SAW: STAGE-AWARE WEIGHTING FOR MULTI-REWARD REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-reward reinforcement learning for language models combines objectives that can improve at different rates. Fixed aggregation does not adapt their weights as training progresses, while gradient-based weighting introduces additional gradient computation. We propose Stage-Aware Weighting (SAW), a lightweight method that adjusts objective weights using coefficients of variation computed from the current rollout batch. SAW operates at the reward level in GRPO and the advantage level in GDPO. Its weighting rule requires neither additional backward passes nor statistics from previous batches. We evaluate SAW on tool calling and summarization. Across six BFCL-v4 settings, SAW improves overall accuracy over default fixed aggregation by 1.09–7.37 percentage points and achieves higher Overall point estimates than the evaluated gradient-based weighting baseline in all six settings. In the 1.5B two-reward setting, SAW also exceeds all three tested fixed ratios in Overall. Summarization gains persist under two evaluation judges not used for training. These results support current-batch reward statistics as a practical and computationally lightweight basis for adaptive multi-reward optimization.

1 INTRODUCTION

Reinforcement learning for language models increasingly combines several requirements in one policy: a tool-using assistant should select the correct function, produce a valid call, and control reasoning length; a summarizer should preserve content while remaining concise. These requirements are often represented by separate reward functions and combined with fixed coefficients. Yet the corresponding reward distributions change at different rates. A weighting that is appropriate early in training can therefore become poorly matched to a later stage.

Group Relative Policy Optimization (GRPO) normalizes an aggregate reward within each response group (Shao et al., 2024), whereas GDPO normalizes each reward dimension before combining advantages (Liu et al., 2026). These mechanisms address how numerical scales enter optimization. They do not by themselves determine how an objective's coefficient should change as its reward distribution evolves. Gradient-based dynamic weighting offers one solution (Lu et al., 2026), but requires additional per-objective gradient computation.

CoV-Weighting uses temporal loss-ratio statistics in supervised learning (Groenendijk et al., 2020). We use reward statistics from the current rollout batch. SAW weights an objective by its coefficient of variation (CV): standard deviation relative to mean reward above a specified lower bound. For binary success rewards, CV decreases with success probability away from the numerical boundary. This gives the statistic a directional interpretation that standard deviation and entropy lack: it distinguishes a rarely achieved objective from a frequently achieved one even when their absolute variances coincide.

Figure 1 summarizes the procedure. We compute the weights once per batch and apply them before reward aggregation in GRPO or after per-dimension normalization in GDPO. The rule is stateless, requires no learned controller, and adds no per-objective backward pass. Its intended use is to adapt allocation when reward dispersion tracks remaining learning opportunities; this is a testable modeling assumption rather than a guarantee that every high-CV objective is useful.

Our contributions are threefold. First, we introduce a stateless weighting rule based on current-batch reward dispersion, integrated at the reward level in GRPO and the advantage level in GDPO.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

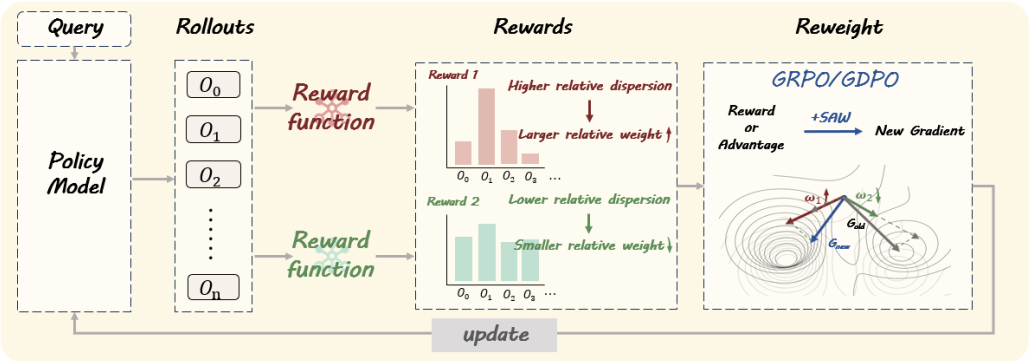


Figure 1: Overview of SAW. Current rollouts are scored along multiple reward dimensions, and their coefficients of variation determine adaptive reward- or advantage-level weights. The landscape illustrates the intended regime in which relative dispersion reflects useful response differences; it is a schematic, not a measured gradient trajectory.

Second, we characterize its binary-reward behavior and analyze how aggregation and normalization affect residual reward variation. Third, we evaluate the rule on tool calling and summarization, with aggregate gains across two model scales, both optimizer families, and two- and three-reward configurations at low measured per-step overhead.

2 PRELIMINARIES

Let $i \in \{1, \dots, B\}$ index prompts, $j \in \{1, \dots, G\}$ the responses to each prompt, and $k \in \{1, \dots, K\}$ the reward dimensions. The current batch contains $N = BG$ scored responses with rewards r_{ijk}. We denote fixed task priorities by $\alpha_k \geq 0$ and adaptive coefficients by w_k.

For group rewards $u_{i,:}$, define stabilized group normalization

$$\mathcal{Z}_g(u)_{ij} = \frac{u_{ij} - \bar{u}_i}{s_i(u) + \epsilon_g}, \qquad s_i(u)^2 = (G-1)^{-1} \sum_j (u_{ij} - \bar{u}_i)^2. \tag{1}$$

Let $\mathcal{Z}_b$ denote whitening over all valid response tokens after broadcasting each response-level value; its statistics are therefore length-weighted (Appendix B). GRPO aggregates rewards before group normalization, while GDPO normalizes each dimension separately and then combines its advantages:

$$A_{ij}^{\mathrm{GRPO}} = \mathcal{Z}_g\left(\sum_k \alpha_k w_k r_{::k} \right)_{ij}, \tag{2}$$

$$a_{ijk} = \mathcal{Z}_g(r_{::k})_{ij}, \qquad A_{ij}^{\mathrm{GDPO}} = \mathcal{Z}_b\left(\sum_k \alpha_k w_k a_{::k} \right)_{ij}. \tag{3}$$

The equal-weight baseline sets $\alpha_k = w_k = 1$ and uses the same normalization. Both advantages enter the usual clipped policy objective; SAW changes their construction while leaving the policy-update rule unchanged. A constant reward within a prompt group contributes zero centered advantage.

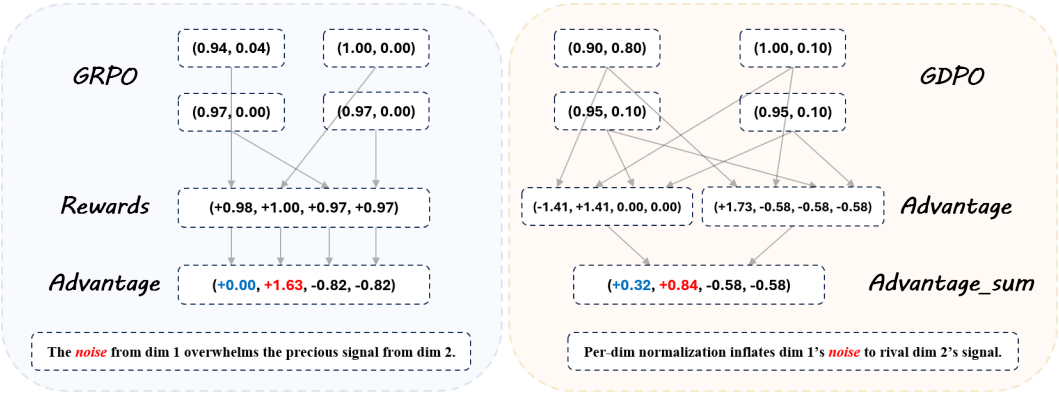


Figure 2: A four-response illustration of two aggregation effects. Here, dimension 1 is stipulated to contain nuisance variation and dimension 2 the desired distinction. Reward summation can cancel this distinction (left), while per-dimension normalization can amplify the nuisance variation (right). Illustrative values use population standard deviations. The noise interpretation is an assumption of this example.

3 Motivation: Asynchronous Reward Learning

3.1 Different Reward Dimensions Evolve at Different Rates

Figure 3 shows a representative two-reward ToolRL training run under equal-weight GRPO. Correctness improves rapidly before format begins its sharp rise. The associated CV trajectories also differ: correctness CV falls earlier, while format retains larger relative dispersion during the early stage. A fixed mixture keeps their coefficients unchanged despite these changing distributions.

The relevant distinction is not simply a larger reward scale. Two objectives can have similar absolute variation yet substantially different mean progress. A statistic relative to the reward origin can capture this distinction. For a binary success reward $R \sim \mathrm{Bernoulli}(p)$, with zero lower bound and without a mean stabilizer,

$$\mathrm{CV}(R) = \sqrt{\frac{1-p}{p}}, \qquad \frac{d\ \mathrm{CV}(R)}{dp} = -\frac{1}{2p^{3/2}\sqrt{1-p}} < 0, \quad 0 < p < 1. \tag{4}$$

In contrast, standard deviation $\sqrt{p(1-p)}$ and binary entropy have the same value at p and $1-p$. For binary rewards, CV is therefore a directional transform of success probability. Its use assumes that further success is learnable; finite-batch and stabilization effects are analyzed in Appendix A.

3.2 How Residual Variation Enters the Update

Figure 2 isolates the aggregation mechanism using four responses. In the left panel, the first objective is near one and the second distinguishes only the first response. If variation in the first objective is irrelevant to the desired distinction, summing rewards cancels the first response's advantage and gives the second response the highest advantage. In the right panel, per-dimension normalization rescales small variation in the first objective to approximately unit variance, making it comparable to the second objective's signal. Their sum again ranks the second response above the first. The GDPO panel shows the sum before final token-level whitening, which preserves this ranking.

This example motivates downweighting a dimension when its remaining variation is mostly nuisance, while preserving a dimension that differentiates useful responses. CV supplies an inexpensive proxy for this allocation in the intended regime. It is not a general noise detector: reward semantics and within-prompt variation still determine whether a statistical difference supports learning. The empirical question is whether this simple proxy improves optimization across tasks and aggregation mechanisms.

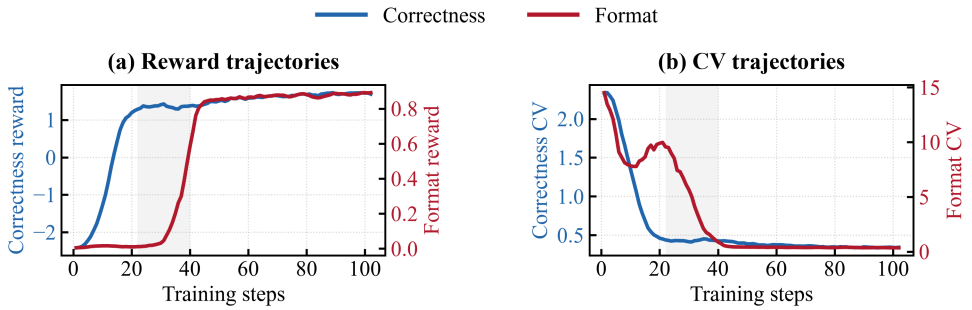


Figure 3: Two-reward ToolRL training dynamics under equal-weight GRPO. Correctness improves before the sharp increase in format reward, while format retains a higher CV during the shaded interval (steps 22–40). Curves show five-step centered moving averages over 102 matched training steps. Correctness and format use separate y-axis scales in both panels.

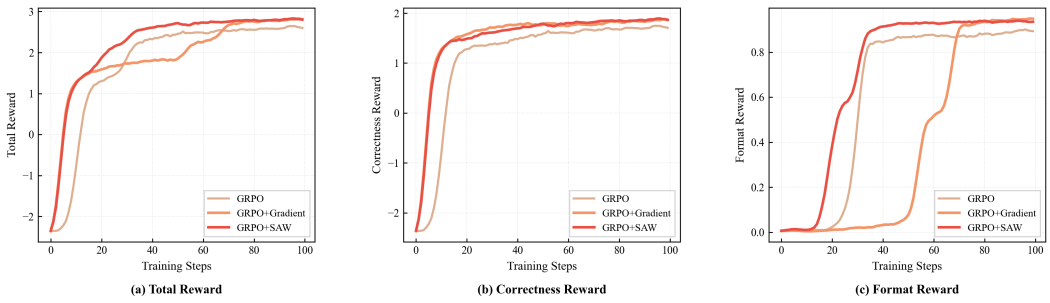


Figure 4: Two-reward ToolRL training trajectories with Qwen2.5-1.5B-Instruct. In the displayed runs, GRPO+SAW reaches high format reward earlier, while terminal correctness rewards remain close across methods. These curves characterize training dynamics separately from the BFCL evaluation in Table 1.

4 Methodology: Stage-Aware Dynamic Weighting

4.1 Current-batch relative dispersion

For each reward dimension, choose a fixed lower bound ℓ_k and form nonnegative scores $x_{ijk} = r_{ijk} - \ell_k$. Compute their current-batch mean, standard deviation, and CV:

$$m_k = \frac{1}{N}\sum_{i,j} x_{ijk}, \qquad s_k^2 = \frac{1}{N-1}\sum_{i,j}(x_{ijk} - m_k)^2, \qquad c_k = \frac{s_k}{m_k + \epsilon_k}. \tag{5}$$

These statistics weight responses equally across prompts, providing one coefficient per objective and batch. Given $C = \sum_k c_k > 0$, use

$$w_k = K\frac{c_k}{C + \epsilon_w} \quad \text{for both GRPO and GDPO.} \tag{6}$$

If $C = 0$, set $w_k = 1$ for all objectives. We use $\epsilon_k = \epsilon_w = 10^{-8}$. The factor K sets the coefficient scale; a common positive scale cancels under exact advantage normalization. The weights are recomputed at each step, without moving averages or cross-step state.

4.2 Integration with GRPO and GDPO

Reward-level weighting. For GRPO, insert the coefficients into Eq. 2 before aggregating rewards. This changes the relative contribution of each dimension to the scalar reward whose within-group differences define the advantage.

Algorithm 1 SAW for a scored rollout batch

Require: Rewards $r \in \mathbb{R}^{B\times G\times K}$, response masks, bounds ℓ_k, stabilizers ϵ_k, ϵ_w
1: **for** $k = 1, \ldots, K$ **do**
2: Compute $x_{ijk} = r_{ijk} - \ell_k$, m_k, s_k, and c_k by Eq. 5.
3: **end for**
4: $C \leftarrow \sum_k c_k$.
5: **if** $C = 0$ **then**
6: Set $w_k \leftarrow 1$ for all k.
7: **else**
8: Set $w_k \leftarrow Kc_k/(C + \epsilon_w)$ for all k.
9: **end if**
10: Construct advantages with Eq. 2 or Eq. 3 and update the policy.

Advantage-level weighting. For GDPO, normalize each reward within its response group, broadcast the per-dimension advantages to valid response tokens, and combine them before masked whitening in Eq. 3. The adaptive coefficients control allocation after per-dimension normalization. In both modes, rewards and weights are fixed targets during the policy update.

4.3 Practical considerations

Reward origins and stability. The chosen lower bounds are -3 for tool-call correctness and zero for the other rewards, including unshifted summarization scores. A positive ϵ_k stabilizes the denominator. Rescaling a reward, its lower bound, and its stabilizer by the same positive factor preserves its CV; shifting its origin without adjusting the bound generally changes the weights. An all-zero objective has zero empirical CV, so the rule relies on the batch containing informative variation. Appendix A gives the corresponding boundary analysis.

Priority weights and cost. Fixed priorities α_k multiply the adaptive coefficients at aggregation time and are kept separate from CV calculation. We use $\alpha_k = 1$ except in the fixed-ratio ablation. Computing the statistics costs $O(NK)$ arithmetic operations on existing reward scores. SAW requires neither another reward-model call nor per-objective policy gradients. It therefore offers a lightweight alternative to gradient-interaction weighting while retaining the same batch-level, per-objective granularity.

5 Experiments

5.1 Experimental setup

We evaluate SAW on tool calling and summarization using Qwen2.5-1.5B-Instruct and Qwen2.5-3B-Instruct (Qwen Team, 2025), trained with verl (Sheng et al., 2025). Within each optimizer family, we compare equal weighting, gradient-based dynamic weighting (Lu et al., 2026), and SAW. All trained models are evaluated at training step 100. Training settings, reward formulations, and prompts are collected in the appendices.

Tool calling. Following ToolRL (Qian et al., 2025), training uses 4,000 examples: 2,000 ToolACE, 1,000 Hammer, and 1,000 xLAM examples (Liu et al., 2025c; Lin et al., 2024; Zhang et al., 2024). The two-reward setting combines structural validity in $\{0, 1\}$ and tool-call correctness in $[-3, 3]$. The three-reward setting adds a capped length reward that favors longer `<think>...</think>` segments, based on their whitespace word count. We evaluate on the Non-live, Live, and Multi-turn BFCL-v4 subsets (Patil et al., 2025), reporting their aggregate Overall score and component results. Format and Length are reported separately; metric aggregation is specified in Appendix B.

Summarization. The Reddit TL;DR task (Völske et al., 2017; Stiennon et al., 2022) uses Qwen2.5-1.5B-Instruct, 2,560 training examples, and 256 evaluation examples. Quality and conciseness are each scored from 1 to 10, and their sum is the total score. We also evaluate the generated summaries with two judges held out from training.

Table 1: Two-reward BFCL results. Bold and underlined values denote the best and second-best result, respectively, within each model and optimizer family; Base is excluded from ranking. Scores are percentages; improvements are computed from unrounded values.

Method	Non-live	Live	Multi-turn	Overall	Format
Qwen2.5-1.5B-Instruct, two rewards					
GRPO	77.65	64.17	1.750	53.74	76.55
GRPO+Gradient	<u>78.61</u>	<u>68.39</u>	<u>2.000</u>	<u>55.86</u>	<u>80.01</u>
GRPO+SAW	**79.13**	**68.76**	**2.380**	**56.29**	**91.67**
GDPO	<u>78.35</u>	67.95	2.120	55.62	93.33
GDPO+Gradient	78.17	<u>68.09</u>	**2.380**	<u>55.68</u>	<u>93.76</u>
GDPO+SAW	**80.41**	**70.24**	<u>2.250</u>	**57.31**	**95.76**
Qwen2.5-3B-Instruct, two rewards					
Base	55.65	61.06	1.750	44.80	20.05
GRPO	80.43	68.47	5.120	57.28	80.76
GRPO+Gradient	**81.65**	<u>69.21</u>	<u>5.250</u>	<u>58.04</u>	<u>88.22</u>
GRPO+SAW	<u>80.70</u>	**70.91**	**7.750**	**59.01**	**89.82**
GDPO	**82.70**	<u>70.98</u>	6.000	<u>59.32</u>	94.31
GDPO+Gradient	81.22	70.39	<u>7.750</u>	58.98	<u>96.43</u>
GDPO+SAW	<u>82.17</u>	**71.50**	**10.380**	**60.40**	**96.52**

5.2 TOOL-CALLING PERFORMANCE

Tables 1 and 2 show that SAW achieves the highest Overall score in all six comparison families. With two rewards, it improves over GRPO and GDPO by $+2.55$ and $+1.69$ percentage points at 1.5B, and by $+1.73$ and $+1.09$ at 3B. With three rewards at 1.5B, the gains are $+7.37$ and $+1.45$. It also exceeds the gradient-weighted comparator in every configuration, by $+0.42$ to $+1.63$ points, using only reward statistics to determine its weights.

The gains span both reward-level and advantage-level integration. At 1.5B with two rewards, SAW improves all three BFCL components over the corresponding equal-weight baseline. At 3B, GRPO Multi-turn rises from 5.12 to 7.75, while GDPO Multi-turn rises from 6.00 to 10.38 and Non-live decreases from 82.70 to 82.17. Format improves over equal weighting in all six configurations. Overall improves across the evaluated settings, with trade-offs among subtasks.

Training dynamics. The training analysis in Figure 4 shows earlier format-reward growth for GRPO+SAW, with comparable terminal correctness scores. These observations are consistent with reallocating weight as reward distributions change, and complement the downstream BFCL results.

Fixed reward-weight ratios. In the 1.5B two-reward setting, we compare fixed correctness:format ratios of 1:2, 1:1, and 2:1 (Appendix D). The highest fixed-ratio Overall scores are 53.98 for GRPO and 55.62 for GDPO, below SAW's 56.29 and 57.31. The 1:2 mixture, however, yields higher Multi-turn scores in both optimizers. Thus, the aggregate advantage extends beyond the default fixed mixture in the tested range.

Choice of weighting statistic. In the two-reward 1.5B setting, replacing CV with normalized standard deviation or entropy reduces Overall from 56.29 to 54.14/53.53 in GRPO and from 57.31 to 54.71/55.01 in GDPO (Appendix C). NormStd nevertheless achieves higher Multi-turn scores than CV in both optimizers. CV also yields the highest Format score among these statistics.

5.3 THREE-REWARD OPTIMIZATION

Table 2 reports BFCL results, Format, and mean think-segment length among responses with an extracted segment. With GRPO, SAW raises Overall from 40.49 to 47.86, while mean Length increases from 463.61 to 508.66 tokens (9.72%). With GDPO, Overall rises from 50.65 to 52.11,

Table 2: Three-reward results for Qwen2.5-1.5B-Instruct. Bold and underline follow Table 1 for BFCL and Format scores. Length is descriptive and is not ranked; it averages tokens inside extracted <think>...</think> segments over responses with an extracted segment. Base does not follow this format ("–").

Method	Non-live	Live	Multi-turn	Overall	Format	Mean think length (tokens)
Qwen2.5-1.5B-Instruct, three rewards						
Base	18.52	28.05	0.750	18.11	5.97	–
GRPO	56.85	50.18	0.625	40.49	70.98	463.61
GRPO+Gradient	<u>65.51</u>	<u>56.85</u>	<u>1.250</u>	<u>46.39</u>	<u>81.86</u>	499.73
GRPO+SAW	**67.65**	**58.47**	**1.500**	**47.86**	**84.29**	508.66
GDPO	<u>69.57</u>	63.88	1.125	<u>50.65</u>	<u>82.54</u>	513.42
GDPO+Gradient	69.13	<u>63.95</u>	**1.500**	50.62	81.40	534.69
GDPO+SAW	**71.74**	**65.51**	<u>1.250</u>	**52.11**	**86.31**	542.65

Table 3: TL;DR evaluation. Quality (Qual.) and Conciseness (Conc.) are each scored from 1 to 10, and Total is their sum; higher is better. Held-out judges evaluate the same summaries. Bold and underlining mark the best and second-best values within each optimizer family.

Method	Gemini-3-flash (training judge)			Qwen3-Max (held out)			DeepSeek-V4-flash (held out)		
	Qual.	Conc.	Total	Qual.	Conc.	Total	Qual.	Conc.	Total
Qwen2.5-1.5B-Instruct									
Base	7.34	7.35	14.69	5.81	5.92	11.73	5.37	6.01	11.38
GRPO	8.71	9.49	18.20	7.98	8.34	16.32	7.83	7.97	15.80
GRPO+Gradient	<u>9.11</u>	<u>9.64</u>	<u>18.75</u>	<u>8.03</u>	<u>8.40</u>	<u>16.43</u>	<u>7.92</u>	<u>8.01</u>	<u>15.93</u>
GRPO+SAW	**9.15**	**9.65**	**18.80**	**8.17**	**8.42**	**16.59**	**7.95**	**8.07**	**16.02**
GDPO	9.10	9.70	18.80	8.09	8.27	16.36	7.84	<u>8.06</u>	<u>15.90</u>
GDPO+Gradient	<u>9.17</u>	**9.77**	<u>18.94</u>	<u>8.15</u>	<u>8.45</u>	<u>16.60</u>	<u>7.88</u>	7.95	15.83
GDPO+SAW	**9.35**	<u>9.72</u>	**19.07**	**8.18**	**8.46**	**16.64**	**7.99**	**8.18**	**16.17**

while mean Length increases from 513.42 to 542.65 tokens (5.69%). Format also improves in both families. These results show higher aggregate correctness and Format scores alongside longer extracted reasoning segments; they do not establish a reduction in reasoning length.

5.4 Summarization and held-out judges

Table 3 shows that SAW obtains the highest total summarization score within both optimizer families under all three judges. Under the training judge, the total improves from 18.20 to 18.80 for GRPO and from 18.80 to 19.07 for GDPO. The ranking persists under Qwen3-Max and DeepSeek-V4-flash, which were not used to provide the training reward; both also rank SAW highest on quality and conciseness in each optimizer family. Under Gemini, GDPO+Gradient has slightly higher conciseness (9.77 versus 9.72), while GDPO+SAW achieves higher quality and total. Unlike the sharply separated stages in the ToolRL example, both summarization rewards continue to evolve during training; the original trajectories are shown in Appendix F.

A supplementary human comparison reports agreement with majority-vote human preferences on 499 of 600 pairwise comparisons (83.17%), sampled where the corresponding judge gave unequal scores. This provides a conditional check on judge preferences. The agreement breakdown appears in Appendix E.

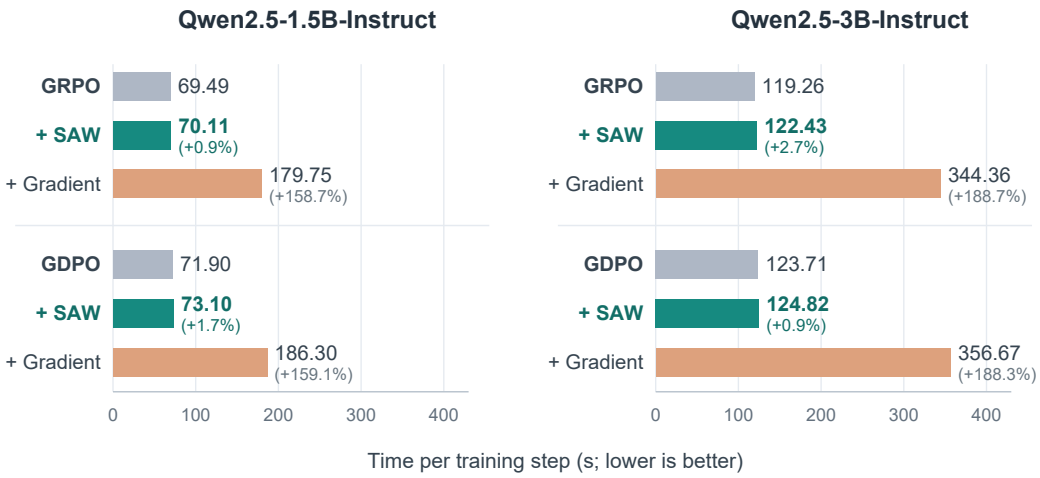


Figure 5: Average wall-clock time per ToolRL training step with two rewards on four NVIDIA A100 80GB GPUs. Both panels share the same zero-based time scale. Labels report seconds per step; parentheses give overhead relative to the corresponding GRPO or GDPO baseline. Lower is better.

5.5 Efficiency analysis

We report amortized wall-clock time per training step in a two-reward ToolRL study on four NVIDIA A100 80GB GPUs (Figure 5; protocol in Appendix B). Relative to the corresponding GRPO or GDPO baseline, SAW adds only 0.9–2.7% overhead, compared with 158.7–188.7% for gradient-based weighting. Across these settings, the gradient comparator takes 2.55–$2.86\times$ as long per step as SAW. This difference is consistent with the computation required by each approach: SAW computes reward moments in $O(NK)$ time, whereas the comparator forms per-objective gradients and their interaction matrix. The measurements establish low per-step overhead; training trajectories provide a separate view of progress over optimization steps.

6 Related work

Language-model alignment. Learning from human preferences and instruction feedback underpins language-model alignment (Christiano et al., 2023; Ouyang et al., 2022; Bai et al., 2022a). Direct Preference Optimization offers a preference-based training formulation (Rafailov et al., 2024), while constitutional, AI-feedback, and self-rewarding approaches expand the sources of supervision (Bai et al., 2022b; Lee et al., 2024; Yuan et al., 2025). Targeted human judgments and rule-based rewards provide additional mechanisms for specifying desired behavior (Glaese et al., 2022; Mu et al., 2024). These approaches motivate studying how multiple feedback dimensions contribute during training.

Group-relative optimization. GRPO (Shao et al., 2024) supports a growing family of language-model RL methods. DAPO changes clipping and sampling (Yu et al., 2025); Dr.GRPO analyzes biases in the training objective (Liu et al., 2025d); and GSPO uses sequence-level importance ratios (Zheng et al., 2025). Reasoning systems illustrate the broader role of RL (Guo et al., 2025; Kimi Team et al., 2025; DeepSeek-AI et al., 2025), with applications also extending to tool calling and tool-augmented audio reasoning (Qian et al., 2025; Tong et al., 2026). GDPO and MO-GRPO investigate per-objective normalization (Liu et al., 2026; Ichihara et al., 2026), while HA-DW addresses difficulty-dependent advantage bias across prompts (Yang et al., 2026). SAW adapts coefficients across reward dimensions within group-relative optimization.

Multiple objectives and efficient reasoning. Safe RLHF formulates helpfulness–safety constraints (Dai et al., 2024); ALARM develops hierarchical rewards (Lai et al., 2024); and fine-grained feedback separates aspects and textual spans (Wu et al., 2023). Conflicting objectives have also motivated reward-free alignment (Chen et al., 2026) and Pareto-oriented methods including PAMA and GAPO

(He & Maghsudi, 2025; Li et al., 2025). Accuracy–length trade-offs are studied through length penalties (Luo et al., 2026; Arora & Zanette, 2025; Liu et al., 2025a), short-response guidance (Yi et al., 2025), explicit length control (Aggarwal & Welleck, 2025), adaptive reward shaping (Liu et al., 2025b), and response-group filtering (Shrivastava et al., 2025). Our three-reward experiments examine adaptive weighting when a length objective accompanies correctness and format.

Dynamic weighting. Multi-task methods balance gradients using norms, multi-objective optimization, gradient surgery, conflict reduction, or bargaining (Chen et al., 2018; Sener & Koltun, 2019; Yu et al., 2020; Liu et al., 2024; Navon et al., 2022). In language-model RL, Lu et al. (2026) motivate dynamic weighting partly by differing objective convergence rates. Their gradient-based variant adapts objective weights using gradient interactions; their hypervolume-guided variant retains prescribed relative weights and applies a common meta-reward scale. We compare against the gradient-based variant to match objective-wise allocation. CoV-Weighting uses temporal loss-ratio statistics in supervised multi-loss learning (Groenendijk et al., 2020). SAW uses current-batch reward variation, integrated at the reward or advantage level without a loss history.

7 Conclusion and limitations

SAW adapts multi-reward optimization using current-batch relative dispersion. Its reward-level and advantage-level forms improve aggregate tool-calling performance across all six evaluated configurations, extend to three objectives, and achieve higher summarization totals under both held-out judges. In the 1.5B two-reward setting, it also exceeds both alternative statistics and all three tested fixed ratios in Overall. The rule requires no additional per-objective backward pass, with measured per-step overhead below 3% in the two-reward timing study.

The evaluation covers two model scales in one family and up to three rewards. The CV proxy can be affected by reward noise, sparse successes, and between-prompt variation; fixed reward origins and numerical stabilization therefore matter. These boundaries motivate broader model coverage and further analysis of when relative dispersion tracks useful learning opportunities.

AI use statement

Generative AI tools were used to assist manuscript writing and polishing, including language editing, LaTeX formatting, and checks of clarity and consistency. The authors independently designed and conducted all experiments and take responsibility for the accuracy and integrity of the manuscript. Language models also served as reward and evaluation judges in the summarization experiments, as described in the paper.

Ethics statement

Tool-call accuracy and format rewards measure task performance, not safe autonomous execution. The method does not impose safety constraints. Reddit summarization data can contain personal information and should be handled according to the source terms. Human evaluation should protect annotator privacy and follow applicable consent and institutional requirements.

Reproducibility statement

Section 4 and Algorithm 1 specify the weighting rule and its integration. The appendices provide analytical details, metric aggregation, training settings, reward definitions, prompts, and additional training trajectories. An anonymized implementation of the weighting rule is available at `https://anonymous.4open.science/r/SAW-9B8E/`.

References

Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.04697.

Daman Arora and Andrea Zanette. Training language models to reason efficiently, 2025. URL https://arxiv.org/abs/2502.04463.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022a. URL https://arxiv.org/abs/2204.05862.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022b. URL https://arxiv.org/abs/2212.08073.

Baolong Bi, Shenghua Liu, Yiwei Wang, Siqian Tong, Lingrui Mei, Yuyao Ge, Yilong Xu, Jiafeng Guo, and Xueqi Cheng. Reward and guidance through rubrics: Promoting exploration to improve multi-domain reasoning, 2025. URL https://arxiv.org/abs/2511.12344.

Peter Chen, Xiaopeng Li, Xi Chen, and Tianyi Lin. Reward-free alignment for conflicting objectives, 2026. URL https://arxiv.org/abs/2602.02495.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks, 2018. URL https://arxiv.org/abs/1711.02257.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023. URL https://arxiv.org/abs/1706.03741.

Juntao Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe rlhf: Safe reinforcement learning from human feedback. In *International Conference on Learning Representations*, volume 2024, pp. 50750–50777, 2024.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan,

Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: A family of highly capable multimodal models, 2025. URL https://arxiv.org/abs/2312.11805.

Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Soňa Mokrá, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. Improving alignment of dialogue agents via targeted human judgements, 2022. URL https://arxiv.org/abs/2209.14375.

Rick Groenendijk, Sezer Karaoglu, Theo Gevers, and Thomas Mensink. Multi-loss weighting with coefficient of variations, 2020. URL https://arxiv.org/abs/2009.01717.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains, 2025. URL https://arxiv.org/abs/2507.17746.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong,

Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Qiang He and Setareh Maghsudi. Pareto multi-objective alignment for language models, 2025. URL https://arxiv.org/abs/2508.07768.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, Xijun Gu, Peiyi Tu, Jiaxin Liu, Wenyu Chen, Yuzhuo Fu, Zhiting Fan, Yanmei Gu, Yuanyuan Wang, Zhengkai Yang, Jianguo Li, and Junbo Zhao. Reinforcement learning with rubric anchors, 2025. URL https://arxiv.org/abs/2508.12790.

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Mitsuki Sakamoto, Ryota Mitsuhashi, and Eiji Uchibe. Mo-grpo: Mitigating reward hacking of group relative policy optimization on multi-objective problems, 2026. URL https://arxiv.org/abs/2509.22047.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

Yuhang Lai, Siyuan Wang, Shujun Liu, Xuanjing Huang, and Zhongyu Wei. Alarm: Align language models via hierarchical rewards modeling, 2024. URL https://arxiv.org/abs/2403.06754.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback, 2024. URL https://arxiv.org/abs/2309.00267.

Chengao Li, Hanyu Zhang, Yunkun Xu, Hongyan Xue, Xiang Ao, and Qing He. Gradient-adaptive policy optimization: Towards multi-objective alignment of large language models, 2025. URL https://arxiv.org/abs/2507.01915.

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, Jun Wang, and Weinan Zhang. Hammer: Robust function-calling for on-device language models via function masking, 2024. URL https://arxiv.org/abs/2410.04587.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning, 2024. URL https://arxiv.org/abs/2110.14048.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Dler: Doing length penalty right - incentivizing more intelligence per token via reinforcement learning, 2025a. URL https://arxiv.org/abs/2510.15110.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization, 2026. URL https://arxiv.org/abs/2601.05242.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping, 2025b. URL https://arxiv.org/abs/2505.15612.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin

Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. Toolace: Winning the points of llm function calling, 2025c. URL https://arxiv.org/abs/2409.00920.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025d. URL https://arxiv.org/abs/2503.20783.

Yining Lu, Zilong Wang, Shiyang Li, Xin Liu, Changlong Yu, Qingyu Yin, Zhan Shi, Zixuan Zhang, and Meng Jiang. Learning to optimize multi-objective alignment through dynamic reward weighting, 2026. URL https://arxiv.org/abs/2509.11452.

Haotian Luo, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Xiaochun Cao, Dacheng Tao, Naiqiang Tan, and Li Shen. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14242–14257, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.697. URL https://aclanthology.org/2026.findings-acl.697/.

Tong Mu, Alec Helyar, Johannes Heidecke, Joshua Achiam, Andrea Vallone, Ian Kivlichan, Molly Lin, Alex Beutel, John Schulman, and Lilian Weng. Rule based rewards for language model safety, 2024. URL https://arxiv.org/abs/2411.01111.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game, 2022. URL https://arxiv.org/abs/2202.01017.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs, 2025. URL https://arxiv.org/abs/2504.13958.

Qwen Team. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL https://arxiv.org/abs/2305.18290.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization, 2019. URL https://arxiv.org/abs/1810.04650.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297. ACM, March 2025. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.

Vaishnavi Shrivastava, Ahmed Awadallah, Vidhisha Balachandran, Shivam Garg, Harkirat Behl, and Dimitris Papailiopoulos. Sample more to think less: Group filtered policy optimization for concise reasoning, 2025. URL https://arxiv.org/abs/2508.09726.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2022. URL https://arxiv.org/abs/2009.01325.

Siqian Tong, Xuan Li, Yiwei Wang, Baolong Bi, Yujun Cai, Shenghua Liu, Yuchen He, and Chengpeng Hao. Autagent: A reinforcement learning framework for tool-augmented audio reasoning, 2026. URL https://arxiv.org/abs/2602.13685.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. TL;DR: Mining Reddit to learn automatic summarization. In Lu Wang, Jackie Chi Kit Cheung, Giuseppe Carenini, and Fei Liu (eds.), *Proceedings of the Workshop on New Frontiers in Summarization*, pp. 59–63, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4508. URL https://aclanthology.org/W17-4508/.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training, 2023. URL https://arxiv.org/abs/2306.01693.

Fengkai Yang, Zherui Chen, Xiaohan Wang, Xiaodong Lu, Jiajun Chai, Guojun Yin, Wei Lin, Shuai Ma, Fuzhen Zhuang, Deqing Wang, Yaodong Yang, Jianxin Li, and Yikun Ban. Your group-relative advantage is biased, 2026. URL https://arxiv.org/abs/2601.08521.

Jingyang Yi, Jiazheng Wang, and Sida Li. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning, 2025. URL https://arxiv.org/abs/2504.21370.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning, 2020. URL https://arxiv.org/abs/2001.06782.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2025. URL https://arxiv.org/abs/2401.10020.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, Zhiwei Liu, Yihao Feng, Tulika Awalgaonkar, Rithesh Murthy, Eric Hu, Zeyuan Chen, Ran Xu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Silvio Savarese, and Caiming Xiong. xlam: A family of large action models to empower ai agent systems, 2024. URL https://arxiv.org/abs/2409.03215.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025. URL https://arxiv.org/abs/2507.18071.

A Analytical details

Binary CV and stabilization. For $R \sim \text{Bernoulli}(p)$, $\mathbb{E}R = p$ and $\text{Var}(R) = p(1-p)$ give Eq. 4. With a positive mean stabilizer, $c_\epsilon(p) = \sqrt{p(1-p)}/(p+\epsilon)$ satisfies

$$\frac{d \log c_\epsilon(p)}{dp} = \frac{\epsilon - p(1+2\epsilon)}{2p(1-p)(p+\epsilon)}. \tag{7}$$

It increases below $p = \epsilon/(1+2\epsilon)$ and decreases above that boundary. An all-zero or all-one empirical batch has zero CV. Equal-weight fallback is used when all objective CVs are zero; an individual constant objective can receive zero weight when another objective varies.

Scale and origin. For $X \geq 0$ and $a > 0$, $s(aX) = as(X)$ and $m(aX) = am(X)$, hence

$$\frac{s(aX)}{m(aX) + a\epsilon} = \frac{s(X)}{m(X) + \epsilon}. \tag{8}$$

With a fixed stabilizer, the right-hand denominator instead becomes $m(X) + \epsilon/a$. Adding a constant b leaves the standard deviation unchanged while adding b to the mean. For example, rewards with moments $(m_1, s_1) = (1, 1)$ and $(m_2, s_2) = (10, 2)$ have CVs 1 and 0.2; adding 100 reverses their order. A consistent affine recoding $r'_k = a_k r_k + b_k$ therefore transforms the bound to $\ell'_k = a_k \ell_k + b_k$ and the stabilizer to $a_k \epsilon_k$. The coefficients depend on CV ratios, not only their ranking.

Reward dispersion and gradient information. Marginal reward moments do not identify a useful policy-gradient direction. Let a scalar policy score Z take values -1 and $+1$ with equal probability, and set $R_+ = (1+Z)/2$, $R_- = (1-Z)/2$. Both rewards are Bernoulli$(1/2)$, yet $\mathbb{E}[ZR_+] = 1/2$ and $\mathbb{E}[ZR_-] = -1/2$. Thus identical CVs can accompany opposite gradient correlations.

Within- and between-prompt variation. For prompt Q, the law of total variance gives

$$\text{Var}(R_k) = \mathbb{E}_Q[\text{Var}(R_k \mid Q)] + \text{Var}_Q(\mathbb{E}[R_k \mid Q]). \tag{9}$$

Batch CV includes both terms, whereas group-relative advantages depend on within-prompt differences. As an extreme example, let all responses to one prompt score zero and all responses to another score one. Global CV is positive but all group-centered advantages are zero. The stage-aware interpretation is most relevant when dispersion includes actionable within-prompt differences.

Coefficient normalization. With a fixed response mask, nonzero variance, and no stabilizer, both group normalization and token-level whitening satisfy $\mathcal{Z}(au) = \mathcal{Z}(u)$ for $a > 0$. Scaling inputs by a is equivalent to replacing ϵ_g by ϵ_g/a in the group denominator $s + \epsilon_g$, or ϵ_b by ϵ_b/a^2 in the whitening denominator $\sqrt{v + \epsilon_b}$. Thus the common factor K sets the coefficient scale, not a separate allocation mechanism. Token-level and response-level whitening can nonetheless have different centers and advantage signs; they are not related by scaling alone.

B Evaluation and training details

BFCL aggregation. The evaluated Non-live, Live, and Multi-turn subsets of BFCL-v4 are combined using

$$A_{\text{overall}} = \frac{1351 A_{\text{live}} + 1150 A_{\text{nonlive}} + 800 A_{\text{multi}}}{3301}. \tag{10}$$

The coefficients above define the reported Overall aggregate. Format measures compliance with the prescribed output schema separately from BFCL correctness, so a correct tool call need not satisfy the Format criterion. Format and Length are auxiliary measurements and do not enter Overall. Length averages the token count inside extracted `<think>...</think>` segments on BFCL-v4. The denominator is the number of responses with an extracted segment; responses without one are excluded. Each reply in a multi-turn task is counted separately. This conditional token-count statistic is distinct from the whitespace word-count reward used during training. Tables report point estimates.

Training configuration. Table 4 lists the training configuration for tool calling and summarization. All trained-model results use the checkpoint saved at training step 100. The PPO clipping ratio is 0.2, and GRPO group normalization uses $\epsilon_g = 10^{-6}$. For CV, correctness is shifted by $+3$; the code's $|m_k|$ equals m_k for these nonnegative scores. Both the CV denominator and the weight-normalization denominator use a stabilizer of 10^{-8}.

Table 4: Training settings.

Setting	Tool calling	TL;DR
Epochs	15	11
Prompt batch size	512	256
Mini-batch size	128	64
Maximum prompt length	2048	2048
Maximum response length	1024	1024
Learning rate	10^{-6}	10^{-6}
Responses per prompt	4	4
Reward-side KL penalty	Disabled	Disabled
Separate KL loss	Disabled	Disabled
Gradient checkpointing	Enabled	Enabled
Rollout engine	vLLM	vLLM
Rollout tensor parallelism	1	1
GPU memory utilization	0.6	0.6
Hardware	$4\times$ A100 80GB	$4\times$ A100 80GB

Tool-call rewards. The format reward is one when a response satisfies the required structured schema and zero otherwise. Correctness uses sequential greedy matching between ordered reference calls T and predictions P. A same-name pair (t, p) has score $q(t, p) = J(K_t, K_p) + \sum_{a \in K_t \cap K_p} \mathbf{1}[v_t(a) = v_p(a)]$, where K denotes parameter-key sets, v their values, and J is set Jaccard ($J(\varnothing, \varnothing) = 1$). References are processed in order, each taking the unused same-name prediction with the highest positive score; ties follow prediction order. Let $\mathcal{M}_g$ be the resulting pairs, and N_T, N_P the tool-name multisets. Then

$$r_{\text{correct}} = 6\frac{J_{\text{ms}}(N_T, N_P) + \sum_{(t,p)\in\mathcal{M}_g} q(t, p)}{1 + |T| + \sum_{t\in T} |K_t|} - 3, \tag{11}$$

where J_{ms} is multiset Jaccard: the sum of minimum name counts divided by the sum of maximum counts (one if both are empty). References without a tool-call block receive zero; otherwise, missing prediction delimiters or parsing/scoring failures receive -3. Let $W_{\text{think}}(o)$ count whitespace-separated words inside the extracted `<think>...</think>` text. The third reward is $r_{\text{length}}(o) = \min\{\text{round}_2(W_{\text{think}}(o)/512), 1\}$, where round_2 rounds to two decimal places. A missing opening or closing delimiter yields zero. This reward is nondecreasing in word count and saturates at one; it does not reward shorter reasoning. Lower bounds used for CV are -3 for correctness and zero for format and length.

GDPO token-level whitening. Let $u_{ij} = \sum_k \alpha_k w_k a_{ijk}$ and let L_{ij} be the number of valid tokens in the full response, with $D = \sum_{i,j} L_{ij} > 1$. Broadcasting u_{ij} to those tokens and applying masked whitening gives

$$\mu_b = \frac{\sum_{i,j} L_{ij} u_{ij}}{D}, \qquad v_b = \frac{\sum_{i,j} L_{ij}(u_{ij} - \mu_b)^2}{D - 1}, \qquad \mathcal{Z}_b(u)_{ij} = \frac{u_{ij} - \mu_b}{\sqrt{v_b + \epsilon_b}},$$

with $\epsilon_b = 10^{-8}$. Each valid token receives this value; masked positions receive zero. GDPO and GDPO+SAW use the same operation. Group normalization and CV use sample standard deviations over responses, whereas final whitening weights responses by their valid token counts, not by their think-segment lengths.

Gradient-weighting configuration. The gradient-based baseline updates its weights as $\mathbf{g}_{t+1} = \mathrm{softmax}(\log \mathbf{g}_t + (\eta_t/\lambda)\mathbf{d}_t)$, where $\mathbf{d}_t$ contains its per-objective gradient scores. We use $\lambda = 10^{-5}$ and actor learning rate $\eta_t = 10^{-6}$, giving a score multiplier of 0.1. The active reward dimensions have equal initial weights; two-reward tool-calling runs use only correctness and format. This softmax update belongs to the gradient baseline, not to SAW.

Timing protocol. For each run, amortized training time per step is the total training wall-clock time divided by the number of completed training steps. The timing study uses four NVIDIA A100 80GB GPUs. Relative overhead is computed against the corresponding equal-weight baseline.

Summarization protocol. The training judge uses the Gemini model family (Gemini Team et al., 2025), specifically Gemini-3-flash-preview-nothinking at temperature 0.1, to score quality and conciseness separately from 1 to 10. Their CVs are computed directly from the unshifted scores ($\ell_{\text{quality}} = \ell_{\text{conciseness}} = 0$). Our rubric design follows rubric-based reward modeling (Huang et al., 2025; Gunjal et al., 2025; Bi et al., 2025). During evaluation, the same generated summaries are scored in four passes and the scores are averaged across passes. The scoring rubric appears in Appendix I.

C Ablation on the weighting statistic

We replace CV with two alternative weighting statistics in the Qwen2.5-1.5B-Instruct two-reward setting, retaining the training settings and BFCL-v4 evaluation used in Table 1. Both alternatives use the same current rollout batch and statistical scope as CV. For the weighting statistics only, normalize each raw reward using its fixed theoretical bounds,

$$z_{ijk} = \frac{r_{ijk} - \ell_k}{u_k - \ell_k}, \qquad (\ell_k, u_k) = (-3, 3) \text{ for correctness}, \quad (0, 1) \text{ for format.} \tag{12}$$

NormStd uses the population standard deviation

$$q_k = \sqrt{\frac{1}{N}\sum_{i,j}(z_{ijk} - \bar{z}_k)^2}. \tag{13}$$

Entropy uses $q_k = -\sum_{b=0}^{9} p_{kb} \log p_{kb}$, where p_{kb} is the empirical frequency in one of ten fixed, equal-width bins on $[0, 1]$. The bin index is $\min(\lfloor 10z \rfloor, 9)$; hence $z = 1$ belongs to the final bin. We use natural logarithms and $0 \log 0 = 0$, without pseudocounts or division by $\log 10$. On binary rewards, this reduces to Bernoulli entropy.

In both variants, substitute q_k for c_k in Eq. 6, retaining the same coefficient normalization, fixed priorities, fallback rule, and update frequency. This range normalization is used only to compute weighting statistics: the original training rewards and advantage construction remain unchanged.

Table 5: Weighting-statistic ablation on BFCL-v4 with Qwen2.5-1.5B-Instruct and two rewards. Accuracies and Format are percentages; zeros are measured scores. Default and CV reproduce the corresponding main-table results. Bold marks the best score within each optimizer family.

Optimizer	Weighting	Non-live	Live	Multi-turn	Overall	Format
GRPO	Default	77.65	64.17	1.75	53.74	76.55
	NormStd	77.57	64.70	**2.62**	54.14	0.00
	Entropy	77.74	64.54	0.12	53.53	0.00
	CV (SAW)	**79.13**	**68.76**	2.38	**56.29**	**91.67**
GDPO	Default	78.35	67.95	2.12	55.62	93.33
	NormStd	77.91	65.73	**2.75**	54.71	0.00
	Entropy	79.13	65.73	2.25	55.01	87.25
	CV (SAW)	**80.41**	**70.24**	2.25	**57.31**	**95.76**

CV achieves the highest Overall score in both optimizer families. NormStd slightly improves Overall over default GRPO but falls below default GDPO, while entropy falls below the default in both

families. NormStd obtains the highest Multi-turn score in each family, so the aggregate advantage does not imply uniform subtask dominance. CV also achieves the highest Format score. Both NormStd variants and GRPO-Entropy have zero Format despite nonzero BFCL accuracy, illustrating the distinction between schema compliance and tool-call correctness. These comparisons support the choice of CV among the tested statistics in this setting; they do not establish that CV separates learning signal from noise.

D Comparison with fixed reward-weight ratios

We compare three prespecified correctness:format ratios, 1:2, 1:1, and 2:1, using Qwen2.5-1.5B-Instruct with two rewards. These ratios specify the effective aggregation coefficients after any fixed priorities; they are distinct from the reward ranges $[-3, 3]$ and $\{0, 1\}$. For ratio $p : q$, the coefficients remain constant throughout training:

$$\beta_{\text{correct}} = S_0 \frac{p}{p+q}, \qquad \beta_{\text{format}} = S_0 \frac{q}{p+q}, \tag{14}$$

where $S_0 = 2$ is the default coefficient sum. Thus 2:1 and 1:2 use $(4/3, 2/3)$ and $(2/3, 4/3)$, respectively; 1:1 uses $(1, 1)$ and reproduces the default baseline. GRPO applies these coefficients before group normalization of the aggregated reward; GDPO applies them to the separately normalized advantages before final masked whitening over valid response tokens. Reward definitions, training settings, checkpoint selection, and BFCL-v4 evaluation match the main experiments.

Table 6: Fixed correctness:format ratios versus SAW on BFCL-v4 with Qwen2.5-1.5B-Instruct and two rewards. Accuracies and Format are percentages; zero is a measured score. The 1:1 and SAW rows reproduce the main-table results. Bold marks the best score within each optimizer family.

Optimizer	Weighting	Non-live	Live	Multi-turn	Overall	Format
GRPO	Fixed 1:2	77.74	63.80	**3.00**	53.92	80.37
	Fixed 1:1	77.65	64.17	1.75	53.74	76.55
	Fixed 2:1	76.43	65.44	2.38	53.98	76.10
	SAW	**79.13**	**68.76**	2.38	**56.29**	**91.67**
GDPO	Fixed 1:2	78.52	65.95	**4.50**	55.44	88.37
	Fixed 1:1	78.35	67.95	2.12	55.62	93.33
	Fixed 2:1	78.87	67.36	1.62	55.44	0.00
	SAW	**80.41**	**70.24**	2.25	**57.31**	**95.76**

SAW achieves higher Overall and Format than all three fixed ratios in both optimizers. The highest fixed-ratio Overall is 53.98 at 2:1 for GRPO and 55.62 at 1:1 for GDPO, compared with 56.29 and 57.31 for SAW. However, fixed 1:2 yields Multi-turn scores of 3.00 and 4.50, above SAW's 2.38 and 2.25. GDPO with fixed 2:1 retains 55.44 Overall despite zero Format. These results support an aggregate advantage over the tested fixed mixtures while exposing subtask and format trade-offs. The comparison is restricted to these three ratios and does not establish optimality over arbitrary fixed mixtures.

E Supplementary summarization results

The per-dimension summarization scores are reported in Table 3; the agreement breakdown is given below.

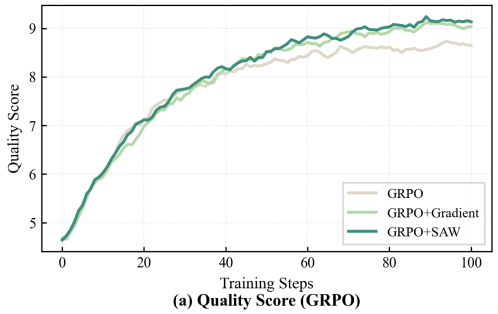
For each judge and evaluation dimension, we sampled 100 response pairs receiving unequal scores from that judge. Each of three annotators selected the better response for every pair. Method identities and judge scores were hidden, and the left–right order was randomized. The final human preference was determined by majority vote. Agreement with these preferences is $499/600 = 83.17\%$ over pairwise comparisons, conditional on non-tied judge scores. This checks judge–human agreement rather than direct human preference between SAW and a baseline.

Table 7: Conditional judge–human agreement. Each cell contains 100 pairs for which the corresponding judge assigned unequal scores.

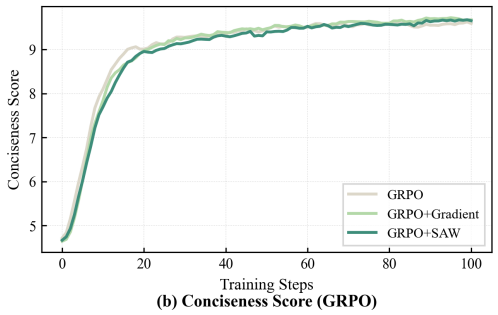
Judge	Quality agreement	Conciseness agreement
Gemini-3-flash	82/100	84/100
Qwen3-Max	83/100	82/100
DeepSeek-V4-flash	85/100	83/100

F SUMMARIZATION TRAINING DYNAMICS

Figures 6 and 7 complement the summarization results in the main text. Both quality and conciseness continue to evolve during training.

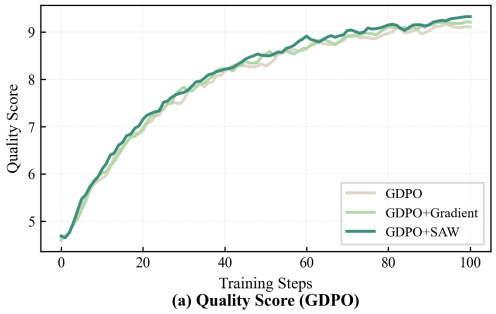


(a) Quality Score (GRPO)

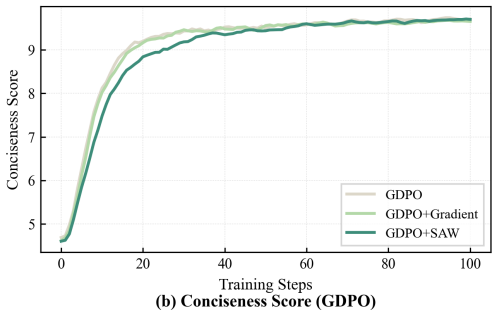


(b) Conciseness Score (GRPO)

Figure 6: Per-dimension reward trajectories for GRPO-based methods on Reddit TL;DR with Qwen2.5-1.5B-Instruct.



(a) Quality Score (GDPO)



(b) Conciseness Score (GDPO)

Figure 7: Per-dimension reward trajectories for GDPO-based methods on Reddit TL;DR with Qwen2.5-1.5B-Instruct.

G TOOLRL TRAINING PROMPT FORMAT

We follow the same setup as GDPO and ToolRL; the training prompt is given below.

System Prompt for ToolRL Training

You are a helpful dialogue assistant capable of leveraging tool calls to solve user tasks and provide structured chat responses.
Available Tools
In your response, you can use the following tools:
{{ Tool List }}
Steps for Each Turn

1. **Think**: Recall relevant context and analyze the current user goal.
2. **Decide on Tool Usage**: If a tool is needed, specify the tool and its parameters.
3. **Respond Appropriately**: If a response is needed, generate one while maintaining consistency across user queries.

Output Format
<think> Your thoughts and reasoning </think>
<tool_call> {"name": "Tool name", "parameters": {"Parameter name": "Parameter content", "... ...": "... ..."}}
{"name": "... ...", "parameters": {"... ...": "... ...", "... ...": "... ..."}}
...
</tool_call>
<response>AI's final response </response>
Important Notes

1. You must always include the <think> field to outline your reasoning. Provide at least one of <tool_call> or <response>. Decide whether to use <tool_call> (possibly multiple times), <response>, or both.
2. You can invoke multiple tool calls simultaneously in the <tool_call> fields. Each tool call should be a JSON object with a "name" field and a "parameters" field containing a dictionary of parameters. If no parameters are needed, leave the "parameters" field an empty dictionary.
3. Refer to the previous dialogue records in the history, including the user's queries, previous <tool_call>, <response>, and any tool feedback noted as <obs> (if exists).

User Prompt for ToolRL Training

Dialogue History
<user> {{ Initial User Input }} </user>
<think> Round 1 Model Thought </think>
{{ Round 1 model output <tool_call> or <response> }}
<obs> Round 1 Observation </obs>
... ...
<user> {{ User Input }} </user>
... ...

H REDDIT TL;DR TRAINING PROMPT FORMAT

The original summarization protocol uses the following system and input templates.

System Prompt for Reddit TL;DR Training

You are a professional content editor. Please read the following post and summarize its main message in concise and accurate language.

User Prompt for Reddit TL;DR Training

SUBREDDIT: r/AskReddit
TITLE: Java question getting desperate!
POST: I am currently enrolled in a “Java 2” Class in college. And my current project is to write a program that can solve mazes using two different methods 1. Depth First Search using a stack and 2. Breadth First Search using a queue. I have depth first search done, and have yet to begin working on breadth first search but i have a few ideas as to how to implement it. My problem is, with the client/driver program that runs all of the classes (contains my main method). I have to scan in text files that contain mazes. I have tested my code by scanning the mazes and everything is working fine. The trouble comes with the instructor says that he wants to be able to put a directory into the command line (args) and have our program parse the directory and solve each maze in the directory. I did directory.list() and place all of the file names in the directory into an array. But I do not know how to make it so my program can scan those files. The program gets upset because the files are not in the same directory as the code. If anyone can help with this I would forever be in your debt.
TL;DR:

I REDDIT TL;DR LLM-AS-JUDGE RUBRIC

The original protocol uses Gemini-3-flash-preview-nothinking (temperature 0.1) to score generated summaries against the source post and human reference. The prompt is reproduced below, with line wrapping and JSON whitespace adjusted for display.

LLM-as-Judge Scoring Prompt

 # Role
You are a highly rigorous linguistic auditor. Your task is to perform a comparative evaluation of a "Model-Generated Summary" against a source text and a reference summary.
 # Evaluation Dimensions
1. Quality (Focus: Accuracy & Coverage)
Score this dimension based on factual correctness and essential information retention.
- 9–10 (Perfect): No hallucinations. Captures all core entities and the main conflict/conclusion mentioned in the human reference.
- 7–8 (Minor Issue): Factual alignment is 100%, but misses one secondary supporting detail from the human reference.
- 5–6 (Major Omission): Misinterprets a secondary detail OR misses the primary "hook"/conclusion of the post.
- 3–4 (Severe Error): Contains at least one significant factual hallucination OR fails to address the main subject.
- 1–2 (Failure): Completely irrelevant, toxic, or incoherent.

2. Conciseness (Focus: Word Efficiency & Redundancy)
Score this dimension based on the "Information-to-Word Ratio."
- 9–10 (Optimal): Zero filler phrases. Every word contributes to the meaning. No repetition.
- 7–8 (Slightly Wordy): Includes 1–2 unnecessary words or "meta-talk" (e.g., "The user says...", "In this story...").
- 5–6 (Redundant): Contains repetitive information or excessive adjectives that do not add meaning.
- 3–4 (Inefficient): Is longer than the human reference without adding extra useful info, OR 30% of the text is filler.
- 1–2 (Bloated): Verbose, contains circular reasoning, or is nearly the same length as the original post.

 # Execution Logic (Chain of Thought)
1. Identify the "Core Message" from the Human Summary.
2. Check if the Model Summary contains this Core Message (Quality).
3. Count "Filler Words": phrases that can be removed without changing the meaning (Conciseness).
4. Penalize hallucinations heavily.

 # Task Input

[Original Reddit Post]:
`$reddit_text`

[Human-Written Summary (Reference)]:
`$human_summary`

[Model-Generated Summary (To be evaluated)]:
`$model_summary`

 # Constraints
1. Output ONLY a valid JSON object.
2. Be clinical and objective.

 # Output Format

```
{
  "quality": {"score": [Integer]},
  "conciseness": {"score": [Integer]}
}
```