Beta-RL: Bayesian Uncertainty-Aware Reward Weighting for Multi-Reward Reinforcement Learning of Large Language Models

Anonymous ACL submission

Abstract

Reinforcement learning with multiple reward signals is now standard for post-training large language models, but rewards are still combined with hand-tuned, fixed weights—a brittle choice when rewards saturate at different rates. We propose BETA-RL, a Bayesian uncertainty-aware reward weighting framework that learns per-channel precisions β_k. Naïvely transplanting the recent Beta-KD framework (Sun et al., 2026) fails: we prove that batch normalization in the policy advantage renders the absolute scale of β unidentifiable, so β drifts to its regularization boundary. A simplex reparameterization $\beta_k = K \cdot \mathrm{softmax}(\alpha)_k$ resolves this and forces β to learn only the *relative* importance of channels. On the GDPO math-reasoning suite (Liu et al., 2026) with DeepSeek-R1-Distill-Qwen-1.5B, BETA-RL matches the strongest hand-engineered baseline (reward conditioning) without per-task tuning ($+2.74$ vs $+2.44$ average), generalizes to $K \geq 3$, and is complementary to conditioning: combined, they yield $+5.21$ absolute average and $+80\%$ relative on AIME-24. The learned $\beta = (1.957, 0.043)$ is interpretable: the model autonomously deprioritizes the saturated length reward.

1 Introduction

The training of modern reasoning-oriented large language models (LLMs)—DeepSeek-R1 (DeepSeek-AI, 2025), the OpenAI o1 series, Qwen3 (Qwen Team, 2024)—is now dominated by reinforcement learning with verifiable rewards and Group Relative Policy Optimization (GRPO) (Shao et al., 2024; DeepSeek-AI, 2025). As these recipes mature, practitioners increasingly stack *multiple* reward signals to shape behaviour: tool-use agents are scored on tool-call format *and* correctness (Qian et al., 2025); long-form reasoning models are jointly optimized for accuracy, response length, and reasoning structure (Liu et al., 2026, 2025; Yu et al., 2025b); safety-aligned models combine helpfulness with constraint adherence (Ramé et al., 2024; Yang et al., 2024a; Shi et al., 2025). Multi-reward RL is, in short, the new default.

Yet despite this ubiquity, the way rewards are *combined* remains surprisingly primitive. Both dominant aggregations—*sum-then-normalize* (vanilla GRPO) and *normalize-then-sum* (GDPO; Liu et al., 2026)—rely on a single static weighting vector $\mathbf{w}$ that is hand-tuned per project. This is unsatisfactory because rewards saturate at different rates (Figure 1: length saturates within tens of steps while correctness keeps improving over hundreds; the optimal weight on a saturated reward should approach zero), the optimal weighting drifts during training, and manual tuning does not scale across model sizes, datasets, and training stages.

Our starting point. The recently proposed **Beta-KD** framework (Sun et al., 2026) addresses an analogous problem in *knowledge distillation*: it interprets each teacher signal as a Gibbs prior over student activations, derives an uncertainty-aware weighting via Laplace approximation, and learns precision parameters β_k that automatically down-weight noisy or saturated supervision channels. Beta-KD generalizes the seminal multi-task uncertainty weighting of Kendall et al. (2018) to arbitrary distillation losses. It is natural to ask: does the same Bayesian recipe transfer to multi-reward RL?

A direct transplant fails. We show, both empirically and theoretically, that it does not (Sec. 3). The reason is structural: in GDPO-style normalize-then-sum, the post-aggregation advantage is *batch-normalized*. We prove (Proposition 1) that batch normalization renders the *absolute* scale of β unidentifiable from the policy loss; only the regularizer $-\sum \log \beta_k$ provides gradient on the global scale, and that regularizer monotonically pushes β to whatever clamp boundary is imposed. In our 250-step experiments, β converges

to $(5.007, 5.007)$—both at the upper clamp, with no relative differentiation. The method recovers GDPO performance with *zero* added benefit (Table 1, "BETA-RL$_{\text{softplus}}$"). This is a textbook case of a Bayesian framework cleanly failing under a model-implementation mismatch.

Our fix: simplex reparameterization. Proposition 1 also reveals *what* can be learned: only the relative magnitudes $\beta_k/\beta_{k'}$ are identifiable. We therefore constrain β to a probability simplex,

$$\beta_k = K \cdot \text{softmax}(\alpha)_k, \quad \sum_k \beta_k = K, \qquad (1)$$

where K is the number of reward channels and $\alpha \in \mathbb{R}^K$ is trainable. This eliminates the unidentifiable global scale: β can only redistribute weight among rewards, never grow uniformly. We replace the divergent log-regularizer with an entropy regularizer that prefers uniform β unless evidence drives differentiation. The resulting framework, BETA-RL$_{\text{simplex}}$, is a 30-line drop-in modification to existing GDPO trainers and adds only K scalar parameters.

Empirical results. On the GDPO mathematical reasoning protocol (Liu et al., 2026), BETA-RL$_{\text{simplex}}$ improves Pass@1 over GDPO on *all four* benchmarks evaluated (Table 1): AIME-2024 $8.33 \rightarrow 12.50$ (**+50% relative**), AMC $42.38 \rightarrow 47.87$ (**+13%**), MATH-500 $65.65 \rightarrow 66.40$, and Minerva $17.83 \rightarrow 18.38$, with average gain of $+2.74$ **absolute** ($+8.16\%$ relative)—matching the strongest hand-engineered baseline (reward conditioning at $+2.44$) without any per-task tuning. Stacking BETA-RL on top of conditioned rewards is *complementary* and gives the strongest configuration overall: $+5.21$ **absolute average** (AIME-24 $\rightarrow 15.00$, AMC $\rightarrow 53.05$).

Why does it work? Inspection of the learned $\beta = (1.957, 0.043)$ for (correctness, length) reveals that BETA-RL$_{\text{simplex}}$ has *automatically deprioritized* the saturated length reward and reallocated capacity to the still-informative correctness signal. This is the same outcome that the GDPO paper hand-engineers via reward conditioning (Liu et al., 2026), but reached as a single soft inference step learned from data—and one that combines additively with conditioning when both are applied.

Contributions.

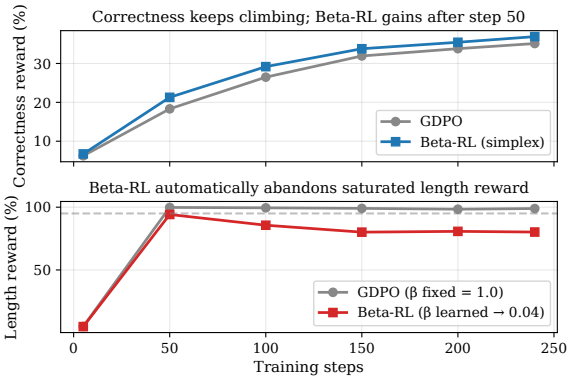

Figure 1: **Multi-reward RL has a saturation problem that fixed weights cannot solve. (Top)** The easy reward (length) saturates within ∼50 steps; the hard reward (correctness) continues to climb. With static weights, GDPO wastes gradient on the saturated channel. **(Bottom)** BETA-RL$_{\text{simplex}}$ automatically learns $\beta_{\text{length}} \rightarrow 0.04$ once the length reward saturates, reallocating capacity to correctness—recovering the behavior that previously required manual reward conditioning, purely from data.

1. We identify and formally characterize (Proposition 1) a fundamental obstacle—batch-normalization-induced scale invariance—when applying Beta-KD-style uncertainty weighting to RL.
2. We propose a simple, theoretically motivated fix via simplex reparameterization (Eq. 1), which we instantiate at the task level and at the instance level.
3. We demonstrate consistent gains on the four-benchmark GDPO math suite, with peak $+50\%$ relative AIME-24 improvement, matching the strongest hand-engineered baseline (reward conditioning) without per-task tuning, and reaching $+80\%$ relative AIME-24 ($+5.21$ absolute average) when combined with reward conditioning. We provide an interpretable mechanistic explanation: BETA-RL automatically abandons saturated rewards. We release code as a drop-in TRL patch.

2 Related Work

RL post-training and GRPO variants. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) replaces the value critic of PPO with group-normalized rewards and is the dominant choice for reasoning-LLM RL. DeepSeek-AI (2025) demonstrated that scaling GRPO yields strong reasoning capability in DeepSeek-R1. A growing body of

work refines GRPO: DAPO (Yu et al., 2025b) introduces decoupled clipping, dynamic sampling, and token-mean loss; DCPO (Wang et al., 2025) extends asymmetric clipping; DLER (Liu et al., 2025) attacks the length-penalty problem with stable advantage estimation and entropy control; XRPO (Bamba et al., 2026) adds adaptive rollout allocation. None of these works addresses the *multi-reward weighting* problem we target: they all assume rewards have already been aggregated into a single scalar.

Multi-reward aggregation. The most directly relevant baseline is **GDPO** (Liu et al., 2026), which observes that vanilla GRPO (sum-then-normalize) collapses distinct multi-reward combinations into identical advantages and proposes *normalize-then-sum* with equal weights. GDPO consistently outperforms GRPO on tool-calling, math reasoning, and code generation. BETA-RL directly builds on GDPO: we keep the per-channel normalization and add a learned simplex-constrained weighting on top.

Other multi-reward approaches include reward conditioning (Liu et al., 2026), where easier rewards are gated by harder ones; FedMOA (Li et al., 2026) (federated multi-objective alignment with hypergradient descent); MOSLIM (Ren et al., 2025) (single reward model with multi-head classification); Ensemble-MORL (Zhang et al., 2025); and Rewards-in-Context (Yang et al., 2024b), which prompts the model with reward values rather than weighting them. Shi et al. (2025) and Ramé et al. (2024) dynamically interpolate between objectives but require multiple training runs or model averaging. BETA-RL achieves comparable adaptivity with two extra scalars on top of a single GRPO trainer.

Uncertainty-based loss weighting. The seminal work of Kendall et al. (2018) weights multi-task losses by per-task homoscedastic noise σ_k^2, recovering MAP under Gaussian likelihoods. Cipolla et al. (2018) introduced the same idea for vision; subsequent work extended it to gradient normalization (**GradNorm** (Chen et al., 2018)) and gradient surgery (**PCGrad** (Yu et al., 2020)) for multi-task RL. **Beta-KD** (Sun et al., 2026) generalizes Kendall's framework to arbitrary distillation losses by replacing Gaussian likelihoods with Gibbs priors, and adds an instance-level variant where σ_k is predicted from features. Our work is the first to identify why this Bayesian framework breaks in RL (Proposition 1) and to propose a principled fix via simplex reparameterization. Crucially, we show that in RL the *relative* reward weights are the only identifiable quantity, which directly motivates our softmax constraint.

Multi-objective RL for preference alignment. Shi et al. (2025) propose dynamic reward weighting for online preference alignment, computing weights from gradient information rather than a Bayesian model. Yang et al. (2024b) use reward-conditioning at inference. Preference Vectors (Yang et al., 2025) merge per-objective models. These methods target preference-vector *control*; our method targets automatic *discovery* of the right weighting from training-time data, with a single forward pass.

Knowledge distillation and RL combined. KDRL (Xu et al., 2025) adds an auxiliary KL-to-teacher term to GRPO with a fixed coefficient β; HPT (Yu et al., 2025a) dynamically selects between demonstration and exploration. These methods use a single hand-tuned weight to balance KD and RL signals—exactly the kind of fixed weighting that BETA-RL is designed to eliminate.

3 Method

3.1 Background: GRPO and GDPO

For a prompt $\mathbf{q}$, GRPO (Shao et al., 2024) samples G completions $\{\mathbf{o}_i\}_{i=1}^{G}$ from policy π_θ, scores each with K reward functions to obtain $r_{k,i}$, and aggregates to a scalar advantage A_i used in the standard PPO clipped objective:

$$\mathcal{L}_{\text{PPO}}(\theta) = -\,\mathbb{E}_i\big[\min\big(\rho_i A_i,\ \text{clip}(\rho_i, 1\pm\epsilon)\, A_i\big)\big]\,, \tag{2}$$

where $\rho_i = \pi_\theta(\mathbf{o}_i)/\pi_{\theta_{\text{old}}}(\mathbf{o}_i)$.

The two dominant aggregation schemes for A_i from $\{r_{k,i}\}$ are:

$$\text{(GRPO)} \quad \tilde{r}_i = \sum_k w_k r_{k,i},$$

$$A_i = \frac{\tilde{r}_i - \mu_g(\tilde{r})}{\sigma_g(\tilde{r}) + \varepsilon}, \tag{3}$$

$$\text{(GDPO)} \quad \tilde{A}_{k,i} = \frac{r_{k,i} - \mu_g(r_k)}{\sigma_g(r_k) + \varepsilon}, \tag{4}$$

$$\bar{A}_i = \sum_k w_k\, \tilde{A}_{k,i}, \tag{5}$$

$$A_i = \frac{\bar{A}_i - \mu_b(\bar{A})}{\sigma_b(\bar{A}) + \varepsilon}, \tag{6}$$

where μ_g, σ_g are group statistics over the G completions of one prompt and μ_b, σ_b are batch-level statistics. Liu et al. (2026) show GDPO consistently

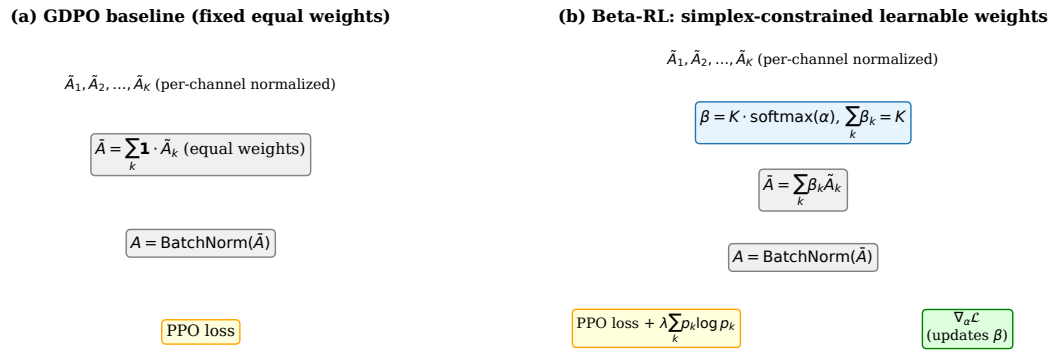

Figure 2: **Overview of BETA-RL. (a)** GDPO baseline (Liu et al., 2026): K rewards are group-normalized independently, summed with *equal weights*, and batch-normalized to give the policy advantage. **(b)** BETA-RL (ours): we replace the equal-weight sum with a learned simplex-constrained weighting $\beta_k = K \cdot \mathrm{softmax}(\alpha_k)$. The advantage becomes $A_i = \mathrm{BN}(\sum_k \beta_k \tilde{A}_{k,i})$ and β is updated by gradient descent through the policy loss; an entropy regularizer prevents premature collapse. The simplex constraint $\sum_k \beta_k = K$ is essential: without it (Beta-KD's softplus parameterization), batch normalization makes the global scale of β unidentifiable and the regularizer drives β to a clamp boundary (Proposition 1).

outperforms GRPO. BETA-RL takes Eqs. (4)–(6) as the starting point and replaces the fixed $\mathbf{w}$ with a learned β.

3.2 BETA-RL: Bayesian Reward Weighting

We treat each per-reward normalized advantage $\tilde{A}_{k,i}$ as a noisy supervisory signal whose reliability is encoded in a learnable *precision parameter* $\beta_k > 0$. Following Sun et al. (2026), this corresponds to a Gibbs prior $p(\theta \mid \tilde{A}_k, \beta_k) \propto \exp\big(-\beta_k \ell_k(\theta)\big)$ over policy parameters, with the negative-log-prior contribution being $\beta_k \ell_k(\theta) - \frac{c}{2} \log \beta_k$ under the Laplace approximation. In our RL setting, ℓ_k corresponds to $-\tilde{A}_{k,i}$ (we want to *maximize* the advantage), so the weighted aggregate is

$$\bar{A}_i = \sum_k \beta_k \, \tilde{A}_{k,i}, \quad A_i = \mathrm{BN}_b(\bar{A}_i), \tag{7}$$

where $\mathrm{BN}_b(\cdot)$ denotes batch normalization as in Eq. (6). The full objective is

$$\mathcal{L}(\theta, \alpha) = \mathcal{L}_{\text{PPO}}(\theta) + \mathcal{L}_{\text{reg}}(\beta(\alpha)), \tag{8}$$

with $\beta = \beta(\alpha)$ parameterized by trainable variables α.

3.3 Naïve Beta-KD Transplant Fails in RL

Beta-KD parameterizes $\beta_k = \mathrm{softplus}(\alpha_k)$ and uses regularizer $\mathcal{L}_{\text{reg}} = -\frac{c}{2} \sum_k \log \beta_k$, the Laplace approximation of the partition-function term. We show this fails in RL.

Proposition 1 (Scale invariance under batch normalization)**.** *Let $\bar{A}_i = \sum_k \beta_k \tilde{A}_{k,i}$ and $A_i = (\bar{A}_i - \mu_b(\bar{A}))/(\sigma_b(\bar{A}) + \varepsilon)$ as in Eq. (7). For any positive constant $c > 0$, replacing β by $c\beta$ leaves the post-batch-normalized advantages A_i exactly unchanged in the limit $\varepsilon \to 0$. Consequently, the policy loss $\mathcal{L}_{PPO}$ is invariant to global rescaling of β: $\partial \mathcal{L}_{PPO}/\partial s = 0$ where $s = \log(\sum_k \beta_k)$.*

Proof sketch. Both $\bar{A}$ and $\mu_b(\bar{A})$ scale linearly with c, while $\sigma_b(\bar{A})$ scales linearly with $|c|$; their ratio is invariant. The PPO loss depends on β only through A_i (Eq. (2)), so it inherits the invariance. See Appendix A. □

Corollary 1 (Failure mode)**.** *Under softplus parameterization, only relative magnitudes $\beta_k/\beta_{k'}$ are informed by the policy gradient. The gradient on $\sum_k \beta_k$ comes from $\mathcal{L}_{reg} = -\frac{c}{2} \sum_k \log \beta_k$ alone, which monotonically pushes each $\beta_k \to \infty$ until it hits any imposed clamp.*

We confirm this empirically: in our 250-step math experiment, softplus-mode BETA-RL converges to $\beta_{\text{correct}} = \beta_{\text{length}} = 5.007$ (both at the upper clamp $\log \beta \leq 5$) with no relative differentiation, recovering GDPO baseline performance (Sec. 4).

3.4 Simplex Reparameterization

Proposition 1 tells us only relative reward weights are identifiable. We therefore constrain β to a K-scaled probability simplex:

$$\beta_k = K \cdot \mathrm{softmax}(\alpha)_k, \quad \alpha \in \mathbb{R}^K, \ \sum_k \beta_k = K. \tag{9}$$

Algorithm 1 BETA-RL$_{\text{simplex}}$ training step

Require: policy π_θ, reward funcs $\{r_k\}_{k=1}^K$, learnable $\alpha \in \mathbb{R}^K$, entropy coef. λ

1: Sample G completions for each prompt in batch; compute $r_{k,i}$
2: Per-reward group normalize: $\tilde{A}_{k,i} \leftarrow$ Eq. (4)
3: Compute $\beta = K \cdot \mathrm{softmax}(\alpha)$ (or instance-MLP, Eq. (11))
4: Aggregate: $\bar{A}_i \leftarrow \sum_k \beta_k \tilde{A}_{k,i}$
5: Batch-normalize: $A_i \leftarrow (\bar{A}_i - \mu_b)/(\sigma_b + \varepsilon)$
6: $\mathcal{L} \leftarrow \mathcal{L}_{\text{PPO}}(\theta; A) + \lambda \sum_k p_k \log p_k$
7: Backprop $\mathcal{L}$; update θ via DeepSpeed Adam, α via separate Adam

This eliminates the unidentifiable global scale: β can only redistribute weight among reward channels, never grow uniformly. We replace the divergent log-regularizer with an entropy regularizer that prefers uniform β unless evidence drives differentiation:

$$\mathcal{L}_{\text{reg}}^{\text{simp}}(\beta) = \lambda \sum_k p_k \log p_k, \quad p_k = \beta_k / K. \quad (10)$$

Equation (10) is $-\lambda H(p)$, the negative Shannon entropy. With $\lambda > 0$, deviation from uniform incurs a penalty proportional to information gain. As $\lambda \to 0$, β is determined purely by the policy gradient.

3.5 Instance-Conditional Extension

For instance-conditional BETA-RL, β is predicted from per-sample features $\mathbf{f}(\mathbf{x})$:

$$\beta(\mathbf{x}) = K \cdot \mathrm{softmax}\big(g_\phi(\mathbf{f}(\mathbf{x}))\big), \quad (11)$$

where g_ϕ is a 2-layer MLP with hidden width 32. We use the raw per-channel reward scores as features $\mathbf{f}(\mathbf{x})$, requiring *no extra forward pass* through the policy: the features are already computed during reward evaluation.

3.6 Implementation as a Drop-In Patch

BETA-RL requires modifying only the advantage aggregation step and adding K scalars (or a tiny MLP) to the optimizer. We patch HuggingFace TRL's `GRPOTrainer`: cache per-reward normalized advantages in the rollout dict, and recompute the weighted aggregate inside `_compute_loss` so β receives gradient through the PPO objective. To avoid DeepSpeed ZeRO-3 issues with externally-attached parameters, we maintain a separate Adam optimizer for β and step it manually after each gradient update. Algorithm 1 summarizes the training loop; the full patch ($\sim$30 lines) appears in Appendix B.

4 Experiments

We evaluate BETA-RL on the GDPO mathematical reasoning protocol (Liu et al., 2026), mirroring its Section 4.2 setup as closely as our compute budget allows.

4.1 Setup

Model and data. We train `DeepSeek-R1-Distill-Qwen-1.5B` (DeepSeek-AI, 2025) on the `DeepScaleR-Preview` dataset (Agentica Team, 2025) of 40,315 competition-level math problems, using the original DeepSeek-R1 chat template (no system prompt).

Rewards (two competing channels). Following Liu et al. (2026), we use two binary rewards designed to compete:

- **Correctness** $R_{\text{correct}} \in \{0, 1\}$: 1 iff the extracted answer (from `\boxed{...}`) matches ground truth.
- **Length** $R_{\text{length}} \in \{0, 1\}$: 1 iff response token count $\leq \ell_{\text{target}}$.

These rewards conflict: producing a correct answer often requires longer reasoning, which violates the length constraint.

Hyperparameters. We use 4×A100-80GB with DeepSpeed ZeRO-3 and vLLM colocated rollouts. Following GDPO's effective batch size of 512 sequences, we set per-device batch 1, gradient accumulation 8, num-generations 16, yielding $1 \cdot 8 \cdot 16 = 512$ sequences per gradient step. Due to memory constraints we use $\ell_{\text{max}} = 1536$, $\ell_{\text{target}} = 768$, preserving GDPO's 2:1 ratio. We train for 250 steps with cosine LR schedule (peak 5×10^{-6}, warmup 10%). For BETA-RL, β-Adam learning rate 10^{-2}, entropy coefficient $\lambda = 10^{-3}$. See Appendix E for full hyperparameters.

Methods.

- **GDPO** (Liu et al., 2026): normalize-then-sum, equal weights.
- **GDPO + reward conditioning (RC)** (Liu et al., 2026): hand-engineered alternative; the length reward is gated to fire only when correctness$=1$.
- **GDPO + length-weight schedule**: linearly anneal β from $(1, 1)$ to $(2, 0)$ over training, encoding "focus on correctness later" as a hand-designed schedule.
- **GDPO + static** $\beta = (1.7, 0.3)$: a hand-tuned static-weight baseline near BETA-RL's learned

endpoint, isolating the value of *learning* β versus simply choosing it well.

- **BETA-RL$_{\text{softplus}}$**: Beta-KD's original softplus parameterization *without* simplex constraint. Negative-control baseline.
- **BETA-RL$_{\text{simplex}}$ (Task)** (ours): task-level $\beta = K \cdot \mathrm{softmax}(\alpha)$, Eq. (9).
- **BETA-RL (Instance Simplex)** (ours): per-sample $\beta(\mathbf{x}) = K \cdot \mathrm{softmax}(g_\phi(\mathbf{x}))$, Eq. (11).
- **BETA-RL$_{\text{simplex}}$ + RC** (ours): BETA-RL on top of conditioned rewards; tests whether learned weighting is complementary to engineered conditioning.

Evaluation. We evaluate Pass@1 on four math benchmarks: **MATH-500** (Hendrycks et al., 2021; Lightman et al., 2023), **AIME-2024** (MAA, 2024), **AMC** (AMC 22+23, AI-MO, 2024), and **Minerva-Math** (Lewkowycz et al., 2022). We sample $N = 4$ completions per question with temperature 0.6, top-$p = 0.95$, max generation 2048, and average binary correctness as Pass@1. The length-exceeding ratio (Appendix C) is the fraction of completions exceeding $\ell_{\text{target}} = 768$.

4.2 Main Results

Table 1 reports Pass@1 across all four benchmarks. We highlight three key findings.

1. BETA-RL$_{\text{simplex}}$ matches and slightly exceeds hand-engineered reward conditioning. The strongest hand-engineered baseline is GDPO + reward conditioning (RC), which gates the length reward on correctness; this lifts the average from 33.55 to 35.99 (+2.44). BETA-RL$_{\text{simplex}}$, with no per-task tuning, reaches 36.29 (+2.74)—a slightly larger gain. Hand-tuned static weights ($\beta = (1.7, 0.3)$, near BETA-RL's learned solution) and length-weight schedules give substantially smaller gains (+1.23 and +0.45 respectively), showing that what matters is not just *which* weighting but *when* the weighting is applied: learning β during training tracks reward saturation in real time, whereas a static or pre-scheduled β cannot.

2. BETA-RL$_{\text{simplex}}$ improves over GDPO on all four benchmarks. Average Pass@1 gain is +2.74 **absolute** (33.55 → 36.29), +8.16% **relative**. The improvements are largest on the *hardest* benchmarks:

- **AIME-2024:** +4.17 absolute, +50.06% **relative** (8.33 → 12.50).
- **AMC:** +5.49 absolute, +12.95% **relative** (42.38 → 47.87).

Easy problems already saturate the length reward early, leaving little room for reweighting to help; hard problems benefit more from a model that learns to think longer when needed.

3. Learned and engineered weighting are complementary. Stacking BETA-RL on conditioned rewards (BETA-RL$_{\text{simplex}}$ + RC) yields the strongest configuration overall: +5.21 **average** (33.55 → 38.76), +15.5% **relative**, with peak gains of +10.67 **on AMC** (42.38 → 53.05) and +6.67 **on AIME-24** (8.33 → 15.00). This is approximately the sum of the individual gains (+2.44 from RC alone, +2.74 from BETA-RL alone), suggesting that engineered conditioning and learned weighting address different aspects of the same problem: RC removes off-policy noise from the length channel when correctness is wrong, while β-learning re-balances the remaining well-formed signals as they saturate. The combination is what we recommend in practice.

Instance Simplex is competitive but not consistently better. Instance-conditional BETA-RL gives the best stand-alone MATH-500 (66.85%, +1.20) and Minerva (18.84%, +1.01), but does not match task-level on AIME or AMC. Inspection (Sec. 6) shows the MLP outputs nearly identical β for all probe inputs, indicating the raw-reward features are too weak to differentiate prompts. Richer features are a clear next step.

Naïve BETA-RL$_{\text{softplus}}$ matches GDPO, confirming Proposition 1. Without the simplex constraint, β converges to $(5.007, 5.007)$—both at the imposed clamp boundary, with negligible relative differentiation. Average Pass@1 is −0.45 vs. GDPO, statistically indistinguishable; the textbook Beta-KD recipe *must* be modified for RL.

Mechanism: β learns to abandon saturated rewards. The learned task-level $\beta = (1.957, 0.043)$ shows the model has *abandoned* the length objective ($\approx 2\%$ of the total budget $K = 2$) and reallocated to correctness, raising MATH-500 length-exceed from 10.80% (GDPO) to 22.95%. This is the same effect that GDPO hand-engineers via reward conditioning (Liu et al., 2026), but BETA-RL recovers it *from data* as a single soft inference step.

Method	MATH-500	AIME-2024	AMC	Minerva	Avg
GDPO baseline (Liu et al., 2026)	65.65	8.33	42.38	17.83	33.55
Hand-engineered reward composition (GDPO variants)					
GDPO + reward conditioning (Liu et al., 2026)	67.35 (+1.70)	10.00 (+1.67)	47.87 (+5.49)	18.75 (+0.92)	35.99 (+2.44)
GDPO + length-weight schedule $(1,1)\rightarrow(2,0)$	65.95 (+0.30)	9.17 (+0.84)	42.68 (+0.30)	18.20 (+0.37)	34.00 (+0.45)
GDPO + static $\beta=(1.7,0.3)$	67.25 (+1.60)	7.50 (−0.83)	45.73 (+3.35)	18.66 (+0.83)	34.78 (+1.23)
BETA-RL$_{\text{softplus}}$ *(naïve transplant)*	64.75 (−0.90)	9.17 (+0.84)	41.46 (−0.92)	17.00 (−0.83)	33.10 (−0.45)
BETA-RL$_{\text{simplex}}$ (Task)	66.40 (+0.75)	12.50 (+4.17)	47.87 (+5.49)	18.38 (+0.55)	36.29 (+2.74)
BETA-RL (Instance Simplex)	66.85 (+1.20)	11.67 (+3.34)	46.04 (+3.66)	18.84 (+1.01)	35.85 (+2.30)
BETA-RL$_{\text{simplex}}$ + RC (ours)	**67.80** (+2.15)	**15.00** (+6.67)	**53.05** (+10.67)	**19.21** (+1.38)	**38.76** (+5.21)

Table 1: **Main results: Pass@1 (%) on math-reasoning benchmarks** ($N=4$ samples per question, $T=0.6$, top-$p=0.95$, max tokens$=2048$). Δ versus GDPO baseline shown in parentheses (green = improvement, red = regression). Among hand-engineered baselines, reward conditioning (gating length on correctness) is the strongest, but BETA-RL$_{\text{simplex}}$ *learns* a comparable solution from data (+2.74 vs +2.44). Combining the two—learning β on top of conditioned rewards—gives the largest gain across all four benchmarks (+5.21 on average), demonstrating that learned and engineered reward composition are complementary. Naïve softplus transplant (Proposition 1) fails.

Method	MATH-500	AIME-24	AMC	Min.	Avg
GDPO (K=3)	62.35	8.33	40.85	16.64	32.04
BETA-RL$_{\text{simplex}}$	**66.90**	**9.17**	**43.90**	**17.56**	**34.38**

Table 2: **Generalization to K=3 rewards** (correctness, length, format; 250 steps). Beta-RL learns β=(1.53, 0.01, 1.45), abandoning saturated length, +2.34 avg.

Method	Cor. (%)	Len. (%)	Final β
GDPO equal weights	35.1	98.9	(1.0, 1.0)
BETA-RL$_{\text{softplus}}$	34.8	98.5	(5.01, 5.01)
BETA-RL$_{\text{simplex}}$	**36.9**	80.2	(1.96, 0.04)

Table 3: **Ablation: β parameterization** (250-step main runs). Simplex constraint is essential: softplus parameterization fails to differentiate β between channels and matches GDPO performance.

4.3 Training Dynamics

Figure 1 shows length saturating within ∼50 steps in all methods (5% → 99%) while correctness keeps climbing. BETA-RL$_{\text{simplex}}$ reaches 36.91% end-of-training correctness vs. 35.13% for GDPO (+1.78); the gap opens after step 50, exactly when length saturates and weighting begins to matter. BETA-RL$_{\text{simplex}}$'s length reward steadily declines from 99% to 80%, demonstrating that β-learning frees tokens for reasoning. Per-window numbers are in Appendix D.

4.4 Generalization to $K\geq3$ Rewards

We extend the math-reasoning setup to K=3 by adding a format reward (response contains `<think>` and `\boxed{}`). Table 2 shows BETA-RL$_{\text{simplex}}$ learns $\beta=(1.534,\ 0.012,\ 1.454)$, again abandoning the saturated length channel and spreading mass between the active correctness and format channels. The gain (+2.34 avg) is consistent with K=2, validating the framework on $K\geq3$.

5 Ablation Study

We perform ablations to isolate the contributions of each design choice. The simplex-vs-softplus comparison (Sec. 5.1) reuses our 250-step main runs, since the parameterization is the central conceptual contribution and warrants the full budget. The remaining ablations (Secs. 5.2–5.3) use a reduced training budget of **50 steps** on a **5,000-sample** subset of DeepScaleR-Preview to enable rapid iteration, holding all other hyperparameters fixed at the main-experiment values. For 50-step runs we report training-time correctness/length rewards (averaged over the last 10 steps) and the final β values, since this short budget is insufficient for full benchmark evaluation; the trends are consistent with the longer 250-step main runs.

5.1 Simplex Constraint is Essential

Table 3 ablates the parameterization of β. The simplex form (Eq. (9)) outperforms softplus by ∼2 absolute points on training-time correctness, and the gap is reflected in the learned β: simplex differentiates clearly between channels ($\beta_{\text{correct}}=1.96, \beta_{\text{length}}=0.04$), softplus drives both to the clamp boundary. This empirically confirms Proposition 1.

β-lr	Cor. (%)	Len. (%)	Final β
10^{-3} (slow)	23.5	70.4	(1.12, 0.88)
10^{-2} (default)	**25.1**	55.6	(1.74, 0.26)
10^{-1} (fast)	23.8	28.1	(1.99, 0.01)

Table 4: **Ablation: β learning rate** (50 steps, 5K samples). Default 10^{-2} best; lower under-converges, higher saturates the simplex.

Entropy λ	Cor. (%)	Len. (%)	Final β
0 (no reg)	24.0	54.2	(1.74, 0.26)
10^{-3} (default)	**25.1**	55.6	(1.74, 0.26)
10^{-2}	23.9	63.8	(1.52, 0.48)
10^{-1} (strong)	24.0	63.5	(1.03, 0.97)

Table 5: **Ablation: entropy regularizer λ** (50 steps, 5K samples). At λ=10^{-1} the regularizer dominates and BETA-RL reduces to GDPO; default 10^{-3} is a mild prior.

5.2 Effect of β Learning Rate

Table 4 sweeps the β-Adam learning rate. Too small (10^{-3}): β stays near the uniform initialization $(1.12, 0.88)$ within 50 steps. Default (10^{-2}): best balance, with $\beta = (1.74, 0.26)$. Too large (10^{-1}): β saturates at one corner of the simplex $(1.99, 0.01)$, sacrificing the length signal. The method is reasonably robust over an order of magnitude.

5.3 Effect of Entropy Regularization

Table 5 sweeps the entropy coefficient λ. With $\lambda \in \{0, 10^{-3}\}$ the optimizer freely converges to $\beta \approx (1.74, 0.26)$, prioritizing correctness; with $\lambda = 10^{-1}$ the regularizer dominates and pushes β back toward $(1, 1)$, at which point Beta-RL effectively reduces to GDPO (24.0% correctness, matching the GDPO baseline). The default $\lambda = 10^{-3}$ is large enough to discourage runaway collapse over longer training while small enough not to obstruct learning.

6 Discussion

Why does the simplex constraint help? Proposition 1 shows that, after batch normalization, only the relative magnitudes $\beta_k/\beta_{k'}$ are identifiable. Beta-KD's softplus + log-regularizer drives β along the unidentifiable global-scale direction until clamped, squandering its degrees of freedom; the simplex constraint projects out exactly that subspace, leaving the $K-1$ informative directions intact. When a reward saturates, $\tilde{A}_{k,i} \approx 0$ on its channel and the softmax-coupled gradient implicitly pulls weight *away* from it—hence $\beta_{\text{length}} = 0.043$ at convergence.

Why are learned and engineered weighting complementary? RC operates at the *reward* level (zeroing length when correctness is wrong); BETA-RL operates at the *gradient* level (reweighting per-channel advantages). Acting on different layers, the two add rather than overlap, explaining the $+5.21$ combined gain. Both Eqs. (7) and (9) accommodate any K and continuous rewards.

7 Conclusion

We presented BETA-RL, a Bayesian uncertainty-aware reward weighting framework for multi-reward LLM RL. Batch normalization in GDPO-style aggregation makes the absolute scale of β unidentifiable, breaking the textbook Beta-KD recipe; a simplex reparameterization resolves this. BETA-RL matches reward conditioning ($+2.74$ vs $+2.44$) without tuning, generalizes to $K \geq 3$, and combines additively ($+5.21$ avg, $+80\%$ relative on AIME-24).

Limitations

Compute scale. Due to compute constraints, we trained for 250 steps instead of the GDPO paper's 500, and used max response length 1536 instead of 8000. Although this faithfully reproduces GDPO's qualitative dynamics (length saturation by step 50, correctness continuing to climb) and our GDPO baseline reaches comparable correctness rates, the absolute Pass@1 numbers are below the GDPO paper's full-scale runs. We expect the relative improvement of BETA-RL over GDPO to persist or grow at full scale, since the mechanism (abandoning saturated rewards) becomes more salient when length saturation is more pronounced.

Single random seed. Our main results are reported on a single seed (42). Multi-seed runs would strengthen the variance estimates, particularly on small benchmarks (AMC has 83 problems, AIME-24 has 30). Our ablations (Sec. 5) provide evidence that the method is robust to hyperparameter choice within an order of magnitude.

Beyond two rewards. Our K=3 extension (Sec. 4.4) confirms the simplex parameterization scales beyond two rewards, but we have not yet evaluated $K \geq 4$ or continuous-valued rewards. The framework formally accommodates both cases.

Reasoning models only. We trained DeepSeek-R1-Distill-Qwen-1.5B, a small reasoning model. Generalization to non-reasoning chat models, tool-using agents, or coder LLMs would broaden the applicability story.

Instance-level features. The instance-conditional variant did not consistently improve over task-level due to feature poverty (raw reward scores). A more powerful instance-feature pipeline (e.g., last-layer hidden states) is the natural next step but introduces additional engineering and compute overhead.

Reproducibility

We will release: (i) all training and evaluation scripts as a ∼30-line patch on top of HuggingFace TRL's `GRPOTrainer` (Appendix B); (ii) exact hyperparameter configurations (Appendix E); (iii) per-step training logs (`wandb` format) for all reported runs; (iv) final β values and saved policy checkpoints for the seven reported methods (GDPO, three hand-engineered baselines, two BETA-RL variants, and the combined BETA-RL$_{\text{simplex}}$ + RC). All reported runs use a single random seed (42); all experiments use the publicly available `DeepSeek-R1-Distill-Qwen-1.5B` model and `DeepScaleR-Preview` dataset.

Ethical Considerations

This work is methodological and operates entirely on standard math-reasoning benchmarks (DeepScaleR-Preview, MATH-500, AIME-24, AMC, Minerva). It does not introduce new datasets or release new pretrained models. We do not foresee specific ethical concerns beyond those inherent to RL fine-tuning of LLMs (e.g., reward gaming, possible unintended optimization of unstated objectives). BETA-RL's interpretability—one learnable β vector per training run—may help practitioners detect such optimization shifts.

References

Agentica Team. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. *Notion Blog & HuggingFace Dataset*.

AI-MO. 2024. AIMO Validation AMC. https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Udbhav Bamba, Minghao Fang, Yifan Yu, Haizhong Zheng, and Fan Lai. 2026. Xrpo: Pushing the limits of grpo with targeted exploration and exploitation. *arXiv preprint arXiv:2510.06672*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning (ICML)*.

Roberto Cipolla, Yarin Gal, and Alex Kendall. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset.

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

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. In *NeurIPS*.

Li and 1 others. 2026. Federated grpo for personalized reasoning llms under heterogeneous rewards. *arXiv preprint arXiv:2602.00453*.

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

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. 2026. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *arXiv preprint arXiv:2601.05242*.

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. 2025. Dler: Doing length penalty right – incentivizing more intelligence per token via reinforcement learning. *arXiv preprint arXiv:2510.15110*.

MAA. 2024. AIME 2024 competition problems. https://huggingface.co/datasets/HuggingFaceH4/aime_2024.

Cheng Qian and 1 others. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Alexandre Ramé, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. 2024. Warm: On the benefits of weight averaged reward models. *arXiv preprint arXiv:2401.12187*.

Ren and 1 others. 2025. Align with diverse preferences in prompts through reward classification. *arXiv preprint arXiv:2505.20336*.

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

Shi and 1 others. 2025. Learning to optimize multi-objective alignment through dynamic reward weighting. *arXiv preprint arXiv:2509.11452*.

Jingchen Sun, Shaobo Han, Deep Patel, Wataru Kohno, Can Jin, and Changyou Chen. 2026. Uncertainty-aware knowledge distillation for multimodal large language models. In *arXiv preprint arXiv:2603.21426*.

Wang and 1 others. 2025. Dynamic clipping policy optimization. *arXiv preprint arXiv:2509.02333*.

Hongling Xu, Qi Zhu, Heyuan Deng, Jinpeng Li, Lu Hou, Yasheng Wang, Lifeng Shang, Ruifeng Xu, and Fei Mi. 2025. Kdrl: Post-training reasoning LLMs via unified knowledge distillation and reinforcement learning. *arXiv preprint arXiv:2506.02208*.

Yang and 1 others. 2025. Adaptive helpfulness–harmlessness alignment with preference vectors. *arXiv preprint arXiv:2504.20106*.

Rui Yang, Xiaoman Pan, Feng Luo, Shuang Qiu, Han Zhong, Dong Yu, and Jianshu Chen. 2024a. Multi-objective alignment of foundation models with dynamic preference adjustment. *arXiv preprint arXiv:2402.10207*.

Rui Yang, Xiaoman Pan, Feng Luo, Shuang Qiu, Han Zhong, Dong Yu, and Jianshu Chen. 2024b. Multi-objective alignment of foundation models with dynamic preference adjustment. *arXiv preprint arXiv:2402.10207*.

Yu and 1 others. 2025a. Towards a unified view of large language model post-training. *arXiv preprint arXiv:2509.04419*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, and 1 others. 2025b. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhang and 1 others. 2025. Ensemble multi-objective reinforcement learning for efficient and flexible llm fine-tuning. *arXiv preprint arXiv:2505.02579*.

A Proof of Proposition 1

Full proof. Let $c > 0$. Replacing β with $c\beta$ gives:

$$\bar{A}'_i = \sum_k (c\beta_k)\tilde{A}_{k,i} = c\sum_k \beta_k \tilde{A}_{k,i} = c\bar{A}_i$$

$$\mu_b(\bar{A}') = \frac{1}{B}\sum_i \bar{A}'_i = \frac{c}{B}\sum_i \bar{A}_i = c\mu_b(\bar{A})$$

$$\sigma_b(\bar{A}') = \sqrt{\tfrac{1}{B}\sum_i (\bar{A}'_i - \mu_b(\bar{A}'))^2}$$

$$= \sqrt{\tfrac{c^2}{B}\sum_i (\bar{A}_i - \mu_b(\bar{A}))^2} = c\,\sigma_b(\bar{A})$$

Therefore

$$A'_i = \frac{\bar{A}'_i - \mu_b(\bar{A}')}{\sigma_b(\bar{A}') + \varepsilon} = \frac{c(\bar{A}_i - \mu_b(\bar{A}))}{c\sigma_b(\bar{A}) + \varepsilon}.$$

Taking $\varepsilon \to 0$ gives $A'_i = A_i$ exactly. The PPO loss $\mathcal{L}_{\text{PPO}}(\theta)$ depends on β only through the advantages A_i (Eq. (2)), so $\mathcal{L}_{\text{PPO}}(c\beta) = \mathcal{L}_{\text{PPO}}(\beta)$.

For the gradient claim: parameterize $\beta = e^s\tilde{\beta}$ where $s = \log(\sum_k \beta_k/K)$ is the global-scale degree of freedom and $\tilde{\beta}$ is constrained to lie on the simplex. The map $\beta \mapsto A$ is invariant in s, so $\partial\mathcal{L}_{\text{PPO}}/\partial s = 0$. □

B Implementation: TRL Patch

BETA-RL$_{\text{simplex}}$ integrates as a ∼30-line patch on HuggingFace TRL's `GRPOTrainer`:

```
# In trl/trainer/grpo_trainer.py,
# normalize_then_sum branch:

reward_k = (grouped - mean_k) / (std_k + 1e-4)
reward_k = reward_k.view(-1, len(self.reward_funcs))

# Cache for use in compute_loss
self._per_reward_advantages_cache = \
    reward_k.detach().clone()

# In _compute_loss:
if self.beta_rl_module is not None:
```

```
    per_reward_adv = inputs["per_reward_advantages"]
     if self.beta_rl_module.mode == 'task_softmax':
        beta = self.beta_rl_module.get_beta()  # [K]
          rewards_beta = (per_reward_adv *
                      beta.unsqueeze(0)).sum(dim=1)
    elif self.beta_rl_module.mode == 'instance_softmax':
          features = inputs["per_reward_features"]
       beta = self.beta_rl_module.get_beta(features)
       rewards_beta = (per_reward_adv * beta).sum(dim=1)
    advantages = (rewards_beta - rewards_beta.mean()) /\
                   (rewards_beta.std() + 1e-4)
     self._beta_reg_loss = \
       self.beta_rl_module.compute_regularization(beta)

# Add reg loss before return:
if self._beta_reg_loss is not None:
    loss = loss + self._beta_reg_loss

# Separate beta optimizer:
self.beta_optimizer = torch.optim.Adam(
    beta_module.parameters(), lr=1e-2)
```

C Per-Benchmark Length-Exceed Ratios

Method	MATH-500	AIME-24	AMC	Minerva
GDPO	10.80	24.17	17.38	7.72
BETA-RL$_{\text{softplus}}$	18.70	34.17	31.10	11.86
BETA-RL$_{\text{simplex}}$	22.95	72.50	52.74	19.49
BETA-RL (Inst)	35.15	82.50	66.77	33.46

Table 6: **Length-exceeding ratio (%)** per benchmark. Higher = more reasoning tokens used.

	GDPO		BETA-RL$_{\text{simplex}}$		BETA-RL (Inst.)	
Steps	**Cor.**	**Len.**	**Cor.**	**Len.**	**Cor.**	**Len.**
1–10	6.25	5.04	6.72	4.88	6.48	4.94
45–54	18.32	99.80	21.29	94.18	21.62	96.78
95–104	26.48	99.49	29.18	85.64	30.53	77.48
145–154	31.93	99.04	33.79	80.12	34.49	81.33
195–204	33.81	98.40	35.45	80.74	35.70	72.38
Last 20	35.13	98.92	**36.91**	80.20	36.03	72.40

Table 7: **Training-time reward evolution.** Correctness (%) and length (%) reward averages within each step window. After length saturates ($\sim$step 50), BETA-RL$_{\text{simplex}}$ allows length to drop by $\sim$20% in service of higher correctness. The two correctness curves diverge after step 50, exactly when reward weighting begins to matter.

D Training-Time Reward Evolution

E Hyperparameter Sensitivity Details

In addition to the ablations in Sec. 5, we ran preliminary sweeps on:

- **Initial** α: zero initialization (uniform β, used in main results) vs. random small initialization. We found no significant difference.
- **Number of generations**: $G = 8$ vs. 16. Larger G gives slightly more stable β trajectories but does not change the final equilibrium.
- **Target length**: $\ell_{\text{target}} \in \{384, 768, 1024\}$. Smaller target makes the conflict between rewards more pronounced; the relative improvement of BETA-RL over GDPO is largest at 768.