Beyond Reasoning Gains: Mitigating General-Capability Forgetting in Large Reasoning Models

Hoang Phan[1,2,*] **Xianjun Yang**[1] **Yuanshun Yao**[1] **Jack Zhang**[1,3,*] **Shengjie Bi**[1]
Xiaocheng Tang[1] **Madian Khabsa**[1] **Lijuan Liu**[1] **Deren Lei**[1]

[1]Meta Superintelligence Labs [2]New York University [3]Johns Hopkins University

hvp2011@nyu.edu, deren@meta.com

Abstract

Reinforcement learning with verifiable rewards (RLVR) has delivered impressive gains in mathematical and multimodal reasoning and has become a standard post-training paradigm for contemporary language and vision-language models. However, the RLVR recipe introduces a significant risk of capability regression, where models forget foundational skills after prolonged training without employing regularization strategies. We empirically confirm this concern, observing that open-source reasoning models suffer performance degradation on core capabilities such as perception and faithfulness. While imposing regularization terms like KL divergence can help prevent deviation from the base model, these terms are calculated for the current task; thus, they do not guarantee broader knowledge. Meanwhile, commonly used experience replay across heterogeneous domains makes it nontrivial to decide how much training focus each objective should receive. To address this, we propose RECAP—a replay strategy with dynamic objective reweighting for general knowledge preservation. Our reweighting mechanism adapts in an online manner using short-horizon signals of convergence and instability, shifting the post-training focus away from saturated objectives and toward underperforming or volatile ones. Our method is end-to-end and readily applicable to existing RLVR pipelines without training additional models or heavy tuning. Extensive experiments on benchmarks using Qwen2.5-VL-3B and Qwen2.5-VL-7B demonstrate the effectiveness of our method, which not only preserves general capabilities but also improves reasoning by enabling more flexible trade-offs among in-task rewards.

1 Introduction

Large Language Models (LLMs) and Vision-Language Models (VLMs) have demonstrated remarkable general-purpose capabilities (Achiam et al., 2023; Yang et al., 2023), yet strengthening their proficiency in complex reasoning remains a key frontier of research. Reinforcement Learning with Verifiable Rewards (RLVR) (Shao et al., 2024), an extension of Reinforcement Learning from Human Feedback (RLHF) (Ziegler et al., 2019; Ouyang et al., 2022), has emerged as a powerful paradigm for this purpose. By providing explicit reward signals such as exact-match correctness, format adherence, and answer brevity, RLVR has been applied to instruction following, STEM problem solving, code generation and logical reasoning models (Lightman et al., 2023b; Peng et al., 2025), resulting in large performance gains on benchmark scores, leading to headlines that language models can "learn to reason" (Guo and DeepSeek-AI, 2025).

Despite strong headline gains, RLVR exhibits recurring failure modes, prompting questions about whether current pipelines genuinely expand reasoning abilities (Shojaee et al., 2025). For example, exploration and diversity collapse occur when on-policy finetuning overly narrows the policy distribution—raising Pass@1 but reducing Pass@k and solution-path diversity (Yue et al., 2025; Dang et al., 2025). Likewise, outcome-only rewards introduce sparse credit assignment and instability, and not every task is naturally cast as a reinforcement-learning problem (e.g., translation, summarization, or captioning). In addition, strict answer formats and format-sensitive graders may conflate genuine reasoning improvements with mere format compliance, even introducing evaluation artifacts (Petrov et al., 2025). Recent studies report that many RL-trained models even underperform the base model in standardized evaluation, where formatting-reward-only baseline degrades the original performance more severely (Prabhudesai et al., 2025). The format reward model might be under-optimized or optimizing it can cause forgetting of math capabilities (Chandak et al., 2025).

[*]Work done while at Meta.

Findings of the Association for Computational Linguistics: ACL 2026, pages 34368–34398
July 2-7, 2026

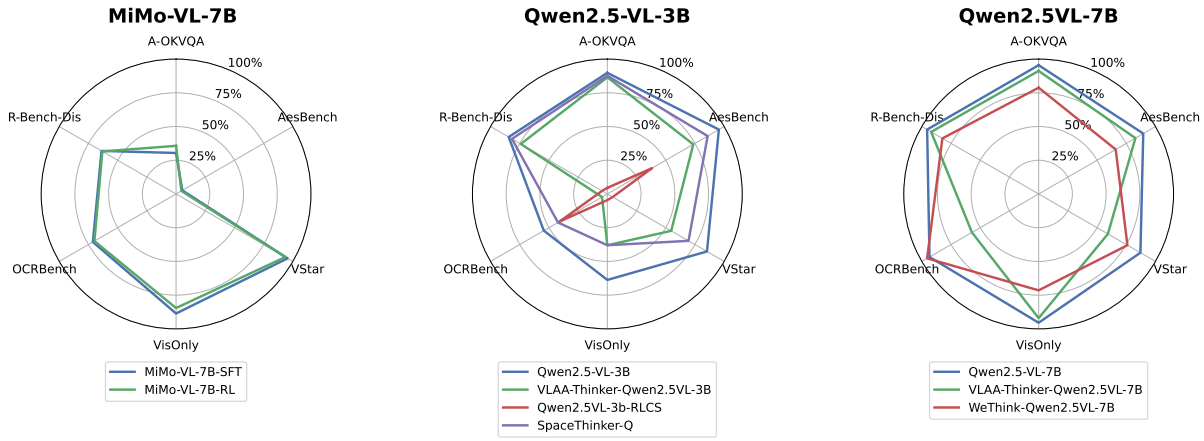


Figure 1: **General capabilities comparison of base VLMs (blue) and their reasoning-tuned variants (green/purple/red) on six representative, non-reasoning benchmarks (higher is better)**: A-OKVQA (knowledge-based VQA), AesBench (Huang et al., 2024) (image aesthetics), VStar (Wu and Xie, 2023) (spatio-temporal reasoning), VisOnly (Kamoi et al., 2025) (vision-only recognition aggregate) , OCRBench (Liu et al., 2024b) (text recognition), and R-Bench-Dis (Li et al., 2025b) (distribution-shift robustness). Across both Qwen2.5-VL families, reasoning-finetuned models generally underperform their base models on perception and robustness tasks, whereas MiMo-VL-7B-RL remains close to its SFT baseline.

Another critical yet under-explored issue in RLVR is that optimizing for a narrow set of targeted, verifiable rewards can lead to regression in general capabilities acquired during pretraining. Although models become proficient in following formatting requirements and solving reasoning tasks, they simultaneously exhibit increased hallucinations (Jaech et al., 2024; Yao et al., 2025b) and are more vulnerable to jailbreak attacks (Lou et al., 2025; Yao et al., 2025a). These results suggest that reasoning-oriented post-training can improve reasoning but at the cost of trading off non-target competencies (e.g., perception, safety, factual grounding), especially when RL training is prolonged without explicit regularization (Liu et al., 2025a).

To examine the forgetting issue of reasoning-focused finetuning, we begin by probing the general abilities of open-source reasoning models beyond their target reasoning tasks. As shown in Fig. 1, models finetuned for chain-of-thought or RL reasoning frequently lag behind their base counterparts on perception and robustness. For example, we observe a consistent drop on VisOnlyQA across both Qwen2.5-VL families, while MiMo-VL-7B-RL performs competitively against its base model on those non-reasoning tasks. We hypothesize this is due to its special finetuning strategy, where they employ mixed on-policy reinforcement learning that tries to improve the model's capabilities across multiple axes beyond math and reasoning, according to the MiMo-VL-7B technical report (Team et al., 2025). However, the detailed framework and the sampling or reweighting strategy is not disclosed. These patterns support our central claim: optimizing for reasoning rewards can erode non-reasoning capabilities, motivating a continual learning method to preserve general skills during reasoning-oriented post-training.

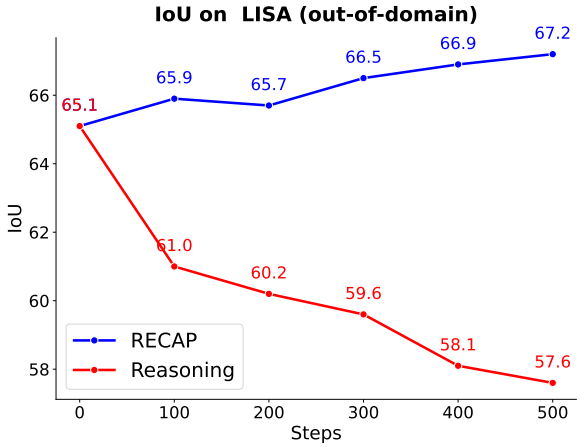


Figure 2: **Performance comparison between fine-tuning solely on math and fine-tuning with RECAP**. Our model not only preserves the base model performance but also improves it (↑ 2%) while the reasoning model quickly falls behind the base model after 100 iterations.

Our initial experiments with Qwen2.5-VL-7B show that training solely on reasoning rewards degrades performance on general capabilities, for ex-

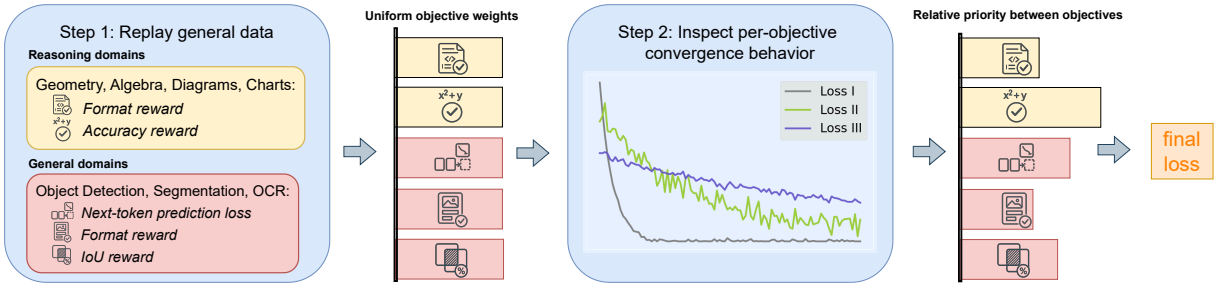


Figure 3: **Overview of our RECAP.** Along with the target reasoning task, we sample data from general domains in order to maintain that knowledge during finetuning. Initially, the objectives of interest are initially weighted uniformly to optimize the main model. After a few iterations, we record the convergence behavior of individual objectives. Based on this, we adjust the focus to prevent the dominance of any objectives and assign less weight to saturated ones.

ample, by 7% on the perception task, as shown in Figure 2. To address this, we propose gathering general-capability data and integrating it into RLVR via an online weighting mechanism. However, due to cross-domain heterogeneity, it is nontrivial to decide how much weight to assign to each loss term or reward. We then measure how different reward signals evolve during RLVR training in Section 4 and find that some rewards converge up to three times faster than others, which should therefore not be emphasized later in training when the model has already learned the corresponding skills

Motivated by the above observations, we propose ***R**eplay-**E**nhanced **CA**pability **P**reservation* (**RECAP**)—a principled replay-based training strategy that mixes general data back into the RL objective, then dynamically reweights objectives based on their convergence rate and instability. As shown in Figure 3, RECAP computes the relative priorities of the objectives of interest by inspecting their convergence behaviors to reweight them in the final loss. Experiments demonstrate that our method not only preserves general capabilities but also improves reasoning performance by allowing flexible trade-offs among reward types. In summary, our contributions are summarized as follows:

- We systematically re-evaluate open-source reasoning models and show that reasoning-focused finetuning consistently regresses general capabilities. This motivates replaying general-capability data during RLVR to preserve pretraining knowledge. We further show that objectives exhibit distinct convergence behaviors, making commonly used, manually tuned reweighting schemes suboptimal in such scenarios.
- We introduce RECAP, a plug-in scheduler that replays general-capability data during RLVR and dynamically reweights both general and main task objectives. Our proposed method naturally down-weights saturated format signals and refocuses capacity on harder, high-variance objectives. The method is end-to-end, magnitude-agnostic, requires no auxiliary models, and can be integrated into RLVR pipelines with negligible overhead.
- In our experiments, RECAP preserves or improves general capabilities while matching or exceeding the reasoning performance of reasoning-only finetuning. It consistently outperforms strong continual-finetuning baselines and is competitive with specialized open models while utilizing less compute. Empirically, we also find that replaying general data yields shorter, more concise rationales while not compromising reasoning ability.

2 Related work

Foundation models and post-training. Large transformer models pretrained on broad corpora serve as general-purpose backbones with strong abilities and wide transfer across domains (Brown et al., 2020; Touvron et al., 2023). Post-training adapts these backbones to downstream tasks via (i) supervised finetuning, from early ULMFiT (Howard and Ruder, 2018) to instruction tuning in FLAN (Wei et al., 2021) or Flan-T5 (Chung et al., 2024); (ii) reinforcement learning from human or AI feedback, typically combining preference modeling with policy optimization; and (iii) direct preference optimization objectives that bypass explicit reward models. For reasoning, reinforcement learn-

ing with verifiable rewards has become a common recipe: verifiers or rule-based checkers score final solutions in math and related domains, often within PPO-style pipelines (Guo and DeepSeek-AI, 2025; Liu et al., 2025b). Process-based neural reward models provide supervision for intermediate progress (Setlur et al., 2025) rather than only the final output. However, PRMs can induce reward hacking: agents learn to exploit the appearance of a correct process rather than achieving the intended outcome (Wang et al., 2025b; Shao et al., 2024).

Catastrophic forgetting in continual and post-training. Catastrophic forgetting describes performance regressions on previously acquired skills when adapting to new data (McCloskey and Cohen, 1989; French, 1999). Early work in this vein introduced regularization-based mitigations such as EWC (Kirkpatrick et al., 2017), SI (Zenke et al., 2017), and MAS (Aljundi et al., 2018) that prevent excessive change on important parameters. Functional approaches like LwF (Li and Hoiem, 2016) constrain outputs via distillation (Hinton et al., 2015), and replay via small episodic memories (Rebuffi et al., 2017; Lopez-Paz and Ranzato, 2017; Rolnick et al., 2019; Buzzega et al., 2020) are consistently strong baselines across settings. In RLHF post-training, a KL penalty toward the reference policy is commonly used to stabilize updates and curb over-optimization (Ouyang et al., 2022). Along with the standard KL-regularization approaches, InstructGPT (Ouyang et al., 2022) interleaves the pretraining gradients with RLHF updates to reduce drift relative to KL-only regularization (Zheng et al., 2023). Concurrent works (Zhang et al., 2025; Fu et al., 2025) integrate verified rollouts to stabilize learning or penalize the discrepancy on augmented training data (Wang et al., 2025d). Yet, those methods do not guarantee preservation of performance on non-target domains. Other approaches tackle forgetting by incorporating mixed, verifiable reward suites (Team et al., 2025) or introducing reflection or re-attention mechanisms under RL objectives (Chu et al., 2025). In addition, recent reasoning-focused RL pipelines often reduce or remove KL to encourage exploration (Hu et al., 2025a; Hao et al., 2025), potentially exacerbating forgetting.

3 Background

This section first provides the essential background on standard post-training for large language models, covering (i) supervised finetuning (SFT), (ii) RL-based alignment from preferences, (iii) reinforcement learning with verifiable rewards, and (iv) GRPO for long-form reasoning. Our approach builds on recent progress in reasoning-centric LLMs—exemplified by DeepSeek-R1 (Guo and DeepSeek-AI, 2025) and other contemporary models (Ji et al., 2025; Yu et al., 2025; Chen et al., 2025a; Wang et al., 2025a). We adopt GRPO as our primary RL algorithm because it effectively reduces memory and compute overhead relative to standard PPO.

Supervised finetuning (SFT). Let an LLM with parameters θ induce a conditional policy $\pi_\theta(\cdot \mid x)$ over responses y to a prompt x. SFT optimizes the negative log-likelihood on instruction–response pairs $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = -\sum_{i=1}^{N} \log \pi_\theta\big(y^{(i)} \mid x^{(i)}\big).$$

SFT has been central to transferring general-purpose LMs to downstream instruction following and broad zero-shot generalization; it typically provides the initialization for subsequent preference- or reward-based alignment.

RL-based post-training. Reinforcement learning from human feedback (RLHF) fits a reward model $r_\phi(x, y)$ from pairwise human preferences (Ziegler et al., 2019; Rafailov et al., 2023; Lambert, 2025), commonly using a Bradley–Terry likelihood (Bradley and Terry, 1952), and then maximizes reward while regularizing toward the pretrained reference π_{ref} (often with a KL penalty) via policy optimization such as PPO (Schulman et al., 2017):

$$\max_\theta \mathbb{E}_{x\sim\mu,\, y\sim\pi_\theta(\cdot|x)}\Big[r_\phi(x, y)\Big] - \beta\, \mathbb{E}_{x\sim\mu}\Big[D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x)\big)\Big].$$

This pipeline improves helpfulness/harmlessness while retaining base-model competence; see early LM-preference work and InstructGPT (Ouyang et al., 2022) for canonical formulations, and PPO for the underlying stable policy-gradient updates (Stiennon et al., 2020). In settings with programmatic or automatic verifiers, reinforcement learning with verifiable rewards (RLVR) replaces learned human-preference rewards with verifiable signals $r(x, y) \in [0, 1]$.

Group Relative Policy Optimization. GRPO is a PPO-style algorithm tailored for LLM reasoning

that forgoes a learned value/critic and instead computes advantages from group-normalized sequence rewards. For each prompt x, sample a group of G rollouts $O = \{o_i\}_{i=1}^{G}$ from a frozen rollout policy $\pi_{\theta_{\text{old}}}$. Let R_i be the verifiable sequence-level reward and define the group-normalized advantage $\widehat{A}_i = \big(R_i - \text{mean}(R)\big)/\text{std}(R)$. With token-wise importance ratio $r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|x,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|x,o_{i,<t})}$, GRPO maximizes the clipped surrogate plus KL regularization:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\Big\{r_{i,t}(\theta)\widehat{A}_i,\ \text{clip}\big(r_{i,t}(\theta),1-\epsilon,1+\epsilon\big)\widehat{A}_i\Big\} - \beta\, D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big)\Bigg].$$

By normalizing across a group of responses per prompt, GRPO stabilizes updates without a critic, which is preferable for long chain-of-thought outputs with sparse, verifier-based rewards. Empirically, GRPO boosts mathematical-reasoning performance in many open models (Shao et al., 2024).

General ability degradation. Unlike traditional continual learning studies, modern post-training pipelines must jointly consider gains in reasoning and retention of inherent general abilities (e.g., perception, grounding, instruction following, safety). Let $\mathcal{G} = \{G_1, \ldots, G_M\}$ denote a suite of general-ability tasks and $\mathcal{R} = \{R_1, \ldots, R_L\}$ a suite of reasoning datasets.

4 Our Method: Replay-Enhanced CApability Preservation (RECAP)

Setting. In the context of supervised learning, let $\mathcal{D} = \{\mathcal{D}_n\}_{n=1}^{N}$ be N domains and $\ell_{n,k}^{(t)}(\theta)$ the mini-batch loss of objective $k \in \{1, \ldots, K\}$ on domain n at iteration t for parameters θ. Note that $K \geq N$ because there are some tasks using more than one reward or objective. The model parameters θ are thus optimized by minimizing the average loss across objectives:

$$L_k^{(t)} \;=\; \frac{1}{N}\sum_{n=1}^{N}\ell_{n,k}^{(t)}(\theta).$$

Our framework acts on $\{L_k^{(t)}\}_{k=1}^{K}$ regardless of whether each L_k arises from an RL reward surrogate or a supervised learning loss term.

Per-objective rate and the stability of convergence. Due to the unstable nature of RL training, we cannot rely solely on the per-step objective or reward value to compute the reweighting coefficients. Instead, for each objective k, we measure the *convergence rate* over a sliding window of length $2 \times W$ by computing the current window average and the previous window average by:

$$\underbrace{\mu_k^{(t)} = \frac{1}{W}\sum_{s=t-W+1}^{t} L_k^{(s)}}_{\text{estimated current loss value}},\quad \underbrace{\tilde{\mu}_k^{(t)} = \frac{1}{W}\sum_{s=t-2W+1}^{t-W} L_k^{(s)}}_{\text{estimated old loss value}}.$$

and the *instability* (coefficient of variation) in the same window:

$$\sigma_k^{(t)} = \sqrt{\frac{1}{2W-1}\sum_{s=t-2W+1}^{t}\big(L_k^{(s)} - \mu_k^{(t)}\big)^2}.$$

Based on these measures, we form two signals (i) the $\boxed{\textit{convergence rate } c_k^{(t)} = \tilde{\mu}_k^{(t)}/\mu_k^{(t)}}$ captures how fast the loss is improved, while (ii) the $\boxed{\textit{inverse signal-to-noise ratio } i_k^{(t)} = \sigma_k^{(t)}/(\mu_k^{(t)} + \tilde{\mu}_k^{(t)})}$ captures loss instability. Intuitively, $c_k^{(t)} > 1$ indicates recent improvement (loss dropping relative to the previous window), while $c_k^{(t)} \approx 1$ signals saturation. The term $i_k^{(t)}$ is larger when the objective is noisy/unstable.

Relative priority between objectives. We convert these signals into normalized coefficients via a temperature-controlled softmax. With the temperature $T > 0$, we define the priority of k-th objective as $s_k^{(t)}$ and compute the coefficients λ for reweighting objectives.

$$s_k^{(t)} = c_k^{(t)} + i_k^{(t)}, \lambda_k^{(t)} = \frac{K\,\exp\big(s_k^{(t)}/T\big)}{\sum_{i=1}^{K}\exp\big(s_i^{(t)}/T\big)}. \quad (1)$$

The prefactor K preserves average scale so that $\frac{1}{K}\sum_k \lambda_k^{(t)} = 1$. Lower T sharpens priorities while higher T approaches uniform mixing. We set $T = 5$ by default in our experiments.

Overall training objective. At step t, we minimize the following weighted objective:

$$\mathcal{L}^{(t)}(\theta) \;=\; \frac{1}{K}\sum_{k=1}^{K}\lambda_k^{(t)}\, L_k^{(t)}.$$

Optimizing θ with $\nabla_\theta \mathcal{L}^{(t)}$ steers learning towards objectives that are *both* slow to converge (high c_k) *and* fluctuating (high i_k), while leaving well-learned, stable objectives with lower weight. The scheme reduces to the standard equal weighting as

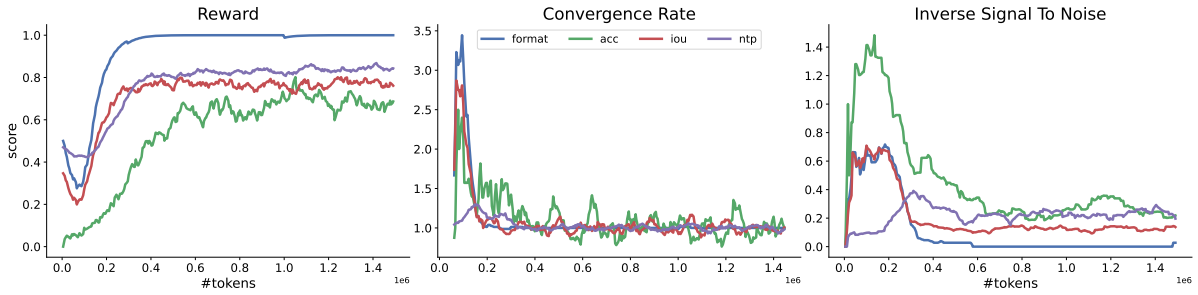


Figure 4: **Different rewards exhibit different convergence behaviors.** While the format reward is easy to optimize and obtains the highest rate of convergence, it quickly saturates and thus yields low convergence rate ($c \sim 1$) and instability ($i \sim 0$) after 50 steps. By contrast, the reasoning accuracy fluctuates the most, thereby steering the optimization toward the corresponding objective. IoU and ntp indicate the IoU reward and the next-token-prediction accuracy during training. The result is obtained on the first setting in our experiment section below.

$T \to \infty$ (i.e., uniformly sample from all domains without loss reweighting).

Figure 4 illustrates the insight behind our proposed method, where we finetune the model on a reasoning dataset (tracked by accuracy) while replaying perception data (IoU) and an SFT dataset for generality. Earlier in training, the format signal is easy to optimize and saturates quickly, so its c falls toward 1 and i toward 0, reducing its priority. After 100 steps, many signals plateau (convergence rate $c \approx 1$) while they still differ in the level of stability. Among them, the reasoning reward remains fluctuates the most (~ 0.3, yielding higher i and thus higher λ. At this point, the model has learned to answer according to the predefined template; thus the corresponding signal-to-noise ratio for formatting reward is ≈ 0. This motivates combining *both* progress (c) and instability (i) in Equation 1 as they complement each other. Tuning the trade-off between these two terms offers finer-grained control and potentially improves the performance. However, we simply take their unweighted sum $s = c + i$ for simplicity, which performs consistently well in our experiments.

Entropy-regularized interpretation of RECAP reweighting. We can show that the RECAP reweighting rule arises as the closed-form solution of an entropy-regularized priority allocation problem. At iteration t, let:

$$s^{(t)} = \left(s_1^{(t)}, \ldots, s_K^{(t)}\right) \in \mathbb{R}^K$$

denote the per-objective priority scores. In RECAP, we define $s_k^{(t)} = c_k^{(t)} + i_k^{(t)}$, where $c_k^{(t)}$ measures the recent convergence behavior of objective k, and $i_k^{(t)}$ captures its instability.

Rather than manually assigning weights to those objectives, we interpret the weights as an allocation vector:

$$p \in \Delta^K := \left\{ p \in \mathbb{R}^K_{\geq 0} : \sum_{k=1}^{K} p_k = 1 \right\}.$$

Given a temperature parameter $T > 0$, the entropy-regularized priority allocation objective has the following formula:

$$\Phi_t(p) = \langle p, s^{(t)} \rangle + T\mathcal{H}(p), \ \mathcal{H}(p) = -\sum_{k=1}^{K} p_k \log p_k,$$

We consider the entropy-regularized priority allocation problem:

$$\max_{p \in \Delta^K} \Phi_t(p).$$

The linear term assigns higher mass to objectives with larger RECAP priority scores, while the entropy term prevents degenerate allocation to a single objective and encourages smoother mixtures. The temperature T controls this trade-off: smaller T yields sharper allocations, whereas larger T approaches uniform weighting.

The following result characterizes the solution.

Theorem 1 (Entropy-regularized priority allocation)**.** *For any $T > 0$ and any score vector $s^{(t)} \in \mathbb{R}^K$, the optimization problem in Eq. (2) has a unique maximizer $p^{(t),\star} \in \Delta^K$, given by*

$$p_k^{(t),\star} = \frac{\exp\left(s_k^{(t)}/T\right)}{\sum_{j=1}^{K} \exp\left(s_j^{(t)}/T\right)}, \qquad k = 1, \ldots, K.$$

The proof is deferred to Appendix A.1. This theorem shows that the softmax form used in RECAP is not merely a heuristic normalization, it is an entropy-regularized allocation problem over objectives.

Table 1: **Benchmark results in large hybrid setting.** We report accuracy scores (higher is better) on nine perception and reasoning benchmarks. Rows above the break are open-source reasoning models with different backbones; the lower block compares variants finetuned from the same Qwen2.5-VL-7B base model. Bold = best; underline = second best within the Qwen2.5-VL-7B family.

Model	LISA	MMMU-PRO	AI2D	MathVista	MathVision	MathVerse	MMBench	VizWiz	OCRBenchv2
Open-source reasoning baselines									
VLAA-Thinker-7B	63.14	26.30	75.45	63.90	11.18	29.87	75.95	47.57	40.23
Vision-R1-7B	47.30	26.76	0.00	61.80	18.75	23.32	69.46	53.12	24.63
OpenVLThinker-7B	42.73	21.79	59.94	59.10	5.59	19.26	71.53	52.89	28.30
Qwen2.5-VL-7B and our variants									
Base model	65.13	25.55	67.62	61.70	9.54	26.29	71.82	50.82	39.49
LwF	65.08	29.59	73.93	63.90	18.42	33.98	73.11	53.12	39.56
PropMix	66.80	31.39	75.32	63.40	21.05	34.75	73.54	57.05	37.60
Uniform	65.18	31.91	76.43	65.60	22.13	36.07	75.34	54.05	38.06
Coreset	64.82	31.91	**79.92**	**66.90**	23.36	37.58	78.09	**63.76**	35.49
Reasoning-only	57.58	33.87	74.97	65.50	24.87	40.74	77.84	62.45	38.55
RECAP	**67.24**	**34.15**	78.21	66.70	**25.11**	**40.83**	**78.52**	61.97	**39.72**

5 Experiments

5.1 Experimental Settings

To evaluate our proposed approach, we conduct our experiments using two base models: Qwen2.5-VL-3B and Qwen2.5-VL-7B across two complementary setups at different scales: **RLVR-Only Setting:** This smaller setup follows the experimental configuration of Liang et al. (2025), focusing on domain-specific RLVR, which could serve as the *upper-bound* baseline of static data mixture approaches and as our closest baseline. The Qwen2.5-VL-3B model is trained until data from a particular domain is exhausted. Since Liang et al. (2025) enforces binary (0–1) rewards across all domains and thus cannot directly handle non-RL tasks, we further extend our evaluation to the **Hybrid Setting:** To bridge RL and supervised training paradigms, we finetune Qwen2.5-VL-7B under a larger mixed objective regime that combines RLVR with SFT-style training. Specifically, we use ThinkLite-VL-70k (Wang et al., 2025c) while jointly replaying perception-oriented datasets such as RefCOCO (Kazemzadeh et al., 2014) and LLaVA-OneVision OCR (Li et al., 2024).

To isolate the effect of replay and dynamic reweighting and also for the ease of implementation, we uniformly sample across data sources by default and reweight only the objectives of interest. Unless otherwise noted, we disable the reference-KL penalty to disentangle the effectiveness of regularization approaches (Li and Hoiem, 2017) and our replay mechanism. We also include a comparison with this regularization-based approach in our list of established baselines for continual learning below:

- **Reasoning only:** We train solely on the target reasoning task with fixed reward weights (no replay). This is the most straightforward approach in continual learning.
- **PropMix:** General data is included during finetuning, with data across domains sampled in proportion to their source size. Losses are not reweighted.
- **Uniform:** Data are sampled uniformly across domains. Losses are not reweighted.
- **Coreset:** Replay a size-limited subset of general data (half the reasoning-data volume in our setup) to align with standard coreset-style replay methods (Rebuffi et al., 2017; Chaudhry et al., 2019).
- **LwF:** Data are sampled uniformly, and we set the KL regularization coefficient β to 0.01. We refer to this method as LwF, as it shares a similar approach as Learning without Forgetting (Li and Hoiem, 2017).

For context, we also include some representative open-source vision language models specializing in reasoning derived from the corresponding base models in each experiment. Note that we list them here for easier benchmarking and we are not aiming to outperform them, as those models often undergo many complex training pipelines. Models are evaluated with LMMS-Eval (Zhang et al., 2024a).

5.2 Experimental Results

Table 1 reports the performance of models finetuned from Qwen2.5-VL-7B across different benchmarks in the hybrid setting, showcasing a more

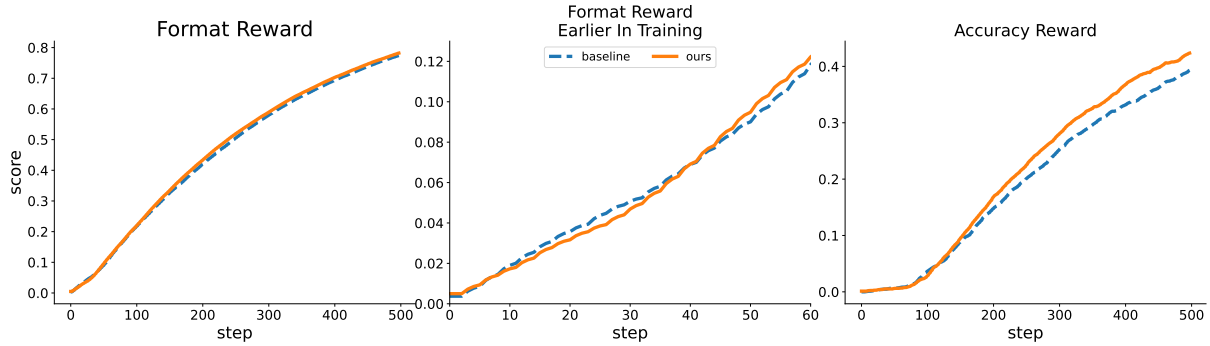


Figure 5: **Evolution of format and accuracy rewards on the reasoning domain during training:** Training curves for the format reward over full training (left), an early-training zoom (middle), and the accuracy reward (right). While the uniform baseline is better in maximizing the format reward, it falls behind our proposed method later in terms of accuracy, as we prioritize correct solutions over formatting once the model can follow the predefined template. Curves are smoothed with an exponential moving average for readability.

general scenario than RLVR-only training. Overall, RECAP achieves the best or runner-up performance across different datasets, except for VizWiz, where we still improve the base model performance by more than 10%. Compared with the base model, naively finetuning on the reasoning domain causes significant forgetting, especially on tasks not requiring extensive thinking. For example, finetuning on ThinkLite-VL-70k reduces the segmentation ability from 65.13 to 57.58 on LISA. Meanwhile, replaying the old data helps preserve the performance across all baselines on this dataset (≈ 64). Compared with uniform sampling, LwF achieves similar scene understanding performance while obtaining lower scores on the reasoning benchmark (e.g., 29.59 vs 31.91 on MMMU-PRO). Similar behaviors are observed in (Wang et al., 2025d; Hu et al., 2025a), where they remove the KL term for more plasticity. Among all baselines, our method obtains the highest segmentation score, boosting the performance of the base model by more than 2%. This improvement highlights the impact of loss reweighting over the uniform baseline.

According to Table 2, RL training on the reasoning domain consistently enhances the base model on both reasoning and perception benchmarks. Especially on ScienceQA, RL lifts the performance of Qwen2.5-VL-3B from 6 to 60. On this benchmark, our proposed method even outperforms comparative open-source reasoning models. We consider MoDoMoDo as the *upper-bound* approach of static data mixture approaches due to (i) MoDoMoDo has access to the target-task performance during finetuning, which requires rerunning the experiments if new target tasks are introduced; and (ii) they train

Table 2: **Benchmark results in RLVR-only setting.** We report the accuracy score over six benchmarks, in which MoDoMoDo is trained to maximize performance. Please note that in this benchmark only, we use the rule-based evaluator on the MathVista dataset instead of "gpt-3.5-turbo" to align with Liang et al. (2025).

Model	SAT	ScienceQA	MathVista	ChartQA	InfoVQA	MMMU
Open-source reasoning baselines						
VLAA-Thinker-3B	49.38	14.63	30.4	45.84	30.81	32.22
MM-R1-MGT-PerceReason	50.83	34.21	33.4	44.88	61.42	40.22
Ocean_R1_3B_Instruct	59.49	68.72	38.7	54.00	38.02	40.89
Qwen2.5VL-3b-RLCS	24.12	21.32	17.2	3.32	10.86	27.11
vision-grpo-qwen-2.5-vl-3b	50.57	4.17	32.4	67.80	58.29	37.22
Qwen2.5-VL-3B-GRPO-deepmath	34.70	45.27	32.3	70.24	49.75	39.11
Qwen2.5-VL-3B and our variants						
Base model	43.98	6.20	23.6	43.88	32.02	38.67
Uniform	44.55	64.85	32.4	69.68	58.30	39.44
MoDoMoDo	49.95	65.74	32.2	**70.40**	59.88	39.11
RECAP	**55.19**	**71.59**	**33.2**	**70.40**	**60.78**	**42.44**

multiple proxy models of the same size as the baseline models to learn the test performance as a function of the mixing ratio, which is computationally expensive, especially in the context of reinforcement learning. Even after selecting an "optimal" mixture, the method still depends on hand-tuned reward weights (e.g., doubling accuracy and IoU relative to formatting rewards). Those trade-off coefficients are also set differently in prior work without clarification, which limits the generality.

5.3 Ablation Studies

We conduct an ablation study on the large hybrid setting by comparing the accuracy and format reward of our proposed method with the uniform baseline to isolate the effect of our reweighting. The uniform baseline employs identical hyperparameters, including data sampling and model training pipeline, yet only differs from RECAP in the loss reweighting mechanism ($\lambda_k = 1/K$). In Figure 5, we present the curves for formatting and accuracy reward during training. In the early phase,

the baseline climbs format faster—consistent with format being a low-variance, easy-to-optimize signal—yet a crossover soon appears and our method surpasses it around step 40 as training progresses. In contrast, for accuracy, our method opens a growing lead over time (right). The behavior aligns with our scheduler: once the format objective shows fast convergence and low instability, its weight is down-regulated and capacity is reallocated to harder, higher-variance objectives (e.g., accuracy), avoiding over-optimization of formatting while improving task correctness.

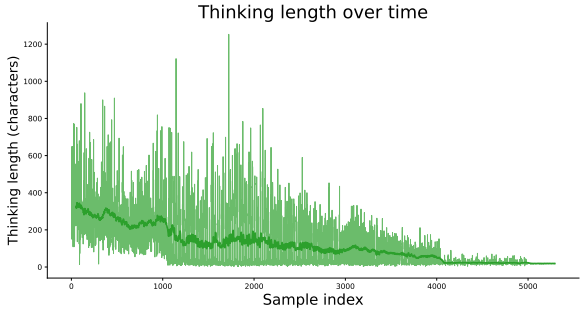


Figure 6: **Thinking length on a segmentation task during finetuning.** When a uniform "thinking reward" is applied to all domains, the model quickly learns that long chains of thought are unnecessary for segmentation. The average response length drops from several hundred characters at the start of training to tens (often near zero) later on.

We also empirically found that using the same format reward for different domains is suboptimal. We start by examining the approach from Liang et al. (2025) by employing the same thinking reward on every domain, including scene understanding tasks. In Figure 6, we plot the response length on the segmentation task during training and find that the Qwen2.5-VL model rapidly trims its chain-of-thought and answers the question directly later in training. This behavior suggests that explicit reasoning is unnecessary for such perception tasks and that encouraging long rationales can even be detrimental. We also include qualitative examples in the appendix to show how the model gradually suppresses its reasoning trace during training. Motivated by this observation, in our broader setting, we keep answer-format rewards for perception domains (no thinking) and reserve thinking rewards for tasks that truly benefit from step-by-step reasoning.

Figure 7 tracks the length of the generated thinking segment on the reasoning task throughout training. When trained only on reasoning data, the model maintains long chains of thought with high variability. In contrast, mixing general-capability replay with dynamic objective reweighting progressively reduces thinking length and stabilizes variance, converging to concise rationales (60% reduction, $67 \rightarrow 27$ words on average) while preserving accuracy. These shorter reasoning effort directly improves inference efficiency—fewer generated tokens reduce latency and compute cost—without sacrificing problem-solving quality.

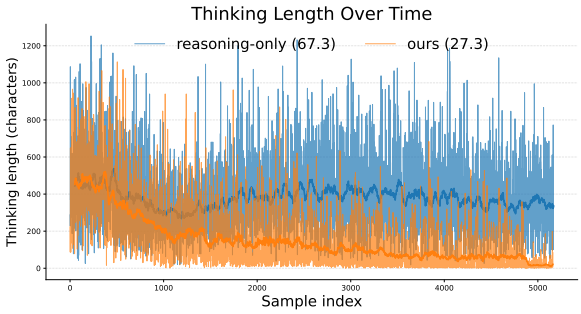


Figure 7: **Thinking length on a reasoning task during finetuning.** We compare a model trained reasoning-only (blue) against our replay + dynamic reweighting method (orange). Highlighted curves show the running average thinking length per example, where our method generates only $\sim$ 27.3 words per question, compared with baselines ($\sim$ 67.3).

6 Conclusion

In this paper, we investigate the forgetting issue in recent reasoning vision-language models and find that those models exhibit clear forgetting of general knowledge obtained during pretraining. Motivated by this, we propose a fix by replaying general data during finetuning and a plug-in method to reweight objectives without the additional cost of training external models. On reasoning benchmarks, our proposed method not only preserves the general knowledge but also improves the target reasoning performance by properly reweighting the rewards of those tasks.

Limitations

Our proposed RECAP framework is generic and extends naturally beyond RLVR and SFT to preference- and alignment-based objectives (Rafailov et al., 2023; Garg et al., 2025; Hong et al., 2024; Ethayarajh, 2024), and process reward models (Lightman et al., 2023a; Setlur et al., 2024). However, due to constraints on the train-

ing datasets available for this work, our empirical evaluation focuses on RLVR and standard SFT settings. We expect our method to yield similar gains over uniform or manually tuned baselines with any heterogeneous objective sets, but we leave a comprehensive evaluation across non-RL objectives to future work. In practice, applying our scheduler to non-RL losses does not require an expensive coefficient search or per-objective normalization due to its magnitude-agnostic nature.

Potential Risks

Our work mitigates general capability forgetting during RLVR-style post-training for multimodal reasoning models, which can improve overall utility but also introduces risks. In particular, reasoning-oriented post-training can still cause safety and reliability regressions that are not fully captured by the reward or evaluation suite, including degraded refusal behavior, factuality, or robustness to adversarial prompts (e.g., jailbreaks). To mitigate these risks, we recommend pairing RLVR with explicit safety objectives, conducting targeted red-teaming (including jailbreak robustness), curating and filtering replay data, reporting safety, factuality, and bias metrics alongside task performance.

References

2021. HME100K: A dataset for handwritten mathematical expressions. https://github.com/Phymond/HME100K. Accessed October 2025.

Josh Achiam, Steven Adler, Sandhini Agarwal, and et al. 2023. GPT-4 technical report. *arXiv:2303.08774*.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *ECCV*, pages 139–154.

Thomas Borsani, Andrea Rosani, Giuseppe Nicosia, and Giuseppe Di Fatta. 2025. Gradient similarity surgery in multi-task deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 95–111. Springer.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. 2020. Dark experience for general continual learning: A strong, simple baseline. In *NeurIPS*, volume 33, pages 15920–15930.

Nikhil Chandak, Shashwat Goel, and Ameya Prabhu. 2025. Incorrect baseline evaluations call into question recent llm-rl claims. https://safe-lip-9a8.notion.site/Incorrect-Baseline-Evaluations-Call-into-Question-Recent pvs=4. Notion Blog.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Efficient lifelong learning with A-GEM. In *ICLR*.

Jiaze Chen and 1 others. 2025a. Seed1.5-thinking: Advancing superb reasoning models with reinforcement learning.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, and 1 others. 2025b. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 794–803.

Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. 2020. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. In *Advances in Neural Information Processing Systems*, volume 33, pages 2039–2050.

Xu Chu, Xinrong Chen, Guanyu Wang, Zhijie Tan, Kui Huang, Wenyu Lv, Tong Mo, and Weiping Li. 2025. Qwen look again: Guiding vision-language reasoning models to re-attention visual information. arXiv:2505.23558. Introduces BRPO with reflection and visual re-attention to reduce hallucinations.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Roberto Cipolla, Yarin Gal, and Alex Kendall. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491.

Xingyu Dang, Christina Baek, J Zico Kolter, and Aditi Raghunathan. 2025. Assessing diversity collapse in reasoning. In *Scaling Self-Improving Foundation Models without Human Supervision*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, and 1 others. 2024. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 11198–11201.

K. Ethayarajh. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Robert M. French. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135.

Ling Fu, Biao Yang, Zhebin Kuang, Jiajun Song, Yuzhe Li, Linghao Zhu, Qidi Luo, Xinyu Wang, Hao Lu, Mingxin Huang, Zhang Li, Guozhi Tang, Bin Shan, Chunhui Lin, Qi Liu, Binghong Wu, Hao Feng, Hao Liu, Can Huang, and 5 others. 2024. Ocrbench v2: An improved benchmark for evaluating large multimodal models on visual text localization and reasoning. *arXiv preprint arXiv:2501.00321*.

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. 2025. Srft: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*.

Shivank Garg, Ayush Singh, Shweta Singh, and Paras Chopra. 2025. Ipo: Your language model is secretly a preference classifier. *arXiv preprint arXiv:2502.16182*.

Daya Guo and DeepSeek-AI. 2025. Deepseek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*.

Yihang Guo, Tianyuan Yu, Liang Bai, Yanming Guo, Yirun Ruan, William Li, and Weishi Zheng. 2025. Revisit the imbalance optimization in multi-task learning: An experimental analysis. *arXiv preprint arXiv:2509.23915*.

Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. 2018. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhezheng Hao, Hong Wang, Haoyang Liu, Jian Luo, Jiarui Yu, Hande Dong, Qiang Lin, Can Wang, and Jiawei Chen. 2025. Rethinking entropy interventions in rlvr: An entropy change perspective. *arXiv preprint arXiv:2510.10150*.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv:1503.02531.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*.

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

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025a. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*.

Zhiyuan Hu, Yibo Wang, Hanze Dong, Yuhui Xu, Amrita Saha, Caiming Xiong, Bryan Hooi, and Junnan Li. 2025b. Beyond'aha!': Toward systematic meta-abilities alignment in large reasoning models. *arXiv preprint arXiv:2505.10554*.

Yipo Huang, Quan Yuan, Xiangfei Sheng, Zhichao Yang, Haoning Wu, Pengfei Chen, Yuzhe Yang, Leida Li, and Weisi Lin. 2024. Aesbench: An expert benchmark for multimodal large language models on image aesthetics perception. *arXiv preprint arXiv:2401.08276*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Yunjie Ji, Xiaoyu Tian, Sitong Zhao, Haotian Wang, Shuaiting Chen, Yiping Peng, Han Zhao, and Xiangang Li. 2025. Am-thinking-v1: Advancing the frontier of reasoning at 32b scale. *arXiv preprint arXiv:2505.08311*.

Ryo Kamoi, Yusen Zhang, Sarkar Snigdha Sarathi Das, Ranran Haoran Zhang, and Rui Zhang. 2025. Visonlyqa: Large vision language models still struggle with visual perception of geometric information. In *Conference on Language Modeling (COLM)*.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. ReferItGame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 787–798, Doha, Qatar. Association for Computational Linguistics.

Aniruddha Kembhavi, Michael Salvato, Bardia Fang, Alaaeldin El-Nouby, Ludovic Schmidt, and 1 others. 2016. A diagram is worth a dozen images: Developing a diagram taxonomy for diagram understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Dataset page: https://prior.allenai.org/projects/diagram-understanding.

Diederik P Kingma. 2015. Adam: A method for stochastic optimization. *The third International Conference on Learning Representations*.

James Kirkpatrick, Razvan Pascanu, and Neil et al. Rabinowitz. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. 2023. Lisa: Reasoning segmentation via large language model.

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. 2024. Lisa: Reasoning segmentation via large language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nathan Lambert. 2025. Reinforcement learning from human feedback. *arXiv preprint arXiv:2504.12501*.

Bo Li, Kaichen Zhang, and Andrés Marafioti. 2025a. Multimodal open r1. `https://github.com/EvolvingLMMs-Lab/open-r1-multimodal`. Accessed: 2025-02-08.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. 2024. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

Chunyi Li, Jianbo Zhang, Zicheng Zhang, Haoning Wu, Yuan Tian, Wei Sun, Guo Lu, Xiongkuo Min, Xiaohong Liu, Weisi Lin, and 1 others. 2025b. R-bench: Are your large multimodal model robust to real-world corruptions? *IEEE Journal of Selected Topics in Signal Processing*.

Zhizhong Li and Derek Hoiem. 2016. Learning without forgetting. In *ECCV*, pages 614–629.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Yiqing Liang, Jielin Qiu, Wenhao Ding, Zuxin Liu, James Tompkin, Mengdi Xu, Mengzhou Xia, Zhengzhong Tu, Laixi Shi, and Jiacheng Zhu. 2025. Modomodo: Multi-domain data mixtures for multimodal llm reinforcement learning. *Preprint*, arXiv:2505.24871.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023a. Let's verify step by step: Improving mathematical reasoning with process supervision. In *OpenAI Technical Report*. Introduces PRM800K and shows advantages of process-supervised verifiers.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023b. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W Tsang. 2022. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, pages 2835–8856.

Bo Liu, Yihao Feng, Peter Stone, and Qiang Liu. 2023. Famo: Fast adaptive multitask optimization. In *Advances in Neural Information Processing Systems*, volume 36, pages 57226–57243.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021a. Conflict-averse gradient descent for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 18878–18890.

Haotian Liu and 1 others. 2024a. Mmbench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision (ECCV)*.

Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. 2021b. Towards impartial multi-task learning. In *International Conference on Learning Representations*.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. 2025a. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*.

Shikun Liu, Edward Johns, and Andrew J. Davison. 2019. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1871–1880.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024b. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Zifan Liu and 1 others. 2024c. Measuring multimodal mathematical reasoning with the math-vision dataset. In *Advances in Neural Information Processing Systems (NeurIPS)*.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *NeurIPS*, pages 6467–6476.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Xinyue Lou, You Li, Jinan Xu, Xiangyu Shi, Chi Chen, and Kaiyu Huang. 2025. Think in safety: Unveiling and mitigating safety alignment collapse in multimodal large reasoning model. *arXiv preprint arXiv:2505.06538*.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022a. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022b. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *NeurIPS*.

U.-V. Marti and H. Bunke. 2002. The IAM-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of ACL*.

Minesh Mathew, Viraj Bagal, Rubèn Pérez Tito, Dimosthenis Karatzas, Ernest Valveny, and C. V. Jawahar. 2022. Infographicvqa. In *WACV*.

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

Paulius Micikevicius, Sharan Narang, Jonah Alben, Greg Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *International Conference on Learning Representations (ICLR)*.

Ankush Mishra, Karteek Alahari, and C. V. Jawahar. 2012. Scene text recognition using higher order language priors. In *British Machine Vision Conference (BMVC)*.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. 2022. Multi-task learning as a bargaining game. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*.

Long Ouyang, Jeff Wu, Xu Jiang, and et al. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2025. Verif: Verification engineering for reinforcement learning in instruction following. *arXiv preprint arXiv:2506.09942*.

Ivo Petrov, Jasper Dekoninck, Lyuben Baltadzhiev, Maria Drencheva, Kristian Minchev, Mislav Balunović, Nikola Jovanović, and Martin Vechev. 2025. Proof or bluff? evaluating llms on 2025 usa math olympiad. *arXiv preprint arXiv:2503.21934*.

Hoang Phan, Lam Tran, Quyen Tran, Ngoc Tran, Tuan Truong, Qi Lei, Nhat Ho, Dinh Phung, and Trung Le. 2025. Beyond losses reweighting: Empowering multi-task learning via the generalization perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2440–2450.

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. 2025. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Arijit Ray, Jiafei Duan, Reuben Tan, Dina Bashkirova, Rose Hendrix, Kiana Ehsani, Aniruddha Kembhavi, Bryan A. Plummer, Ranjay Krishna, Kuo-Hao Zeng, and Kate Saenko. 2024a. Sat: Spatial aptitude training for multimodal language models.

Arijit Ray, Jiafei Duan, Reuben Tan, Dina Bashkirova, Rose Hendrix, Kiana Ehsani, Aniruddha Kembhavi, Bryan A. Plummer, Ranjay Krishna, Kuo-Hao Zeng, and Kate Saenko. 2024b. Sat: Spatial aptitude training for multimodal language models. *arXiv preprint arXiv:2412.07755*.

Sylvestre Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. iCaRL: Incremental classifier and representation learning. In *CVPR*, pages 5533–5542.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne. 2019. Experience replay for continual learning. In *NeurIPS*, pages 350–360.

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

Dmitry Senushkin, Nikolay Patakin, Arseny Kuznetsov, and Anton Konushin. 2023. Independent component alignment for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20083–20093.

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

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2025. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning Representations*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.06941*.

Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. 2020. Textcaps: A dataset for image captioning with reading comprehension. In *European Conference on Computer Vision (ECCV)*.

A. Singh and 1 others. 2021. TextOCR: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. *arXiv preprint arXiv:2105.05486*.

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

Core Team, Zihao Yue, Zhenru Lin, Yifan Song, Weikun Wang, Shuhuai Ren, Shuhao Gu, Shicheng Li, Peidian Li, Liang Zhao, Lei Li, Kainan Bao, Hao Tian, Hailin Zhang, Gang Wang, Dawei Zhu, Cici, Chenhong He, Bowen Ye, and 55 others. 2025. Mimo-vl technical report. *Preprint*, arXiv:2506.03569.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhu Chen. 2025a. Vl-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. *arXiv preprint arXiv:2504.08837*.

Shibo Wang and Przemyslaw Kanwar. 2019. Bfloat16: The secret to high performance on cloud tpus. Google Cloud Blog.

Teng Wang, Zhangyi Jiang, Zhenqi He, Shenyang Tong, Wenhan Yang, Yanan Zheng, Zeyu Li, Zifan He, and Hailei Gong. 2025b. Towards hierarchical multi-step reward models for enhanced reasoning in large language models. *arXiv preprint arXiv:2503.13551*.

Xiyao Wang, Zhengyuan Yang, Chao Feng, Hongjin Lu, Linjie Li, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. 2025c. Sota with less: Mcts-guided sample selection for data-efficient visual reasoning self-improvement. *arXiv preprint arXiv:2504.07934*.

Zhenhailong Wang, Xuehang Guo, Sofia Stoica, Haiyang Xu, Hongru Wang, Hyeonjeong Ha, Xiusi Chen, Yangyi Chen, Ming Yan, Fei Huang, and 1 others. 2025d. Perception-aware policy optimization for multimodal reasoning. *arXiv preprint arXiv:2507.06448*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Penghao Wu and Saining Xie. 2023. V*: Guided visual search as a core mechanism in multimodal llms. *arXiv preprint arXiv:2312.14135*.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. 2023. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*.

Yang Yao, Xuan Tong, Ruofan Wang, Yixu Wang, Lujundong Li, Liang Liu, Yan Teng, and Yingchun Wang. 2025a. A mousetrap: Fooling large reasoning models for jailbreak with chain of iterative chaos. *arXiv preprint arXiv:2502.15806*.

Zijun Yao, Yantao Liu, Yanxu Chen, Jianhui Chen, Junfeng Fang, Lei Hou, Juanzi Li, and Tat-Seng Chua. 2025b. Are reasoning models more prone to hallucination? *arXiv preprint arXiv:2505.23646*.

En Yu, Kangheng Lin, Liang Zhao, Jisheng Yin, Yana Wei, Yuang Peng, Haoran Wei, Jianjian Sun, Chunrui Han, Zheng Ge, and 1 others. 2025. Perception-r1: Pioneering perception policy with reinforcement learning. *arXiv preprint arXiv:2504.07954*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836.

Xin Yue and 1 others. 2023. Mmmu: A massive multi-discipline multimodal understanding benchmark for expert agi. *arXiv preprint arXiv:2311.16502*.

Xin Yue and 1 others. 2024. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. *arXiv preprint arXiv:2409.02813*.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *2nd AI for Math Workshop @ ICML 2025*.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526. Supplementary/alternate venue listings exist.

Hongzhi Zhang, Jia Fu, Jingyuan Zhang, Kai Fu, Qi Wang, Fuzheng Zhang, and Guorui Zhou. 2025. Rlep: Reinforcement learning with experience replay for llm reasoning. arXiv:2507.07451. Experience replay improves stability and final accuracy on AIME/AMC.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2024a. Lmms-eval: Reality check on the evaluation of large multimodal models. *Preprint*, arXiv:2407.12772.

Rui Zhang and 1 others. 2024b. Mathverse: Does your multi-modal llm truly see the diagrams? *arXiv preprint arXiv:2403.14624*.

Ruxin Zheng, Yifei Li, Xiang Li, Chong Chen, and 1 others. 2023. Secrets of rlhf in large language models part i: Ppo. arXiv:2307.04964. Analyzes PPO variants and shows PPO-ptx mitigates knowledge decline.

Daniel M. Ziegler, Nisan Stiennon, Jeff Wu, and et al. 2019. Fine-tuning language models from human preferences. *arXiv:1909.08593*.

A Appendix

In this section, we provide detailed statistics of the training datasets used in our experiments, along with implementation details and additional experimental results.

A.1 Proof of Entropy-Regularized Priority Allocation

We provide the proof that the RECAP reweighting rule is the unique solution of an entropy-regularized priority allocation problem.

Lemma 1 (Existence of an optimizer)**.** *For any $T > 0$ and any score vector $s^{(t)} \in \mathbb{R}^K$, the optimization problem in Equation 4 admits at least one maximizer $p^\star \in \Delta^K$.*

Proof. The feasible set Δ^K is compact. The map

$$p \mapsto \langle p, s^{(t)} \rangle$$

is continuous, and the entropy $\mathcal{H}(p)$ is continuous on Δ^K under the convention $0 \log 0 = 0$. Therefore, Φ_t is continuous on a compact set. By the extreme value theorem, Φ_t attains its maximum on Δ^K. Hence, there exists at least one maximizer $p^\star \in \Delta^K$. □

Lemma 2 (Interior optimality)**.** *Any maximizer $p^\star$ of Equation 4 lies in the interior of the simplex, i.e.,*

$$p_k^\star > 0, \qquad k = 1, \dots, K.$$

Proof. Assume for contradiction that $p^\star$ is a maximizer and that $p_j^\star = 0$ for some j. Since $\sum_{k=1}^K p_k^\star = 1$, there exists some index $i \neq j$ such that $p_i^\star > 0$.

For $\varepsilon \in (0, p_i^\star)$, define the feasible perturbation

$$p(\varepsilon) = p^\star + \varepsilon e_j - \varepsilon e_i,$$

where e_i and e_j denote the standard basis vectors. Then $p(\varepsilon) \in \Delta^K$. The change in objective value is

$$\Phi_t(p(\varepsilon)) - \Phi_t(p^\star) = \varepsilon\big(s_j^{(t)} - s_i^{(t)}\big) + T\Big(\mathcal{H}(p(\varepsilon)) - \mathcal{H}(p^\star)\Big).$$

Only coordinates i and j change, so the entropy difference is

$$\mathcal{H}(p(\varepsilon)) - \mathcal{H}(p^\star) = -\varepsilon \log \varepsilon - (p_i^\star - \varepsilon) \log(p_i^\star - \varepsilon) + p_i^\star \log p_i^\star.$$

Using a first-order expansion around $p_i^\star > 0$, we have

$$-(p_i^\star - \varepsilon) \log(p_i^\star - \varepsilon) + p_i^\star \log p_i^\star = \varepsilon(\log p_i^\star + 1) + O(\varepsilon^2).$$

Therefore,

$$\mathcal{H}(p(\varepsilon)) - \mathcal{H}(p^\star) = -\varepsilon \log \varepsilon + \varepsilon(\log p_i^\star + 1) + O(\varepsilon^2).$$

Substituting this into the objective difference gives

$$\Phi_t(p(\varepsilon)) - \Phi_t(p^\star) = -T\varepsilon \log \varepsilon + \varepsilon\Big[s_j^{(t)} - s_i^{(t)} + T(\log p_i^\star + 1)\Big] + O(\varepsilon^2).$$

As $\varepsilon \to 0^+$, the term $-\varepsilon \log \varepsilon$ is positive and dominates all linear $O(\varepsilon)$ terms. Since $T > 0$, it follows that for sufficiently small $\varepsilon > 0$,

$$\Phi_t(p(\varepsilon)) > \Phi_t(p^\star),$$

which contradicts the optimality of $p^\star$. Hence, no coordinate of $p^\star$ can be zero, and therefore $p_k^\star > 0$ for all k. □

Proposition 1 (First-order optimality conditions)**.** *Let $p^\star$ be a maximizer of Equation 4. Then there exists a scalar $\nu \in \mathbb{R}$ such that, for every $k = 1, \dots, K$,*

$$s_k^{(t)} - T(\log p_k^\star + 1) + \nu = 0.$$

Proof. By Lemma Equation 2, any maximizer $p^\star$ lies in the interior of the simplex. Hence, the non-negativity constraints are inactive at $p^\star$, and we only need to enforce the equality constraint $\sum_{k=1}^K p_k = 1$.

The Lagrangian is

$$\mathcal{L}(p, \nu) = \sum_{k=1}^K p_k s_k^{(t)} - T \sum_{k=1}^K p_k \log p_k + \nu \left(\sum_{k=1}^K p_k - 1 \right).$$

Stationarity at $p^\star$ requires

$$\frac{\partial \mathcal{L}}{\partial p_k}(p^\star, \nu) = 0, \qquad k = 1, \ldots, K.$$

Since

$$\frac{\partial}{\partial p_k} \left(-p_k \log p_k \right) = -(\log p_k + 1),$$

we obtain

$$s_k^{(t)} - T(\log p_k^\star + 1) + \nu = 0,$$

which proves the claim. □

Theorem 2 (Closed-form solution of entropy-regularized priority allocation)**.** *For any $T > 0$, the optimization problem in Equation 4 has a unique maximizer $p^{(t),\star} \in \Delta^K$, given by*

$$p_k^{(t),\star} = \frac{\exp\left(s_k^{(t)} / T \right)}{\sum_{j=1}^K \exp\left(s_j^{(t)} / T \right)}, \qquad k = 1, \ldots, K.$$

Proof. Existence follows from Lemma 1. By Lemma 2 and Proposition 1, any maximizer $p^\star$ satisfies

$$s_k^{(t)} - T(\log p_k^\star + 1) + \nu = 0$$

for some scalar $\nu \in \mathbb{R}$. Rearranging gives

$$\log p_k^\star = \frac{s_k^{(t)} + \nu}{T} - 1.$$

Exponentiating both sides yields

$$p_k^\star = \exp\left(\frac{s_k^{(t)}}{T} \right) \exp\left(\frac{\nu}{T} - 1 \right).$$

The factor $\exp(\nu/T - 1)$ is independent of k. Therefore,

$$p_k^\star \propto \exp\left(\frac{s_k^{(t)}}{T} \right).$$

Enforcing the simplex constraint $\sum_{k=1}^K p_k^\star = 1$ gives

$$p_k^\star = \frac{\exp\left(s_k^{(t)} / T \right)}{\sum_{j=1}^K \exp\left(s_j^{(t)} / T \right)},$$

which proves the closed form.

It remains to show uniqueness. The map

$$p \mapsto \langle p, s^{(t)} \rangle$$

is linear, and the entropy map $p \mapsto \mathcal{H}(p)$ is strictly concave on the interior of Δ^K. Since $T > 0$, Φ_t is strictly concave on the interior of Δ^K. By Lemma 2, every maximizer lies in the interior. Therefore, there cannot be two distinct maximizers, and the maximizer is unique. □

Proposition 2 (RECAP reweighting rule). *Define the RECAP loss weights by*

$$\lambda_k^{(t)} := K p_k^{(t),\star}.$$

Then

$$\lambda_k^{(t)} = \frac{K \exp\left(s_k^{(t)}/T\right)}{\sum_{j=1}^{K} \exp\left(s_j^{(t)}/T\right)}, \qquad k = 1, \ldots, K,$$

and the weights satisfy the mean-normalization property

$$\frac{1}{K} \sum_{k=1}^{K} \lambda_k^{(t)} = 1.$$

Proof. Substituting the closed-form optimizer from Theorem 2 into the definition $\lambda_k^{(t)} := K p_k^{(t),\star}$ gives

$$\lambda_k^{(t)} = K \frac{\exp\left(s_k^{(t)}/T\right)}{\sum_{j=1}^{K} \exp\left(s_j^{(t)}/T\right)} = \frac{K \exp\left(s_k^{(t)}/T\right)}{\sum_{j=1}^{K} \exp\left(s_j^{(t)}/T\right)}.$$

Moreover,

$$\frac{1}{K} \sum_{k=1}^{K} \lambda_k^{(t)} = \frac{1}{K} \sum_{k=1}^{K} K p_k^{(t),\star} = \sum_{k=1}^{K} p_k^{(t),\star} = 1,$$

where the last equality follows from $p^{(t),\star} \in \Delta^K$. This proves the proposition. □

A.2 Data statistics and implementation details

Table 3 reports the full statistics of the training corpora used in our experiments. For LLaVA-OneVision-OCR, we extract OCR-focused subsets from the official LLaVA-OneVision release (Li et al., 2024): IIIT5K (Mishra et al., 2012), HME100K (hme, 2021), IAM (Marti and Bunke, 2002), TextCaps (Sidorov et al., 2020), and TextOCR (Singh et al., 2021) alongside release-provided synthetic/curated subsets (rendered_text, k12_printing, chrome_writing). These images are resized so that the longer side is ≤ 512 px while preserving aspect ratio to mitigate out-of-memory errors without altering task semantics.

Table 3: **Data statistics of each data source.** We present the original volume of data (# samples).

Dataset	Domain	Answer Type	Rewards/Objectives	# samples
RefCOCO (Kazemzadeh et al., 2014)	Referring Expression Comprehension	2D Bounding Box	IoU, Answer Format	321327
LLaVA-OneVision-OCR (Li et al., 2024)	Scene Text-Centric Visual Question Answering	Natural Language	Next Token Prediction	66468
ThinkLite-VL-70k (Wang et al., 2025c)	Math Reasoning & Natural Image/Chart Understanding	Natural Language	Acc, Thinking Format	69997
LISA-train (Lai et al., 2023)	Referring Expression	2D Bounding Box	IoU, Thinking Format	1326
GeoQAV (Li et al., 2025a)	Math Visual Question Answering	Multiple Choice	Acc, Thinking Format	1969
SAT-train (Ray et al., 2024a)	Spatial Visual Question Answering	Natural Language	Acc, Thinking Format	15000
ScienceQA-train (Lu et al., 2022a)	Science Visual Question Answering	Multiple Choice	Acc, Thinking Format	6218

We optimize with GRPO and SFT losses using AdamW (Loshchilov and Hutter, 2017) (β_1=0.9, β_2=0.999, ε=10^{-8}). The learning rate follows a linear schedule: 10% warmup to $\eta_{\max}$=1×10^{-6}, then linear decay to 0. Window size W and temperature T are set to 10 and 5.0, respectively, in our experiments. Due to the large computational requirements of RL training, we find that setting $T = 5$ and $\alpha = 0.5$ works reasonably well in the RLVR-only setting. For simplicity, we keep this configuration for the Hybrid setup and do not perform additional hyperparameter tuning in the large-scale setting. All runs use bfloat16 precision (Wang and Kanwar, 2019; Micikevicius et al., 2018) and FlashAttention kernels (Dao et al., 2022) for memory- and throughput-efficient attention. We enable thinking mode on reasoning tasks by enforcing structured traces (i.e., wrapping thoughts in `<think>...</think>`), which has been shown to improve reasoning and transparency (Hu et al., 2025b; Xie et al., 2025; Chen et al., 2025b).

For the larger hybrid setting, each model is trained for 500 steps on 8 GPUs using data parallelism, with per-device batch size of 1 and 2 gradient accumulation steps (effective batch size 16), and 4 rollouts per prompt (64 rollouts per optimizer step). We evaluate our models on a broad suite of widely used VLM benchmarks spanning general multimodal understanding, visual reasoning, math-in-vision, OCR, and accessibility: LISA (Lai et al., 2024), MMMU-Pro (Yue et al., 2024), AI2D (Kembhavi et al., 2016), MathVista (Lu et al., 2023), MathVision (Liu et al., 2024c), MathVerse (Zhang et al., 2024b), MMBench (Liu et al., 2024a), VizWiz (Gurari et al., 2018), and OCRBench v2 (Fu et al., 2024). Regarding the smaller setup, we also follow Liang et al. (2025) and train on 8 GPUs using data parallelism, with a per-device batch size of 2 and 4 rollouts per prompt, and then evaluate the models on SAT (Ray et al., 2024b), ScienceQA (Lu et al., 2022b), MathVista (Lu et al., 2023), ChartQA (Masry et al., 2022), InfoVQA (Mathew et al., 2022), and MMMU (Yue et al., 2023). Our evaluation protocol closely follows LMMS-Eval (Zhang et al., 2024a) and VLMEvalKit (Duan et al., 2024).

Evaluation prompt

Non-Thinking:
{Question}
Output the answer in `<answer>` `</answer>` tags.
Thinking:
{Question}
Output the thinking process in `<think>` `</think>` and final answer (option) in `<answer>` `</answer>` tags.

We provide the description for RECAP in Algorithm 1 and its pseudocode in Algorithm 2. To use it, one first computes the task losses, calls `update` to update the task weighting, and then obtains the weighted loss via `get_weighted_loss` to perform standard backpropagation. For typical settings (e.g., Qwen-3B/7B, $K < 10$ objectives, window size $W = 10$), our method introduces only $\Theta(KW)$ extra scalar operations and $\Theta(KW)$ memory, which is negligible compared to the $\Theta\left(10^{11}\right) - \Theta\left(10^{12}\right)$ FLOPs per step of the underlying model; in practice we observed no measurable slowdown.

Algorithm 1 Replay-Enhanced CApability Preservation (RECAP).

Require: Base parameters $\theta^{(0)}$; domain list $\{\mathcal{D}_n\}_{n=1}^{N}$ is a union of general $\{\mathcal{D}_1^G, \ldots, \mathcal{D}_M^G\}$ and reasoning domain $\{\mathcal{D}_1^R, \ldots, \mathcal{D}_L^R\}$; objectives $\{L_k\}_{k=1}^{K}$; window size W; temperature T; total iterations $T_{\max}$

Initialize $\lambda_k^{(0)} \leftarrow 1$ for all k

Initialize loss history buffers $\mathcal{B}_k$ of length $2W$ for each objective k

for $t = 1$ to $T_{\max}$ **do**

  Sample mini-batches from reasoning and replay data on each domain $\mathcal{D}_n$

  Compute per-domain, per-objective losses $\ell_{n,k}^{(t)}(\theta^{(t)})$

  Compute per-objective averaged losses

$$L_k^{(t)} \leftarrow \frac{1}{N} \sum_{n=1}^{N} \ell_{n,k}^{(t)}(\theta^{(t)}), \quad \forall k$$

  for $k = 1$ to K **do**

    Push $L_k^{(t)}$ into buffer $\mathcal{B}_k$ (FIFO)

  end for

  if $t \geq 2W$ **then**

    for $k = 1$ to K **do**

      Compute current-window mean: $\mu_k^{(t)} \leftarrow \frac{1}{W} \sum_{s=t-W+1}^{t} L_k^{(s)}$ and previous-window mean $\tilde{\mu}_k^{(t)} \leftarrow \frac{1}{W} \sum_{s=t-2W+1}^{t-W} L_k^{(s)}$

      Compute the instability:

$$\sigma_k^{(t)} = \sqrt{\frac{1}{2W-1} \sum_{s=t-2W+1}^{t} \left(L_k^{(s)} - \mu_k^{(t)}\right)^2}$$

      Compute the convergence rate $c_k^{(t)} \leftarrow \frac{\tilde{\mu}_k^{(t)}}{\mu_k^{(t)}}$, the inverse signal-to-noise ratio $i_k^{(t)} \leftarrow \frac{\sigma_k^{(t)}}{\mu_k^{(t)} + \tilde{\mu}_k^{(t)}}$ and the relative priority between domains:

$$s_k^{(t)} \leftarrow c_k^{(t)} + i_k^{(t)}$$

    end for

    Calculate softmax weights:

$$\lambda_k^{(t)} \leftarrow \frac{K \, \exp\left(s_k^{(t)}/T\right)}{\sum_{j=1}^{K} \exp\left(s_j^{(t)}/T\right)}, \quad \forall k$$

  else

    $\lambda_k^{(t)} \leftarrow 1$ for all k

  end if

  Compute final objective:

$$\mathcal{L}^{(t)}(\theta^{(t)}) \leftarrow \frac{1}{K} \sum_{k=1}^{K} \lambda_k^{(t)} L_k^{(t)}$$

  Update parameters:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_\theta \mathcal{L}^{(t)}(\theta^{(t)})$$

end for

Algorithm 2 Implementation of our proposed method in PyTorch-like Pseudocode

```
class RECAP:
def __init__(self, num_objectives, window_size, T=5.0):
    # num_objectives  (K,) number of objectives (rewards / losses)
    # window_size  W, length of each averaging window
    self.K = num_objectives
    self.W = window_size
    self.T = T
#     loss_history[k] stores a list of recent scalar losses for objective k
    self.loss_history = [deque(maxlen=2 * self.W)
                              for _ in range(num_objectives)]
#     current weights λ (no grad, treated as buffer)
    self.lambdas = torch.ones(self.K)
    self.step = 0

def get_weighted_loss(self, losses):
    # losses  (K,) tensor of per-objective losses L_k^(t)
    # returns L^(t) = (1/K) Σ_k λ_k^(t) L_k^(t)
    weights = self.lambdas.detach()
    loss = (weights * losses).mean()
    return loss

def update(self, losses):
    # losses  (K,) tensor of current per-objective losses (no grad needed)
    self.step += 1
#     append current losses into history (FIFO of length at most 2W)
    for k in range(self.K):
        self.loss_history[k].append(losses[k].detach())

#     if not enough history, keep uniform mixing
    if self.step < 2 * self.W:
        self.lambdas = torch.ones(self.K)
        return

#     compute per-objective signals c_k^(t) and i_k^(t)
    c = torch.zeros(self.K)
    i = torch.zeros(self.K)
    for k in range(self.K):
        hist = torch.stack(self.loss_history[k])
        recent = hist[-self.W:] # current window
        old = hist[-2*self.W:-self.W] # previous window
        mu = recent.mean()
        mu_old = old.mean()
        sigma = hist.std(unbiased=True)
#         c_k^(t) = μ̃_k^(t) / μ_k^(t)
        c[k] = mu_old / mu
#         i_k^(t) = σ_k^(t) / (μ_k^(t) + μ̃_k^(t))
        i[k] = sigma / (mu + mu_old)

#     priority scores s_k^(t) = c_k^(t) + i_k^(t)
    s = c + i
#     temperature-controlled softmax, normalized so (1/K) Σ_k λ_k = 1
    w = torch.softmax(s / self.T, dim=-1)
    self.lambdas = self.K * w.detach()
```

A.3 Prompts used in our experiments

The prompt used for training our model is shown in Figure 8.

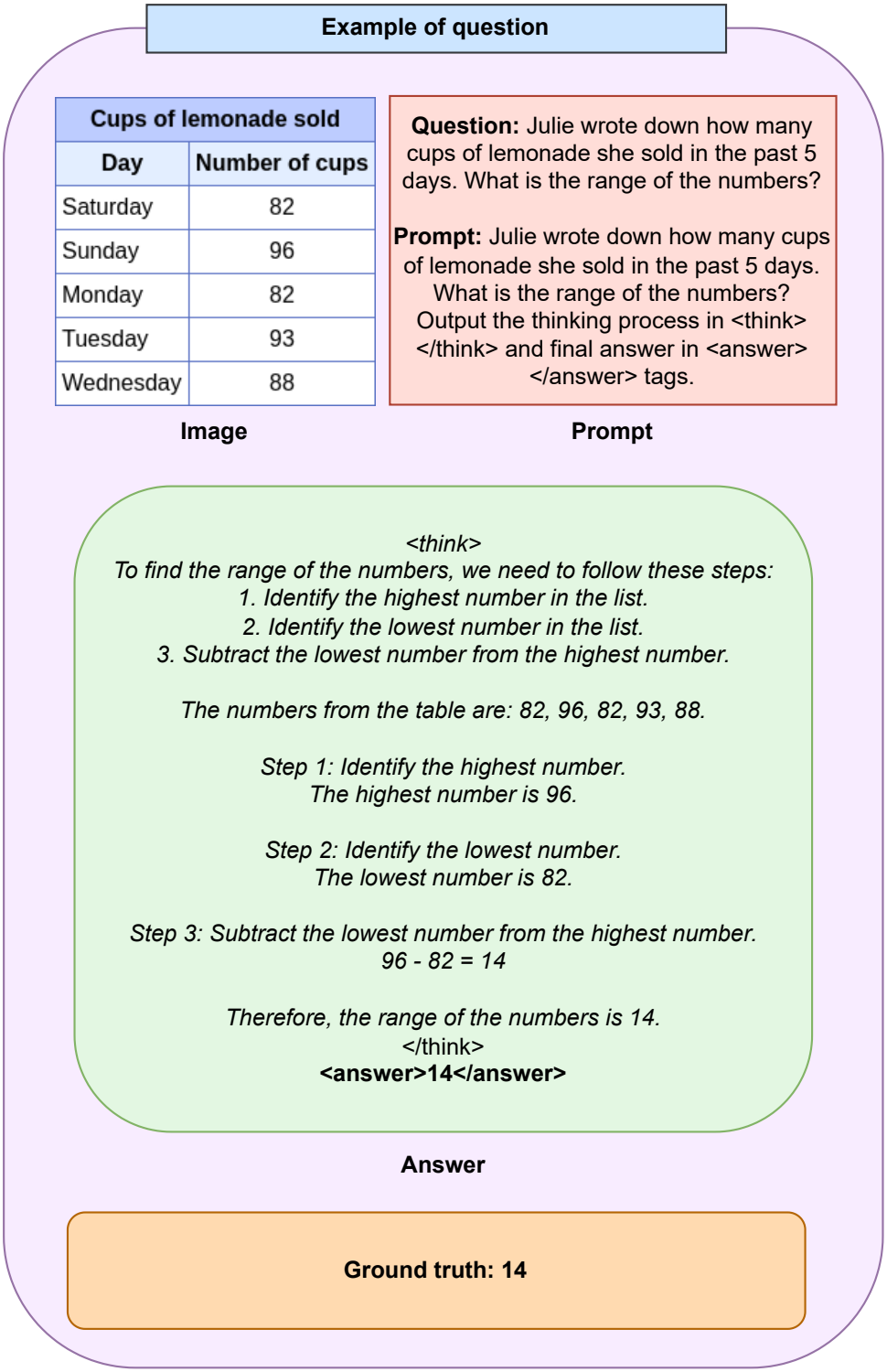


Figure 8: **Example conversation for finetuning.** We ask the model to produce its reasoning chain within `<think>` ... `</think>` and the final answer in `<answer>`...`</answer>` tags.

A.4 Qualitative examples

Figure 9 presents responses from models trained with reasoning data only and when including general data.

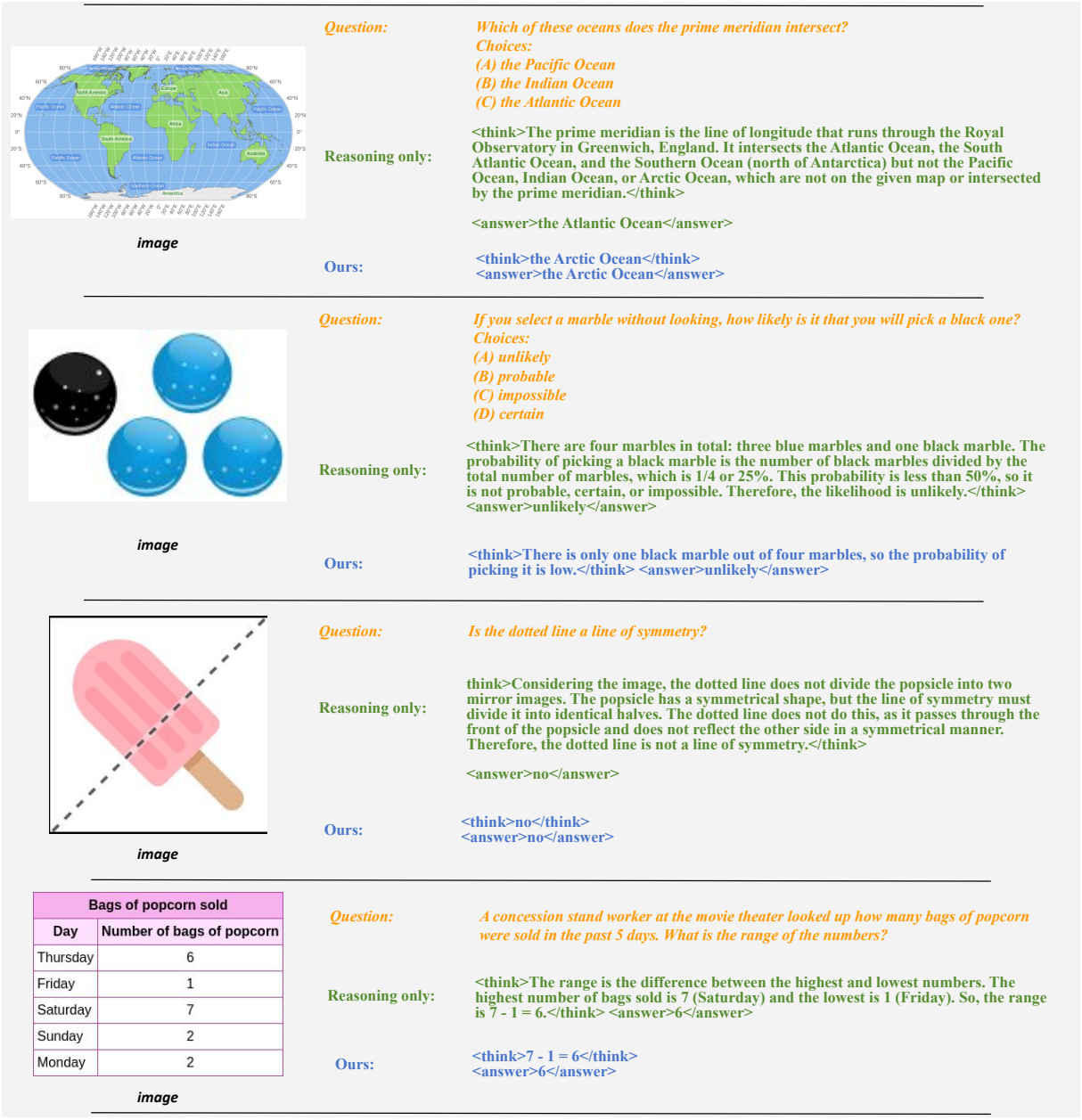


Figure 9: **Generated answers from the reasoning-only finetuned model and our model.** While the reasoning-only finetuned model generates long reasoning chains for simple visual questions, our model produces more concise answers, especially on non-math tasks.

A.5 Evolution of rollouts during training with the thinking reward

Below, we show generated responses from the model at different points during training, where we can observe the finetuned model gradually shortening its chain of thought on image-understanding tasks to provide final answers directly. This motivates our use of non-thinking rewards on tasks that do not require reasoning, instead of employing the same thinking-format reward for every task (Liang et al., 2025).

Example 1: rollout # 0

Prompt: Please provide the bounding box coordinate of the region this sentence describes: flicking tail.
Output the thinking process in <think> </think> and final answer in <answer> </answer> tags.
The output answer format should be as follows:
<answer>[x1, y1, x2, y2]</answer>
Answer: To determine the bounding box coordinates for the zebra that is flicking its tail, we need to follow these steps:
1. Identify the zebra in the image that is flicking its tail.
2. Observe the position and direction of the zebra's tail.
3. Define a bounding box that encapsulates the entire zebra, keeping in mind that the tail should be included in the box.
Upon inspection:
- The zebra on the right is flicking its tail, as evident from the blurred tail motion and the position of the tail.
- The bounding box should encompass the entire zebra, including its tail.
<answer>[291, 41, 432, 315]</answer>

Example 2: rollout # 500

Prompt: Please provide the bounding box coordinate of the region this sentence describes: second from right.
Output the thinking process in <think> </think> and final answer in <answer> </answer> tags.
The output answer format should be as follows:
<answer>[x1, y1, x2, y2]</answer>
Answer: <think>To determine the bounding box for the second from right, I first identify the rightmost object, which is the one on the far right, identified by its position and bounding box. The second from right is the one to the left of the rightmost object. By visual inspection, the bounding box for the second from right should start after the farthest object and extend leftward.</think>
<answer>[237, 72, 374, 328]</answer>

Example 3: rollout # 1000

Prompt: Please provide the bounding box coordinate of the region this sentence describes: boat blocked by cow stomach.
Output the thinking process in <think> </think> and final answer in <answer> </answer> tags.
The output answer format should be as follows:
<answer>[x1, y1, x2, y2]</answer>
Answer: <think> The sentence refers to the boat that is partially covered by the cow's stomach. This boat seems to be the one with the green and white colors.</think>
<answer>[327,21,631,280]</answer>

Example 4: rollout # 5000

Prompt: Please provide the bounding box coordinate of the region this sentence describes: guy on left with backpack on.
Output the thinking process in <think> </think> and final answer in <answer> </answer> tags.
The output answer format should be as follows:
<answer>[x1, y1, x2, y2]</answer>
Answer: <think>[114,53,236,378]</think>
<answer>[114,53,236,378]</answer>

A.6 Reward values at the end of training

Similar to what we observe in the main paper, Figure 10 showcases the final performance of our model and the uniform baseline. Results show near-parity on thinking formatting (<think></think> <answer> </answer>) and direct answer reward (<answer> </answer>) but consistent improvements on reasoning score, IoU and mean token accuracy (+2.01, +1.11, and +1.40 points respectively). This aligns with our design goal: once format signals saturate, down-weight them and shift capacity to harder, higher-variance objectives, improving accuracy while maintaining output format.

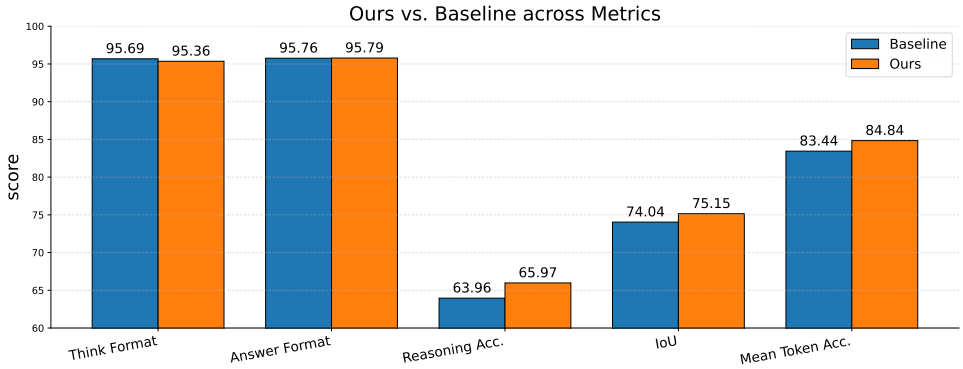


Figure 10: **Final performance across metrics.** We compare a uniform baseline with our dynamic reweighting. The gains on correctness-oriented metrics indicate that reallocating weight away from saturated format rewards toward harder objectives yields better solutions without sacrificing adherence to templates.

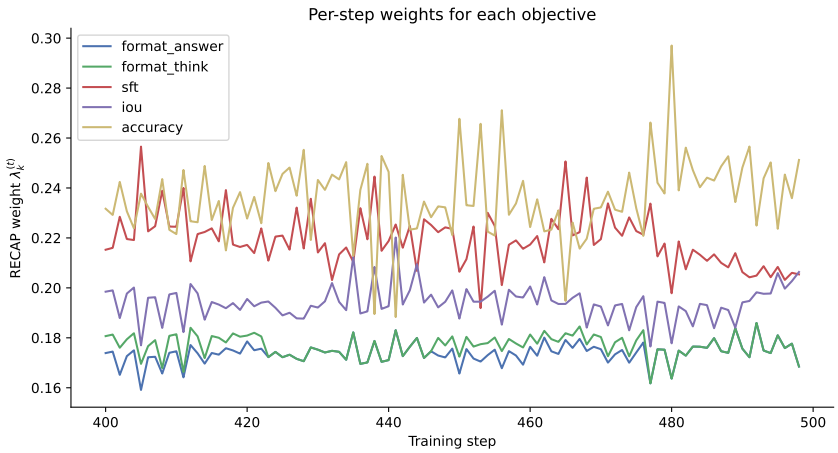


Figure 11: **Evolution of per-objective coefficients.** In the last 100 iterations, the coefficient for each objective is relatively consistent, with format rewards receiving the lowest focus while the supervised finetuning objective and accuracy rewards are emphasized due to their instability.

In Figure 11, we plot the coefficients of the five objectives used in the hybrid setup. From these

coefficients, we can rank the objectives by how strongly our method focuses on each objective, from low to high: format rewards, IoU reward, next-token prediction (on the OCR task), and reasoning accuracy.

A.7 Reward dynamics during training

Given the multi-objective nature of the problem, one might apply existing methods in the multi-task learning literature (Guo et al., 2025) for reweighting different objectives and rewards. In practice, this is difficult for two reasons. First, computing per-objective gradients is prohibitively expensive at LLM scale, especially under reinforcement learning. Second, on-policy RL signals are high-variance and non-stationary (Henderson et al., 2018), making per-iteration statistics unreliable indicators of task progress. As shown in Figure 12, all rewards fluctuate substantially within their $[0, 1]$ range, with the standard deviation of the total reward peaking near 0.9 around step ~ 20. Thus, we propose a method that utilizes a sliding window, which provides a more robust proxy for understanding convergence behavior.

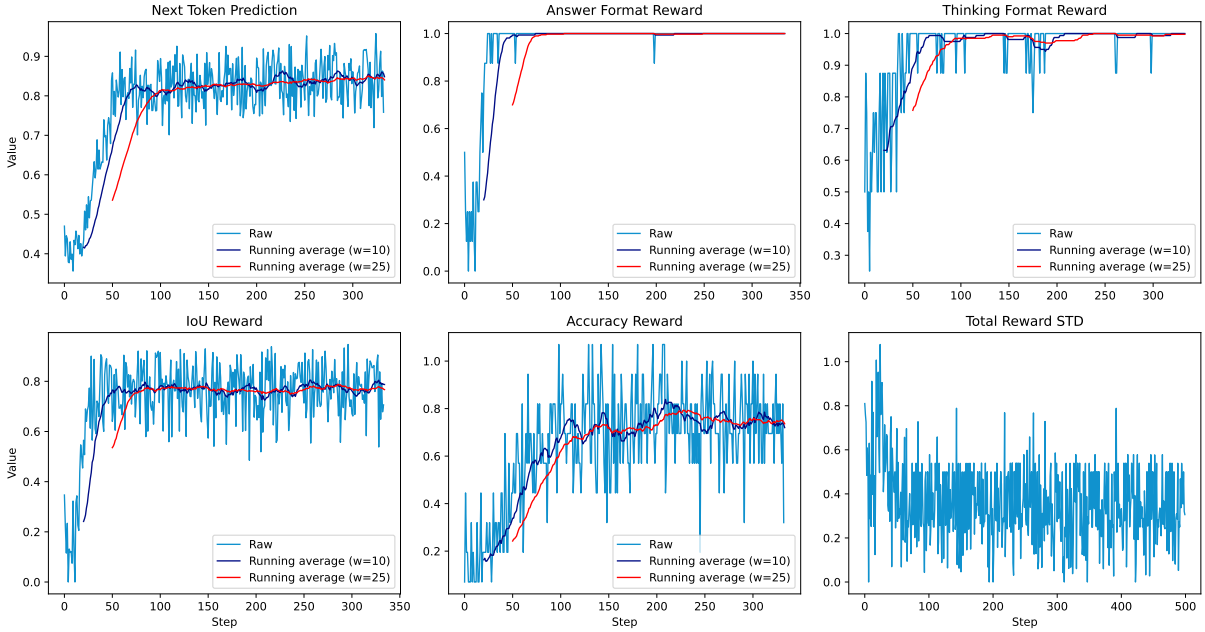
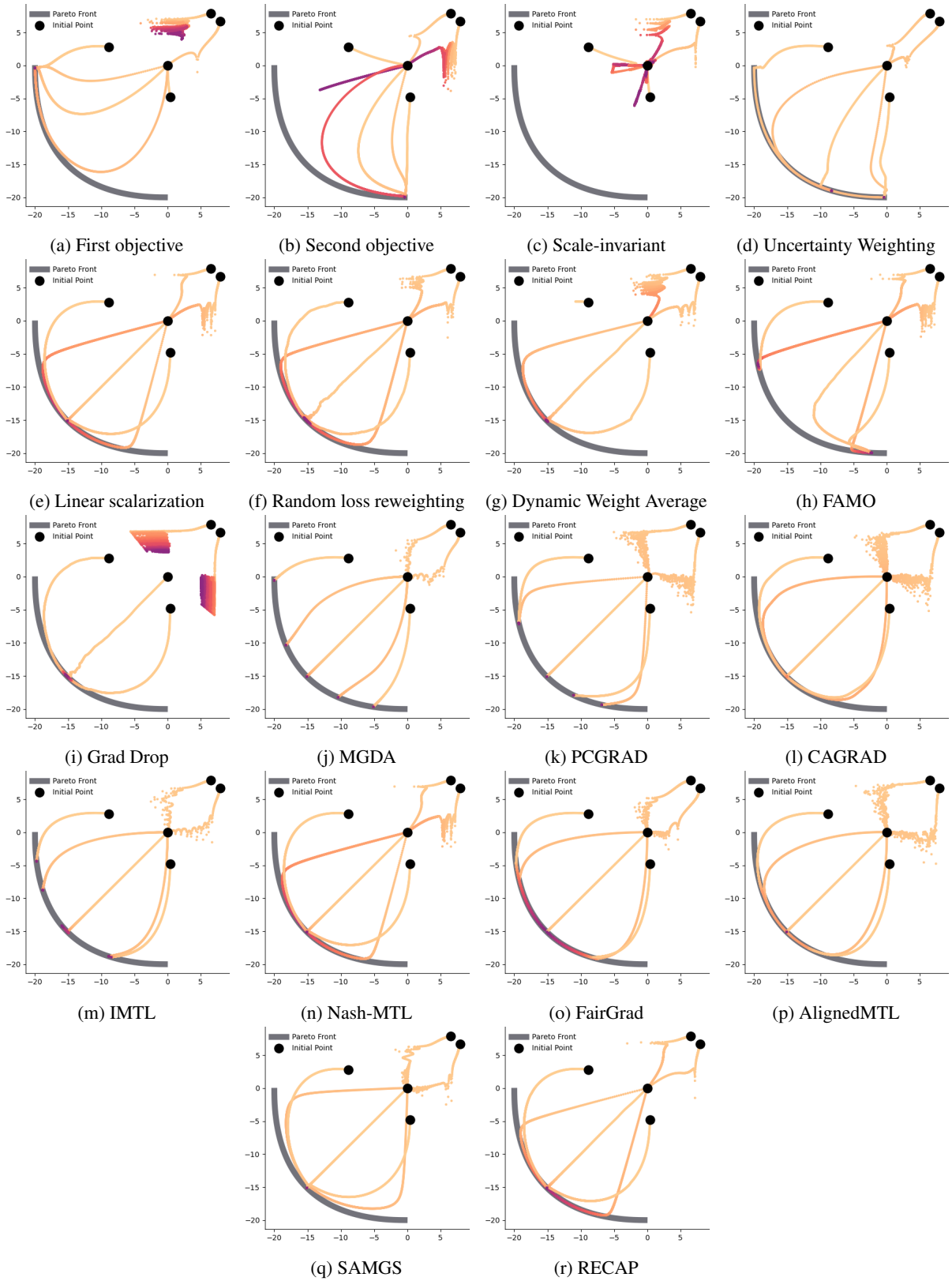


Figure 12: **Reward dynamics and variability during RLVR training.** Per-step rewards (light traces) and sliding-window means (dark curves) for six metrics: Next-Token Prediction, Answer-Format, Thinking-Format, IoU, Accuracy, and the Total-Reward Standard Deviation (lower-right). Asynchronous convergence and high variance motivate short-horizon statistics for dynamic objective reweighting rather than per-iteration magnitudes.

We also conduct an ablation on the effect of the window size W by increasing it from 10 (our default throughout the experiments) to larger values, up to 25, as shown in Figure 12. Since we train Qwen2.5-VL-7B for 500 iterations, setting $W = 25$ delays the onset of dynamic reweighting by 50 iterations, according to Algorithms 1 and 2, because our method requires $2W$ steps of history. By the time reweighting becomes active, some rewards have already entered a near-converged regime, which reduces the usefulness of the convergence-rate term and makes the scheme rely mostly on the instability term. We therefore choose $W = 10$ as a reasonable compromise between sensitivity and robustness: it accumulates enough information while remaining responsive to the current state of training.

Following the illustrative setup of Navon et al. (2022), we consider a synthetic two-task problem with a shared parameter vector and two scalar objectives. The corresponding Pareto front can be computed analytically and is shown in gray in Figure 13. We benchmark our method against established loss-magnitude-based methods (first row) and gradient-based multi-task learning methods (second row). To mimic the unstable nature of RL training, we inject noise into the first objective, which induces substantial fluctuations for competing methods, whereas our approach remains stable and closely tracks the Pareto front. Additional runtime comparisons in the appendix highlight that our method also achieves favorable wall-clock efficiency, a crucial advantage for large-scale RL training. The two tasks $L_1(x)$ and $L_2(x)$ are



(a) First objective (b) Second objective (c) Scale-invariant (d) Uncertainty Weighting

(e) Linear scalarization (f) Random loss reweighting (g) Dynamic Weight Average (h) FAMO

(i) Grad Drop (j) MGDA (k) PCGRAD (l) CAGRAD

(m) IMTL (n) Nash-MTL (o) FairGrad (p) AlignedMTL

(q) SAMGS (r) RECAP

Figure 13: A modified illustrative two-task example from (Navon et al., 2022) to show the convergence of comparative methods from different initialization points (black dots •). Each optimization trajectory is colored from yellow to red. The bold gray line represents the Pareto front. Overall, introducing noise in the first objective causes instability among many MTL methods. Methods that leverage per-step loss magnitude statistics like FAMO, DWA and UW exhibit considerable unstable convergence on different initializations.

defined on $x = (x_1, x_2)^\top \in \mathbb{R}^2$,

$$L_1(x) = f_1(x)g_1(x) + f_2(x)h_1(x) + 3\epsilon$$
$$L_2(x) = f_1(x)g_2(x) + f_2(x)h_2(x),$$

where $\epsilon \sim N(0, 1)$, and the functions are given by:

$$f_1(x) = \max\big(\tanh(0.5x_2), 0\big)$$
$$f_2(x) = \max\big(\tanh(-0.5x_2), 0\big)$$
$$g_1(x) = \log\Big(\max\big(|0.5(-x_1 - 7) - \tanh(-x_2)|, 0.000005\big)\Big) + 6$$
$$g_2(x) = \log\Big(\max\big(|0.5(-x_1 + 3) - \tanh(-x_2) + 2|, 0.000005\big)\Big) + 6$$
$$h_1(x) = \big((-x_1 + 7)^2 + 0.1(-x_1 - 8)^2\big)/10 - 20$$
$$h_2(x) = \big((-x_1 - 7)^2 + 0.1(-x_1 - 8)^2\big)/10 - 20.$$

We use five different starting points $\{(-8.5, 7.5), (0, 0), (9.0, 9.0), (-7.5, -0.5), (9.0, -1.0)\}$. Those points are optimized with Adam (Kingma, 2015) with a learning rate of 1e-2 for 10000 iterations. To justify our proposed reweighting mechanism, we provide comparisons against established multi-task learning techniques by including loss balancing methods such as UW (Cipolla et al., 2018), DWA (Liu et al., 2019), GradNorm (Chen et al., 2018), and RGW (Lin et al., 2022), FAMO (Liu et al., 2023) and gradient-based methods: PCGrad (Yu et al., 2020), CAGrad (Liu et al., 2021a), GradDrop (Chen et al., 2020), MGDA (Dong et al., 2015), IMTL (Liu et al., 2021b), Nash-MTL (Navon et al., 2022), FS-MTL (Phan et al., 2025), Aligned-MTL (Senushkin et al., 2023) and SAM-GS (Borsani et al., 2025). Their convergence behaviors are presented in Figure 13, from which we can see improvements across all initialized solutions over other MTL methods. While gradient-based MTL methods such as CAGRAD and NashMTL do not depend on the initial solutions, they suffer from slow convergence and higher per-step computational cost compared to RECAP.

Table 4: **Evaluation results on NYUv2 scene understanding.** Test performance for three tasks: semantic segmentation, depth estimation, and surface normal. We highlight the best loss-magnitude based MTL method in **bold** and gradient-based MTL method by underscore.

Complexity		Segmentation		Depth		Surface Normal					$\Delta\mathbf{m}\% \downarrow$
		mIoU ↑	Pix Acc ↑	Abs Err ↓	Rel Err ↓	Angle Distance ↓		Within t° ↑			
						Mean	Median	11.25	22.5	30	
	STL	38.30	63.76	0.6754	0.2780	25.01	19.21	30.14	57.20	69.15	
$\Theta(1)$	LS	39.29	65.33	0.5493	0.2263	28.15	23.96	22.09	47.50	61.08	5.59
	SI	38.45	64.27	0.5354	0.2201	27.60	23.37	22.53	48.57	62.32	4.39
	RLW	37.17	63.77	0.5759	0.2410	28.27	24.18	22.26	47.05	60.62	7.78
	DWA	39.11	65.31	0.5510	0.2285	27.61	23.18	24.17	50.18	62.39	3.57
	UW	36.87	63.17	0.5446	0.2260	**27.04**	22.61	23.54	49.05	63.65	4.05
	Ours	**41.26**	**66.79**	**0.5303**	**0.2203**	27.11	**22.23**	**24.64**	**50.88**	**64.02**	**0.77**
$\Theta(K)$	GradNorm	20.09	64.64	0.7200	0.2800	24.83	18.86	30.8	57.94	69.73	7.22
	MGDA	30.47	59.90	0.6070	0.2555	24.88	19.45	29.18	56.88	69.36	1.38
	PCGrad	38.06	64.64	0.5550	0.2325	27.41	22.80	23.86	49.83	63.14	3.97
	GradDrop	39.39	65.12	0.5455	0.2279	27.48	22.96	23.38	49.44	62.87	3.58
	CAGrad	39.79	65.49	0.5486	0.2250	26.31	21.58	25.61	52.36	65.58	0.20
	IMTL-G	39.35	65.60	0.5426	0.2256	26.02	21.19	26.2	53.13	66.24	−0.76
	FS-MTL	40.42	65.61	0.5389	0.2121*	25.03	19.75	28.90	56.19	68.72	−4.77*
	Nash-MTL	40.13	65.93	0.5261	0.2171	25.26	20.08	28.4	55.47	68.15	−4.04

Table 4 reports the performance of different MTL methods on the real-scene understanding benchmark, which includes one segmentation task and two pixel-level regression tasks. Overall, our method nearly matches the single-task baselines ($\Delta\mathbf{m}\% \downarrow \approx 0$) while being roughly $3\times$ more efficient in both runtime and memory, and it consistently outperforms all other loss-reweighting methods across all metrics (except

Angle Distance Mean, where it is competitive with Uncertainty Weighting). Notably, our approach even surpasses several established gradient-based methods, such as GradNorm, MGDA, PCGRAD, and GradDrop, while remaining roughly three times faster than gradient-based alternatives. We also observe a clear Pareto trade-off: although NashMTL achieves the highest overall relative improvement in $\mathbf{\Delta m}\% \downarrow$, it lags behind GradNorm and MGDA on the surface-normal task, whereas these methods incur substantial performance drops on segmentation and depth estimation.

Figure 14 plots the loss curves for three different objectives, showing stable optimization across all of them. In contrast, our RL rewards are much sparser than in this SFT setting, and the training curves in Figures 4 and 12 exhibit substantially higher fluctuations. This motivates a more robust loss-reweighting mechanism, as relying solely on instantaneous per-step loss values is not sufficiently representative of the underlying learning dynamics or objective progress.

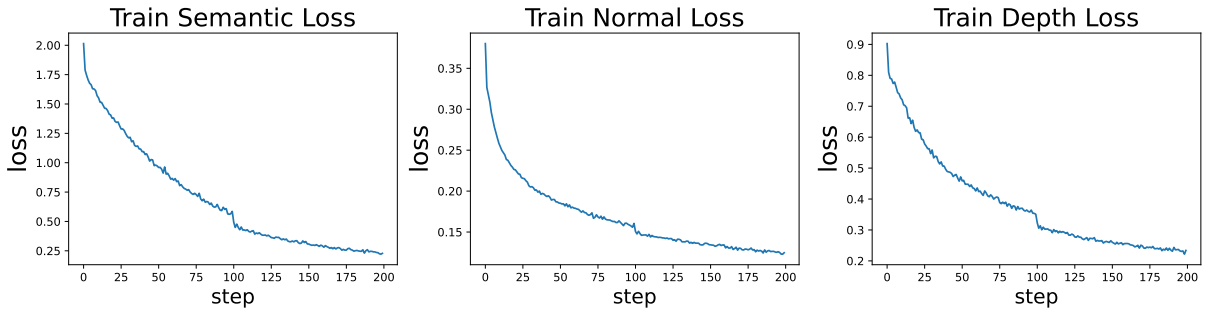


Figure 14: **Loss curves during training on NYUv2.** Compared to the training curves in our experiments (e.g. Figure 12), these curves in this experiment are much more smooth and stable, where per-step statistics can provide informative signals of the learning progress.

The running-time comparison in Figure 15 shows that, although effective in some scenarios, gradient-based MTL methods require storing and computing all task gradients, incurring $\Theta(K)$ space and time overhead where K is the number of objectives. In our illustrative setup with K=3, this already makes these methods about three times slower ($\sim$ 300s vs. $\sim$ 100s) than single-task baselines and other loss-reweighting approaches. In our main RLVR experiments, we have 4 domains with 2 objectives per domain (K=8), which would make gradient-manipulation methods roughly $8\times$ slower than standard training. For this reason, we focus on loss-reweighting mechanisms, which avoid such substantial computational overhead.

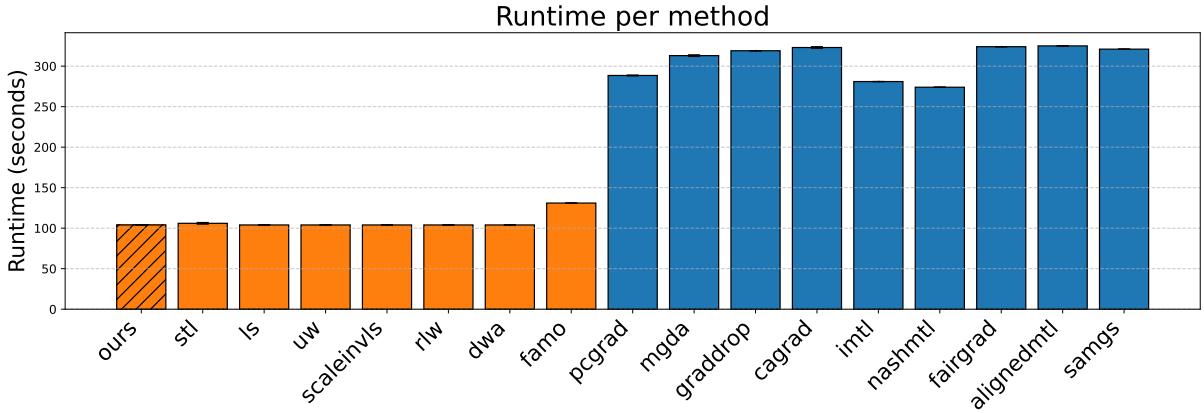
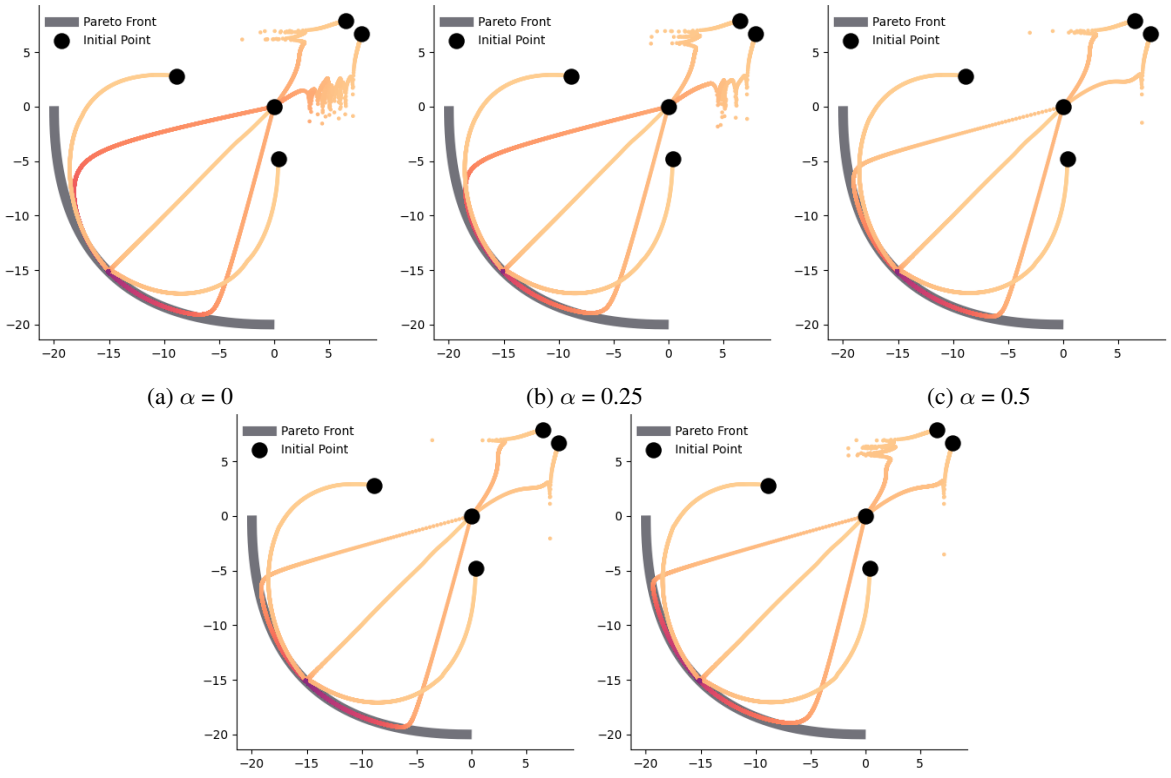


Figure 15: **Running time of different MTL methods.** While being robust to noise in some scenarios, gradient-based methods (denoted by blue) often cause significant overhead ($\approx K$ times as they compute per-objective gradients) compared to loss-magnitude based methods (denoted by orange) .

We conduct ablation studies on the temperature hyperparameter T and the trade-off α between the convergence rate and the inverse signal-to-noise ratio: $s_k^{(t)} = \alpha c_k^{(t)} + (1-\alpha) i_k^{(t)}$. From Figure 16, we

observe that intermediate values such as $\alpha = 0.5$ or 0.75 strike a good balance between the two terms and yield noticeably more stable convergence across all initializations.

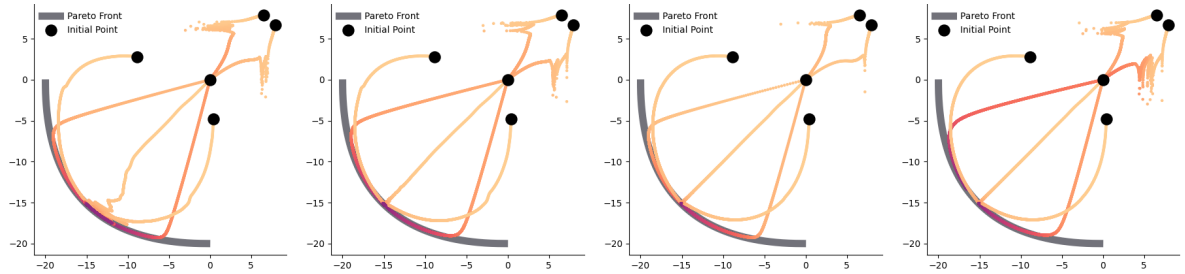


(a) $\alpha = 0$ (b) $\alpha = 0.25$ (c) $\alpha = 0.5$

(d) $\alpha = 0.75$ (e) $\alpha = 1.$

Figure 16: **Ablation on the trade-off α.** Using only the convergence rate ($\alpha = 1$) or only the inverse signal-to-noise ratio ($\alpha = 0$) leads to unstable learning for the second and first objectives, respectively.

For the temperature, setting T too low makes training unstable: as shown in Figure 17a, the trajectories exhibit strong fluctuations near the Pareto front. Conversely, setting T to a high value (e.g., $T = 30$) also harms convergence: for the two initializations farthest from the Pareto front, optimization requires many more steps to approach the front (the trajectories remain red for longer).



(a) $T = 0.1$ (b) $T = 1.0$ (c) $T = 10.0$ (d) $T = 30.0$

Figure 17: **Ablation studies on the temperature T.** $T > 1$ acts as a regularization to avoid extreme reweighting (one domain dominates others) and stabilizes the training.

Due to the large computational requirements of RL training, we find that setting $T = 5$ and $\alpha = 0.5$ works reasonably well in the RLVR-only setting. For simplicity, we keep this configuration for the Hybrid setup and do not perform additional hyperparameter tuning in the large-scale setting. Table 5 reports the results when varying the trade-off α, the temperature T, and the window size W. Although up-weighting the instability term can increase the weight assigned to the accuracy reward since this term is highly fluctuating, it comes at the cost of sacrificing essential perception skills. For example, $\alpha = 0.25$

Table 5: **Benchmark performance in RLVR-only setting.** Ablation results when varying the temperature and convergence rate-instability trade-off.

Model	SAT	ScienceQA	MathVista (mini)	ChartQA	InfoVQA	MMMU
MoDoMoDo	50.0	65.7	32.2	70.4	59.9	39.1
RECAP	55.2	71.6	33.2	70.4	60.8	42.4
α = 0.50, T = 1.0, W=10	52.0	71.6	33.4	68.1	58.5	40.4
α = 0.75, T = 5.0, W=10	54.4	71.2	32.9	70.0	60.7	41.0
α = 0.25, T = 5.0, W=10	55.3	72.2	33.7	66.1	56.8	39.3
α = 0.50, T = 5.0, W=50	51.9	70.5	32.9	69.9	59.7	40.8

improves performance on SAT and ScienceQA by 0.1% and 0.6%, respectively, but reduces ChartQA and InfoVQA performance by 3%. Similar to our illustrative example, decreasing the temperature induces higher variation across tasks—for instance, it yields the highest score on MathVista while reducing SAT performance by 3.2%.