Towards Vision-LanguageFoundationModels: Limitations,Improvements, and Generalization

Simon Zhai

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-9
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-9.html

March 6, 2025

Towards Vision-Language Foundation Models: Limitations, Improvements, and Generalization

by

Yuexiang Zhai

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Yi Ma, Co-chair
Professor Jiantao Jiao, Co-chair
Professor Song Mei

Spring 2025

Towards Vision-Language Foundation Models: Limitations, Improvements, and Generalization

Abstract

Towards Vision-Language Foundation Models: Limitations, Improvements, and Generalization

by

Yuexiang Zhai

Doctor of Philosophy in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Yi Ma, Co-chair

Professor Jiantao Jiao, Co-chair

This dissertation investigates critical challenges in the development and training of multimodal foundation models, focusing on limitations in current supervised fine-tuning (SFT) approaches and exploring the potential of reinforcement learning (RL) to achieve robust generalization. The work is presented in two main parts:

Part 1: Understanding Limitations of Multimodal Foundation Models under Supervised Fine-Tuning

Despite their impressive capabilities on benchmark tasks, multimodal large language models (MLLMs) often exhibit surprising weaknesses when faced with seemingly simple tasks requiring deeper understanding or adaptation to novel situations. This dissertation first investigates the phenomenon of catastrophic forgetting in MLLMs, where fine-tuning on new tasks can lead to a significant decline in performance on previously learned tasks. We introduce the Evaluating MulTimodality (EMT) framework, a novel evaluation methodology designed to systematically assess this forgetting. Our findings reveal that even MLLMs leveraging powerful pre-trained vision encoders suffer from substantial performance degradation on basic image classification tasks after SFT. Furthermore, we delve into the specific visual shortcomings of MLLMs. We introduce the MultiModal Visual Patterns (MMVP) benchmark, a carefully curated set of visual question-answering tasks designed to probe the visual grounding capabilities of these models. The results demonstrate systematic failures in state-of-the-art MLLMs, highlighting a strong correlation between weaknesses in the underlying visual encoder (CLIP) and overall model performance. These findings suggest that current SFT approaches, while effective for task-specific adaptation, may not be sufficient for imbuing MLLMs with robust visual understanding and the ability to retain previously acquired knowledge.

Part 2: Enhancing Generalization through Reinforcement Learning

Recognizing the limitations of SFT, this dissertation then explores the potential of reinforcement learning (RL) to achieve more robust and generalizable multimodal intelligence. We propose a novel framework for fine-tuning large vision-language models (VLMs) with RL, enabling end-to-end training on tasks that require both visual understanding and language reasoning. A key component of this framework is the incorporation of chain-of-thought (CoT) prompting, which leverages the inherent reasoning capabilities of VLMs to facilitate more efficient exploration and learning. We conduct a comparative analysis of RL and SFT, focusing on generalization to unseen rule variations and novel visual contexts. The results demonstrate that RL fine-tuning consistently leads to superior generalization compared to SFT. Models trained with RL exhibit improved performance on tasks with modified rules, adapt more effectively to variations in visual input, and even show enhanced underlying visual recognition abilities. Furthermore, we investigate the role of inference-time computation, demonstrating that increasing the number of verification iterations during RL training further improves generalization. This highlights that while SFT provides a necessary foundation for instruction following, RL is crucial for achieving robust, adaptable performance in complex, dynamic environments.

In summary, this dissertation provides compelling evidence for the limitations of current SFT-based training of multimodal foundation models and showcases the significant potential of RL to overcome these limitations, paving the way for more generalizable and intelligent AI systems.

To my wife and my parents.

Contents

List of Figures

List of Tables

Acknowledgments

This dissertation represents 7 years of my research at Berkeley (2 years as a "pre-doc" RA + 5 years as a PhD). I would like to thank everyone who added any flavor to this journey.

I am extremely fortunate to have been mentored by two amazing professors, Prof. Yi Ma and Prof. Sergey Levine, who ended up shaping my research taste into "formulating the toughest question and solving it in the hardest way".[1] First and foremost, I would like to thank Prof. Yi Ma, who supervised me as a "pre-doc" research assistant (a decent way of saying unemployed) for 2 years and later admitted me to Berkeley. Yi always asks the next toughest research question, which creates a bittersweet experience for every project – every time I finish a project, I feel happy for solving a "hard question", but then Yi will start asking the next toughest question. Next, I would like to thank Prof. Sergey Levine, for giving me a chance to work in one of the best RL research groups. Working with Sergey is like working with a magician with an arsenal – who always has some magic to turn my unpolished paper drafts into glowing writing, while also being able to provide me with weapons to solve the problems I have. Sergey's insight and technicality always inspire me to think about the most fundamental version of a research question, without dodging the hardest part. Hence, my research ended up tackling the toughest problems in the hardest way.

In addition to Yi and Sergey, I would also like to thank Prof. Alane Suhr, and Prof. Song Mei and Prof. Jiantao Jiao, for serving as my qualification (and dissertation) committee members, who inspired me to shape my research path as a mature researcher. Of course, RL researchers always care about trajectories. I was also very fortunate to learn from Prof. Hanwang Zhang, Prof. Shih-Fu Chang, Prof. Qing Qu, and Prof. John Wright before coming to Berkeley from Columbia University. I was also very lucky to have Dr. Samuel Ainsworth as my intern mentor, who encouraged me to get out of my comfort zone and follow my heart.

Throughout my years at Berkeley, I have also been very fortunate to learn from these collaborators (in chronological order): Yichao Zhou, Haozhi Qi, Qi Sun, Zhili Chen, Li-Yi Wei, Xiao Li (CUHK-SZ), Yuqian Zhang, Zhihui Zhu, Bai Jiang, Hao Chen, Hermish Mehta, Zhengyuan Zhou, Zhenyu Liao, John Wright, Sheng Liu, Xiao Li (UMich), Chong You, Carlos Fernandez-Granda, Jiantao Jiao, Christina Baek, Abhishek Gupta, Aldo Pacchiano, Sham Kakade, Qiyang (Colin) Li, Jianlan Luo, Perry Dong, Yaodong Yu, Sam Buchanan, Druv Pai, Ziyang Wu, Shengbang (Peter) Tong, Hao Bai, Benjamin Haeffele, Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Kelvin Xu, Mitsuhiko Nakamoto, Anikait Singh, Max Sobol Mark, Chelsea Finn, Zipeng (Leo) Lin, Xili Dai, Ke Chen, Jingyuan Zhang, Xingjian Gao, Mingyang Li, Heung-Yeung Shum, Mu Cai, Yong Jae Lee, Zhuang Liu, Yann LeCun, Saining Xie, Jiayi Pan, Yifei Zhou, Tianzhe Chu, Jihan Yang, Dale Schuurmans, Quoc V. Le.

I would also like to thank these professors at Berkeley who I have always been looking up to: Prof. Trevor Darrell, Prof. Jitendra Malik, Prof. Benjamin Recht, Prof. Martin

[1]Not sure whether I should be proud of this.

Wainwright. Although I started my PhD in 2020 (as the only COVID generation where orientation and everything happened on Zoom), having a short period of time talking to these great professors did indeed help me understand myself better. Outside of Berkeley, I have also learned from these Professors: Ju Sun, Yuejie Chi, Beidi Chen, Yuxin Chen, Shih-Fu Chang, Carl Vondrick from their unique career paths and research passions.

I also want to thank Yi's group members: Yichao Zhou, Haozhi Qi, Chong You, Yaodong Yu, Michael Psenka, Brent Yi, Druv Pai, Chun-Hsiao (Daniel) Yeh, Ziyang Wu, Zitong Yang, Christina Baek, Kwan Ho Ryan Chan, Tianzhe Chu, and Feng Chen, for making the weekly meetings fun and entertaining.

At RAIL, I am also very honored to learn from my amazing labmates: Amy Zhang, Yi Su, Ilya Kostrikov, Kuan Fang, Jianlan Luo, Oier Mees, Karl Pertsch, Oleh Rybkin, Masatoshi Uehara, Xinyang Geng, Michael Janner, Michael Chang, Aviral Kumar, Dhruv Shah, Kevin Black, Dibya Ghosh, Jakub Grudzien, Katie Kang, Vivek Myers, Seohong Park, Amrith Setlur, Kyle Stachowicz, Laura Smith, Homer Walke, Pranav Atreya, James Bowden, William Chen, Kevin Frans, Catherine Glossop, Zhiyuan (Paul) Zhou, Philip Ball.

I would also like to thank my research friends/mentors at BAIR and other places: Banghua Zhu, Tom Hu, Jasper Zhang, Jason Chou, Fangchen Liu, Hao Liu, Toru Lin, Ruiqi Zhang, Andrea Zanette, Tianhe (Kevin) Yu, Xudong Wang, Boyi Li, Wenlong Mou, Reese Pathak, Ruiqi Zhou, Ruiqi Zhang, Zhi-Yang (Jerry) He, Alex Pan, Fred Zhang, Ellis Brown, Haotian Liu, Tengyang Xie, Yong Lin, Hanjun Dai, Jincheng Mei, Bo Dai, Kuang-Huei Lee, Sherry Yang, Dale Schuurmans, Quoc V. Le, for sharing different research perspectives and constructive feedback on my research.

Coming as someone who did not know how to code in undergrad, I would like to specially thank Haozhi Qi for teaching me how to use Python, Qiyang (Colin) Li for teaching me how to run deep RL codes, and Haotian Liu & Mu Cai for teaching me how to set up multimodal model training scripts. As an amateur comedian, I would like to thank Druv Pai and Kyle Stachowicz for bearing with me and the "not-so-fun" jokes I randomly shared with them for collecting human feedback.

Having learned from many senior researchers, I also have the privilege of working with the amazing undergrads at Berkeley. Among all the undergrads I have worked with, I want to specifically thank Zipeng (Leo) Lin, who let me feel the happiness of witnessing the growth of a young researcher. I would also like to thank other amazing collaborators: Zitong Yang, Christina Baek, Perry Dong, Shengbang (Peter) Tong, and Tianzhe Chu for showing me that there are always younger people who are much smarter than me.

Beyond research, I would like to thank my friends Yaohong Wu, Xiyu Ouyang, Deyi Liu, Yuchen Cao, Miaoding Dai, Mengchen Shi, Fangzhu Zhao, Yuchen Jin, Jingling Li, Denny Wu, Ci Chen and Xinru Yang, for showing me another possibility of life without doing a PhD.

I would also like to thank my parents, Ming Zhai and Hongying Jiang, who raised a boy until 22, and then allowed their boy to be abroad for more than 8 years without coming back home for a single time. I would also like to thank my cousin Zhongjian Jiang and Wei Li, for taking me around the Bay Area for good food.

Most importantly, I would like to thank my wife, Xiaoxuan (Iris) Feng, who inspired me to pursue a PhD, encouraged me to persist during the countless down moments, and witnessed the end of this journey. Iris has been my strongest mental and financial support throughout the entire PhD journey, and she is the real hero behind this dissertation.

Chapter 1

Introduction

1.1 The Promise and Perils of Multimodal Foundation Models

This dissertation explores the rapidly evolving landscape of multimodal foundation models, with a particular emphasis on leveraging domain knowledge to enhance their performance and generalization. The recent surge in the capabilities of large language models (LLMs), and their subsequent extension to encompass multiple modalities (creating MLLMs or VLMs), has been nothing short of revolutionary. These models, trained on vast quantities of text and image data, exhibit impressive proficiency in tasks that demand both linguistic and visual understanding. They can answer complex questions about images, generate descriptive captions, and even engage in sophisticated reasoning that bridges the gap between language and perception. However, beneath this impressive facade lie fundamental questions about the depth of their understanding and their ability to adapt to new situations. While these models excel at benchmark tasks, do they truly grasp the underlying concepts and principles, or are they merely sophisticated pattern-matchers, optimized for specific datasets and training paradigms?

This research is driven by the observation that even state-of-the-art multimodal foundation models can falter on surprisingly basic tasks. They might struggle with visual details that a human would find trivial, or fail to apply learned rules in slightly altered contexts. These shortcomings suggest that the prevalent pre-training and supervised fine-tuning (SFT) approaches, while powerful, may not be sufficient to equip these models with the robust domain knowledge required for genuine, adaptable intelligence. Reinforcement learning (RL), with its emphasis on learning through interaction and feedback, offers a promising alternative methodology. This dissertation argues that RL can unlock the full potential of multimodal models, pushing them beyond memorization and towards true generalization.

1.2 Key Research Questions and Contributions

This dissertation tackles several core research questions at the intersection of multimodal learning and domain knowledge. First, it investigates the limitations of current SFT methods when applied to MLLMs. Specifically, it examines the phenomenon of catastrophic forgetting, where fine-tuning on new tasks can lead to a significant decline in performance on previously learned tasks. Second, it delves into the specific visual shortcomings of MLLMs, exploring whether these weaknesses originate from limitations in the visual encoders themselves. Third, the dissertation explores the feasibility and effectiveness of adapting reinforcement learning (RL) techniques to train large multimodal models end-to-end. This involves developing a framework that allows these models to interact with environments, receive feedback, and learn through trial and error. Finally, it investigates whether RL fine-tuning leads to superior generalization compared to SFT, particularly when faced with novel rule variations and unseen visual contexts.

The contributions of this work are multifaceted. A novel evaluation framework, Evaluating MulTimodality (EMT), is introduced to systematically assess catastrophic forgetting in MLLMs. The MultiModal Visual Patterns (MMVP) benchmark is developed to probe the visual grounding capabilities of these models, revealing systematic failures and highlighting the link between visual encoder weaknesses and overall model performance. A significant contribution is the development of a novel framework for fine-tuning VLMs with RL, incorporating chain-of-thought (CoT) prompting to facilitate reasoning and exploration. Crucially, this dissertation provides compelling empirical evidence that RL fine-tuning leads to superior generalization compared to SFT, enabling models to adapt to new rules and visual variations more effectively.

1.3 Dissertation Outline

The structure of this dissertation is as follows:

- Chapter 2: Investigating the Forgetting in Multimodal Model Fine-tuning: This chapter explores the phenomenon of catastrophic forgetting in MLLMs, introducing the EMT framework and analyzing the performance degradation of fine-tuned models on basic image classification.

- Chapter 3: Understanding the Visual Short Coming in Multimodal Models: This chapter presents the MMVP benchmark and delves into the systematic visual failures of MLLMs, connecting these issues to limitations in the underlying visual encoders.

- Chapter 4: Adapting RL to Foundation Model Training: This chapter introduces the novel framework for fine-tuning VLMs with RL, explaining the key components and demonstrating its effectiveness on various decision-making tasks.

- Chapter 5: Understanding Foundation Model Post-Training: This chapter offers a comparative analysis of RL and SFT, showcasing the superior generalization capabilities of RL in handling both textual rule-based variations and novel visual contexts.
- Chapter 6: Conclusion, Discussion, and Future Directions: This chapter summarizes the key findings and contributions, discusses the limitations of the current work, and proposes potential avenues for future research.

1.4 Concluding Remarks

Ultimately, this dissertation seeks to contribute to the development of more robust, reliable, and adaptable multimodal AI systems. By understanding the limitations of current training paradigms and exploring the potential of reinforcement learning, we aim to pave the way for models that can not only perform well on specific benchmarks but also exhibit true understanding and generalize effectively to the complexities of the real world.

Part I

Understanding the Limitations in Vision-language Foundation Models

Chapter 2

Investigating the Forgetting in Multimodal Model Fine-tuning

2.1 Introduction

The recent progress in language models (LMs) has demonstrated impressive competency in engaging in a natural dialogue and in complex examinations [Devlin et al., 2018, Brown et al., 2020, OpenAI, 2022, 2023d]. Besides text generation, GPT4 [OpenAI, 2023b] has recently shown impressive multimodal capability by performing a range of tasks with visual and language inputs. The emergent multimodal reasoning capabilities of GPT4 have propelled a surge of interest in multimodal large language models (MLLMs) [Li et al., 2023c, Liu et al., 2023d, Zhu et al., 2023a, Li et al., 2023a, Dai et al., 2023]. This line of research typically involves (1) integrating pre-trained vision encoders [Radford et al., 2021, Ilharco et al., 2021] with open-source LLMs [Chung et al., 2022, Touvron et al., 2023a,b], and (2) applying instruction tuning on the resulting vision-language models [Dai et al., 2023, Liu et al., 2023d, Li et al., 2023a].

While many of these fine-tuned MLLMs have demonstrated remarkable capabilities in general purpose vision-language comprehension [Yin et al., 2023, Fu et al., 2023a], these models still suffer from *catastrophic forgetting* [Chen et al., 2020a, Dong et al., 2021, Mosbach et al., 2021, Korbak et al., 2022]. That is, the models tend to overfit to the fine-tuning dataset and consequently experience a decline in performance on pre-training tasks. Catastrophic forgetting in image classification has been extensively studied in computer vision and machine learning [Goodfellow et al., 2013, Yang et al., 2023b]. However, recent developments in MLLMs [Li et al., 2023c, Liu et al., 2023d, Zhu et al., 2023a, Li et al., 2023a, Dai et al., 2023] have mainly focused on creating multimodal chatbots for visual question answering [Antol et al., 2015], without evaluating their fundamental image classification capabilities, let alone explore the catastrophic forgetting in MLLM. That being said, prior MLLM evaluation frameworks [Li et al., 2023d, Fu et al., 2023a] mainly focus on assessing cognitive reasoning capability or hallucinations, which overlooks the necessity to critically examine how well MLLMs inherit the image classification capability from their base vision encoders [Radford et al., 2021, Ilharco et al., 2021].

To comprehensively investigate the catastrophic forgetting in fine-tuned MLLM, we present the **E**valuating **M**ul**T**imodality (EMT) framework, which, to the best of our knowledge, is the *first* evaluation framework that studies the catastrophic forgetting in MLLMs. The EMT framework is a two-stage approach that treats each MLLM as an image classifier. In particular, for an input text and image pair, EMT first prompts the testing MLLM by asking it to classify the input image, and then post-processes the outputs to obtain a classification accuracy.

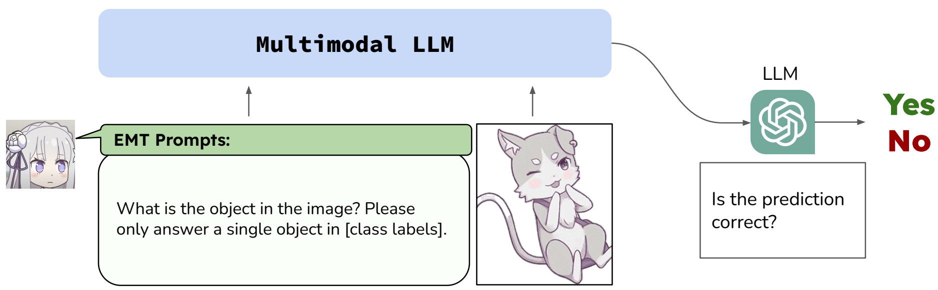


Figure 2.1: The EMT evaluation pipeline for MLLM. We prompt each MLLM as an image classifier by (1) inputting an image from a classification task; (2) asking the MLLM to explicitly answer a single label from the classification task. We evaluate the correctness of each output using another LLM [Zhai et al., 2024b].

We first apply EMT to several open-source fine-tuned MLLMs [Li et al., 2023a, Liu et al., 2023d, Berrios et al., 2023, Dai et al., 2023] and observe a severe catastrophic forgetting phenomenon among *all* the tested models. That is, the majority of the tested MLLMs *fail to retain a comparable classification accuracy when compared to the zero-shot performance of their vision encoders*. After analyzing the results from the tested open-source models, we identify hallucination Zhu et al. [2023a], Li et al. [2023d], Ji et al. [2023], Gudibande et al. [2023] as one the major factors contributing to the performance degradation in MLLMs. Specifically, the tested MLLMs hallucinate by generating additional outputs that are irrelevant to the input question, including outputting more than one label or generating unverifiable descriptions of a label.

To gain deeper insights into how fine-tuning impacts the performance of MLLMs, we continue to fine-tune LLaVA Liu et al. [2023d], a popular MLLM achieving state-of-the-art accuracy on Science QA Lu et al. [2022], and then apply the EMT evaluation to the fine-tuned LLaVA. Our fine-tuning experiments reveal two main observations. *Initially, fine-tuning on one dataset demonstrates generalization to other datasets*, as it improves the alignment between textual and visual features. However, *as the fine-tuning progresses, LLaVA starts to hallucinate* by disregarding the questions and exclusively generating text based on the examples in the fine-tuning datasets.

To summarize, this paper makes two key contributions.

- We propose EMT, an evaluation framework designed specifically to evaluate the phenomenon of catastrophic forgetting in MLLMs. To the best of our knowledge, EMT is the first evaluation framework to investigate catastrophic forgetting in MLLM through classification. Through EMT, we discover that nearly all the tested models fail to retain the classification performance of their vision encoders.

- We conduct fine-tuning experiments on LLaVA. Our fine-tuning results indicate that while moderate fine-tuning is advantageous for non-fine-tuned tasks, excessive fine-tuning ultimately leads to catastrophic forgetting in these tasks.

Our findings suggest that the fine-tuning process of MLLMs can still be further improved, particularly in mitigating catastrophic forgetting and reducing hallucinations.

2.2 Related Works

Fine-Tuning and Catastrophic Forgetting. Fine-tuning large pre-trained models has significantly transformed the field of natural language processing [Devlin et al., 2018, Radford et al., 2018, 2019, Brown et al., 2020, Lin et al., 2023]. Despite its ubiquity and remarkable achievements, fine-tuning LLM still suffers from core machine learning problems such as catastrophic forgetting [McCloskey and Cohen, 1989]. Catastrophic forgetting widely appears in LLM fine-tuning [Howard and Ruder, 2018, Lee et al., 2020, Dong et al., 2021, Zhang et al., 2021b, Korbak et al., 2022] or in-context learning [Alayrac et al., 2022, Wang et al., 2023a], as the LLMs tend to overfit to the small fine-tuning dataset resulting in losing performance on other tasks [Howard and Ruder, 2018]. Various approaches have been proposed to mitigate the catastrophic forgetting problem in LLM fine-tuning, including pre-trained weight decay [Zhang et al., 2021b], learning rate decay [Howard and Ruder, 2018], regularizations [Lee et al., 2020], and adversarial fine-tuning [Dong et al., 2021]. However, in MLLM, such a catastrophic forgetting phenomenon has not been thoroughly studied yet. Our work is most related to several evaluation metrics for MLLMs [Fu et al., 2023a, Li et al., 2023d], which proposed a comprehensive framework for evaluating the perception and recognition [Fu et al., 2023a] or hallucinations [Li et al., 2023d], while the proposed EMT specifically aims at evaluating the catastrophic forgetting in MLLMs.

Multimodal Large Language Models. Multimodal Large Language Models (MLLMs) have emerged as a significant advancement in vision-language models, which significantly improves the model's reasoning capability. These models are designed to process and interpret information from multiple modalities, such as text and images, to perform complex tasks that require a comprehensive understanding of the context. Recent works [Li et al., 2023c, Dai et al., 2023, Li et al., 2023a, Berrios et al., 2023, Liu et al., 2023d, Zhu et al., 2023a, Awadalla et al., 2023, Zhang et al., 2023c, Huang et al., 2023b, Cai et al., 2023] have contributed to the development and enhancement of MLLMs by leveraging the strong reasoning capability of LLMs such as LLaMA [Touvron et al., 2023a,b]. LLaVA [Liu et al., 2023d], as presented in the paper under discussion, represents a novel approach to instruction tuning on machine-generated multimodal language-image instruction-following data, achieving impressive multimodal chat abilities and state-of-the-art accuracy on Science QA [Lu et al., 2022]. Following the instruction tuning approach, various works came out focusing on other modalities such as video [Zhang et al., 2023b] and point cloud [Hong et al., 2023b]. See Yin et al. [2023] for a more comprehensive overview of the current state and future directions of MLLMs.

A Theoretical Perspective of Catastrophic Forgetting through Minority Collapse. Recently, Yang et al. [2023b] introduced an approach to address the issue of catastrophic forgetting, drawing inspiration from the principles of Neural Collapse (NC) [Papyan et al., 2020, Zhu et al., 2021, Fang et al., 2021, Thrampoulidis et al., 2022, Behnia et al., 2022, Zhong et al., 2023]. In particular, Fang et al. [2021] proposes *minority collapse* as a subsequent research direction of NC. Minority collapse describes a phenomenon in supervised learning with imbalanced data, where the classifiers of the minority classes converge to one vertex when the sample size ratio between the majority and minority classes reaches infinity. This result implies that all minority classes are indistinguishable when the imbalance ratio reaches infinity. To connect the fine-tuning with minority collapse: (1) Treating the absent class in fine-tuning as minority classes with a sample size of zero, directly implies the imbalanced training scenarios with a ratio of infinity; (2) Such an imbalance training in the fine-tuning phase will make the classifiers of the pre-trained classes converges to one vertex [Fang et al., 2021]; (3) Hence, the pre-trained classes become indistinguishable during fine-tuning, which results in catastrophic forgetting.

2.3 Fine-Tuning Image Classification

To verify the theoretical results inspired by minority collapse [Fang et al., 2021, Thrampoulidis et al., 2022], where supervised fine-tuning leads to catastrophic forgetting, we first perform pre-training and fine-tuning of image classification via ResNet [He et al., 2016]. Next, to further investigate the catastrophic forgetting in the vision-language model, we conduct experiments in fine-tuning the Contrastive Language-Image Pre-Training network (CLIP) [Radford et al., 2021].

Pre-Training and Fine-Tuning for Image Classification

To initiate our investigation, we train ResNet18 [He et al., 2016] on conventional image classification benchmarks. In particular, we first pre-train using the initial 50% of classes for 100 epochs. Then, we fine-tune with the remaining 50% of classes for 100 epochs, so that the fine-tuning classes and the pre-training classes do not overlap. Since the NC theory [Papyan et al., 2020, Zhu et al., 2021] mainly focuses on analyzing the training loss, we only present the average training accuracy for the first 50% pre-trained classes (See Figure 2.2). Notably, when the fine-tuning starts, the training accuracy of pre-trained classes rapidly diminishes to zero across all datasets. As discussed in previous sections, such a catastrophic forgetting phenomenon can be directly associated with minority collapse, where the classifiers of all minority classes converge to a single vertex, when the imbalance ratio between majority and minority classes approaches infinity. Therefore, the observed decline in performance is in line with our expectations. For completeness, we provide *the theoretical formulation of minority collapse* of fine-tuning in Appendix A.1 and implementation details in Appendix A.2.

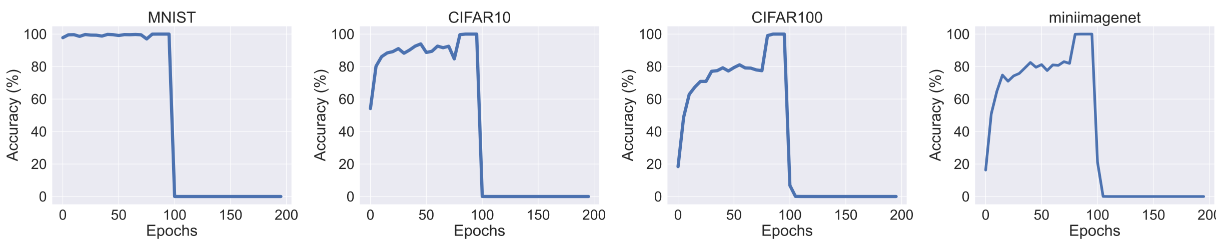


Figure 2.2: **Catastrophic forgetting happens in traditional classification tasks.** To corroborate the NC theory [Papyan et al., 2020, Zhu et al., 2021, Fang et al., 2021, Thrampoulidis et al., 2022], we only plot the average *training accuracy* of the first 50% classes of MNIST, CIFAR-10, CIFAR-100, and *mini*Imagenet, respectively.

Fine-Tuning Contrastive Language-Image Pre-Training Network

We then fine-tune the vision encoder from the CLIP ViT-L-14 model [Radford et al., 2021], starting from a checkpoint provided by OpenAI's CLIP, available through openCLIP [Ilharco et al., 2021]. In our experiments, we employ the standard cross-entropy loss, consistent with the approach used in CLIP pre-training and the analysis in Neural Collapse [Papyan et al., 2020, Zhu et al., 2021] as well as minority collapse [Fang et al., 2021]. Text inputs are created by concatenating labels with short descriptions. See examples in Appendix A.2.

Empirical results demonstrate that vision-language models like CLIP are susceptible to neural collapse after fine-tuning. In particular, we observe a significant rise in the in-domain performance, while the out-of-domain dataset performance begins to decline. By the time we reach 15 epochs, nearly all in-domain performance metrics have escalated to close to 99%, but the out-of-domain performance has suffered.

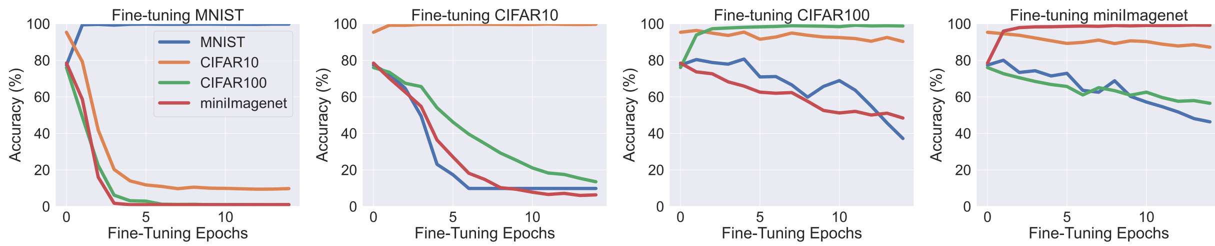


Figure 2.3: Accuracy of 1 to 14 epoch fine-tuned CLIP on MNIST, CIFAR-10, CIFAR-100, and *mini*Imagenet. Detailed accuracy numbers are presented in Table A.10 of Appendix A.2.

2.4 EMT: Evaluating Multimodal Large Language Models

Since prior MLLM evaluation frameworks [Fu et al., 2023a, Li et al., 2023d] focus on assessing cognitive reasoning [Fu et al., 2023a] or hallucinations [Li et al., 2023d] rather than the catastrophic forgetting from an image classification perspective, we propose EMT, a framework for **E**valuating **M**ul**T**imodal LLM. EMT works as follows: (1) We start by inputting an image from a classification task; (2) Then we prompt the testing MLLM by asking it to classify the input images and collect its outputs via the prompt provided below, according to each dataset. (3) Next, since the output from MLLMs may not adhere to a specific format, we apply `GPT-3.5` to evaluate the classification accuracy;[1] (4) Finally, we output the prediction accuracy of the testing MLLM on different datasets.

EMT Prompt:

```
What is the number/object in the image?  Please only answer a single
number/object in [class labels].
```

The detailed prompts for predictions and evaluations for each dataset are provided in Appendix A.3.

Catastrophic Forgetting in Open-Source MLLMs

In this subsection, we initially apply EMT to assess four MLLMs: LLaVA [Liu et al., 2023d], Otter [Li et al., 2023a], InstructBLIP [Dai et al., 2023], and LENS [Berrios et al., 2023]. As shown in Figure 2.4, most of the tested open-source MLLMs suffer from catastrophic forgetting by failing to retain a similar classification performance, compared to the zero-shot classification outcome of their respective vision encoders. A notable exception is `InstructBLIP-7b`, which performs slightly better on the CIFAR-10 dataset. `InstructBLIP`

[1] It is a common practice to adopt `openaiAPI` for evaluating the performance of different LMs, e.g., see Rafailov et al. [2023], Gudibande et al. [2023]. See more discussion on other potential evaluation methods in Section 2.7.

slightly performing better than its base vision model, `InstructBLIP` cannot achieve similar performance in CIFAR-100 and *mini*Imagenet, compared to `LLaVA` and `Otter`.[2] It may seem surprising that most of the tested MLLMs fail to retain similar performance of their foundational vision models, but such a performance degradation can be anticipated in hindsight. This performance degradation may stem from the fact that classifications of MNIST, CIFAR-10, CIFAR-100, and *mini*Imagenet are not incorporated into the training dataset of the evaluated MLLMs.[3]

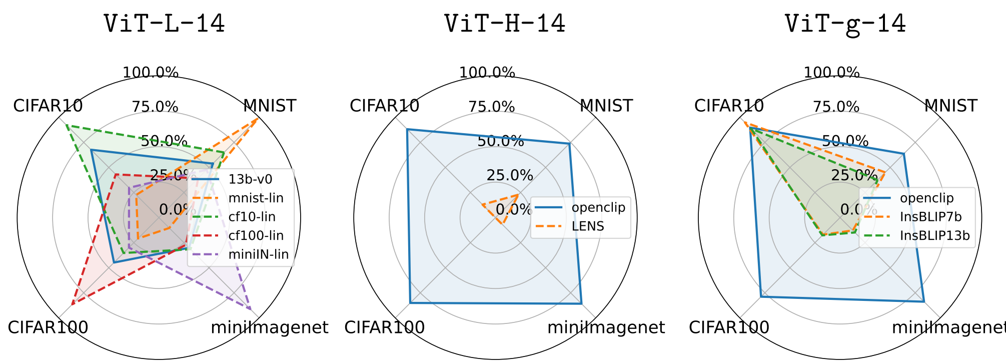


Figure 2.4: EMT evaluation accuracy of different MLLMs on MNIST, CIFAR-10, CIFAR-100, and *mini*Imagenet, against the zero-shot performance of their vision encoders. Models are grouped according to their underlying vision encoder architecture. Detailed accuracy numbers are presented in Table A.2 in Appendix A.3.

Analyzing Failure Modes of MLLMs

After checking the outputs of different models using our EMT prompt, we have identified three major issues causing performance degradation: incorrect prediction, intrinsic hallucination, and extrinsic hallucination. It is evident that MLLMs could produce incorrect predictions, just like classifiers. In the example shown below, `LLaVA-7B` incorrectly predicts "0" as "8" in the MNIST classification.

img	**Label: 0** | `LLaVA-7b`
	`The number in the image is 8:`

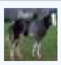
Aside from incorrect prediction, the performance is also significantly impacted by hallucination [Ji et al., 2023, Liu et al., 2023b] – the tested MLLMs sometimes generate seem-

[2]We hypothesize that the performance variations amongst these MLLMs are attributable to differences in their training methodologies. However, the precise causes contributing to the performance discrepancy in these open-source MLLMs are beyond the scope of this research.

[3]For completeness, we leave the detailed discussion of different datasets adopted by each tested MLLMs in Appendix A.3. We also have some examples of the outputs by EMT prompt in Appendix A.3

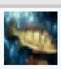
ingly related but incorrect or unverifiable contents. Ji et al. [2023] further characterizes hallucinations into two distinct categories: *intrinsic* and *extrinsic* hallucinations. Intrinsic hallucinations are defined as instances in which the generated output directly contradicts the source content. Extrinsic hallucinations, on the other hand, are those where the output bears no verifiable connection to the original source content.

Intrinsic Hallucination. Our EMT prompt has identified intrinsic hallucinations within the tested MLLMs. One example can be drawn from asking LENS to perform a classification on CIFAR-10:

`img`	**Label: horse** | `LENS`
	`airplane, automobile, bird, cat, deer, dog, frog, horse,`

It is important to note that EMT prompt explicitly instructed the testing MLLM to identify only a single object within all class labels. Despite these clear instructions, LENS still produces an intrinsically hallucinated output - `airplane, automobile, bird, cat, deer, dog, frog, horse,`, an answer that contains multiple labels.

Extrinsic Hallucination. In addition to intrinsic hallucination, we have also discovered extrinsic hallucinations when applying `InstructBLIP` to classify CIFAR-100:

`img`	**Label: aquarium_fish** | `InstructBLIP-7b`
	`a picture of a fish in a tank`

In the example provided above, while the generated output text partially includes the label "aquarium fish", it also exhibits additional descriptors that are not only challenging to verify, but also extraneous to the original request outlined by the prompt.

Base LMs are Important. Among all the tested MLLMs, Figure 2.4 shows that LENS achieves the worst performance, compared to all other models, in each individual task and overall performance. Considering that `ViT-H-14`, the underlying vision encoder of LENS, does not exhibit a significant performance shortfall, we hypothesize that the observed performance gap is attributed to the base LM. This is because Otter, LLaVA, and InstructBLIP all adopt the LLaMA model [Touvron et al., 2023a], while `LENS` uses the Flan-T5 model [Chung et al., 2022], which is less powerful than LLaMA. Nonetheless, our results do not necessarily imply that larger LMs consistently yield superior performance, as our experiments have revealed varying outcomes. For instance, although `LLaVA-13b` generally surpasses `LLaVA-7b`, `InstructBLIP-13b` does not demonstrate superiority over `InstructBLIP-7b`. Therefore, we believe that additional experiments are required to conclusively determine whether larger LMs improve the integration of vision and text data in MLLMs.

2.5 EMT on Multimodal Large Language Models Fine-Tuning

Equipped with EMT, we now investigate the hallucinations in MLLM fine-tuning. We use `LLaVA-7b` and `LLaVA-13b` as our based MLLM for fine-tuning. And we conduct fine-tuning experiments on MNIST, CIFAR-10, CIFAR-100, and *mini*Imagenet, respectively. All of our fine-tuning experiments were conducted based on the LLaVA model released on July 4th, 2023.[4]

Linear and LoRA Fine-Tuning As discussed by Liu et al. [2023d], the LLaVA model contains a frozen base vision encoder $g(\cdot)$ and a pre-trained LLM $f_\phi(\cdot)$ parameterized by ϕ. For an input image $\boldsymbol{X}_{\mathtt{v}}$, LLaVA first maps $\boldsymbol{X}_{\mathtt{v}}$ into a visual feature vector $\boldsymbol{Z}_{\mathtt{v}}$ by applying the visual encoder $\boldsymbol{Z}_{\mathtt{v}} = g(\boldsymbol{X}_{\mathtt{v}})$. Then, LLaVA applies a linear adapted layer $\boldsymbol{W}$, that maps the visual features into text feature spaces $\boldsymbol{H}_{\mathtt{v}} = \boldsymbol{W} \cdot \boldsymbol{Z}_{\mathtt{v}}$, and concatenate $\boldsymbol{H}_{\mathtt{v}}$ with the embedding of language queries $\boldsymbol{H}_{\mathtt{q}}$ into a visual and text embedding vector $[\boldsymbol{H}_{\mathtt{v}}, \boldsymbol{H}_{\mathtt{q}}]$. Finally, LLaVA feeds $[\boldsymbol{H}_{\mathtt{v}}, \boldsymbol{H}_{\mathtt{q}}]$ as the input to the pre-trained LLM $f_\phi(\cdot)$ to obtain responses. As for specific fine-tuning methods: (1) Linear fine-tuning method *only* fine-tunes the linear adapter layer $\boldsymbol{W}$; (2) LoRA fine-tuning method fine-tunes the linear adapter layer $\boldsymbol{W}$ *and* the pre-trained LLM $f_\phi(\cdot)$ with LoRA [Hu et al., 2021].

Experimental Setup and Overview

Given that LLaVA relies on visual and language instruction data for training and fine-tuning processes, we have manually reformatted several datasets, namely MNIST, CIFAR-10, CIFAR-100, and *mini*Imagenet to comply with the required format for fine-tuning. For more detailed information on the format of the fine-tuning data used, as well as the specifics of the LLaVA fine-tuning process, please refer to Appendix A.4. All of our fine-tuning experiments were conducted using 2 Nvidia A100 GPUs. We fine-tune `LLaVA-7b` and `LLaVA-13b` using linear and LoRA [Hu et al., 2021] fine-tuning respectively, due to the limitation of computational resources, we cannot afford to fine-tune the entire LLaMA model. We first report the EMT evaluated accuracy of fine-tuned `LLaVA-7b` and `LLaVA-13b` after 3 epochs of linear and LoRA fine-tuning in Figure 2.5. To assess accuracy variations during training, we then report the EMT evaluation results from 1-3 fine-tuning epochs in Figure 2.6 and 2.7.

Excessive Fine-Tuning Causes Forgetting

We first present the 3-epoch fine-tuning results in Figure 2.5. While LLaVA's performance indeed improves on the fine-tuning dataset, Figure 2.5 unveils a critical issue of MLLM fine-tuning:

[4]See this git commit.

Fine-tuning MLLM on one dataset decreases the performance on another non-fine-tuning dataset.

This phenomenon, though not unexpected, is noteworthy. As the model doesn't have exposure to datasets other than the one it has been fine-tuned on, it stands to reason that a similar effect to catastrophic forgetting would be observed, as discussed previously in Section 2.4.

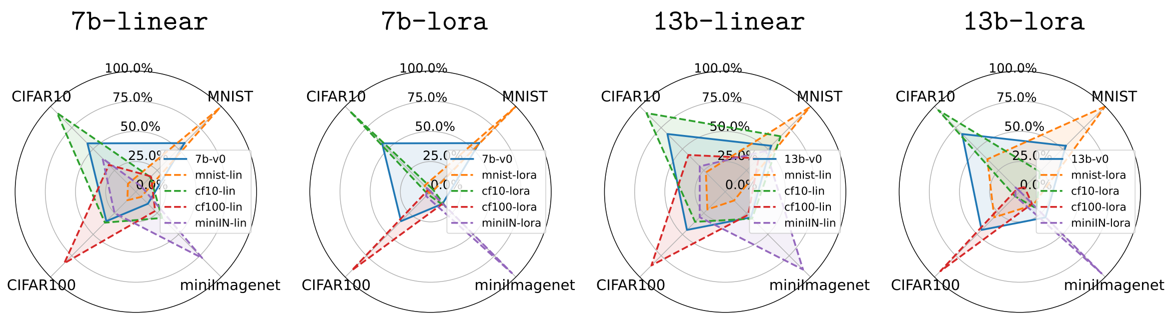


Figure 2.5: EMT evaluation accuracy of 3-epoch fine-tuned `LLaVA-7b` and `LLaVA-13b` on MNIST, CIFAR-10, CIFAR-100, and *mini*Imagenet, against the zero-shot performance of their vision encoders. Detailed accuracy numbers are presented in Table A.9 of Appendix A.4.

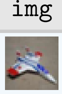
As we examine the output from fine-tuned LLaVA, we discover that

Fine-tuning MLLM causes hallucinations, by outputting texts that are related to its fine-tuned dataset while ignoring the question related to its original prompt.

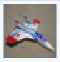
To further illustrate this phenomenon, we provide explicit examples of classifying the `LLaVA-7b` and `LLaVA-13b`, which have been fine-tuned on different datasets using the EMT prompt.

EMT Prompt:

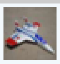
```
What is the object in the image? Please only answer a single object
in airplane, automobile, bird, cat, deer, dog, frog, horse, ship,
truck.
```

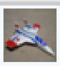
img	**Label: airplane** | `LLaVA-7b-lora-ft-cifar10`
	`The object is an airplane.`

The earlier demonstration illustrates that, when the CIFAR-10 fine-tuned model is tested on CIFAR-10, LLaVA indeed successfully identifies the object. Nevertheless, the LLaVA model begins to hallucinate in CIFAR-10 classifications after being fine-tuned on other datasets.

img	**Label: airplane** | `LLaVA-7b-lora-ft-mnist`
	`The airplane is 8.`

In the preceding example, the classification of CIFAR-10 through an MNIST fine-tuned model, the model not only partially generates the keyword "airplane", but concurrently produces hallucinated outputs by yielding the representation of the number "8". Similar phenomena are also observed in the CIFAR-100 and *mini*Imagenet fine-tuned models. Specifically, these fine-tuned models begin to hallucinate by predicting "airplane" as classes that bear resemblance or are related to an "airplane", such as "butterfly" and "aircraft carrier" in the CIFAR-100 and *mini*Imagenet models, respectively.

img	**Label: airplane** | `LLaVA-7b-lora-ft-cifar100`
	`The object is a(n) butterfly.`

img	**Label: airplane** | `LLaVA-7b-lora-ft-miniimagenet`
	`The object is a(n) aircraft carrier.`

For completeness, we attach additional outputs of different fine-tuned LLaVA models in Appendix A.4 for further reference.

Moderate Fine-Tuning is Beneficial

In the preceding subsection, we have demonstrated that 3-epoch fine-tuned LLaVA achieves superior performance on the fine-tuned dataset, at the expense of generating hallucinated texts when tested on other datasets. However, this outcome does not necessarily imply that fine-tuning undermines the performance. Notably, we actually observe performance improvement on non-fine-tuned datasets. For instance, as shown in Figure 2.5, `LLaVA-7b` exhibits improved performance on *mini*Imagenet after 3 epochs of fine-tuning on CIFAR-10. To better understand the generalizability in fine-tuning, we conduct fine-tuning experiments on all four datasets for 3 epochs and report their accuracy at each epoch.

Fine-Tuning Adapter Improves Feature Alignments. As illustrated in Figure 2.6, we observe that the linear fine-tuned LLaVA achieves generalization performance upon being fine-tuned on RGB datasets, namely, CIFAR-10, CIFAR-100, and *mini*Imagenet. Given that linear fine-tuning only affects the linear projection layer connecting visual features to the text embedding space, Figure 2.6 implies that early-stage fine-tuning contributes to the enhancement of alignment between visual and textual features. However, in subsequent fine-tuning epochs (2-3), LLaVA starts to overfit the fine-tuning dataset by generating hallucinated texts.

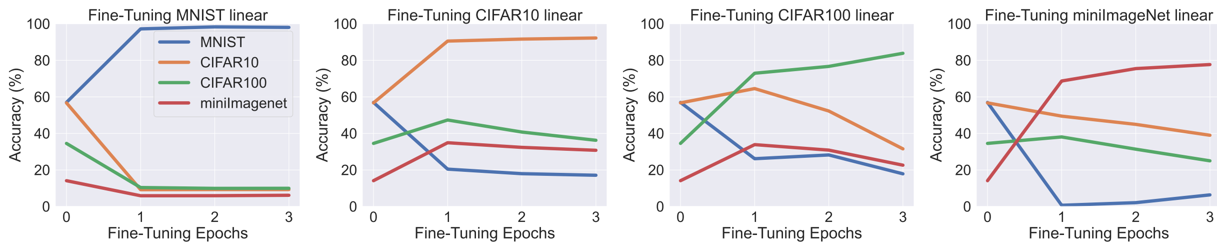


Figure 2.6: EMT evaluation accuracy of 1-3 epoch linear fine-tuned `LLaVA-7b` on MNIST, CIFAR-10, CIFAR-100, and *mini*Imagenet. Detailed accuracy numbers are presented in Table A.10.

Fine-Tuning LLM and Adapter Causes Hallucinations. Contrary to the linear fine-tuning, Figure 2.7 implies that jointly fine-tuning both the LLM and the linear adapter directly causes overfitting on the fine-tuning dataset. This is evidenced by the significant degradation in the LoRA fine-tuned model's performance on the non-fine-tuning datasets after just a single epoch of training.

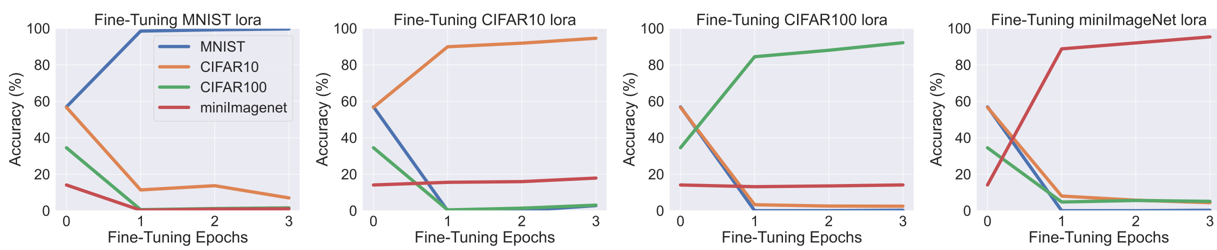


Figure 2.7: EMT evaluation accuracy of 1-3 epoch LoRA fine-tuned `LLaVA-7b` on MNIST, CIFAR-10, CIFAR-100, and *mini*Imagenet. Detailed accuracy numbers are presented in Table A.11.

2.6 Conclusions

In this project, we have studied how fine-tuning affects catastrophic forgetting in MLLMs. To quantitatively evaluate this issue, we propose EMT, a framework for evaluating the fine-tuning performance of MLLMs. We then conduct extensive experiments in fine-tuning LLaVA, an MLLM, and apply EMT to evaluate the performance of different fine-tuned LLaVA models. We have discovered that: (1) Almost all the open-source MLLMs tested in this paper fail to achieve a similar level of accuracy, compared to the zero-shot performance of their base vision encoder; (2) After excessive fine-tuning on one dataset, LLaVA's performance on non-fine-tuning datasets deteriorate as it starts to overfit and hallucinate; (3) Moderate fine-tuning actually improves the performance of LLaVA on similar tasks, as fine-tuning helps visual and text feature alignment in the early-stage.

2.7 Discussions and Future Work

Dataset Diversity is Important for Fine-Tuning. Figure 2.6 shows that LLaVA fine-tuned on CIFAR-10, CIFAR-100, and *mini*Imagenet for one epoch, could generalize to the other two datasets, while fine-tuning LLaVA on MNIST leads to performance degradation on all remaining datasets. This observation implies that having a diverse fine-tuning dataset is important. This is because a more diverse dataset will have features of more modes, hence making the fine-tuned MLLMs suffer less from catastrophic forgetting.

Catastrophic Forgetting Beyond Image Classifications. As a starting point, we only study the catastrophic forgetting in MLLM from the image classification perspective, since it is a standard classification problem. In the future, we believe similar evaluation methods can be developed for other scenarios, such as reducing bias towards potentially unsafe outputs [Awadalla et al., 2023], degrading visual localization reasoning capabilities Zhu et al. [2023a], or even hallucinations [Li et al., 2023d].

Post-processing the Outputs. Note that in step (3) of EMT, using the `openaiAPI` is not the only solution for evaluating the correctness of the outputs generated by MLLMs. In the future, there are several solutions. (1) Utilize a sentence embedding model. N formatted ground truth phrases can be fed into a sentence embedding model such as CLIP text encoding resulting in N ground truth embedding $\{e_i\}$, where $i \in \{1, \cdots, N\}$. Given a generated text y for a test sample, we can feed its CLIP text embedding $e(y)$ and compute the matching ground truth i using $\arg\min_i \|e_i - e(y)\|_2$. (2) One can also hard code (such as finding the existence of the label names) the decision criteria for dealing with hallucination. Note that finding a perfect post-processing method for EMT is not easy, as the labels from different datasets may have many synonyms. For example, when evaluating LLaVA on the label `African_hunting_dog` in *mini*Imagenet, it is hard to determine whether a prediction of "dog" should be correct or not. Hence, we believe such confusion in synonyms should also be taken into consideration in the future when building post-processing methods.

Chapter 3

Understanding the Visual Short Coming in Multimodal Models

3.1 Introduction

Figure 3.1: Instances are systematically identified where the visual question answering (VQA) capabilities of GPT-4V [OpenAI, 2023b] fall short (`Date accessed: Nov 04, 2023`). Our research highlights scenarios in which advanced systems like GPT-4V struggle with seemingly simple questions due to inaccurate visual grounding. Text in **red** signifies an incorrect response, while text in **green** represents hallucinated explanations for the incorrect answer. All the images referenced are sourced from ImageNet-1K and LAION-Aesthetic datasets [Tong et al., 2024d].

Multimodal Large Language Models (MLLMs) [OpenAI, 2023a, Google, 2023a, Liu et al., 2023d, Dai et al., 2023] have been rapidly developing in recent times. MLLMs integrate images into large language models (LLMs) and leverage the powerful abilities of LLMs [OpenAI, 2023b, Touvron et al., 2023b, Zheng et al., 2023], showcasing remarkable proficiency in tasks such as image understanding, visual question answering, and instruction following. In particular, the recently released GPT-4V(ision) [OpenAI, 2023a] has pushed performance to an unprecedented level [OpenAI, 2023b, Yang et al., 2023c].

Beneath the advancements of these models, we find there exists a notable weakness: they still exhibit visual shortcomings, some of which are surprisingly elementary and evident (see Figure 5.1). We ask: *Where do these problems originate? Is it a deficiency in visual modality, language understanding, or their alignment?* In this work, we suggest that these shortcomings observed in MLLMs might stem from a problem related to the **visual representations**.

At their core, most MLLMs [Dai et al., 2023, Liu et al., 2023d, Zhu et al., 2023a] are built on *pretrained* vision [Radford et al., 2021, Sun et al., 2023a] and language [Zhang et al., 2023c, Touvron et al., 2023b, Zheng et al., 2023] models. These models are connected using various types of adapters [Alayrac et al., 2022, Li et al., 2023c, Liu et al., 2023d] to integrate the different modalities. A natural hypothesis is that any limitation in the pretrained vision models can cascade into the downstream MLLMs that adopt them. Studies have explored a similar issue for language. For example, Yuksekgonul et al. [2022], Tong et al. [2023] demonstrate that failure patterns in the pretrained text encoder [Radford et al., 2021, Raffel et al., 2020] will lead to downstream failures in text-guided generative models [Rombach et al., 2022, Jun and Nichol, 2023].

On the vision side, most open-source MLLMs [Alayrac et al., 2022, Li et al., 2023c, Liu et al., 2023d] adopt the pretrained Contrastive Language-Image Pre-Training (CLIP) model [Radford et al., 2021] as the visual encoder. We begin by identifying failure examples that CLIP struggles to encode properly (Section 3.2). Inspired by Tong et al. [2023], we exploit the *erroneous agreements* in the embedding space. If two visually different images are encoded similarly by CLIP, then at least one of the images is likely ambiguously encoded. We call such a pair of images a *CLIP-blind* pair. To measure the visual similarity between images, we use a vision-only self-supervised encoder such as DINOv2 [Oquab et al., 2023]. In this context, *CLIP-blind* pairs are images with similar CLIP embeddings but different DINOv2 embeddings.

We discover that these CLIP-blind pairs indeed lead to errors in downstream MLLMs. With these pairs, We introduce the **M**ulti**M**odal **V**isual **P**atterns (MMVP) benchmark. This benchmark is specifically designed to inquire about differences in CLIP-blind pairs and evaluate the visual abilities of *state-of-the-art* MLLMs with straightforward questions. We evaluate a variety of open-source [Liu et al., 2023c,d, Dai et al., 2023, Zhu et al., 2023a] and closed-source models [OpenAI, 2023b, Google, 2023a] including GPT-4V [OpenAI, 2023a], and conduct a user study to measure human performance. The results show that MLLM models struggle with straight-forward visual questions. Most of these models perform below the level of random guessing, with GPT-4V being the exception. Yet, even

GPT-4V exhibits a considerable disparity in performance – exceeding 50% – compared to human performance.

Having identified a large number of individual failure instances in MLLMs, we continue to study the systematic visual patterns in MMVP which CLIP models struggle (Section 3.3). We summarize nine prevalent patterns of the CLIP-blind pairs in MMVP, such as "orientation", "counting", and "viewpoint", which pose significant challenges for the CLIP vision encoder. Notice that there has been significant and ongoing progress in scaling up both training data and model size for CLIP [Radford et al., 2021, Sun et al., 2023a, Fang et al., 2024, Xu et al., 2024, Zhai et al., 2023a]. We categorize examples from MMVP into visual patterns to systematically assess whether scaling alone can mitigate these challenges. Our findings suggest that 7 out of the 9 identified visual patterns cannot be resolved by any large-scale CLIP-based models, indicating that model/data scaling alone is not sufficient. Moreover, we identify a strong correlation between the visual patterns that challenge CLIP models and the performance of MLLMs. If CLIP struggles with a particular visual pattern, such as "orientation", MLLMs will likely also fall short. This shows that the CLIP vision encoders could become a bottleneck in such systems.

Finally, we take a step towards improving the visual grounding of MLLMs. Since the visual shortcomings of MLLMs stem from their reliance on the CLIP model, we investigate the impact of integrating vision-centric representations into MLLMs (Section 3.4). Specifically, we explore ways to incorporate a vision-only self-supervised model, such as DINOv2 [Oquab et al., 2023], to enhance the visual grounding capabilities of MLLMs. We refer to these techniques as Mixture-of-Features (MoF). First, we linearly mix CLIP and DINOv2 features in different ratios, which we refer to as Additive-MoF (A-MoF). This process reveals that DINOv2 features are more effective in visual grounding, though they come at the cost of diminished instruction-following ability. To address this, we introduce Interleaved-MoF (I-MoF) that spatially mixes visual tokens from both CLIP and DINOv2 models. We find that this practice significantly enhances visual grounding while maintaining the instruction-following capabilities.

3.2 The Multimodal Visual Patterns (MMVP) Benchmark

Currently, the majority of open-source MLLMs [Liu et al., 2023d, Zhu et al., 2023a, Dai et al., 2023] use the *off-the-shelf* CLIP vision encoders to process images. In this section, we begin by identifying CLIP-blind pairs in the CLIP model (Section 3.2). Subsequently, we construct the Multimodal Visual Patterns-MLLM (MMVP-MLLM) benchmark using these CLIP-blind pairs (Section 3.2). We evaluate SOTA MLLMs including GPT-4V on the benchmark (Section 3.2) and find that all the tested models struggle with simple questions on visual details. A visualization of this process is provided in Figure 3.2.

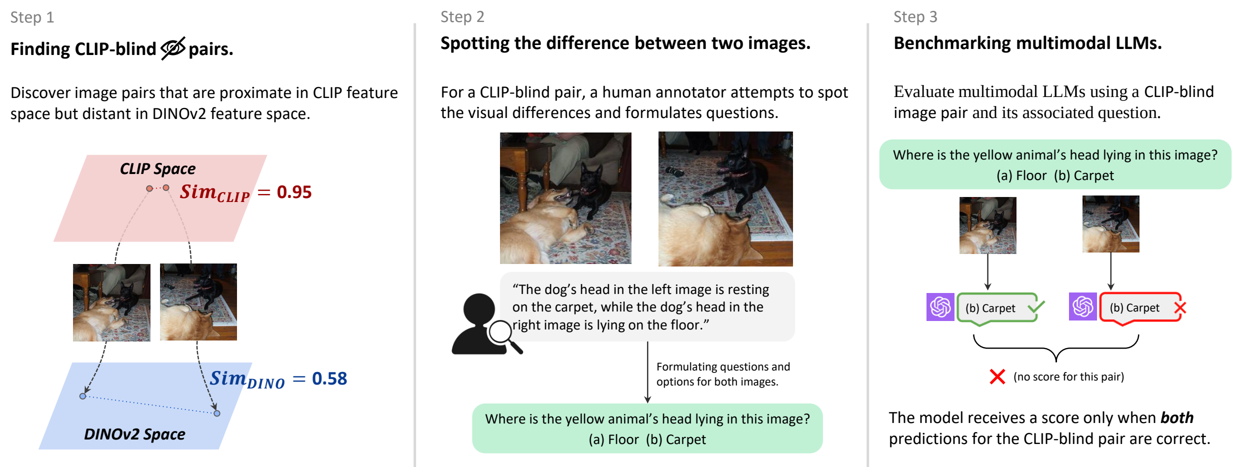


Figure 3.2: Constructing MMVP benchmark via CLIP-blind pairs. **Left:** We start with finding CLIP-blind pairs that have similar CLIP embedding but different DINOv2 embedding. **Center:** We manually inspect the differences between pair-wise images and formulate questions based on the differences in the images. **Right:** We ask MLLMs the question alongside the CLIP-blind pair. The model receives a score only when both questions for the CLIP-blind pair are answered correctly.

Finding CLIP-blind Pairs

It is challenging to directly find instances (images) that the CLIP vision encoder struggles to encode "properly". To circumvent this issue, we extend the idea proposed in Tong et al. [2023] to automatically find blind pairs in vision models. The underlying principle is simple: if two images, despite having stark visual differences, are encoded similarly by the CLIP vision encoder, then one of them is likely encoded ambiguously (See Figure 3.2 left for example). To measure the visual difference between two images, we examine the images' representations within a reference model: a vision-only self-supervised model trained without any language guidance, e.g., DINOv2 [Oquab et al., 2023]. These models are shown to capture more visual details and information [Oquab et al., 2023, Singh et al., 2023].

We take ImageNet [Russakovsky et al., 2015] and LAION-Aesthetics [Schuhmann et al., 2022], to collect these CLIP-blind pairs. For each pair, we compute its CLIP embeddings using CLIP-ViT-L-14 [Dosovitskiy et al., 2021, Radford et al., 2021] model and their DINOv2 embeddings using DINOv2-ViT-L-14 [Dosovitskiy et al., 2021, Oquab et al., 2023] model. We return pairs such that the cosine similarity exceeds 0.95 for CLIP embeddings and less than 0.6 for DINOv2 embeddings.

Designing Benchmark from CLIP-blind Pairs

We introduce the Multimodal Visual Patterns (MMVP) benchmark, and a Visual Question Answering (VQA) benchmark. Utilizing the collected CLIP-blind pairs, we carefully design 150 pairs with 300 questions. For each CLIP-blind pair of images, we manually pinpoint the visual details that the CLIP vision encoder overlooks (see the middle of Figure 3.2) and craft questions that probe these visual details, for example "Is the dog facing left or right?" (See the right of Figure 3.2 and more examples in Figure 3.3). The primary goal is to determine whether MLLM models would fail when posed with these seemingly basic questions and overlook critical visual details. Hence, the questions are intentionally straightforward and unambiguous.

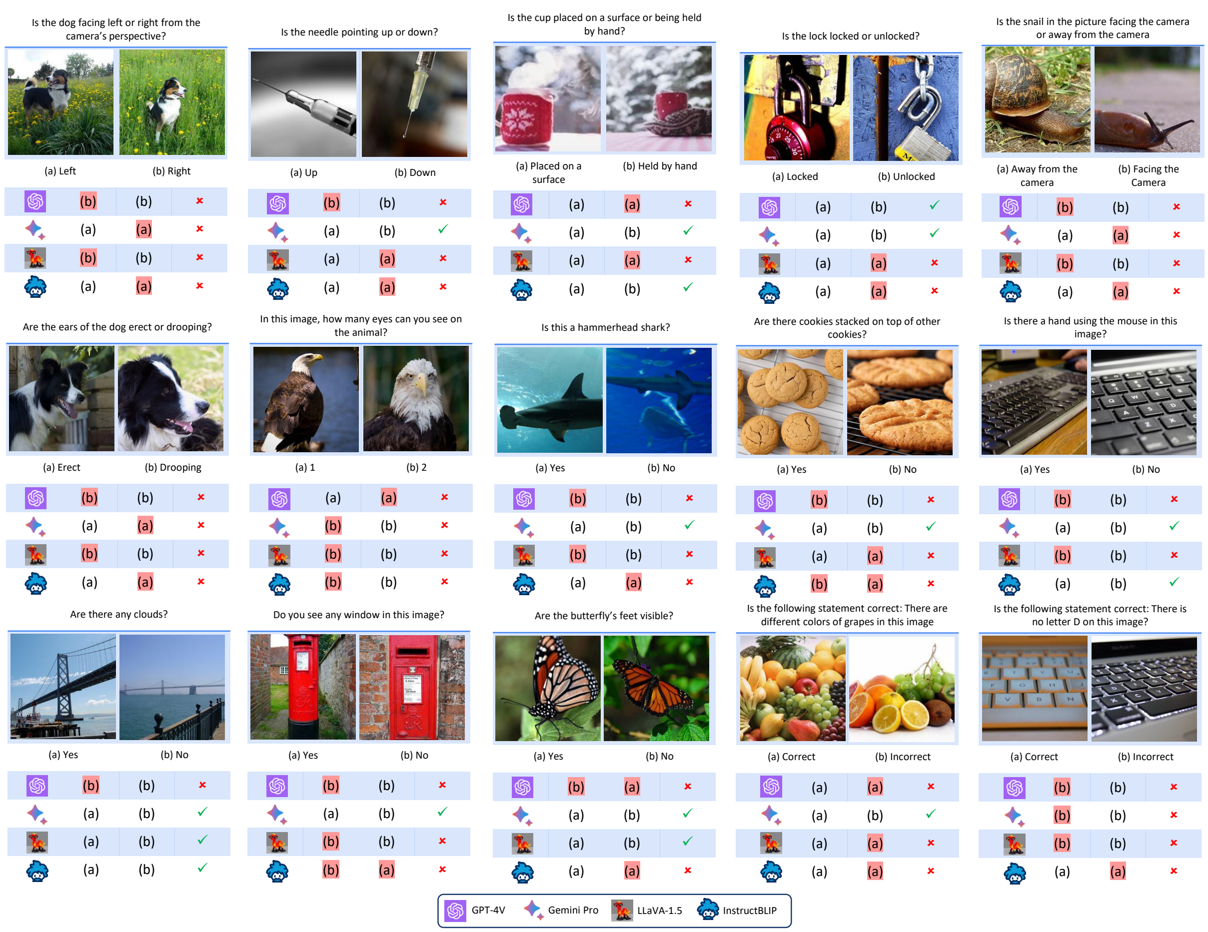


Figure 3.3: **Examples of Questions in the MMVP benchmark.** Incorrect answers are shaded in red. A model is considered correct only if it answers both questions in a pair correctly. Both leading closed-source models (GPT-4V, Gemini) and open-source models (LLaVA-1.5, InstructBLIP) fail these simple visual questions. (See Appendix B.2 for all the questions in MMVP benchmark.)

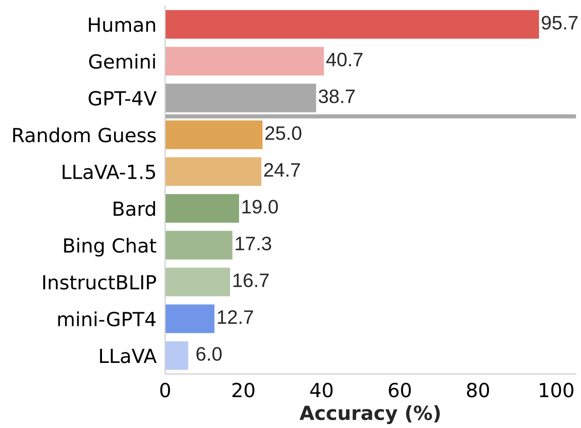


Figure 3.4: **Benchmark results of current *SOTA* MLLM models and humans.** We evaluate benchmark questions for current *SOTA* MLLM models and human performances through user studies.

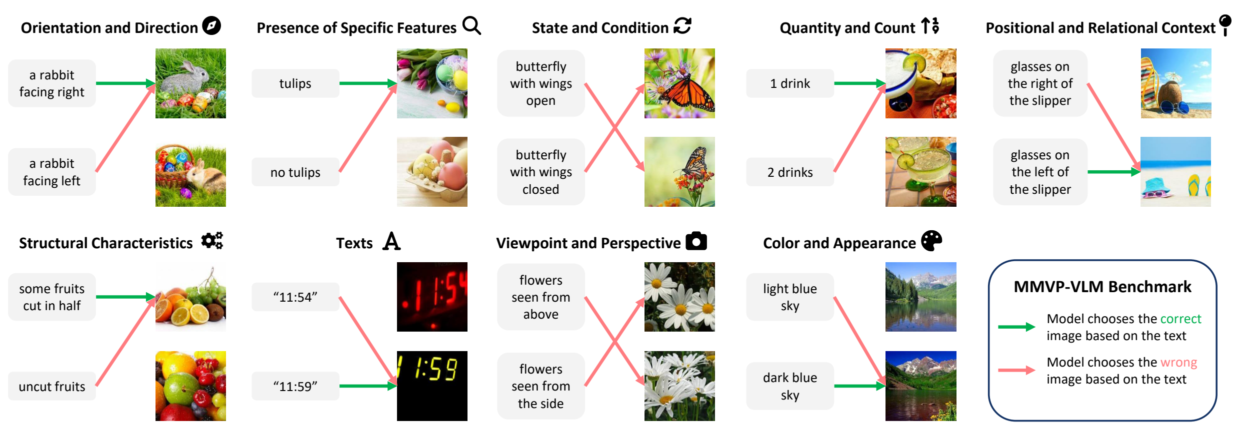


Figure 3.5: **Examples from MMVP-VLM**. MMVP-VLM consists of image pairs across nine visual patterns. The examples in the figure are from EVA01 ViT-g-14 model [Sun et al., 2023a], one of the largest CLIP models that also fails to choose the right image given the text description.

Benchmark Results

We assess the questions on *SOTA* open-source models (LLaVA-1.5 [Liu et al., 2023d], InstructBLIP [Dai et al., 2023], Mini-GPT4 [Zhu et al., 2023a]) and closed-source models (GPT-4V [OpenAI, 2023a], Gemini [Google, 2023b], Bard [Google, 2023a]) We leave details of how we access the model in Appendix B.2. In our evaluation, each question is queried independently, eliminating any biases from chat histories. We also evaluate human performance through a user study where users are presented with 300 questions in a randomized sequence. For any given pair of images, we consider a pair of images to be correctly answered if both the questions associated with the pair are answered accurately.

Human study confirms questions are straightforward. As shown in Figure 3.4, human participants accurately answer an average of 95.7% of the questions. This high accuracy rate underscores the ease of the questions. More details can be found in Appendix B.2.

Current MLLMs struggle with visual details. As shown in Figure 3.4, there is a significant performance gap between human and MLLM models, despite the latter often demonstrating impressive results [Bubeck et al., 2023, Li et al., 2023d]. Models except GPT-4V and Gemini, scored below random guess level (25%). Most advanced GPT-4V and Gemini also face challenges in addressing basic visual grounding questions. Figures 5.1 and 3.3 provide examples of errors made by models. The outcomes suggest that irrespective of model size or training data, struggle with visual details.

We have also conducted an ablation study, such as swapping options and changing notations in the question formulation (see Appendix B.2 for more details), to further confirm that this poor performance stems from visual incapability, not hallucination in the language models.

3.3 Systematic Failures in CLIP

In the previous section, we identify CLIP-blind pairs and use them to find failures in MLLMs. Here, we delve deeper into these pairs to investigate (i) systematic visual patterns emerged from CLIP-blind pairs (Section 3.3), (ii) whether these visual patterns pose challenges for CLIP-based models with massive scaling up (Section 3.3), and (iii) the correlation between failure patterns in CLIP models and those in MLLMs (Section 3.3).

Visual Patterns in CLIP-blind Pairs

Having identified the CLIP-blind pairs, we summarize systematic visual patterns that the CLIP vision encoders might consistently misinterpret. It is too abstract to directly capture systematic visual patterns in the CLIP-blind pairs. Therefore, we turn to the questions and options from the MMVP benchmark. With these questions, we transform abstract visual patterns in images into clearer, language-based descriptors that are easier to categorize.

In this work, we use GPT-4 [OpenAI, 2023b] to categorize general patterns by prompting it with the following:

> User
>
> I am analyzing an image embedding model. Can you go through the questions and options, trying to figure out some general patterns that the embedding model struggles with? Please focus on the visual features and generalize patterns that are important to vision models [MMVP Questions and Options]

We identify 9 visual patterns:

	Image Size	Params (M)	IN-1k ZeroShot										MMVP Average
OpenAI ViT-L-14	224^2	427.6	75.5	13.3	13.3	20.0	20.0	13.3	53.3	20.0	6.7	13.3	19.3
OpenAI ViT-L-14	336^2	427.9	76.6	0.0	20.0	40.0	20.0	6.7	20.0	33.3	6.7	33.3	20.0
SigLIP ViT-SO-14	224^2	877.4	82.0	26.7	20.0	53.3	40.0	20.0	66.7	40.0	20.0	53.3	37.8
SigLIP ViT-SO-14	384^2	878.0	83.1	20.0	26.7	60.0	33.3	13.3	66.7	33.3	26.7	53.3	37.0
DFN ViT-H-14	224^2	986.1	83.4	20.0	26.7	73.3	26.7	26.7	66.7	46.7	13.3	53.3	39.3
DFN ViT-H-14	378^2	986.7	84.4	13.3	20.0	53.3	33.3	26.7	66.7	40.0	20.0	40.0	34.8
MetaCLIP ViT-L-14	224^2	427.6	79.2	13.3	6.7	66.7	6.7	33.3	46.7	20.0	6.7	13.3	23.7
MetaCLIP ViT-H-14	224^2	986.1	80.6	6.7	13.3	60.0	13.3	6.7	53.3	26.7	13.3	33.3	25.2
EVA01 ViT-g-14	224^2	1136.4	78.5	6.7	26.7	40.0	6.7	13.3	66.7	13.3	13.3	20.0	23.0
EVA02 ViT-bigE-14+	224^2	5044.9	82.0	13.3	20.0	66.7	26.7	26.7	66.7	26.7	20.0	33.3	33.3

Table 3.1: Performance of various CLIP based models on different visual patterns in MMVP-VLM benchmark. Models scaled up in resolution show minimal improvement, whereas a slight advantage is observed when scaling up the network. For each visual pattern, ImageNet-1k Zero-shot accuracy and MMVP average, we use light gray to highlight the best performance. For most of the visual patterns, all CLIP-based methods show struggle, as evident from the scores. We use symbols for visual patterns due to space limit: : Orientation and Direction, : Presence of Specific Features, : State and Condition, : Quantity and Count, : Positional and Relational Context, : Color and Appearance, : Structural and Physical Characteristics, : Texts, : Viewpoint and Perspective.

- Orientation and Direction
- Presence of Specific Features
- State and Condition
- Quantity and Count
- Positional and Relational Context
- Color and Appearance
- Structural and Physical Characteristics
- Text
- Viewpoint and Perspective

These visual patterns suggest that CLIP vision encoders overly focus on high-level semantic understanding, overlooking intricate details of the visual world. Full descriptions of the visual patterns can be found in Appendix B.4.

The MMVP-VLM Benchmark

CLIP-based models have developed rapidly since the introduction in the first paper [Radford et al., 2021]. We want to test whether these visual patterns still impose challenges to the more recent CLIP models [Fang et al., 2024, Sun et al., 2023a, Zhai et al., 2023a, Xu et al., 2024], which significantly scale up in terms of training data and model size. In doing so, we introduce a new benchmark: MMVP-VLM to systematically study if CLIP models handle this visual pattern well.

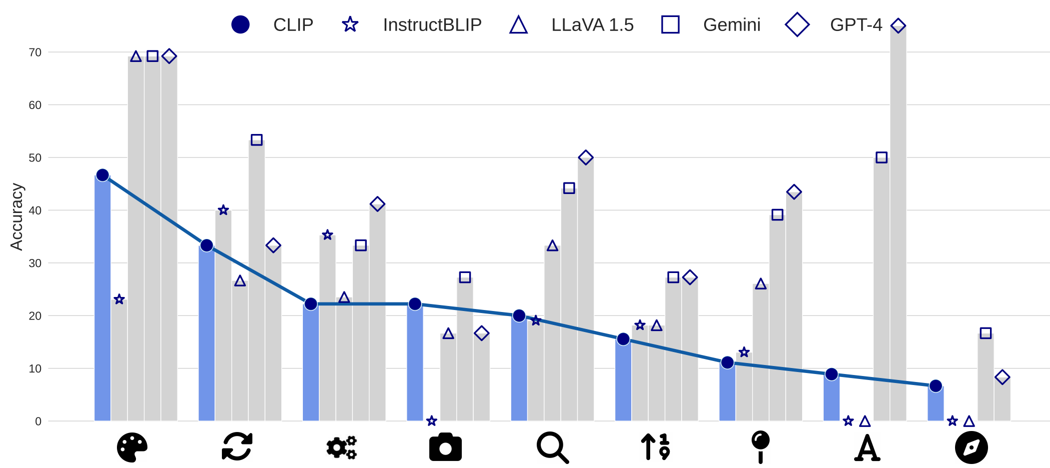


Figure 3.6: **CLIP and MLLM's performance on visual patterns.** If CLIP performs poorly on a visual pattern such as " orientation", MLLMs also underperform on the visual pattern.

We distill a subset of questions from the MMVP benchmark into simpler language descriptions and categorize them into visual patterns. To maintain a balanced number of questions for each visual pattern, we add a few questions, if needed, to ensure that each visual pattern is represented by 15 text-image pairs. Examples of pairs are shown in Figure 3.5. A pair is deemed correctly answered if the model can accurately match both image-text combinations.

We evaluate MMVP-VLM on a variety of CLIP models [Radford et al., 2021, Sun et al., 2023a, Fang et al., 2024, Xu et al., 2024, Zhai et al., 2023a]. These models vary in aspects like size, training data, and methodology. As evidenced in Table 3.1, increasing network size and training data only aids in identifying two visual patterns – "color and appearance" and "state and condition". The rest of the visual patterns continue to challenge all CLIP-based models. We also find that the ImageNet-1k zero-shot accuracy is not a definitive indicator of a model's performance regarding visual patterns. This underscores the necessity for additional evaluation metrics, such as MMVP-VLM, to accurately assess the model's capabilities in areas beyond image classification.

How CLIP's Errors Affect MLLMs

After analyzing the visual patterns that CLIP models struggle with, we pose the following question: Is there a correlation between the underperformance of CLIP and MLLMs' visual incapability? To explore this, we categorize questions from MMVP into these visual patterns summarized and calculate each MLLM's performance on these patterns.

In Figure 3.6, we plot CLIP's performance and MLLMs' performance for each visual pattern. When the CLIP vision encoder underperforms on a certain visual pattern, the MLLM tends to exhibit similar shortcomings. Open-source models such as LLaVA 1.5 [Liu et al., 2023c] and InstructBLIP [Dai et al., 2023] that explicitly use the CLIP vision encoder display a strong correlation in performance.

Further, we calculate the Pearson Correlation Coefficient between the CLIP model and MLLM's performance on each visual pattern. Results show that LLaVA 1.5 and InstructBLIP all possess a coefficient score greater than 0.7. This high score indicates a strong correlation that weaknesses in visual pattern recognition in the CLIP model are transferred to MLLMs. More details on the Pearson Correlation Coefficient can be found in Appendix B.3.

3.4 Mixture-of-Features (MoF) for MLLM

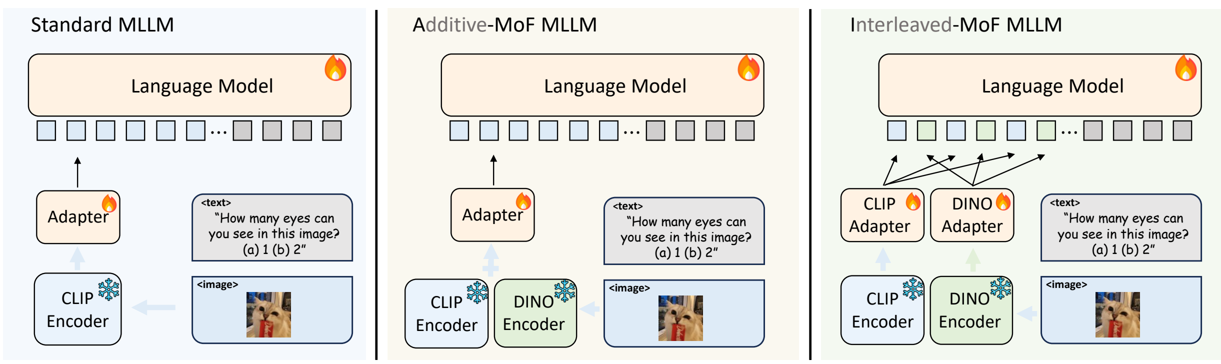


Figure 3.7: **Different Mixture-of-Feature (MoF) Strategies in MLLM.** *Left*: Standard MLLM that uses CLIP as *off-the-shelf* pretrained vision encoder; *Middle*: Additive MoF (A-MoF) MLLM: Linearly mixing CLIP and DINOv2 features before the adapter; *Right*: Interleaved MoF (I-MoF MLLM) Spatially interleaving CLIP visual tokens and DINOv2 visual tokens after the adapter.

Based on our exploration in earlier sections, a natural question arises: *If open-sourced MLLM's visual shortcomings come from the CLIP vision encoder, how do we build a more competent visual encoder?* In this section, we take initial steps to answer the question by studying Mixture-of-Features (MoF). We start with additive MoF that mixes CLIP features and vision-only SSL model features. Results show that each encoder presents unique advantages and limitations when employed as the pretrained model in MLLM (Section 3.4). We subsequently propose Interleaved MoF that integrates the features from both CLIP and SSL into MLLM to enhance visual grounding without compromising the model's ability to follow instructions (Section 3.4).

Experiment Setting

We adopt LLaVA [Liu et al., 2023d,c] as the framework to study visual encoders in MLLM. LLaVA uses a pretrained CLIP encoder and trains an adapter to align visual tokens with language tokens in the LLM. (See left side of Figure 3.7). We use DINOv2 [Oquab et al., 2023] as the vision-only SSL model in our work because it is currently the most scalable vision-only model. Our exploration includes the use of two visual encoders: CLIP-ViT-L-14 [Radford et al., 2021] and DINOV2-ViT-L-14 [Oquab et al., 2023]. To ensure consistent and fair comparisons, we train and finetune our model with the same experiment setting in LLaVA. We include the additional experimental details in Appendix B.1.

Additive MoF

We add a pretrained DINOv2 encoder into MLLM and mix the CLIP pretrained encoder with it. We use a coefficient α to control the portion of CLIP features and $1-\alpha$ to control the amount of DINOv2 features and *linearly* add them together (See middle part of Figure 3.7 for visualization).

We evaluate the model's visual grounding ability by the MMVP proposed earlier in Section 3.2 and the model's instruction-following capability by LLaVA benchmark introduced in Liu et al. [2023d]. Initially, we conduct five experiments where we linearly transition from using 100% CLIP features to 100% DINOv2 features. In these tests, the DINOv2 feature proportions are set at $\{0.00, 0.25, 0.50, 0.75, 1.00\}$. To further verify the observed trends, we introduce two additional experiments with DINOv2 proportions of $\{0.625, 0.875\}$. Our findings, presented in Table 3.2, reveal two insights:

1. As the proportion of DINOv2 features increases, MLLM exhibits a decline in its instruction-following capability. Notably, there is a sharp decrease when the DINOv2 proportion reaches 87.5%.

2. A higher proportion of DINOv2 features enhances the model's visual grounding capability, but this advantage diminishes when the DINOv2 proportion surpasses 0.75, at which point instruction-following is notably impaired.

Hence, if we were to add DINOv2 features or completely replace CLIP with DINOv2, it would result in a trade-off between visual grounding and instruction-following. A higher proportion of DINOv2 features improves the model's visual perception at the expense of its ability to follow linguistic instructions, while CLIP features enhance language comprehension but reduce visual grounding.

Interleaved MoF

We propose interleaved MoF to leverage advantages from both CLIP and DINOv2 embeddings to enhance image representation. An image concurrently passes into CLIP and DINOv2 encoders, and the resulting embeddings are individually processed by adapters. We take the processed features from CLIP and DINOv2 and interleave them while maintaining their original spatial order. We then feed the interleaved features to LLM (See right part of Figure 3.7).

method	SSL ratio	MMVP	LLaVA
LLaVA	0.0	5.5	**81.8**
LLaVA + A-MoF	0.25	7.9 +2.4	79.4 -2.4
	0.5	12.0 +6.5	78.6 -3.2
	0.625	15.0 +9.5	76.4 -5.4
	0.75	**18.7** +13.2	75.8 -6.0
	0.875	16.5 +11.0	69.3 -12.5
	1.0	13.4 +7.9	68.5 -13.3

Table 3.2: **Empirical Results of Additive MoF.** We use DINOv2 as the image SSL model in our work. With more DINOv2 features added, there is an improvement in visual grounding, while a decline in instruction following ability.

method	res	#tokens	MMVP	LLaVA	POPE
LLaVA	224^2	256	5.5	81.8	50.0
LLaVA	336^2	576	6.0	81.4	50.1
LLaVA + I-MoF	224^2	512	16.7 +10.7	82.8	51.0
LLaVA$^{1.5}$	336^2	576	24.7	84.7	85.9
LLaVA$^{1.5}$ + I-MoF	224^2	512	28.0 +3.3	82.7	86.3

Table 3.3: **Empirical Results of Interleaved MoF.** Interleaved MoF improves visual grounding while maintaining same level of instruction following ability.

We summarize the results in Table 3.3. Under the LLaVA setting, interleave MoF significantly enhances visual grounding, with a 10.7% increase observed in MMVP, without compromising the model's ability to follow instructions. This experiment is replicated with the LLaVA-1.5 setting and under various image resolution settings, yielding similar enhancements in performance. We also evaluate on POPE [Li et al., 2023d] which is designed to test hallucination in visual grounding. Interleaved-MoF also shows consistent improvement against the original LLaVA models. Merely increasing the image resolution, and consequently, the number of tokens does not boost visual grounding capabilities. Instead, it is the interleaving of MoF between vision-only SSL models and VLM models that leads to improved performance in visual grounding tasks. We conduct more experiments using MAE or MoCoV3 as vision-only SSL models in I-MoF and show similar improvements in visual grounding tasks in Appenfix B.5. We also evaluated Interleaved MoF on additional benchmarks such as MM-Bench [Liu et al., 2023e] and GQA [Hudson and Manning, 2019], finding that Interleaved MoF achieves similar performance on these benchmarks. Please refer to Appendix B.5 for more results on these benchmarks.

3.5 Related Works

Multimodal LLMs. We study the limitations of MLLMs [OpenAI, 2023a, Google, 2023a, Liu et al., 2023c,d, Dai et al., 2023] and explore possible ways to improve these models. Multimodal LLMs build from pretrained Large Language Models [OpenAI, 2023b, Anil et al., 2023, Touvron et al., 2023a,b, Zheng et al., 2023] and CLIP vision encoder [Radford et al., 2021, Sun et al., 2023a]. These systems then use an adapter, such as MLPs [Liu et al., 2023c,d], Q-Former [Li et al., 2023c, Dai et al., 2023], and gated attention [Alayrac et al., 2022, Laurençon et al., 2023], to integrate the pretrained CLIP vision encoder into LLMs. More recently, instructBLIP [Dai et al., 2023], LLaVA-1.5 [Liu et al., 2023c] highlight the importance of high-quality training data. Yet, there is a scarcity of research focusing on the impact of visual encoders, which is an important gap our work aims to address through a systematic study.

Evaluating Multimodal LLMs. MMVP assesses MLLMs using a set of simple yet critical Visual Question Answering (VQA) questions constructed from CLIP-blind pairs. Previous benchmarks such as TextVQA [Singh et al., 2019], VQAv2 [Goyal et al., 2017], and GQA [Hudson and Manning, 2019] have centered on traditional VQA queries. Recently, there are works like MM-Vet [Yu et al., 2023], POPE [Li et al., 2023d], and MM-Bench [Liu et al., 2023e] designed to specifically evaluate multimodal LLMs including hallucination, reasoning, and robustness. The previous benchmarks and evaluations have shown that Multimodal LLMs can suffer from hallucination [Liu et al., 2023b,a], catastrophic forgetting [Zhai et al., 2023b] and lack of robustness [Fu et al., 2023a]. In taking a step back to the fundamentals, our work uncovers that even the most advanced multimodal LLMs, such as GPT-4V [OpenAI, 2023a], Gemini [Google, 2023b], Bard [Liu et al., 2023c], and LLaVA-1.5 [Liu et al., 2023c], are not immune to stumbling over elementary visual questions. We also identified part of the problem as being the incapable visual encoder.

Visual Encoders. MMVP-VLM provides a detailed analysis of the visual capabilities of various CLIP variants [Radford et al., 2021, Sun et al., 2023a, Xu et al., 2024, Zhai et al., 2023a]. These models mostly follow the method proposed in Radford et al. [2021] that uses contrastive loss to train on large volumes of image-text pairs. They differ in training data [Xu et al., 2024], training recipes [Sun et al., 2023a], and objective functions [Zhai et al., 2023a]. Nonetheless, our studies show that all of these CLIP variants struggle with simple visual patterns such as “orientation”, “count”, “presence of specific features”, etc. Another line of research focuses on vision-only self-supervised learning (SSL). This category includes contrastive SSL [Chen et al., 2020b, Grill et al., 2020, Bardes et al., 2022, He et al., 2020] and mask-based SSL [Zhou et al., 2021, He et al., 2022, Assran et al., 2023]. SLIP [Mu et al., 2022] explores the synergy between CLIP and contrastive SSL, but focusing primarily on standard classification tasks. In fact, a common practice to evaluate the quality of these vision models is through linear probing or fine-tuning on ImageNet

[Russakovsky et al., 2015, Ridnik et al., 2021]. Although current evaluation methods provide a basic level of assessment on representation quality, our findings indicate a growing detachment from the needs of recent use cases. As demonstrated in the MoF experiments in Section 3.4, the CLIP vision model and the vision-only SSL models learn complementary features. However, the linear probing accuracy on ImageNet alone provides a limited understanding of feature utility in MLLMs. This observation suggests the need for more diverse evaluations [Vishniakov et al., 2024] in visual representation learning, to better align with current and emerging applications.

Ambiguities in Embedding Models. Our work exploits CLIP-blind pairs within the CLIP vision embedding space to generate examples of failures in CLIP models and subsequently MLLMs. This concept has ties to previous research focused on documenting failure modes in text embedding models [Gonen and Goldberg, 2019, May et al., 2019, Sun et al., 2019]. More recently, Thrush et al. [2022], Yuksekgonul et al. [2022] and Hsieh et al. [2023] study the binding problems CLIP faces in processing text queries, noting that CLIP models treat text input as a bag of words. Tong et al. [2023] examines the implications for downstream text-guided generative models. Tschannen et al. [2023] suggests image captioners as promising alternatives to CLIP for improving attribute binding. Our work focuses on the visual patterns.

3.6 Discussion

Circling back to the very first question we ask: is vision good enough for language? Perhaps not yet, as our study shows that vision models might become a bottleneck in multimodal systems. MLLMs fail in simple questions because their pre-trained CLIP vision encoders overlook crucial visual details in images, and systematically fail to sort important visual patterns. Yet, CLIP-type models remain the most scalable and widely used vision models today. Contrary to the popular belief that data and model scaling is a panacea, our research demonstrates that scaling alone does not rectify the inherent deficiencies in CLIP models.

Our study reveals that popular visual representation learning models – vision-and-language models and vision-only self-supervised learning models – excel in different aspects. The distinction in their capabilities go beyond conventional benchmarks such as linear probing or zero-shot accuracy on ImageNet. Although a carefully designed Mixture-of-Features approach could alleviate visual limitations and utilize the strengths of these two learning paradigms, it is necessary to develop new evaluation metrics to facilitate the development of new visual representation learning algorithms. We hope our work can motivate further innovation in vision models.

Part II

Training Vision-language Foundation Models with Reinforcement Learning

Chapter 4

Adapting RL to Foundation Model Training

4.1 Introduction

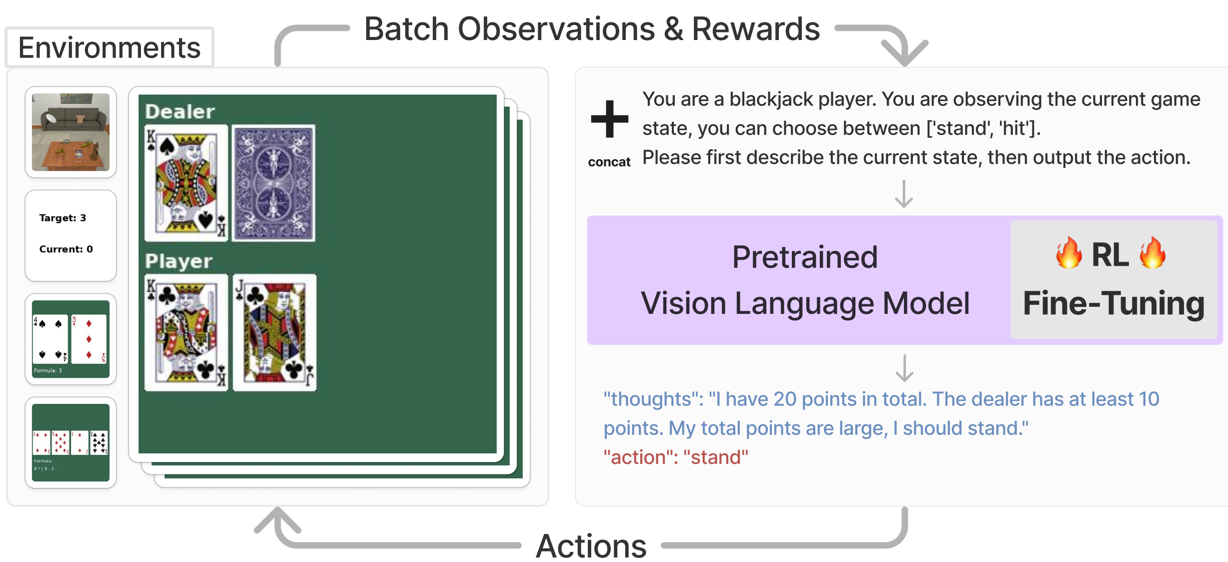


Figure 4.1: **Method overview [Zhai et al., 2024a].** We propose a framework for training large Vision-Language Models (VLM) with Reinforcement Learning (RL). At each time step, the VLM takes the current observation and a predesigned prompt as input and outputs an utterance containing a chain of thought reasoning and a text action. The text action is parsed into the environment for generating task rewards. Finally, we apply RL with the task reward to fine-tune the entire VLM.

Large vision-language models (VLMs) [Bommasani et al., 2021, OpenAI, 2023c, Google, 2023c] demonstrate remarkable capabilities as general-purpose agents in solving various tasks through language reasoning. In particular, fine-tuning VLMs with specialized visual instruction following data appears to be a key technique for improving the capabilities of VLMs [Liu et al., 2023d, Zhu et al., 2023a, Liu et al., 2023c, Li et al., 2023b]. However, visual instruction tuning may not be optimal for training decision-making agents in multi-step interactive environments requiring visual recognition and language understanding, as visual instruction tuning mainly performs supervised learning on pre-collected datasets without interacting with the environments [Huang et al., 2023a]. Consequently, if the pre-collected datasets lack sufficient diversity to cover a wide range of decision-making scenarios, visual instruction tuning may fail to improve the VLM agent's decision-making capabilities.

To unleash the learning capabilities of VLM agents in multi-step goal-directed decision-making environments, reinforcement learning (RL), a method that has proven effective in training multi-step interactive agents [Mnih et al., 2015, Silver et al., 2016, Berner et al., 2019, Vinyals et al., 2019], naturally offers a paradigm that supports this purpose. However, while RL has been widely adopted for training purely text-based tasks for large language models (LLMs) [Snell et al., 2023, Ramamurthy et al., 2023, Abdulhai et al., 2023, Zhou et al., 2024b], end-to-end VLM fine-tuning with RL for goal-directed multi-step tasks has not yet been studied, to the best of our knowledge.

Our main contribution in this paper is an algorithmic framework that directly fine-tunes VLMs with RL for multi-step goal-directed decision-making tasks requiring vision-language understanding. In our framework, the VLM first receives a task description prompt, which guides it to generate task-specific chain-of-thought (CoT) reasoning [Wei et al., 2022b, Wang et al., 2023b] (blue parts in Figure 4.1), followed by a text-based action (red parts in Figure 4.1). The CoT reasoning is designed for efficient explorations by prompting the VLMs to generate intermediate reasoning that leads to the final text-based action. Our framework then parses the text-based actions into executable actions for the environment, which generates potentially goal-directed rewards and the next state for RL training.

To evaluate the effectiveness of our method in enhancing a VLM's decision-making capabilities, we adopt a 7b model [Liu et al., 2024a] as the backbone VLM and apply our method to five decision-making tasks. These tasks come from two domains: an original domain, which evaluates the VLM's decision-making capabilities requiring fine-grained visual recognition and language reasoning, and an embodied AI domain [Shridhar et al., 2021] focusing on testing tasks demanding visual semantic reasoning capabilities. Empirical results show that our method enhances the decision-making capabilities of VLMs in both domains, enabling 7b models to surpass the performance of commercial models such as GPT4-V [OpenAI, 2023c] and Gemini [Google, 2023c]. Moreover, our experiments reveal that CoT reasoning is crucial for performance improvement in our RL training. Specifically, we test our method on the same tasks *without* the CoT reasoning and observe a significant drop in overall performance in both domains.

4.2 Related Work

Training LLMs or VLMs with RL. RL has been widely adopted for training LLMs and VLMs [Ziegler et al., 2019, Stiennon et al., 2020, von Werra et al., 2020, Ouyang et al., 2022, Castricato et al., 2023, Ramamurthy et al., 2023, Carta et al., 2023, OpenAI, 2023b, Google, 2023c, Sun et al., 2023b, Snell et al., 2023, Abdulhai et al., 2023, Hong et al., 2023a, Zhou et al., 2024b]. Some studies [Ziegler et al., 2019, Stiennon et al., 2020, Ouyang et al., 2022, Castricato et al., 2023, OpenAI, 2023b, Google, 2023c, Sun et al., 2023b] focus on applying RL from human feedback (RLHF), which involves learning reward models from human feedback before deploying RL. Other research [Ramamurthy et al., 2023, Carta et al., 2023, Snell et al., 2023, Abdulhai et al., 2023, Hong et al., 2023a, Zhou et al., 2024b] focuses on deploying RL with task-specific reward functions without using human preference data. Our paper is similar to the latter [Ramamurthy et al., 2023, Carta et al., 2023, Snell et al., 2023, Abdulhai et al., 2023, Hong et al., 2023a, Zhou et al., 2024b] which applies RL to train LLMs on customized reward functions from different environments. There are two major differences between our paper and prior works [Ramamurthy et al., 2023, Snell et al., 2023, Abdulhai et al., 2023, Hong et al., 2023a, Zhou et al., 2024b]. Firstly, our method incorporates visual inputs, broadening its applicability to a wider range of tasks that require vision-language understanding or multimodal reasoning [Li, 2023, Lu et al., 2024]. Secondly, while previous works do not explore how CoT reasoning affects RL training on large models in general, we identify CoT reasoning as a crucial component for enhancing RL training. We empirically observe that incorporating CoT reasoning significantly improves the overall performance of RL training on *all* tested domains.

Adopting LLMs and VLMs as decision-making agents. Many prior works have studied various methods of using frozen LLMs and VLMs for decision-making. One line of work studies the prompting techniques [Wei et al., 2022b, Dong et al., 2022, Yao et al., 2023b,a, Wang et al., 2023c, Lightman et al., 2023, Xi et al., 2023, Pan et al., 2023, Wang et al., 2023a, Park et al., 2023, Huang et al., 2023c] for enhancing the decision-making capabilities of large foundation models, see Dong et al. [2022], Yang et al. [2023a] for a detailed survey for other prompting based methods. Our work differs from all prompting-based methods since we directly use RL to fine-tune the entire VLM as decision-making agents. Other studies [Mu et al., 2023, Szot et al., 2023, Baumli et al., 2023, Rocamonde et al., 2024, Chen et al., 2024c] integrate frozen VLMs ot LLMs into their training pipeline for processing task descriptions or feature extraction, without using text-based actions. focuses on integrating different components from VLMs for downstream RL training. For example, some studies use the VLMs or CLIP vision encoder [Pan et al., 2024, Mu et al., 2023, Szot et al., 2023] as reward models for training, which differs from our method since we adopt rewards from the environments. Other studies [Mu et al., 2023, Szot et al., 2023, Chen et al., 2024c] integrate frozen VLMs/LLMs into their training pipeline for processing task descriptions [Mu et al., 2023, Szot et al., 2023, Pan et al., 2024] or feature extraction [Chen et al., 2024c], without using text-based actions. Our paper differs from these works [Mu et al., 2023, Szot et al., 2023, Chen et al., 2024c] in two major aspects. From a technical perspective, we focus on a more challenging paradigm by directly fine-tuning the entire VLM with RL, whereas previous methods [Mu et al., 2023, Szot et al., 2023, Chen et al., 2024c] only train additional MLP or transformer layers to connect the frozen LLM/VLM with the action space. More importantly, our method directly interacts with the environments using *open-ended text*, enabling it to utilize the CoT reasoning capability of VLMs for more efficient explorations for decision-making.

Evaluating VLMs as decision-making agents. Previous studies have thoroughly examined the fundamental evaluations of VLMs in non-interactive tasks [Antol et al., 2015, Liu et al., 2023e, Yu et al., 2023, Liu et al., 2023a, Tong et al., 2024d, Zhai et al., 2024b, Fu et al., 2023a]. Our focus, however, is on evaluating a VLM's decision-making capabilities in interactive environments that require both visual recognition and language reasoning. Representative interactive environments include purely text-based environments [Côté et al., 2019, Küttler et al., 2020, Wang et al., 2022] or embodied AI environments [Manolis Savva* et al., 2019, Shridhar et al., 2021, Shen et al., 2021, Fan et al., 2022]. We adopt the ALFWorld [Shridhar et al., 2021] embodied environment for evaluating our method's ability to improve VLM's visual semantic reasoning capabilities. In addition to the ALFWorld embodied AI environment, we also design an original "gym-like" [Brockman et al., 2016] environment to test VLM's decision-making capabilities in tasks that require fine-grained visual recognition and language reasoning.

CoT prompting. Recent studies in prompting for LLMs have demonstrated the crucial role of CoT in enhancing complex reasoning capabilities [Wei et al., 2022b, Kojima et al., 2022, Fu et al., 2023b, Wang et al., 2023b, Zhou et al., 2023, Yao et al., 2023b]. Wei et al. [2022b] show that CoT reasoning can significantly boost LLMs' performance across different reasoning tasks by showing that adding simple exemplar-based prompts, leading to better performance on benchmarks such as the GSM8K [Cobbe et al., 2021]. A follow-up study [Wang et al., 2023b] proposes a novel self-consistency decoding strategy that explores multiple reasoning paths, demonstrating substantial gains in arithmetic and commonsense reasoning tasks. Other works [Kojima et al., 2022, Zhou et al., 2023, Fu et al., 2023b] have shown that adding prompts to break complex tasks into subtasks and solve them step-by-step significantly improves LLM's reasoning capability. Our work differs from these CoT prompting studies as we aim to provide an algorithmic framework that can train VLMs with RL, where the CoT prompting appears as a key component of the framework. In contrast, prior works focus on improving the reasoning capabilities of LLMs with increasingly sophisticated prompting of frozen models.

4.3 Preliminaries

Standard RL terminologies. We follow the standard notations from classic RL literature [Sutton and Barto, 2018, Agarwal et al., 2019]. Specifically, we use $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$ to denote an MDP, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, P denotes the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function and $\gamma \in [0, 1]$ denotes the discount factor. Our goal is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the overall discounted return $\max_{\pi\in\Pi} \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$, where T is the maximum number of steps per episode. Without loss of generality, we use $\pi(a|s) \in [0, 1]$ to denote probability of π choosing a at s.

Adapting the RL formalism to VLMs. We use $\mathcal{V}$ to denote the discrete and finite vocabulary (token) space, and we use $\mathcal{V}^m, \mathcal{V}^n$ to represent the input and output text space, where m and n represent the maximum token length of the input and output sequence. We adapt the RL formalism to VLMs by treating the combination of the *vision and language inputs* to VLMs as the state space: $\mathcal{S} = \mathcal{O} \times \mathcal{V}^m$, where $\mathcal{O}$ is the space of all RGB images. We view each utterance [Abdulhai et al., 2023, Zhou et al., 2024b] of the language outputs from VLMs as the action space $\mathcal{V}^n$. Therefore, the input and output of a VLM policy with parameter θ can be written as $\pi_\theta : \mathcal{O} \times \mathcal{V}^m \to \mathcal{V}^n$. For example, in the `Blackjack` task shown in Figure 4.1, each state s consists of an RGB image o with the cards of the dealer and the player, as well as an input prompt $\boldsymbol{v}^{\text{in}}$ with maximum token length m, and the text output $\boldsymbol{v}^{\text{out}} = \pi_\theta(o, \boldsymbol{v}^{\text{in}})$ (with a maximum token n) will later be parsed as an action to interact with the environment. Similar to the standard RL setting, we use $\pi_\theta(\boldsymbol{v}^{\text{out}}|o, \boldsymbol{v}^{\text{in}}) \in [0, 1]$ to denote the probability of a VLM policy π_θ outputting $\boldsymbol{v}^{\text{out}}$ with input image o and input prompt $\boldsymbol{v}^{\text{in}}$.

4.4 Training VLMs with RL

Compared to classic MLP-based policy networks [Schulman et al., 2015, 2016, 2017], a natural advantage of VLM policies is that they can leverage CoT reasoning for efficient exploration, by performing intermediate reasoning steps that lead to the final decision. However, training a VLM policy π_θ with RL presents additional challenges. First, the VLM policy $\pi_\theta(o, \boldsymbol{v}^{\text{in}})$ directly generates open-ended text rather than vectorized actions in classic policy gradient-based RL methods [Schulman et al., 2015, 2016, 2017, Haarnoja et al., 2018], complicating direct interaction with the environment. Even with a parsing mechanism $f : \mathcal{V}^n \to \mathcal{A}$ that maps open-ended text $\boldsymbol{v}^{\text{out}}$ to a *legal* action a for interaction with the environment, it remains unclear how to estimate the action probability $\pi_\theta(a|o, \boldsymbol{v}^{\text{in}})$ from the text generation process.

Figure 4.2 presents an overview of our framework, leveraging the CoT reasoning and addressing the two aforementioned challenges. We design a task-specific prompt $\boldsymbol{v}^{\text{in}}$ that requires the VLM to generate a formatted output $\boldsymbol{v}^{\text{out}}$, including the CoT reasoning. Next, we adopt a post-processing function f to parse open-ended text into a *legal* action a_t that can directly interact with the environment. To compute $\pi_\theta(a|o, \boldsymbol{v}^{\text{in}})$, we develop a method to estimate its value based on the probability of each output token in $\boldsymbol{v}^{\text{out}}$.

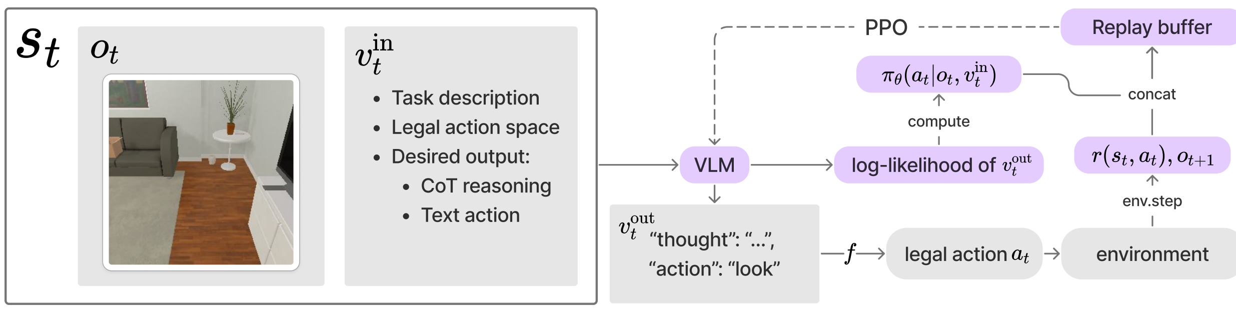


Figure 4.2: **A diagram of the proposed RL fine-tuning framework.** At time step t, the state s_t contains an input prompt $\boldsymbol{v}_t^{\text{in}}$ and a visual observation o_t. The VLM takes $s_t = [o_t, \boldsymbol{v}_t^{\text{in}}]$ as input and outputs open-ended text $\boldsymbol{v}_t^{\text{out}}$ containing the CoT reasoning, keywords `"action" : "`a_t`"`, and the log-likelihood of $\boldsymbol{v}_t^{\text{out}}$. We first apply a post-processing function f on $\boldsymbol{v}_t^{\text{out}}$, to obtain a *legal* action a_t which can interact with the environment. Then, we input a_t to the environment for obtaining reward $r(s_t, a_t)$ and the next observation o_{t+1}. Afterward, we devise a method to compute a numerical value of $\pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$. Finally, we use $r(s_t, a_t)$ and $\pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$ for the RL training.

The remaining Section is structured as follows. First, we describe the format of our input prompt $\boldsymbol{v}_t^{\text{in}}$ and the desired output $\boldsymbol{v}_t^{\text{out}}$ (Section 4.4). Next, we present the post-processing function f (Section 4.4). Then, we introduce a method to compute a numerical value of $\pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$ (Section 4.4). Finally, we conclude our framework in Algorithm 1 (Section 4.4).

Prompt Design for Domain-Specific Outputs

For each task $\mathcal{M}$, our input prompt $\boldsymbol{v}_t^{\text{in}}$ contains a description of the task, the legal action space of the current observation, and the desired output format (including the CoT reasoning). Our desired output $\boldsymbol{v}_t^{\text{out}}$, contains a CoT reasoning followed by the keywords `"action" : "`a_t`"` for post-processing. Figure 4.3 provides an example of our input prompt $\boldsymbol{v}_t^{\text{in}}$ and the desired formatted output $\boldsymbol{v}_t^{\text{out}}$. In particular, we define a function h which constructs $\boldsymbol{v}_t^{\text{in}}$ from the current observation o_t: $\boldsymbol{v}_t^{\text{in}} = h(o_t)$, to accommodate for tasks that may contain observation-dependent information.[1] We provide additional examples of $\boldsymbol{v}^{\text{in}}$ and $\boldsymbol{v}^{\text{out}}$ in Appendix C.1.

[1]E.g., the `alfworld` environment (to be introduced in Section 4.5) contains an observation-dependent *admissible action* space.

CoT prompt v_t^{in} for task $\mathcal{M}$
You are trying to solve a task $\mathcal{M}$. {Description of the task}. You are observing the current status of the task. The action space of $\mathcal{M}$ is {text version of all legal actions $a \in \mathcal{A}$}. Your response should be a valid json file in the following format:
{
"thoughts": "{first describe the current status of the task, then think carefully about which action to choose}",
"action": {Choose an action "$a \in \mathcal{A}$"}
}

Formatted text output v_t^{out}
{
"thoughts": "I am solving task $\mathcal{T}$, given the current status of the task, I should choose a_t",
"action": "a_t"
}

Figure 4.3: **A template of our input prompt and output text.** The blue part represents the CoT reasoning and the red part is the text-based action. Note that the CoT reasoning may contain **other task-specific descriptions**, see Appendix C.1 for more details.

Post-Processing Open-Ended Text for Legal Actions

Our post-processing mechanism involves both $\boldsymbol{v}_t^{\text{in}}$ and f. In the input prompt $\boldsymbol{v}_t^{\text{in}}$, we directly ask the VLM to output a text-based action in the format of `"action" : "`a_t`"` (see Figure 4.1 and Figure 4.2 for examples). After obtaining $\boldsymbol{v}_t^{\text{out}}$, our post-processing function f directly searches for the text-based keywords `"action" : "`a_t`"` from $\boldsymbol{v}_t^{\text{out}}$, and maps it to a legal action a_t, either in symbolic or in text depending on the task of interest. For the case shown in Figure 4.1, f will map $\boldsymbol{v}_t^{\text{out}}$ to the symbolic operator that represents the action `"stand"` in the `Blackjack` task (to be introduced in Section 4.5), as the `Blackjack` task takes symbolic actions as input. For the `alfworld` [Shridhar et al., 2021] environment shown in Figure 4.2, f will map $\boldsymbol{v}_t^{\text{out}}$ to the text `"look"`, because the `alfworld` environment takes text-based actions as inputs.

However, VLMs are not always guaranteed to generate a $\boldsymbol{v}_t^{\text{out}}$ that contains the keywords `"action" : "`a_t`"`, even when we explicitly request a formatted output from $\boldsymbol{v}_t^{\text{in}}$. To continue the RL training when $\boldsymbol{v}_t^{\text{out}}$ does not contain any legal action, we perform *random exploration* by selecting a legal action $a_t \in \mathcal{A}$ uniformly at random. Mathematically, f is defined as follows:

$$f(\boldsymbol{v}^{\text{out}}) = \begin{cases} a, & \text{if } \texttt{"action" : "}a\texttt{"} \in \boldsymbol{v}^{\text{out}}, \\ \texttt{Unif}(\mathcal{A}), & \text{otherwise.} \end{cases} \tag{4.1}$$

Estimating Action Probabilities of VLM Policies

To estimate the action probability $\log \pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$ (or equivalently $\log \pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$) for policy gradient-based methods [Schulman et al., 2017], a naïve calculation is directly using $\log \pi_\theta(\boldsymbol{v}_t^{\text{out}}|o_t, \boldsymbol{v}_t^{\text{in}})$ as $\log \pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$, by summing the log-likelihood of all tokens in $\boldsymbol{v}_t^{\text{out}}$. This is because

$$\begin{aligned}
&\log \pi_\theta(\boldsymbol{v}_t^{\text{out}}|o_t, \boldsymbol{v}_t^{\text{in}}) = \log \frac{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{out}})}{P(o_t, \boldsymbol{v}_t^{\text{in}})} \\
=&\log \left[\frac{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:n]})}{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:n-1]})} \cdots \frac{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:2]})}{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:1]})} \frac{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:1]})}{P(o_t, \boldsymbol{v}_t^{\text{in}})}\right] = \sum_{i=1}^{n} \log \left[\frac{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:i]})}{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:i-1]})}\right].
\end{aligned} \tag{4.2}$$

In the equation above, we use $\boldsymbol{v}$ to denote the output token $\boldsymbol{v}_t^{\text{out}}$ for simplicity, and we use $\boldsymbol{v}_{[:i]}$ to denote the first i tokens in $\boldsymbol{v}_t^{\text{out}}$, and we slightly abuse our notion by using $P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:0]})$ to denote $P(o_t, \boldsymbol{v}_t^{\text{in}})$ in the log summation. Hence, a natural way to compute a numerical value for $\log \pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$ is $\sum_{i=1}^{n} \log \left[\frac{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:i]})}{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:i-1]})}\right]$.

However, the naïve calculation $\log \pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}}) \leftarrow \sum_{i=1}^{n} \log \left[\frac{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:i]})}{P(o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_{[:i-1]})}\right]$ may not be ideal for computing $\pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$ since our formatted output $\boldsymbol{v}_t^{\text{out}}$ also contains CoT reasoning. This is because in $\boldsymbol{v}_t^{\text{out}} = [\boldsymbol{v}_t^{\text{tht}}, \boldsymbol{v}_t^{\text{act}}]$, the CoT reasoning tokens $\boldsymbol{v}_t^{\text{tht}}$ are generally much longer than the action tokens $\boldsymbol{v}_t^{\text{act}}$ (see the blue and red parts in Figure 4.3 for examples, and see Table 4.1 for a relative scaling of their sum log-likelihood). Hence the naïve computation $\log \pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}}) \leftarrow \log \pi_\theta(\boldsymbol{v}_t^{\text{tht}}|o_t, \boldsymbol{v}_t^{\text{in}}) + \log \pi_\theta(\boldsymbol{v}_t^{\text{act}}|o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{tht}})$ will make $\log \pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$ largely determined by the CoT tokens $\log \pi_\theta(\boldsymbol{v}_t^{\text{tht}}|o_t, \boldsymbol{v}_t^{\text{in}})$, which is practically undesirable because our post-processing function f only relies on $\boldsymbol{v}_t^{\text{act}}$ for decision-making.

log	NL	BJ	EZP	P24	ALF
$\boldsymbol{v}_t^{\text{tht}}$	-3.4	-2.2	-9.0	-37.6	-20.3
$\boldsymbol{v}_t^{\text{act}}$	0.0	0.0	0.0	0.0	-0.4

Table 4.1: **The absolute values of sum log probability of $\boldsymbol{v}_t^{\text{tht}}$ is much larger than $\boldsymbol{v}_t^{\text{act}}$.** Each number is averaged among 1000 samples on our evaluation tasks to be introduced in Section 4.5.

As shown in Table 4.1, $\log \pi_\theta(\boldsymbol{v}_t^{\text{tht}}|o_t, \boldsymbol{v}_t^{\text{in}})$ typically has a much larger magnitude than $\log P(\boldsymbol{v}_t^{\text{act}}|o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{tht}})$ across all tasks we have tested (in terms of absolute value). Hence, to mitigate the effect of the CoT tokens, we adopt a scaling factor $\lambda \in [0, 1]$ to scale down $\log \pi_\theta(\boldsymbol{v}_t^{\text{tht}}|o_t, \boldsymbol{v}_t^{\text{in}})$ for obtaining a regularized version of $\log \pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$, which results in

$$\log \pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}}) \leftarrow \lambda \log \pi_\theta(\boldsymbol{v}_t^{\text{tht}}|o_t, \boldsymbol{v}_t^{\text{in}}) + \log \pi_\theta(\boldsymbol{v}_t^{\text{act}}|o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{tht}}). \tag{4.3}$$

Empirically, we observe the scaling factor λ could largely affect the final performance. As we will show in Section 4.6, choosing an extreme λ value (close to 1 or 0) will degrade overall performance. All of our experiments adopt $\lambda \in [0.2, 0.5]$.

Formal Implementation

Putting the prompt construction function h (Section 4.4), the post-processing function f (Section 4.4), and the computation of $\pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$ (Section 4.4) together, we conclude our method in Algorithm 1.

Algorithm 1 Training VLM with RL

Input: An environment `env`, an initial VLM with parameters θ_0.
Input: A post-processing function f, a CoT reasoning scaling factor λ.
Input: Replay buffer size B, maximum episode length T.
for $k = 0, \ldots, K-1$ **do**
    $t = 0$ ▷ Reset RL time step
    $o_t =$ `env.reset()` ▷ Reset the initial state
    $\boldsymbol{v}_t^{\text{in}} = h(o_t)$ ▷ Generate $\boldsymbol{v}_t^{\text{in}}$ from o_t, h is defined in Section 4.4
    $\mathcal{B}_k = \emptyset$ ▷ Initialize an on-policy replay buffer
    while $|\mathcal{B}_k| \leq B$ **do**
        $\boldsymbol{v}_t^{\text{out}} = \pi_{\theta_k}(o_t, \boldsymbol{v}_t^{\text{in}})$ ▷ Generate text output
        $a_t = f(\boldsymbol{v}_t^{\text{out}})$ ▷ Obtain a legal action from $\boldsymbol{v}_t^{\text{out}}$, f is defined in Equation 4.1
        $\log \pi_{\theta_k}(a_t|o_t, \boldsymbol{v}_t^{\text{in}}) = \lambda \log \pi_{\theta_k}(\boldsymbol{v}_t^{\text{tht}}|\boldsymbol{v}_t^{\text{in}}) + \log \pi_{\theta_k}(\boldsymbol{v}_t^{\text{act}}|o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{tht}})$ ▷ Equation 4.2
        $r_t, o_{t+1} = env.step(a_t)$
        $\mathcal{B}_k = \mathcal{B}_k \cup \{(o_t, a_t, r_t, \boldsymbol{v}_t^{\text{out}}, \log \pi_{\theta_k}(a_t|o_t, \boldsymbol{v}_t^{\text{in}}))\}$ ▷ Add data to the buffer $\mathcal{B}_k$
        $t = t + 1$
        if $t = T$ **then**
            $t = 0$ ▷ Reset RL time step if the maximum step is reached
            $o_0 = env.reset()$ ▷ Reset environment
        end if
        $\boldsymbol{v}_t^{\text{in}} = h(o_t)$ ▷ Prepare the next $\boldsymbol{v}_t^{\text{in}}$
    end while
    Run PPO [Schulman et al., 2017] with data $\mathcal{B}_k$ to obtain θ_{k+1}
end for
Output: θ_K.

4.5 Evaluation Tasks

How does our method improve a VLM's decision-making capabilities in tasks that require fine-grained vision-language reasoning or semantic understanding? To study this question, we adopt two different domains: `gym_cards` and `alfworld` [Shridhar et al., 2021]. Our original `gym_cards` domain is a "gym-like" environment [Brockman et al., 2016] containing four tasks designed to test the decision-making capabilities of VLMs. These tasks require fine-grained visual-language reasoning, specifically focusing on recognizing numbers for arithmetic reasoning. In addition, we also adopt `alfworld` [Shridhar et al., 2021], which assesses the decision-making capabilities of VLMs in an embodied AI setting that demands visual semantic understanding. We present some examples of the visual observations of each task in Figure 4.4. We do not include standard image-based Atari benchmarks [Bellemare et al., 2013, Machado et al., 2018] due to limited computation resources.[2]

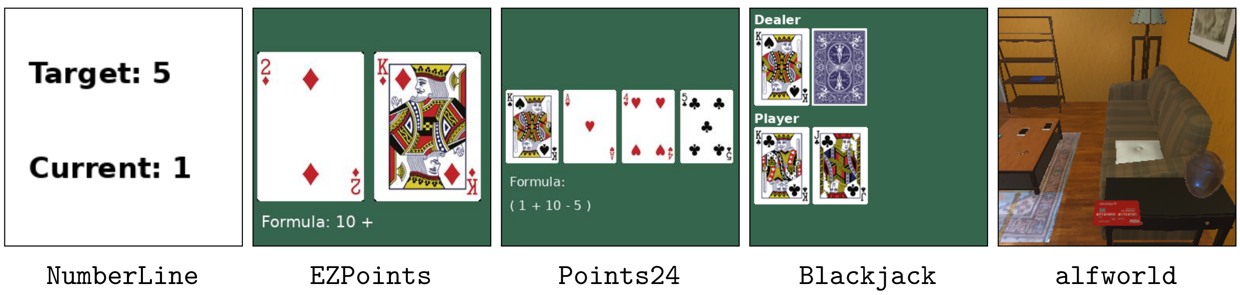


Figure 4.4: **Examples of observation of our evaluation tasks**. (a)-(d) are from our original `gym_cards` domain. (a)-(c) are deterministic tasks with *increasing difficulties*; (d) is a stochastic task.

Gym Cards

Our `gym_cards` domain is designed to evaluate a VLM's decision-making capabilities requiring fine-grained vision recognition and language reasoning. More precisely, tasks in the `gym_cards` domain require the VLM to recognize the numbers (potentially from cards) and utilize the numbers for language reasoning. As depicted in Figure 4.4, the first three tasks—`NumberLine`, `EZPoints`, and `Points24`—are deterministic, and developed to assess the VLMs' ability to identify and process numbers or mathematical operators at each time step. These tasks increase in complexity: `NumberLine` requires recognition of two numbers in an image, `EZPoints` involves identifying numbers from two cards, and `Points24` extends to recognizing four cards. The `Blackjack` task challenges the VLM further by requiring the agent to reason based on visual information and adapt to stochastic outcomes. This subsection outlines the goals of each task, and we leave the detailed descriptions of their state spaces, action spaces, and reward functions to Appendix C.1.

[2]Image-based Atari tasks generally take at least 2 million environment steps to reach a reasonable performance [Huang et al., 2024]. Our method needs roughly 30 hours to run 15k environment steps due to the model size of the backbone VLMs, which requires roughly half a year to run 2 million environment steps.

NumberLine. In this task, **the goal is to move a number to the target on a synthetic number line.** At each state s_t, the visual observation o_t contains two lines of text: "Target: x" and "Current: y_t". The agent needs to move the current number y_t to the target number x, by outputting text $\boldsymbol{v}_t^{\text{out}}$ that interacts with the discrete action space $\{\texttt{"}+\texttt{"}, \texttt{"}-\texttt{"}\}$. Mapping the $\boldsymbol{v}_t^{\text{out}}$ to $\texttt{"}+\texttt{"}$ or $\texttt{"}-\texttt{"}$ will increase or decrease the current number by 1, respectively.

EZPoints. In this task, **the goal is to output a formula using the numbers in the cards that evaluates to 12.** At each state s_t, the agent observes an image of two cards and a text version of (potentially incomplete) "formula" below the cards. The goal is to use *all* numbers in the cards (only once) to compute 12. The action space contains natural numbers in $[1, 10]$, as well as operator in $\{\texttt{"}+\texttt{"}, \texttt{"}*\texttt{"}, \texttt{"}=\texttt{"}\}$. At each state s_t, only operators and numbers that appear in the cards are *legal* actions, and "J", "Q", or "K" are treated as "10". In particular, if the output text $\boldsymbol{v}_t^{\text{out}}$ is mapped to a legal action a_t at state s_t, the text version of a_t will be appended to the "formula" in the current image of s_t resulting s_{t+1}, otherwise s_{t+1} will remain the same as s_t.

Points24. In this task, **the goal is to output a formula using the numbers in the cards that evaluates to 24.** The `Points24` task is a harder version of `EZPoints` as it contains 4 cards, hence requiring the VLMs to generate a longer formula. The rules of `Points24` are similar to `EZPoints`, despite two minor differences: the `Points24` task requires the VLM to compute a target number of 24, and its action space contains more operators: $\{\texttt{"}+\texttt{"}, \texttt{"}-\texttt{"}, \texttt{"}*\texttt{"}, \texttt{"}/\texttt{"}, \texttt{"}=\texttt{"}\}$.

Blackjack. In this task, **the goal is to win the current blackjack game.** At each state s_t, the visual observation o_t consists of two cards (one face-down) from the dealer and all cards from the player. The agent's goal in this task is to win the current game, by outputting text $\boldsymbol{v}_t^{\text{out}}$ that can be mapped to {`"stand"`, `"hit"`}. The agent will receive one more card if $\boldsymbol{v}_t^{\text{out}}$ is mapped to `"hit"`, and the game will terminate if $\boldsymbol{v}_t^{\text{out}}$ is mapped to `"stand"`.

ALFWorld

While the `gym_cards` domain is designed to assess the VLM's arithmetic reasoning requiring fine-grained visual recognition, the `alfworld` environment aims at testing VLM's decision-making tasks requiring visual semantic understanding.

ALFWorld. The ALFWorld embodied environment [Shridhar et al., 2021] is combines a text-based interactive environment [Côté et al., 2019] with a large vision-language instruction following dataset [Shridhar et al., 2020]. It contains 6 different types of goal-conditioned tasks ("Pick & Place", "Examine in Light", "Clean & Place", "Heat & Place", "Cool & Place", and "Pick Two & Place"), and **the agent's goal is to navigate in the environment via text-based actions** (e.g., `"go to shelf 1"`, `"examine sidetable 1"`). Unlike our original `gym_cards` environment, where all states share the same action space, the `alfworld` environment contains a state-dependent *admissible action* space – some actions are only available at certain states. For example, if the agent's goal is to "put some pillows on armchair", then the agent can only put a pillow *after* picking up a pillow. Hence, to incorporate the state-dependent admissible action set, our prompt of `alfworld` asks the VLM to choose among an admissible action. See Figure 4.2 for an example of the visual observation of `alfworld`. We leave the detailed descriptions of the `alfworld` (state space, action space, reward functions, and the CoT prompt) to Appendix C.1.

4.6 Experimental Results

The first part of our experiment examines how our method improves the decision-making capabilities of VLMs (Section 4.6). The second part investigates the role of CoT reasoning in our method (Section 4.6). Details of our experimental setup are provided in Appendix C.2.

Improving VLM Decision-Making Capabilities

Does our method improve the decision-making capabilities of VLM agents across various domains? To investigate this, we assess how our method improves arithmetic tasks requiring fine-grained visual recognition in the `gym_cards` domain and visual semantic reasoning in the `alfworld` domain. The `gym_cards` experiments include deterministic tasks (`NumberLine`, `EZPoints`, and `Points24`, each with increasing difficulty) and a stochastic task (`Blackjack`). In the `alfworld` domain, we evaluate overall performance and detailed task-specific performance as discussed in Section 4.5. We instantiate our method on top of the llava-v1.6-mistral-7b [Liu et al., 2024a] model and compare it against commercial models (GPT4-V and Gemini), a supervised fine-tuned version of the llava-v1.6-mistral-7b model (LLaVA-sft),[3] and a vanilla RL implementation using a CNN-based policy network (CNN+RL).[4] The final results and learning curves are presented in Table 4.2 and Figure 4.5. Details of the experimental setup are provided in Appendix C.2.

Enhancing decision-making capabilities of VLM agents across various tasks. As illustrated in Table 4.2 and Figure 4.5, our method demonstrates consistent improvement

[3]To ensure the RL training starts from a model with reasonable instruction following capabilities [Ouyang et al., 2022], our RL training for VLM starts from the LLaVA-sft checkpoint of each task, we leave the detailed training pipeline of our method to Appendix C.2.

[4]The CNN-based method adopts the same CLIP vision encoder as LLaVA-7b. Additionally, for tasks that require text inputs (e.g., `alfworld`), we adopt the `RoBERTa-base` [Liu et al., 2019] model to encode the text feature and concatenate the text and CLIP visual features for downstream RL training. Details of our CNN-based model are provided to Appendix C.2.

	gym_cards					alfworld							
	`NL`	`EZP`	`P24`	`BJ`	Avg.	Exp. Data	Pick	Look	Clean	Heat	Cool	Pick2	Avg.
BUTLER$_g$	-	-	-	-	-	✓	33.0	17.0	26.0	70.0	76.0	12.0	22.0
BUTLER	-	-	-	-	-	✓	46.0	22.0	39.0	74.0	100.0	24.0	37.0
CNN+RL	87.1	0	0	38.8	31.5	✗	0	0	0	0	0	0	0
GPT4-V	65.5	10.5	0	25.5	25.4	✗	38.2	12.1	**18.8**	6.7	17.8	14.6	19.4
Gemini	82.5	2.0	0	30.0	28.6	✗	34.6	**16.7**	0	0	0	12.0	13.5
LLaVA-sft	24.8	23.0	**2.6**	23.1	18.4	✗	39.2	0	14.4	11.1	0	**28.6**	17.7
Ours	**89.4**	**50.0**	2.3	**40.2**	**45.5**	✗	**47.4**	14.7	10.4	**14.4**	**18.8**	18.0	**21.7**

Table 4.2: **Average episode success rates (%) of different methods on `gym_cards` and `alfworld`.** For all RL-based methods (CNN+RL and our method), we present the peak numbers (first 15k environment steps for the `gym_cards` and 5k environment steps for `alfworld`) from each training curve from Figure 4.5. We average the performance of all 4 tasks on `gym_cards` with equal weight. Due to the nature of the `alfworld` environment, where each subtask does not appear with equal probability, the average performance on `alfworld` is a weighted average among all types of tasks. We mark the BUTLER$_g$ and BUTLER agent [Shridhar et al., 2021] in gray since they require expert data, while the remaining methods do not require expert data. As discussed by Shridhar et al. [2021], the performance discrepancy between BUTLER$_g$ and BUTLER happens due to different decoding strategies in evaluation strategies: BUTLER$_g$ uses greedy decoding, which may repeat failed actions, whereas BUTLER employs beam search during evaluation.

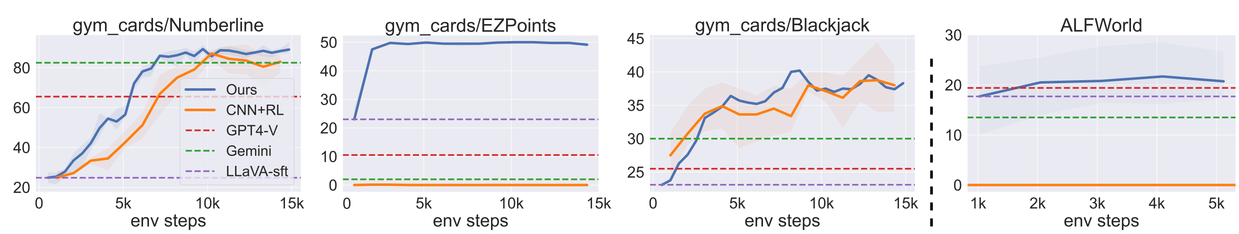


Figure 4.5: **Episode success rates (%) of different methods on `gym_cards` and `alfworld` during training**. Left to right: `gym_cards/Numberline`, `gym_cards/EZPoints`, `gym_cards/Blackjack`, and `alfworld` (all). The curves of `Points24` are not included because none of the tested methods achieve reasonable performance.

across various tasks, including deterministic (`NumberLine` and `EZPoints`)[5] or stochastic (`Blackjack`) arithmetic tasks and visual semantic reasoning task (`alfworld`). Specifically, our method improves the average performance from the initial LLaVA-sft model by **27.1%** on arithmetic tasks (18.4% → 45.5%) and **4.0%** on visual semantic decision-making task (17.7% → 21.7%). In addition, our method also achieves the best performance among all comparative methods, surpassing the second-best method by 14.0% (CNN+RL) on

[5]Although `Points24` shares similar rules with `EZPoints`, it requires the VLM to recognize all four cards and generate much longer equations. Most failure cases in `Points24` are caused by either inaccurate visual perception or flawed language reasoning. We provide some examples of these failures in Appendix C.2.

gym_cards and 2.3% (GPT4-V) on alfworld.

Understanding the Role of the CoT Reasoning

In Section 4.6, we have demonstrated that our method improves the arithmetic and visual semantic reasoning capabilities of VLM agents. Conceptually, our method can be viewed as an augmented version of the standard CNN-based RL, where the text output $[v^{\text{tht}}, v^{\text{act}}]$ (from Figure 4.3) serve as the text action v^{act}, augmented by CoT reasoning v^{tht}. This raises an important question: How does the CoT reasoning v^{tht} influence the overall performance of our method? To assess the impact of CoT reasoning on our method's performance, we conduct two sets of ablation experiments. The first set (presented in Table 4.3 and Figure 4.6) evaluates our method without the CoT reasoning, and the second part (shown in Figure 4.7) examines various scaling hyperparameters λ for the log-likelihood of CoT tokens, as defined in Equation 4.3.

	gym_cards					alfworld						
CoT	NL	EZP	P24	BJ	Avg.	Pick	Examine	Clean	Heat	Cool	Pick 2	Avg.
✓	**89.4**	**50.0**	**2.3**	40.2	**45.5**	**47.4**	**14.7**	**10.4**	**14.4**	**18.8**	**18.0**	**21.7**
✗	26.9	29.9	0	**40.4**	24.3	40.5	12.0	2.8	8.5	14.4	17.7	16.3
✓-✗	**+62.5**	**+20.1**	**+2.3**	**-0.2**	**+21.2**	**+6.9**	**+2.7**	**+7.6**	**+5.9**	**+4.4**	**+0.3**	**+5.4**

Table 4.3: **Episode success rates (%) of our method with and without CoT reasoning.** We report the *best* results from Figure 4.6 (first 15k environment steps for the gym_cards and 5k environment steps for alfworld).

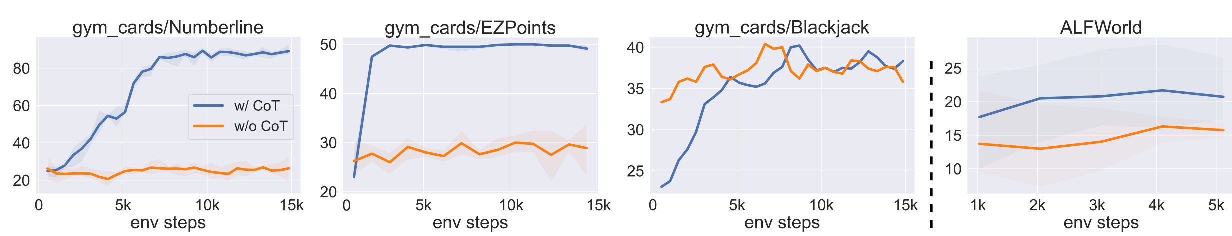


Figure 4.6: **Training curves of our method without and without the CoT reasoning.** Left to right: gym_cards/Numberline, gym_cards/EZPoints, gym_cards/Blackjack, and alfworld (all). The curves of Points24 are not included because none of the tested methods achieve reasonable performance.

The crucial role of CoT reasoning in performance improvement. As presented in Table 4.3 and Figure 4.6, the performance of our method significantly decreases without the

CoT reasoning.[6] Besides the improvement in the final performance, CoT reasoning appears to be a crucial component for deterministic arithmetic tasks (`NumberLine` and `EZPoints`), as our method fails to improve these two tasks without the CoT reasoning.

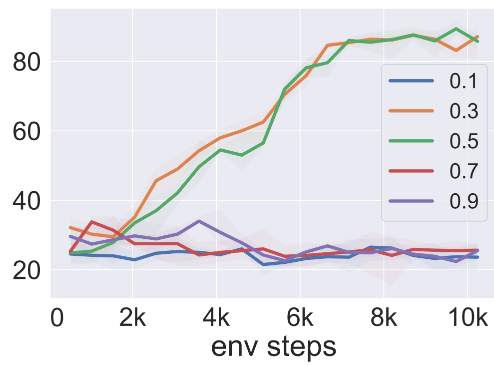


Figure 4.7: **Episode success rates (%) of our method under different λ on `NumberLine`.**

The importance of moderate scaling factors λ. As discussed in Section 4.4, integrating CoT reasoning into our framework involves tuning an additional hyperparameter, $\lambda \in [0, 1]$ (proposed in Equation 4.3). To identify an optimal range for λ, we conduct experiments assessing the impact of various λ. Our results in Figure 4.7 indicate that a moderate λ (between 0.3 and 0.5) enables effective training on the `NumberLine` task. Conversely, our method fails when λ is set too large (≥ 0.7) or too small (≤ 0.1), and we empirically find that an optimal λ typically falls within 0.2 to 0.5. This is because a large λ results in an estimate of $\log \pi_\theta(a_t|o_t, \boldsymbol{v}_t^{\text{in}})$ being overly influenced by $\log \pi_\theta(\boldsymbol{v}_t^{\text{tht}}|o_t, \boldsymbol{v}_t^{\text{in}})$, while a small λ value causes π_θ to be predominantly affected by $\log \pi_\theta(\boldsymbol{v}_t^{\text{act}}|o_t, \boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{tht}})$, thereby reducing the effect of the CoT reasoning in RL training.

4.7 Conclusions, Limitations, and Future Directions

In project, we introduce an algorithmic framework that directly fine-tunes VLMs using RL, with the help of the VLM's CoT reasoning capability. Empirical results demonstrate that our method can enhance the decision-making abilities of VLMs across diverse domains that require fine-grained visual recognition or visual semantic understanding. In addition, we demonstrate that CoT reasoning is a crucial component for enabling RL training, allowing 7b VLMs to outperform established commercial models such as GPT-4V and Gemini on most tasks. While our results suggest that CoT reasoning is crucial to the performance improvement of VLM training with RL, we have not extensively explored the

[6]Except for the `Blackjack` task, where the peak performance without CoT is slightly better (+0.2%).

effects of different prompting techniques in this work, which will be an interesting future direction. The performance gain of our method is also limited by the size of the action space and the difficulties of the task. For example `alfworld` does not enjoy as much performance gain as `gym_cards`, since `alfworld` is a multi-task environment and it has a much larger action space than `gym_cards`.

Chapter 5

Understanding Foundation Model Post-Training

5.1 Introduction

While SFT and RL are both widely used for foundation model training [OpenAI, 2023d, Google, 2023c, Jaech et al., 2024, DeepSeekAI et al., 2025], their distinct effects on *generalization* [Bousquet and Elisseeff, 2000, Zhang et al., 2021a] remain unclear, which makes it challenging to build reliable and robust AI systems. A key challenge in analyzing the generalization ability of foundation models [Bommasani et al., 2021, Brown et al., 2020] is separating data memorization[1] from the acquisition of transferable principles. We therefore investigate the key question of whether SFT or RL primarily memorize the training data [Allen-Zhu and Li, 2023a, Ye et al., 2024, Kang et al., 2024], or whether they learn generalizable principles that can adapt to novel task variants.

To address this question, we focus on two aspects of generalization: textual rule-based generalization and visual generalization. For textual rules, we study a model's ability to apply learned rules (given text instructions) to variants of those rules [Zhu et al., 2023b, Yao et al., 2024, Ye et al., 2024]. For vision-language models (VLMs), visual generalization measures performance consistency to variations in visual input, such as color and spatial layout, within a given task. For studying text-based and visual generalization, we investigate two different tasks that embody rule-based and visual variants. Our first task is `GeneralPoints`, an original card game task that is similar to the `Points24` task from RL4VLM [Zhai et al., 2024a], which is designed to evaluate a model's *arithmetic reasoning capabilities*. In `GeneralPoints`, the model receives four cards (presented as a text description or an image), and is required to compute a target number (24 by default) using each card's numerical value exactly once. Second, we adopt `V-IRL` [Yang et al., 2024a], a real-

[1]We use "memorization" the refer a model's capacity to generate near-exact copies of training examples when prompted based on information present in the training dataset. This definition explicitly excludes bit-wise or code-wise replication of training data within the model itself.

world navigation task, that focuses on the model's *spatial reasoning capabilities*.

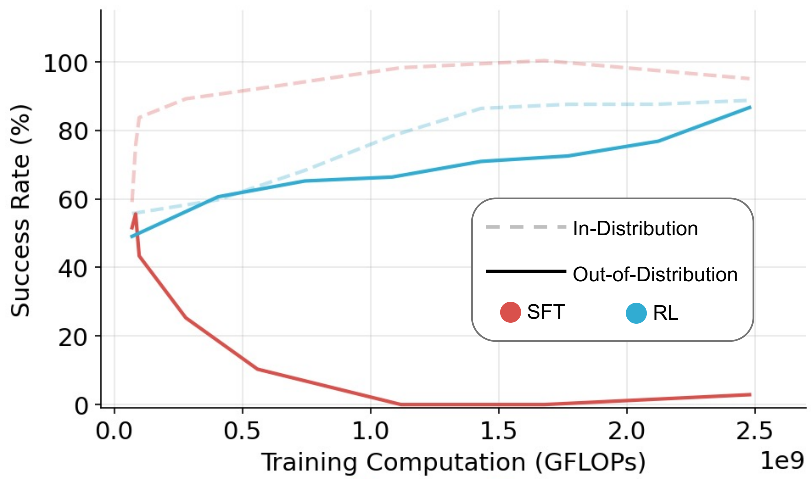


Figure 5.1: **A comparative study of RL and SFT on the visual navigation environment `V-IRL` [Yang et al., 2024a] for OOD generalization [Chu et al., 2025].** OOD curves represent performance on the same task, using *a different textual action space*. See detailed descriptions of the task in Section 5.5.

We adopt a multi-step RL framework similar to Zhai et al. [2024a], by instantiating RL after running SFT on the backbone model [Dubey et al., 2024], using the sequential revision formulation [Snell et al., 2024]. In both `GeneralPoints` and `V-IRL`, we observe that RL learns generalizable rules (expressed in text), where in-distribution performance gains also transfer to unseen rules. In contrast, SFT appears to memorize the training rules and fails to generalize (see Figure 5.1 for an example). Beyond textual rule-based generalization, we further investigate generalization in the visual domain and observe that RL also generalizes to visual OOD tasks, whereas SFT continues to struggle. As a by-product of the visual OOD generalization capability, our multi-turn RL approach **achieves state-of-the-art performance** on the `V-IRL` mini benchmark, by **+33.8%** (44.0%→77.8%) [Yang et al., 2024a], highlighting the generalization capability of RL. To understand how RL impacts a model's visual abilities, we conduct additional analysis on `GeneralPoints`, revealing that training RL with an outcome-based reward function [Cobbe et al., 2021] improves visual recognition capabilities. While RL exhibits superior generalization compared to SFT, we show that SFT is still helpful for stabilizing the model's output format, enabling RL to achieve its performance gains. Last but not least, we observe that scaling up the inference time compute by increasing the number of maximal steps leads to better generalization.

5.2 Related Works

Post-training. Post-training is crucial for enhancing model performance [Zhang et al., 2022, Hoffmann et al., 2023, OpenAI, 2023d, Google, 2023c, Touvron et al., 2023a]. This stage commonly utilizes large-scale supervised fine-tuning (SFT) [Radford et al., 2018, Brown et al., 2020, Radford et al., 2021, Wei et al., 2022a, Chung et al., 2022, Zhou et al., 2024a] and/or reinforcement learning (RL) [Ziegler et al., 2019, Ouyang et al., 2022, Sun et al., 2024, Abdulhai et al., 2023, Zhou et al., 2024b, Zhai et al., 2024a]. SFT adapts pre-trained models to downstream tasks by training them on task-specific, often instruction-formatted datasets. Previous work, such as FLAN [Wei et al., 2022a], demonstrates that fine-tuning on diverse instruction-tuning datasets significantly enhances zero-shot performance on unseen tasks. Furthermore, LIMA [Zhou et al., 2024a] shows that supervised fine-tuning acts as a "format teacher" effectively adapting the model's responses to a desired format while leveraging the capabilities of pre-trained LLMs. In contrast, RL [Ziegler et al., 2019, Ouyang et al., 2022, Sun et al., 2024, Ramamurthy et al., 2023, Abdulhai et al., 2023, Zhou et al., 2024b, Zhai et al., 2024a] has been primarily used to align models with human preferences or training the foundational model to solve a specific task [Abdulhai et al., 2023, Zhou et al., 2024b, Zhai et al., 2024a, Chen et al., 2024b]. Our work differs from prior studies, as we aim to comparatively analyze the generalization and memorization of SFT and RL on *both LLM and VLM*, while previous studies have focused primarily on only one of these two post-training methods (or only study LLM or VLM) or on only one post-training method.

Memorization and generalization in LLM/VLM. Several studies have examined the interplay between memorization and generalization in neural networks [Han et al., 2022, Carlini et al., 2022, Yang et al., 2023d]. In LLMs, memorization can manifest as the model memorizing the training data [Carlini et al., 2022, Jiang et al., 2024, Kang et al., 2024], while generalization reflects the divergence between the model's output distribution and the pre-training data distribution [Zhang et al., 2023a]. Prior studies suggest that LLMs exhibit more overfitting on simpler, knowledge-intensive tasks and greater generalization on more complex, reasoning-intensive ones [Wang et al., 2024, Qi et al., 2024]. For example, recent studies [Ye et al., 2024, Allen-Zhu, 2024, Allen-Zhu and Li, 2023a,b, 2024, Tong et al., 2024b] have demonstrated that LLMs develop reasoning skill sets beyond their training data by pre-computing reasoning graphs before autoregressive generation, which provides compelling evidence of generalization. Our study takes a different approach by investigating the role of different post-training paradigms on memorization versus generalization in the context of textual ruled-based and visual variants. We conduct comparative studies in both unimodal (LLM) and multimodal (VLM) settings, and demonstrate that RL leads to better generalization performance than SFT.

Scaling up inference-time compute. Recent research has increasingly focused on scaling up inference-time computation to improve model performance [Wei et al., 2022b, Yao et al., 2024, Snell et al., 2024, Jaech et al., 2024]. Early studies [Wei et al., 2022b, Yao et al., 2024] prompted models to generate intermediate reasoning steps and extend the responses before producing a final answer. Subsequent work [Zelikman et al., 2022, Feng et al., 2023, Tian et al., 2024, Chen et al., 2024a, Snell et al., 2024] has demonstrated that fine-tuning verifiers during inference improves model accuracy, effectively utilizing test-time computation. Notably, recent findings [Jaech et al., 2024, DeepSeekAI et al., 2025] reveal "scaling laws" for inference-time compute, highlighting significant performance gains with increased computational resources. Our work builds upon these findings in two ways. First, we integrate insights from inference-time verification into a multi-turn RL formulation that allows the model to identify and correct its errors. Second, we examine the impact of inference-time verification on RL generalization, demonstrating that scaling up inference-time verification (in terms of the maximum number of verification steps) is a key for RL to generalize.

Improving visual capability in VLMs. While VLMs have demonstrated remarkable skill across a wide range of challenging tasks, such as solving advanced college exam questions [Lu et al., 2023, Yue et al., 2024a,b] and spatial understanding tasks [Yang et al., 2024a,b], they also exhibit limitations in visual perception [Zhai et al., 2024a,b, Tong et al., 2024c,d, Rahmanzadehgervi et al., 2024]. Prior efforts to enhance VLMs' visual perception include combining multiple visual encoders [Tong et al., 2024d, Kar et al., 2025, Tong et al., 2024a], curating high-quality SFT data [Chen et al., 2023, Liu et al., 2024b, Tong et al., 2024a], and improving the SFT training recipe by unfreezing the visual backbone [Liu et al., 2023c, Tong et al., 2024a]. While these prior works primarily focus on experiments during the SFT stage, our work demonstrates that RL can also improve visual perception.

5.3 Preliminaries

Standard RL terminology. We consider finite horizon decision making, and adopt standard notation from the classical RL literature [Sutton and Barto, 2018, Agarwal et al., 2019], where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function, and T denotes the maximum number of steps per episode. The goal is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the overall return $\max_{\pi \in \Pi} \mathbb{E}_\pi \left[\sum_{t=0}^{T} r_t\right]$, where r_t denotes $r(s_t, a_t)$. Without loss of generality, we use $\pi(a|s) \in [0, 1]$ to denote probability of π choosing a at s.

Adapting RL terminology to LLM/VLM with a verifier. We adopt a multi-turn RL setting for foundation model training [Zhai et al., 2024a]. Let $\mathcal{V}$ represent the discrete and finite vocabulary (token) space. The input and output text spaces are denoted by $\mathcal{V}^m$ and $\mathcal{V}^n$ respectively, where m and n are the maximum token length of the input sequence $\boldsymbol{v}^{\text{in}}$

and output sequence $\boldsymbol{v}^{\text{out}}$. For models requiring visual inputs (VLM), we define $\mathcal{O}$ as the space of all RGB images. The state space, denoted by $\mathcal{S}$, is defined as $\mathcal{S} := \mathcal{V}^m \times \mathcal{O}$ for VLM, and $\mathcal{S} := \mathcal{V}^m$ for LLM. The action space $\mathcal{A}$ is defined as $\mathcal{A} := \mathcal{V}^n$. We use $\mathsf{VER} : \mathcal{V}^n \to \mathbb{R} \times \mathcal{V}^k$ to denote a verifier, which evaluates the outcome of $\boldsymbol{v}^{\text{out}}$ and generates an outcome-based reward function [Cobbe et al., 2021, Hosseini et al., 2024, Snell et al., 2024, Setlur et al., 2024] r along with textual information $\boldsymbol{v}^{\text{ver}}$. Mathematically, at time t, $\mathsf{VER}(\boldsymbol{v}_t^{\text{out}}) \mapsto (r_t, \boldsymbol{v}_t^{\text{ver}})$. Similar to Zhai et al. [2024a], we treat the model with parameter θ as our policy network $\pi_\theta : \mathcal{S} \to \mathcal{V}^n$, and adopt PPO [Schulman et al., 2017] as the backbone RL algorithm for updating π_θ.

Sequential revision. For modeling the state-action transition, we adopt the sequential revision formulation [Snell et al., 2024]. Specifically, at time step $t = 0$ the initial input $\boldsymbol{v}_0^{\text{in}}$ consists of the system prompt. For subsequent time steps ($t \geq 1$), the input prompt $\boldsymbol{v}_t^{\text{in}}$ comprises the system prompt concatenated with all prior model and verifier outputs, denoted by $[\boldsymbol{v}_k^{\text{out}}, \boldsymbol{v}_k^{\text{ver}}]_{k=0}^{t-1}$. An illustration of the sequential revision is provided in Figure 5.2 (also see Figure 5 of Snell et al. [2024]), and an example of the state-action transition is shown in Figure 5.3.

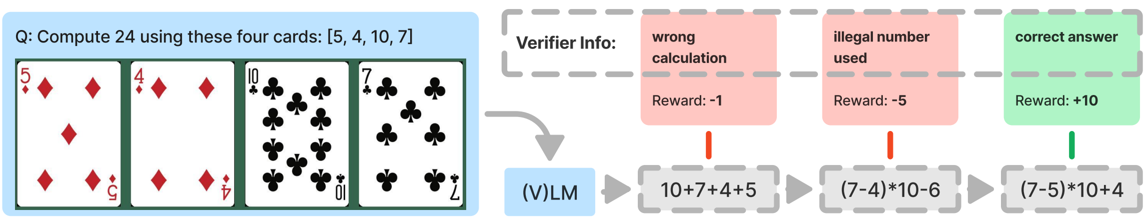


Figure 5.2: **An example of the sequential revision formulation with a verifier.** The model generate the next answer $\boldsymbol{v}_{t+1}^{\text{out}}$ conditioned on *all previous answers and information* $(\boldsymbol{v}_i^{\text{out}}, \boldsymbol{v}_t^{\text{ver}}, 0 \leq i \leq t)$ from the verifier.

5.4 Evaluation Tasks

To evaluate the generalization of different post-training methods, we select two tasks that each offer *rule* and *visual variations*. The first task, `GeneralPoints`, is a new environment we have designed that allows assessment of arithmetic reasoning abilities (Section 5.4). The second task, `V-IRL` [Yang et al., 2024a], is chosen to examine the model's reasoning capabilities in an open-world visual navigation domain (Section 5.4).

The General Points Environment

Our original `GeneralPoints` environment, instantiated on top of the `Points24` environment [Zhai et al., 2024a], is designed to evaluate generalization of arithmetic reasoning.

System Prompt ($\boldsymbol{v}_0^{\text{in}}$)

[Task Description] You are an expert in {task name}, you are observing {purely language/vision-language inputs + <image>}. You are currently at {state related info}. Please follow {tasks rules}.

[Output] Your response should be a valid json file in the following format: {task related information and answer}

Appending previous model and verifier outputs to obtain $\boldsymbol{v}_t^{\text{in}}$

$\boldsymbol{v}_t^{\text{in}} = [\boldsymbol{v}_0^{\text{out}}, \boldsymbol{v}_0^{\text{ver}}, \boldsymbol{v}_1^{\text{out}}, \boldsymbol{v}_1^{\text{ver}}, \dots, \boldsymbol{v}_{t-1}^{\text{out}}, \boldsymbol{v}_{t-1}^{\text{ver}}]$ $\quad \triangleright\ \boldsymbol{v}_t^{\text{in}} = \text{concat}\left(\boldsymbol{v}_0^{\text{in}}, [\boldsymbol{v}_k^{\text{out}}, \boldsymbol{v}_k^{\text{ver}}]_{k=0}^{t-1}\right)$

Model output ($\boldsymbol{v}_t^{\text{out}}$) and **Verifier Output ($\boldsymbol{v}_t^{\text{ver}}$)**

{Task related json outputs}, {You success/fail}. $\quad \triangleright\ \boldsymbol{v}_{t+1}^{\text{in}} = \text{concat}(\boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{out}}, \boldsymbol{v}_t^{\text{ver}})$

Figure 5.3: **An template of our prompt update for constructing $\boldsymbol{v}_{t+1}^{\text{in}}$.** The brown parts marks the task and related information, and the purple parts denote the state (s_t) specific info. The blue and red describe the output from the model and verifier, respectively.

Each state s of the environment contains 4 cards, described as text (in the `GP-L` variant) or presented as an image (in the `GP-VL` variant); see Figure 5.2 left for a visual example of `GeneralPoints`. The goal is to *produce an equation that equals a target number* (24 by default) using all 4 numbers from the cards *exactly once*. Detailed examples of the state-action transitions are provided in Appendix D.1. Note that when input from `GeneralPoints` is presented in an image (`GP-VL`), it naturally introduces additional visual challenges requiring the VLM to *recognize all cards before solving the equation*.

Rule variations. To study whether the model learns arithmetic operations or simply memorizes the post-training data, we introduce rule variations in `GeneralPoints`. These variations consist of interpreting the symbols `'J'`, `'Q'`, and `'K'` either as `'11'`, `'12'`, and `'13'`, respectively, or all as the same number `'10'`. These variations ensure a rigorous evaluation of the model's ability to generalize arithmetic reasoning across diverse settings. Each rule is specified as text in the input prompt, see the {tasks rules} part in Figure 5.3. For studying ruled based generalization, we post-train the model *using one rule*, then *evaluate using a different rule*.

Visual variations. The `GeneralPoints` environment can also be naturally customized to evaluate generalization across visual variants. Since the major visual challenge is to *recognize the number of each card*, agnostic to the *the color* of the cards, we consider the cards with different colors as visual variants of the task. In the visual generalization setting, we train the model using cards of one color, then test OOD performance using the other color.

Figure 5.4: **Demonstration of one navigation task in `V-IRL`.** Agent navigates from place to place following the given linguistic navigation instructions in `V-IRL`. The navigation procedure is shown at the top, with the navigation instructions displayed below. Visual observation-related information is highlighted in green, while action-related information is marked in orange.

The V-IRL Environment

While the `GeneralPoints` environment is designed to assess arithmetic reasoning abilities, we further utilize the `V-IRL` environment [Yang et al., 2024a] to study *spatial reasoning* ability in an open-world navigation domain that uses *realistic visual input*. As in `GeneralPoints` we consider two versions of the environment, one (`V-IRL-L`) that consists of pure language descriptions,[2] and another (`V-IRL-VL`) that includes vision-language input. The major visual challenge in `V-IRL` involves *recognizing different landmarks from the visual observation*[3] before taking an action. The goal is to *navigate to a target location by following a set of instructions that contain spatial information.* A detailed example of one environment step is shown in Appendix D.2.

Rule variations. To evaluate whether the model possesses spatial knowledge or simply memorizes post-training data, we consider two distinct action space configurations. The first variant utilizes an *absolute orientation* action space, which includes {'north', 'northeast', 'east', 'southeast', 'south', 'southwest', 'west', 'northwest'}. The second variant employs a *relative orientation* action space, containing {'left', 'right', 'slightly left', 'slightly right'}. This relative configuration adjusts the current orientation by 90 degrees or 45 degrees to the left or right, respectively. An overview of a navigation task in `V-IRL` is provided in Figure 5.4, and a detailed state-action transition in `V-IRL` is provided in Figure D.3 (in Appendix D.2).

Visual variations. The key visual challenge in `V-IRL` is to recognize landmarks from the visual observations (e.g., the green parts in Figure 5.4). Since the `V-IRL` environment contains visual observations from different cities, we can assess visual generalization in `V-IRL` by training the model to navigate in one location and then evaluate its performance in *different locations*.

5.5 Results

In this section, we present experiments that investigate the generalization abilities induced by post-training with RL and SFT. We adopt Llama-3.2-Vision-11B [Dubey et al., 2024] as the backbone model. Following the standard pipelines of RLHF [Ouyang et al., 2022] and RL4VLM [Zhai et al., 2024a], we initialize the model with SFT before running RL. We specifically study the following questions. Section 5.5: how does SFT or RL affect the model's generalization to different rules? Section 5.5: when the model contains a visual component, how does RL/SFT affect its generalization to different visual variants? Section 5.5: how does RL/SFT affect visual recognition capability in a VLM? Section 5.5: what role does SFT play in RL training? Section 5.5: how does the number of verification iterations affect generalization?

[2]The visual input can be parsed into pure text description, see more details in Yang et al. [2024a] and an illustration of pure text the version in Figure D.4.

[3]See Figure 5.4, the model needs to recognize landmarks like **The Dutch**, **Lola Taverna**, and **Shuka** from the visual observation, and relate these landmarks with the textual instructions for taking the right action.

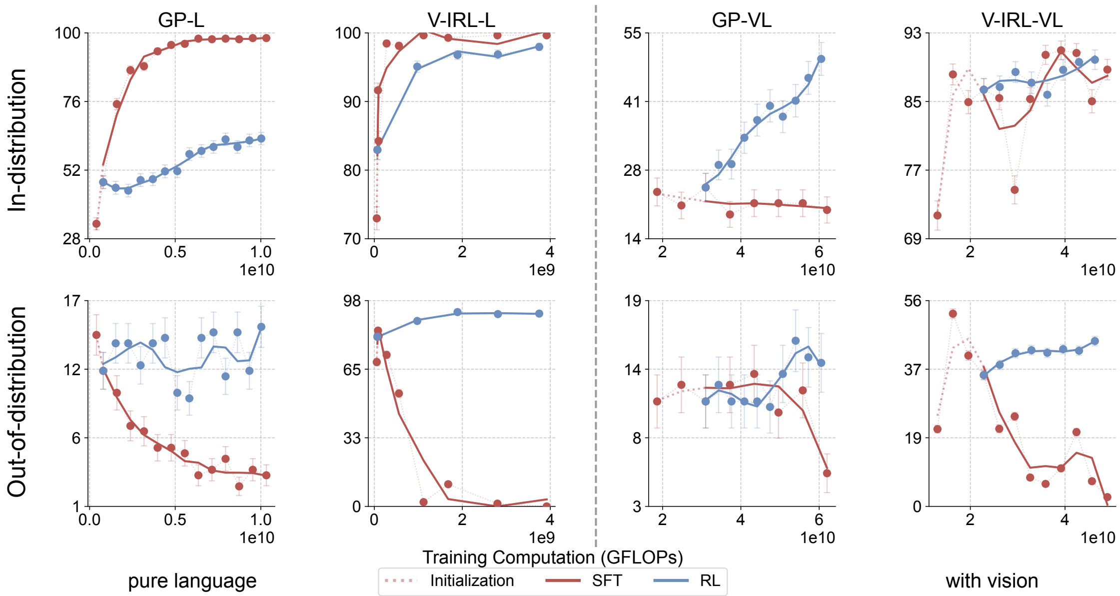


Figure 5.5: **Success rate (%) - GFLOPs trendlines for RL and SFT on `GeneralPoints` and `V-IRL`.** The top row shows in-distribution performance, while the bottom row shows out-of-distribution performance. Results are presented for both pure language (`-L`) and vision-language (`-VL`) variants of each task. For `GeneralPoints`, we report the episode success rate, while for `V-IRL`, we report per-step accuracy with overall success rate in Figures 5.1 and D.8. Detailed evaluation setups (and curve smoothing) are provided in Appendix D.3.

Generalization across Rules

We evaluate the performance of different post-training methods on `GeneralPoints` and `V-IRL`, each of which has a pure language (`-L`) and a vision-language (`-VL`) variant, and each encompassing rule variations. For each task, we separately scale the training compute for RL and SFT on a single rule. We consider the results on the trained rule as in-distribution (ID) performance, whereas results on the *unseen* rules measures out-of-distribution (OOD) generalization. In `GeneralPoints`, the ID case treats all `'J'`, `'Q'`, `'K'` as 10, and the OOD cases interprets them as 11, 12, and 13. As for `V-IRL`, the ID case adopts the *absolute orientation* coordinate system and the OOD case uses the *relative orientation* action space. Other details and additional experimental setup can be found in Appendix D.3.

RL generalizes, SFT memorizes. As illustrated in Figure 5.5, RL consistently improves OOD performance on all tasks, including both unimodal (LLM) and multimodal (VLM). Specifically, Figure 5.6 demonstrates that RL achieves an increase of **+3.5%** on `GP-L` (11.5% $\rightarrow$ 15.0%) and **+11.0%** on `V-IRL-L` (80.8% $\rightarrow$ 91.8%). Even with the additional challenge of visual recognition in the VLM, RL maintains consistent performance improvements of **+3.0%** (11.2% $\rightarrow$ 14.2%) on `GP-VL` and **+9.3%** (35.7% $\rightarrow$ 45.0%) on `V-IRL-VL`, respectively.

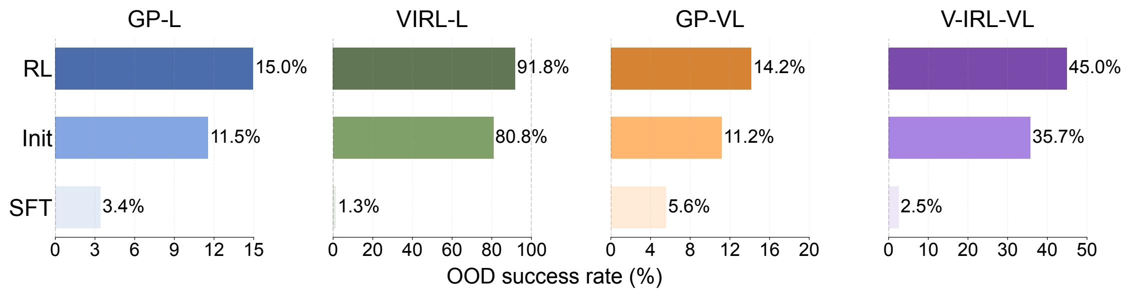


Figure 5.6: **Comparison of out-of-distribution performance under rule variants.** We report the success rate for `GeneralPoints` and **per-step-accuracy** for `V-IRL`. For each subplot, RL and SFT are trained with equal computation, and their shared initial checkpoint (marked as Init) is set as baseline. Detailed setups are provided in Appendix D.3.

In contrast, SFT consistently exhibits performance degradation across all OOD evaluations on all tasks: **-8.1%** on `GP-L` (11.5% → 3.4%), **-79.5%** on `V-IRL-L` (80.8% → 1.3%), **-5.6%** (11.2% → 5.6%) on `GP-VL`, and **-33.2%** (35.7% → 2.5%) on `V-IRL-VL`.

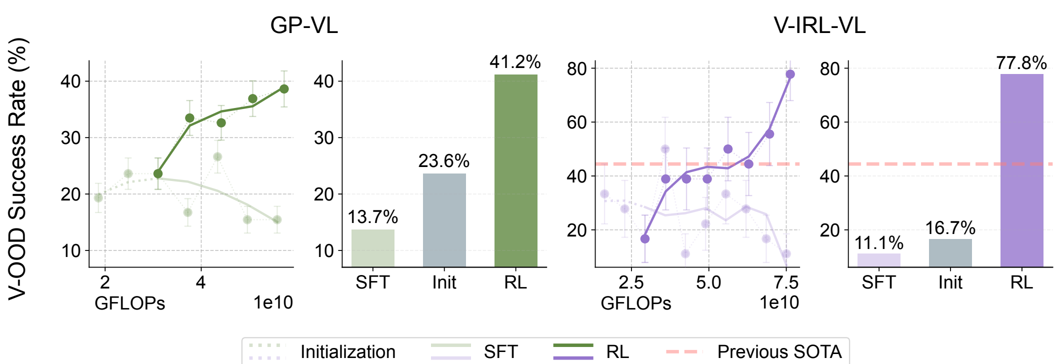


Figure 5.7: **Comparison of out-of-distribution performance under visual variants.** Similar to Figures 5.5 and 5.6, we present both the performance dynamics (shown as lines) and final performance (shown as bars) for visual out-of-distribution evaluations. The previous state-of-the-art on `V-IRL` VLN mini benchmark [Yang et al., 2024a] is marked in orange. Detailed evaluation setups (and curve smoothing) are provided in Appendix D.3.

Generalization in Visual Out-of-Distribution Tasks

Section 5.5 demonstrates that RL yields generalization across rule variations, whereas SFT exhibits the opposite trend. Since VLMs also incorporate a visual modality, we next study the effects of *visual* variation in OOD generalization. For `GeneralPoints`, we train the VLM using the black suits (♠, ♣) and test out-of-distribution performance on the red suits (♥,

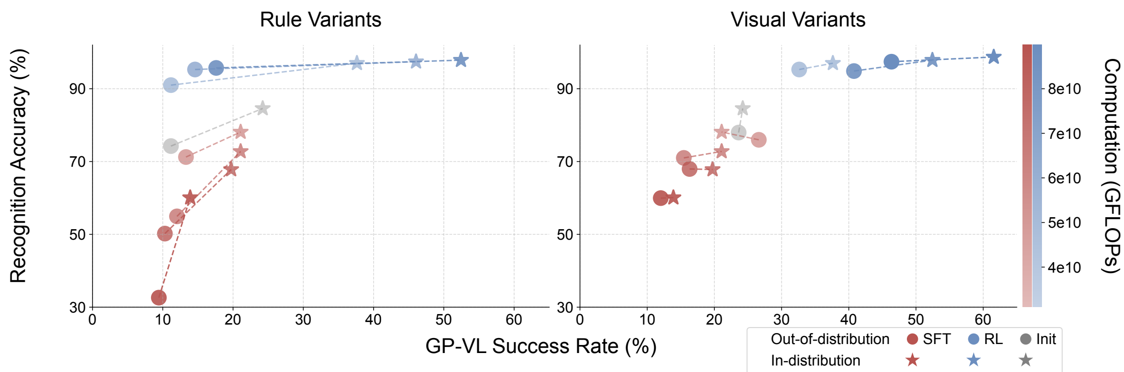


Figure 5.8: **Recognition vs. success rate for RL and SFT under different variants in `GP-VL`.** We report both in-distribution (red) and OOD (blue) performance of recognition (y-axis) and episode success rate (x-axis). We denote the training compute of each data point via transparency (color bar) while connected (⋆-○) pairs are evaluated using same checkpoints. As scaling up post-training compute, RL improves both recognition and overall accuracy, while SFT shows opposite effect.

♦). For `V-IRL`, we train the model on routes collected in New York City and evaluate it on the original `V-IRL` VLN mini benchmark [Yang et al., 2024a] containing routes from *various cities worldwide* (see Appendix D.2 for details). Note that the rules remain consistent across experiments in this section.

RL generalizes in visual OOD tasks. As shown in Figure 5.7, we observe that RL still *generalizes in visual OOD tasks*, while SFT continues to suffer. Specifically, in `GP-VL` and `VIRL-VL`, RL achieves performance improvements of **+17.6%** (23.6% → 41.2%), **+61.1%** (16.7% → 77.8%), whereas SFT suffers from performance decreases of **-9.9%** (23.6% → 13.7%) and **-5.6%** (16.7% → 11.1%). As a byproduct of this visual OOD study, we also show that our multi-turn RL formulation **improves the state-of-the-art results** (see Table 5 of Yang et al. [2024a]) on the `V-IRL` mini benchmark by **+33.8%** (44.0% → 77.8%). Notably, unlike the previous state-of-the-art approach reported in `V-IRL`, which relies on a two stage VLM-LLM collaboration technique and tailored prompt engineering on closed-sourced model [OpenAI, 2023b], our end-to-end RL approach enables an open-sourced model [Dubey et al., 2024] to reach superior performance.

RL Improves Visual Capabilities

Building upon the above observation that VLMs trained with RL generalize to visual OOD tasks (Section 5.5), we consider a natural follow-up question: *How does RL affect VLMs' visual capabilities*? To study this question, we conducted additional ablation studies in the GP-VL environment to investigate the OOD performance of RL and SFT, along with the model's visual recognition accuracy, in terms of recognizing the 4 cards from the input image. In particular, we study how scaling post-training compute via RL/SFT both affects generalization in rule-based OOD (Figure 5.8 left), and visual recognition accuracy and visual OOD (Figure 5.8 right).

Scaling RL improves visual recognition accuracy in VLM training. As shown in Figure 5.8, we observe that the VLM's visual recognition accuracy largely affects the overall performance, which was similarly observed in Zhong et al. [2024]. In addition, scaling up RL compute also improves visual recognition accuracy, as a byproduct of its generalization capability, while scaling SFT deteriorates both visual recognition accuracy and overall performance. Additional experimental results are provided in Figures D.6 and D.7 of Appendix D.4.

The Role of SFT for RL Training

Despite the superiority of RL in generalizing the model's reasoning and visual capabilities, as discussed previously, the experimental pipeline still instantiates RL *after SFT*. In this subsection, we focus on another key question: *Is SFT necessary for RL training*? To answer this question, we conduct additional experiments that directly apply end-to-end RL to post-train the base model Llama3.2 using `GeneralPoints` in the purely language case (Figure 5.9).

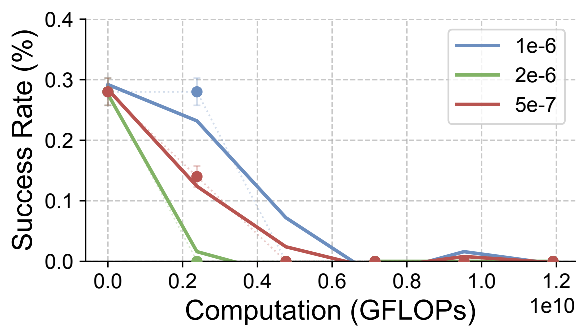


Figure 5.9: **RL experiments on `GP-L` without SFT initialization.** All trials fail due to poor instruction following capability of the base model.

SFT is necessary for RL training when the backbone model does not follow instructions. Figure 5.9 shows that without SFT, all end-to-end RL runs fail to improve. More specifically, we observe that without SFT, the base model suffers from poor instruction following capability. A detailed failure case is provided in Figure D.10 (in Appendix D.4), revealing that the base Llama-3.2-Vision-11B model tends to generate long, tangential, and unstructured responses. This issue makes it impossible to retrieve task-related information and rewards for RL training. Note that due to the difference in backbone model, our results do not contradict with DeepSeekAI et al. [2025], which suggests that SFT is unnecessary for downstream RL training.

Role of Verification Iterations

Verification serves as another crucial component in our multi-step training and evaluation pipeline (see Figures 5.2 and 5.3). To validate its necessity and better understand its effect, we conduct RL experiments with different verification iterations $\{1, 3, 5, 10\}$ using `GP-L` (Figure 5.10).

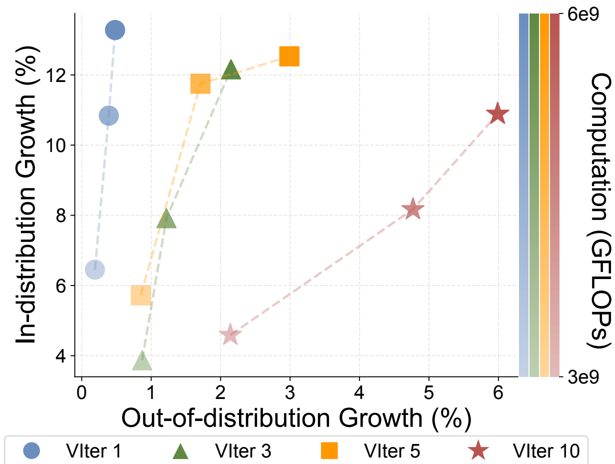


Figure 5.10: **In-distribution vs. OOD performance growth on `GP-L`.** We record RL experiments with different number of verification iterations (VIter) as scaling up training compute (color transparency).

Scaling up verification improves generalization. In Figure 5.10, we observe that RL generalizes better with more verification steps. More specifically, under the same computational budget across all experiments, we observe improvements of **+2.15%** (3 steps), **+2.99%** (5 steps), **+5.99%** (10 steps). In contrast, in the case with one verification step, we only observe a marginal improvement of **+0.48%** in OOD performance improvement.

5.6 Conclusion, Discussion and Limitations

In this project, we presented a comprehensive analysis of the generalization effects of foundation model post-training techniques, specifically RL and SFT. Through extensive experiments on the `GeneralPoints` and `V-IRL` tasks, we demonstrated that RL exhibits superior performance in learning generalizable knowledge, while SFT tends to merely memorize the training data, across both the rule and visual variations. This phenomenon consistently occurs across multimodal arithmetic and spatial reasoning capabilities. In addition, we studied the effect of RL on visual recognition, the role of SFT, and the role of verification steps. During our study, two challenges were not resolved.

Failure of SFT on `GP-VL`. In Figure 5.5 for `GP-VL`, we observe that SFT fails to achieve a comparable in-distribution performance with RL. To mitigate the variance introduced by hyperparameter choices, we additionally conduct 10 more experiments with different learning rates and tunable components (Figure D.6), none of which exhibits a strong increasing trend like RL (Figure D.7). Given our observation that scaling up SFT degrades visual recognition capabilities (Figure 5.8), we hypothesize that SFT locally overfits to reasoning tokens while neglecting recognition tokens, possibly due to the higher frequency of reasoning tokens (see Figure D.1 as example). We leave further investigation to future work.

Limits of RL in corner cases. As discussed in Section 5.5, SFT is necessary for effective RL training on Llama-3.2. We investigate applying RL to an overly-tuned SFT checkpoint. As demonstrated in Figure D.9, RL is unable to recover out-of-distribution performance when starting from such a checkpoint. Example failure cases are illustrated in Figure D.11, where the model collapses to the training rule. These results, together with findings in Section 5.5, indicate that RL has limited effectiveness when applied to extremely underfit or overfit initial checkpoints. Further research is needed to delineate the conditions under which SFT facilitates effective RL.

Chapter 6

Conclusion, Discussion, and Future Directions

6.1 Summary of Key Contributions

This dissertation has explored the crucial, yet often overlooked, role of domain knowledge in the training and generalization capabilities of multimodal foundation models. We moved beyond the prevailing focus on supervised fine-tuning (SFT) and demonstrated its limitations, particularly its tendency to overfit to training data and suffer from catastrophic forgetting. In contrast, we showed that reinforcement learning (RL) fine-tuning offers a powerful alternative, fostering the development of models that can generalize effectively to unseen scenarios.

This research makes several key contributions. First, the Evaluating MulTimodality (EMT) framework provides a valuable tool for systematically assessing catastrophic forgetting in MLLMs. Second, the MultiModal Visual Patterns (MMVP) benchmark exposes specific visual shortcomings of current models and reveals a critical link between these weaknesses and the capabilities of the underlying visual encoders. Third, a novel and practical framework was developed for fine-tuning large VLMs with RL. This framework incorporates chain-of-thought (CoT) prompting to enhance reasoning and exploration, enabling end-to-end training on tasks that demand both visual and linguistic understanding. Finally, and perhaps most importantly, this dissertation provides compelling empirical evidence that RL fine-tuning significantly outperforms SFT in terms of generalization, enabling models to adapt to novel rule variations, unseen visual contexts, and even improve their underlying visual recognition abilities. The SFT is also important to ensure the model has a decent instruction following capability.

6.2 Discussion of Implications

The findings presented in this dissertation have far-reaching implications for the future of multimodal learning and the pursuit of more robust and generalizable AI. The consistent observation of overfitting and catastrophic forgetting with SFT, across diverse tasks and model architectures, highlights a fundamental limitation of this widely used approach. While SFT remains valuable for adapting models to specific task formats and datasets, it is insufficient for achieving the kind of adaptable intelligence required for real-world deployment. The "memorization" effect, where models become overly reliant on the specific patterns encountered during training, severely restricts their ability to handle variations in input or task requirements.

In stark contrast, the success of RL fine-tuning underscores the power of learning through interactions. By allowing models to actively explore their environment and receive rewards based on their actions, RL encourages the acquisition of underlying principles and strategies that are not tied to specific training examples. The incorporation of CoT prompting further amplifies this effect, enabling more efficient exploration and facilitating the development of reasoning capabilities.

This research also highlights the critical role of the visual encoder. The MMVP benchmark, designed to pinpoint specific visual weaknesses, demonstrated a strong correlation between limitations in the visual encoder (specifically, CLIP) and the overall performance of the MLLM. This suggests that future progress in multimodal learning will depend not only on advances in language modeling and RL techniques but also on the development of more robust and capable visual representations. Furthermore, the experiments with varying numbers of verification iterations revealed the importance of inference-time computation. Allowing the model to refine its responses through multiple rounds of self-assessment, guided by a verifier, significantly boosted both in-distribution and out-of-distribution performance. This underscores the value of incorporating mechanisms for self-reflection and correction into the model's architecture and training process.

6.3 Limitations

Despite the significant insights gained, this research is subject to certain limitations. The computational demands of training large multimodal models, especially with RL, restricted the scale of some experiments and the exploration of even larger model architectures. The evaluation, while encompassing diverse tasks (arithmetic reasoning, visual navigation, basic image classification), could be extended to a broader range of multimodal challenges. Additionally, while the chosen baselines represent established approaches, the rapidly evolving nature of the field means that newer, more advanced models and techniques are constantly emerging. The hyper-param tuning for the RL training part can still be improved.

6.4 Promising Future Research Directions

This dissertation lays the groundwork for several exciting avenues of future research. Scaling up RL training to even larger models and more complex environments is a paramount challenge. This will likely require the development of more efficient RL algorithms, the exploration of distributed training techniques, and the design of novel reward shaping strategies that can effectively guide learning in high-dimensional, sparsely rewarded settings.

Addressing the identified visual shortcomings of current MLLMs is another critical area for future work. This could involve creating new visual encoder architectures, incorporating more diverse and representative visual training data, and exploring alternative self-supervised learning methods that can capture more nuanced and detailed visual information.

Further investigation into hybrid approaches that combine the strengths of RL and SFT is also warranted. SFT could potentially be used for initial task adaptation or for shaping the format of the model's responses, while RL could then be employed to enhance generalization and robustness.

The development of more comprehensive and challenging benchmarks is essential for driving progress in the field. These benchmarks should go beyond simple accuracy metrics and assess a wider range of capabilities, including reasoning, generalization, robustness to adversarial examples, and explanability. Finally, exploring alternative verification methods, such as incorporating human feedback or leveraging external knowledge sources, could significantly improve the effectiveness of RL training and lead to more reliable and trustworthy models. The prompt can also be improved. The theoretical analysis is another direction.

6.5 Concluding Remark

This dissertation has demonstrated that while current multimodal foundation models exhibit impressive capabilities, they are not yet capable of achieving truly generalizable intelligence. By exposing the limitations of supervised fine-tuning and showcasing the power of reinforcement learning, this work contributes to a deeper understanding of the principles governing learning and generalization in these complex systems. The findings and methodologies presented here provide a foundation for building more robust, adaptable, and ultimately, more intelligent multimodal AI.

Bibliography

ShareGPT, 2023. URL https://sharegpt.com/.

Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. Lmrl gym: Benchmarks for multi-turn reinforcement learning with language models. *arXiv preprint arXiv:2311.18232*, 2023.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 2019.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

Zeyuan Allen-Zhu. ICML 2024 Tutorial: Physics of Language Models, July 2024. Project page: https://physics.allen-zhu.com/.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023a.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.2, knowledge manipulation. *arXiv preprint arXiv:2309.14402*, 2023b.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. *arXiv preprint arXiv:2404.05405*, 2024.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *CVPR*, 2023.

Anas Awadalla, Irena Gao, Joshua Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Gadre, Jenia Jitsev, Simon Kornblith, Pang Wei Koh, Gabriel Ilharco, Mitchell Wortsman, and Ludwig Schmidt. Openflamingo, March 2023. URL https://doi.org/10.5281/zenodo.7733589.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

Kate Baumli, Satinder Baveja, Feryal Behbahani, Harris Chan, Gheorghe Comanici, Sebastian Flennerhag, Maxime Gazeau, Kristian Holsheimer, Dan Horgan, Michael Laskin, et al. Vision-language models as a source of rewards. *arXiv preprint arXiv:2312.09187*, 2023.

Tina Behnia, Ganesh Ramachandra Kini, Vala Vakilian, and Christos Thrampoulidis. On the implicit geometry of cross-entropy parameterizations for label-imbalanced data. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022. URL https://openreview.net/forum?id=1piyfD_ictW.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, jun 2013.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

William Berrios, Gautam Mittal, Tristan Thrush, Douwe Kiela, and Amanpreet Singh. Towards language models that can see: Computer vision through the lens of natural language. *arXiv preprint arXiv:2306.16410*, 2023.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Olivier Bousquet and André Elisseeff. Algorithmic stability and generalization performance. In *Advances in neural information processing systems*, volume 13, 2000.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Mu Cai, Zeyi Huang, Yuheng Li, Haohan Wang, and Yong Jae Lee. Leveraging large language models for scalable vector graphics-driven image understanding. *arXiv preprint arXiv:2306.06094*, 2023.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. Quantifying memorization across neural language models. *arXiv preprint arXiv:2202.07646*, 2022.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023.

Louis Castricato, Alex Havrilla, Shahbuland Matiana, Duy V. Phung, Aman Tiwari, Jonathan Tow, and Maksym Zhuravinsky. trlX: A scalable framework for RLHF, June 2023. URL https://github.com/CarperAI/trlx.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. AlphaMath almost zero: Process supervision without process. *arXiv preprint arXiv:2405.03553*, 2024a.

Jie Chen, Xintian Han, Yu Ma, Xun Zhou, and Liang Xiang. Unlock the correlation between supervised fine-tuning and reinforcement learning in training code large language models. *arXiv preprint arXiv:2406.10305*, 2024b.

Lin Chen, Jisong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. ShareGPT4V: Improving large multi-modal models with better captions. *arXiv preprint arXiv:2311.12793*, 2023.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online, November 2020a. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-main.634.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020b.

William Chen, Oier Mees, Aviral Kumar, and Sergey Levine. Vision-language models provide promptable representations for reinforcement learning. *arXiv preprint arXiv:2402.02651*, 2024c.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Marc-Alexandre Côté, Akos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised Selected Papers 7*, pages 41–75. Springer, 2019.

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

DeepSeekAI et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

Xinshuai Dong, Anh Tuan Luu, Min Lin, Shuicheng Yan, and Hanwang Zhang. How should pre-trained language models be fine-tuned towards adversarial robustness? *Advances in Neural Information Processing Systems*, 34:4356–4369, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

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

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.

Alex Fang, Albin Madappally Jose, Amit Jain, Ludwig Schmidt, Alexander Toshev, and Vaishaal Shankar. Data filtering networks. In *ICLR*, 2024.

Cong Fang, Hangfeng He, Qi Long, and Weijie J Su. Exploring deep neural networks via layer-peeled model: Minority collapse in imbalanced training. *Proceedings of the National Academy of Sciences*, 118(43):e2103091118, 2021.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. AlphaZero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023a.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=yf1icZHC-l9.

Hila Gonen and Yoav Goldberg. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *NAACL*, 2019.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Google. Bard, 2023a. URL https://blog.google/technology/ai/bard-google-ai-search-updates/.

Google. Gemini, 2023b. URL https://blog.google/technology/ai/google-gemini-ai/.

DeepMind Google. Introducing gemini: our largest and most capable ai model, 2023c. URL https://blog.google/technology/ai/google-gemini-ai/.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*, 2023.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Junlin Han, Huangying Zhan, Jie Hong, Pengfei Fang, Hongdong Li, Lars Petersson, and Ian Reid. What images are more memorable to machines? *arXiv preprint arXiv:2211.07625*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *NeurIPS*, 2023.

Joey Hong, Sergey Levine, and Anca Dragan. Zero-shot goal-directed dialogue via rl on imagined conversations. *arXiv preprint arXiv:2311.05584*, 2023a.

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. *arXiv preprint arXiv:2307.12981*, 2023b.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=stmqBSW2dV.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

Cheng-Yu Hsieh, Jieyu Zhang, Zixian Ma, Aniruddha Kembhavi, and Ranjay Krishna. Sugarcrepe: Fixing hackable benchmarks for vision-language compositionality. In *NeurIPS*, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jennifer Hu and Roger Levy. Prompt-based methods may underestimate large language models' linguistic generalizations. In *EMNLP*, 2023.

Jiaxing Huang, Jingyi Zhang, Kai Jiang, Han Qiu, and Shijian Lu. Visual instruction tuning towards general-purpose multimodal model: A survey. *arXiv preprint arXiv:2312.16602*, 2023a.

Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Qiang Liu, et al. Language is not all you need: Aligning perception with language models. *arXiv preprint arXiv:2302.14045*, 2023b.

Shengyi Huang, Quentin Gallouédec, Florian Felten, Antonin Raffin, Rousslan Fernand Julien Dossa, Yanxiao Zhao, Ryan Sullivan, Viktor Makoviychuk, Denys Makoviichuk, Mohamad H Danesh, et al. Open rl benchmark: Comprehensive tracked experiments for reinforcement learning. *arXiv preprint arXiv:2402.03046*, 2024.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *7th Annual Conference on Robot Learning*, 2023c. URL https://openreview.net/forum?id=9_8LF30mOC.

Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 2019.

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Minhao Jiang, Ken Ziyu Liu, Ming Zhong, Rylan Schaeffer, Siru Ouyang, Jiawei Han, and Sanmi Koyejo. Investigating data contamination for pre-training language models. *arXiv preprint arXiv:2401.06059*, 2024.

Heewoo Jun and Alex Nichol. Shap-E: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

Katie Kang, Amrith Setlur, Dibya Ghosh, Jacob Steinhardt, Claire Tomlin, Sergey Levine, and Aviral Kumar. What do learning dynamics reveal about generalization in LLM reasoning? *arXiv preprint arXiv:2411.07681*, 2024.

Oğuzhan Fatih Kar, Alessio Tonioni, Petra Poklukar, Achin Kulshrestha, Amir Zamir, and Federico Tombari. Brave: Broadening the visual encoding of vision-language models. In *European Conference on Computer Vision*, pages 113–132. Springer, 2025.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Tomasz Korbak, Hady Elsahar, German Kruszewski, and Marc Dymetman. Controlling conditional language models without catastrophic forgetting. In *International Conference on Machine Learning*, pages 11499–11528. PMLR, 2022.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`, 2018.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.

Hugo Laurençon, Lucile Saulnier, Léo Tronchon, Stas Bekman, Amanpreet Singh, Anton Lozhkov, Thomas Wang, Siddharth Karamcheti, Alexander M Rush, Douwe Kiela, et al. Obelisc: An open web-scale filtered dataset of interleaved image-text documents. *arXiv preprint arXiv:2306.16527*, 2023.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HkgaETNtDB.

Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*, 2023a.

Chunyuan Li. Large multimodal models: Notes on cvpr 2023 tutorial. *arXiv preprint arXiv:2306.14895*, 2023.

Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, and Jianfeng Gao. Multimodal foundation models: From specialists to general-purpose assistants. *arXiv preprint arXiv:2309.10020*, 1(2):2, 2023b.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023c.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023d.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Yong Lin, Lu Tan, Hangyu Lin, Zeming Zheng, Renjie Pi, Jipeng Zhang, Shizhe Diao, Haoxiang Wang, Han Zhao, Yuan Yao, et al. Speciality vs generality: An empirical study on catastrophic forgetting in fine-tuning foundation models. *arXiv preprint arXiv:2309.06256*, 2023.

Fuxiao Liu, Tianrui Guan, Zongxia Li, Lichang Chen, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. Hallusionbench: You see what you think? or you think what you see? an image-context reasoning benchmark challenging for gpt-4v (ision), llava-1.5, and other multi-modality models. *arXiv preprint arXiv:2310.14566*, 2023a.

Fuxiao Liu, Kevin Lin, Linjie Li, Jianfeng Wang, Yaser Yacoob, and Lijuan Wang. Aligning large multi-modal model with robust instruction tuning. *arXiv preprint arXiv:2306.14565*, 2023b.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023c.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023d. URL https://openreview.net/forum?id=w0H2xGHlkw.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, 2024a.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. LLaVA-NeXT: Improved reasoning, ocr, and world knowledge, 2024b. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023e.

I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Chaochao Lu, Chen Qian, Guodong Zheng, Hongxing Fan, Hongzhi Gao, Jie Zhang, Jing Shao, Jingyi Deng, Jinlan Fu, Kexin Huang, et al. From gpt-4 to gemini and beyond: Assessing the landscape of mllms on generalizability, trustworthiness and causality through four modalities. *arXiv preprint arXiv:2401.15071*, 2024.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. *ICLR*, 2023.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *CVPR*, 2016.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. OK-VQA: A visual question answering benchmark requiring external knowledge. In *CVPR*, 2019.

Chandler May, Alex Wang, Shikha Bordia, Samuel R Bowman, and Rachel Rudinger. On measuring social biases in sentence encoders. In *NAACL*, 2019.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

Microsoft. newbing, 2023.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. *International Conference on Document Analysis and Recognition*, 2019.

Dustin G Mixon, Hans Parshall, and Jianzong Pi. Neural collapse with unconstrained features. *arXiv preprint arXiv:2011.11619*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=nzpLWnVAyah.

Norman Mu, Alexander Kirillov, David Wagner, and Saining Xie. Slip: Self-supervision meets language-image pre-training. In *ECCV*, 2022.

Yao Mu, Qinglong Zhang, Mengkang Hu, Wenhai Wang, Mingyu Ding, Jun Jin, Bin Wang, Jifeng Dai, Yu Qiao, and Ping Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *arXiv preprint arXiv:2305.15021*, 2023.

OpenAI. Chatgpt, 2022. URL https://openai.com/blog/chatgpt.

OpenAI. GPT-4V(ision) System Card, 2023a. URL https://cdn.openai.com/papers/GPTV_System_Card.pdf.

OpenAI. Gpt-4, 2023b. URL https://openai.com/research/gpt-4.

OpenAI. Gpt-4v, 2023c. URL https://openai.com/research/gpt-4v-system-card.

R OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023d.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *TMLR*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. In *COLM*, 2024.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*, 2023.

Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22, 2023.

Zhenting Qi, Hongyin Luo, Xuliang Huang, Zhuokai Zhao, Yibo Jiang, Xiangjun Fan, Himabindu Lakkaraju, and James Glass. Quantifying generalization complexity for large language models. *arXiv preprint arXiv:2410.01769*, 2024.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *ICML 2023 Workshop The Many Facets of Preference-Based Learning*, 2023. URL https://openreview.net/forum?id=53HUHMvQLQ.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.

Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind. In *Proceedings of the Asian Conference on Computer Vision*, pages 18–34, 2024.

Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=8aHzds2uUyB.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506, 2020.

Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. In *NeurIPS*, 2021.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=N0I2RtD8je.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. In *NeurIPS*, 2022.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-OKVQA: A benchmark for visual question answering using world knowledge. In *ECCV*, 2022.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, 2018.

Bokui Shen, Fei Xia, Chengshu Li, Roberto Martín-Martín, Linxi Fan, Guanzhi Wang, Claudia Pérez-D'Arpino, Shyamal Buch, Sanjana Srivastava, Lyne Tchapmi, et al. igibson 1.0: A simulation environment for interactive tasks in large realistic scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7520–7527. IEEE, 2021.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. Textcaps: a dataset for image captioning with reading comprehension, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *CVPR*, 2019.

Mannat Singh, Quentin Duval, Kalyan Vasudev Alwala, Haoqi Fan, Vaibhav Aggarwal, Aaron Adcock, Armand Joulin, Piotr Dollár, Christoph Feichtenhofer, Ross Girshick, et al. The effectiveness of MAE pre-pretraining for billion-scale pretraining. In *ICCV*, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Charlie Victor Snell, Ilya Kostrikov, Yi Su, Sherry Yang, and Sergey Levine. Offline RL for natural language generation with implicit language q learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=aBH_DydEvoH.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. Eva-clip: Improved training techniques for clip at scale. *arXiv preprint arXiv:2303.15389*, 2023a.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. Mitigating gender bias in natural language processing: Literature review. In *ACL*, 2019.

Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, et al. Aligning large multimodal models with factually augmented rlhf. *arXiv preprint arXiv:2309.14525*, 2023b.

Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liangyan Gui, Yu-Xiong Wang, Yiming Yang, Kurt Keutzer, and Trevor Darrell. Aligning large multimodal models with factually augmented RLHF. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13088–13110, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.775. URL https://aclanthology.org/2024.findings-acl.775.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazoure, Walter Talbott, Katherine Metcalf, Natalie Mackraz, Devon Hjelm, and Alexander Toshev. Large language models as generalizable policies for embodied tasks. *arXiv preprint arXiv:2310.17722*, 2023.

Christos Thrampoulidis, Ganesh Ramachandra Kini, Vala Vakilian, and Tina Behnia. Imbalance trouble: Revisiting neural-collapse geometry. *Advances in Neural Information Processing Systems*, 35:27225–27238, 2022.

Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. In *CVPR*, 2022.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. Toward self-improvement of LLMs via imagination, searching, and criticizing. *arXiv preprint arXiv:2404.12253*, 2024.

Shengbang Tong, Erik Jones, and Jacob Steinhardt. Mass-producing failures of multimodal systems with language models. In *NeurIPS*, 2023.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal LLMs. In *NeurIPS*, 2024a.

Shengbang Tong, David Fan, Jiachen Zhu, Yunyang Xiong, Xinlei Chen, Koustuv Sinha, Michael Rabbat, Yann LeCun, Saining Xie, and Zhuang Liu. Metamorph: Multimodal understanding and generation via instruction tuning. *arXiv preprint arXiv:2412.14164*, 2024b.

Shengbang Tong, Erik Jones, and Jacob Steinhardt. Mass-producing failures of multimodal systems with language models. In *NeurIPS*, 2024c.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. *arXiv preprint arXiv:2401.06209*, 2024d.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

Michael Tschannen, Manoj Kumar, Andreas Steiner, Xiaohua Zhai, Neil Houlsby, and Lucas Beyer. Image captioners are scalable vision learners too. *NeurIPS*, 2023.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Kirill Vishniakov, Zhiqiang Shen, and Zhuang Liu. Convnet vs transformer, supervised vs clip: Beyond imagenet accuracy, 2024.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Ruoyao Wang, Peter Alexander Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Conference on Empirical Methods in Natural Language Processing*, 2022. URL https://api.semanticscholar.org/CorpusID:247451124.

Xinyi Wang, Antonis Antoniades, Yanai Elazar, Alfonso Amayuelas, Alon Albalak, Kexun Zhang, and William Yang Wang. Generalization vs memorization: Tracing language models' capabilities back to pretraining data. *arXiv preprint arXiv:2407.14985*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=1PL1NIMMrw.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, 2023c.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=gEZrGCozdqR.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022b.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.

Liang Xie, Yibo Yang, Deng Cai, and Xiaofei He. Neural collapse inspired attraction–repulsion-balanced loss for imbalanced learning. *Neurocomputing*, 527:60–70, 2023.

Hu Xu, Saining Xie, Xiaoqing Ellen Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, and Christoph Feichtenhofer. Demystifying clip data. In *ICLR*, 2024.

Jihan Yang, Runyu Ding, Ellis Brown, Xiaojuan Qi, and Saining Xie. V-IRL: Grounding virtual intelligence in real life. In *European conference on computer vision*, 2024a.

Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. *arXiv preprint arXiv:2412.14171*, 2024b.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*, 2023a.

Yibo Yang, Shixiang Chen, Xiangtai Li, Liang Xie, Zhouchen Lin, and Dacheng Tao. Inducing neural collapse in imbalanced learning: Do we really need a learnable classifier at the end of deep neural network? *Advances in Neural Information Processing Systems*, 35:37991–38002, 2022.

Yibo Yang, Haobo Yuan, Xiangtai Li, Zhouchen Lin, Philip Torr, and Dacheng Tao. Neural collapse inspired feature-classifier alignment for few-shot class-incremental learning. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=y5W8tpojhtJ.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The Dawn of LMMs: Preliminary Explorations with GPT-4V (ision). *arXiv preprint arXiv:2309.17421*, 2023c.

Zitong Yang, Michal Lukasik, Vaishnavh Nagarajan, Zonglin Li, Ankit Singh Rawat, Manzil Zaheer, Aditya Krishna Menon, and Sanjiv Kumar. ResMem: Learn what you can and memorize the rest. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023d. URL https://openreview.net/forum?id=HFQFAyNucq.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.1, grade-school math and the hidden reasoning process. *arXiv preprint arXiv:2407.20311*, 2024.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *arXiv preprint arXiv:2306.13549*, 2023.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*, 2023.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *CVPR*, 2024a.

Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Ming Yin, Botao Yu, Ge Zhang, et al. MMMU-Pro: A more robust multi-discipline multimodal understanding benchmark. *arXiv preprint arXiv:2409.02813*, 2024b.

Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. When and why vision-language models behave like bags-of-words, and what to do about it? In *ICLR*, 2022.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023a.

Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Yi Ma. Investigating the catastrophic forgetting in multimodal large language models. *arXiv preprint arXiv:2309.10313*, 2023b.

Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and Sergey Levine. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=nBjmMF2IZU.

Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Yi Ma. Investigating the catastrophic forgetting in multimodal large language model fine-tuning. In *Conference on Parsimony and Learning*, pages 202–227. PMLR, 2024b.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021a.

Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. Counterfactual memorization in neural language models. *Advances in Neural Information Processing Systems*, 36:39321–39362, 2023a.

Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023b.

Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023c.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. Revisiting few-sample {bert} fine-tuning. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=cO1IH43yUF.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena, 2023.

Ming Zhong, Aston Zhang, Xuewei Wang, Rui Hou, Wenhan Xiong, Chenguang Zhu, Zhengxing Chen, Liang Tan, Chloe Bi, Mike Lewis, et al. Law of the weakest link: Cross capabilities of large language models. *arXiv preprint arXiv:2409.19951*, 2024.

Zhisheng Zhong, Jiequan Cui, Yibo Yang, Xiaoyang Wu, Xiaojuan Qi, Xiangyu Zhang, and Jiaya Jia. Understanding imbalanced semantic segmentation through neural collapse. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19550–19560, 2023.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. LIMA: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024a.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WZH7099tgfM.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. iBOT: Image BERT pre-training with online tokenizer. In *ICLR*, 2021.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024b.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023a.

Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, and Hanjun Dai. Large language models can learn rules. *arXiv preprint arXiv:2310.07064*, 2023b.

Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. *Advances in Neural Information Processing Systems*, 34:29820–29834, 2021.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Part III

Appendices

Appendix A

Investigating the Forgetting in Multimodal Model Fine-tuning

A.1 Neural Collapse

Notations. We use bold capital letters (e.g., $\boldsymbol{M}$) to denote matrices and bold smaller case letters (e.g., $\boldsymbol{v}$) to denote vectors. And for a vector $\boldsymbol{v} \in \mathbb{R}^d$, we use $v_i, \forall i \in [d]$ to denote its ith entry. We consider a K class classification setting, where $\forall k \in [K]$, we use $\boldsymbol{y}_k$ to denote the kth one-hot vector: $\boldsymbol{y}_k = [\underbrace{0, \ldots, 0}_{k-1 \text{ 0s.}}, 1, 0, \ldots, 0]^\top$. We then use $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]^\top \in \mathbb{R}^{K \times d}$ to denote the weight matrix of the *last fully connected layer of a neural network* and $\boldsymbol{w}_k$ is the kth row vector of $\boldsymbol{W}$, $\forall k \in [K]$. Next, we use $\boldsymbol{H} = [\boldsymbol{h}_{k,i} : 1 \leq k \leq K, i \leq n_k] \in \mathbb{R}^{d \times N}$ to denote the matrix of all feature vectors corresponding to all inputs, where n_k represents the number of samples of the hth class and N is the total number of samples. Our analysis will focus on the cross-entropy loss w.r.t. the one hot label vectors $\boldsymbol{y}_k, \forall k \in K, \forall \boldsymbol{z} \in \mathbb{R}^d$: $\mathcal{L}(\boldsymbol{z}, \boldsymbol{y}_k) = -\log \left[\frac{\exp(z_k)}{\sum_{k'=1}^{K} \exp(z'_k)} \right]$.

Preliminary Results of Neural Collapse and Minority Collapse

Neural Collapse. The *neural collapse* (NC) phenomenon is first observed in Papyan et al. [2020] when minimizing the cross-entropy loss for classification. NC characterizes the optimal geometric structures of model classifiers and features in balanced multi-class classification, leveraging such structures during training has been proven effective in mitigating the challenges of imbalanced data training [Yang et al., 2022, Xie et al., 2023], as well as incremental learning settings [Yang et al., 2023b]. Theoretical insights have also emerged from various studies [Fang et al., 2021, Thrampoulidis et al., 2022, Behnia et al., 2022, Zhong et al., 2023] to explore the understanding of data imbalance training from the NC perspective. Fang et al. [2021] introduced Minority Collapse.

Minority Collapse. Fang et al. [2021] extends the theoretical results of NC into the imbalance training setting, revealing that when the *imbalance ratio* of the dataset improves, the feature vectors of *the minority class will converge to one single vector*, which is known as *minority collapse*. Mathematically, Fang et al. [2021] assumes the majority classes and minority classes have n_a, n_b samples per class, respectively. Under such imbalance data model, Fang et al. [2021] shows that when the imbalance ratio $n_a/n_b \to \infty$, any pair of the weight vectors $\boldsymbol{w}_k^\star, \boldsymbol{w}_{k'}^\star$ from the minority classes will satisfy $\lim_{n_a/n_b\to\infty} \boldsymbol{w}_k^\star - \boldsymbol{w}_{k'}^\star = \mathbf{0}$ at convergence during cross-entropy training. We leave the formulation and main results of Fang et al. [2021] in Appendix A.1 for completeness.

Data Imbalance and the SELI geometry. While Fang et al. [2021] aims to characterize the geometric attributes of feature representation and classifier weights in an asymptotic manner, particularly as the imbalance ratio tends towards infinity, the work by Thrampoulidis et al. [2022] offers a more comprehensive exploration of the geometry within the final layer. The latter paper introduces a novel geometric construct known as the Simplex Encoded-Labels Interpolation (SELI) geometry. This geometry characterizes the optimal logit arrangement within the constraints of the unconstrained feature model assumption [Mixon et al., 2020], irrespective of data imbalance. Moreover, the study demonstrates that this framework transitions to the simplex Equiangular Tight Frame (ETF) structure in scenarios of balanced data and aligns with the phenomenon of minority collapse under conditions of asymptotic imbalance.

Theoretical Formulation of NC. NC illustrates that the last-layer features and classifiers assuming *each class has the same training samples*, namely $n_k = n_{k'}, \forall k, k' \in [K]$. As also discussed in later works [Fang et al., 2021, Zhu et al., 2021], one particular NC phenomenon reveals that the last-layer classifiers (weight matrix $\boldsymbol{W}$) will be maximally contrastive [Chen et al., 2020b]. Mathematically, this implies that the cosine similarity $\frac{\langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle}{\|\boldsymbol{w}_i\|\|\boldsymbol{w}_j\|}$ between any pair of the row vectors $\boldsymbol{w}_i, \boldsymbol{w}_j, \forall i, j \in [K], i \neq j$ of $\boldsymbol{W}$ *reaches the largest value* for K equiangular vectors from solving the regularized cross-entropy minimization problem: reformulates the ℓ^2-regularized cross-entropy minimization problem:

$$\min_{\boldsymbol{W}_{\text{full}}} \frac{1}{N} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \mathcal{L}\left(\boldsymbol{f}\left(\boldsymbol{x}_{k,i}; \boldsymbol{W}_{\text{full}}\right), \boldsymbol{y}_k\right), \tag{A.1}$$

where $\{\boldsymbol{x}_{k,i}\}_{i=1}^{n_k}$ denotes the training examples in the kth class, and

$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{W}_{\text{full}}) = \boldsymbol{b}_L + \left(\boldsymbol{W}_L \sigma\left(\boldsymbol{W}_{L-1} \sigma\left(\cdots \sigma\left(\boldsymbol{b}_1 + \boldsymbol{W}_1 \boldsymbol{x}\right) \cdots\right)\right)\right)$$

is the output of an L layer fully connected neural network, and $\boldsymbol{W}_{\text{full}} = \{\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_L\}$ denotes the weights of all L layers.

A Minority Collapse Perspective of Fine-Tuning

We consider the following setting for the pre-training and fine-tuning using the cross-entropy loss. In the pre-training phase, we only assume access to a certain classes of labels $\mathcal{U}_{\text{pt}}, \mathcal{U}_{\text{ft}} \subsetneq [K]$ during the pre-training and fine-tuning phase respectively. Since we assume both $\mathcal{U}_{\text{pt}}$ and $\mathcal{U}_{\text{ft}}$ are strict subset of $[K]$, hence in both the pre-training and fine-tuning phases, the classification problems naturally become imbalanced problems, with the imbalance ratio of ∞. This is because the missing classes $\bar{\mathcal{U}}_{\text{pt}} := [K] \backslash \mathcal{U}_{\text{pt}}, \bar{\mathcal{U}}_{\text{ft}} := [K] \backslash \mathcal{U}_{\text{ft}}$ during the pre-training and fine-tuning phases have 0 samples. Applying the main result from minority collapse [Fang et al., 2021], we know that the classification accuracy of the classes that *appear in the pre-training phase but are absent during fine-tuning* ($\forall k \in \mathcal{U}_{\text{pt}} \cap \bar{\mathcal{U}}_{\text{ft}}$), will degrade after fine-tuning since the weight vectors for these classes will collapse to the same vector ($\boldsymbol{w}_k^\star - \boldsymbol{w}_{k'}^\star = \boldsymbol{0}$). In the next section, we will demonstrate such degradation in pre-training and fine-tuning in image classification, fine-tuning pre-trained contrastive language-image networks, and multimodal visual large language models.

Minority Collapse

Instead of directly analyzing the cross-entropy minimization objective Eq. (A.1), the theoretical literature studies the ℓ^2-norm regularized version since weight norm regularization methods (such as weight decay) are commonly adopted in practical deep learning training [He et al., 2016]:

$$\min_{\boldsymbol{W}_{\text{full}}} \frac{1}{N} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \mathcal{L}\left(\boldsymbol{f}\left(\boldsymbol{x}_{k,i}; \boldsymbol{W}_{\text{full}}\right), \boldsymbol{y}_k\right) + \frac{\lambda}{2} \left\|\boldsymbol{W}_{\text{full}}\right\|^2. \tag{A.2}$$

Fang et al. [2021] reformulates the ℓ^2-regularized cross-entropy minimization problem Eq. A.2 into the Layer-Peeled Model by only consider the weight matrix $\boldsymbol{W}$ and feature vectors $\boldsymbol{h}_{k,i}$ of the last layer with certain norm constraints:

$$\min_{\boldsymbol{W},\boldsymbol{H}} \frac{1}{N} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \mathcal{L}\left(\boldsymbol{W}\boldsymbol{h}_{k,i}, \boldsymbol{y}_k\right), \quad \text{subject to } \frac{1}{K} \sum_{k=1}^{K} \|\boldsymbol{w}_k\|_2^2 \leq E_W, \ \frac{1}{K} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i=1}^{n_k} \|\boldsymbol{h}_{k,i}\|_2^2 \leq E_H. \tag{A.3}$$

Under the Layer-Peeled Model Eq. (A.3), Fang et al. [2021] further assumes that, among the entire K classes, the first K_A classes are the majority classes, with n_a samples per class. While the remaining $[K] \backslash [K_A]$ classes are the minority classes, with n_b samples per class. We state the original results on the minority collapse as follows:

Theorem A.1.1 (Thm.5 of Fang et al. [2021])**.** *Assume $p \geq K$ and $n_A/n_B \to \infty$, and fix K_A and K_B. Let $(\boldsymbol{H}^\star, \boldsymbol{W}^\star)$ be any global minimizer of the Layer-Peeled Model Eq. (A.3) with the cross-entropy loss. When $n_a/n_b \to \infty$, we have*

$$\lim \boldsymbol{w}_k^\star - \boldsymbol{w}_{k'}^\star = \boldsymbol{0}_p, \quad \forall K_A < k < k' \leq K. \tag{A.4}$$

A Neural Collapse Perspective of Next Token Prediction

Note that the MLLM training [Liu et al., 2023d, Li et al., 2023a, Dai et al., 2023] also adopts the cross-entropy loss and the outputs are performing sequential token generations, one could still treat the per-token prediction process as a sequential classification and apply the NC model to understand its behavior. Specifically, when treating the next token generation task as a prediction problem, one can view the preceding text as input for classification, and the next token as the prediction output chosen from the entire vocabulary. Similar to the previous subsections, as long as the pre-training dataset and fine-tuning dataset have different supports, one should expect to see catastrophic forgetting in multimodal model fine-tuning, similar to the aforementioned cases in image classifications.

A.2 Additional Results of Fine-Tuning on Image Classification

Experimental Details of Training ResNet

For each dataset in {MNIST, CIFAR10, CIFAR100, and *mini*Imagenet}, we initiate preprocessing to normalize each dataset by its mean and variance channel-wise and do data augmentation of RandomCrop and RandomHorizontalFlip. Then we train ResNet18 [He et al., 2016] for 200 epochs where we pre-train the initial 50% of classes of all datasets for 100 epochs and subsequently fine-tune the remaining 50% of classes for 100 epochs. Throughout the experiments, we use SGD optimizers with a learning rate of 0.1, momentum of 0.9, and weight decay of 5e-4. We use a learning rate step decay scheduler where we decay the learning rate by the factor of 10 every 80 epochs and we choose the batch size to be 128 for all datasets. We note that for the *mini*Imagenet dataset since we are not doing few-shot learning, we split the total 60k images into the training set (50k images) and validation set (10k images) such that both the training and validation set include the full 100 classes. During pre-training, we set the weight of the first 50% of pre-training classes to be 1, and the remaining classes to 0. Whereas during fine-tuning we set the weight of the last 50% of fine-tuning classes to 1, and the remaining classes to 0.

Additional Results of Training ResNet

To study the catastrophic problem more comprehensively, besides the results we demonstrated in Section 2.3, we conducted additional experiments aimed at exploring the strategy to use a new classifier (instead of changing the criterion weights) during fine-tuning. The results of these experiments are reported in Figure A.1. Notably, our findings indicate that this strategy does indeed have an impact, mitigating catastrophic forgetting to a certain extent. Moreover, we observed a correlation between the degree of catastrophic forgetting and task complexity. Specifically, in the case of MNIST, forgetting occurs but is less severe, whereas, for CIFAR100 and miniImageNet, the degree of forgetting remains similar to the original results.

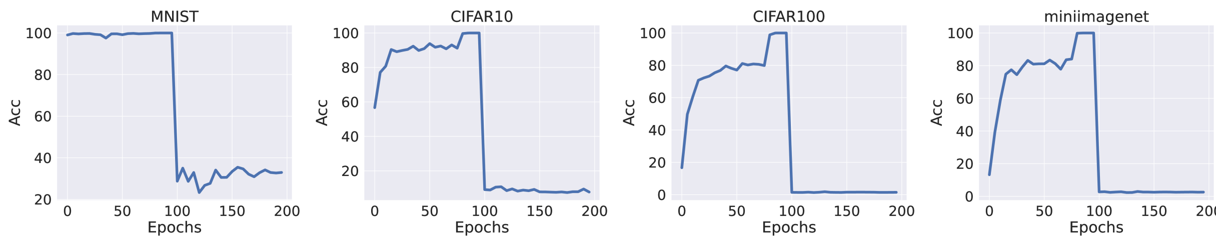


Figure A.1: **Re-initialize classifier during fine-tuning helps to mitigate catastrophic forgetting slightly.** We follow the experimental setup as in Figure 2.2 but adopt a new classifier during fine-tuning.

Moreover, we also conducted additional experiments involving resetting the learning rate during fine-tuning as an ablation study. To be more specific, we restarted both the optimizer and the learning rate scheduler at the moment of the transition to the fine-tuning phase, and we present the results in Figure A.2. As demonstrated, the training accuracy for the pre-trained classes exhibits a curve nearly identical to that depicted in the manuscript, that catastrophic forgetting still happens.

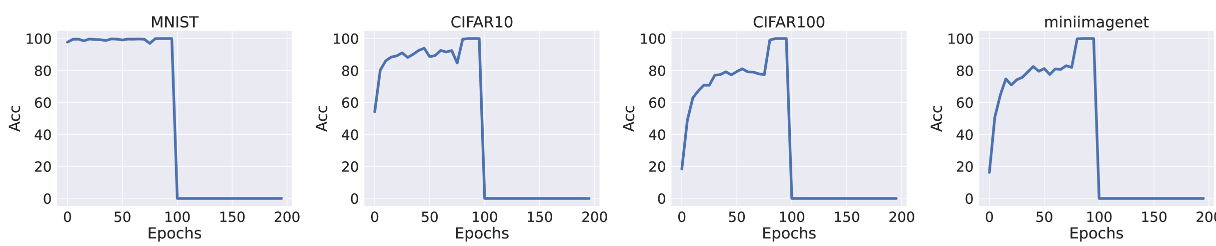


Figure A.2: **Re-initialize optimizer during fine-tuning has no effect on catastrophic forgetting.** We follow the experimental setup as in Figure 2.2 but restart the optimizer and learning rate scheduler during fine-tuning

Experimental Details of Fine-Tuning CLIP

Prompts Used in Fine-Tuning and Zero-Shot Classification. Our prompts for fine-tuning and zero-shot classification differ across datasets for better performance. For each dataset in {MNIST, CIFAR10, CIFAR100, *mini*Imagenet}, we use the following prompts:

- MNIST:

Prompt:

```
a photo of the number: "{digit}".
```

- CIFAR-10:

Prompt:

```
a photo of a {object}.
```

- CIFAR-100:

Prompt:

```
a photo of a {object}.
```

- *mini*Imagenet:

Prompt:

```
a photo of a {object}.
```

Training Details of Fine-Tuning CLIP. When fine-tuning a given dataset, we use all the datasets available to fine-tune the CLIP model. We use a batch size of 128, Adam Optimizer [Kingma and Ba, 2014]. We set the learning rate to 1e-5 and weight decay to 0.001.

	Datasets			
Checkpoints	MNIST	CIFAR-10	CIFAR-100	*mini*Imagenet
`openai`	77.41%	95.36%	76.05%	78.49%
`mnist-ft-5ep`	99.84%	11.80%	2.93%	1.00%
`cifar10-ft-5ep`	17.40%	99.65%	46.21%	27.13%
`cifar100-ft-5ep`	70.92%	91.55%	98.33%	62.58%
`miniimagenet-ft-5ep`	72.90%	89.24%	65.70%	98.59%
`mnist-ft-15ep`	99.76%	9.18%	0.99%	1.00%
`cifar10-ft-15ep`	9.87%	99.41%	11.04%	5.67%
`cifar100-ft-15ep`	39.64%	85.85%	99.27%	46.64%
`miniimagenet-ft-15ep`	30.93%	87.18%	56.39%	98.95%

Table A.1: Zero-short performance of fine-tuned CLIP Radford et al. [2021] on the vision model `ViT-L-14` `openai` Ilharco et al. [2021].

A.3 Evaluating Existing Models with EMT

EMT Prompts

Classification Prompts. Our prompts for classification differs across datasets. For each dataset in {MNIST, CIFAR10, CIFAR100, *mini*Imagenet}, we use the following prompts:

- MNIST:

  EMT Prompt:

  ```
  What is the number in the image?  Please only answer a single
  number in 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
  ```

- CIFAR-10:

  EMT Prompt:

  ```
  What is the object in the image?  Please only answer a single
  object in airplane, automobile, bird, cat, deer, dog, frog,
  horse, ship, truck.
  ```

- CIFAR-100:

  EMT Prompt:

  ```
  What is the object in the image?  Please only answer a single
  object in apple, [other CIFAR-100 labels in text], worm.
  ```

- *mini*Imagenet:

  EMT Prompt:

  ```
  What is the object in the image?  Please only answer a single
  object in African hunting dog, [other miniImagenet labels in
  text], yawl.
  ```

Evaluation Prompts. For evaluating the prediction accuracy of the testing MLLM, we ask `gpt-3.5-turbo` with the following prompt:

EMT Prompt:

```
Please only answer the question in yes or no.  Is the "Prediction"
correctly predicting the right "Label"?  Label:  {label};
Prediction:  {outputs}.
```

For the parameters in `gpt-3.5-turbo`, we set `temperature`=0.2, `max_tokens`=64, `top_p`=1, `frequency_penalty`=0, `presence_penalty`=0.

Fine-Tuning Dataset of Tested MLLMs

We list all datasets used by all tested MLLMs here:

- `LLaVA` is pre-trained on CC3M [Sharma et al., 2018], a dataset with text-image pairs, and fine-tuned on GPT collected language image instruction-following data and ScienceQA [Lu et al., 2022], see more detailed in Section 4.2 of Liu et al. [2023d].
- `Otter` [Li et al., 2023a] proposes Multi-Modal In-Context Instruction Tuning (MIMIC-IT) dataset, a dataset that augments OpenFlamingo into an instruction-following format [Awadalla et al., 2023]. Then `Otter` uses the proposed MIMIC-IT dataset for fine-tuning. See more details in Section 3 of Li et al. [2023a].
- `InstructBLIP` [Dai et al., 2023] transforms different public vision-language datasets into instruction tuning format, and used the transformed instruction tuning datasets for training. See details in Section 2.1 of Dai et al. [2023].
- `LENS` [Berrios et al., 2023] does not explicitly use other vision-language datasets for fine-tuning, as it only proposes a framework to enhance the visual reasoning capability of a frozen LLM, by utilizing different visual modules. See details in Section 3.2 of Berrios et al. [2023].

Classification Accuracy

We present the classification accuracy of different vision models and different MLLMs in Table A.2.

Hallucinated Outputs

We provide some outputs of different models by prompting with EMT in this subsection.

Classifying MNIST with Otter

EMT Prompt:

```
What is the number in the image? Please only answer a single number
in 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
```

Label: 2 | `Otter`:

```
The object is a letter.
```

		Datasets			
Vision Model	Checkpoints	MNIST	CIFAR-10	CIFAR-100	*mini*Imagenet
ViT-L-14	CLIP	77.36%	95.39%	76.04%	78.92%
	LLaVA-7b	56.96%	56.71%	34.53%	14.12%
	LLaVA-13b	53.84%	67.67%	44.79%	30.11%
	Otter	49.59%	78.11%	57.63%	14.94%
ViT-H-14	openclip	73.75%	88.17%	85.09%	85.84%
	LENS	22.84%	12.97%	0.03%	7.08%
ViT-g-14	openclip	63.67%	89.72%	78.86%	83.84%
	InsBLIP-7b	45.12%	94.95%	17.00%	12.76%
	InsBLIP-13b	37.49%	88.72%	17.70%	14.82%

Table A.2: Prediction accuracy of `CLIP` [Radford et al., 2021], `Otter` [Li et al., 2023a], `LLaVA` [Liu et al., 2023d], `openclip s34b_b88k` [Ilharco et al., 2021, Schuhmann et al., 2022], `LENS` [Berrios et al., 2023], `InstructBLIP` [Dai et al., 2023], and `openclip s32b_b79k` [Ilharco et al., 2021, Schuhmann et al., 2022]. For vision encoders, `LLaVA` and `Otter` adopt `ViT-L-14` as their vision encoder; `LENS` adopts `ViT-H-14`; and `InstBLIP` uses `ViT-g-14`. `LLaVA`, `Otter`, `LENS`, and `InstBLIP` are generated via EMT. Total accuracy difference is calculated by summing up the performance gap between a multimodal LLM and its base vision model, for all 4 tasks.

Label: 5 | `Otter`:

`5.`

Label: 7 | `Otter`:

`7.`

Label: 8 | `Otter`:

`2.`

Label: 4 | `Otter`:

`Number 4.`

Classifying CIFAR10 with LLaVA-7b

EMT Prompt:

```
What is the object in the image? Please only answer a single object
in airplane, automobile, bird, cat, deer, dog, frog, horse, ship,
truck.
```

Label: airplane | `LLaVA-7b`**:**

```
The object in the image is a frog.
```

Label: bird | `LLaVA-7b`**:**

```
The object in the image is a large black bird.
```

Label: deer | `LLaVA-7b`**:**

```
The object in the image is a deer.
```

Label: horse | `LLaVA-7b`**:**

```
The object in the image is a horse.
```

Label: truck | `LLaVA-7b`**:**

```
The object in the image is a large airplane.
```

Classifying CIFAR100 with LENS

EMT Prompt:

```
What is the object in the image? Please only answer a single object
in apple, [other CIFAR-100 labels in text], worm.
```

Label: orchid | `LENS`**:**

```
bird
```

Label: dolphin | `LENS`**:**

```
bird
```

Label: mushroom | `LENS`**:**

```
bird
```

Label: girl | `LENS`:

`bird`

Label: boy | `LENS`:

`bird`

Classifying *mini*Imagenet with InstructBLIP

EMT Prompt:

```
What is the object in the image? Please only answer a single object
in African hunting dog, [other miniImagenet labels in text], yawl.
```

Label: street_sign | `InstructBLIP-7b`:

`a street sign for st. catherine's street`

Label: street_sign | `InstructBLIP-7b`:

`a blue street sign says arkansas on it`

Label: coral_reef | `InstructBLIP-7b`:

`a picture of a coral reef with seaweed in the foreground`

Label: harvestman | `InstructBLIP-7b`:

`a close up of a tree branch with a ladybug on it`

Label: African_hunting_dog | `InstructBLIP-7b`:

```
a dog is standing in front of a forest with a sign that says walker
hound
```

A.4 Experimental Details of Fine-Tuning LLaVA

Constructing Datasets for LLaVA Fine-Tuning and Data Usage

Data Construction. We construct vision and language instruction data in the following format for LLaVA fine-tuning.

```
   "image": "airplane/2604.jpg"
"conversations":
"from": "human", "value": "What is the object in the image? <image>"
"from": "gpt", "value": "The object is a(n) airplane."
```

Note that the above example only illustrate one data point from the class `airplane` of CIFAR-10, we also similarly construct other data for MNIST, CIFAR-100, and *mini*Imagenet in our experiments, respectively.

Data Usage. During fine-tuning, we only fine-tune LLaVA model with training data in all dataset. EMT evaluations are only conducted using 20% random selected training data.

Additional Hallucinated Outputs of Fine-Tuning LLaVA

In this section, we provide additional examples to compare the outputs of LLaVA with linear and Lora fine-tune. In particular, we want to illustrate that linear fine-tuning LLaVA on RGB images dataset will still preserve some of LLaVA's capability in binding visual and language data, while lora fine-tuning LLaVA directly lead to overfitting.

Classifying CIFAR10 with 1-epoch fine-tuned LLaVA-7b linear on miniImagenet

EMT Prompt:

```
What is the object in the image? Please only answer a single object
in airplane, automobile, bird, cat, deer, dog, frog, horse, ship,
truck.
```

Label: airplane | `LLaVA-7b-linear-ft-miniimagenet:`

```
The object is a(n) airplane.
```

Label: cat | `LLaVA-7b-linear-ft-miniimagenet:`

```
The object is a(n) cat.
```

Label: deer | `LLaVA-7b-linear-ft-miniimagenet:`

```
The object is a(n) deer.
```

Label: frog | `LLaVA-7b-linear-ft-miniimagenet:`

```
The object is a(n) frog.
```

Label: truck | `LLaVA-7b-linear-ft-miniimagenet:`

```
The object is a(n) automobile.
```

Classifying CIFAR10 with 1-epoch fine-tuned LLaVA-7b lora on miniImagenet

EMT Prompt:

```
What is the object in the image? Please only answer a single object
in airplane, automobile, bird, cat, deer, dog, frog, horse, ship,
truck.
```

Label: airplane | `LLaVA-7b-lora-ft-miniimagenet`:

```
The object is a(n) aircraft carrier.The object is a(n) aircraft
carrier.
```

Label: cat | `LLaVA-7b-lora-ft-miniimagenet`:

```
The object is a(n) black-footed ferret.
```

Label: deer | `LLaVA-7b-lora-ft-miniimagenet`:

```
The object is a(n) white wolf.
```

Label: frog | `LLaVA-7b-lora-ft-miniimagenet`:

```
The object is a(n) rhinoceros beetle.
```

Label: truck | `LLaVA-7b-lora-ft-miniimagenet`:

```
The object is a(n) garbage truck.
```

More Analysis on Hallucination after Fine-tuning

We further conduct some experiments to analyze the "hallucinations" during fine-tuning. In particular, we found that the visual information is preserved after fine-tuning. More specifically, we observed that the ground truth label in the testing dataset will be mostly predicted to a few labels of the fine-tuning datasets, which are "visually similar" to the ground truth class.

We provide several examples in the following. The testing dataset is *mini*Imagenet, and the testing model is fine-tuned on CIFAR-10, using LoRa, for 3 epochs. We provide the top 3 predicted labels as well as the percentage of the appearance.

From the examples shown above, we can see that for all selected ground truth labels in *mini*Imagenet, the fine-tuned model hallucinates by producing labels in CIFAR-10 that a "visually similar" to each ground truth. E.g., African_hunting_dog → Dog, Deer; Arctic_fox → Dog, Cat; French_bulldog → Dog, Cat.

Note that we only provide the prediction percentages of 10 classes in one setting (testing miniimagenet on a CIFAR10 fine-tuned model), due to the space limitation. But the

Ground Truth Label	Top 3 Predictions
African Hunting Dog	dog: 62.12%, deer: 24.24%, bird: 6.06%
Arctic Fox	dog: 73.53%, cat: 17.65%, deer: 4.41%
French Bulldog	dog: 94.64%, cat: 3.57%, deer: 0.89%
Gordon Setter	dog: 94.81%, bird: 1.48%, horse: 1.48%
Ibizan Hound	dog: 88.97%, horse: 8.82%, deer: 1.47%
Newfoundland	dog: 93.69%, horse: 2.70%, ship: 0.90%
Saluki	dog: 85.19%, horse: 11.11%, bird: 2.78%
Tibetan Mastiff	dog: 98.28%, horse: 1.72%
Walker Hound	dog: 96.67%, horse: 1.67%, deer: 0.83%
Aircraft Carrier	ship: 75.21%, The object is an airplane: 24.79%

Table A.3: Top 3 Predictions for Ground Truth Labels after fine-tuning on CIFAR-10.

hallucinated outputs follow this pattern: fine-tuned LLaVA will generate the labels in the fine-tuned dataset, which are most “visually similar” to the ground truth label being tested.

Additional Study on post-processing

As we have discussed in Section 2.7, we further conducted experiments that use the similarity between the CLIP text features for the outputs and labels. In particular:

1. We applied the OpenAI CLIP text embedding [Radford et al., 2021] to extract the text feature vector e of the output “The object is a(n) lion.” into a text embedding feature.

2. Then we also followed OpenAI CLIP text embedding to tokenize the labels of CIFAR10 into $e(1), e(2), \ldots .e(10)$.

3. We outputed the label $i \in [10]$, whose feature embedings has the smallest ℓ^2 distance with the text embedding feature of e.

We report the results in Table A.4, Table A.5, Table A.6, Table A.7, Table A.8.

With the new clip text feature post-processing method, we observed a similar phenomena as the original submission. In particular: Fine-tuning on one dataset causes catastrophic forgetting on other datasets LoRa fine-tuning causes more severe catastrophic forgetting than linear fine-tuning. Overall, using the CLIP text feature, we observe the sim-

	Datasets			
Checkpoints	MNIST	CIFAR-10	CIFAR-100	*mini*Imagenet
llava-7b-v0	46.13%	68.10%	44.87%	28.35%
llava-13b-v0	40.69%	78.62%	46.40%	34.27%

Table A.4: Zero-shot Classification Performance, post-processing with CLIP features

	Datasets			
Checkpoints	MNIST	CIFAR-10	CIFAR-100	*mini*Imagenet
mnist-linear	98.89%	78.94%	58.21%	50.83%
c10-linear	66.34%	92.16%	33.08%	21.99%
c100-linear	17.20%	64.60%	86.39%	27.52%
miniIN-linear	29.70%	64.88%	38.05%	90.40%
mnist-lora	99.85%	77.29%	58.36%	43.22%
c10-lora	22.18%	95.49%	9.13%	8.58%
c100-lora	10.49%	41.04%	93.17%	17.23%
miniIN-lora	9.55%	44.05%	18.92%	95.77%

Table A.5: 3-Epoch Fine-Tuned Classification Performance of llava-13b-v0, post-processing with CLIP features

ilar catastrophic forgetting phenomenon. Note that some of accurcracies using the new method are higher than the post-processing method using OpenAI API, because the CLIP text feature post processing will still make a correct prediction, even when the output sentence is logically incorrect. For the example provided in Section 2.5: label "airplane", when the output is "The airplane is 8.", ChatGPT will classify result as "No", since the output "The airplane is 8." is not making a classification.

But this result does not indicate we shall in general prefer CLIP embedding to ChatGPT or vice versa, since CLIP embedding only works for selecting the "most similar labels", while sometimes ignoring the correctness of the output. On one hand, the clip embedding is designed for classification will still classify "The airplane is 8." to "airplane". On the other hand clip embeddings is perhaps a reasonable option for classification task and economically friendly (as it does not query the openai API).

Classification Accuracy of Fine-Tuning LLaVA

	Datasets			
Checkpoints	MNIST	CIFAR-10	CIFAR-100	*mini*Imagenet
1ep-linear	97.22%	80.60%	58.87%	49.85%
2ep-linear	98.30%	81.09%	59.93%	49.11%
3ep-linear	98.03%	80.87%	59.04%	49.29%
1ep-lora	98.52%	39.84%	22.46%	16.18%
2ep-lora	99.24%	55.79%	35.75%	24.78%
3ep-lora	99.71%	61.95%	42.42%	29.03%

Table A.6: 1-3 Epoch Fine-Tuned Classification Performance of llava-7b-v0 on Mnist, post-processing with CLIP features

	Datasets			
Checkpoints	MNIST	CIFAR-10	CIFAR-100	*mini*Imagenet
1ep-linear	21.47%	90.72%	45.43%	27.08%
2ep-linear	18.77%	91.69%	39.53%	26.15%
3ep-linear	17.91%	92.25%	34.77%	24.77%
1ep-lora	9.93%	89.90%	3.09%	3.45%
2ep-lora	9.64%	91.90%	3.96%	3.57%
3ep-lora	12.78%	94.59%	6.06%	6.79%

Table A.7: Classification Performance of llava-7b-v0 on CIFAR-10, post-processing with CLIP features

	Datasets			
Checkpoints	MNIST	CIFAR-10	CIFAR-100	*mini*Imagenet
1ep-linear	19.32%	71.10%	71.56%	28.93%
2ep-linear	22.17%	58.40%	76.05%	24.63%
3ep-linear	24.20%	45.30%	80.73%	19.50%
1ep-lora	9.80%	43.17%	84.45%	18.06%
2ep-lora	9.63%	43.92%	88.01%	17.91%
3ep-lora	10.07%	41.06%	92.18%	18.24%

Table A.8: Classification Performance of llava-7b-v0 on CIFAR-100, post-processing with CLIP features

	Datasets			
Checkpoints	MNIST	CIFAR-10	CIFAR-100	*mini*Imagenet
`7b-v0`	56.96%	56.71%	34.53%	14.12%
`7b-linear-ft-mnist`	98.03%	9.26%	9.96%	6.14%
`7b-linear-ft-cifar10`	17.10%	92.23%	36.24%	30.76%
`7b-linear-ft-cifar100`	17.88%	31.54%	83.86%	22.61%
`7b-linear-ft-miniImagenet`	6.36%	38.99%	24.99%	77.69%
`7b-lora-ft-mnist`	99.71%	7.03%	1.55%	1.04%
`7b-lora-ft-cifar10`	2.80%	94.59%	3.06%	17.90%
`7b-lora-ft-cifar100`	0.26%	2.47%	92.15%	14.14 %
`7b-lora-ft-miniImagenet`	0.24%	4.41%	5.14%	95.34%
`13b-v0`	53.84%	67.67%	44.79%	30.11%
`13b-linear-ft-mnist`	98.90%	22.63%	20.83%	10.35%
`13b-linear-ft-cifar10`	64.98%	92.15 %	35.29%	31.55%
`13b-linear-ft-cifar100`	39.17%	43.07%	86.42%	27.03%
`13b-linear-ft-miniImagenet`	47.75%	29.77%	29.89%	91.24%
`13b-lora-ft-mnist`	99.85%	38.09%	30.05%	15.58%
`13b-lora-ft-cifar10`	23.43%	95.50%	5.48%	19.68%
`13b-lora-ft-cifar100`	5.48%	3.24%	93.16%	14.20%
`13b-lora-ft-miniImagenet`	0.24%	4.99%	5.22%	95.76%

Table A.9: EMT evaluation accuracy of 3-epoch linear/lora fine-tuned `LLaVA-7/13b` on MNIST, CIFAR-10, CIFAR-100, and *mini*Imagenet, against `LLaVA-7/13b`.

	Datasets			
Checkpoints	MNIST	CIFAR-10	CIFAR-100	*mini*Imagenet
`7b-v0`	56.96%	56.71%	34.53%	14.12%
`ft-mnist-1ep`	97.21%	9.20%	10.38%	5.88%
`ft-cifar10-1ep`	20.45%	90.54%	47.34%	34.90%
`ft-cifar100-1ep`	26.16%	64.54%	72.94%	33.86%
`ft-miniImagenet-1ep`	0.67%	49.46%	38.07%	68.69%
`ft-mnist-2ep`	98.31%	9.27%	9.92%	5.91%
`ft-cifar10-2ep`	17.96%	91.63%	40.76%	32.36%
`ft-cifar100-2ep`	28.21%	52.26%	76.68%	30.86%
`ft-miniImagenet-2ep`	2.10%	44.95%	31.37%	75.48%

Table A.10: Finetuning `LLaVA-7b-linear` under different epochs.

	Datasets			
Checkpoints	MNIST	CIFAR-10	CIFAR-100	*mini*Imagenet
`7b-v0`	56.96%	56.71%	34.53%	14.12%
`ft-mnist-1ep`	98.52%	11.41%	0.57%	0.28%
`ft-cifar10-1ep`	0.01%	89.92%	0.44%	15.58%
`ft-cifar100-1ep`	0.01%	3.29%	84.46%	13.17%
`ft-miniImagenet-1ep`	0.01%	8.04%	4.78%	88.77%
`ft-mnist-2ep`	99.24%	13.72%	1.18%	0.86%
`ft-cifar10-2ep`	0.01%	91.89%	1.39%	15.96%
`ft-cifar100-2ep`	0.01%	2.55%	88.00%	13.59%
`ft-miniImagenet-2ep`	0.00%	5.77%	5.61%	92.03%

Table A.11: Finetuning `LLaVA-7b-lora` under different epochs.

Appendix B

Understanding the Visual Short Coming in Multimodal Models

B.1 Experiment Details

Hyperparameters. In this work, we adopt the same set of hyperparameters as LLaVA [Liu et al., 2023d] and LLaVA-1.5 [Liu et al., 2023c]. We use Vicuna-13b-v1.3 [Zheng et al., 2023] in LLaVA experiments and Vicuna-13b-v1.5 [Zheng et al., 2023] in LLaVA-1.5 experiments. We show the training hyperparameters for LLaVA and LLaVA-1.5 experiments in Table B.1. All experiments are conducted using a maximum of 8 Nvidia A100 GPUs.

Hyperparameter	LLaVA		LLaVA-1.5	
	Stage 1	Stage 2	Stage 1	Stage 2
batch size	128	128	256	128
lr	1e-3	2e-5	2e-3	2e-5
lr schedule decay	cosine	cosine	cosine	cosine
lr warmup ratio	0.03	0.03	0.03	0.03
weight decay	0	0	0	0
epoch	1	3	1	1
optimizer	AdamW [Loshchilov, 2017]			
DeepSpeed stage	2	3	2	3

Table B.1: Hyperparameters for MoF training on LLaVA and LLaVA-1.5.

Pretrain Datasets. We use the same dataset for both LLaVA and LLaVA-1.5 experiments. For LLaVA experiments, stage 1 uses CC595k [Sharma et al., 2018] and stage 2 uses LLaVA 158k [Liu et al., 2023d] instruction data; For LLaVA-1.5 experiments, stage 1 uses CC595k [Sharma et al., 2018] and stage 2 uses DataMix 665k [Liu et al., 2023d, sha, 2023, Goyal et al., 2017, Hudson and Manning, 2019, Marino et al., 2019, Mishra et al., 2019, Schwenk et al., 2022, Sidorov et al., 2020, Mao et al., 2016, Kazemzadeh et al., 2014, Krishna et al., 2017] proposed in Liu et al. [2023c].

B.2 MMVP Benchmark

We provide more details on the MMVP benchmark.

Details of evaluating SOTA models

We access GPT-4V through ChatGPT in October and November 2023. We also evaluate Gemini-Pro through Vertex AI API in December 2023. We use the official checkpoints for InstructBLIP [Dai et al., 2023]. We access mini-GPT4 [Zhu et al., 2023a],[1] LLaVA and LLaVA-1.5 [Liu et al., 2023d] through their playgrounds. We test Bard [Google, 2023a] using the official website in September and October 2023. Moreover, we test new-Bing Microsoft [2023] through new-Bing chat creative mode and GPT-4V [OpenAI, 2023a] in September 2023.

Questions in MMVP Benchmark

We present more examples in MMVP at the end in Figures B.3, B.4, B.5.

Ablation Studies

To further verify that MLLMs make mistakes in MMVP due to their incapable visual grounding instead of hallucination in the language model [Hu and Levy, 2023]. We conduct additional ablation experiments on the format and notations of VQA questions and options in MMVP. We choose GPT-4V to do these experiments, as it is currently the best model.

Swapping options The first experiment swaps the two options in the MMVP benchmark. For example, we change the question from "Are the butterfly's wings closer to being open or closed? (a) Open (b) Closed" to "Are the butterfly's wings closer to being open or closed? (a) Closed (b) Open".

Empirically, we find that GPT-4V obtains a 40.3% accuracy on the option swapping in our study, as opposed to the original 38.7%. We observe that a few questions are answered differently, while the majority remain the same. This further suggests that the visual incapabilities are in the vision encoder rather than in alignment or the LLMs.

[1] To circumvent response hallucination in mini-GPT4 we prefix our questions with "Please only choose an option to answer the question below without explanation: "

Changing notations in the options We conducted an ablation study to assess the impact of altering notations. For example, we changed "(a) Closed (b) Open" to "(1) Closed (2) Open". The results are comparable to the original findings, achieving a performance of 37.3%, closely matching the original 38.7%. The study further suggests that the core challenge in MLLMs is their inherent visual incapability, rather than hallucinations in the language model.

Human Study Details

In this study, we ask four participants to volunteer in our study. An example user interface for labeling is shown in Figure B.1. We collect their responses and calculate the average score as the human-level performance.

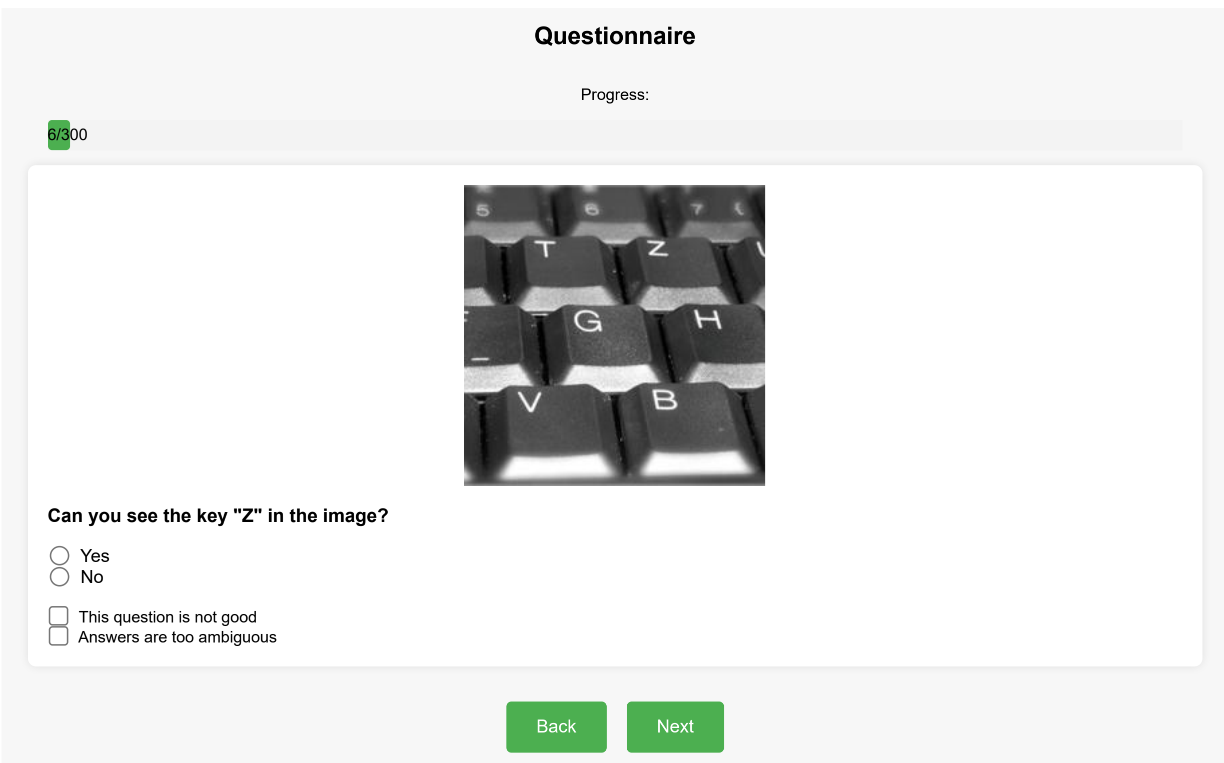


Figure B.1: **Example of user study interface.** The questions in the user study are randomly shuffled to avoid any potential bias. Users choose answers for the VQA questions as well as potential concerns for the VQA question.

	LLaVA-1.5	InstructBLIP	Bard	Gemini	GPT-4
Correlation	0.87	0.71	0.79	0.72	0.31

Table B.2: Pearson Correlation between the CLIP model and MLLMs. Open-source models that explicitly use CLIP-based models are highlighted in gray.

B.3 CLIP-MLLM Failure Correlation

Correlation between CLIP and MLLM models. We compute the Pearson Correlation between the CLIP model and MLLMs and show results in Table B.2. Notably, both open-source models – LLaVA and InstructBLIP – exhibit remarkably high Pearson Correlation, exceeding 0.7. This finding indicates a strong correlation between the errors made by the CLIP model and those made by MLLMs. Bard also displays a very high correlation. This suggests that some of the most advanced closed-source models are also affected by the visual limitations in the CLIP models.

Correlation between ImageNet-1k and MMVP performance. We plot the ImageNet-1k Zero-shot accuracy against MMVP-VLM average performance in Figure B.2. For models with ImageNet-1k Zero-shot accuracy below 80, a higher Zero-shot accuracy tends to indicate improved MMVP performance. However, in models with superior ImageNet-1k Zero-shot performance, this trend does not necessarily hold for MMVP-VLM accuracy. This distinction accentuates the value of MMVP-VLM as an evaluation metric, which probes into visual patterns such as orientation – aspects that are pivotal for downstream tasks and go beyond what is captured by ImageNet accuracy alone.

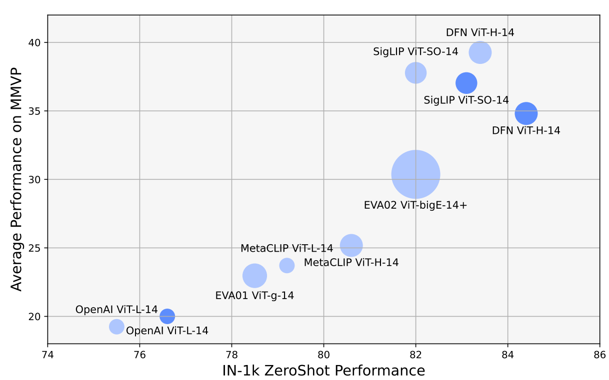


Figure B.2: **Correlation between ImageNet-1k Zero-shot and MMVP-VLM average.** The area of each bubble corresponds to the model's number of parameters. A higher ImageNet-1k zero-shot performance does not necessarily imply superior performance in MMVP-VLM.

B.4 Visual Patterns for CLIP

Here, we provide the full description of visual patterns that pose challenges to all CLIP-based models.

- **Orientation and Direction**: Questions about the direction something is facing or moving, such as the direction the dog or duck is facing, or the orientation of the school bus.
- **Presence of Specific Features**: Questions that focus on the existence or non-existence of certain elements or features in the image.
- **State and Condition**: Questions that pertain to the state or condition of an object, such as whether a flag is blowing in the wind or if the ground is wet.
- **Quantity and Count**: Questions about the number of objects or features present in the image.
- **Positional and Relational Context**: This aspect refers to the model's ability to understand the position and relationship of objects or elements within an image in relation to each other and their surroundings.
- **Color and Appearance**: Questions regarding the color of certain objects or elements.
- **Structural and Physical Characteristics**: This category involves the model's ability to identify and analyze the physical attributes and structural features of objects in an image.
- **A Text**: Questions related to text or symbols present in the image.
- **Viewpoint and Perspective**: Questions concerning the perspective from which the photo was taken.

B.5 More Benchmark Results

Different vision-only backbones

Here, we conduct extra experiments to study MoF involving MAE [He et al., 2022] or MoCoV3 [He et al., 2020] instead of DINOv2; See Table B.3. In Table B.3, we observe that with MAE/MoCov3, there is a consistent improvement in visual grounding ability, as shown in the MMVP and POPE benchmarks.

method	SSL Model	res	#tokens	MMVP	POPE
LLaVA$^{1.5}$	None	336^2	576	24.7	85.9
LLaVA$^{1.5}$ + I-MoF	MoCov3	224^2	512	26.7+2.0	86.1
LLaVA$^{1.5}$ + I-MoF	MAE	224^2	512	27.3+2.6	86.1
LLaVA$^{1.5}$ + I-MoF	DINOv2	224^2	512	28.0+3.3	86.3

Table B.3: Results of Interleaved MoF with different vision-only SSL model

method	res	#tokens	MMVP	LLV^B	LLV^W	MMB	VQA^T	POPE	VQA^{V2}	MM-V
LLaVA$^{1.5}$	336^2	576	24.7	**84.7**	70.7	**67.7**	**61.3**	85.9	**80.0**	**35.4**
LLaVA$^{1.5}$ + I-MoF	224^2	512	28.0	82.7	**73.3**	61.6	55.3	86.3	77.3	33.5
LLaVA$^{1.5}$ + I-MoF	336^2	1152	**31.3**	81.8	**73.3**	65.4	58.7	**86.7**	79.3	34.6

Table B.4: **Comparison with LLaVA-1.5 on 6 more benchmarks**. Interleaved-MoF LLaVA-1.5 obtains performance on par with the original method while showing improvements on benchmarks evaluating visual grounding. Benchmark names are abbreviated due to space limits. LLV^B: LLaVA Benchmark [Liu et al., 2023d]; LLV^W: LLaVA-In-the-Wild [Liu et al., 2023c]; MMB: MMBench [Liu et al., 2023e]; VQA^T: TextVQA[Singh et al., 2019]; POPE: POPE [Li et al., 2023d]; VQA^{V2}: VQA-v2 [Goyal et al., 2017]; MM-V: MM-Vet [Yu et al., 2023].

Scaling up to larger resolution

We conduct additional experiments on Interleaved-MoF that further scale up the resolution to 336 and evaluate on more benchmarks. The summarized results in Table B.4 reveal that Interleaved-MoF achieves comparable performance on most benchmarks while demonstrating improvements in benchmarks focused on visual grounding. We also observe that MMVP are more sensitive to the model's visual capabilities, underscoring the significance of our benchmark in assessing visual proficiency.

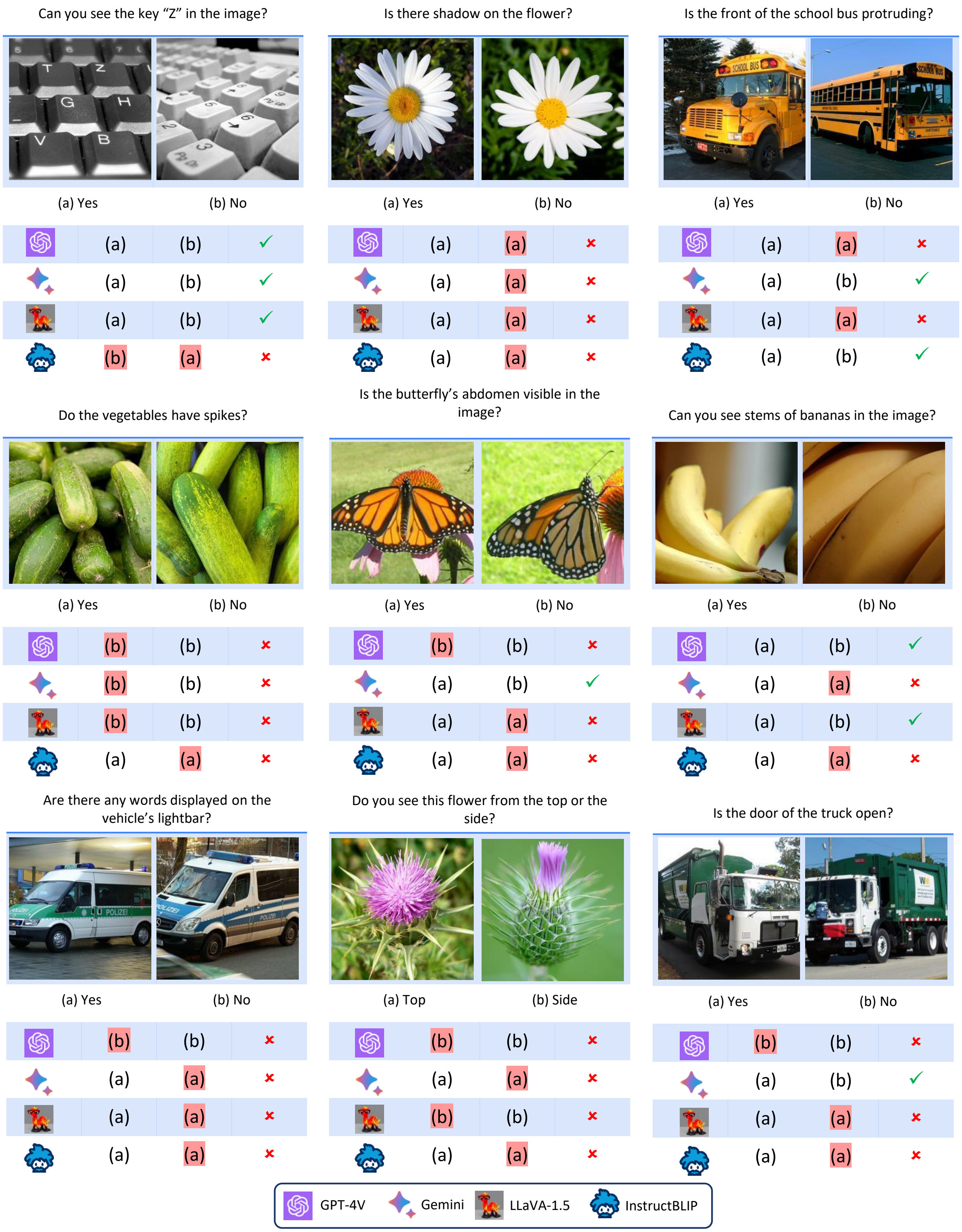


Figure B.3: **More examples of questions in the MMVP benchmark (Part I).**

Does the keyboard have a backlight?

(a) Yes (b) No

GPT-4V	(a)	(a)	✗
Gemini	(a)	(b)	✓
LLaVA-1.5	(a)	(a)	✗
InstructBLIP	(a)	(a)	✗

How many eyes of the cat can you see in the picture?

(a) 1 (b) 2

GPT-4V	(a)	(a)	✗
Gemini	(b)	(b)	✗
LLaVA-1.5	(b)	(b)	✗
InstructBLIP	(b)	(b)	✗

Does this corn have white kernels?

(a) Yes (b) No

GPT-4V	(a)	(b)	✓
Gemini	(a)	(b)	✓
LLaVA-1.5	(a)	(b)	✓
InstructBLIP	(b)	(b)	✗

What does the center button say?

(a) OK/SELECT (b) OK

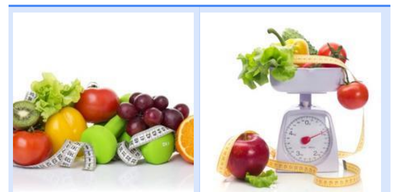
GPT-4V	(a)	(a)	✗
Gemini	(a)	(b)	✓
LLaVA-1.5	(a)	(a)	✗
InstructBLIP	(a)	(a)	✗

Where is the yellow animal's head lying in this image?

(a) Floor (b) Carpet

GPT-4V	(b)	(b)	✗
Gemini	(a)	(b)	✓
LLaVA-1.5	(a)	(a)	✗
InstructBLIP	(b)	(b)	✗

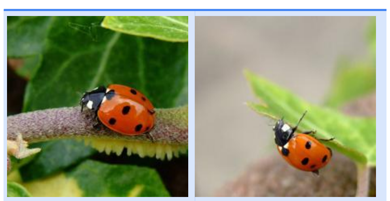
Are some fruits cut open or are all the fruits uncut?



(a) Yes (b) No

GPT-4V	(a)	(a)	✗
Gemini	(a)	(a)	✗
LLaVA-1.5	(a)	(a)	✗
InstructBLIP	(a)	(a)	✗

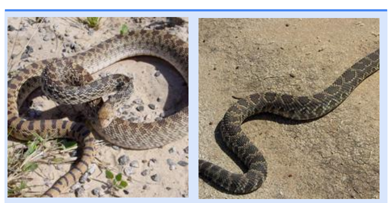
Is the ladybug positioned upright or upside down?



(a) Yes (b) No

GPT-4V	(a)	(b)	✓
Gemini	(a)	(b)	✓
LLaVA-1.5	(b)	(b)	✗
InstructBLIP	(a)	(a)	✗

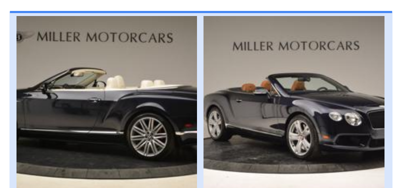
In this picture, is the snake's head visible or not visible?



(a) Visible (b) Not VIsible

GPT-4V	(a)	(b)	✓
Gemini	(a)	(b)	✓
LLaVA-1.5	(b)	(b)	✗
InstructBLIP	(b)	(b)	✗

How many wheels can you see in the image?



(a) 1 (b) 2

GPT-4V	(b)	(b)	✗
Gemini	(b)	(b)	✗
LLaVA-1.5	(b)	(b)	✗
InstructBLIP	(b)	(b)	✗

GPT-4V Gemini LLaVA-1.5 InstructBLIP

Figure B.4: **More examples of questions in the MMVP benchmark (Part II).**

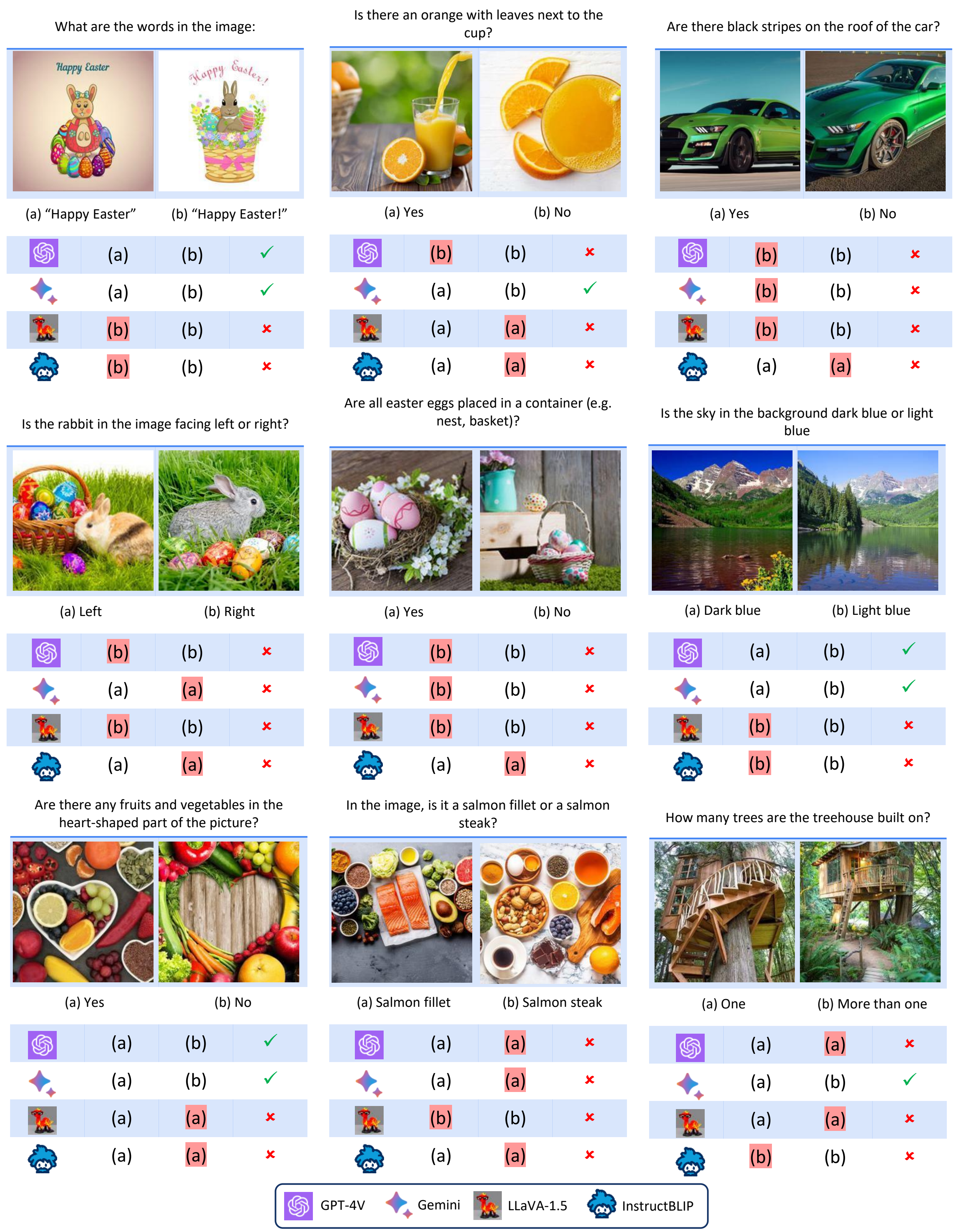


Figure B.5: **More examples of questions in the MMVP benchmark (Part III).**

Appendix C

Adapting RL to Foundation Model Training

C.1 Additional Details of the Evaluation Tasks

Gym Cards

NumberLine

State and action space. In the `NumberLine` task, the visual observation at each state s_t contains two lines of text: "Target: x" and "Current: y_t", where x, y_t are both integers such that $x, y_t \in [0, n_{\text{max}}]$, where n_{max} is an environment input variable that controls the maximum position of the numbers. The goal is to move the current number y_t to the target number x, by sequentially choosing actions from the discrete action space $\{$"$+$", "$-$"$\}$. We set $n_{\text{max}} = 5$ for all experiments in this work, but n_{max} can be set to any positive integers. Choosing "$+$" or "$-$" will increase or decrease the current number y_t by 1, respectively, and the agent will stay at the boundary if it takes an action that attempts to cross the boundary (e.g., taking $a_t =$ "$+$" when $y_t = n_{\text{max}}$ or $a_t =$ "$-$" when $x_t = 0$). See an example of the state action transition in Figure C.1.

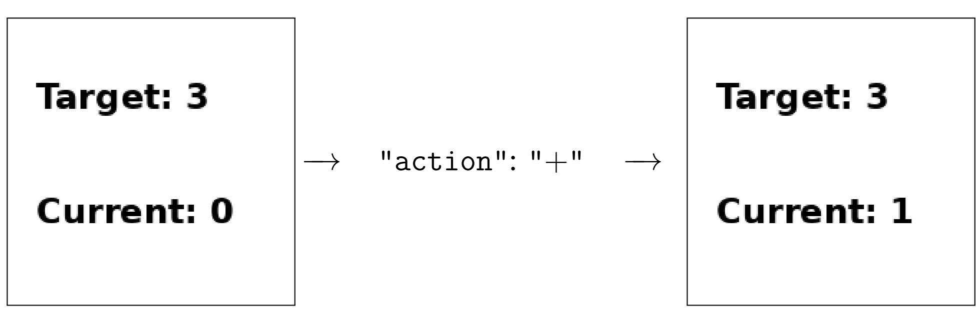


Figure C.1: **An example of the transition in `NumberLine`.**

Reward functions and the CoT prompts. An episode in `NumberLine` ends when the current number equals the target number or the maximum step $T = 2n_{\max}$ is reached. The agent receives a terminal reward of $r(s_t, a_t) = 1$ when $y_{t+1} = x$. The agent also receives a reward penalty of $r(s_t, a_t) = -1$ upon taking an incorrect action that does not result in a closer position to the target $(|x - y_t| \geq |x - y_{t+1}|)$, otherwise the agent receives reward $r(s_t, a_t) = 0$. In the example provided above (Figure C.1), the agent receives a reward $r = 0$, since it moves closer to the target, but not reaching the target yet. For the `NumberLine` task, we adopt the following CoT prompt in Figure C.2, and for the case without CoT reasoning (discussed in Section 4.6), we use the same prompt but without the blue CoT reasoning parts.

CoT prompt $_t$ for task `NumberLine`

You are playing a game called number line. You will see a target number and a current number in the image. And your goal is to move the current number closer to the target by choosing either adding or subtracting one to the current number. Your response should be a valid json file in the following format:
{
"current number": "x",
"target number": "x",
"thoughts": {first read out the current and target number, then think carefully about which action to choose},
"action": "-" or "+"
}

Figure C.2: **Task-specific CoT prompt input $_t$ for `NumberLine`.** The blue part represents the CoT reasoning and the red part is the text-based action.

EZPoints

State and action space. In the `EZPoints` task, the agent will observe an image of two cards and a text version of "formula" below the cards, at each state s_t. The goal is to use the cards in the image to compute a target number of 12 and we view {`"J"`, `"Q"`, `"K"`} as `"10"`. The action space of `EZPoints` is {`"1"`, `"2"`, ..., `"10"`, `"+"`, `"*"`, `"="`} and each number in the cards can *only be used once*. Any action attempting to either select a number not shown in the cards or use a card more than once are *illegal*. At s_t, if a *legal* action a_t is taken, the action will be appended to the text "formula" in s_t and becomes the next state s_{t+1}. On the other hand, when an illegal action is taken, s_{t+1} will remain the same as s_t. All images generated from the `EZPoints` environment are guaranteed to have a viable solution for computing 12.

Reward functions and the CoT prompts. An episode terminates when `"="` is taken or the maximum step $T = 5$ is reached. The agent receives a reward of $r = -1$ upon taking

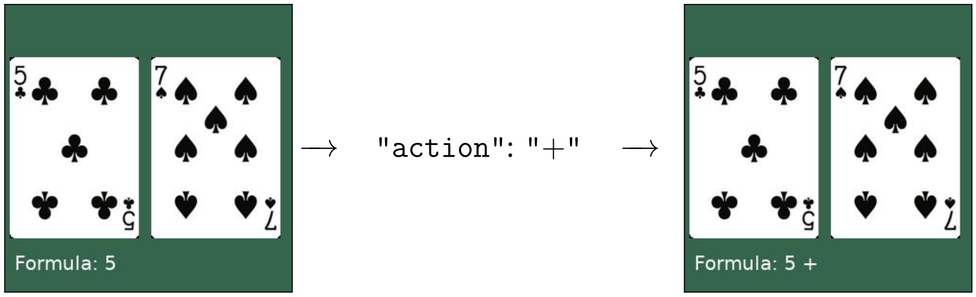


Figure C.3: **An example of the transition in `EZPoints`.**

an *illegal* action, and $r = 0$ while taking a legal action. When "=" is taken, the agent will receive a positive reward $r = 10$ if the formula equals 12, and $r = -1$ otherwise. For the `EZPoints` task, we adopt the following CoT prompt in Figure C.4, and for the case without CoT reasoning (discussed in Section 4.6), we use the same prompt but without the blue CoT reasoning parts and the brown part in Figure C.4 is the text version of the current formula directly extracted from the current state s_t.

CoT prompt $_t$ for `EZPoints`
You are an expert card game player. You are observing two cards in the image. You are observing the current formula: '5'. You can choose between ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '+', '*', '=']. The number or operator you choose will be appended to the current formula. Note that 'J', 'Q', and 'K' count as '10'. Your goal is to output a formula that evaluates to 12, and each number can only be used once. Your response should be a valid json file in the following format:
{
"cards": [x, y],
"current formula": '5',
"thoughts": {First check whether the current formula 'z' is complete. If the current formula 'z' is complete, output '='. Otherwise consider which number or operator should be appended to the current formula to make it equal 12.}
"action": "{number}" or "{operator}"
}

Figure C.4: **Task-specific CoT prompt input $_t$ for `EZPoints` given the observation in Figure C.3**. The blue part represents the CoT reasoning, the red part is the text-based action, and the brown part is the state-dependent text from the formula in the image.

Points24

State and action space. Similar to `EZPoints`, the goal of `Points24` is also to generate a formula to compute the target number of 24, using all four cards. `Points24` has a slightly larger action space: {"1", "2", ..., "10", "+", "−", "∗", "/", "(", ")", "="} and two more cards. Each number in the cards can *only be used once*. Similar to `EZPoints`, any action attempting to either select a number not shown in the cards or use a card more than once are *illegal*. At s_t, if a *legal* action a_t is taken, the action will be appended to the text "formula" in s_t and becomes the next state s_{t+1}. When an illegal action is taken, s_{t+1} will remain the same as s_t. Different from `EZPoints` where all images are guaranteed to have a viable solution for computing 12, the images generated by `Points24` do not always have a viable solution to 24.

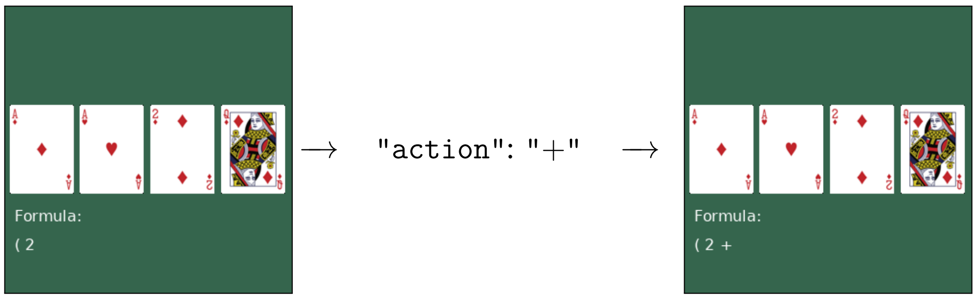


Figure C.5: **An example of the transition in `Points24`.**

Reward functions and the CoT prompts. The reward functions and termination conditions of `Points24` are the same as those in `EZPoints`. An episode terminates when "=" is taken or the maximum step $T = 20$ is reached. The agent receives a reward of $r = -1$ upon taking an *illegal* action, and $r = 0$ while taking legal actions. When "=" is taken, the agent will receive a positive reward $r = 10$ when the formula equals 24, and $r = -1$ otherwise. For the `Points24` task, we adopt the following CoT prompt in Figure C.6, and for the case without CoT reasoning (discussed in Section 4.6), we use the same prompt but without the blue CoT reasoning parts and the brown part in Figure C.6 is the text version of the current formula directly extracted from the current state s_t. We also provide an additional feature that allows us to view {"J", "Q", "K"} as {"11", "12", "13"}, instead of {"10"}.

CoT prompt ${}_t$ for `Points24`
You are an expert 24 points card game player. You are observing these four cards in the image. You are observing the current formula: `'(2'`. You can choose between [`'1'`, `'2'`, `'3'`, `'4'`, `'5'`, `'6'`, `'7'`, `'8'`, `'9'`, `'10'`, `'+'`, `'-'`, `'*'`, `'/'`, `'('`, `')'`, `'='`]. The number or operator you choose will be appended to the current formula. Note that `'J'`, `'Q'`, and `'K'` count as `'10'`. Your goal is to output a formula that evaluates to 24, and each number can only be used once. Your response should be a valid json file in the following format:
{
"cards": [x, y, z, w],
"current formula": `'(2'`
"thoughts": {First check whether the current formula equals 24. If the current formula equals 24, output `'='`. Otherwise consider which number or operator should be appended to the current formula to make it equal 24.}
"action": "{number}" or "{operator}"
}

Figure C.6: **Task-specific CoT prompt input ${}_t$ for `Points24` given the observation in Figure C.5.** The blue part represents the CoT reasoning and the red part is the text-based action, brown part is the state-dependent text that directly obtained from the formula in the image.

Blackjack

State and action space. For the `Blackjack` task, the visual observation at state s_t consists of two cards (one face-down) from the dealer and all cards from the player. The agent's goal in this task is to win the current game, by choosing actions in {`"stand"`, `"hit"`}. The agent will receive a new card upon choosing `"hit"`. See Figure C.7 for an example transition.

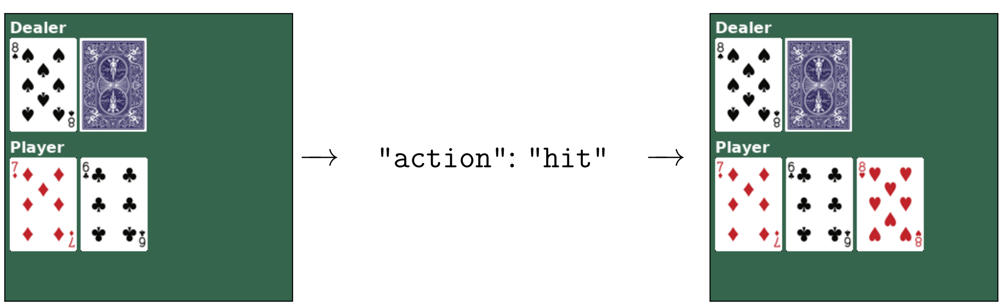


Figure C.7: **An example of the transition in `Blackjack`.**

Reward functions and the CoT prompts. The game terminates when the player chooses `"stand"` or busts (total points exceed 21). We adopt the same reward function as the `Blackjack-v1` task in Gymnasiym [Towers et al., 2023], where $r(s_t, a_t) = 1, 0, -1$ upon win, draw, and loss, respectively. We also provide a similar feature as Gymnasium [Towers et al., 2023], where the "blackjack" winning (the agent win with an `"A"` and a `"10"`, `"J"`, `"Q"` or `"K"`) reward r of the player will become 1.5. In the example provided in Figure C.7, the game has not terminated after taking the action `"hit"`, hence the agent will not receive any rewards, even though it has total points of 21. For the `Blackjack` task, we adopt the following CoT prompt in Figure C.8, and for the case without CoT reasoning (discussed in Section 4.6), we use the same prompt but without the blue CoT reasoning parts.

CoT prompt $_t$ for `Blackjack`
You are a blackjack player. You are observing the current game state, you can choose between [`'stand'`, `'hit'`]. Your response should be a valid json file in the following format:
{
"thoughts": "{first describe your total points and the dealer's total points then think about which action to choose}",
"action": "stand" or "hit"
}

Figure C.8: **Task-specific CoT prompt input $_t$ for `Blackjack`.** The blue part represents the CoT reasoning and the red part is the text-based action.

ALFWorld

State and action space. Inherited from Text World [Côté et al., 2019], at each state s_t of `alfworld`, the agent will observe an RGB image and text-based description. The action space of `alfworld` can be summarized these following format [Shridhar et al., 2021]: (1) `goto {recep}`; (2) `take {obj} from {recep}`; (3) `put {obj} in/on {recep}`; (4) `open {recep}`; (5) `close {recep}`; (6) `toggle {obj}{recep}`; (7) `clean {obj} with {recep}`; (8) `heat {obj} with {recep}`; (9) `cool {obj} with {recep}`, where `{obj}` and `{recep}` stands for objects and receptacles. See Figure C.9 for an example of the state action transition in the `alfworld` environment.

Reward functions and the CoT prompts. Each state $s \in \mathcal{S}$ of `alfworld` has a set of *admissible actions* $\mathcal{A}_{\texttt{adm}}(s)$, a final goal $g_{\texttt{task}}$, and subgoals $g_{\texttt{sub}}$. Since the goal of `alfworld` is to complete the language-based goal-conditioned tasks, we reward the agent upon reaching subgoals and completing the task, while penalizing the agent upon taking inadmissible actions. To summarize, we define the reward function of `alfworld` as $r(s_t, a_t, s_{t+1}|g_{\texttt{task}}) = 50 * \mathbf{1}\left[s_{t+1} = g_{\texttt{task}}\right] + \mathbf{1}\left[s_{t+1} = g_{\texttt{sub}}\right] - \mathbf{1}\left[a_t \notin \mathcal{A}_{\texttt{adm}}(s_t)\right]$. For the `alfworld` task, we adopt the following CoT prompt in Figure C.10, and for the case without CoT reasoning (discussed in Section 4.6), we use the same prompt but without the blue CoT reasoning parts and the brown part in Figure C.10 is the text description of the task directly extracted from the current state s_t.

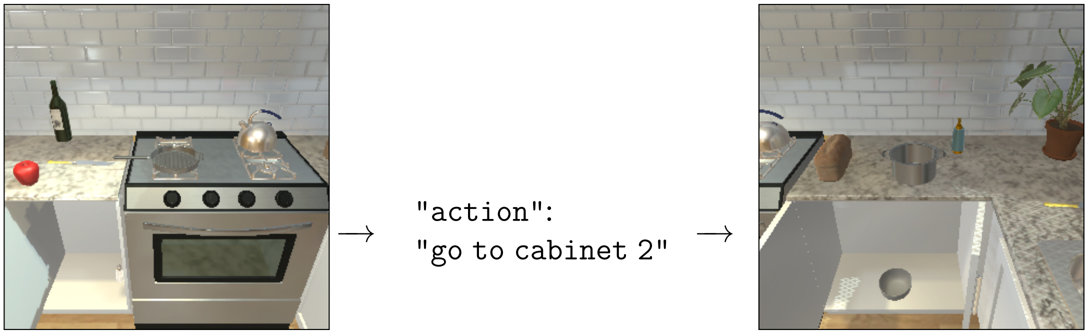


Figure C.9: **An example of the transition in `alfworld`.**

CoT prompt $_t$ for `alfworld`
Your are an expert in the ALFRED Embodied Environment. You are also given the following text description of the current scene: ['You arrive at loc 0. The cabinet 1 is open. On the cabinet 1, you see a pan 1, a kettle 1, a winebottle 1, a apple 1, a stoveknob 1, a stoveknob 2, a stoveknob 3, a stoveknob 4, a knife 1, a saltshaker 1, and a bread 1.']. Your task is to put a cool mug in cabinet. Your admissible actions of the current situation are: ['go to countertop 1', 'go to cabinet 2', 'go to countertop 2', 'go to stoveburner 1', 'go to drawer 1', 'go to drawer 2', 'go to drawer 3', 'go to stoveburner 2', 'go to stoveburner 3', 'go to stoveburner 4', 'go to drawer 4', 'go to cabinet 3', 'go to cabinet 4', 'go to microwave 1', 'go to cabinet 5', 'go to cabinet 6', 'go to cabinet 7', 'go to sink 1', 'go to sinkbasin 1', 'go to fridge 1', 'go to toaster 1', 'go to coffeemachine 1', 'go to cabinet 8', 'go to drawer 5', 'go to drawer 6', 'go to drawer 7', 'go to drawer 8', 'go to shelf 1', 'go to shelf 2', 'go to countertop 3', 'go to shelf 3', 'go to drawer 9', 'go to garbagecan 1', 'open cabinet 1', 'close cabinet 1', 'take pan 1 from cabinet 1', 'take kettle 1 from cabinet 1', 'take winebottle 1 from cabinet 1', 'take apple 1 from cabinet 1', 'take stoveknob 1 from cabinet 1', 'take stoveknob 2 from cabinet 1', 'take stoveknob 3 from cabinet 1', 'take stoveknob 4 from cabinet 1', 'take knife 1 from cabinet 1', 'take saltshaker 1 from cabinet 1', 'take bread 1 from cabinet 1', 'inventory', 'look', 'examine cabinet 1']. Your response should be a valid json file in the following format:
{
"thoughts": "first describe what do you see in the image using the text description, then carefully think about which action to complete the task. ",
"action": "an admissible action"
}

Figure C.10: **Task-specific CoT prompt input $_t$ for `alfworld` given the observation in Figure C.9.** The blue part represents the CoT reasoning and the red part is the text-based action, brown part is the state-dependent text that directly obtained from the text description and the admissible actions of the current state.

C.2 Additional Details on the Experiments

We provide additional detailed of the experimental results in Section 4.6 here. Details of our experimental pipeline is provided in Section C.2, including preparing the initial SFT checkpoints and the RL training. Section C.2 contains details setup of all comparative methods. We list task-specific training details in Section C.2. We provide additional experimental results in Section C.2. Section C.2 lists several failure examples of the `Points24` tasks.

Experimental Pipeline

Our experiments adopt a similar pipeline as RLHF [Ouyang et al., 2022], where we first apply supervised fine-tuning (SFT) to the backbone llava-v1.6-mistral-7b model, before RL training. As outlined by Ouyang et al. [2022], the RLHF training procedure consists of three distinct stages: SFT, learning reward models from human preference data, and applying RL with the learned reward models. Our pipeline is analogous to RLHF but without requiring the collection of human preference data for learning reward models, as we can directly collect rewards from the environment.[1] Consequently, our experimental pipeline only contains two stages: SFT and RL, which we will explain below.

Supervised fine-tuning. For the original `gym_cards` environment, we manually construct instruction-following data for all tasks following the format specified in Figure 4.3 of Section 4.4. As for `alfworld`, we use GPT4-V [OpenAI, 2023c] to collect instruction following data for SFT. For all tasks, we prepare two versions of the instruction-following data, one with CoT and one without. We leave the details of the CoT prompts for each task, and the details of each fine-tuning dataset in Appendix C.3. After constructing the instruction-following data (with and without CoT), we fine-tune llava-v1.6-mistral-7b for 1 epoch on the collected data for each task and report the results for LLaVA-sft.

RL training. For each task, we start our RL training from the LLaVA-sft checkpoint. The LLaVA model [Liu et al., 2023d] consists of three jointly trainable components, a CLIP vision encoder [Radford et al., 2021], an LLM backbone [Touvron et al., 2023a,b, Jiang et al., 2023], and an MLP projector that connects visual features and the word embeddings, and we directly apply PPO [Schulman et al., 2017] to train all three components. Due to computation resource limitations, we instantiate our experiments via LoRA [Hu et al., 2022], with the LoRA configuration of $r = 128, \alpha = 256, \texttt{dropout} = 0.05$, for all trainable components. For the CoT coefficient λ, we set $\lambda = 0.5$ in the `gym_cards` domain and $\lambda = 0.2$ in `alfworld`.

[1] We adopt the same pipeline for the evaluation without CoT reasoning (discussed in Section 4.6) while changing the data for SFT as well as (see more details on our SFT data and in Appendix C.3)

Experimental Setup for Comparative Methods

GPT4-V and Gemini. All of our experimental results on GPT4-V [OpenAI, 2023c] and Gemini [Google, 2023c] are tested on March 15, 2024, using the same prompt for our RL training (see detailed prompts in Appendix C.3). For `gym_cards`, the numbers from both GPT4-V and Gemini are averaged among the same number of episodes: 200 episodes for deterministic tasks (`NumberLine`, `EZPoints` and `Points24`); 1000 episodes for stochastic task (`Blackjack`). As for `alfworld`, we report the performance of GPT4-V on all 1000 episodes we collected, see Appendix C.3 for our data collection on `alfworld` using GPT4-V. Due to the financial budget, we report the results of Gemini using 100 episodes.

LLaVA-sft. For each number of LLaVA-sft, we collect the instruction-following dataset for each task and then fine-tune LLaVA-1.6-7b for 1 epoch on the collected data using the official LLaVA fine-tuning script.[2] Details of our data collection process is provided in Appendix C.3. We also *use the same LLaVA-sft checkpoint as initializations for the downstream RL training*.

CNN-based RL. Since the LLaVA-7b model adopts a CLIP ViT-L/14 vision encoder which is more powerful than vanilla CNN embeddings, we instantiate our CNN-based method using the feature from the same CLIP ViT-L/14 for a fair comparison. For tasks (`EZPoints`, `Points24`, and `alfworld`, see our detailed prompt in Appendix C.3) that require text inputs, we adopt the `RoBERTa-base` [Liu et al., 2019] model to encode the text feature and concatenate the text and CLIP visual features for downstream RL training. After obtaining the CLIP (potentially concatenated with text) features, we adopt 2 MLP layers followed by a fully connected layer to map the clip features into the action space. We adopt the PPO [Schulman et al., 2017] implementation from Kostrikov [2018] as the backbone RL algorithm. In addition, we adopt a `CosineAnnealingLR` learning rate scheduler, with the initial learning rate of $3e-4$, the final learning rate of $1e-8$, and the maximum learning rate step of 25. The remaining task specific hyperparameters are the same as the VLM case in Section C.2.

General Setup for End-to-End RL Training

All experiments are conducted on an 8 A100s DGX machine (80G), while the maximum VRAM requirement is < 40G. Each curve from Figure 4.5 and 4.6 takes at most 36 hours to finish. We adopt DeepSpeed zero2 [Rasley et al., 2020] for multi-gpu training. During our training for the VLM, we directly train all trainable components (vision encoder, LLM, and the MLP projector). We adopt an open-source implementation [Kostrikov, 2018] for the PPO. Inspired by von Werra et al. [2020], Castricato et al. [2023], we apply a 3-layer MLP as

[2] `https://github.com/haotian-liu/LLaVA/blob/main/scripts/v1_5/finetune.sh`. We start from the llava-v1.6-mistral-7b instead of the v1.5 checkpoint in the script.

the value head, on top of the output hidden states layer *before the output tokens*, to estimate the value function V^{π_θ}. After obtaining the value estimate V_ϕ, we adopt the generalized advantage estimator (GAE) [Schulman et al., 2016] to estimate the return function $\hat{R}(s)$ and the advantage function $\hat{A}^{\pi_\theta}$ of π_θ. In addition, we adopt a `CosineAnnealingLR` learning rate scheduler, with the initial learning rate of $1e-5$, the final learning rate of $1e-9$, and the maximum learning rate step of 25. For all experiments in the `gym_cards` and `alfworld` environment, we set the scaling hyperparameter $\lambda = 0.5, 0, 2$, respectively. The learning rate decay happens after every PPO update, which consists of 4 epochs of gradient updates with PPO. The number of data for on-policy training and batch size is task-dependent, we list them below.

Numberline and Blackjack. For `NumberLine` and `Blackjack`, our VLM training curves in Figure 4.5 use 4 GPUs. Our implementation naturally enables different random seeds on different GPUs, hence our VLM curves are averaged among 4 seeds. For one PPO update on each GPU, we collect 512 transitions, with a batch size of 128 per GPU (batch size = 512 in total). The episode return and success rate are averaged with `NumberLine`, `Blackjack` are averaged among 200 and 1000 episodes, respectively. `Blackjack` is calculated with more episodes because of its stochasticity, while `NumberLine` is a deterministic task. We adopt the same number of transitions and batch size for the on-policy training in the CNN-based method on both tasks. The CNN-based methods are averaged among 4 random seeds as well.

EZPoints and Points24. For `EZPoints` and `Points24`, our VLM training curves in Figure 4.5 use 4 GPUs. Our implementation naturally enables different random seeds on different GPUs, hence our VLM curves are averaged among 4 seeds. For one PPO update on each GPU, we collect 1024 transitions, with a batch size of 128 per GPU (batch size = 512 in total). We use 1024 transitions because the episodes of `EZPoints` and `Points24` usually have longer horizons than `NumberLine` and `Blackjack`. The episode return and success rate are averaged with `EZPoints` and `Points24` are averaged among 200. We adopt the same number of transitions and batch size for the on-policy training in the CNN-based method on both tasks. The CNN-based methods are averaged among 4 random seeds as well.

ALFWorld. For the `alfworld` environment, each run of our VLM training curves in Figure 4.5 and Figure C.12 are conducted on one GPU, and each curve is averaged among 4 seeds. We do not conduct multi-GPU training for `alfworld` because the on-policy sampling time has a huge variance on different GPUs, which will largely increase the synchronization time across different GPUs. For each PPO update, we collect 1024 transitions, and with a batch size of 256. The episode success rates are averaged among 200 episodes. We adopt the same number of transitions and batch size for the on-policy training in the CNN-based method on both tasks. The CNN-based methods are averaged among 4 random seeds as well.

Additional Experimental Results

We provide some additional experimental results on the episode returns on the `gym_cards` and the task-specific training curves for `alfworld` here.

	Episode Success Rate (%)				Episode Return			
	`NL`	`EZP`	`P24`	`BJ`	`NL`	`EZP`	`P24`	`BJ`
CNN+RL	87.1	0	0	38.8	0.79	-1.02	**-1.12**	-0.17
GPT4-V	65.5	10.5	0	25.5	-0.59	-1.30	-4.39	-0.44
Gemini	82.5	2.0	0	30.0	0.74	-2.57	-2.68	-0.35
LLaVA-sft	24.8	23.0	**2.6**	23.1	-2.30	-0.50	-13.52	-0.50
Ours	**89.4**	**50.0**	2.3	**40.2**	**0.87**	**4.46**	-11.84	**-0.13**

Table C.1: **Average episode success rates and returns of different methods on `gym_cards`.** For all RL-based methods (CNN and our method), we report the *best* results in each training curve from Figure C.11.

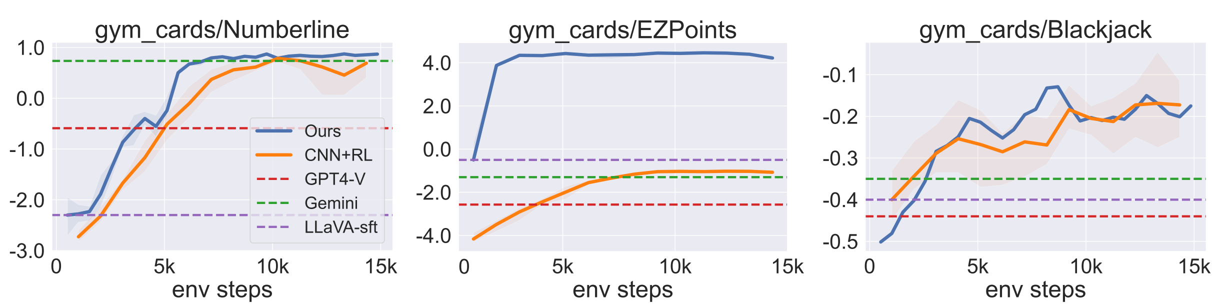


Figure C.11: **Episode returns of different methods on `gym_cards`.** An extended version of Figure 4.5 containing episode success rates and returns.

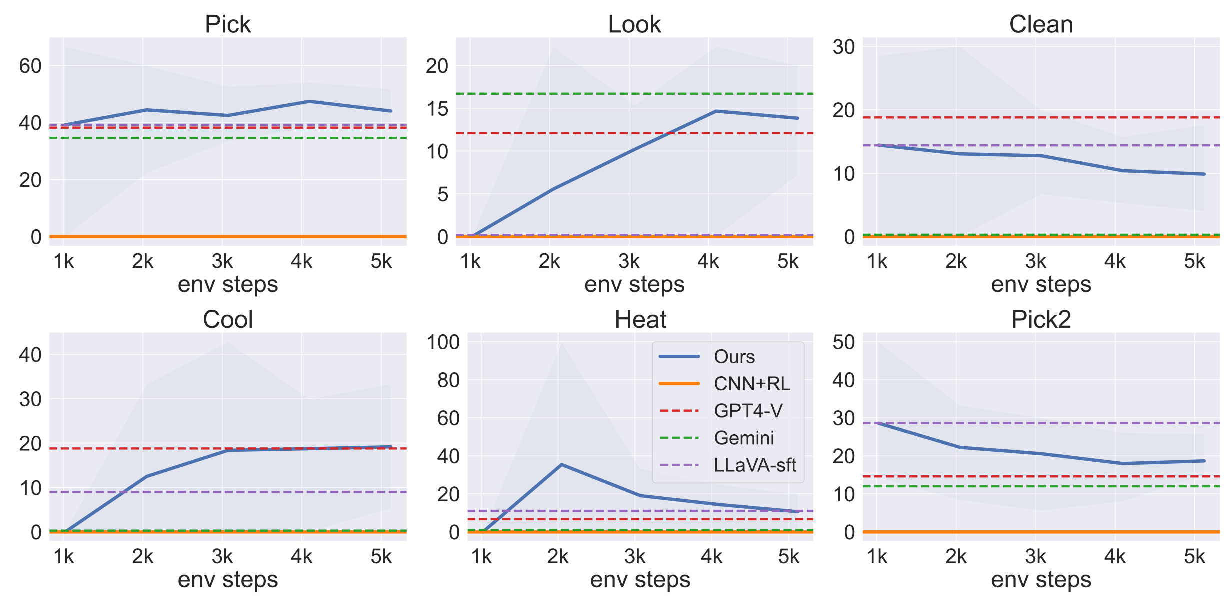


Figure C.12: **Episode success rates our method on different types of tasks from `alfworld`.** The curves come from the same run as experiments from the `alfworld` figure of Figure 4.5.

Failure Examples of `Points24`

Ground Truth Cards: [2, 8, 5, J]
Error Type: Fail to recognize all numbers
{
"cards": [10, 10, 5, 9],
"formula": "10+",
"thoughts": "`'10+'` is an incomplete formula, since `'10+10-5+9=24'`, I should append `'10'` to the current formula",
"action": "10"
}

Figure C.13: **Failure example of `Points24` due to incorrect perception.** In this example, the VLM fails to recognize the four numbers from the cards correctly as it only recognizes 2 numbers correctly. The reasoning part is correct, as it indeed generates a formula that equals 24.

Ground Truth Cards: [J, K, 5, 6]
Error Type: Fail to recognize all numbers

{
"cards": [10, 10, 5, 7],
"formula": "10+10+5+",
"thoughts": "'10+10+5' is an incomplete formula, since '10+10+5+7=24', I should append '7' to the current formula",
"action": "7"
}

Figure C.14: **Failure example of `Points24` due to incorrect perception and reasoning.** In this example, the VLM fails to recognize the four numbers from the cards correctly as it only recognizes 3 numbers correctly. The reasoning part is also incorrect, 10+10+5+7 does not equal 24.

C.3 Details on Supervised Fine-tuning Data

In this section, we provide detailed descriptions of the supervised fine-tuning data for all tasks.

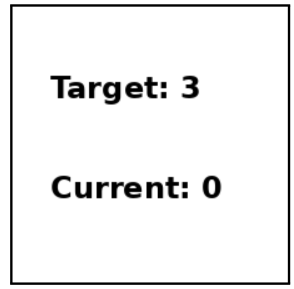
NumberLine

In the `NumberLine` task, we collect 20k instruction-following expert data samples using a task solver. See examples with and without CoT in Figure C.15 and C.16, respectively.

Inputs:

You are playing a game called number line. You will see a target number and a current number in the image. And your goal is to move the current number closer to the target by choosing either adding or subtracting one to the current number. Your response should be a valid json file in the following format:

{
"current number": "x",
"target number": "x",
"thoughts": {first read out the current and target number, then think carefully about which action to choose},
"action": "-" or "+"
}

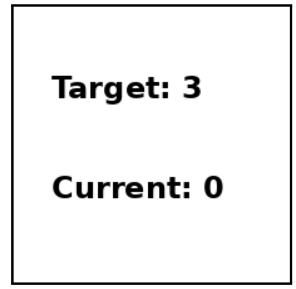


Responses:

{
"current number": "0",
"target number": "3",
"thoughts": "The current number is 0, which is smaller than the target number is 3. To move the current number closer to the target, I should increase the current number by choosing +.",
"action": "+"
}

Figure C.15: An example of the supervised fine-tuning data for `NumberLine` with CoT.

Inputs:
You are playing a game called number line. You will see a target number and a current number in the image. And your goal is to move the current number closer to the target by choosing either adding or subtracting one to the current number. Your response should be a valid json file in the following format:
{
"action": "-" or "+"
}



Responses:
{
"action": "+"
}

Figure C.16: An example of the supervised fine-tuning data for `NumberLine` without CoT.

Blackjack

In the `Blackjack` task, we collect 50k instruction-following expert data samples following an open-sourced solver.[3] See examples with and without CoT in Figure C.17 and C.18, respectively.

Inputs:
You are a blackjack player. You are observing the current game state, you can choose between ['stand', 'hit']. Your response should be a valid json file in the following format:
{
"thoughts": "{first describe your total points and the dealer's total points then think about which action to choose}",
"action": "stand" or "hit"
}

Responses:
{
"thoughts": "I have 13 points and the dealer has 8 points. I think I should hit.",
"action": "hit"
}

Figure C.17: An example of the supervised fine-tuning data for `Blackjack` with CoT.

[3] https://www.cs.cmu.edu/~chengwen/blackjack/blackjack.html

Inputs:

You are a blackjack player. You are observing the current game state, you can choose between `['stand', 'hit']`. Your response should be a valid json file in the following format:

{
"action": "stand" or "hit"
}

Responses:

{
"action": "hit"
}

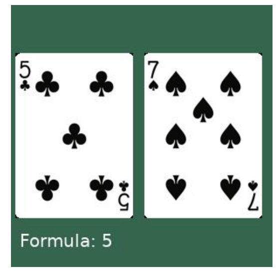
Figure C.18: An example of the supervised fine-tuning data for `Blackjack` without CoT.

EZPoints

In the `EZPoints` task, we directly collect 50k instruction-following expert data samples using a task solver. See examples with and without CoT in Figure C.19 and C.20, respectively.

Inputs:

You are an expert card game player. You are observing two cards in the image. You are observing the current formula: '5'. You can choose between ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '+', '*', '=']. The number or operator you choose will be appended to the current formula. Note that 'J', 'Q', and 'K' count as '10'. Your goal is to output a formula that evaluates to 12, and each number can only be used once. Your response should be a valid json file in the following format:



{
"cards": [x, y],
"current formula": '5',
"thoughts": {First check whether the current formula 'z' is complete. If the current formula 'z' is complete, output '='. Otherwise consider which number or operator should be appended to the current formula to make it equal 12.}
"action": "{number}" or "{operator}"
}

Responses:

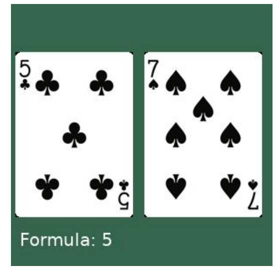
{
"cards": [7, 5],
"formula": "5",
"thoughts": "'5' is an incomplete formula, since '5+7=12', I should append '+' to the current formula", "action": "+"
}"

Figure C.19: An example of the supervised fine-tuning data for `EZPoints` with CoT.

Inputs:

You are an expert card game player. You are observing two cards in the image. You are observing the current formula: `'5'`. You can choose between [`'1'`, `'2'`, `'3'`, `'4'`, `'5'`, `'6'`, `'7'`, `'8'`, `'9'`, `'10'`, `'+'`, `'*'`, `'='`]. The number or operator you choose will be appended to the current formula. Note that `'J'`, `'Q'`, and `'K'` count as `'10'`. Your goal is to output a formula that evaluates to 12, and each number can only be used once. Your response should be a valid json file in the following format:

{
"action": "{number}" or "{operator}"
}



Responses:

{
"action": "+"
}"

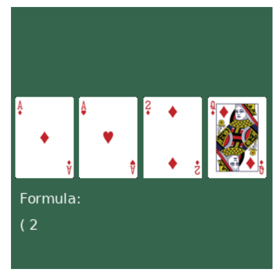
Figure C.20: An example of the supervised fine-tuning data for `Blackjack` without CoT.

Points24

In the Points24 task, we directly collect 50k instruction-following expert data samples using a task solver. See examples with and without CoT in Figure C.21 and C.22, respectively. Note that the supervised fine-tuning data has a slightly different distribution as the states in the Points24 environment, since the expert data always have a viable solution to 24, while not all states from Points24 has a viable solution to 24.

Inputs:

You are an expert card game player. You are observing two cards in the image. You are observing the current formula: '5'. You can choose between ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '+', '*', '=']. The number or operator you choose will be appended to the current formula. Note that 'J', 'Q', and 'K' count as '10'. Your goal is to output a formula that evaluates to 12, and each number can only be used once. Your response should be a valid json file in the following format:



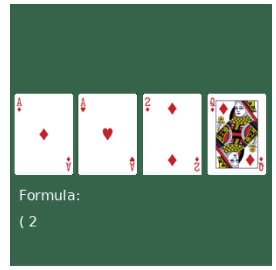
{
"cards": [x, y],
"current formula": '(2',
"thoughts": {First check whether the current formula 'z' is complete. If the current formula 'z' is complete, output '='. Otherwise consider which number or operator should be appended to the current formula to make it equal 12.}
"action": "{number}" or "{operator}"
}

Responses:

{
"cards": [1, 2, 10, 1],
"formula": "(2",
"thoughts": "'(2' is an incomplete formula, since '(2+10)*(1+1)=24', I should append '+' to the current formula",
"action": "+"
}"

Figure C.21: An example of the supervised fine-tuning data for Points24 with CoT.

Inputs:
You are an expert card game player. You are observing two cards in the image. You are observing the current formula: '5'. You can choose between ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '+', '*', '=']. The number or operator you choose will be appended to the current formula. Note that 'J', 'Q', and 'K' count as '10'. Your goal is to output a formula that evaluates to 12, and each number can only be used once. Your response should be a valid json file in the following format:
{
"action": "{number}" or "{operator}"
}



Responses:
{
"action": "+"
}"

Figure C.22: An example of the supervised fine-tuning data for Points24 without CoT.

ALFWorld

The data collection procedure of the alfworld embodied environment is slightly different than the gym_cards, as we do not have solvers to generate the instruction-following data, even with the expert text action. Therefore, we take a step back and directly use our prompt as presented in Figure C.10 (or Figure C.23) to collect 1k episodes (around 45k) instruction-following data from GPT4-V [OpenAI, 2023c] with the CoT reasoning for the instruction-following fine-tuning with CoT. As for the case without CoT, we directly clean the collected CoT by removing the CoT reasonings. See examples with and without CoT in Figure C.23 and C.24, respectively.

Inputs:

Your are an expert in the ALFRED Embodied Environment. You are also given the following text description of the current scene: ['You arrive at loc 0. The cabinet 1 is open. On the cabinet 1, you see a pan 1, a kettle 1, a winebottle 1, a apple 1, a stoveknob 1, a stoveknob 2, a stoveknob 3, a stoveknob 4, a knife 1, a saltshaker 1, and a bread 1.']. Your task is to put a cool mug in cabinet. Your admissible actions of the current situation are: ['go to countertop 1', 'go to cabinet 2', 'go to countertop 2', 'go to stoveburner 1', 'go to drawer 1', 'go to drawer 2', 'go to drawer 3', 'go to stoveburner 2', 'go to stoveburner 3', 'go to stoveburner 4', 'go to drawer 4', 'go to cabinet 3', 'go to cabinet 4', 'go to microwave 1', 'go to cabinet 5', 'go to cabinet 6', 'go to cabinet 7', 'go to sink 1', 'go to sinkbasin 1', 'go to fridge 1', 'go to toaster 1', 'go to coffeemachine 1', 'go to cabinet 8', 'go to drawer 5', 'go to drawer 6', 'go to drawer 7', 'go to drawer 8', 'go to shelf 1', 'go to shelf 2', 'go to countertop 3', 'go to shelf 3', 'go to drawer 9', 'go to garbagecan 1', 'open cabinet 1', 'close cabinet 1', 'take pan 1 from cabinet 1', 'take kettle 1 from cabinet 1', 'take winebottle 1 from cabinet 1', 'take apple 1 from cabinet 1', 'take stoveknob 1 from cabinet 1', 'take stoveknob 2 from cabinet 1', 'take stoveknob 3 from cabinet 1', 'take stoveknob 4 from cabinet 1', 'take knife 1 from cabinet 1', 'take saltshaker 1 from cabinet 1', 'take bread 1 from cabinet 1', 'inventory', 'look', 'examine cabinet 1']. Your response should be a valid json file in the following format:

{
"thoughts": "first describe what do you see in the image using the text description, then carefully think about which action to complete the task. ",
"action": "an admissible action"
}

Responses:

{
"thoughts": "Based on the text description, I am at loc 0 with an open cabinet 1 in front of me. Items such as a pan, kettle, wine bottle, apple, several stoveknobs, a knife, saltshaker, and bread are visible on the cabinet 1. To find a cool mug to put in the cabinet, I may need to first locate a mug, which might be in one of the cabinets or drawers. Since the task does not involve any items currently in cabinet 1, and the mug is not mentioned in the description, my best course of action would be to start by searching the other cabinets or drawers for the mug.",
"action": "go to cabinet 2"
}"

Figure C.23: An example of the supervised fine-tuning data for `alfworld` with CoT.

Inputs:

Your are an expert in the ALFRED Embodied Environment. You are also given the following text description of the current scene: ['You arrive at loc 0. The cabinet 1 is open. On the cabinet 1, you see a pan 1, a kettle 1, a winebottle 1, a apple 1, a stoveknob 1, a stoveknob 2, a stoveknob 3, a stoveknob 4, a knife 1, a saltshaker 1, and a bread 1.']. Your task is to put a cool mug in cabinet. Your admissible actions of the current situation are: ['go to countertop 1', 'go to cabinet 2', 'go to countertop 2', 'go to stoveburner 1', 'go to drawer 1', 'go to drawer 2', 'go to drawer 3', 'go to stoveburner 2', 'go to stoveburner 3', 'go to stoveburner 4', 'go to drawer 4', 'go to cabinet 3', 'go to cabinet 4', 'go to microwave 1', 'go to cabinet 5', 'go to cabinet 6', 'go to cabinet 7', 'go to sink 1', 'go to sinkbasin 1', 'go to fridge 1', 'go to toaster 1', 'go to coffeemachine 1', 'go to cabinet 8', 'go to drawer 5', 'go to drawer 6', 'go to drawer 7', 'go to drawer 8', 'go to shelf 1', 'go to shelf 2', 'go to countertop 3', 'go to shelf 3', 'go to drawer 9', 'go to garbagecan 1', 'open cabinet 1', 'close cabinet 1', 'take pan 1 from cabinet 1', 'take kettle 1 from cabinet 1', 'take winebottle 1 from cabinet 1', 'take apple 1 from cabinet 1', 'take stoveknob 1 from cabinet 1', 'take stoveknob 2 from cabinet 1', 'take stoveknob 3 from cabinet 1', 'take stoveknob 4 from cabinet 1', 'take knife 1 from cabinet 1', 'take saltshaker 1 from cabinet 1', 'take bread 1 from cabinet 1', 'inventory', 'look', 'examine cabinet 1']. Your response should be a valid json file in the following format:

{
"action": "an admissible action"
}

Responses:

{
"action": "go to cabinet 2"
}"

Figure C.24: An example of the supervised fine-tuning data for `alfworld` without CoT.

Appendix D

Understanding Foundation Model Post-Training

D.1 Details on the General Points Environment

In this section, we demonstrate the design details for `GeneralPoints` mentioned in Section 5.4. We first present the data used for this environment (Appendix D.1). Then, we show examples of the environment's transition dynamics (Appendix D.1), followed by a description of key arguments and reward design specification (Appendix D.1).

Data

`GeneralPoints` card quadruples are sampled from a deck of 52 standard poker cards. Each sampled quadruple is guaranteed to have at least one solution equals the target point, i.e. 24. We ensure this by using an expert solver during the sampling process.

Detailed Examples on the Transition Dynamics

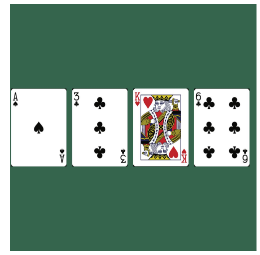
As shown in Figure D.1 and Figure D.2, we treat the system prompt as $\boldsymbol{v}_0^{\text{in}}$ and then subsequently appending the future outputs $\boldsymbol{v}_{1:t}^{\text{out}}$ and verifier info $\boldsymbol{v}_{1:t}^{\text{ver}}$ into the prompt for getting the $t+1$ output. Figure D.1 provides an example with the visual inputs, while Figure D.2 shows the language only case.

Additional Eetails on the Environmental Design

Arguments. The `GeneralPoints` environment supports the following configurable arguments:

- Target point: Any positive integer
- Face cards rule: Two options

System Prompt (v_0^{in})



[Task Description]
You are an expert 24 points card game player. You are observing these four cards in the image. Note that 'J', 'Q', and 'K' count as '10', and each card must be used once. Your goal is to output a formula that evaluates to 24 using numbers from the cards and operators such as '+', '-', '*', '/', '(', ')', and '='.

[Output]
Your response should be a valid json file in the following format:
{
"cards": [x, y, z, w], where 'J', 'Q', and 'K' count as '10',
"number": [a, b, c, d], where a, b, c, and d are the numbers on the cards,
"formula": "an equation that equals 24",
}

Appending model and verifier outputs to obtain v_t^{in}

$v_t^{\text{in}} = [v_0^{\text{out}}, v_0^{\text{ver}}, v_1^{\text{out}}, v_1^{\text{ver}}, \ldots, v_{t-1}^{\text{out}}, v_{t-1}^{\text{ver}}]$ $\quad \triangleright\ v_t^{\text{in}} = \text{concat}\left(v_0^{\text{in}}, [v_k^{\text{out}}, v_k^{\text{ver}}]_{k=0}^{t-1}\right)$

Model output (v_t^{out})

{
"cards": ['A', '3', 'K', '6'],
"number": [1, 3, 13, 6],
"formula": "(1+6)*3+13=24",
}

Verifier Output (v_t^{ver})

You failed this trial because your formula is incorrect. $\triangleright\ v_{t+1}^{\text{in}} = \text{concat}(v_t^{\text{in}}, v_t^{\text{out}}, v_t^{\text{ver}})$

Figure D.1: **An example of our prompt update** for constructing v_{t+1}^{in} using v_t^{in}, v_t^{out} and v_t^{ver}. This example provides an optional vision input for VLMs, adding a visual recognition challenge. The brown parts marks the task and related information, and the purple parts denote the state (s_t) specific info. The blue and red describe the output from the model and verifier, respectively.

  - 'J', 'Q', and 'K' all count as '10'
  - 'J', 'Q', and 'K' count as '11', '12', and '13' respectively

- Card sampling: Two options
  - Sample 4 cards without replacement from a deck of 52 poker cards

  - Sample at least one card from 'J', 'Q', and 'K'
- Card color: Three options
  - Black suits only: ♣, ♠.
  - Red suits only: ♥, ♦.
  - All suits: ♠, ♥, ♣, ♦.

For all experiments, we fix the target point at 24. In Figure 5.5, training and in-domain evaluation use the rule where face cards count as '10'. For out-of-domain evaluation, we use the alternative face cards rule and require at least one face card, forcing calculations with numbers above 10 that are not encountered during training. For visual distribution shift experiments (Section 5.5), we train the model on black suits ♠, ♣ and evaluate out-of-domain performance on red suits ♥, ♦.

Reward design. An episode terminates when either a correct equation is generated or the maximum verification step of 5 is reached. The reward function is as follows:

- $r = 5$: For generating a legal equation that equals the target point
- $r = -1$: For legal equations using each card once but not equaling the target point
- $r = -1$: For exceeding maximum verification step
- $r = -2$: For legal equations containing numbers not among the given choices
- $r = -3$: For all other illegal equations

In the vision-language variant (`GeneralPoints-VL`), an additional penalty of $r = -1.5$ is applied when the agent fails to correctly recognize the given cards.

D.2 Details on the V-IRL Environment

Similar to Appendix D.1, we present the design details for `V-IRL` discussed in Section 5.4. First, we introduce the database used for this environment (Appendix D.2) and demonstrate transition examples (Appendix D.2). We then describe the environment by explaining its fundamental component—*route*. Finally, we outline our modifications and reward design choices made to adapt the original `V-IRL` for reinforcement learning training (Appendix D.2).

Data

Leveraging the data collection pipeline of Yang et al. [2024a], we construct a training database with 1000 unique routes from New York City. We evaluate all rule-variant experiments and visual in-distribution experiments using randomly sampled routes from this database. For visual out-of-distribution experiments, we directly adopt the VLN mini benchmark from Yang et al. [2024a]. This benchmark consists of 18 distinct routes across nine cities: Milan, New Delhi, Buenos Aires, London, Hong Kong, New York,[1] Melbourne, Lagos, and San Francisco, with two routes per city.

Detailed Examples on the Transition Dynamics

We provide detailed transition examples of the `V-IRL` environment in Figure D.3 (vision and language) and Figure D.4 (pure language).

Additional Details on the Environmental Design

Concept of *route*. The route serves as the fundamental navigation object in the `V-IRL` environment. As illustrated in Figure 5.4, each route corresponds to a real-world path with associated language instructions and visual signals. Using Figure 5.4 as an example, a route comprises:

- Destination: Shuka
- Starting point: Start
- Turning points: The Dutch, Lola Taverna
- Straight road: Roads connecting turning points, starting point, and destination
- Street views: 360-degree panoramic views at each movable point
- Oracle information: Expert observation data for each movable point
- Expert trajectory
- Instruction

Although the instructions in Figures 5.4, D.3 and D.4 are presented in different formats, they convey equivalent information, with Figure 5.4 using natural language.

[1]These NYC routes in the VLN mini benchmark do not overlap with our training data.

Simplification and arguments. We simplify the original `V-IRL` design from Yang et al. [2024a] to better accommodate RL training. The modifications include eliminating the 2-stage navigation pipeline that required a separate visual detector for street view processing, and removing online queries to reduce training time and cost. Our `V-IRL` environment contains 2 additional configuration arguments compared with the original design:

- Action space: two options
  - Absolute direction:
    "turn_direction(x)" where x∈{'north', 'northeast', 'east', 'southeast', 'south', 'southwest', 'west', 'northwest'}, "forward()", "stop()"
  - Relative direction:
    "turn_direction(x)" where x∈{'left', 'right', 'slightly left', 'slightly right'}, "forward()", "stop()"
- Maximum straight road length: any positive integer

The action space argument accommodates the rule variants described in Section 5.4. For experiments shown in Figure 5.5, we use absolute direction action space during training and in-domain evaluation, while using the alternative rule for out-of-domain evaluation. We implement a maximum straight road length to limit the number of movable coordinates between turning points, preventing sequences of repetitive "forward()" actions. We conduct visual distribution shift experiments (Section 5.5) via training the model on New York City regions and evaluating the out-of-domain performance on the worldwide navigation routes from the benchmark released by Yang et al. [2024a].

Reward design. An episode terminates when either the navigation agent stops at the destination or the maximum verification step of 2 is reached. The reward function is as follows:

- $r = 1$: For generating a correct action at the current coordinate
- $r = -1$: For generating wrong action at the current coordinate
- $r = -1$: For exceeding maximum verification step
- $r = -1.5$: For failed detection of landmarks

System Prompt (v_0^{in})

[Task Description]
You are an expert 24 points card game player. You are observing these four cards in the image. Note that 'J', 'Q', and 'K' count as '11', '12', and '13' respectively, and each card must be used once. Your goal is to output a formula that evaluates to 24 using numbers from the cards and operators such as '+', '-', '*', '/', '(', ')', and '='.

[Input]
Cards: ['A', '3', 'K', '6']

[Output]
Your response should be a valid json file in the following format:
{
"cards": [x, y, z, w], where 'J', 'Q', and 'K' count as '10',
"number": [a, b, c, d], where a, b, c, and d are the numbers on the cards,
"formula": "an equation that equals 24",
}

Appending model and verifier outputs to obtain v_t^{in}

$\boldsymbol{v}_t^{\text{in}} = [\boldsymbol{v}_0^{\text{out}}, \boldsymbol{v}_0^{\text{ver}}, \boldsymbol{v}_1^{\text{out}}, \boldsymbol{v}_1^{\text{ver}}, \ldots, \boldsymbol{v}_{t-1}^{\text{out}}, \boldsymbol{v}_{t-1}^{\text{ver}}]$ $\qquad \triangleright\ \boldsymbol{v}_t^{\text{in}} = \text{concat}\left(\boldsymbol{v}_0^{\text{in}}, [\boldsymbol{v}_k^{\text{out}}, \boldsymbol{v}_k^{\text{ver}}]_{k=0}^{t-1}\right)$

Model output (v_t^{out})

{
"cards": ['A', '3', 'K', '6'],
"number": [1, 3, 13, 6],
"formula": "(1+6)*3+13=24",
}

Verifier Output (v_t^{ver})

You failed this trial because your formula is incorrect. $\triangleright\ \boldsymbol{v}_{t+1}^{\text{in}} = \text{concat}(\boldsymbol{v}_t^{\text{in}}, v_t^{\text{out}}, v_t^{\text{ver}})$

Figure D.2: **An example of our prompt update** for constructing $\boldsymbol{v}_{t+1}^{\text{in}}$ using $\boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{out}}$ and $\boldsymbol{v}_t^{\text{ver}}$. This example provides an optional vision input for VLMs, adding a visual recognition challenge. The brown parts marks the task and related information, and the purple parts denote the state (s_t) specific info. The blue and red describe the output from the model and verifier, respectively.

System Prompt (v_0^{in})

[Task Description]
You are an expert in navigation. You will receive a sequence of instructions to follow while observing your surrounding street views. You are also provided with your observation and action history in text. your goal is to take the action based on the current observation and instruction.

[Instruction]
1. First, turn left to face east.
2. Move forward until you reach the next intersection where Hotel 32One is on your right behind.
3. Turn right to face north.
4. Move forward until you reach the next intersection where Dragon Gate Chinatown SF is on your right front.
5. Turn left to face east.
6. Move forward until the destination Café de la Presse is on your right.

[Current observation]
You observe a 2x2 grid of street view images with the following headings:
[front, right
back, left]
You need to identify if any of the landmarks in the instruction are visible in the street view grid.

[Action space]
- "forward()": indicates moving forward for 1 step;
- "turn_direction(x)": indicates turn direction to the target heading, where x∈['north', 'northeast', 'east', 'southeast', 'south', 'southwest', 'west', 'northwest'];
- "stop()": indicates the navigation is finished;

[Observations and actions sequence]
O_1: No landmarks nearby;
A_1: turn_direction(east)
O_2: No landmarks nearby;
A_2: forward()
O_3: No landmarks nearby;
A_3: forward()
O_4: You observe an image of 4 views; You observe an intersection
A_4:

[Output]
Your response should be a valid json file in the following format:
{
"current observation": latest observation from the street view grid,
"current instruction": analyze the full instruction and identify the sentence to be executed,
"action": the action to be taken chosen from the action space,
}

Appending model and verifier outputs to obtain v_t^{in}

$v_t^{\text{in}} = [v_0^{\text{out}}, v_0^{\text{ver}}, v_1^{\text{out}}, v_1^{\text{ver}}, \ldots, v_{t-1}^{\text{out}}, v_{t-1}^{\text{ver}}]$ $\triangleright$ $v_t^{\text{in}} = \text{concat}\left(v_0^{\text{in}}, [v_k^{\text{out}}, v_k^{\text{ver}}]_{k=0}^{t-1}\right)$

Model output (v_t^{out})

{
"current observation": "Hotel 32One is on my right behind; I observe an intersection",
"current instruction": "Turn right to face north.",
"action": "turn_direction(north)",
}

Verifier Output (v_t^{ver})

Correct solution. $\triangleright$ Move to the next scene upon correct action

Figure D.3: **An example of our prompt update** for constructing v_{t+1}^{in} using v_t^{in}, v_t^{out} and v_t^{ver}. The brown parts marks the task and related information, and the purple parts denote the state (s_t) specific info. The blue and red describe the output from the model and verifier, respectively.

System Prompt (v_0^{in})

[Task Description]
You are an expert in navgation. You will receive a sequence of instructions to follow. You are also provided with your observation and action histroy in text. Your goal is to first analyze the instruction and identify the next sentence to be executed. Then, you need to provide the action to be taken based on the current observation and instruction.

[Instruction]
1. First, turn left to face east.
2. Move forward until you reach the next intersection where Hotel 32One is on your right behind.
3. Turn right to face north.
4. Move forward until you reach the next intersection where Dragon Gate Chinatown SF is on your right front.
5. Turn left to face east.
6. Move forward until the destination Café de la Presse is on your right.

[Action space]
- "forward()": indicates moving forward for 1 step;
- "turn_direction(x)": indicates turn direction to the target heading, where x∈['north', 'northeast', 'east', 'southeast', 'south', 'southwest', 'west', 'northwest'];
- "stop()": indicates the navigation is finished;

[Observations and actions sequence]
O_1: No landmarks nearby;
A_1: turn_direction(east)
O_2: No landmarks nearby;
A_2: forward()
O_3: No landmarks nearby;
A_3: forward()
O_4: Hotel 32One is on your right behind; You observe an intersection
A_4:

[Output]
Your response should be a valid json file in the following format:
{
"current observation": latest observation from the street view grid,
"current instruction": analyze the full instruction and identify the sentence to be executed,
"action": the action to be taken chosen from the action space,
}

Appending model and verifier outputs to obtain v_t^{in}

$v_t^{\text{in}} = [v_0^{\text{out}}, v_0^{\text{ver}}, v_1^{\text{out}}, v_1^{\text{ver}}, \ldots, v_{t-1}^{\text{out}}, v_{t-1}^{\text{ver}}]$ ▷ $v_t^{\text{in}} = \text{concat}\left(v_0^{\text{in}}, [v_k^{\text{out}}, v_k^{\text{ver}}]_{k=0}^{t-1}\right)$

Model output (v_t^{out})

{
"current observation": "Hotel 32One is on my right behind; I observe an intersection",
"current instruction": "Turn right to face north.",
"action": "turn_direction(north)",
}

Verifier Output (v_t^{ver})

Correct solution. ▷ Move to the next scene upon correct action

Figure D.4: **An example of our prompt update** for constructing v_{t+1}^{in} using v_t^{in}, v_t^{out} and v_t^{ver}. This example provides an optional vision input for VLMs, adding a visual recognition challenge. The brown parts marks the task and related information, and the purple parts denote the state (s_t) specific info. The blue and red describe the output from the model and verifier, respectively.

D.3 Experimental Setup

This section details the experimental setup used in Section 5.5. We first describe our data collection setup for supervised fine-tuning (Appendix D.3). Then, we present the training pipeline (Appendix D.3). Finally, we describe our evaluation metrics and the statistical tools used for generating plots (Appendix D.3).

Data

SFT data collection. As illustrated in Figures D.1 to D.4, `GeneralPoints` and `V-IRL` environments naturally align with prompt-response dialogue structures. We create training samples by pairing each system prompt with its corresponding expert response. All SFT experiments in the main body use optimal single-turn prompt-response pairs, without any verification or revision steps.

SFT on sub-optimal trajectories To examine how more diverse SFT data affects the out-of-distribution performance of SFT, we conduct an ablation study on `GP-L` using sub-optimal trajectories as training data. Unlike expert prompt-response pairs, these sub-optimal trajectories include errors and verification messages in their prompts. This format aligns with evaluation scenarios where multiple verification iterations are allowed, similar to the data being used for the downstream RL training. In Figure D.5, we observe that SFT still merely memorizes the training data with degraded out-of-distribution performance. This evidence suggests that memorization occurs due to the fundamental nature of SFT training rather than the SFT data.

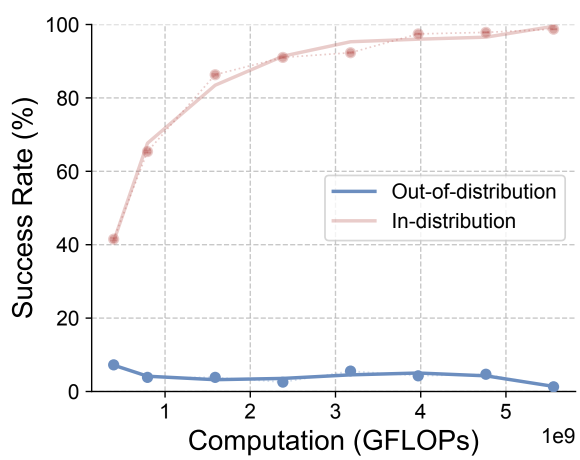


Figure D.5: **SFT experiments on `GP-L` with suboptimal trajectories.** Similar to results in Figure 5.5, SFT overfits the training data even we increase the trajectory diversity.

Training Pipeline

As illustrated in Section 5.5, we follow the training pipeline by RL4VLM [Zhai et al., 2024a], where we first initialize the model with SFT, then separately scale up the compute for SFT and RL [Schulman et al., 2017], starting from this initialized model. For all experiments of SFT and RL in the main body, we tune all components using a shared learning rate per experiment. All training experiments are conducted on an 8 H800 machine (80GB).

Evaluation Metric

Per-step accuracy. We report the per-step accuracy for `V-IRL-VL` task in Figures 5.5 and 5.6. An individual step is considered correct when the model's chosen action matches the expert trajectory at that position. Note that intermediate verification steps are counted as independent samples here.

Success rate. We report the success rate (%) of `GP-L`, `GP-VL`, `V-IRL-L` and `V-IRL-VL` in Figures 5.5 and 5.6. In the `GeneralPoints` task, success is defined as succeeding at least once during the inference time verification. In the `V-IRL` task, a sample is recorded as success when the model takes correct action at each movable point on the route.

Computation estimation. We estimate the FLOPs for training X following the similar manner of [Snell et al., 2024, Hoffmann et al., 2023], where $X_{train} = 6ND_{train}$ and $X_{inference} = 2ND_{inference}$. Here, N represents the model parameters and D_{train} represents the number of tokens during training. Suppose our SFT and RL experients starts from a checkpoint trained on D_{init} tokens, we can estimate the training computation of SFT and RL via the following equations:

$$\begin{aligned} X_{SFT} &= 6N(D_{init} + D_{SFT}) \\ X_{RL} &= 6N(D_{init} + D_{RL}) + 2ND_{buffer} \end{aligned}$$

Note that the used on-policy RL algorithm PPO [Schulman et al., 2017] contains iterative stages of replay buffer collection and optimization, hence requiring additional inference computation. For simplicity, we approximate the term via:

$$\begin{aligned} D_{buffer} &\approx \frac{E\bar{d}_i\bar{d}_o}{D_{RL}} \cdot D_{RL} \\ &= \lambda D_{RL} \end{aligned}$$

where $E \in \mathbb{N}$ denotes the number of auto-regressive generation processes, $\bar{d}_i, \bar{d}_o$ denote average input tokens and output tokens. We estimate the λ for `GeneralPoints` and `V-IRL` as 6 and 5.1 respectively after calculation.

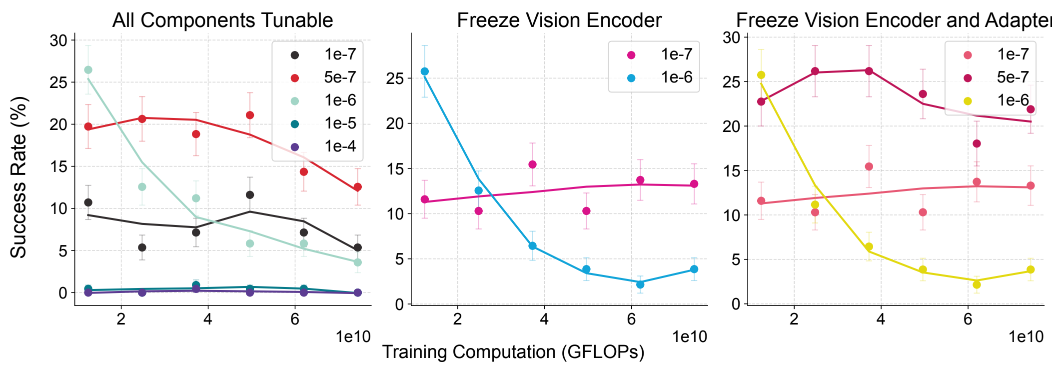


Figure D.6: **Ablation studies on `GeneralPoints-VL` SFT.** We ablate the learning rate and report the in-distribution episode success rate (%) of all experiments. None of the experiments shows an increasing trend beyond 30% success rate.

Line smoothing and error bar. All line plots in our paper adopt Savitzky–Golay filter with polynomial order 3 as smoothing function. We assume each evaluated data point follows a binomial distribution and approximate the standard error using $\sqrt{\frac{P(1-P)}{N}}$, where P is the demical success rate and N is the number of samples.

D.4 Additional Experimental Results

In this section, we provide additional experimental results that are not covered in the main body.

Ablation Studies on `GP-VL`

As mentioned in Section 5.6, we observe an abnormal phenomenon that SFT fails to achieve comparable in-distribution performance with RL (see Figure 5.5 subplot row 1 column 3). To further explore this, we conduct ablation studies over different hyperparameter choices.

SFT. We ablate the hyperparameter choices under the same task setting of `GP-VL` in Section 5.5. For experiments fine-tuning all parameters, we search learning rates from $\{1 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-5}, 1 \times 10^{-6}, 5 \times 10^{-7}, 1 \times 10^{-7}\}$. Freezing the vision encoder, we search learning rates $\{1 \times 10^{-6}, 1 \times 10^{-7}\}$. Freezing vision encoder and adapter, we search learning rates $\{1 \times 10^{-6}, 5 \times 10^{-7}, 1 \times 10^{-7}\}$. We provide the in-distribution success rate curve in Figure D.6.

RL. Finding suitable hyperparameters for RL experiments requires minimal effort. We conduct a search over learning rates $2 \times 10^{-6}, 1 \times 10^{-6}$, with the in-distribution success rate curves shown in Figure D.7. All parameters are tunable in our RL experiments.

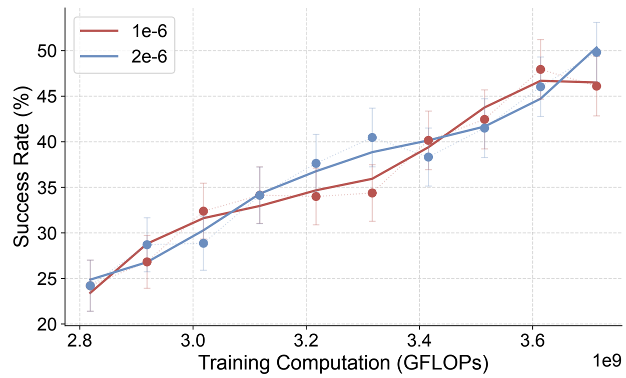


Figure D.7: **Ablation studies on `GeneralPoints-VL` RL.** Echoing Figure D.6, we ablate the learning rate and rreport the in-distribution episode success rate (%) of the two experiments. All components are tunable here.

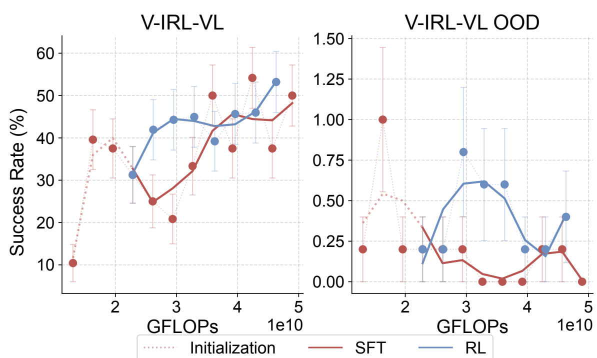


Figure D.8: **Overall success rate (%) - GFLOPs for `V-IRL-VL` under rule variants.** Due to the nature of the task requiring aggregating a trajectory of correct actions, neither training method achieves reasonable out-of-distribution performance.

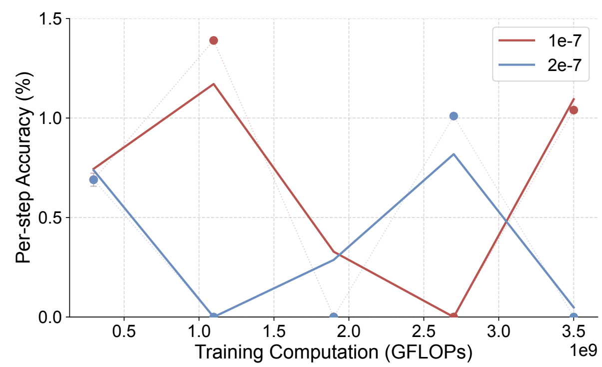


Figure D.9: **Out-of-distribution per-step accuracy (%) - GFLOPs for `V-IRL-VL` under rule variants with overfitted initial checkpoint.** Evaluation metric details can be found in Appendix D.3.

More results on `V-IRL-VL`

Echoing per-step accuracy results in Figure 5.5, we report the overall success rate of `V-IRL-VL` in Figure D.8. Due to the task's complexity, both training methods achieve overall success rates no higher than 1%. For `V-IRL`, the overall success rate is a significantly more demanding metric since it aggregates per-step errors. For example, a random policy achieving 10% per-step accuracy would achieve achieve only approximately $10^{-8}\%$ success rate on enough routes averaging 10 steps in length.

Failure Cases

In this section, we present 2 failure cases in our experiments as mentioned in Sections 5.5 and 5.6.

Without SFT, RL fails. In Figure 5.9, we present the training dynamics of failed RL experiments without SFT initialization. We additionally provide output examples of these experiments in Figure D.10, where the model tends to generate unstructured response and fail.

RL cannot save overfitted checkpoints. As shown in Figure D.9, RL cannot recover the out-of-distribution performance when initialized from a extremely overfitted checkpoint that has an initial per-step accuracy of less than 1%. We additionally provide an output example in Figure D.9, where the model fails to adjust to the new rule.

System Prompt (v_0^{in})

[Task Description]
You are an expert 24 points card game player. You are observing these four cards in the image. Note that 'J', 'Q', and 'K' count as '11', '12', and '13' respectively, and each card must be used once. Your goal is to output a formula that evaluates to 24 using numbers from the cards and operators such as '+', '-', '*', '/', '(', ')', and '='.

[Input]
Cards: ['2', 'J', '3', 'K']

[Output]
Your response should be a valid json file in the following format:
{
"cards": [x, y, z, w], where 'J', 'Q', and 'K' count as '10',
"number": [a, b, c, d], where a, b, c, and d are the numbers on the cards,
"formula": "an equation that equals 24",
}

Appending model and verifier outputs to obtain v_t^{in}

$\boldsymbol{v}_t^{\text{in}} = [\boldsymbol{v}_0^{\text{out}}, \boldsymbol{v}_0^{\text{ver}}, \boldsymbol{v}_1^{\text{out}}, \boldsymbol{v}_1^{\text{ver}}, \ldots, \boldsymbol{v}_{t-1}^{\text{out}}, \boldsymbol{v}_{t-1}^{\text{ver}}]$ $\triangleright$ $\boldsymbol{v}_t^{\text{in}} = \text{concat}\left(\boldsymbol{v}_0^{\text{in}}, [\boldsymbol{v}_k^{\text{out}}, \boldsymbol{v}_k^{\text{ver}}]_{k=0}^{t-1}\right)$

Failed model output (v_t^{out})

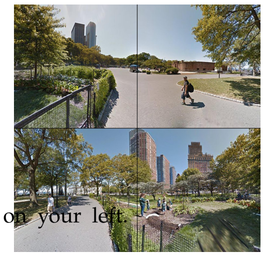
To solve this problem, we can use a brute force approach by generating all possible combinations of the cards and checking if any of them can be used to form an equation that equals 24.
Here is a Python solution for this problem:
"'python
def calculate_formula(cards):
# Define the mapping of cards to their corresponding numbers
...

Verifier Output (v_t^{ver})

You failed this trial because your formula is incorrect. $\triangleright$ $\boldsymbol{v}_{t+1}^{\text{in}} = \text{concat}(\boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{out}}, \boldsymbol{v}_t^{\text{ver}})$

Figure D.10: **Example model outputs without SFT initialization.** We record model responses using prompts similar to the one shown in Figure D.1. The results demonstrate that Llama-3.2-Vision-11B fails to follow instructions properly. We omit the long response which tries to solve the puzzle via code but fails to finish within finite context length.

System Prompt (v_0^{in})



[Task Description]
You are an expert in navigation. You will receive a sequence of instructions to follow while observing your surrounding street views. You are also provided with your observation and action history in text. Your goal is to first analyze the instruction and identify the next sentence to be executed. Then, you need to provide the action to be taken based on the current observation and instruction.
[Instruction]
1. First, turn right to face northwest.
2. Move forward until you reach next intersection where Korean War Memorial is on your left.
3. Turn left to face southwest.
4. Move forward until you reach next intersection where Korean War Memorial is on your left behind.
5. Turn right to face north.
6. Move forward until you reach next intersection.
7. Turn left to face east.
8. Move forward until you reach next intersection.
9. Turn left to face north.
10. Move forward until you reach next intersection.
11. Turn right to face east.
12. Move forward until you reach next intersection.
13. Turn left to face northeast.
14. Move forward until you reach next intersection where 9/11 Memorial & Museum is on your left.
15. Turn right to face northwest.
16. Move forward until you reach destination where The destination 9/11 Memorial & Museum is on your right front.
[Action space]
"forward()": indicates moving forward one step
"turn_direction(x)": indicates adjust the ego agent direction towards x direction. x could be any following ['left', 'right', 'slightly left', 'slightly right']
"stop()": indicates the navigation is finished.
[Observations and actions sequence]
O_1: You observe an image of 4 views;
A_1:
[Output]
{
"current observation": latest observation from the streetview grid,
"current instruction": analyze the full instruction and identify the sentence to be executed,
"action": the action to be taken chosen from the action space,
}

Appending model and verifier outputs to obtain v_t^{in}

$v_t^{\text{in}} = [v_0^{\text{out}}, v_0^{\text{ver}}, v_1^{\text{out}}, v_1^{\text{ver}}, \ldots, v_{t-1}^{\text{out}}, v_{t-1}^{\text{ver}}]$ $\quad \triangleright\ v_t^{\text{in}} = \text{concat}\left(v_0^{\text{in}}, [v_k^{\text{out}}, v_k^{\text{ver}}]_{k=0}^{t-1}\right)$

Failed model output (v_t^{out})

{
"current observation": "No landmarks nearby",
"current instruction": "Turn right to face northwest.",
"action": "turn_direction(northwest)",
}

Verifier Output (v_t^{ver})

Incorrect action. $\triangleright$ Fail this sample.

Figure D.11: **Failed example of `V-IRL` transition due to overfitting.** This phenomenon happens more frequently during scaling up supervised fine-tuning.