HYDRA: A Hyper Agent for Dynamic Compositional Visual Reasoning

Fucai Ke*[1,2], Zhixi Cai*[2], Simindokht Jahangard*[2], Weiqing Wang[2], Pari Delir Haghighi[2], and Hamid Rezatofighi[2]

[1] Building 4.0 CRC, Melbourne 3145, Australia
[2] Faculty of Information Technology, Monash University, Melbourne 3800, Australia
{fucai.ke1,zhixi.cai,simindokht.jahangard,teresa.wang, pari.delir.haghighi,hamid.rezatofighi}@monash.edu
https://hydra-vl4ai.github.io/

Abstract. Recent advances in visual reasoning (VR), particularly with the aid of Large Vision-Language Models (VLMs), show promise but require access to large-scale datasets and face challenges such as high computational costs and limited generalization capabilities. Compositional visual reasoning approaches have emerged as effective strategies; however, they heavily rely on the commonsense knowledge encoded in Large Language Models (LLMs) to perform planning, reasoning, or both, without considering the effect of their decisions on the visual reasoning process, which can lead to errors or failed procedures. To address these challenges, we introduce HYDRA, a multi-stage dynamic compositional visual reasoning framework designed for reliable and incrementally progressive general reasoning. HYDRA integrates three essential modules: a planner, a Reinforcement Learning (RL) agent serving as a cognitive controller, and a reasoner. The planner and reasoner modules utilize an LLM to generate instruction samples and executable code from the selected instruction, respectively, while the RL agent dynamically interacts with these modules, making high-level decisions on selection of the best instruction sample given information from the historical state stored through a feedback loop. This adaptable design enables HYDRA to adjust its actions based on previous feedback received during the reasoning process, leading to more reliable reasoning outputs and ultimately enhancing its overall effectiveness. Our framework demonstrates state-of-the-art performance in various VR tasks on four different widely-used datasets.

Keywords: Visual reasoning · Large language Models (LLMs) · Reinforcement learning

1 Introduction

Visual reasoning (VR) involves constructing a detailed representation of a visual scene and reasoning through it in steps, similar to human cognition, often in response to textual queries or prompts [2]. It encompasses various tasks, including

* These authors contributed equally to this work

but not limited to Visual Question Answering (VQA) [2], Visual Commonsense Reasoning (VCR) [50], and Visual Grounding (VG) [47]. In recent years, ad-

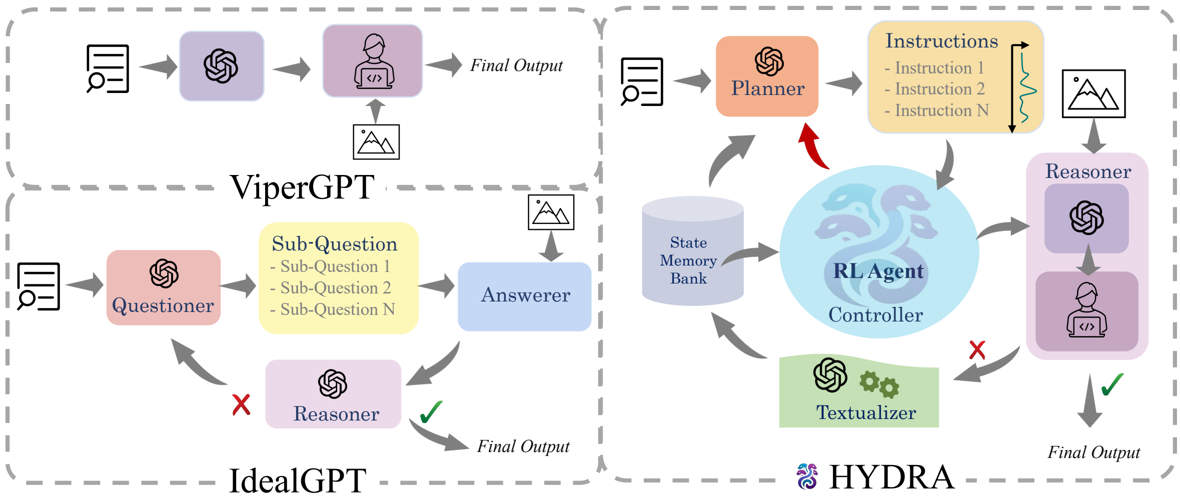

Fig. 1: Comparison of ViperGPT [38], IdealGPT [46], and HYDRA: ViperGPT employs a single feed-forward process approach, IdealGPT breaks down questions into sub-questions using a loop, while HYDRA utilizes diverse instructions and an RL agent in an incremental loop for feedback, showcasing its superior adaptability and efficiency in handling complex visual reasoning challenges.

vancements in Large Language Models (LLMs) [3,6,28,31] and their derivatives, such as VLMs [18,35,43,53] have sparked hope for their effectiveness in solving visual reasoning tasks. While these models have shown promising results in certain tasks like VQA and VCR [52], their training as single monolithic end-to-end models necessitates large-scale datasets, imposing significant computational resource requirements. Additionally, while these models excel within their training domain, they may require further adaptation to achieve reliable performance when applied to diverse datasets or domains [35,38,46].

In recent advancements, compositional approaches [9,24,38,42] have emerged as effective strategies for addressing VR challenges. These approaches break down complex tasks into simpler sub-components, employing a divide-and-conquer methodology. They employ LLMs alongside Visual Foundation Models (VFMs) without requiring extensive training. LLMs can function as planners, code generators, or reasoner, while VFMs act as visual perception components, facilitating structured analysis and task-specific plan generation to enhance adaptability and improve generalization across diverse scenarios. A recent SoTA compositional model is ViperGPT [38], which utilizes LLMs to generate code programs for visual queries and solve the task in a single feed forward process. IdealGPT [46] proposed an enhanced framework by utilising LLMs as both questioners and reasoners, with a pre-trained Vision-Language Model (VLM) serving as the answerer, Figure 1. In this model, LLM decomposes main questions into

sub-questions, with the reasoner determining whether further sub-question generation is required through iterations or if the final output has been reached.
However, these models come with certain limitations. Primarily, the outputs generated by LLMs may sometimes lack meaning, and when these outputs proceed to subsequent steps without verification, they can impact the outputs of other components, thus adversely affecting overall performance. Moreover, LLMs utilized in the planner or questioner during the initial step lack information from visual content (perception module) in later states to adjust their outputs [9,13]. Additionally, the process of generating subsequent questions often begins from scratch without storing information from previous steps, potentially leading to more iterations. Furthermore, these approach heavily rely on commomsense knowledge encoded in LLMs to do planning and reasoning for VR tasks.
In this paper, we present HYDRA, a HYper agent for Dynamic compositional visual ReAsoning, an innovative framework designed to address the aforementioned challenges. HYDRA is composed of three main modules planner, controller (Reinforcement Learning-based agent (RL)) and reasoner. Notably, in the planner, upon receiving textual queries, unlike prior compositional approaches, we employed LLM to generate some instruction samples with varying depths based on a distribution, instead of relying on a single instruction sample. Furthermore, we integrate a hyper RL agent to dynamically interact with some modules to make an high-level decision on the instruction samples generated by LLM in the planner to evaluate their validity. If the RL agent detects any invalid instruction samples, a request is sent back to the planner for alternative suggestions. Conversely, if the instruction samples are considered valid, the chosen instruction sample is forwarded to the reasoner. In the reasoner, the selected instruction sample undergoes analysis by LLM, and the resulting tailored code is sent to the code generator. The code generator employs Python API code to utilize VFMs for additional visual content processing. If the reasoner output is incomplete or fails, the output is converted to textual format in the textualizermodule and then stored in State Memory Bank. Afterwards, another request is then sent back to the planner to generate new instructions, which are again fed to the controller module to select an instruction sample. This iterative process continues incrementally until the final desired output is achieved. The design of HYDRA integrates not only the incremental storage of information from previous states (incremental reasoning), considered by the RL agent, but also the capability to utilize feedback from VFMs acquired from earlier perception processes. This enables dynamic adjustment of actions and responses based on feedback from visual perception modules. This innovative design facilitates hyper decision-making by the hyper RL agent, thereby refining reasoning capabilities and overall effectiveness. The overall design of HYDRA compared with the previous compositional approach is shown in Figure 1. We evaluated our framework on several popular VR datasets and compared it with the advanced models, showing state-of-the-art performance. In sum, the key contributions of this work are as follows:

1. Integrating a cognitive reinforcement learning-based agent as a controller into a framework to foster hyper decision-making and behavior across di-

verse environments, enhancing system cohesion, performance, and reasoning capabilities.

2. Employing LLM as a natural language planner that enables the dynamic generation of valid instruction samples for iterative processing. The samples are vary in both the complexity and scope of perception tasks assigned with validity probabilities.
3. Applying incremental reasoning, storing information from previous states aids both the LLM and RL agent in acquiring fine-grained visual information through VFMs and the visual-perception-to-text module, thereby refining their reasoning processes.

2 Related Work

Single Monolithic End-to-End Methods. Recent advancements in Large Language Models (LLMs) [3,6,28,31] have notably improved their ability to understand and reason visual content. Their derivatives, VLMs, like Video-LLaMA [52] and NExT-GPT [43] excel in comprehending detailed videos and seamlessly integrating text, images, videos, and audio for cross-modal reasoning. Otter [18], Flamingo [1], and Visual ChatGPT [42] further enhance visual reasoning by integrating visual inputs into their language understanding processes, enabling contextually relevant responses. Initiatives like InstructBLIP [8], M^{3}IT [20], and VisionLLM [41] emphasize instruction tuning, multilingual datasets, and vision-centric tasks, advancing language understanding and nuanced video comprehension through a blend of language and visual cues. These developments signal a significant shift towards AI systems proficient in reasoning across textual and visual domains. However, these single monolithic end-to-end models suffer from reduced interpretability, require significant computational power and extensive training data resources. Besides, these models exhibit limited generalization capabilities due to the vast scale of the trained neural networks [35]. Various vision challenges often necessitate distinct models, typically involving the manual selection and assembly of specific models tailored to each particular scenario. Given the exponentially large long tail of compositional tasks, the proposed data-intensive and compute-intensive single monolithic end-to-end models may fall short in solving these types of tasks [40,45]. Consequently, compositional reasoning, generalization, fine-grained spatial reasoning abilities, and counting capabilities remain significant challenges for even the most advanced, large-scale single monolithic end-to-end models [5,12,38,49].
Compositional Visual Reasoning Methods. The compositional approach introduces a strategy aimed at addressing the challenges faced by end-to-end VLMs [9,24,35,38,46]. These models tackle complex tasks by breaking them down into multiple subtasks, solving each one individually, and then utilizing the intermediate outcomes to address the overarching task. These models utilize the potent chain-of-thought (CoT) functionality of LLMs acting as planners, reasoner, etc. This capability facilitates the breakdown of intricate problems into manageable and individually solvable intermediate steps through the provision

of instructions [4,7,14,17]. The instructions may take the form of Python execution code that embodies logical operations [9,38]. For example, Visprog [9] and ViperGPT [38] seek to eliminate the requirement for task-specific training in both programming logic and perception modules by employing code generation models. These strategies facilitate the assembly of VLMs into subroutines, thereby enabling the production of results. An alternative strategy, emblematic of the divide-and-conquer methodology, is exemplified by IdealGPT [46]. This approach harnesses a captioning model for the acquisition of elementary visual data and engages a LLM to serve as a planner. The high-level inquiries are methodically deconstructed into three distinct sub-questions, which are processed concurrently. Following this, perception tools (VFMs) are employed to individually address each sub-question. The outcomes are then aggregated and analyzed by the reasoning mechanism to deduce the comprehensive final response. Moreover, the activation status and the sequential order of VFMs, as utilized by visual perception tools, constitute a form of instruction [24]. The system implements predefined functionalities based on these instructions to systematically activate perception tools in a specified sequence. This process culminates in the aggregation of data, which is subsequently analyzed by the reasoning mechanism to formulate the ultimate conclusion.

All these compositional processing heavily depends on the capability of LLMs to perform commonsense reasoning and make decisions. However, despite their capabilities, LLMs have certain limitations. Primarily, the outputs they generate may lack meaningfulness, and if these outputs proceed to subsequent steps without verification, they can adversely affect the performance of other components. Additionally, LLMs used in planning or questioning lack access to visual content information in later stages, which hinders their ability to adjust outputs accordingly. Moreover, the process of generating subsequent questions often starts anew without retaining information from previous steps, potentially leading to more iterations. Furthermore, these methodologies heavily rely on the common-sense knowledge encoded in LLMs for planning and reasoning within virtual reality tasks. In this paper, we introduce a new framework that utilizes a cognitive reinforcement learning-based agent to address these challenges. This framework enhances decision-making, system performance, and reasoning across different

Table 1: Summary of compositional models, including HYDRA. *IR: Incremental Reasoning. VQA: Visual Question Answering. VG: Visual Grounding. HF: HuggingFace.*

Model	Module					Task	
	Planner	Perception	Reasoner	Controller	IR	VQA	VG
Visprog [9]	✗	VFMs	GPT-3	✗	✗	✓	✓
Chameleon [24]	ChatGPT	VFMs	ChatGPT	✗	✗	✓	✗
IdealGPT [46]	ChatGPT	BLIP2	ChatGPT	✗	✗	✓	✗
HuggingGPT [34]	ChatGPT	HF-VFMs	ChatGPT	✗	✗	✓	✓
ViperGPT [38]	✗	VFMs	Codex	✗	✗	✓	✓
HYDRA	ChatGPT	VFMs	ChatGPT	RL-Agent	✓	✓	✓

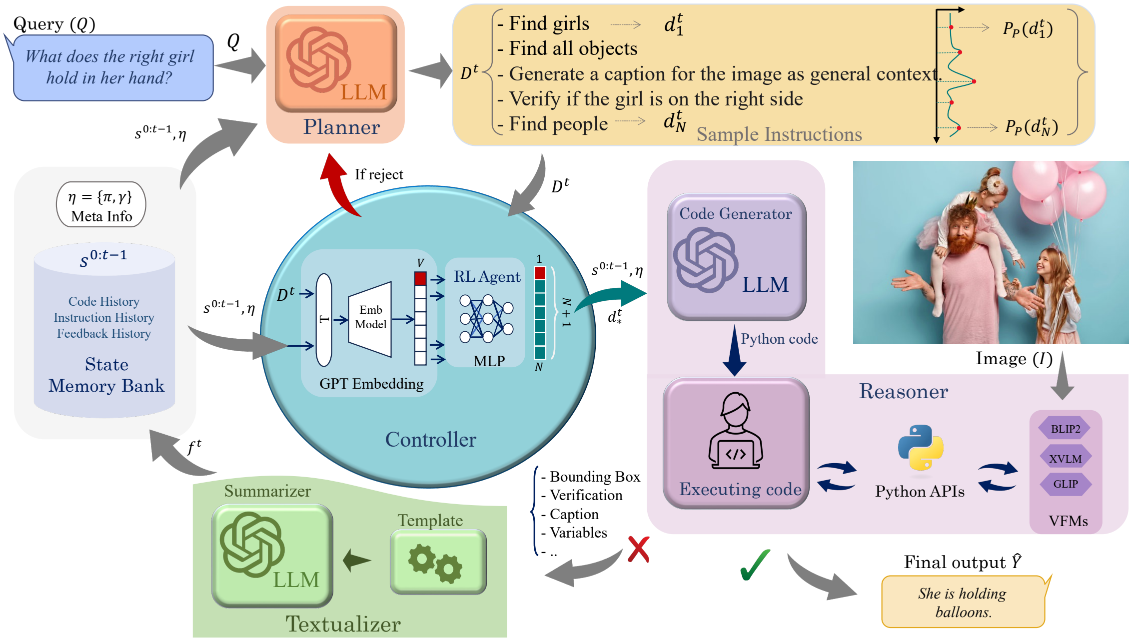

Fig. 2: The HYDRA detailed design includes key modules: planner, controller, reasoner, textualizer, State Memory Bank (s^{t-1}), and meta information (η). Input Q is given to the planner to generate instructions D^t using s^{t-1} and η. The controller receives D^t, and if invalid, requests alternative samples from the planner. Otherwise, it sends chosen instruction d_*^t to the reasoner, which generates perceptual output using Python APIs and VFMs. Incomplete output is converted to textual format, f^t, by the textualizer and stored in State Memory Bank. This process iterates until the desired final output, $\hat{Y}$, is achieved.

scenarios. Moreover, we effectively harness LLM knowledge to generate instructional samples and facilitate incremental reasoning for acquiring detailed visual information. A comparison between recent compositional VR models and our approach is presented in Table 1.

3 Approach

The design of HYDRA are provided Figure 2 in detail, comprising several key modules: planner ($\mathcal{F}_P$), controller ($\mathcal{F}_C^{\theta}$), reasoner($\mathcal{F}_R$), textualizer ($\mathcal{F}_T$), a State Memory Bank and meta information (η). The framework's input comprises query-image pairs, denoted as $X = \{Q, I\}$, and the final output, $\hat{Y}$, can be textual answers or bounding boxes for the visual grounding task. The planner $\mathcal{F}_P$, utilizing LLM, generates some instruction samples based on the input query Q using some information from meta information and State Memory Bank. Then, the generated instruction samples are fed to controller $\mathcal{F}_C$ which is composed of GPT embedding and RL agent that evaluate the validity of instruction samples. If the RL agent detects invalid instruction samples, it forwards a request to the planner for alternative instruction samples; conversely, an instruction sample is picked as the chosen sample, d_*, and sent to the reasoner. The chosen instruc-

Algorithm 1 HYDRA Inference

Require: $X, \mathcal{F}_P, \mathcal{F}_C, \mathcal{F}_R, \mathcal{F}_T, \eta, \theta$
1: $\{Q, I\} \leftarrow X; t \leftarrow 1; f \leftarrow \{\}; d \leftarrow \{\}$ ▷ Initialize the inputs and state
2: **while** not final **do**
3: $s^{0:t-1} \leftarrow \{f, d\}$
4: $D^t \leftarrow \mathcal{F}_P(Q, s, \eta)$ ▷ Generate instructions
5: $d_*^t \leftarrow \text{argmax}_{d_i^t \in D^t} \mathcal{F}_C^\theta(D^t, s^{0:t-1}, \eta) * P_P(d_i^t)$ ▷ Select the optimal instruction
6: **if** D^t is rejected **then go to** 4
7: **end if**
8: $f^t \leftarrow \mathcal{F}_T(\mathcal{F}_R(Q, d_*^t, s, \eta))$ ▷ Execute code and textualize perception results
9: **if** execution error **then go to** 8
10: **end if**
11: $t \leftarrow t + 1$
12: f.append(f^t); d.append(d_*^t) ▷ Update the state
13: **end while**
14: $\hat{Y} \leftarrow$ Extract answer from f ▷ Resolve the final answer
15: **return** $\hat{Y}$

tion sample is fed to the LLM in the reasoning module, and the corresponding Python code is generated in the code generator submodule. Subsequently, this Python code is executed in the executing code submodule utilizing Python APIs and VFMs. If the output is incomplete or unsuccessful, it is converted to textual format through the textualizer module and stored in the State Memory Bank. Thereafter, another request is sent to the planner to generate new instruction samples, which are then provided to the controller module to select a valid instruction sample. This iterative process continues incrementally until the desired final output is obtained.

As HYDRA is a framework that operates through several iterations to simplify the process, we use $s^{0:t}$ to depict the progression from the initial state to the current state $0:t$. Additionally, in the first iteration, there is no information from the previous iteration, denoted as $s^0 = \{\}$. Note that all LLMs in the planner, reasoner, and textualizer are the same, with only their prompts being changed in different modules, and for enhanced clarity, we present them separately in the figure. The algorithm of the whole inference process is provided in Algorithm 1. The technical details for each module, along with further elaboration, are provided in the following.

State Memory Bank & Meta Information. As HYDRA progresses through multiple iterations and considers information from previous ones, all data, including code, instruction, and the output of the reasoner from former iteration, are stored in State Memory Bank, represented by a grey cylinder in Figure 2. Furthermore, meta information encompasses crucial data such as a subset of skills $\pi \in \Pi$ and various task descriptions $\gamma \in \Gamma$ tailored for different tasks that the LLM needs as a prompt. For simplicity, these are denoted as $\eta = \gamma, \pi$ in the subsequent equations.

Planner Module. Highlighted in orange in Figure 2, this module receives Q and other data from the State Memory Bank. It generates N instruction samples (e.g., "find girls", "verify if the girl is on the right side"), d_i^t of varying depth, where each instruction sample can have different actions or levels of complexity. For instance, some instructions may involve simple tasks, while others may entail more intricate actions or multi-step processes. Along with these instruction samples, corresponding confidence probabilities $P_p(d_i^t)$ are provided, indicating the likelihood of each instruction being accurately executed. These outputs are generated by the LLM ChatGPT[*] and are represented by $D^t = \{(d_i^t, P_P(d_i^t))\}_{i=1}^N$ in the yellow box. This process is described by the equation:

$$D^t = \mathcal{F}_P(Q, s^{0:t-1}, \eta) \tag{1}$$

Controller Module. This module serves as the central component of HYDRA, dynamically interacting with other modules to facilitate hyper decision-making and functioning as a cognitive controller.This module integrates embedding, leveraging GPT-3 [4], to extract the features highlighted in a cyan circle in the Figure 2. It takes D^t, η and $s^{0:t-1}$ and embeds them into a vector, V. Subsequently, it passes through an RL agent, which consists of a trainable MLP layer followed by a softmax function with an output size of N+1. Through this module, the instruction samples undergo evaluation and if the RL agent considers them invalid, a request is sent to the planner to regenerate new instruction samples, as indicated by the red arrow in Figure 2. Otherwise, the chosen instruction sample, d_*^t, is selected and proceeds to the reasoner, depicted by the green arrow.

$$d_*^t = \operatorname*{argmax}_{d_i^t \in D^t} \mathcal{F}_C^\theta(D^t, s^{0:t-1}, \eta) * P_P(d_i^t) \tag{2}$$

Training phase. As mentioned earlier, the RL agent is a trainable MLP layer based on Reinforcement Learning, employing the DQN algorithm [27]. During the training phase, the objective of the RL agent is to maximize the expected cumulative reward. The reward function is designed to favour fewer iterations and correct output while penalizing more iterations and incorrect output. We iteratively accumulate the reward function as shown below.

$$R^t = \begin{cases} R^{t-1} - t & \text{if not final step,} \\ R^{t-1} + \alpha m & \text{if answer is relate} \\ R^{t-1} - \alpha & \text{if answer is unrelated} \\ R^1 & \text{if } t = 1 \end{cases} \tag{3}$$

where m is the performance metrics (e.g. accuracy, intersection over union),

[*] https://chat.openai.com/

α and R^1 are the hyperparameter constants. Additional details regarding this phase are provided in the supplementary material.

Reasoner Module. Illustrated in light pink in the Figure 2, this module consists of an LLM as code generator and a code executor sub-module. In this setup, ChatGPT* receives the selected instruction sample d_*^t from the controller module, along with necessary information from the previous iteration, $s^{0:t-1}$, and η , to generate Python code. This Python code is then transferred to the execution sub-module within perception tools, such as VFMs including GLIP [21], BLIP2 [19], LLaVA-1.5 [22], MiDaS [32], and XVLM [51]. Python interpreter to execute the code in the Python context loaded with the predefined Python APIs. In the execution, all the variable values (perceptual output) are collected and logged and will be sent to the next module via the feedback.

$$\text{perceptual output} = \mathcal{F}_R(Q, d_*^t, s^{0:t-1}, \eta) \tag{4}$$

Textualizer Module. If the perceptual output from the reasoner module is incomplete or unsuccessful, it undergoes conversion to textual format within this module, as depicted by the green in Figure 2. The perceptual output from the reasoner, which may consist of bounding boxes, verifications, or captions, is transformed into a textual format using a template. This conversion ensures that the input is understandable for the LLM and ensures that all information stored in the State Memory Bank has the same format that can be used in the next iterations. Subsequently, the LLM summarizes the current state information, f^t, and stores it in State Memory Bank. Further details about these templates are available in the supplementary material.

Technical Details: The iterative process continues incrementally until the desired final output is achieved, which we refer to as the incremental reasoning mechanism. It's worth noting that the HYDRA does not always require iterations; by efficiently integrating the RL agent, the final output of the task can be generated in just a single iteration. That could be due to the simplicity of the task, or the RL agent may choose to select an instruction that includes all the necessary steps for generating the final output in a single iteration.

4 Experiments and Results

Implementation Details: To train our framework, we utilized PyTorch [29] with NVIDIA RTX 4090 GPUs, employing a learning rate of 1×10^{-4} and a batch size of 128. The Multi-Layer Perceptron (MLP) used for the RL agent, consists of three layers with dimensions 1536, 512, and 6. The hyper-parameters for reinforcement learning are set as $R^1 = 100$ and $\alpha = 100$. During the training process, early stopping is applied once the reward converges. For a fair comparison, we evaluated the state-of-the-art (SoTA) baselines using configurations from their official code repositories and papers [35]. We utilized the largest available backbone for the end-to-end VLMs. We also replaced ChatGPT as the code generator in ViperGPT [38], given the discontinuation of GPT3 Codex by OpenAI [38]. Supplementary materials provide additional

Table 2: Performance on External Knowledge-dependent Image Question Answering and Visual Grounding tasks.

(a) Performance on OK-VQA.

Type	Method	ACC(%)
E2E	PNP-VQA [39]	35.9
	PICa [44]	43.3
	BLIP-2 [19]	45.9
	Flamingo (9B) [1]	44.7
	MiniGPT-4 (13B) [53]	37.5
	LLaVA (13B) [23]	42.5
	InstructBLIP (13B) [8]	47.9
Comp	IdealGPT [46]	19.4
	ViperGPT [38]	40.7
	HYDRA	**48.6**

(b) Performance on RefCOCO and RefCOCO+

Type	Method	IoU(%) Ref	Ref+
E2E	OWL-ViT [11]	30.3	29.4
	OWLv2 [26]	33.5	31.7
	GLIP [21]	55.0	52.2
	ReCLIP [37]	58.6	60.5
	KOSMOS-2 [30]	57.4	50.7
Comp	Code-bison [35]	44.4	38.2
	ViperGPT [38]	59.8	60.0
	HYDRA	**61.7**	**61.1**

details on implementation including instructions and prompts for the planner, code generator, and controller.

Datasets and Evaluation Metric: We evaluated our framework across three key tasks in visual reasoning. Firstly, External Knowledge-dependent Image Question Answering, for which we utilize the OK-VQA dataset [25] and evaluate performance based on accuracy (ACC) score [38, 46]. Secondly, Compositional Image Question Answering, where the GQA [15] dataset serves as our benchmark, again measured by ACC score [38, 46]. Lastly, Visual Grounding tasks are addressed using the RefCOCO [48] and RefCOCO+ [48] datasets, with evaluation based on Intersection over Union (IoU) metrics [11, 21, 26, 30, 37]. These diverse tasks and corresponding datasets offer comprehensive assessments, collectively contributing to the advancement of our framework's capabilities in visual understanding and interpretation.

Visual Reasoning Tasks and Result Analysis: Detailed elaboration and both quantitative and quantitative results for each task, External Knowledge-dependent Image Question Answering, Visual Grounding, and Compositional Image Question Answering, respectively, are provided below.

External Knowledge-dependent Image Question Answering involves using external sources of information, such as databases, to provide context and answer questions about images that cannot be inferred solely from visual content [38]. Following previous works [38], we additionally employ the LLM [4] knowledge with the module llm-query. The quantitative results from Table 2a highlight the comparison between end-to-end models and compositional models, including HYDRA, on the OK-VQA dataset. HYDRA surpasses previous models by 48.6%, showcasing a remarkable improvement. The incorporation of advanced techniques in HYDRA, such as incremental reasoning mechanisms and leveraging LLM for generating different instructions, greatly contributes to its outstanding performance.

Table 3: Performance on GQA Dataset

Type	Method	ACC(%)
E2E	BLIP-2 [19]	45.5
	MiniGPT-4 (13B) [53]	30.8
	LLaVA (13B) [23]	41.3
	PandaGPT (13B) [36]	41.6
	ImageBind-LLM (7B) [10]	41.2
Comp	IdealGPT [46]	41.7
	ViperGPT [38]	37.9
	HYDRA	**47.9**

Visual Grounding involves predicting bounding boxes based on the input prompt. HYDRA are equipped with reasoner module which contain grounding-related VFM APIs such as find, exists, and verify-property, similar to ViperGPT. Our method, as shown in Table 2b, surpasses the state-of-the-art baselines for IoU on RefCOCO [48] and RefCOCO+ [48] datasets. Among the end-to-end methods, grounding-specialized approaches like GLIP [21] and ReCLIP [37] achieve superior performance compared to the VLM KOSMOS-2 [30]. Considering that KOSMOS-2 can also handle other text-based tasks. When comparing methods between end-to-end and compositional approaches, we observe that both compositional visual reasoning approaches (ViperGPT [38] and HYDRA) achieve better performance than end-to-end baselines. This indicates that the compositional approach design is more adept at solving the VG task.

Compositional Image Question Answering contains complex questions. These questions require the decomposition into simpler steps for answering. Similar to previous works [38], we utilize the BLIP2 [19] API simple-query to enhance our understanding of image content. As demonstrated in Table 3 with implementation on the GQA dataset, among the end-to-end models, the 30.8% performance of MiniGPT underscores the importance of instruct tuning. IdealGPT surpasses ViperGPT in performance by leveraging a planner to enhance reasoning capability. Notably, ViperGPT's performance is impeded by the generation of non-executable code snippets, while HYDRA enhances code quality through the integration of multiple sampling and a RL agent controller for code validation, leading to superior performance compared to ViperGPT. Additionally, it highlights that HYDRA achieves an impressive accuracy of 47.9%, underscoring its robustness and effectiveness in handling the GQA dataset. Further results can be found in the supplementary materials.

Generalization Analysis: Generalization abilities play a crucial role in adapting approaches to unseen data distributions without necessitating re-training. Given that the RL agent in HYDRA is the sole component requiring training, we conducted generalization experiments on the OK-VQA and A-OK-VQA [33] dataset, as presented in Table 4, to assess the module's capacity to operate

Table 4: Generalization performance for the RL-Agent. The *Train* column is the training data for training the RL agent, and the *Test* column is the test data for evaluating the method.

Method	Train	Test	ACC(%)
ViLT [16]	GQA	OK-VQA	32.13
ViperGPT [38]	–	OK-VQA	40.74
HYDRA	GQA	OK-VQA	48.17
HYDRA	OK-VQA	OK-VQA	48.63
HYDRA	OKVQA	A-OKVQA	55.94
HYDRA	A-OKVQA	A-OKVQA	56.35

Table 5: An ablation study for HYDRA on GQA.

Models	RL-Agent	IR	Sampling	ACC
ViperGPT	✗	✗	✗	37.94
	✓	✗	✗	43.71
	✗	✗	✓	39.84
	✓	✗	✓	45.98
	✗	✓	✗	41.08
	✓	✓	✗	47.07
	✗	✓	✓	46.93
HYDRA	✓	✓	✓	**47.88**

effectively on unseen data without explicit training. ViLT [16] is chosen as the baseline end-to-end method, which does not require expensive computational resources. Notably, the performance of our model, HYDRA, in the cross-dataset experiments (i.e., training on GQA and testing on OK-VQA, and training on OK-VQA and testing on A-OK-VQA) closely matches intra-dataset performance as shown in Table 4. Furthermore, this cross-dataset performance surpasses that of the baseline ViLT [16], which achieved an accuracy of 32.13%. Additionally, ViperGPT [38] exhibits superior performance compared to ViLT, showcasing the superiority of compositional over end-to-end methods in generalizability. Comparison with ViperGPT also reveals superior performance, as HYDRA trained on alternative datasets achieved accuracies of 48.17%. These findings underscore the generalizability of the RL agent controller within HYDRA.

Qualitative Analysis: Figure 3 demonstrates intermediate processes of HYDRA for two examples, one for visual question-answering and one for visual grounding tasks. We show detailed examples with multiple steps in the first example in each figure, and the brief examples only show the last iteration in the loop. It is observed that the meaningful perception results are summarized as useful feedback for the next iteration of planning and reasoning. Figure 4 includes more qualitative examples of the results using HYDRA on these tasks.
Failure Analysis. While HYDRA has achieved SoTA performance, there is still room for further improvement in its design. In complex cases, as illustrated in Figure 5, HYDRA may fail due to potential mistakes made by the LLMs within the reasoner and textualizer module. In future iterations, we plan to enhance the complexity of the RL agent, enabling it to exert greater control over the output

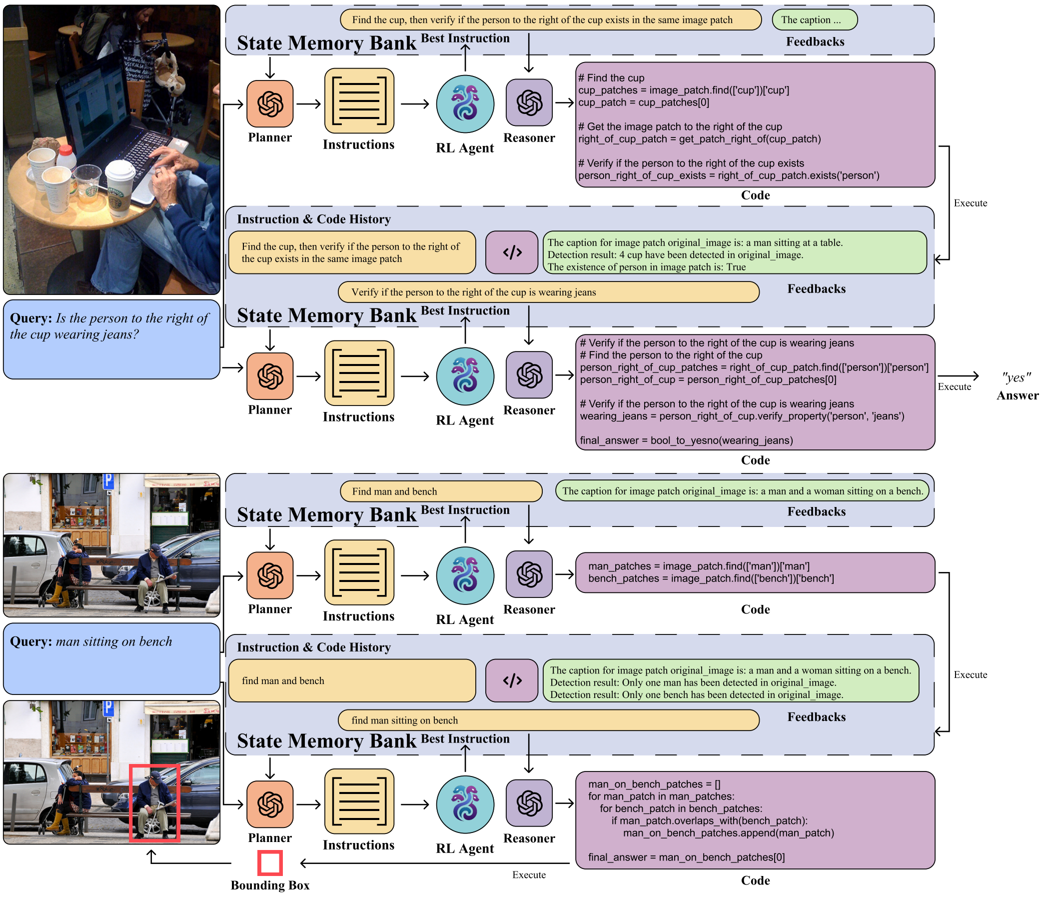

Fig. 3: Detailed result examples from HYDRA. The first example describes the intermediate results of the full two iterations in the loop for question answering, whereas the second example is about the grounding task.

of LLMs, whether functioning as code generators or summarizers. Additional fail rate analysis can be found in the supplementary materials.

4.1 Ablation Study

In this section, we provide an ablation study on suggested key components of HYDRA demonstrating their contributions to the final results.

Component Analysis. As previously mentioned, there are three main contributions in HYDRA: the RL agent, Sampling (involving instruction sampling numbers), and Incremental Reasoning (IR). Through this experiment, the efficacy of each component is evaluated and presented in Table 5. As depicted in Table 5, the first column displays the models and their variants, while the following three columns represent each key component: RL agent, Incremental Reasoning (IR), and Sampling respectively. The last column, denoted as ACC, represents the accuracy achieved by each model on the GQA dataset. As shown in Table 5, the RL-Agent significantly improves the overall architecture, achieving an average enhancement of 4.71% in accuracy compared to the variants with the same settings on IR and sampling but without the RL-Agent. Additionally,

Fig. 4: More result examples from HYDRA for question answering and visual grounding tasks.

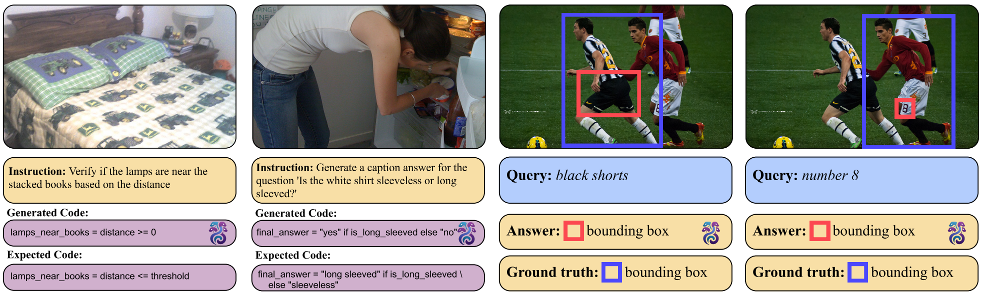

Fig. 5: Failure result examples from HYDRA. The left two samples are due to wrong generating codes. The right two failure cases are due to wrong annotation.

both IR and Sampling further boost the framework's performance by 3.87% and 2.70% on average, compared with the corresponding variant without IR or sampling. Further implementation details can be found in the supplementary.

5 Conclusion

In this paper, we introduced HYDRA, a multi-step dynamic compositional visual reasoning framework designed to improve reasoning steadily and reliably. HYDRA combines three key parts: a planner, a RL agent acting as a cognitive controller, and a reasoner. The planner and reasoner modules use an LLM to create instruction samples and executable code from chosen instructions, while the RL agent interacts with these modules to make decisions based on past feedback, adjusting its actions as needed. This flexible setup allows HYDRA to learn from previous experiences during the reasoning process, resulting in more dependable outcomes and overall better performance. In future, our goal is to enhance our framework by fostering greater interaction between the LLM in the reasoner and the texturizer module to mitigate potential errors.

Acknowledgements

This research is partially supported by Building 4.0 CRC. We acknowledge the support of the Commonwealth of Australia through the Cooperative Research Centre Programme. Additionally, this material is based on research partially sponsored by the DARPA Assured Neuro Symbolic Learning and Reasoning (ANSR) program under award number FA8750-23-2-1016.

References

1. Alayrac, J.B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al.: Flamingo: a visual language model for few-shot learning. Advances in Neural Information Processing Systems **35**, 23716–23736 (2022)
2. Amizadeh, S., Palangi, H., Polozov, A., Huang, Y., Koishida, K.: Neuro-symbolic visual reasoning: Disentangling. In: International Conference on Machine Learning. pp. 279–290. PMLR (2020)
3. Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al.: Constitutional ai: Harmlessness from ai feedback. arXiv preprint arXiv:2212.08073 (2022)
4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
5. Bugliarello, E., Sartran, L., Agrawal, A., Hendricks, L.A., Nematzadeh, A.: Measuring progress in fine-grained vision-and-language understanding. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1559–1582. Association for Computational Linguistics, Toronto, Canada (Jul 2023). https://doi.org/10.18653/v1/2023.acl-long.87, https://aclanthology.org/2023.acl-long.87
6. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al.: Palm: Scaling language modeling with pathways. Journal of Machine Learning Research **24**(240), 1–113 (2023)
7. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al.: Palm: Scaling language modeling with pathways. Journal of Machine Learning Research **24**(240), 1–113 (2023)
8. Dai, W., Li, J., Li, D., Tiong, A.M.H., Zhao, J., Wang, W., Li, B., Fung, P., Hoi, S.: Instructblip: Towards general-purpose vision-language models with instruction tuning (2023)
9. Gupta, T., Kembhavi, A.: Visual programming: Compositional visual reasoning without training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14953–14962 (2023)
10. Han, J., Zhang, R., Shao, W., Gao, P., Xu, P., Xiao, H., Zhang, K., Liu, C., Wen, S., Guo, Z., et al.: Imagebind-llm: Multi-modality instruction tuning. arXiv preprint arXiv:2309.03905 (2023)

11. Heigold, G., Minderer, M., Gritsenko, A., Bewley, A., Keysers, D., Lučić, M., Yu, F., Kipf, T.: Video owl-vit: Temporally-consistent open-world localization in video. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13802–13811 (2023)
12. Hsieh, C.Y., Zhang, J., Ma, Z., Kembhavi, A., Krishna, R.: Sugarcrepe: Fixing hackable benchmarks for vision-language compositionality. Advances in Neural Information Processing Systems **36** (2024)
13. Hu, Y., Stretcu, O., Lu, C.T., Viswanathan, K., Hata, K., Luo, E., Krishna, R., Fuxman, A.: Visual program distillation: Distilling tools and programmatic reasoning into vision-language models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9590–9601 (June 2024)
14. Huang, W., Abbeel, P., Pathak, D., Mordatch, I.: Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In: International Conference on Machine Learning. pp. 9118–9147. PMLR (2022)
15. Hudson, D.A., Manning, C.D.: 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. pp. 6700–6709 (2019)
16. Kim, W., Son, B., Kim, I.: Vilt: Vision-and-language transformer without convolution or region supervision. In: Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 5583–5594. PMLR (18–24 Jul 2021), https://proceedings.mlr.press/v139/kim21k.html
17. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. Advances in neural information processing systems **35**, 22199–22213 (2022)
18. Li, B., Zhang, Y., Chen, L., Wang, J., Yang, J., Liu, Z.: Otter: A multi-modal model with in-context instruction tuning. arXiv preprint arXiv:2305.03726 (2023)
19. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In: Proceedings of the 40th International Conference on Machine Learning. ICML'23, JMLR.org (2023)
20. Li, L., Yin, Y., Li, S., Chen, L., Wang, P., Ren, S., Li, M., Yang, Y., Xu, J., Sun, X., et al.: M^3it: A large-scale dataset towards multi-modal multilingual instruction tuning. arXiv preprint arXiv:2306.04387 (2023)
21. Li, L.H., Zhang, P., Zhang, H., Yang, J., Li, C., Zhong, Y., Wang, L., Yuan, L., Zhang, L., Hwang, J.N., et al.: Grounded language-image pre-training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10965–10975 (2022)
22. Liu, H., Li, C., Li, Y., Lee, Y.J.: Improved baselines with visual instruction tuning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 26296–26306 (2024)
23. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. Advances in neural information processing systems **36** (2024)
24. Lu, P., Peng, B., Cheng, H., Galley, M., Chang, K.W., Wu, Y.N., Zhu, S.C., Gao, J.: Chameleon: Plug-and-play compositional reasoning with large language models. Advances in Neural Information Processing Systems **36** (2024)
25. Marino, K., Rastegari, M., Farhadi, A., Mottaghi, R.: Ok-vqa: A visual question answering benchmark requiring external knowledge. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. p. 3195–3204 (2019), https://openaccess.thecvf.com/content_CVPR_2019/

html/Marino_OK-VQA_A_Visual_Question_Answering_Benchmark_Requiring_External_Knowledge_CVPR_2019_paper.html
26. Minderer, M., Gritsenko, A., Houlsby, N.: Scaling open-vocabulary object detection. Advances in Neural Information Processing Systems **36** (2024)
27. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
28. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems **35**, 27730–27744 (2022)
29. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html
30. Peng, Z., Wang, W., Dong, L., Hao, Y., Huang, S., Ma, S., Ye, Q., Wei, F.: Grounding multimodal large language models to the world. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=lLmqxkfSIw
31. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
32. Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE transactions on pattern analysis and machine intelligence **44**(3), 1623–1637 (2020)
33. Schwenk, D., Khandelwal, A., Clark, C., Marino, K., Mottaghi, R.: A-okvqa: A benchmark for visual question answering using world knowledge. In: European Conference on Computer Vision. pp. 146–162. Springer (2022)
34. Shen, Y., Song, K., Tan, X., Li, D., Lu, W., Zhuang, Y.: Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. Advances in Neural Information Processing Systems **36** (2024)
35. Stanić, A., Caelles, S., Tschannen, M.: Towards truly zero-shot compositional visual reasoning with LLMs as programmers. Transactions on Machine Learning Research (2024), https://openreview.net/forum?id=WYGiqSVstK
36. Su, Y., Lan, T., Li, H., Xu, J., Wang, Y., Cai, D.: Pandagpt: One model to instruction-follow them all. arXiv preprint arXiv:2305.16355 (2023)
37. Subramanian, S., Merrill, W., Darrell, T., Gardner, M., Singh, S., Rohrbach, A.: 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). pp. 5198–5215 (May 2022). https://doi.org/10.18653/v1/2022.acl-long.357
38. Sur'is, D., Menon, S., Vondrick, C.: Vipergpt: Visual inference via python execution for reasoning. 2023 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 11854–11864 (2023), https://api.semanticscholar.org/CorpusID:257505358
39. Tiong, A.M.H., Li, J., Li, B., Savarese, S., Hoi, S.C.: Plug-and-play VQA: Zero-shot VQA by conjoining large pretrained models with zero training. In: Findings

of the Association for Computational Linguistics: EMNLP 2022. pp. 951–967 (Dec 2022). https://doi.org/10.18653/v1/2022.findings-emnlp.67
40. Villalobos, P., Sevilla, J., Heim, L., Besiroglu, T., Hobbhahn, M., Ho, A.: Will we run out of data? an analysis of the limits of scaling datasets in machine learning. arXiv preprint arXiv:2211.04325 (2022)
41. Wang, W., Chen, Z., Chen, X., Wu, J., Zhu, X., Zeng, G., Luo, P., Lu, T., Zhou, J., Qiao, Y., et al.: Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. Advances in Neural Information Processing Systems **36** (2024)
42. Wu, C., Yin, S., Qi, W., Wang, X., Tang, Z., Duan, N.: Visual chatgpt: Talking, drawing and editing with visual foundation models. arXiv preprint arXiv:2303.04671 (2023)
43. Wu, S., Fei, H., Qu, L., Ji, W., Chua, T.S.: Next-gpt: Any-to-any multimodal llm. arXiv preprint arXiv:2309.05519 (2023)
44. Yang, Z., Gan, Z., Wang, J., Hu, X., Lu, Y., Liu, Z., Wang, L.: An empirical study of gpt-3 for few-shot knowledge-based vqa. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 3081–3089 (2022)
45. Yang, Z., Li, L., Wang, J., Lin, K., Azarnasab, E., Ahmed, F., Liu, Z., Liu, C., Zeng, M., Wang, L.: Mm-react: Prompting chatgpt for multimodal reasoning and action. arXiv preprint arXiv:2303.11381 (2023)
46. You, H., Sun, R., Wang, Z., Chen, L., Wang, G., Ayyubi, H.A., Chang, K.W., Chang, S.F.: Idealgpt: Iteratively decomposing vision and language reasoning via large language models. arXiv preprint arXiv:2305.14985 (2023)
47. Yu, L., Poirson, P., Yang, S., Berg, A.C., Berg, T.L.: Modeling context in referring expressions. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14. pp. 69–85. Springer (2016)
48. Yu, L., Poirson, P., Yang, S., Berg, A.C., Berg, T.L.: Modeling context in referring expressions. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14. pp. 69–85. Springer (2016)
49. Yuksekgonul, M., Bianchi, F., Kalluri, P., Jurafsky, D., Zou, J.: When and why vision-language models behave like bags-of-words, and what to do about it? In: The Eleventh International Conference on Learning Representations (2022)
50. Zellers, R., Bisk, Y., Farhadi, A., Choi, Y.: From recognition to cognition: Visual commonsense reasoning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 6720–6731 (2019)
51. Zeng, Y., Zhang, X., Li, H.: Multi-grained vision language pre-training: Aligning texts with visual concepts. In: International Conference on Machine Learning. pp. 25994–26009. PMLR (2022)
52. Zhang, H., Li, X., Bing, L.: Video-llama: An instruction-tuned audio-visual language model for video understanding. arXiv preprint arXiv:2306.02858 (2023)
53. Zhu, D., Chen, J., Shen, X., Li, X., Elhoseiny, M.: Minigpt-4: Enhancing vision-language understanding with advanced large language models. arXiv preprint arXiv:2304.10592 (2023)