. Our results show that AI struggles with multi-step reasoning and practical tool integration, underscoring the need for further advancements.



## Limitation

This initial dataset has a limited sample size. Future work will expand it to include a variety of image quality, content diversity, and a broader range of daily life scenarios encountered in photographs.

## Ethical Consideration

All the data in this project were annotated by the authors. We did not hire any crowd-sourced workers. The images in this dataset were sourced from publicly accessible platforms, with no copyright claimed. The dataset excludes personally identifiable or sensitive information.

## References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Yonatan Bitton, Hritik Bansal, Jack Hessel, Rulin Shao, Wanrong Zhu, Anas Awadalla, Josh Gardner, Rohan Taori, and Ludwig Schmidt. 2023. Visit-bench: A benchmark for vision-language instruction following inspired by real-world use. *arXiv preprint arXiv:2308.06595*.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. 2024. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*.

Long Hoang Dang, Thao Minh Le, Vuong Le, Tu Minh Phuong, and Truyen Tran. 2024. Sadl: An effective in-context learning method for compositional visual qa. *CoRR*.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. 2024. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Difei Gao, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2023. Cric: A vqa dataset for compositional reasoning on vision and commonsense. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5561–5578.

Elliot Glazer, Ege Erdil, Tamay Besiroglu, Diego Chicharro, Evan Chen, Alex Gunning, Caroline Falkman Olsson, Jean-Stanislas Denain, Anson Ho, Emily de Oliveira Santos, Olli Järvinen, Matthew Barnett, Robert Sandler, Matej Vrzala, Jaime Sevilla, Qiuyu Ren, Elizabeth Pratt, Lionel Levine, Grant Barkley, Natalie Stewart, Bogdan Grechuk, Tetiana Grechuk, Shreepranav Varma Enugandla, and Mark Wildon. 2024. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in AI. *CoRR*, abs/2411.04872.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Yushi Hu, Otilia Stretcu, Chun-Ta Lu, Krishnamurthy Viswanathan, Kenji Hata, Enming Luo, Ranjay Krishna, and Ariel Fuxman. 2024a. Visual program distillation: Distilling tools and programmatic reasoning into vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 9590–9601. IEEE.

Zhe Hu, Yixiao Ren, Jing Li, and Yu Yin. 2024b. VIVA: A benchmark for vision-grounded decision-making with human values. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2294–2311, Miami, Florida, USA. Association for Computational Linguistics.

Drew A. Hudson and Christopher D. Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Brian Jalaian, Nathaniel D Bastian, et al. 2025. Hydra: An agentic reasoning approach for enhancing adversarial robustness and mitigating hallucinations in vision-language models. *arXiv preprint arXiv:2504.14395*.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

- Liqliang Jing, Ruosen Li, Yunmo Chen, and Xinya Du. 2024. [FaithScore: Fine-grained evaluations of hallucinations in large vision-language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5042–5063, Miami, Florida, USA. Association for Computational Linguistics.
- Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910.
- Chris Kelly, Luhui Hu, Bang Yang, Yu Tian, Deshun Yang, Cindy Yang, Zaoshan Huang, Zihao Li, Jiayin Hu, and Yuxian Zou. 2024. [Visiongpt: Vision-language understanding agent using generalized multimodal framework](#). *CoRR*, abs/2403.09027.
- Quang-Hung Le, Long Hoang Dang, Ngan Le, Truyen Tran, and Thao Minh Le. 2024. Progressive multi-granular alignments for grounded reasoning in large vision-language models. *arXiv preprint arXiv:2412.08125*.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. [API-bank: A comprehensive benchmark for tool-augmented LLMs](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.
- Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Weixian Lei, Lijuan Wang, and Mike Zheng Shou. 2024. [Showui: One vision-language-action model for gui visual agent](#). *Preprint*, arXiv:2411.17465.
- Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024a. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning.
- Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024b. Visual instruction tuning. *Advances in neural information processing systems*, 36.
- Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, et al. 2025. Llava-plus: Learning to use tools for creating multimodal agents. In *European Conference on Computer Vision*, pages 126–142. Springer.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanxu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023b. Agent-bench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.
- Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. *Advances in neural information processing systems*, 27.
- Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. 2019. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204.
- Grégoire Mialon, Clémentine Fourier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. *arXiv preprint arXiv:2311.12983*.
- Dang Nguyen, Viet Dac Lai, Seunghyun Yoon, Ryan A Rossi, Handong Zhao, Ruiyi Zhang, Puneet Mathur, Nedim Lipka, Yu Wang, Trung Bui, et al. 2024. [Dynasaur: Large language agents beyond predefined actions](#). *arXiv preprint arXiv:2411.01747*.
- Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.
- Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*.
- Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Exploring models and data for image question answering. *Advances in neural information processing systems*, 28.
- Alexis Roger, Esma Aïmeur, and Irina Rish. 2023. Towards ethical multimodal systems. *arXiv preprint arXiv:2304.13765*.
- Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11888–11898.
- Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. 2024. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*.
- Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. 2024. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

- Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. 2023. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36:71995–72007.
- Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhui Chen. 2024. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9556–9567.
- Huaxiang Zhang, Yaojia Mu, Guo-Niu Zhu, and Zhongxue Gan. 2024. [Insightsee: Advancing multi-agent vision-language models for enhanced visual understanding](#). *CoRR*, abs/2405.20795.
- Linhai Zhang and Deyu Zhou. 2022. [Temporal knowledge graph completion with approximated Gaussian process embedding](#). In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4697–4706, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhaghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.
- Kaiwen Zhou, Kwongjoon Lee, Teruhisa Misu, and Xin Wang. 2024. [Vicor: Bridging visual understanding and commonsense reasoning with large language models](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 10783–10795. Association for Computational Linguistics.
- Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, et al. 2025. Internv13: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*.



Model	EM
DynaSaur (w/o incentive)	9.0
DynaSaur (w/ incentive)	11.0

Table 5: Performance of LLM with tool-use

## A Data Annotation Discussion

**Diversity Consideration** For questions involving counting people, we deliberately included a wide range of settings, cultures, and geographic regions to ensure broad representation. For questions involving object detection, our focus was on identifying objects that are commonly encountered in everyday life, as opposed to rare or obscure objects that most people are unlikely to encounter. The goal was to create questions that align with practical use cases and test AI systems on their ability to recognize and reason about objects that play a significant role in daily human activities.

**Multiple-Choice Consideration** One major concern in using multiple-choice answers is the potential difficulty in assessing contamination, as a model may arrive at the correct choice despite following an incorrect reasoning path. This raises challenges in evaluating whether the model truly understands the question or has simply arrived at the answer through spurious correlations.

Upon analyzing the model results, we found that this phenomenon is rare. While multiple-choice formats may introduce some ambiguity in reasoning validation, the majority of cases reflect genuine model comprehension rather than accidental correctness. Further investigation into model reasoning patterns can provide deeper insight into this effect and its implications for benchmarking AI systems.

## B Experiment Details

All the experiments with open-source models were done on a single P4de node with 8 A100-80GB GPUs. We use the original non-quantized checkpoint using bfloat16 and temperature = 0.

We use GPT 4o (gpt-4o-2024-11-20) in the LLM Match, the HF-Agents, and DynaSaur models.

In this work, we use the following model cards from HuggingFace for our evaluation:

- deepseek-ai/deepseek-vl2 (Wu et al., 2024)
- deepseek-ai/deepseek-vl2-small

- liuhaotian/llava-v1.6-mistral-7b (Liu et al., 2024a) 828
- liuhaotian/llava-v1.5-13b 829
- Claude-3-5-Sonnet (claude-3-5-sonnet-20241022) 830
- GPT-4V (2023-12-01-preview) 831
- GPT-4o (gpt-4o-2024-11-20) 832
- Qwen/Qwen2-VL-7B-Instruct (Wang et al., 2024) 833
- Qwen/Qwen2-VL-72B-Instruct 834
- Qwen/Qwen2.5-VL-7B-Instruct (Yang et al., 2024) 835
- Qwen/Qwen2.5-VL-72B-Instruct 836
- Gemini-1.5-pro 837
- Gemini-2.0-flash 838
- Gemini-2.5-flash (gemini-2.5-flash-preview-04-17) 839
- Gemini-2.5-pro (gemini-2.5-pro-preview-03-25) 840
- OpenGVLab/InternVL2.5-8B (Chen et al., 2024) 841
- OpenGVLab/InternVL2.5-38B 842
- OpenGVLab/InternVL2.5-78B 843
- OpenGVLab/InternVL3-8B (Zhu et al., 2025) 844
- OpenGVLab/InternVL3-38B 845
- OpenGVLab/InternVL3-78B 846
- meta-llama/Llama-3.2-11B-Vision-Instruct (Dubey et al., 2024) 847
- meta-llama/Llama-3.2-90B-Vision-Instruct 849
- allenai/Molmo-7B-D-0924 (Deitke et al., 2024) 850
- allenai/Molmo-72B-0924 851

## C Prompts

### C.1 Answer Format Specifications

#### Prompt C.1: Answer Format Specification

##### # Answer format

Your final answer should be a number OR as few words as possible OR a comma-separated list of numbers and/or strings.

- If you are asked for a number, don't use commas to write your number nor use units such as \$ or percent sign unless specified otherwise.

- If you are asked for a string, don't use articles, or abbreviations (e.g. for cities), and write the digits in plain text unless specified otherwise.

- If you are asked for a comma-separated list, apply the above rules depending on whether the element to be put in the list is a number or a string.

C.2 LLM Prompt

Prompt C.2: System Prompt

# Answer format

You are a Visual Language Model capable of performing Visual Question Answering tasks.  
You are provided with:  
An image as input.  
A question about the image.

Your task is to:

Understand the input: Briefly explain what is visible in the image.  
Analyze the question: Break down the question into components (e.g., type of query, objects or attributes referenced).  
Identify reasoning path: Describe the step-by-step reasoning used to answer the question.  
Tools or techniques used: Explain which tools (e.g., object detection, attribute recognition, spatial reasoning) are involved in each step.

Final answer: Provide a clear and concise answer to the question.

When describing the reasoning path, include:

Visual features extracted from the image (e.g., objects, relationships, or regions of interest).  
Logical steps that are taken to interpret the image in the context of the question.  
Connections between extracted features and the question components.

Example:  
Image: [Attach an image here]  
Question: "What is the color of the car parked next to the tree?"

Expected Response:  
Image Description: The image shows a parking lot with several cars and trees. A red car is parked next to a tree on the left side of the image.  
Question Analysis: The question asks for the color of a specific car relative to its position near a tree.

Reasoning Path:  
Step 1: Detect objects in the image (cars, trees).  
Step 2: Identify spatial relationships (car parked next to the tree).  
Step 3: Extract the attribute (color) of the relevant car.

Tools or Techniques Used:  
Object detection to locate cars and trees.  
Spatial reasoning to determine proximity.  
Attribute recognition to identify the color of the car.  
Final Answer: The car parked next to the tree is red.

Question: {{question}}  
Your response:

C.3 LLM Match Prompt

Prompt C.3: LLM Match Prompt

**Task:** Compare the two answers to determine whether they are semantically equivalent. If Answer 1 contains multiple valid options separated by semicolons (";"), consider Answer 2 semantically the same if it matches the meaning of any one of the options in Answer 1.

**Instructions:**  
- If the meanings are equivalent or convey the same idea, return True.  
- If the meanings are different, unrelated, or contain conflicting information, return False.  
- Return only "True" or "False" as output. Do not explain.

**Input:**  
- **Question:** question  
- **Answer 1:** answer\_1  
- **Answer 2:** answer\_2  
- **Output format:** Return only True or False.

**Example 1:**  
- **Question:** "How many parking slots are available in this photo? Choose one option: a. at least 24 b. no more than 20"  
- **Answer 1:** "b"  
- **Answer 2:** "no more than 20"  
- **Output:** True

**Example 2:**  
- **Question:** "How many people are wearing plain white shirts in this photo? Choose one option: a. at least 15, b. 12, c. 13, d 14"  
- **Answer 1:** "a"  
- **Answer 2:** "12"  
- **Output:** False

D Output examples

The following tables show 3 failure cases of Gemini, Molmo, HF Agent, and Dynasaur. Here are some observations:

- Even though Gemini and Molmo were instructed to perform chain-of-thought, the image description does not help provide information to answer the question.
- HF Agent, as designed to use a set of predefined tools, never creates new tools.
- DynaSaur does not generate new tools without incentives. DynaSaur with incentive to generate new tools seems to only explore basic VQA and OCR tools.

**Question:** What is the number that appears in front of the face of the person on the poster on the second floor? Provide a numeric answer.



**Correct answer:** 12

**Gemini:**

1. *Image Description:* The image shows a vibrant, neon-lit street scene, possibly in an Asian city. Buildings are adorned with colorful signs and advertisements, and people are walking along the street below. The image appears to be taken at night.
2. *Question Analysis:* The question asks for a specific number displayed on a poster located on the second floor of a building, positioned near a person's face.
3. *Reasoning Path:*
  - Step 1: Locate the second floor of the building in the image. This is identifiable by the distinct floor levels and signage.
  - Step 2: Search for posters or advertisements on the second floor.
  - Step 3: Identify the poster that features a person's face. There is a face partially obscured by some railings on the second floor right.
  - Step 4: Locate the number appearing in front of the person's face on this poster. The number 779 appears above and to the left of the partially visible face.
4. *Tools or Techniques Used:*
  - Visual inspection to identify the building floors and signage.
  - Object recognition to locate posters and faces.
  - Spatial reasoning to relate the number to the person's face on the poster.
5. *Final Answer:* 779

**Molmo:**

1. *Image Description:* The image shows a bustling city street at night with numerous neon signs and advertisements. On the second floor of a building, there's a poster featuring a person's face with a number displayed in front of it.
2. *Question Analysis:* The question asks for a specific numeric value visible on a poster on the second floor.
3. *Reasoning Path:*
  - Step 1: Identify the second floor of the building.
  - Step 2: Locate the poster with a person's face.
  - Step 3: Detect any numbers displayed on or near the poster.
  - Step 4: Extract the numeric value from the detected number.
4. *Tools or Techniques Used:*
  - Object detection to find the building and poster.
  - Facial recognition to confirm the presence of a person's face.
  - Optical character recognition (OCR) to identify the number.
5. *Final Answer:* 1

Table 6: Output example 1: Gemini and Molmo



HF Agent:
<p><b>Assistant:</b>  <i>Thought: I will start by using the ‘visualizer’ tool to analyze the image and identify the location of the second floor by looking for visual cues such as the Bravo sign in green lettering and advertisements.</i>  <i>Code:</i></p> <pre>question="Where is the second floor located in the image, and does it contain the Bravo sign in green lettering?" second_floor_location = visualizer(question=question, image_path="data/viza/validation/city-night.jpg") print(second_floor_location)</pre> <p><b>Environment:</b> <i>The second floor is located above the ground level, where you can see the neon lights and the railing on the right side of the image. The green "Bravo" sign is positioned on the first floor, underneath the second floor, as it appears just above the entrance where people are walking in.</i></p> <p><b>Assistant:</b>  <i>Thought: Now that I know the location of the second floor, I will use the ‘visualizer’ tool to scan for posters on the second floor that feature a person. I will look for any numbers that appear in front of the person’s face on these posters.</i>  <i>Code:</i></p> <pre>question="Are there any posters on the second floor featuring a person? If so, is there a number in front of the person’s face?" poster_info = visualizer(question=question, image_path="data/viza/validation/city-night.jpg") print(poster_info)</pre> <p><b>Environment:</b> <i>Yes, there is a poster on the second floor featuring a person. There is no number in front of the person’s face.</i></p> <p><b>Assistant:</b>  <i>Thought: I will re-examine the second floor of the image using the ‘visualizer’ tool to ensure no other posters or visual elements with numbers are overlooked. I will broaden the search to include any numbers on the second floor that might be relevant to the task.</i>  <i>Code:</i></p> <pre>question="Are there any numbers on the second floor, either on posters, advertisements, or other visual elements?" second_floor_details = visualizer(question=question, image_path="data/viza/validation/city-night.jpg") print(second_floor_details)</pre> <p><b>Environment:</b> <i>There are no visible numbers on the second floor in the image provided; it primarily features colorful neon signs with various designs and text but no numerical elements are discernible.</i></p> <p><b>Assistant:</b>  <i>Thought: Given that there are no visible numbers on the second floor, it seems the task may have been based on a misunderstanding or incorrect assumption about the image content. However, it’s important to provide an answer based on the information we have gathered. Since there are no numbers in front of the person’s face on the second floor, nor any other numbers visible on the second floor, I will conclude that there is no number to report.</i>  <i>Code:</i></p> <pre>final_answer("There is no number in front of the face of the person on the poster on the second floor in the image provided.")</pre> <p><b>Environment:</b></p>

Table 7: Output example 1 (cont.)

DynaSaur (w/o incentive):
<b>Assistant:</b> <i>Thought: I'll use the 'inspect_image' tool to analyze the image.</i> <i>Code:</i>
<code>inspect_image('data/viz_inspector/validation/images/city-night.jpg')</code>
<b>Environment:</b> <Image Tokens> <b>Assistant:</b> <i>Thought: I will carefully examine the image to identify the number in front of the face of the person on the second floor.</i> <i>Code:</i>
<code># Observing the image, I will analyze visible details to identify the required number.</code>
<b>Environment:</b> (Empty) <b>Assistant:</b> <i>Thought: I will zoom into the image to locate the person on the second floor and identify the number in front of their face.</i> <i>Code:</i>
<code># Using tools to zoom into the image isn't possible directly here. Observing the image manually instead.</code>
<b>Environment:</b> (Empty) <i>... (Omitted for brevity. The agent repeats similar examining thoughts and generates code comments for direct reasoning.) ...</i> <b>Assistant:</b> <i>Thought: I will carefully analyze the image again to identify the number in front of the face of the person on the second floor.</i> <i>Code:</i>
<code># Observing the image closely to determine the number in front of the person on the second floor.</code>
<b>Environment:</b> (Empty) <b>Assistant:</b> <i>Thought: The number in front of the face of the person on the second floor appears to be '88'.</i> <i>Code:</i>
<code>submit_final_answer(88)</code>
<b>Environment:</b> Final answer: 88

Table 8: Output example 1 (cont.)

<div data-bbox="673 185 943 217" data-label="Section-Header"> <p><b>DynaSaur (w/ Incentive):</b></p> </div> <div data-bbox="194 215 303 241" data-label="Text"> <p><b>Assistant:</b></p> </div> <div data-bbox="194 239 1287 271" data-label="Text"> <p><i>Thought: To identify the number on the poster, I will apply OCR techniques to the image and analyze the results.</i></p> </div> <div data-bbox="194 268 266 295" data-label="Text"> <p><i>Code:</i></p> </div> <div data-bbox="194 311 946 508" data-label="Text"> <pre>from PIL import Image import pytesseract # Load the image image_path = "data/viz_inspector/validation/images/city-night.jpg" image = Image.open(image_path) # Perform OCR to extract text extracted_text = pytesseract.image_to_string(image) # Output the extracted text extracted_text</pre> </div> <div data-bbox="194 526 343 555" data-label="Text"> <p><b>Environment:</b></p> </div> <div data-bbox="194 580 303 607" data-label="Text"> <p><b>Assistant:</b></p> </div> <div data-bbox="194 604 1434 663" data-label="Text"> <p><i>Thought: The OCR extraction did not yield any results. I will use a visual inspection tool to locate and analyze the specific area of the image.</i></p> </div> <div data-bbox="194 660 266 687" data-label="Text"> <p><i>Code:</i></p> </div> <div data-bbox="194 701 967 730" data-label="Text"> <pre>inspect_image("data/viz_inspector/validation/images/city-night.jpg")</pre> </div> <div data-bbox="194 748 343 777" data-label="Text"> <p><b>Environment:</b></p> </div> <div data-bbox="194 775 847 806" data-label="Text"> <p><i>Inspect image: data/viz_inspector/validation/images/city-night.jpg</i></p> </div> <div data-bbox="194 826 303 855" data-label="Text"> <p><b>Assistant:</b></p> </div> <div data-bbox="194 853 1070 884" data-label="Text"> <p><i>Thought: I'll crop and save the second floor's region from the image for closer inspection.</i></p> </div> <div data-bbox="194 882 266 909" data-label="Text"> <p><i>Code:</i></p> </div> <div data-bbox="194 925 1000 1292" data-label="Text"> <pre>from PIL import Image  # Load the image image_path = 'data/viz_inspector/validation/images/city-night.jpg' image = Image.open(image_path)  # Define the approximate coordinates for the second floor # These coordinates are estimated based on the image structure second_floor_region = (0, 100, image.width, 300) # Adjust as necessary # Crop the second floor region second_floor_image = image.crop(second_floor_region)  # Save the cropped image for inspection cropped_image_path = 'workspace/second_floor.jpg' second_floor_image.save(cropped_image_path)  cropped_image_path</pre> </div> <div data-bbox="194 1310 343 1339" data-label="Text"> <p><b>Environment:</b></p> </div> <div data-bbox="194 1337 481 1366" data-label="Text"> <p><i>'workspace/second_floor.jpg'</i></p> </div> <div data-bbox="194 1388 303 1417" data-label="Text"> <p><b>Assistant:</b></p> </div> <div data-bbox="194 1415 1163 1447" data-label="Text"> <p><i>Thought: I will now inspect the cropped image to focus on the second floor and identify the number.</i></p> </div> <div data-bbox="194 1444 266 1471" data-label="Text"> <p><i>Code:</i></p> </div> <div data-bbox="194 1487 686 1516" data-label="Text"> <pre>inspect_image('workspace/second_floor.jpg')</pre> </div> <div data-bbox="194 1534 767 1563" data-label="Text"> <p><b>Environment:</b> <i>Inspect image: workspace/second_floor.jpg</i></p> </div> <div data-bbox="194 1585 303 1615" data-label="Text"> <p><b>Assistant:</b></p> </div> <div data-bbox="194 1612 1434 1644" data-label="Text"> <p><i>Thought: The cropped section of the second floor is visible. I will use OCR again on this region to extract any numbers or text.</i></p> </div> <div data-bbox="194 1639 266 1668" data-label="Text"> <p><i>Code:</i></p> </div> <div data-bbox="194 1684 924 1899" data-label="Text"> <pre>from PIL import Image import pytesseract  # Perform OCR on the cropped second floor image cropped_image_path = 'workspace/second_floor.jpg' second_floor_image = Image.open(cropped_image_path) extracted_text = pytesseract.image_to_string(second_floor_image)  # Output the extracted text extracted_text</pre> </div> <div data-bbox="194 1917 343 1946" data-label="Text"> <p><b>Environment:</b></p> </div>
---

Table 9: Output example 1 (cont.)



**Question:** In this photo, how many books are displayed adjacent to the glass window with their full front covers visible? Provide a numeric answer.  
**Correct answer:** 38



**Gemini output:**

1. *Image Description:* The image showcases the storefront of John Sandoe Books Ltd. The window display features numerous books, both stacked and upright, along with two hanging lamps. A person can be seen inside the store, seemingly browsing or working amongst more bookshelves.
2. *Question Analysis:* The question asks for a count of books displayed directly against the windowpane, with their front covers fully visible to someone outside looking in.
3. *Reasoning Path:*  
*Step 1:* Locate the bookshelves adjacent to the window. These are the lowest shelves, visually separated from the interior bookshelves by the window's frame.  
*Step 2:* Identify books placed upright with their spines facing outwards towards the window, making their covers fully visible. Books stacked horizontally or at angles where the full cover isn't directly facing outwards are excluded from the count.  
*Step 3:* Count the number of books meeting this criteria across the whole window display.
4. *Tools or Techniques Used:*
  - Object detection (books)
  - Spatial reasoning (adjacent to the window, facing outwards)
  - Counting
5. *Final Answer:* Approximately **53** books are displayed with their front covers visible in the window. (This count might have a small margin of error due to the image resolution and some partially obscured books.)

**Molmo output:**

1. *Image Description:* The image shows a bookstore window display with books arranged on shelves. There are multiple rows of books, with the front row showing their covers.
2. *Question Analysis:* The question asks for the count of books with their full front covers visible adjacent to the glass window.
3. *Reasoning Path:*  
*Step 1:* Identify the area adjacent to the glass window.  
*Step 2:* Count the number of books in the front row of this area.  
*Step 3:* Ensure each counted book has its full front cover visible.
4. *Tools or Techniques Used:*
  - Object detection to locate books.
  - Counting algorithm to determine the number of books.
  - Spatial reasoning to identify the area near the window.
  - Attribute recognition to confirm full front covers are visible.
5. *Final Answer:* **15**

Table 10: Output example 2

HF Agent:
<p><b>Assistant:</b> Here's the plan of action that I will follow to solve the task:</p> <ol style="list-style-type: none"> <li>1. Use the 'visualizer' tool to analyze the image "data/viza/validation/book display.jpg" to answer the specific question: "How many books are displayed adjacent to the glass window with their full front covers visible?"</li> <li>2. Based on the output from the 'visualizer', determine the exact number of books that meet the criteria of being adjacent to the glass window with their full front covers visible.</li> <li>3. Use the 'final_answer' tool to provide the determined number as the final answer to the task.</li> </ol> <p>Let's proceed with the first step by using the 'visualizer' tool.</p> <p><b>Error:</b> Error in code parsing: The code blob you used is invalid: due to the following error: 'NoneType' object has no attribute 'group' This means that the regex pattern “‘(?:py python)? (.*)’” was not respected: make sure to include code with the correct pattern, for instance: Thoughts: Your thoughts Code: “‘py # Your Python code here “‘&lt;end_action&gt;. Make sure to provide correct code</p> <p><b>Assistant:</b> Thought: I will use the 'visualizer' tool to analyze the image and find out how many books have their full front covers visible adjacent to the glass window. This will help me determine the exact count needed for the task. Code:</p> <pre>result = visualizer(question="How many books are displayed adjacent to the glass window with their full front covers visible?", image_path="data/viza/validation/book display.jpg")\ print(result)\</pre> <p><b>Environment:</b> There are 28 books displayed adjacent to the glass window with their full front covers visible.</p> <p><b>Assistant:</b> Code:</p> <pre>final_answer("There are 28 books displayed adjacent to the glass window with their full front covers visible .")</pre> <p><b>Environment:</b></p>

Table 11: Output example 2 (cont.)

DynaSaur (w/o Incentive):
<p><b>Assistant:</b> Thought: I will inspect the image using the appropriate tool to analyze its content. Code:</p> <pre>inspect_image("data/viz_inspector/validation/images/book display.jpg")</pre> <p><b>Environment:</b> Inspect image: data/viz_inspector/validation/images/book display.jpg</p> <p><b>Assistant:</b> Thought: I will carefully count the books adjacent to the glass window with their full front covers visible. Code:</p> <pre># Observing the image visually, I will count the books with their full front covers visible next to the glass window. # From the image, I can see the books displayed adjacent to the glass window. # Counting them manually, there are 23 books with their full front covers visible. # Submitting the answer based on this observation. submit_final_answer("23")</pre> <p><b>Environment:</b> Final answer: 23</p>

Table 12: Output example 2 (cont.)

<p align="center"><b>DynaSaur (w/ Incentive):</b></p>
<p><b>Assistant:</b>  <i>Thought: I will use OCR to count the number of books with visible front covers next to the glass window.</i>  <i>Code:</i></p>
<pre>from PIL import Image import pytesseract # Load the image image_path = 'data/viz_inspector/validation/images/book_display.jpg' image = Image.open(image_path) # Perform OCR to detect text text = pytesseract.image_to_string(image) # Display the OCR result print(text)</pre>
<p><b>Environment:</b> JOHN SANDOE BOOKS LTD  : &lt;% +". . = Aaa Tana wane' fat ... <i>(Omitted for brevity. The generated code failed to return coherent text.) ...</i></p>
<p><b>Assistant:</b>  <i>Thought: To accurately count the books, I will use object detection to identify the distinct full front covers visible in the image adjacent to the glass window.</i>  <i>Code:</i></p>
<p>... <i>(Omitted for brevity. The agent attempts to use YOLO and the detecto package but encounters syntactic errors.) ...</i></p>
<p><b>Assistant:</b>  <i>Thought: I will count the books by analyzing the image using OpenCV and outputting the count directly without visual display.</i>  <i>Code:</i></p>
<pre>import cv2 import numpy as np  # Load the image image_path = 'data/viz_inspector/validation/images/book_display.jpg' image = cv2.imread(image_path) # Convert the image to grayscale gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) # Apply edge detection edges = cv2.Canny(gray, 50, 150) # Find contours contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) # Filter contours by size to approximate book covers book_contours = [c for c in contours if cv2.contourArea(c) &gt; 1000] # Count the filtered contours book_count = len(book_contours) book_count</pre>
<p><b>Environment:</b> 72</p>
<p><b>Assistant:</b>  <i>Thought: The detected contours suggest there are 72 objects that could be books. However, this number seems high due to potential over-detection of objects. I will refine the method to filter out irrelevant contours and focus on book-shaped objects.</i>  <i>Code:</i></p>
<pre># Refine the detection method to count only book-shaped objects # Filter contours by aspect ratio (to match book dimensions) book_contours_refined = [c for c in book_contours if 1.5 &lt; cv2.boundingRect(c)[2] / cv2.boundingRect(c)[3] &lt; 3] # Count the refined contours refined_book_count = len(book_contours_refined) refined_book_count</pre>
<p><b>Environment:</b> 13</p>
<p><b>Assistant:</b>  <i>Thought: The refined method detected 13 books with visible front covers adjacent to the glass window. This count is more accurate and aligns with the task requirements.</i>  <i>Code:</i></p>
<pre>submit_final_answer("13")</pre>
<p><b>Environment:</b> Final answer: 13</p>

Table 13: Output example 2 (cont.)



**Question:** What is the name of the shop that is located across the street from the lot for sale in this photo? Provide an answer in fewer than 3 words.

**Answer:** Wheat Bay;Uniquely Chengdu; Wheat Bay(Uniquely Chengdu)



**Gemini output:**

1. *Image Description:* An aerial view of a city block with streets, buildings, parking lots, and trees. A rectangular lot is highlighted in yellow, marked "For Sale."

2. *Question Analysis:* The question asks for the name of a shop across the street from the "For Sale" lot. It requires identifying the lot, the street it faces, the opposite side of that street, and the shop located there.

3. *Reasoning Path:*

Step 1: Locate the "For Sale" lot (highlighted in yellow).

Step 2: Identify the street it faces (Alder Street).

Step 3: Determine the opposite side of Alder Street.

Step 4: Identify the shop across Alder Street from the for-sale lot. It appears to be a multi-tenant building.

4. *Tools or Techniques Used:*

- Object recognition (identifying the for-sale lot and buildings).

- Spatial reasoning (determining "across the street").

- Text recognition ("For Sale" label).

5. *Final Answer:* **Not identifiable.**

**Molmo output:**

1. *Image Description:* The image shows a city street with buildings and a "For Sale" sign on a vacant lot. Across the street, there's a building with a visible sign.

2. *Question Analysis:* The question asks for the name of a specific shop located opposite the lot for sale.

3. *Reasoning Path:*

Step 1: Locate the "For Sale" sign on the vacant lot.

Step 2: Identify the building directly across the street from the lot.

Step 3: Read the sign on that building to determine the shop name.

4. *Tools or Techniques Used:*

- Object detection to locate the "For Sale" sign and buildings.

- Spatial reasoning to determine "across the street" relationships.

- Text recognition to read the shop name on the sign.

5. *Final Answer:* **Kincaid Street**

Table 14: Output example 3

<b>HF Agent:</b>
<b>Assistant:</b> <i>Thought: I will start by analyzing the image using the 'visualizer' tool to identify any visible signage or distinguishing features of the building across the street from the lot for sale. This will help in determining the name of the shop.</i> <i>Code:</i>
<pre>image_path = "data/viza/validation/821-825-E-13th-Ave-Eugene-OR-Building-Photo-1-HighDefinition.png" question = "What is the name of the shop or any visible signage on the building across the street from the lot for sale?" visualizer_result = visualizer(question=question, image_path=image_path) print(visualizer_result)\\</pre>
<b>Environment:</b> <i>The name visible on the building across the street from the "For Sale" lot is "STONEWORKS MASSAGE &amp; SKIN CARE."</i>
<b>Assistant:</b> <i>Thought: The visualizer tool has successfully identified the name of the shop across the street from the lot for sale as "STONEWORKS MASSAGE &amp; SKIN CARE." I will now provide this as the final answer.</i> <i>Code:</i>
<pre>final_answer("STONEWORKS MASSAGE &amp; SKIN CARE")</pre>
<b>Environment:</b>

Table 15: Output example 3 (cont.)

<b>DynaSaur (w/o Incentive):</b>
<b>Assistant:</b> <i>Thought: I will analyze the image to identify the shop located across the street from the lot marked 'For Sale'.</i> <i>Code:</i>
<pre># Since I have the image loaded, I will inspect it visually to identify the shop across the street. image_path = "data/viz_inspector/validation/images/821-825-E-13th-Ave-Eugene-OR-Building-Photo-1-HighDefinition.png" inspect_image(image_path)</pre>
<b>Environment:</b> <i>Inspect image: data/viz_inspector/validation/images/821-825-E-13th-Ave-Eugene-OR-Building-Photo-1-HighDefinition.png</i>
<b>Assistant:</b> <i>Thought: I will analyze the details in the image to identify the shop's name and provide a definitive answer.</i> <i>Code:</i>
<pre># Observing the image, I will determine the name of the shop located directly across the street from the highlighted lot.</pre>
<b>Environment:</b> (empty)
<b>Assistant:</b> <i>Thought: The shop across the street from the highlighted lot is visible in the image. I will zoom in and analyze the details to identify its name.</i> <i>Code:</i>
<pre># The image already provides a clear view. I will analyze the text and features visible in the image to identify the shop.</pre>
<b>Environment:</b> (empty)
<b>Assistant:</b> <i>Thought: The shop across the street from the lot for sale is visible in the image. Its name can be determined by observing the signage on the building directly opposite the lot.</i> <i>Code:</i>
<pre># Based on the image, the shop's name can be derived from the signage visible across the street. No additional processing is needed.</pre>
<b>Environment:</b> (empty)
<b>Assistant:</b> <i>Thought: The name of the shop across the street is visible in the image. It is "Starbucks Coffee".</i> <i>Code:</i>
<pre>submit_final_answer("Starbucks Coffee")</pre>
<b>Environment:</b> <i>Final answer: Starbucks Coffee</i>

Table 16: Output example 3 (cont.)

DynaSaur (w/ Incentive):
<p><b>Assistant:</b>  Thought: I will use OCR to identify the name of the shop across the street from the lot marked as 'For Sale' in the image.  ... (Omitted for brevity. The agent repeatedly used pytesseract to extract the street name but failed to retrieve any text.) ...</p> <p><b>Assistant:</b>  Thought: I will attempt a different approach by using an external search to identify the businesses in the area based on the address and image context.  Code:</p>
<pre># I will search for businesses in the area of '821-825 E 13th Ave, Eugene, OR' to identify the shop across the street.\\ query = "businesses near 821-825 E 13th Ave, Eugene, OR"\\ informational_web_search(query)</pre>
<p><b>Environment:</b> A Google search for 'businesses near 821-825 E 13th Ave, Eugene, OR' found 9 results: ... (Omitted for brevity) ...</p> <p><b>Assistant:</b>  Thought: I will refine the search to focus on identifying the specific shop across the street from the lot for sale.  Code:</p>
<pre># Searching for the specific shop across the street from the lot for sale in Eugene, OR.\\ query = "shop across the street from 821-825 E 13th Ave, Eugene, OR"\\ informational_web_search(query)</pre>
<p><b>Environment:</b> A Google search for 'shop across the street from 821-825 E 13th Ave, Eugene, OR' found 10 results: ... (Omitted for brevity) ...</p> <p><b>Assistant:</b>  Thought: Based on the search results, it seems Espresso Roma is a notable shop near the specified location. I will confirm this by cross-referencing additional sources.  Code:</p>
<pre>navigational_web_search("Espresso Roma 825 E 13th Ave Eugene OR")</pre>
<p><b>Environment:</b> "Address: <a href="https://www.yelp.com/biz/espresso-roma-eugene">https://www.yelp.com/biz/espresso-roma-eugene</a> Title: ESPRESSO ROMA - Updated February 2025 - 64 Photos &amp; 138 Reviews - 825 E 13th Ave, Eugene, Oregon - Coffee &amp; Tea - Phone Number - Yelp Viewport position: Showing page 1 of 2. ... (Omitted for brevity) ..."</p> <p><b>Assistant:</b>  Thought: Based on the evidence gathered, the shop across the street is likely Espresso Roma. I will finalize the answer.  Code:</p>
<pre>submit_final_answer("Espresso Roma")</pre>
<p><b>Environment:</b> Final answer: <b>Espresso Roma</b></p>

Table 17: Output example 3 (cont.)