Unified Generation and Self-Verification for Vision-Language Models via Advantage Decoupled Preference Optimization

Xinyu Qiu[1,2*] Heng Jia[1,2*] Zhengwen Zeng[2] Shuheng Shen[2†] Changhua Meng[2] Yi Yang[1] Linchao Zhu[1†]
[1]College of Computer Science and Technology, Zhejiang University
[2]Venus Team, Ant Group
{qiuxy020,jiaheng.dlut,zhulinchao7}@gmail.com
{zengzhengwen.zzw,shuheng.ssh,changhua.mch}@antgroup.com

Abstract

*Parallel test-time scaling typically trains separate generation and verification models, incurring high training and inference costs. We propose Advantage Decoupled Preference Optimization (**ADPO**), a unified reinforcement learning framework that jointly learns answer generation and self-verification within a single policy. ADPO introduces two innovations: **a preference verification reward** improving verification capability and **a decoupled optimization mechanism** enabling synergistic optimization of generation and verification. Specifically, the preference verification reward computes mean verification scores from positive and negative samples as decision thresholds, providing positive feedback when prediction correctness aligns with answer correctness. Meanwhile, the advantage decoupled optimization computes separate advantages for generation and verification, applies token masks to isolate gradients, and combines masked GRPO objectives, preserving generation quality while calibrating verification scores. ADPO achieves up to **+34.1%** higher verification AUC and **-53.5%** lower inference time, with significant gains of **+2.8%/+1.4%** accuracy on MathVista/MMMU, **+1.9** cIoU on ReasonSeg, and **+1.7%/+1.0%** step success rate on AndroidControl/GUI Odyssey.*

1. Introduction

Test-time scaling enhances reliability and performance by spending more compute at inference time, which can generally be realized in two ways [7, 25, 26]. Serial test-time scaling enables LLMs to enhance problem-solving capabilities by sequentially generating additional thinking tokens [25, 39]. Parallel test-time scaling generates multiple solutions simultaneously, then aggregates candidates and selects the best one [37, 42].

Serial test-time scaling, as demonstrated by DeepSeek-R1 [7] and OpenAI-o1 [26], achieves substantially improved performance in mathematics and coding domains. However, when transferring to multimodal domains, recent works have found that serial test-time scaling provides only limited performance improvements on image classification, video understanding and visual spatial understanding tasks [12, 14, 16, 34, 46]. These observations highlight the inherent limitations of serial test-time scaling and critically motivate the development of principles for parallel test-time scaling that more effectively support robust and efficient multimodal reasoning [3, 40, 43].

Current approaches [5, 33] deploy two separate models: a generator that creates potential solutions and a verifier that evaluates and ranks these candidates, leading to substantial performance gains. However, this methodology suffers from two major limitations: 1) *training resource intensive*, as it requires preparing two specialized datasets and training two independent models; 2) *deployment inefficiency*, since both models must operate concurrently during inference, demanding significant computational resources. Alternatively, training only a generator (with majority voting for selection) or only a verifier (using a base model for generation) yields limited performance improvements compared to training both components [4, 33, 37]. To address these limitations, we propose a novel reinforcement learning framework that trains a unified policy model to concurrently generate both solutions and self-verification scores of the solutions (Fig. 1).

We propose Advantage Decoupled Preference Optimization (ADPO), a unified framework that enables reliable self-verification while maintaining generation quality. This unified paradigm faces two challenges. First, training the model to verify its own outputs using binary rewards creates severe class imbalance. As the model improves, the proportion of correct answers increases substantially, causing ver-

[*] Equal contribution.
[†] Corresponding Authors.

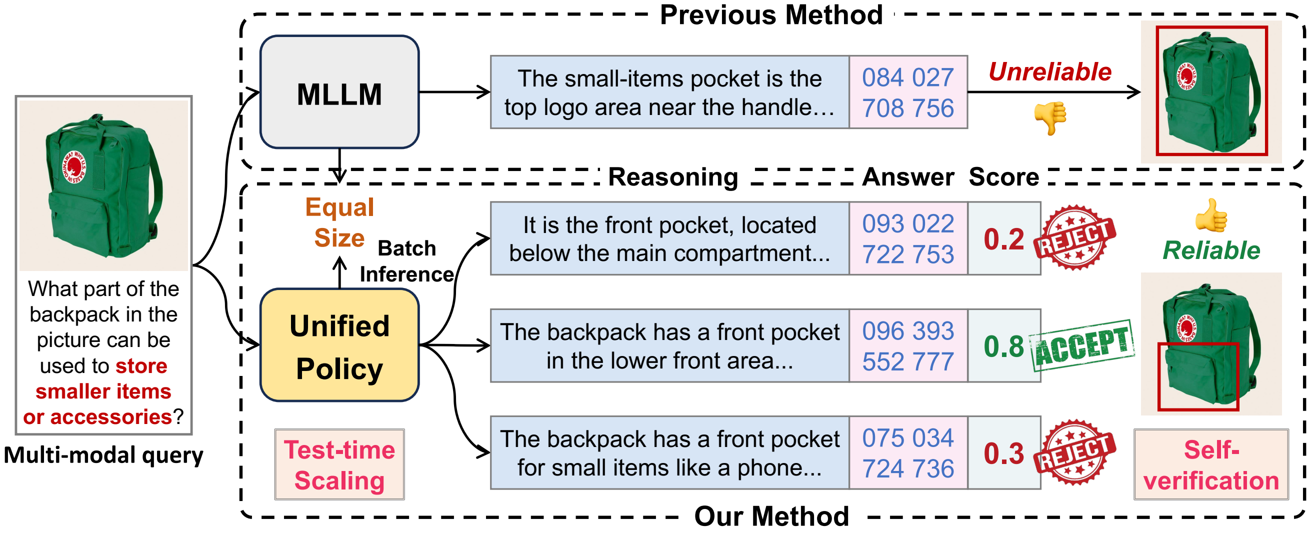

Previous Method
MLLM
The small-items pocket is the top logo area near the handle…
084 027 708 756
Unreliable
Reasoning
Answer
Score
Equal Size
Batch Inference
Unified Policy
It is the front pocket, located below the main compartment...
093 022 722 753
0.2
REJECT
Reliable
The backpack has a front pocket in the lower front area...
096 393 552 777
0.8
ACCEPT
The backpack has a front pocket for small items like a phone...
075 034 724 736
0.3
REJECT
What part of the backpack in the picture can be used to store smaller items or accessories?
Multi-modal query
Test-time Scaling
Self-verification
Our Method

Figure 1. **Overview of ADPO.** We build a unified reinforcement learning framework that jointly learns answer generation and self-verification within a single policy. The unified policy model provides reliable scoring that enables effective test-time scaling via best-of-N selection, significantly improving performance across multimodal tasks.

ification scores to collapse toward uniform predictions and eliminating gradient signals (Sec. 3.2). Second, naively aggregating answer and verification rewards leads to reward hacking. The model exploits the objective by deliberately producing incorrect answers while assigning low verification scores, achieving high total rewards while severely degrading generation performance.

To address the class imbalance challenge, we introduce a preference verification reward that reframes verification as a ranking problem. Instead of comparing scores against fixed thresholds, our approach adaptively partitions samples into positive/negative groups based on answer quality, then rewards the model when verification scores respect the relative quality ordering within each group. By computing advantages from pairwise comparisons rather than absolute labels, this formulation maintains informative gradient signals even under severe class imbalance.

To address the reward hacking challenge, we introduce advantage decoupled optimization that computes separate advantages for answer rewards and preference rewards. Specifically, answer advantages are estimated from answer rewards only, while preference advantages are derived solely from preference rewards. We then apply disjoint token-level masks to isolate gradient flows. In this way, answer advantages exclusively update generation tokens, while preference advantages solely affect verification score tokens. This decoupled computation ensures that improvements in generation quality are driven only by answer-based gradients, while verification calibration is shaped only by preference-based gradients, thereby eliminating reward hacking. Our contributions are summarized as follows:

1. Preference verification reward. We develop Preference Verification Reward, which maintains informativeness under severe class imbalance while improving calibration and robustness.

2. Advantage decoupled optimization. We introduce a principled approach to disentangle content generation and verification learning within a unified GRPO framework.

3. Comprehensive evaluation. Our method significantly improves task performance and verification quality: ADPO achieves +2.8/+1.4 accuracy gains on MathVista/MMMU, +1.9 cIoU on ReasonSeg, and +1.7/+1.0 step success rates on AndroidControl/GUI Odyssey.

2. Related Work

Test-Time Scaling. Recent work scales reasoning at test time via longer thinking tokens and majority voting for LLMs [7, 26, 31, 37]. Multimodal variants adapt this paradigm with R1-style objectives and structured CoT for VLMs [9, 19, 20, 28, 32, 41, 47]. In agentic settings, GUI agents adopt RL with explicit reasoning traces [6, 10, 17, 24, 29, 50]. However, recent "no-think" results suggest that more internal tokens do not always translate to better multimodal reasoning [12, 14]. We instead couple solution generation with a learned self-verification signal, enabling reliable performance scaling through best-of-N selection without fragile dependence on longer chains.

Multimodal Reward Modeling and Generative Verifiers. Another line studies reward modeling for multimodal alignment, including RLHF-style pipelines and chain-of-thought verification [33, 49]. Process or scalar reward models provide step-level or outcome supervision for reasoning [2, 4, 5, 8, 15, 27, 36]. Generative verifiers and LLM-as-judge train models to both solve and judge [48, 51]. In contrast, we use reinforcement learning to train a single policy for answer and calibrated confidence with separate advantages and mutual masking, and we do not finely control the positive/negative ratio in training data; instead, we employ preference reward rather than binary reward, enabling dependable best-of-N across multimodal tasks.

3. Method

We propose **Advantage Decoupled Preference Optimization**, a unified reinforcement learning framework that extends GRPO to jointly learn answer generation and self verification within a single policy. For each multimodal query, our model first generates an answer and then predicts a verification score. At inference, we use batch decoding to sample multiple candidate answers and select the one with the highest verification score as the final output. This unified generation and verification paradigm achieves reliable self verification and reduces inference latency. To realize this paradigm, we develop preference verification reward to strengthen verification (Sec. 3.2) and advantage decoupled optimization to enable joint optimization (Sec. 3.3).

3.1. Preliminary

Group Relative Policy Optimization (GRPO). GRPO [31] is a reinforcement learning algorithm that optimizes language models through group-based advantage estimation. Given a question q, the behavior policy $\pi_{\theta_{\text{old}}}$ samples a group of G candidate responses $\{o_i\}_{i=1}^{G}$. Each response $o_i{=}(o_{i,1},\ldots,o_{i,|o_i|})$ is a token sequence of length $|o_i|$ that receives a sequence-level reward R_i. GRPO estimates advantages by normalizing rewards within each group. The current policy π_θ is optimized with a PPO-style [30] clipped objective:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) &= \mathbb{E}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(\min\big(r_{i,t}(\theta)\hat{A}_{i,t},\\ &\text{clip}(r_{i,t}(\theta),1-\varepsilon,1+\varepsilon)\hat{A}_{i,t}\big)-\beta\,D_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\Big)\Bigg].\end{aligned} \tag{1}$$

where $r_{i,t}(\theta){=}\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}$ is the likelihood ratio, ε is the clipping parameter, β is the KL coefficient, D_{KL} is the KL regularization, π_{ref} is the reference policy, and the group-normalized advantage is defined as $\hat{A}_{i,t}{=}\frac{R_i-\text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})}$.
Answer Reward. We define the answer reward R^a to evaluate generation quality by comparing model outputs against ground-truth answers. The formulation of R^a varies based on task characteristics.

Table 1. **Prompt for ADPO training.**

<image><image_pad></image>
{Question} Output the thinking process in <think></think> and final answer in <answer></answer> tags.
After outputting the answer, you will act as a correctness evaluation assistant and assign a score between 0 and 1 to indicate how accurate the answer is. If you believe the answer is correct, the score should be close to 1; otherwise, it should be close to 0.
For example:
<think>reasoning process here</think>
<answer>answer here</answer>
<score>score number here</score>.

For *discrete tasks* with exact ground-truth answers (e.g., mathematical reasoning, agent navigation), we use correctness-based rewards:

$$R^a_{\text{discrete}} = \text{match}(y, y^*) \in \{0,1\}, \tag{2}$$

where match$(\cdot,\cdot)$ performs equivalence checking (e.g., numerical equivalence for math, action sequence matching for agents), and y and y^* represent the predicted and ground truth answers, respectively.

For *continuous tasks* where answer quality cannot be captured as simply correct or incorrect (e.g., visual grounding), we employ continuous rewards:

$$R^a_{\text{continuous}} = \text{sim}(y, y^*) \in [0,1], \tag{3}$$

where sim$(\cdot,\cdot)$ denotes a similarity metric. For visual grounding, we use Intersection-over-Union (IoU) to quantify the overlap between predicted and ground-truth regions.

3.2. Preference Verification Reward

We enable self-verification by instructing the model to evaluate its response. After generating an answer, the model outputs a verification score $s \in [0,1]$, which represents the predicted correctness. The prompt is provided in Tab. 1.

We first consider a simple binary verification reward as a baseline. The goal of self-verification is to align verification score with answer correctness so that correct answers receive higher scores and incorrect answers receive lower scores. We define a typical **binary verification reward** R^b by thresholding verification score s and answer reward R^a:

$$R^b = \mathbb{1}\{(s-\tau_s)(R^a-\tau_a) > 0\}, \tag{4}$$

The reward equals one when the predicted correctness and the actual correctness agree, while equals zero other-

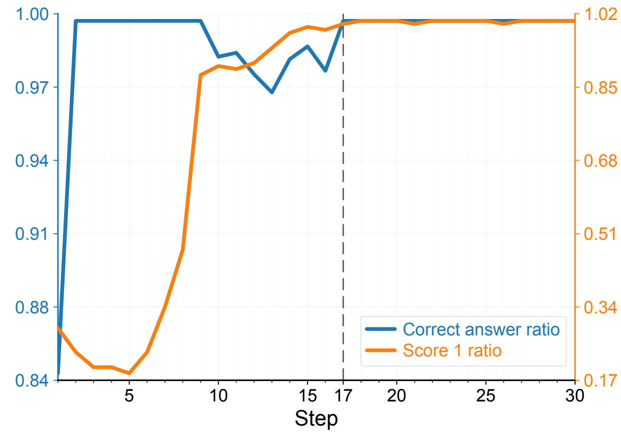


Figure 2. **Effect of class imbalance on the *Binary Verification Reward*.** The **Blue** line shows the proportion of correct answers among responses with binary verification reward = 1. The **Orange** line shows the proportion of answers with verification score = 1.

wise. This objective calibrates the verifier to produce scores aligned with answer correctness.

Class Imbalance Challenge. The binary verification reward is susceptible to the class imbalance challenge. As the model improves during training and generates more correct solutions, the proportion of correct samples grows substantially larger than incorrect ones. This severe imbalance between positive and negative samples causes the verification scores to collapse to the same value, thereby eliminating the model's discriminative capability.

As shown in Fig. 2, among responses that receive a binary verification reward of 1, more than 80% answers are correct, which encourages the model to assign a verification score of 1 with increasingly high probability. Within 17 optimization steps, the verification scores for nearly all answers converge to 1, regardless of their correctness. When all answers receive the same verification score, their rewards collapse to the same value, the advantage becomes zero and the learning gradient vanishes. This induces an inescapable local optimum that traps the model in producing uninformative scores and weakens its ability to distinguish correct from incorrect solutions.

Preference Verification Reward. To address this limitation, we propose a preference verification reward that preserves discriminative signals even under severe class imbalance. Instead of using fixed thresholds for verification scores, we introduce *adaptive thresholds* to provide relative ranking supervision within each group. We partition samples into contrastive sets according to answer quality and encourage the policy to assign higher verification scores to better answers and lower scores to worse ones. Formally, for sample i with verification score s_i and answer reward R_i^a, we define the preference verification reward R_i^p as:

$$R_i^p = \frac{1}{\max\big(|\mathcal{C}_i|, 1\big)} \sum_{j \in \mathcal{C}_i} \mathbb{1}\big\{(s_i - s_j)(R_i^a - R_j^a) > 0\big\}, \quad (5)$$

where $\mathcal{C}_i$ denotes the *contrastive set* for sample i and contains samples with different answer qualities. The verification reward R_i^p measures *ranking accuracy*, i.e., the proportion of contrastive pairs with matching verification score and answer quality orderings. For example, when $R_i^a > R_j^a$ (sample i is better), we expect $s_i > s_j$ (higher verification score), and vice versa. The indicator function $\mathbb{1}(s_i - s_j)(R_i^a - R_j^a) > 0$ equals 1 exactly when the ranking of verification scores and answer qualities are consistent.

For *discrete tasks*, we partition each group into positive samples (correct answers) and negative samples (incorrect answers) based on R^a for each sample, then define the contrastive set $\mathcal{C}_i$ for sample i as:

$$\mathcal{C}_i = \{j \in \{1, \ldots, G\} \mid R_j^a \neq R_i^a\}. \quad (6)$$

Each sample is only compared against other samples in the same contrastive set, encouraging the verifier to distinguish correct from incorrect answers. This formulation rewards the model when: (1) correct samples have verification scores above the incorrect samples' average, or (2) incorrect samples have verification scores below the correct samples' average. Rather than treating verification as a coarse binary prediction, our method explicitly reinforces relative quality rankings between samples, thereby substantially enhancing verification capability.

For *continuous tasks*, we regard answers with similar quality as positives and others as negatives. We impose a margin $\gamma > 0$ on quality differences and define the contrastive set for sample i as:

$$\mathcal{C}_i = \{j \in \{1, \ldots, G\} \mid |R_j^a - R_i^a| > \gamma\}. \quad (7)$$

When all rollouts are all same correct or incorrect, all assigned verification rewards are zero, resulting in no gradient updates. This fundamentally addresses the issue where, as training progresses, an increasing proportion of samples receive identical rewards, which would otherwise lead to model outputs converging to binary values of 0 or 1. This preference reward provides dense, contrastive supervision that maximizes *quality-dependent score margins* while accommodating both discrete correctness and continuous paradigms.

Preference verification reward optimizes ranking consistency between the verification score and answer quality rather than the absolute probability scale; because the method outputs relative scores rather than calibrated probabilities, we report ranking metrics that do not depend on probability calibration, namely AUC and AP. In ablation, preference verification reward improves both AUC and AP

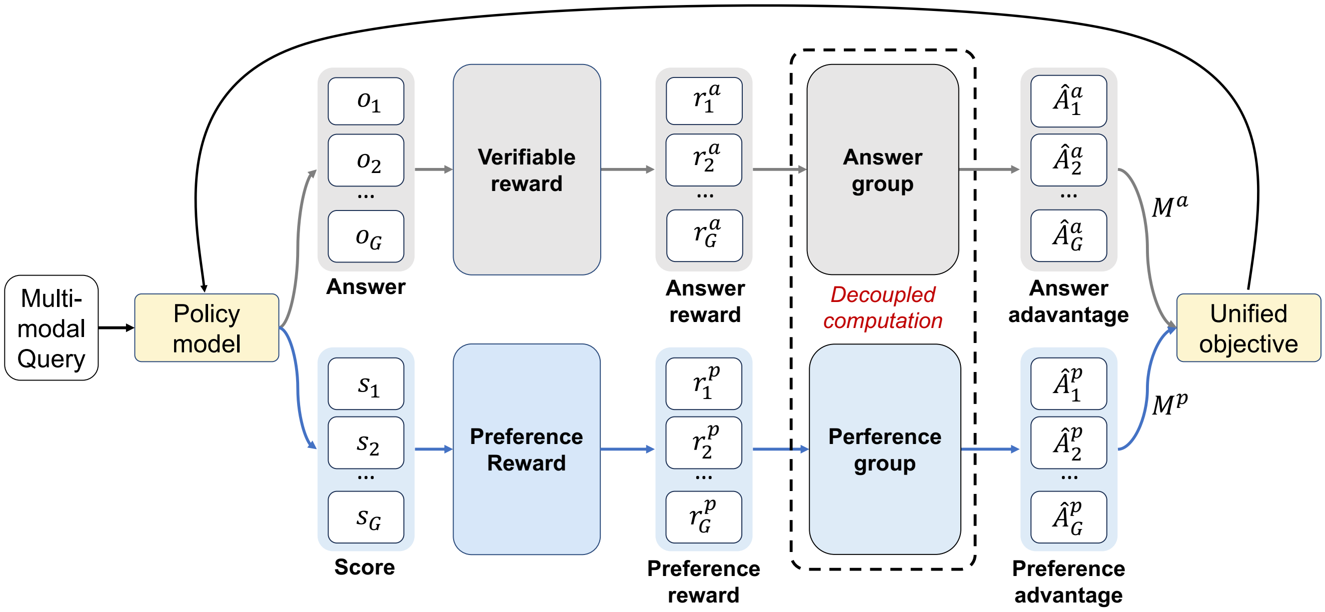


Figure 3. **The framework of ADPO.** Given a multimodal input, our unified policy produces an answer and a self-verification score to rank answer candidates. We design **a preference verification reward** to improve verification capability and **a decoupled optimization mechanism** to enable synergistic optimization of generation and verification. Preference verification reward aligns verification scores with answer correctness by providing relative ranking supervision. Advantage decoupled optimization computes separate advantages for generation and verification, and applies token masks to isolate gradients, thereby preventing reward hacking and reducing gradient interference between the two objectives.

over the binary reward, indicating stronger alignment between scores and answer quality (see Fig. 4).

Discussion. Preference verification reward targets ranking consistency between the verification score and answer quality rather than the absolute scale of predicted probabilities. We therefore use AUC (ROC-AUC) and AP (average precision) to show that better answers receive higher scores. In ablation study, preference verification reward improves AUC and AP up to **0.19** over the binary reward, indicating stronger alignment between verification scores and answer quality (Fig. 4).

3.3. Advantage Decoupled Optimization

Entangled advantage. A straightforward approach to jointly optimize generation and verification is to simply aggregate the answer and verification rewards:

$$R_{\text{total}} = R^a(y, y^*) + R^p(y, y^*, s). \tag{8}$$

where R_{total} denotes the aggregated reward used to compute advantages in the GRPO objective.

However, we found that naively aggregating the answer reward and the verification reward degrades generation capability. The model learns to exploit the objective by producing an obviously incorrect answer while assigning a very low self-verification score to exploit the reward, thereby still achieving a high total reward.

This issue stems from the fact that generation and verification constitute two distinct tasks with different optimization objectives. Specifically, answer rewards favor samples with higher response quality, while verification rewards favor samples with better calibrated verification scores, making their rewards fundamentally incompatible for simple aggregation.

Decoupled advantage. To address this conflict, we decouple the advantage group by reward type and isolate gradients using disjoint token masks. As shown in Fig. 3, we compute separate advantages within each reward group: $\hat{A}^{(a)}$ from answer group and $\hat{A}^{(p)}$ from preference group. We then apply task-specific token masks to prevent gradient interference: M^a covers answer generation tokens (including reasoning when present), while M^p covers only verification score tokens. This design prevents gradient interference.

The unified training objective is:

$$\mathcal{J}(\theta) = M^a \odot \mathcal{J}_\theta(\hat{A}^{(a)}) + M^p \odot \mathcal{J}_\theta(\hat{A}^{(p)}). \tag{9}$$

where $\odot$ denotes element-wise multiplication over tokens and $M^a, M^p \in \{0,1\}^T$ are token-level masks that gate contributions and gradients to their respective regions. By construction, improvements in answer quality are driven only by $\hat{A}^{(a)}$ on generation tokens, while calibration is shaped only by $\hat{A}^{(p)}$ on verification tokens, thereby resolving the gradient conflict in joint optimization.

4. Experiments

We evaluate ADPO on three multimodal domains: Math Reasoning, Visual Grounding, and Mobile Agent tasks. Our experiments demonstrate that ADPO consistently outperforms strong baselines on standard benchmarks, achieving improvements in both task accuracy and self-verification reliability.

4.1. Experimental Setup

Datasets. We evaluate across three domains using standard benchmarks to assess model capabilities. (1) **Multimodal math reasoning**: we train on curated multimodal math reasoning dataset [21] and evaluate in-domain on MathVista [22] and out-of-distribution on MMMU [45], reporting accuracy. (2) **Visual grounding**: we train on RefCOCO [44] and evaluate on ReasonSeg [11], measuring cIoU for referring expression comprehension. (3) **Mobile agents**: we train separately on AndroidControl [13] and GUI Odyssey [23] and evaluate on their official test sets for UI navigation, reporting step success rate (SR).

Baselines. We compare against three primary baselines: (i) GRPO, (ii) GRPO with majority voting, and (iii) ADPO with majority voting. For the verification baseline, we evaluate three judge variants—the base model, a GRPO-trained model as judge, and our ADPO-trained model as judge. For multimodal mathematical tasks, we additionally compare against a baseline where the base model serves as the generator and a specialized reward model fine-tuned on mathematical data acts as the verifier.

Implementation details. Unless otherwise noted, all experiments use a shared configuration: learning rate 1×10^{-6}, batch size 128, group size G=8, GRPO clipping parameter ε=0.2, and KL coefficient β=0.01. For **Multimodal Math Reasoning**, we fine-tune Qwen2-VL-7B [35] for 1,200 steps. For **Visual Grounding** and **Mobile Agent**, we initialize from Qwen2.5-VL-7B [1] and train for 1,200 and 8,000 steps, respectively. During policy rollouts, we decode with temperature T=1.0 and top-p=0.99; at evaluation, we use T=0.2 with the same top-p.

4.2. Main Results

Our evaluation compares three generators: the base model, the GRPO-finetuned model, and the ADPO-finetuned model, each paired with majority voting as verification strategies (Tabs. 11 to 13). We conducted a comprehensive comparison of base, GRPO, and ADPO as both generators and verifiers across three domains (Tab. 5). We report the performance under pass@1, majority voting and best-of-N evaluation protocols ($N \in 4, 8, 12$), and find that employing ADPO as a unified generator and verifier achieves the best performance across all domains under a fixed sampling budget. Notably, this unified approach maintains a pass@1 generation quality comparable to that of the GRPO model.

Table 2. **Performance on MathVista [22] and MMMU [45].** We adopt Qwen2-VL-7B [35] as the base model and use majority voting for both the base and GRPO models. We report accuracy (%) and highlight the best results in **bold**.

Method	MathVista (In-domain)			MMMU (OOD)						
	GVQA	MVQA	**ALL**	ARD	BUS	HEM	HSS	SCI	TEN	**ALL**
Sample 1										
R1-VL-7B [47]	-	-	63.5	-	-	-	-	-	-	-
Base	68.9	48.5	57.9	**67.5**	39.1	49.3	69.0	33.9	36.7	47.1
GRPO	**69.8**	55.7	62.2	65.0	45.9	48.2	68.2	**35.9**	**39.8**	**48.7**
ADPO	68.7	**57.0**	**62.4**	63.1	**46.2**	**50.2**	**71.1**	33.3	35.3	47.7
Sample 4										
MM-Verifier [33]	67.0	53.7	59.8	-	-	-	-	-	-	-
Base	65.7	51.9	58.2	66.7	47.3	50.7	65.8	34.0	38.1	48.6
GRPO	69.8	58.0	63.4	65.8	44.7	50.0	**70.0**	**42.0**	36.7	49.4
ADPO	**71.3**	**59.3**	**64.8**	**68.3**	**48.0**	**52.0**	69.2	39.3	**39.5**	**50.8**
Sample 8										
MM-Verifier [33]	68.5	57.4	62.5	-	-	-	-	-	-	-
Base	68.0	53.3	60.1	**68.3**	50.0	53.3	68.3	32.7	36.7	49.4
GRPO	70.4	56.5	62.9	66.7	48.7	51.3	**74.2**	**42.7**	36.7	51.1
ADPO	**72.2**	**58.9**	**65.0**	65.8	**54.0**	**54.7**	66.7	40.7	**41.0**	**52.1**
Sample 12										
MM-Verifier [33]	70.4	58.7	64.1	-	-	-	-	-	-	-
Base	67.4	55.0	60.7	**69.2**	52.0	50.7	70.8	38.0	36.7	50.7
GRPO	70.7	57.2	63.4	64.2	50.0	51.3	73.3	**43.3**	39.5	51.7
ADPO	**71.7**	**59.8**	**65.3**	67.5	**53.3**	**54.0**	**71.7**	38.7	**40.5**	**52.3**

Table 3. **Performance on ReasonSeg [11].** We use Qwen2.5-VL-7B [1] as the base model.

Method	Short query			Long query			Overall		
	gIoU	cIoU	ACC	gIoU	cIoU	ACC	gIoU	cIoU	ACC
Sample 1									
LISA-7B [11]	47.1	48.5	-	49.2	48.9	-	48.7	48.8	-
SegLLM [38]	-	-	-	-	54.2	-	-	48.4	-
Seg-Zero-7B [18]	-	-	-	-	-	-	57.5	52.0	-
VLM-R1 [32]	-	-	-	-	-	-	-	-	63.1
Base	49.5	53.0	67.0	56.8	57.5	68.5	56.3	57.2	68.4
GRPO	51.8	**55.5**	67.9	59.1	**59.7**	71.3	**58.6**	**59.5**	71.1
ADPO	**51.7**	54.8	**68.0**	**60.2**	59.4	**71.9**	58.1	59.1	**71.7**
Sample 4									
Base	47.8	52.0	66.0	57.3	57.9	69.3	56.7	57.5	69.1
GRPO	**54.5**	**57.0**	**68.0**	58.8	59.5	72.1	58.5	59.4	71.8
ADPO	52.2	55.1	67.0	**61.0**	**61.5**	**73.3**	**60.5**	**61.1**	**72.9**
Sample 8									
Base	47.8	51.4	63.1	57.2	57.8	69.2	56.6	57.4	68.8
GRPO	52.0	55.6	**68.0**	59.2	59.9	72.0	58.7	59.6	71.7
ADPO	**53.2**	**56.0**	67.0	**60.9**	**61.5**	**73.7**	**60.4**	**61.2**	**73.5**
Sample 12									
Base	50.2	53.7	66.0	57.2	57.8	69.3	56.8	57.6	69.1
GRPO	**55.6**	**58.1**	**69.9**	58.8	59.5	72.2	58.6	59.4	72.0
ADPO	53.9	56.2	67.0	**61.3**	**62.0**	**73.6**	**60.9**	**61.6**	**73.2**

Multimodal Math Reasoning. On MathVista (Tab. 11), ADPO's best-of-N performance steadily improves as the sample size increases, achieving 64.8% (N=4), 65.0% (N=8), and 65.3% (N=12). This approach consistently outperforms GRPO with majority voting by +1.4, +2.1, and

Table 4. **Performance on AndroidControl [13] and GUI Odyssey [23].** We adopt Qwen2.5-VL-7B as base model and report type accuracy, grounding accuracy and step success rate (SR).

Generator	**AndroidControl**			**GUI Odyssey**		
	Type	Grounding	**SR**	Type	Grounding	**SR**
Sample 1						
UI-TARS-7B [29]	83.7	-	72.5	86.1	-	67.9
SpiritSight-8B [10]	-	-	68.1	-	-	75.8
AgentCPM-GUI-8B [50]	77.7	-	69.2	90.8	-	75.0
Base	82.2	73.6	61.3	81.1	61.4	52.8
GRPO	**86.0**	**76.9**	**71.0**	93.1	**83.9**	**79.8**
ADPO	85.8	76.2	70.9	**94.2**	82.5	79.7
Sample 4						
Base	76.3	68.1	56.0	76.9	55.3	46.5
GRPO	85.5	77.2	71.0	94.7	83.9	81.3
ADPO	**86.3**	**79.5**	**72.7**	**94.7**	**84.5**	**81.6**
Sample 8						
Base	78.7	68.8	58.3	76.7	55.4	46.6
GRPO	85.6	76.9	70.8	94.6	84.4	81.5
ADPO	**86.4**	**78.7**	**72.7**	**94.8**	**84.7**	**81.7**
Sample 12						
Base	78.9	68.7	58.3	76.9	55.5	46.9
GRPO	85.6	77.4	71.1	**94.5**	84.0	81.1
ADPO	**86.3**	**78.9**	**72.9**	94.4	**84.5**	**81.4**

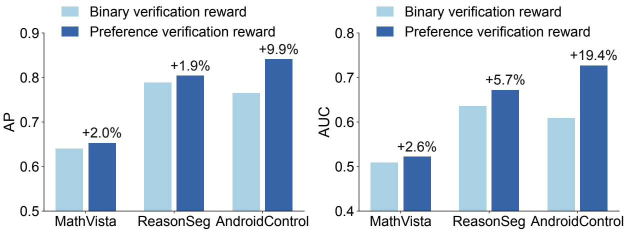
+1.9 percentage points at the respective sample sizes, while also exceeding MM-Verifier by +5.0, +2.5, and +1.2 percentage points at these same budgets. On MMMU, ADPO's pass@1 is competitive at 47.7% (vs. 48.7% for GRPO), while leading several categories. Under best-of-N, ADPO establishes clear advantages, reaching 50.8%, 52.1%, and 52.3% at N=4, 8, and 12, surpassing GRPO (majority) by +1.4, +1.0, and +0.6 points. These results show that ADPO's unified generation–verification training strengthens sample selection and scales effectively with N, yielding higher in-domain accuracy and consistent OOD gains.

Visual Grounding. On ReasonSeg (Tab. 12), ADPO is competitive at pass@1, with overall cIoU 59.1 (vs. 59.5 for GRPO and 57.2 for the base) and the highest overall ACC of 71.7 (vs. 71.1 and 68.4). Under best-of-N, using ADPO as a unified generator–verifier to select candidates yields overall cIoU of 61.1/61.2/61.6 at N=4/8/12, exceeding GRPO (majority voting) by +1.7/ +1.6/+2.2 and the base (majority voting) by +3.6/+3.8/+4.0; overall ACC reaches 72.9/73.5/73.2 at the same budgets. Gains persist on long queries, where ADPO attains pass@1 gIoU/ACC of 60.2/71.9, indicating more robust localization and self-verification that continue to benefit from larger sample budgets.

Mobile Agent. On AndroidControl (Tab. 13), ADPO reaches 70.9% pass@1 SR, comparable to GRPO (71.0%). With best-of-N self-verification, ADPO remains stable at 72.7/72.7/72.9 for N=4/8/12, surpassing GRPO (majority) by +1.7/+1.9/+1.8 and the base model by +16.7/+14.4/+14.6 points. On GUI Odyssey, ADPO attains 79.7% pass@1 (vs. 79.8% for GRPO) and, under best-of-N, improves over GRPO (majority) at N=4/8/12.

Table 5. **Performance of different generator–verifier settings on MathVista [22], ReasonSeg [11] and AndroidControl [13].**

Generator \ **Verifier**	**MathVista**			**ReasonSeg**			**AndroidControl**		
	Base	GRPO	ADPO	Base	GRPO	ADPO	Base	GRPO	ADPO
Sample 4									
Base	55.7	55.5	56.4	57.1	57.7	57.7	52.5	57.7	60.7
GRPO	62.4	62.1	62.0	60.2	59.5	60.9	71.0	70.8	71.2
ADPO	61.5	62.1	**64.8**	59.6	60.3	**61.1**	71.0	72.0	**72.7**
Sample 8									
Base	57.0	56.4	56.5	56.9	57.0	57.9	54.3	61.0	64.7
GRPO	60.7	60.8	60.5	60.4	60.4	61.1	71.0	70.9	71.4
ADPO	62.3	62.3	**65.0**	59.9	60.5	**61.2**	70.8	71.4	**72.7**
Sample 12									
Base	56.9	56.3	55.0	57.4	57.6	57.8	53.6	60.7	64.5
GRPO	62.5	62.5	61.8	59.7	59.6	61.3	71.4	70.9	71.5
ADPO	63.0	63.5	**65.3**	60.7	60.7	**61.6**	71.6	71.9	**72.9**



(a) AP improvement (b) AUC improvement

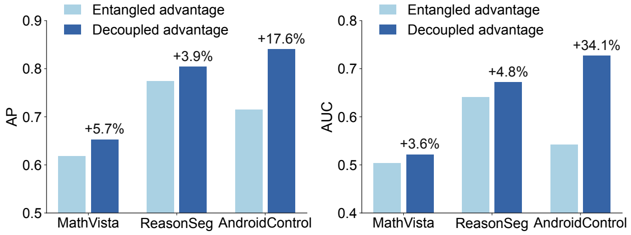
Figure 4. **Ablation of binary verification reward and preference verification reward.**

Discussion. Using ADPO purely as a verifier improves selection across generators and domains (Tab. 5). Paired with the ADPO generator, it achieves the strongest outcomes at all budgets (e.g., MathVista 64.8/65.0/65.3, ReasonSeg 61.1/61.2/61.6, AndroidControl 72.7/72.7/72.9 at N=4/8/12). When judging outputs from base or GRPO generators, ADPO is typically best on ReasonSeg and AndroidControl and competitive on MathVista. These findings support that decoupled advantage training calibrates verification scores well and translates into consistent best-of-N gains while preserving pass@1 performance.

Empirical results demonstrate that ADPO enables effective self-verification without degrading generation quality. ADPO strengthens verification capability without sacrificing pass@1 performance, remaining comparable to GRPO across three domains—MathVista 62.4% vs 62.2% (+0.2%), ReasonSeg 59.1% vs 59.5% (-0.4%), and AndroidControl 70.9% vs 71.0% (-0.1%). The unified training framework yields models that both generate high-quality outputs and reliably select the best among multiple candidates, thereby achieving superior best-of-N performance across diverse multimodal benchmarks under rigorously consistent evaluation protocols.

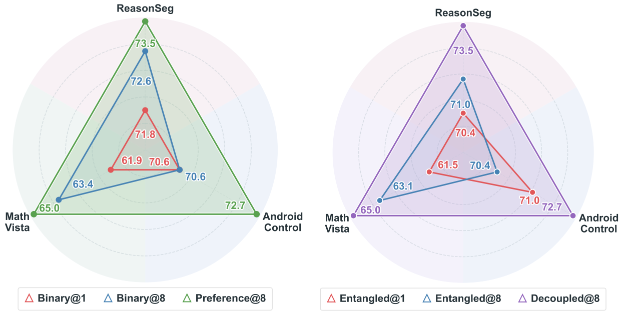
4.3. Ablation Studies

We conduct comprehensive ablation studies to analyze the key components of our decoupled advantage preference optimization framework.



(a) AP improvement (b) AUC improvement

Figure 5. **Ablation of entangled and decoupled advantage.** Entangled and decoupled correspond to models trained with *entangled advantage* in Eq. (8) and *decoupled advantage* in Eq. (9).



(a) Preference verification reward (b) Decoupled advantage

Figure 6. **Ablation of preference verification reward and advantage decoupled optimization.** Binary and preference denote models trained with *binary verification reward* and *preference verification reward*, respectively. The suffix @k indicates evaluation with k samples.

Effect of preference reward. Figure 4 shows the impact of preference reward compared to binary reward across all three domains. The preference formulation consistently improves both task performance and self-verification quality. For mathematical reasoning, we observe +1.6% improvement in best@8 performance (Fig. 6) and +1.3% improvement in average precision (AP). The benefits are even more pronounced for self-verification metrics, with AUC improvements of +1.3%, +3.6%, and +11.8% for math, grounding, and agent tasks respectively. This demonstrates that preference reward provide more stable training signals and better calibrated confidence scores, particularly important under the naturally imbalanced positive/negative distributions in self-verification learning.

Effect of decoupled advantage. Figure 5 illustrates the contribution of our decoupled advantage computation with mutual loss masking compared to simple reward aggregation. Decoupled advantage consistently outperform entangled advantages across all domains, with particularly significant improvements in self-verification quality. For GUI agent tasks, decoupled advantage achieve +2.8% improvement in best@8 performance (Fig. 6) and substantial gains in AUC +18.5 This validates our hypothesis that separating gradient flows between content generation and self-judgment prevents reward hacking and enables more effective optimization of both objectives.

Table 6. **Ablation of the margin γ for preference verification reward on ReasonSeg.**

γ	Short query			Long query			Overall		
	gIoU	cIoU	ACC	gIoU	cIoU	ACC	gIoU	cIoU	ACC
0.025	**53.7**	56.5	**69.9**	58.1	58.9	71.3	57.8	58.8	71.1
0.050	52.6	54.4	63.1	60.2	61.0	73.3	59.8	60.5	72.7
0.100	53.2	56.0	67.0	**60.9**	**61.5**	**73.7**	**60.4**	**61.2**	**73.5**
0.200	53.2	55.7	66.0	59.9	60.7	72.7	59.6	60.4	72.3
0.250	**53.7**	**56.8**	68.9	59.7	60.4	72.5	59.3	60.2	72.3

Table 7. **Comparison of unified and separate verification.** *GRPO:* GRPO post-trained model as generator. *+Major:* majority voting as verifier. *+Judge:* GRPO post-trained model as verifier.

Method	MathVista Acc. ↑	Latency (s) ↓
GRPO+Major	62.9	**2.1**
GRPO+Judge	60.8	5.6
ADPO	**65.0**	2.6

Margin parameter analysis. Table 6 analyzes the effect of margin parameter γ in continuous preference reward computation for visual grounding tasks. We find that $\gamma = 0.1$ provides the optimal balance, achieving 73.5% overall accuracy. Too small margins ($\gamma = 0.025$) may not provide sufficient discrimination between similar quality outputs, while too large margins ($\gamma \geq 0.2$) may be overly restrictive and reduce the density of preference signals. This hyperparameter study confirms the importance of carefully tuning the preference threshold for optimal performance.

Discussion. Table 7 summarizes a comparison on MathVista under a best-of-8 setting (sample size N=8). Our unified policy (ADPO) achieves 65.0% accuracy with a 2.6s latency per sample, improving over GRPO paired with majority voting (62.9%, 2.1s) and GRPO-as-judge (60.8%, 5.6s). This shows that, at the same sampling budget, the unified generator–verifier trained by ADPO delivers stronger verification capability than GRPO while substantially reducing inference time relative to GRPO-as-judge. In practice, this unified policy model lowers overall system complexity and deployment overhead, providing a more cost-effective path to competitive best-of-N performance on MathVista.

5. Conclusion

We introduce Advantage Decoupled Preference Optimization (ADPO), a reinforcement learning framework that trains a unified policy to both generate solutions and verification scores. ADPO addresses three key challenges in parallel test-time scaling: (i) it enables reliable parallel best-of-N selection through unified generator-verifier training; (ii) it replaces binary verification reward with preference verification reward that improve calibration across both discrete and continuous tasks; and (iii) it employs decoupled advantage to separate gradient flows for generation and verification, thereby mitigating reward hacking and gradient interference. Extensive evaluation across five benchmarks spanning three domains: MathVista, MMMU, ReasonSeg, AndroidControl, and GUI Odyssey, which demonstrates that ADPO achieves superior pass@1 performance while consistently improving best-of-N selection and delivering superior self-verification calibration.

References

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 6

[2] Qi Cao, Ruiyi Wang, Ruiyi Zhang, Sai Ashish Somayajula, and Pengtao Xie. Dreamprm: Domain-reweighted process reward model for multimodal reasoning. *arXiv preprint arXiv:2505.20241*, 2025. 3

[3] Yuanlin Chu, Bo Wang, Xiang Liu, Hong Chen, Aiwei Liu, and Xuming Hu. Ssr: Speculative parallel scaling reasoning in test-time. *arXiv preprint arXiv:2505.15340*, 2025. 1

[4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 1, 3

[5] Lingxiao Du, Fanqing Meng, Zongkai Liu, Zhixiang Zhou, Ping Luo, Qiaosheng Zhang, and Wenqi Shao. Mm-prm: Enhancing multimodal mathematical reasoning with scalable step-level supervision. *arXiv preprint arXiv:2505.13427*, 2025. 1, 3

[6] Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, et al. Ui-venus technical report: Building high-performance ui agents with rft. *arXiv preprint arXiv:2508.10833*, 2025. 2

[7] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 1, 2

[8] Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*, 2024. 3

[9] Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025. 2

[10] Zhiyuan Huang, Ziming Cheng, Junting Pan, Zhaohui Hou, and Mingjie Zhan. Spiritsight agent: Advanced gui agent with one look. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29490–29500, 2025. 2, 7

[11] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9579–9589, 2024. 6, 7

[12] Ming Li, Jike Zhong, Shitian Zhao, Yuxiang Lai, Haoquan Zhang, Wang Bill Zhu, and Kaipeng Zhang. Think or not think: A study of explicit thinking in rule-based visual reinforcement fine-tuning. *arXiv preprint arXiv:2503.16188*, 2025. 1, 2

[13] Wei Li, William E Bishop, Alice Li, Christopher Rawles, Folawiyo Campbell-Ajala, Divya Tyamagundlu, and Oriana Riva. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems*, 37: 92130–92154, 2024. 6, 7

[14] Zhenyi Liao, Qingsong Xie, Yanhao Zhang, Zijian Kong, Haonan Lu, Zhenyu Yang, and Zhijie Deng. Improved visual-spatial reasoning via r1-zero-like training. *arXiv preprint arXiv:2504.00883*, 2025. 1, 2

[15] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023. 3

[16] Yujie Lin, Ante Wang, Moye Chen, Jingyao Liu, Hao Liu, Jinsong Su, and Xinyan Xiao. Investigating inference-time scaling for chain of multi-modal thought: A preliminary study. *arXiv preprint arXiv:2502.11514*, 2025. 1

[17] Yuhang Liu, Pengxiang Li, Congkai Xie, Xavier Hu, Xiaotian Han, Shengyu Zhang, Hongxia Yang, and Fei Wu. Infigui-r1: Advancing multimodal gui agents from reactive actors to deliberative reasoners. *arXiv preprint arXiv:2504.14239*, 2025. 2

[18] Yuqi Liu, Bohao Peng, Zhisheng Zhong, Zihao Yue, Fanbin Lu, Bei Yu, and Jiaya Jia. Seg-zero: Reasoning-chain guided segmentation via cognitive reinforcement. *arXiv preprint arXiv:2503.06520*, 2025. 6

[19] Yuqi Liu, Tianyuan Qu, Zhisheng Zhong, Bohao Peng, Shu Liu, Bei Yu, and Jiaya Jia. Visionreasoner: Unified visual perception and reasoning via reinforcement learning. *arXiv preprint arXiv:2505.12081*, 2025. 2

[20] Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning. *arXiv preprint arXiv:2503.01785*, 2025. 2

[21] LMMs-Lab. multimodal-open-r1-8k-verified, 2025. Commit e3c8f3a (2025-01-27). 6

[22] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023. 6, 7

[23] Quanfeng Lu, Wenqi Shao, Zitao Liu, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, Yu Qiao, and Ping Luo. Gui odyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. *arXiv preprint arXiv:2406.08451*, 2024. 6, 7

[24] Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Guanjing Xiong, and Hongsheng Li. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. *arXiv preprint arXiv:2503.21620*, 2025. 2

[25] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332, 2025. 1

[26] OpenAI. Learning to reason with llms, 2024. 1, 2

[27] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022. 3

[28] Yi Peng, Peiyu Wang, Xiaokun Wang, Yichen Wei, Jiangbo Pei, Weijie Qiu, Ai Jian, Yunzhuo Hao, Jiachun Pan, Tianyidan Xie, et al. Skywork r1v: Pioneering multimodal reasoning with chain-of-thought. *arXiv preprint arXiv:2504.05599*, 2025. 2

[29] Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, et al. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv preprint arXiv:2501.12326*, 2025. 2, 7

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 3

[31] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 2, 3

[32] Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615*, 2025. 2, 6

[33] Linzhuang Sun, Hao Liang, Jingxuan Wei, Bihui Yu, Tianpeng Li, Fan Yang, Zenan Zhou, and Wentao Zhang. Mm-verify: Enhancing multimodal reasoning with chain-of-thought verification. *arXiv preprint arXiv:2502.13383*, 2025. 1, 3, 6

[34] Fei Tang, Zhangxuan Gu, Zhengxi Lu, Xuyang Liu, Shuheng Shen, Changhua Meng, Wen Wang, Wenqi Zhang, Yongliang Shen, Weiming Lu, et al. Gui-g^2: Gaussian reward modeling for GUI grounding. *arXiv preprint arXiv:2507.15846*, 2025. 1

[35] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, 2024. 6

[36] Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, et al. Visualprm: An effective process reward model for multimodal reasoning. *arXiv preprint arXiv:2503.10291*, 2025. 3

[37] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022. 1, 2

[38] XuDong Wang, Shaolun Zhang, Shufan Li, Konstantinos Kallidromitis, Kehan Li, Yusuke Kato, Kazuki Kozuka, and Trevor Darrell. Segllm: Multi-round reasoning segmentation. *arXiv preprint arXiv:2410.18923*, 2024. 6

[39] Zili Wang, Tianyu Zhang, Haoli Bai, Lu Hou, Xianzhi Yu, Wulong Liu, Shiming Xiang, and Lei Zhu. Faster and better llms via latency-aware test-time scaling. *arXiv preprint arXiv:2505.19634*, 2025. 1

[40] Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Parathinker: Native parallel thinking as a new paradigm to scale llm test-time compute. *arXiv preprint arXiv:2509.04475*, 2025. 1

[41] Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, et al. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. *arXiv preprint arXiv:2503.10615*, 2025. 2

[42] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023. 1

[43] Runyang You, Yongqi Li, Meng Liu, Wenjie Wang, Liqiang Nie, and Wenjie Li. Parallel test-time scaling for latent reasoning models. *arXiv preprint arXiv:2510.07745*, 2025. 1

[44] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European conference on computer vision*, pages 69–85. Springer, 2016. 6

[45] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024. 6

[46] Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? *arXiv preprint arXiv:2502.12215*, 2025. 1

[47] Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. *arXiv preprint arXiv:2503.12937*, 2025. 2, 6

[48] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative

verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024. 3

[49] Yi-Fan Zhang, Tao Yu, Haochen Tian, Chaoyou Fu, Peiyan Li, Jianshu Zeng, Wulin Xie, Yang Shi, Huanyu Zhang, Junkang Wu, et al. Mm-rlhf: The next step forward in multimodal llm alignment. *arXiv preprint arXiv:2502.10391*, 2025. 3

[50] Zhong Zhang, Yaxi Lu, Yikun Fu, Yupeng Huo, Shenzhi Yang, Yesai Wu, Han Si, Xin Cong, Haotian Chen, Yankai Lin, et al. Agentcpm-gui: Building mobile-use agents with reinforcement fine-tuning. *arXiv preprint arXiv:2506.01391*, 2025. 2, 7

[51] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023. 3

A. Training Details

The key optimization and training hyperparameters for all experiments are summarized in Table 8.For all experiments, we fully fine-tune with freezing vision modules. Unless otherwise specified, we conduct training on a single node with 8 NVIDIA A100 GPUs.

For multimodal math reasoning and visual grounding, we prompt the model first to generate its reasoning within `<think>...</think>`, then produce a final answer in `<answer>...</answer>`, and finally output a verification score in `<score>...</score>`. Reward function consists of accuracy and formatting components. The accuracy component uses the widely adopted Python library math-verify to extract the model's answer and compare it with the ground truth. The format component ensures the correct structure and ordering of the `<think>` and `<answer>` tags, as well as the `<score>` tag.

For the math reasoning domain, we follow the open-r1-multimodal settings. We train on the curated multimodal math reasoning dataset, containing curated multimodal math problems with images and answers and evaluated on MathVista and MMMU. For the visual grounding tasks, we follow the VLM_R1 settings. We train on RefCOCO training set and evaluate on ReasonSeg.

For mobile agent tasks, we prompt the model dirctly to generate the tool call following the Qwen2.5-VL mobile use format in `<tool_call>...</tool_call>` and then output the verification score in `<score>...</score>`. Each sample contains a resized high-resolution screenshot, a natural-language goal together with a history of previous actions (`pre_act`), and a ground-truth tool call of the form `mobile_use(...)` specifying the action type (click, swipe, type, open, system_button, etc.) and its parameters (coordinates, text, button type, and so on).

The answer reward comprises both format and accuracy components. The format component ensures the correct structure and ordering of the `<tool_call>` and `<score>` tags. The accuracy component follows a stepwise verification process. First, we validate the action type against the ground truth, granting a $+1$ reward for a match. Subsequently, we evaluate the action parameters. For example, a click action earns an additional $+1$ reward if the Euclidean distance between the predicted coordinates and the ground truth label is less than $0.14 \times$ screen diagonal.

Hyperparameter	Value
num_generations	8
per_device_train_batch_size	8
gradient_accumulation_steps	2
torch_dtype	bfloat16
data_seed	42
gradient_checkpointing	true
attn_implementation	flash_attention_2
learning_rate	1e-6
β	0.01

Table 8. Hyperparameter settings used in the training experiments.

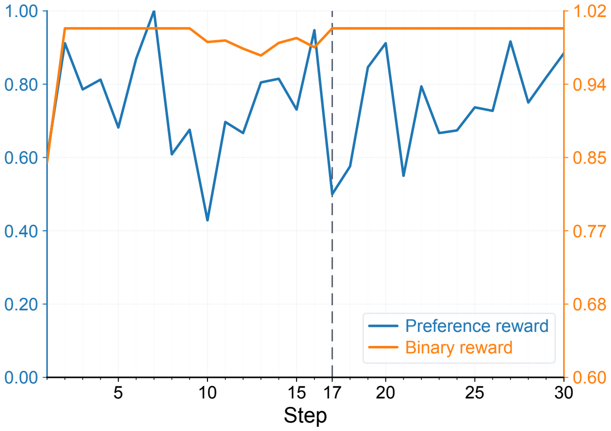


Figure 7. **Analysis of reward signal distribution during training.** The **Blue** line shows the proportion of correct answers among responses with preference verification reward = 1. The **Orange** line shows the proportion of correct answers among responses with binary verification reward = 1.

B. Ablation and Analysis

To better understand the effect of our preference verification reward in the mobile agent setting, we analyze training dynamics on the AndroidControl. As the policy improves, the binary verification reward quickly becomes dominated by correct trajectories, leading to a severe class imbalance where almost all reward-1 samples correspond to already-correct actions. As shown in Fig. 7, the subset of rollouts that receive binary reward $= 1$ rapidly collapses to near-perfect accuracy, providing little signal to separate moderate-quality actions from the very best ones. In contrast, the preference verification reward maintains a more informative mixture of correct and incorrect rollouts among its positive signals, preserving contrastive supervision even when overall task success is high.

This difference in supervision is reflected in the learned verification scores. Figure 8 compares score distributions for models trained with binary versus preference verification reward on AndroidControl. The binary reward model tends to output highly discretized scores concentrated near the extremes, consistent with the imbalanced 0/1 supervision. In contrast, the preference reward model produces a smoother and more diverse score distribution, assigning fine-grained scores that better reflect relative action quality. This diversity is crucial for best-of-N selection, where ranking among multiple candidate trajectories matters more than predicting a single calibrated probability.

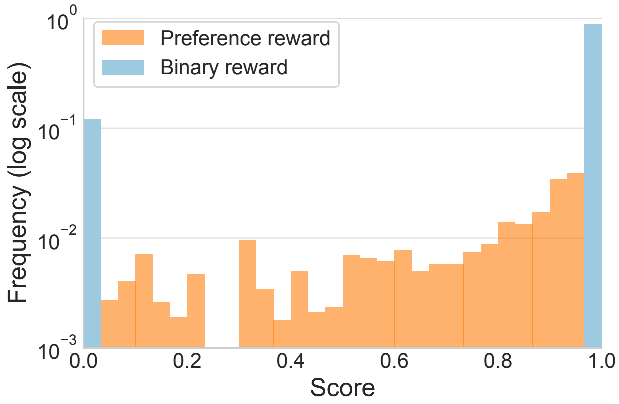


Figure 8. **Comparison of score distributions between models trained with Binary Reward and Preference Reward.** While the binary reward model tends to output discrete scores, the preference reward model produces a more diverse distribution.

Tables 9 and 10 provide more fine-grained ablation results complementing the main figures. For MathVista and AndroidControl, we report task performance using answer accuracy under pass@1 and best@8, while for ReasonSeg we use acc@(IoU > 0.5) under the same best-of-k protocol; all three domains share the same self-verification metrics AUC and AP. In the reward ablation, replacing the binary verification reward with our preference verification reward consistently strengthens self-verification, with especially large gains in AUC and AP on AndroidControl, indicating a much more reliable verifier under class imbalance. In the advantage ablation, decoupled advantage improves both pass@1 and best@8 performance across domains compared to entangled advantage, while also enhancing verification metrics, showing that separating generation and verification advantages benefits both task accuracy and ranking quality.

Table 9. **Ablation of Preference reward.** Replacing the binary answer reward with our *preference reward* consistently strengthens self-verification (↑AUC/AP) and improves best of N selection performance on *Math*, *Grounding*, and *GUI Agent*.

Domain	**Method**	**Performance**		**Verification**	
		pass@1	**best@8**	**AUC**	**AP**
MathVista	Binary verification reward	61.9	63.4	0.509	0.640
	Preference verification reward	**62.4**	**65.0**	**0.522**	**0.653**
ReasonSeg	Binary verification reward	**71.8**	72.6	0.636	0.789
	Preference verification reward	71.7	**73.5**	**0.672**	**0.804**
AndroidControl	Binary verification reward	70.6	70.6	0.609	0.765
	Preference verification reward	**70.9**	**72.7**	**0.727**	**0.841**

Table 10. **Ablation study on decoupled advantages.** Our advantage decoupled optimization consistently outperforms entangled advantage in both task performance and solution verification across mathematical reasoning, grounding, and GUI agent tasks.

Domain	**Method**	**Performance**		**Verification**	
		pass@1	**best@8**	**AUC**	**AP**
MathVista	Entangled Advantage	61.5	63.1	50.4	61.8
	Decoupled Advantage	**62.4**	**65.0**	**52.2**	**65.3**
ReasonSeg	Entangled Advantage	70.4	71.0	0.641	0.774
	Decoupled Advantage	**71.7**	**73.5**	**0.672**	**0.804**
AndroidControl	Entangled Advantage	71.0	70.4	0.542	0.715
	Decoupled Advantage	**70.9**	**72.7**	**0.727**	**0.841**

C. Extended Experimental Results

We conduct extensive experiments by enumerating all combinations of base, GRPO, and ADPO as both generators and verifiers across math reasoning, visual grounding, and mobile agent benchmarks (Tabs. 11 to 13). These results show that ADPO matches GRPO in pass@1 generation quality while providing substantially stronger verification for best-of-N selection, and that this unified training only incurs about 10% additional training time compared to GRPO with exactly the same data. In contrast to traditional pipelines that train a separate reward model with extra preference data, ADPO jointly learns generation and verification within a single policy, avoiding additional data collection and separate training runs and thus reducing both training and deployment cost.

Table 11 reports extended MathVista and MMMU results. With single-sample decoding (Sample 1), GRPO and ADPO generators reach similar pass@1 accuracy (62.2% vs 62.4% on MathVista; 48.7% vs 47.7% on MMMU), showing ADPO matches GRPO. Under best-of-12 decoding (Sample 12), pairing the ADPO generator and verifier increases MathVista accuracy from 62.5% to 65.3% and MMMU from 50.8% to 52.3%, indicating a stronger verifier under the same sampling budget.

Table 12 gives analogous ReasonSeg visual grounding results across short, long, and overall queries with gIoU, cIoU, and acc@(IoU > 0.5). On Sample 1, overall accuracy rises from 68.4% for the base generator to 71.1% for GRPO and 71.7% for ADPO. For best-of-12 (Sample 12), using ADPO for both generation and verification attains 73.2% overall, outperforming GRPO–GRPO (71.7%) and the base generator with majority voting (69.1%), confirming ADPO as the strongest box-ranking verifier.

Table 13 summarizes mobile agent results on AndroidControl and GUI Odyssey using step success rate (SR) and related metrics. On Sample 1, GRPO and ADPO generators achieve nearly identical SR (71.0% vs 70.9% on AndroidControl; 79.8% vs 79.7% on GUI Odyssey), indicating no pass@1 degradation. Under best-of-8 (Sample 8), the ADPO generator–verifier pair improves SR from 70.9% to 72.7% on AndroidControl and from 80.6% to 81.7% on GUI Odyssey, while the base generator with majority voting lags behind (58.3% and 46.6%), quantifying ADPO's stronger verification despite identical training data.

Table 11. **Evaluation results on multimodal math reasoning benchmarks.** Rows correspond to generators and columns correspond to verifiers. We use Qwen2-VL-7B as the base model, with GRPO and ADPO representing the finetuned models. Majority voting serves as the verifier baseline. Values are accuracy (%). GVQA: General VQA; MVQA: Math Target VQA; ARD: Art & Design; BUS: Business; HEM: Health & Medicine; HSS: Human & Social Science; SCI: Science; TEN: Technology & Engineering.

Generator	**Verifier**	**MathVista (In-domain)**			**MMMU (OOD)**						
		GVQA	MVQA	**ALL**	ARD	BUS	HEM	HSS	SCI	TEN	**ALL**
Sample 1											
Base	✗	68.9	48.5	57.9	67.5	39.1	49.3	69.0	33.9	36.7	47.1
GRPO	✗	69.8	55.7	62.2	65.0	45.9	48.2	68.2	35.9	39.8	48.7
ADPO	✗	68.7	57.0	62.4	63.1	46.2	50.2	71.1	33.3	35.3	47.7
Sample 4											
Base	Major	65.7	51.9	58.2	66.7	47.3	50.7	65.8	34.0	38.1	48.6
	Base	63.9	48.7	55.7	60.0	44.0	50.7	60.8	32.7	33.8	45.2
	GRPO	63.3	48.9	55.5	61.7	43.3	50.0	63.3	30.0	36.7	45.8
	ADPO	63.3	50.6	56.4	66.7	49.3	53.3	70.8	34.0	37.6	49.9
GRPO	Major	69.8	58.0	63.4	65.8	44.7	50.0	70.0	42.0	36.7	49.4
	Base	70.2	55.7	62.4	62.5	44.0	50.0	70.8	40.0	39.5	49.3
	GRPO	69.6	55.7	62.1	62.5	44.0	50.0	70.8	40.0	39.5	49.9
	ADPO	69.6	55.6	62.0	64.2	46.7	50.7	71.7	39.3	39.5	50.1
ADPO	Major	71.7	56.1	63.3	66.7	48.0	52.7	70.0	39.3	39.0	50.7
	Base	68.5	55.6	61.5	64.2	44.0	52.7	67.5	36.7	36.7	48.3
	GRPO	68.3	56.9	62.1	65.0	44.0	52.0	68.3	40.7	36.7	49.1
	ADPO	71.3	59.3	64.8	68.3	48.0	52.0	69.2	39.3	39.5	50.8
Sample 8											
Base	Major	68.0	53.3	60.1	68.3	50.0	53.3	68.3	32.7	36.7	49.4
	Base	63.0	51.9	57.0	65.0	40.7	48.0	65.0	32.0	32.4	45.0
	GRPO	62.8	50.9	56.4	65.8	40.0	49.3	65.8	32.7	37.1	46.6
	ADPO	63.5	50.6	56.5	67.5	48.7	54.0	71.7	36.0	41.0	51.2
GRPO	Major	70.4	56.5	62.9	66.7	48.7	51.3	74.2	42.7	36.7	51.1
	Base	67.6	55.0	60.7	62.5	47.3	51.3	72.5	38.7	37.6	49.7
	GRPO	67.6	55.0	60.8	62.5	46.7	50.0	70.0	39.3	38.6	49.3
	ADPO	67.6	54.4	60.5	63.3	46.7	53.3	68.3	38.7	39.0	49.8
ADPO	Major	71.1	58.0	64.0	65.0	49.3	56.7	71.7	38.7	40.5	51.8
	Base	70.0	55.7	62.3	63.3	52.0	53.3	66.7	42.7	41.0	51.6
	GRPO	69.8	55.9	62.3	63.3	52.7	54.7	65.8	42.0	39.0	51.2
	ADPO	72.2	58.9	65.0	65.8	54.0	54.7	66.7	40.7	41.0	52.1
Sample 12											
Base	Major	67.4	55.0	60.7	69.2	52.0	50.7	70.8	38.0	36.7	50.7
	Base	63.7	51.1	56.9	59.2	47.3	51.3	64.2	30.0	33.8	45.8
	GRPO	62.4	51.1	56.3	58.3	45.3	49.3	63.3	30.0	35.2	45.2
	ADPO	62.6	48.5	55.0	65.0	52.7	53.3	70.0	40.0	35.2	50.6
GRPO	Major	70.7	57.2	63.4	64.2	50.0	51.3	73.3	43.3	39.5	51.7
	Base	70.0	56.3	62.6	63.3	48.0	54.0	68.3	42.7	41.0	51.2
	GRPO	69.3	56.7	62.5	63.3	48.7	52.7	67.5	42.0	40.5	50.8
	ADPO	69.6	55.2	61.8	64.2	48.0	52.7	69.2	43.3	41.0	51.3
ADPO	Major	72.0	58.3	64.6	65.8	50.0	53.3	75.0	36.7	39.0	51.2
	Base	70.7	56.5	63.0	62.5	52.0	54.7	70.0	41.3	41.4	52.0
	GRPO	71.3	56.9	63.5	63.3	53.3	52.7	70.8	41.3	43.3	52.6
	ADPO	71.7	59.8	65.3	67.5	53.3	54.0	71.7	38.7	40.5	52.3

Table 12. **Evaluation results on image grounding benchmarks.** Rows correspond to generators and columns correspond to verifiers. We use Qwen2.5-VL-7B as the base model, with GRPO and ADPO representing the finetuned models. Majority voting serves as the verifier baseline. Models are trained on RefCOCO training set and tested on ReasonSeg (out-of-domain).

Generator	Verifier	Short query			Long query			Overall		
		gIoU	cIoU	ACC	gIoU	cIoU	ACC	gIoU	cIoU	ACC
Sample 1										
Base	✗	49.5	53.0	67.0	56.8	57.5	68.5	56.3	57.2	68.4
GRPO	✗	51.8	55.5	67.9	59.1	59.7	71.3	58.6	59.5	71.1
ADPO	✗	51.7	54.8	68.0	60.2	59.4	71.9	58.1	59.1	71.7
Sample 4										
Base	Major	47.8	52.0	66.0	57.3	57.9	69.3	56.7	57.5	69.1
	Base	47.2	51.4	66.0	56.9	57.5	68.6	56.3	57.1	68.4
	GRPO	49.6	53.2	67.0	57.4	57.9	68.7	56.9	57.7	68.6
	ADPO	50.4	53.5	68.0	57.3	57.9	69.1	56.9	57.7	69.1
GRPO	Major	54.5	57.0	68.0	58.8	59.5	72.1	58.5	59.4	71.8
	Base	55.2	57.5	68.0	59.7	60.3	72.9	59.4	60.2	72.6
	GRPO	53.4	56.4	68.9	59.1	59.7	72.0	58.8	59.5	71.8
	ADPO	55.1	57.8	68.0	60.5	61.1	73.4	60.2	60.9	73.1
ADPO	Major	52.7	55.3	67.0	60.1	60.5	72.0	59.6	60.2	71.7
	Base	51.2	54.1	66.0	59.3	59.9	71.7	58.8	59.6	71.4
	GRPO	51.0	54.3	67.0	60.0	60.7	72.7	59.4	60.3	72.4
	ADPO	52.2	55.1	67.0	61.0	61.5	73.3	60.5	61.1	72.9
Sample 8										
Base	Major	47.8	51.4	63.1	57.2	57.8	69.2	56.6	57.4	68.8
	Base	47.9	51.4	62.1	56.6	57.2	68.1	56.1	56.9	67.7
	GRPO	47.1	50.5	62.1	56.8	57.5	68.4	56.2	57.0	68.0
	ADPO	47.4	50.9	61.2	57.7	58.4	69.4	57.1	57.9	68.9
GRPO	Major	52.0	55.6	68.0	59.2	59.9	72.0	58.7	59.6	71.7
	Base	51.7	55.0	67.0	60.1	60.7	72.4	59.6	60.4	72.1
	GRPO	51.5	55.3	68.0	60.0	60.7	72.4	59.5	60.4	72.1
	ADPO	54.4	57.2	67.0	60.6	61.3	73.8	60.2	61.1	73.3
ADPO	Major	53.2	56.1	67.0	58.8	59.4	71.3	58.5	59.2	71.0
	Base	52.9	55.4	67.0	59.6	60.2	72.0	59.2	59.9	71.7
	GRPO	55.6	57.8	68.9	59.9	60.6	72.9	59.6	60.5	72.7
	ADPO	53.2	56.0	67.0	60.9	61.5	73.7	60.4	61.2	73.5
Sample 12										
Base	Major	50.2	53.7	66.0	57.2	57.8	69.3	56.8	57.6	69.1
	Base	49.5	53.3	68.0	56.9	57.6	68.9	56.5	57.4	68.8
	GRPO	50.0	53.1	66.0	57.1	57.9	69.1	56.7	57.6	68.9
	ADPO	49.8	52.6	65.1	57.6	58.2	69.2	57.1	57.8	68.9
GRPO	Major	55.6	58.1	69.9	58.8	59.5	72.2	58.6	59.4	72.0
	Base	48.8	52.4	65.1	59.6	60.2	72.5	58.9	59.7	72.1
	GRPO	53.2	55.8	68.0	59.1	59.9	71.9	58.8	59.6	71.7
	ADPO	54.1	56.7	68.9	60.9	61.6	74.0	60.5	61.3	73.7
ADPO	Major	53.3	55.3	66.0	59.3	60.0	71.8	58.9	59.8	71.5
	Base	55.0	57.4	68.9	60.2	60.9	72.1	59.9	60.7	71.9
	GRPO	53.8	56.5	68.9	60.3	61.0	72.4	59.9	60.7	72.1
	ADPO	53.9	56.2	67.0	61.3	62.0	73.6	60.9	61.6	73.2

Table 13. **Evaluation results on mobile agent benchmarks.** Rows correspond to generators and columns correspond to verifiers. We use Qwen2.5-VL-7B as the base model, with GRPO and ADPO representing the finetuned models.

Generator	**Verifier**	**AndroidControl**			**GUI Odyssey**		
		Type	Grounding	**SR**	Type	Grounding	**SR**
Sample 1							
Base	✗	82.2	73.6	61.3	81.1	61.4	52.8
GRPO	✗	86.0	76.9	71.0	93.1	83.9	79.8
ADPO	✗	85.8	76.2	70.9	94.2	82.5	79.7
Sample 4							
Base	Major	76.3	68.1	56.0	76.9	55.3	46.5
	Base	72.1	67.8	52.5	75.3	55.6	45.2
	GRPO	74.9	71.4	57.7	75.3	55.2	45.3
	ADPO	76.4	74.5	60.7	75.1	55.7	45.6
GRPO	Major	85.5	77.2	71.0	94.7	83.9	81.3
	Base	85.4	77.2	71.0	94.3	83.8	81.0
	GRPO	85.4	77.3	70.8	94.4	83.7	80.7
	ADPO	85.6	77.7	71.2	94.5	84.0	81.4
ADPO	Major	86.6	77.1	71.6	93.9	83.5	79.8
	Base	86.4	76.4	71.0	94.7	84.2	81.2
	GRPO	86.4	77.9	72.0	94.7	84.2	81.1
	ADPO	86.3	79.5	72.7	94.7	84.5	81.6
Sample 8							
Base	Major	78.7	68.8	58.3	76.7	55.4	46.6
	Base	73.9	68.4	54.3	75.1	55.2	44.9
	GRPO	77.3	73.4	61.0	74.4	54.3	44.5
	ADPO	79.7	76.5	64.7	73.9	54.6	44.6
GRPO	Major	85.6	76.9	70.8	94.6	84.4	81.5
	Base	85.6	77.1	71.0	93.7	84.4	80.7
	GRPO	85.4	77.4	70.9	93.7	84.4	80.6
	ADPO	85.6	77.7	71.4	93.9	84.8	81.2
ADPO	Major	86.5	76.4	71.3	94.8	84.0	80.9
	Base	86.1	76.2	70.8	95.1	84.6	81.6
	GRPO	85.8	77.7	71.4	94.9	84.4	81.4
	ADPO	86.4	78.7	72.7	94.8	84.7	81.7
Sample 12							
Base	Major	78.9	68.7	58.3	76.9	55.5	46.9
	Base	73.4	67.9	53.6	74.5	55.5	44.6
	GRPO	76.8	73.2	60.7	73.5	54.3	44.0
	ADPO	79.2	76.7	64.5	72.9	53.8	43.6
GRPO	Major	85.6	77.4	71.1	94.5	84.0	81.1
	Base	85.6	78.0	71.4	93.0	84.0	79.9
	GRPO	85.4	77.5	70.9	93.1	83.9	79.7
	ADPO	85.7	77.9	71.5	93.2	84.2	80.3
ADPO	Major	86.6	76.7	71.9	94.4	83.7	80.5
	Base	86.5	76.3	71.6	94.8	84.6	81.5
	GRPO	85.4	78.6	71.9	94.6	84.1	81.1
	ADPO	86.3	78.9	72.9	94.4	84.5	81.4