SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training

Tianzhe Chu[1*] Yuexiang Zhai[2*] Jihan Yang[3] Shengbang Tong[3]
Saining Xie[3] Sergey Levine[2] Yi Ma[1,2]
[1]HKU [2] UC Berkeley [3] NYU

Although supervised fine-tuning (SFT) and reinforcement learning (RL) enhance the performance of language and vision-language models, their roles in enhancing model generalization capabilities remain unclear. In this work, we study the different effects of SFT and RL on generalization and overfitting in rule-based tasks. We introduce `GeneralPoints`, an arithmetic reasoning card game, and adopt `V-IRL`, a real-world navigation environment, to assess how models trained with SFT/RL generalize to unseen rule variants. Our results consistently demonstrate that RL, especially when trained with an outcome-based reward, generalizes across both textual and visual rule variations. SFT, on the contrary, memorizes training data and does not generalize to unseen rules or out-of-distribution scenarios. Further analysis reveals that RL improves the model's underlying visual recognition capabilities, contributing to its generalization. However, we also find that SFT plays a necessary role in stabilizing RL training by providing a structured output format. These findings highlight the distinct learning dynamics of SFT and RL, demonstrating the importance of RL for acquiring generalizable rule-based knowledge in complex, multi-modal tasks.

1. Introduction

Post-training methods, such as supervised fine-tuning (SFT) or reinforcement learning (RL) improve the performance of large language models (LLM) and vision language models (VLM) on various tasks [2–8]. Specifically, SFT demonstrates significant performance gains across a range of tasks [4, 6, 7, 9, 10], while RL is crucial for aligning models with human preferences [3, 11, 12] or task outcomes [13, 14] hence improving reasoning capabilities [15, 16]. Although SFT and RL are commonly used, their learning dynamics remain underexplored. This paper investigates the different effects of SFT and RL on generalization and overfitting, specifically focusing on rule-based knowledge [17].

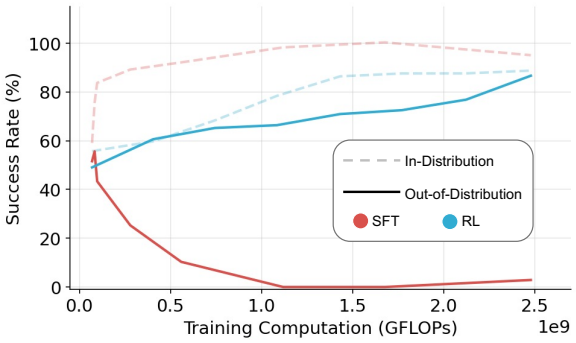


Figure 1: **A comparative study of RL and SFT on the visual navigation environment `V-IRL` [1].** An example of generalization experiments in a rule-based navigation task [1]. Out-of-distribution curves represent performance on the same task but under a different rule. See detailed descriptions of the task in Section 5.1.

For verifying learned knowledge, standard vision and language benchmarks [7, 18–22] may not be ideal, as the fixed pre-trained model may suffer from data contamination. This is because pre-trained models can suffer from data contamination, where the pretraining data (sourced from the internet) may inadvertently include information present in the test sets [23]. Inspired by Allen-Zhu and Li [24], we investigate knowledge general-

*Equal Contribution. Emails: `tianzhechu@gmail.com`, `simonzhai@berkeley.edu`

In submission to CPAL 2025 spotlight track.

ization in rule-based tasks. Specifically, we study whether performance gains on one rule, learned during SFT or RL, generalize to new rules within the same task.

We employ two rule-based tasks for our study. First, we introduce `GeneralPoints`, an original card game task similar to the `Points24` task from RL4VLM [14], designed to evaluate a model's *arithmetic reasoning capabilities*. In `GeneralPoints`, the model receives four cards (presented as text or images), and is required to compute a target number (24 by default) using each card's numerical value exactly once. Rule variations in `GeneralPoints` involve changing the target number and altering how suit cards ('J', 'Q', and 'K') are interpreted. Second, we utilize `V-IRL` [1], a real-world navigation environment, which focuses on *testing spatial reasoning capabilities*. The rule variants in `V-IRL` involves using an absolute or a relative orientation action space.

In both tasks, we observe that RL learns generalizable rules (expressed in text), where in-distribution performance gains also transfer to unseen rules. In contrast, supervised fine-tuning (SFT) appears to memorize training rules and fails to generalize (an example is shown in Figure 1). Beyond this rule-based generalization (in text), we further investigate generalization in the visual domain and observe that RL also generalizes to visual OOD tasks, whereas SFT continues to struggle. With this visual strength, our RL approach achieves state-of-the-art performance on the `V-IRL` mini benchmark, by **+33.8%** (44.0%→77.8%) [1]. 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 [25] improves visual recognition capabilities. Finally, while RL exhibits superior generalization compared to SFT, we show that SFT is still crucial for stable RL training, as SFT stabilizes the model's output format, enabling RL to achieve its performance gains.

2. Related Works

Post-Training. Post-training is crucial for enhancing model performance [5, 26–28]. This stage commonly utilizes large-scale supervised fine-tuning (SFT) [2, 29–33] and/or reinforcement learning (RL) [3, 11, 12, 14, 34, 35]. SFT adapts pre-trained language models to downstream tasks by training them on task-specific, often instruction-formatted datasets. Previous work, such as FLAN [2], demonstrate that fine-tuning on diverse instruction-tuning datasets significantly enhances zero-shot performance on unseen tasks. Furthermore, LIMA [33] 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 [3, 11, 12, 14, 34–36] has been primarily used to align models with human preferences or to train models to solve specific tasks [3, 11, 12] or using training the foundational model to solve a specific task with RL [14, 34, 35]. Our work differs from prior studies on both SFT and RL; we aim to comparatively analyze the generalization and memorization characteristics of SFT and RL, while previous studies have focused primarily on only one of these two post-training methods.

Overfitting and generalization in LLMs. A significant body of research studying the overfitting and generalization in neural networks [37–40]. In LLMs, overfitting can manifest as the model memorizing training data [38, 41], while generalization reflects the divergence between the model's output distribution and the pre-training data distribution [42]. Prior studies suggest that LLMs exhibit more overfitting on simpler, knowledge-intensive tasks and greater generalization on more complex, reasoning-intensive ones [43, 44]. For example, research [17, 23, 24, 45–48] demonstrates that LLMs develop reasoning skill sets beyond their training data by pre-computing reasoning graphs before the autoregressive generation, providing compelling evidence of generalization. Our study takes a different approach by investigating the role of different post-training paradigms on overfitting versus generalization in the context of rule-based tasks. We conduct controlled studies in both unimodal (LLM) and multimodal (VLM) settings, demonstrating that RL facilitates generalization more than SFT.

Scaling up inference-time compute. Recent research has increasingly focused on scaling up inference-time computation to improve model performance [15, 49–51]. Early studies [49, 50] prompt models to generate intermediate reasoning steps and extended responses before produc-

ing a final answer. Subsequent works [51–55] have demonstrated that fine-tuning verifiers during inference improves model accuracy, effectively utilizing test-time computation. Notably, recent findings [15] 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, allowing 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 is the key for RL to generalize. Together, we demonstrate that these techniques are effective across both training and inference stages.

Improving visual capability in VLMs. While VLMs have demonstrated remarkable skills in a wide range of challenging tasks, such as solving advanced college exam questions [56–58] and visual understanding tasks [1, 59], they also exhibit limitations in visual perception [60–63]. Prior efforts to enhance VLMs' visual perception include combining multiple visual encoders [7, 62, 64], curating high-quality SFT data [7, 65, 66], and improving the SFT training recipe by unfreezing the visual backbone [7, 67]. While prior works primarily focus on experiments during the SFT stage, our work demonstrates that RL can also improve visual perception, beyond its generalization capabilities.

3. Preliminaries

Standard RL terminologies. We adopt the standard notation from classic RL literature [68, 69], where we use $\mathcal{S}$ 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 of RL is to learn the a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the overall return $\max_{\pi \in \Pi} \mathbb{E}_\pi \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$. Without loss of generality, we use $\pi(a|s) \in [0, 1]$ to denote probability of π choosing a at s.

Adapting RL terminologies to foundation models. We adopt a multi-turn RL setting for foundation model training [14, 34, 35]. Let $\mathcal{V}$ represent the discrete and finite vocabulary (token) space. The input and output text spaces are denoted by $\mathcal{V}^m, \mathcal{V}^n$, 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 (VLMs), we define $\mathcal{O}$ as the space of all RGB images. The state space, denoted by $\mathcal{S}$ is dedined 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 utilize a verifier function $\text{VER} : \mathcal{V}^n \to \mathbb{R} \times \mathcal{V}^k$ to denote a verifier, which evaluates the outcome of $\boldsymbol{v}^{\text{out}}$ and generates an outcome-based reward function [25, 51, 70, 71] r along with textual information $\boldsymbol{v}^{\text{ver}}$. Mathematically, at time t, $\text{VER}(\boldsymbol{v}_t^{\text{out}}) \mapsto (r_t, \boldsymbol{v}_t^{\text{ver}})$, where we use r_t to denote $r(s_t, a_t)$ for simplicity.

Sequential revision with a verifier. For modeling state-action transition, we adopt the sequential revision formulation [51]. 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 formulation is provided in Figure 2, and an example of the state-action transition is shown in Figure 3.

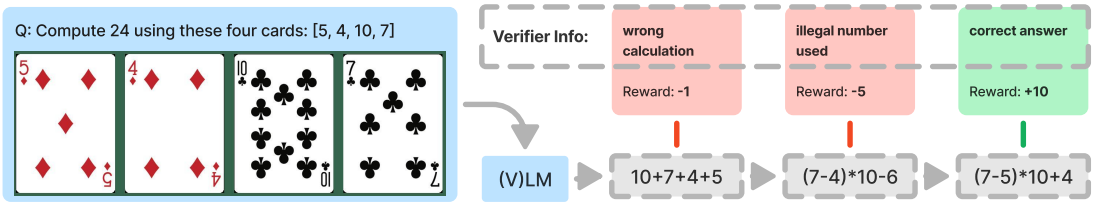


Figure 2: **An example of the sequential revision formulation with a verifier.** The model generate the next answer $\boldsymbol{v}_{t+1}^{\text{out}}$ conditioned on *all previous answers and information* $(\boldsymbol{v}_i^{\text{out}}, \boldsymbol{v}_t^{\text{ver}}, 0 \leq i \leq t)$ from the verifier.

System Prompt (v_0^{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 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}) and Verifier Output (v_t^{ver})

{Task related json outputs}, {You success/fail}. ▷ $v_{t+1}^{\text{in}} = \text{concat}(v_t^{\text{in}}, v_t^{\text{out}}, v_t^{\text{ver}})$

Figure 3: **An example of our prompt update.** for constructing v_{t+1}^{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.

4. Evaluation Tasks

To evaluate the generalization of different post-training methods, we select two rule-base tasks which require the model to tackle the task *varying rule sets*. First, we develop an original `GeneralPoints` environment designed to assess models' arithmetic abilities (Section 4.1). Second, we adopt the `V-IRL` environment [1] to examine foundation models' reasoning capabilities in an open-world visual navigation domain (Section 4.2).

4.1. The General Points Environment

Our original `GeneralPoints` environment is based on the `Points24` environment [14] and is designed to evaluate generalization of arithmetic abilities. Each state s of the environment contains 4 cards, described as text or presented as an image (see Figure 2 for a visual example of `GeneralPoints`). Note that when the input from `GeneralPoints` is presented in an image, it naturally introduces additional visual challenges requiring the VLM to *recognize cards before solving the equation*. The goal is to *produce an equation that equals a target number* using all 4 numbers from the cards *only once*.

Rule variants for verifying arithmetic reasoning capabilities. To study whether the model learns arithmetic operations or simply overfits by memorizing the post-training data, we introduce rule variations in `GeneralPoints`. These variations consist of: (1) changing the target number to different positive integers, and (2) 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. Detailed examples of the state-action transitions of `GeneralPoints` are provided in Appendix A.1.

4.2. The V-IRL Environment

While the `GeneralPoints` environment is designed to assess arithmetic abilities, we further utilize the `V-IRL` environment [1] to study the foundation model's spatial reasoning capability in an open-world navigation domain with *realistic visual input* (can be parsed into pure text description, see more details in Yang et al. [1] and an illustration in Figure 4). The major visual challenge in `V-IRL` involves *recognizing different landmarks from the visual observation*,[2] before taking an action, and the goal is to *navigate to a target location by following instructions containing spatial information.*

[2]See Figure 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 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 demonstration of the navigation procedure is shown at the top, with the corresponding navigation instructions displayed below. Visual observation-related information is highlighted in green, while action-related information is marked in orange.

Rule variations for assessing spatial reasoning. To evaluate whether the model possesses spatial knowledge or simply overfits 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'}. In contrast, the second variant employs a *relative orientation* action space, containing {'left', 'right', 'slightly left', 'slightly right'}. This relative configuration adjusts the model's 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 4, and a detailed example of an environment step using relative orientation is shown in Appendix B.1.

5. Results

In this section, we present experimental results focusing on the generalization of RL and SFT. Our experiments adopt the Llama-3.2-Vision-11B [72] as the backbone model. Following pipelines of RLHF [3] and RL4VLM [14], we initialize the model with SFT before running RL. We specifically study the following questions: (1) How does RL/SFT affect the generalization to different rules? (Section 5.1); (2) When the model contains visual component, how does RL/SFT affect the generalization to different visual variants? (Section 5.2); (3) How does RL/SFT affect the visual recognition capability in the case of VLM (Section 5.3) (4) What roles do SFT play for RL training? (Section 5.4).

5.1. SFT memorizes, RL generalizes

We evaluate the performance of different post-training methods on `GeneralPoints` and `V-IRL`, each of which contains multiple rule-base variants. For each task, we separately scale up the training computes of RL and SFT on a single rule. We consider the results on the trained rule as in-distribution (ID) performance, while 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 of the experimental setups can be found in Appendix C.

RL generalizes, SFT overfits. As illustrated in Figure 5, RL consistently improves OOD performance on all tasks, including both unimodal (LLM) and multimodal (VLM). Specifically, Figure 6 demonstrates that RL increase of **+1.7%** on `GP-L` (9.8% → 11.5%) and **+11.0%** on `V-IRL-L` (80.8% → 91.8%). Even with the additional challenge of visual recognition in the vision-language variants, RL maintains consistent performance improvements of **+3.0%** (11.2% → 14.2%) on `GP-VL` and **+9.3%** (35.7% → 45.0%) on `V-IRL-VL`, respectively. In contrast, SFT consistently exhibits performance degradation across all OOD evaluations on all tasks: **-5.9%** on `GP-L` (9.8% → 3.9%), **-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`.

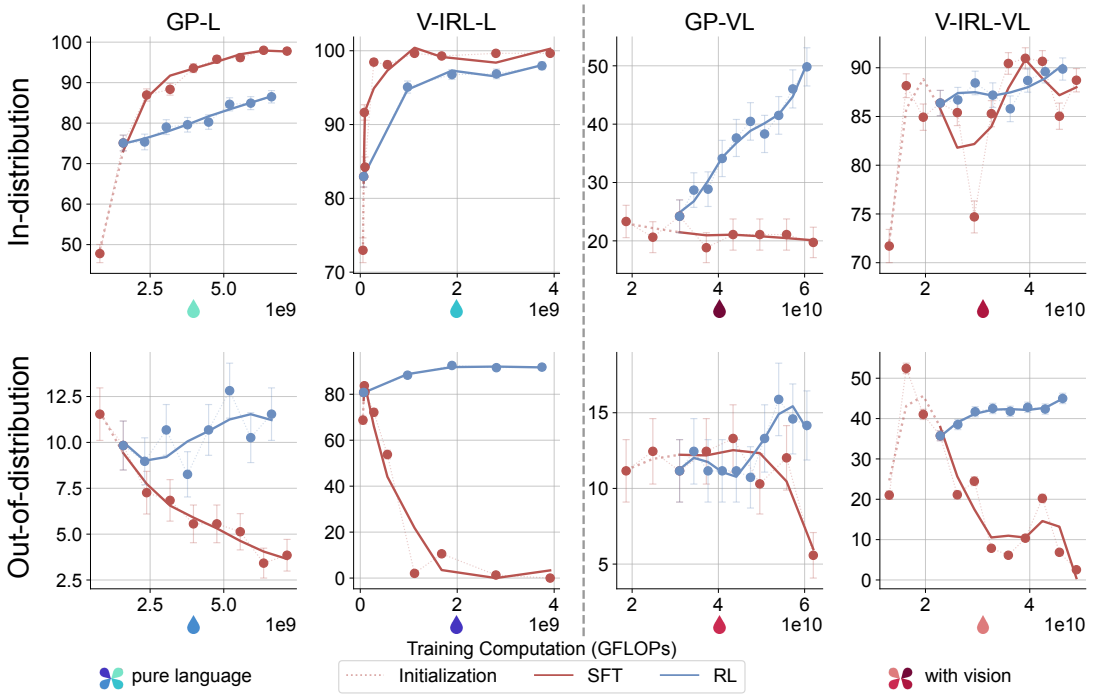


Figure 5: **Success rate (%) - GFLOPs trendlines for RL and SFT on `GeneralPoints` and `V-IRL`.** The top row shows in-distribution performance, while the bottom row shows out-of-distribution performance. Results are presented for both pure language (-L) and vision-language (-VL) variants of each task. For `GeneralPoints`, we report the episode success rate, while for `V-IRL`, we report per-step accuracy with overall success rate in Figures 1 and 16. Detailed evaluation setups are provided in Appendix C.1.

5.2. Generalization in Visual Out-of-Distribution Tasks

In Section 5.1, we demonstrate that RL generalizes in *language-only rule-based* tasks, whereas SFT exhibits the opposite trend. Given this observation, and that VLM include an additional visual modality, we move one-step forward beyond rule-based tasks, to study the generalization capability

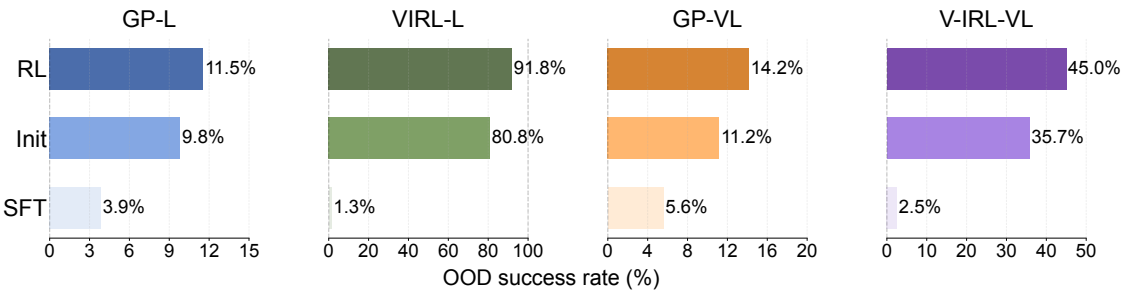


Figure 6: **Comparison of out-of-distribution performance under rule variants.** We report the success rate for `GeneralPoints` and **per-step-accuracy** for `V-IRL`. For each subplot, RL and SFT are trained with equal computation, and their shared initial checkpoint (marked as Init) is set as baseline. Detailed evaluation setups are provided in Appendix C.1.

in *visual OOD* scenarios. For the visual OOD task, we design *visual variants* of `GeneralPoints` and `V-IRL`. For `GeneralPoints`, we train the foundation model using black suits (♠, ♣) and test out-of-distribution performance on red suits (♥, ♦). For `V-IRL`, we train the model on New York City routes and evaluate it on original `V-IRL` VLN mini benchmark [1] containing routes from *various cities worldwide*. Note that the rules remain consistent across all comparative experiments.

RL generalizes in visual OOD tasks. As shown in Figure 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 improvement of **+17.6%** (23.6% → 41.2%), **+61.1%** (16.7% → 77.8%), whereas SFT suffers from performance decrease of **-9.9%** (23.6% → 13.7%) and **-5.6%** (16.7% → 11.1%). As a byproduct of our visual OOD study, we also show that our multi-turn RL formulation *improves state-of-the-art results* on the `V-IRL` mini benchmark [1] by **+33.8%** (44.0% → 77.8%).[3]

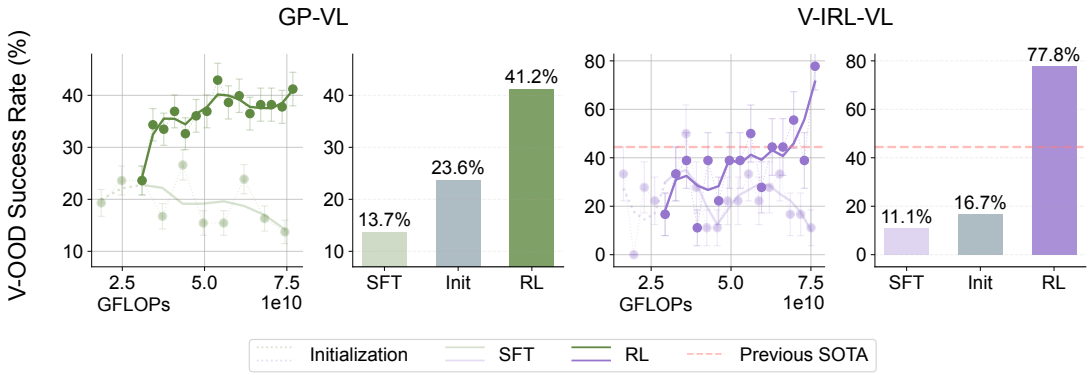


Figure 7: **Comparison of out-of-distribution performance under visual variants.** Similar to Figures 5 and 6, we present both the performance dynamics (shown as lines) and final performance (shown as bars) for visual out-of-distribution evaluations. We additionally mark the previous state-of-the-art on `V-IRL` VLN mini benchmark [1] (right).

5.3. RL Improves Visual Capabilities

Continuing our studies in VLMs, we have shown that VLMs trained with RL generalize to rule-based OOD tasks (Section 5.1) and visual OOD tasks (Section 5.2). This leads us to question: *how does RL enhance VLMs' visual capabilities*? To study this question, we conduction additional ablation studies on the OOD performance of RL and SFT in `GP-VL` along with the model's visual recognition accuracy, in terms of the recognizing all 4 cards from the input image. In particular, we study how

[3] As shown in Table 5 of Yang et al. [1], the previous state-of-the-art performance of `V-IRL` VLN mini benchmark is 44.0%, while we achieve an result of 77.8%.

scaling the post-training compute via RL or SFT affect the generalization and visual recognition accuracy in the language-only rule-based OOD (Figure 8 left) and visual OOD (Figure 8 right).

Scaling RL Improves Visual Recognition Accuracy in VLM Training. As shown in Figure Figure 8, we observe that the VLM's visual recognition accuracy largely affects the overall performance, as similarly observed in Zhong et al. [73]. In addition, scaling up RL compute also improve the visual recognition accuracy, as a byproduct of its generalization capability, while scaling SFT deteriorates both the visual recognition accuracy and overall performance. Additional experimental results are provided in Figures 14 and 15 of Appendix D.1.

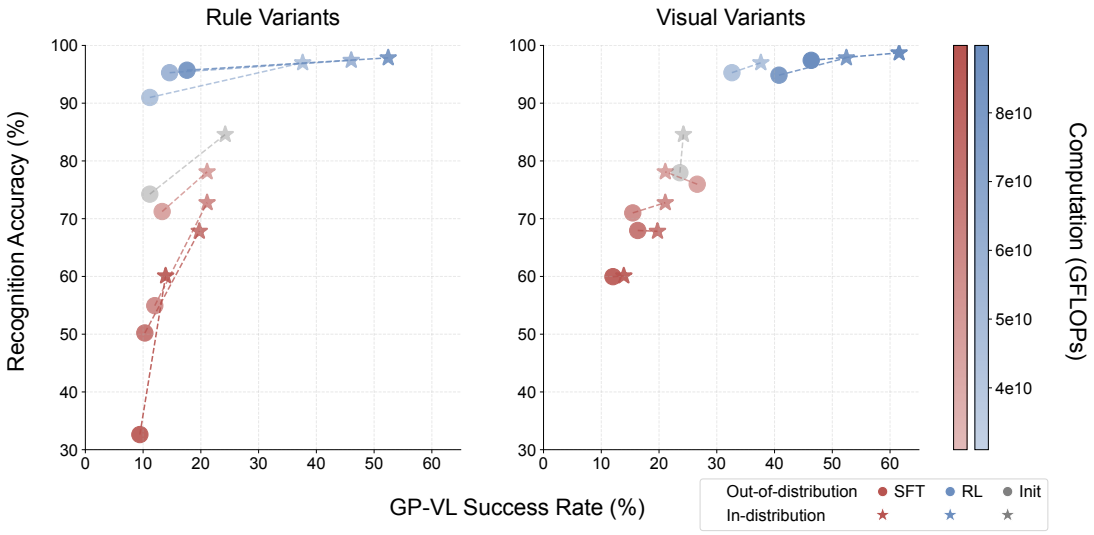


Figure 8: **Recognition vs. success rate for RL and SFT under different variants.** The results show that increasing SFT compute leads to degraded performance in both reasoning and recognition capabilities, while scaling up RL compute yields improvements in both areas. Connected datapoints are evaluated using the same checkpoint (same compute).

5.4. Role of SFT

Despite the superiority of RL in generalizing reasoning and visual capabilities, SFT remains essential as necessary initialization stage. Without SFT, direct RL training fails, as demonstrated in Figure 9 where all three experiments crash due to poor instruction following. Detailed example outputs in Figure 17 (in Appendix D.3) reveal that the original Llama-3.2-Vision-11B tends to generate long, tangential, and unstructured responses. This issue makes it extremely hard to retrieve task-related information and properly reward responses during RL training.

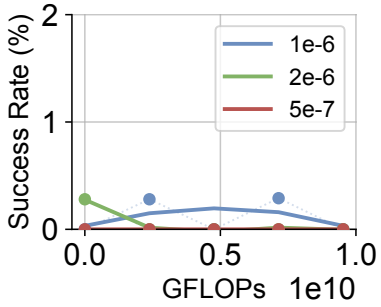


Figure 9: **RL experiments on `GP-L` without SFT initialization.** All of our 3 trials with different learning rate fail due to poor zero-shot instruction following capability of the base model.

References

[1] 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*, 2024. 1, 2, 3, 4, 7, 16

[2] 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*, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR. 1, 2

[3] 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. In *NeurIPS*, 2022. 1, 2, 5

[4] DeepMind Google. Introducing gemini: our largest and most capable ai model, 2023. URL https://blog.google/technology/ai/google-gemini-ai/. 1

[5] OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023. 2

[6] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 1

[7] 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*, 2024. 1, 3

[8] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024. 1

[9] OpenAI. Gpt-4, 2023. URL https://openai.com/research/gpt-4. 1

[10] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. 2023. 1

[11] 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. 1, 2

[12] 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. 1, 2

[13] 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. 1

[14] 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*, 2024. URL https://openreview.net/forum?id=nBjmMF2IZU. 1, 2, 3, 4, 5

[15] 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. 1, 2, 3

[16] Google DeepMind. Gemini2.0, 2024. URL https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/. 1

[17] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. *arXiv preprint arXiv:2404.05405*, 2024. 1, 2

[18] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 1

[19] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

[20] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

[21] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. *arXiv preprint arXiv:2403.04132*, 2024.

[22] 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. 1

[23] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. *arXiv preprint arXiv:2309.14316*, 2023. 1, 2

[24] Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.2, knowledge manipulation. *arXiv preprint arXiv:2309.14402*, 2023. 1, 2

[25] 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. 2, 3

[26] 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. 2

[27] 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. 17

[28] 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*, 2023. 2

[29] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. 2

[30] 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.

[31] 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.

[32] 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.

[33] 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, 2024. 2

[34] 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. 2, 3

[35] 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*, 2024. 2, 3

[36] 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. 2

[37] 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. 2

[38] 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. 2

[39] 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*, 2023. URL https://openreview.net/forum?id=HFQFAyNucq.

[40] 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. 2

[41] 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. 2

[42] 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, 2023. 2

[43] 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. 2

[44] 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. 2

[45] 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. 2

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

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

[48] 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*, 2024. 2

[49] 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, 2022. 2

[50] 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*, 2023. 2

[51] 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. 2, 3, 17

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

[53] 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.

[54] 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.

[55] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*, 2024. 3

[56] 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. 3

[57] 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*, 2024.

[58] 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*, 2024. 3

[59] 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*, 2024. 3

[60] 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, 2024. 3

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

[62] Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *CVPR*, 2024. 3

[63] 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. 3

[64] 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. 3

[65] 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. 3

[66] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, 2024. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/. 3

[67] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023. 3

[68] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 3

[69] Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32, 2019. 3

[70] 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. 3

[71] 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. 3

[72] 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. 5

[73] 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. 8

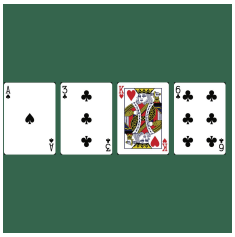
[74] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 17

A. Details on the General Points Environment

A.1. Detailed examples on the transition dynamics

As shown in Figure 10 and Figure 11, we treat the system prompt as v_0^{in} and then subsequently appending the future outputs $v_{1:t}^{\text{out}}$ and verifier info $v_{1:t}^{\text{ver}}$ into the prompt for getting the $t+1$ output. Figure 10 provides an example with the visual inputs, while Figure 11 shows the language only case.

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. $\quad \triangleright\ v_{t+1}^{\text{in}} = \text{concat}(v_t^{\text{in}}, v_t^{\text{out}}, v_t^{\text{ver}})$

Figure 10: **An example of our prompt update** for constructing v_{t+1}^{in} using $v_t^{\text{in}}, v_t^{\text{out}}$ 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.

A.2. Additional details on the environmental design

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

- Target point: Any positive integer
- Face cards rule: Two options
  - '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 full set of 52 poker cards
  - Sample at least one card from 'J', 'Q', and 'K'
- Card color: Three options
  - Black suits only: ♣, ♠

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}

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

Model output (v_t^{out})

{
"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. ▷ $v_{t+1}^{\text{in}} = \text{concat}(v_t^{\text{in}}, v_t^{\text{out}}, v_t^{\text{ver}})$

Figure 11: **An example of our prompt update** for constructing v_{t+1}^{in} using $v_t^{\text{in}}, v_t^{\text{out}}$ 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.

  - Red suits only: ♥, ♦.
  - All suits: ♠, ♥, ♣, ♦.

For all experiments, we fix the target point at 24. In Figure 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 5.2), 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.

B. Details on the V-IRL Environment

B.1. Detailed examples on the transition dynamics

B.2. Additional details on the environmental design

Basics. The route serves as the fundamental navigation object in the `V-IRL` environment. As illustrated in Figure 4, each route corresponds to a real-world path with associated language instructions and visual signals. Using Figure 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 4, 12 and 13 are presented in different formats, they convey equivalent information, with Figure 4 using natural language.

Simplification and arguments. We simplify the original `V-IRL` design from Yang et al. [1] 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: "forward()", "turn_direction(x)" where x∈['north', 'northeast', 'east', 'southeast', 'south', 'southwest', 'west', 'northwest'], "stop()"
  - Relative direction: "forward()", "turn_direction(x)" where x∈['left', 'right', 'slightly left', 'slightly right'], "stop()"
- Maximum straight road length: any positive integer

The action space argument accommodates the rule variants described in Section 4. For experiments shown in Figure 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 5.2) 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. [1].

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

C. Experimental Setup

This section details the experimental setup used in Section 5.

C.1. Evaluation Metric

Per-step accuracy. We report the per-step accuracy for `V-IRL-VL` task in Figures 5 and 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 `GeneralPoints-L`, `GeneralPoints-VL`, `V-IRL-L` in Figures 5 and 6. In `GeneralPoints` task, success is defined as succeeding at least once during the inference time verification. In `V-IRL` task, a sample is recorded as success when the model takes correct action at each movable point on the route. For `V-IRL`, the overall success rate is a significantly more demanding metric than per-step accuracy. For example, a random policy achieving 10% per-step accuracy would yield approximately 1e-10 success rate on routes averaging 10 steps in length.

Computation estimation. We estimate the FLOPs for training X following the similar manner of [27, 51], 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 RL (PPO [74]) contains iterative 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.

D. Additional Experimental Results

In this section, we provide additional experimental results that are not covered in the main body.

D.1. Ablation Studies on `GeneralPoints-VL` RL and SFT

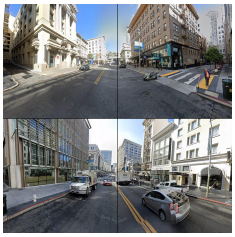
SFT. We ablate the hyperparameter choices under the same task setting of `GneralPoints-VL` in Section 5.1. For experiments fine-tuning all parameters, we search learning rates from $\{1 \times 10^{-4}, 1 \times 10^{-4}, 1 \times 10^{-5}, 1 \times 10^{-6}, 5 \times 10^{-7}, 1 \times 10^{-7}\}$. Freezing 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 14.

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 15. All parameters are tunable in our RL experiments.

D.2. More results on `V-IRL-VL`

D.3. Failure Cases

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}

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

Model output (v_t^{out})

{
"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 12: **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 $\boldsymbol{v}_t^{\text{ver}}$. The brown parts marks the task and related information, and the purple parts denote the state (s_t) specific info. 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}

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

Model output (v_t^{out})

{
"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 13: **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.

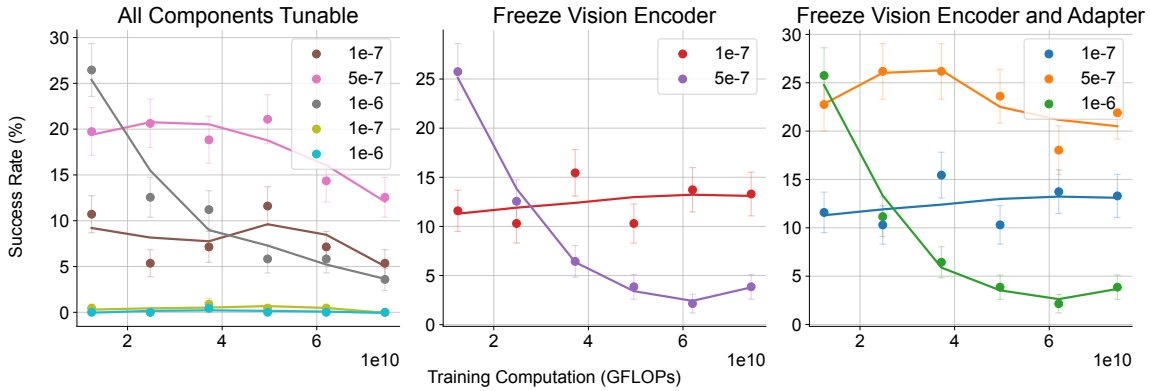


Figure 14: **Ablation studies on `GeneralPoints-VL` SFT.** We ablate the learning rate and report the episode success rate (%) of all experiments.

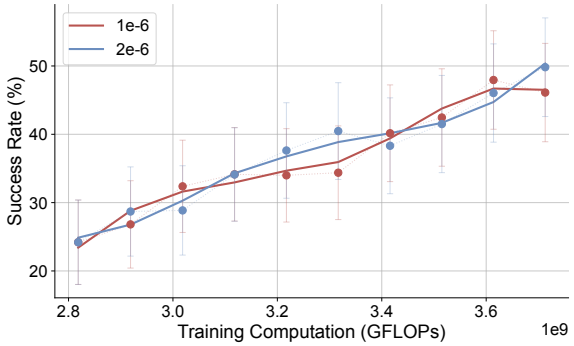


Figure 15: **Ablation studies on `GeneralPoints-VL` RL.** Echoing Figure 14, we ablate the learning rate and rreport the episode success rate (%) of the two experiments. All components are tunable here.

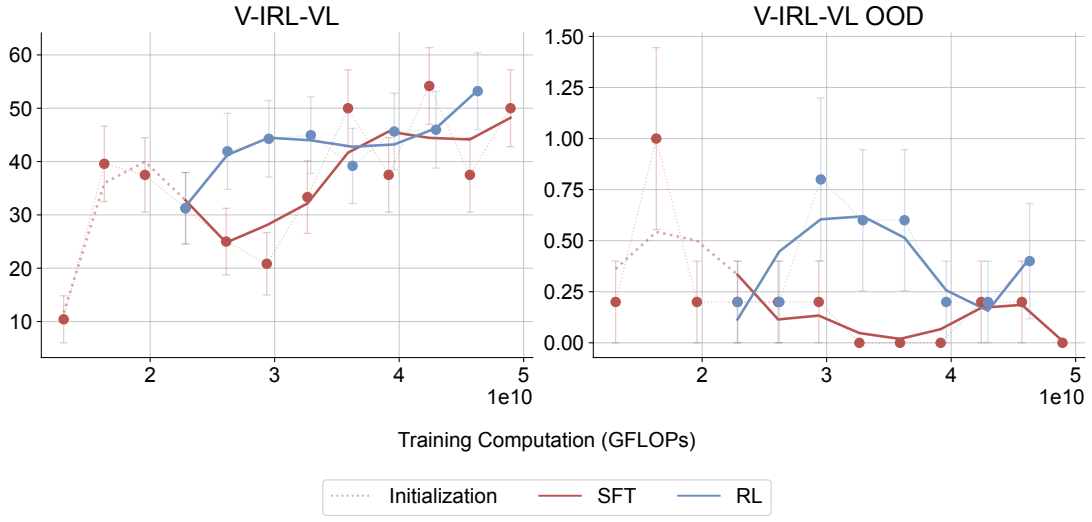


Figure 16: **Overall success rate (%) - GFLOPs for `V-IRL-VL` under rule variants.** Due to the task's complexity, neither training method achieves reasonable out-of-domain performance. Evaluation metric details can be found in Appendix C.1.

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}

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

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. $\quad \triangleright\ \boldsymbol{v}_{t+1}^{\text{in}} = \text{concat}(\boldsymbol{v}_t^{\text{in}}, \boldsymbol{v}_t^{\text{out}}, \boldsymbol{v}_t^{\text{ver}})$

Figure 17: **Example model outputs without SFT initialization.** We record model responses using prompts similar to the one shown in Figure 10. 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.